Grokking vs. Learning: Same features, different encodings

To cite this article before publication: Dmitry Manning-Coe *et al* 2026 *Mach. Learn.: Sci. Technol.* in press https://doi.org/10.1088/2632-2153/ae67d0

Grokking vs. Learning: Same features, different encodings

Dmitry Manning-Coe[1], Jacopo Gliozzi[1], Alexander G. Stapleton*[2], Edward Hirst[2], Giuseppe De Tomasi[1,3], Barry Bradlyn[1], and David Berman[2]

[1]Anthony J. Leggett Institute for Condensed Matter Theory, University of Illinois Urbana-Champaign, Urbana, USA
[2]Centre for Theoretical Physics, Queen Mary University of London, UK
[3]CeFEMA-LaPMET, Departamento de Física, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

Keywords: Information Theory, Grokking

Abstract

Grokking typically achieves similar losses to ordinary, 'steady', learning. This work asks whether these different learning paths lead to fundamental differences in the learned models. To do so, we compare the features, compressibility, and learning dynamics of models trained via each path in two controlled toy tasks. We find that grokked and steadily trained models learn the same features, but there can be large differences in the efficiency with which these features are encoded. In particular, we find a novel 'compressive regime' of steady training in which there emerges a linear trade-off between model loss and compressibility, which is absent in grokking. In this regime, one can realise compression factors of 25x in the model obtained by steady learning, and 5x in the model achieved by grokking. Model features and compressibility are then tracked through training. We show that model development in grokking is task-dependent, and that peak compressibility is achieved immediately after the grokking plateau. Finally, novel information-geometric measures are introduced which demonstrate that models undergoing grokking follow a straight path in information space.

Contents

*Corresponding author: a.g.stapleton@qmul.ac.uk

1 Introduction

Grokking [1] is a mode of training in which generalisation emerges suddenly after a long period of overfitting. This paper empirically addresses the following questions: are there fundamental differences between models learned by grokking and through ordinary 'steady' training, and is grokking practically useful? Since grokking happens after a large number of epochs, it is expensive. Whilst some works have reported that grokking may lead to models which exhibit better generalization performance [1], others have shown the contrary [2]. As such, the advantages of grokking appear to be heavily dependent on the training objective, dataset, and learning scheme.

Grokking has been observed in a variety of tasks [3], and several mechanisms have been identified as its possible origin [3–8]. For modular addition, the workhorse of grokking studies, an exactly solvable model of grokking has been proposed [9], and network interpretability measures can track how overfitting gives way to generalisation [10].

All of these works contrast the generalising, post-grokking model to the memorizing, pre-grokking model of the same training run. This leaves it unclear whether grokking is simply a slower form of ordinary learning, or whether there are more fundamental differences between these two learning paths. Explanations of grokking typically revolve around two points. First, that there are multiple representations that can solve the task, and second, that the model generalises as a result of compressing to a more efficient representation. This raises two immediate questions: do grokking and conventional learning lead to different representations, and are there differences in compression between grokking and learning?

This work therefore compares the features learned and the compressibility of models trained via grokking and conventional learning. To make this comparison, two simple toy tasks with known interpretable features are chosen. In this context, toy models are deliberately small setups with controlled optimiser, dataset and architecture. In this work, the toy models have interpretable internal representations and are sufficiently small that hyperparameter sweeps are analytically tractable.

The first task is to classify snapshots of the two-dimensional Ising model [11, 12], which can be interpreted in terms of physical variables - the energy and the magnetization of a snapshot [13, 14]. The second task is modular addition. Networks trained on modular addition are well known to learn a Fourier representation of the problem, and previous work has explicitly quantified how sharply the model is localized in this basis [9, 10, 15]. These tasks are used to isolate mechanisms rather than approximate state-of-the-art models.

After comparing the terminal models, we ask whether differences in model development through training can be identified. To do so, we define summary measures of model development and track their evolution. These measures are task-dependent and must be constructed by hand. Important features of model development are then shown to appear in the *information geometry* of the model, which can be defined for a general task. Using the recently developed framework of Bayesian renormalization, [16–19], we introduce novel measures based on the Fisher Information Metric (FIM) to study model development in the example tasks.

The main conclusions are:

1. In both tasks, the features learned via grokking and steady learning are the same.

2. The encodings, however, are different. In the modular addition task, we discover a 'compressive regime' of training. In this regime, tuning the weight scale at initialization results in a linear trade-off between the *train* loss of a model and its compressibility.

3. The modular addition and Ising tasks have different model development trajectories. In modular addition, consistent with previous literature, the interpretability tools used here allow 'progress measures' to be defined in the grokking plateau. In the Ising task, however, no sign of model development is found before generalisation. This raises the question of whether new progress measures can be defined that would indicate a coming jump in capability before generalisation.

4. In both tasks, the grokking trajectory follows an approximately straight line in model space, as defined by the Fisher Information Metric.

5. In agreement with the literature, we find all models and architectures which have exhibited grokking may be modified to undergo steady learning [10] and still exhibit good training performance.

The first two points concern the terminal model after training and are addressed in sections 3.1 and 3.2. The second two points concern the dynamics during training and are addressed in section 3.3. The task setup is introduced in section 2, and the grokking protocol is outlined in section 2.3.

2 Experimental setup

This section defines the two benchmark tasks used throughout the paper and summarizes the corresponding datasets and architectures. One physics-inspired image-classification problem (Ising phase recognition) and one algorithmic task (modular addition) are used to compare grokking and steady learning across qualitatively different settings, while keeping all model architectures and hyperparameters other than initialisation scale w_0 fixed.

2.1 Ising Task

Our first task is classifying the phases of the two-dimensional Ising model, perhaps the simplest physical system that exhibits a phase transition [11, 12]. On each vertex of a two-dimensional lattice, the model hosts a vector called a 'spin' that can either point up or down. This system has two distinct phases: an 'ordered' phase where the spins align with each other, and a 'disordered' phase where the spins are randomly oriented.

The two phases are separated by a transition that occurs at a critical temperature, T_c. For $T < T_c$, the system is in an ordered phase, while for $T > T_c$, it is in a disordered phase. In our model, we consider a square grid where each site i contains a spin of value $\sigma_i = \pm 1$, corresponding to up and down. We call an assignment of $\sigma_i = \pm 1$ to every site i in the grid a 'snapshot'. A snapshot is hence just a binary image, and in fig. 1 we show representative snapshots for the different phases.

Snapshots are characterized by two physical quantities: energy and magnetization. The energy is defined as a sum over nearest-neighbour pairs of spins,

$$E = -\sum_{\langle i,j \rangle} \sigma_i \sigma_j, \tag{1}$$

where $\langle i, j \rangle$ are adjacent sites and periodic boundary conditions are used. The energy determines the probability distribution of different snapshot configurations (see appendix B for more details). Based on Eq. (1), neighbouring spins that are misaligned 'cost' positive energy. Disordered snapshots have many misaligned spins, and therefore a higher energy than ordered snapshots. The magnetization of a snapshot is the sum of all of its spins:

$$M = \sum_i \sigma_i \tag{2}$$

In the disordered phase, opposite spins cancel in the sum and M is small compared to the number of spins. In the ordered phase, spins align with each other and $|M|$ is large. Viewing a snapshot as an image, the magnetization quantifies its brightness and the energy captures the number of light to dark interfaces. Classifying snapshots is therefore analogous to the problem of image recognition.

In the first task, we train a Convolutional Neural Network (CNN) to classify a given snapshot of the Ising model as 'ordered' or 'disordered'. Concretely, the input data $\mathbf{x}$ is a 16×16 square grid of ± 1 values, and the labels are $y = 0, 1$ corresponding to whether the phase is ordered or disordered. For all Ising classification tasks in the main text, two convolutional layers and one fully connected layer are used, with ReLU activations and cross entropy loss optimized by Adam [20]. The training data are 300 snapshots equally divided between disordered and ordered configurations. The snapshots are generated by running a local-update Monte Carlo simulation at different temperatures surrounding the critical point between the two phases (see fig. 1).

In an infinite system at equilibrium, the magnetization of a snapshot uniquely characterizes its phase. However, because the snapshots are of a small system that may not be fully equilibrated due to the local update Monte Carlo simulations, the classification problem is more difficult than simply counting how many spins have a positive sign. In particular, ordered snapshots with an anomalously low magnetization can be misclassified if their energy is not taken into account.

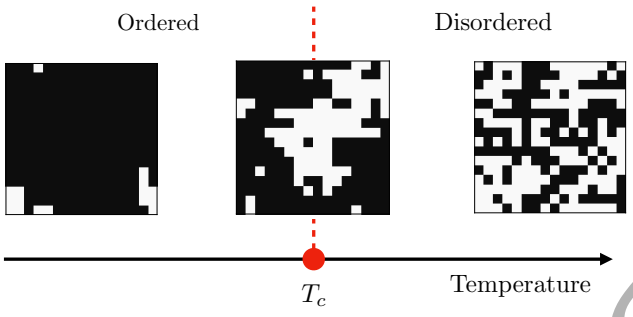


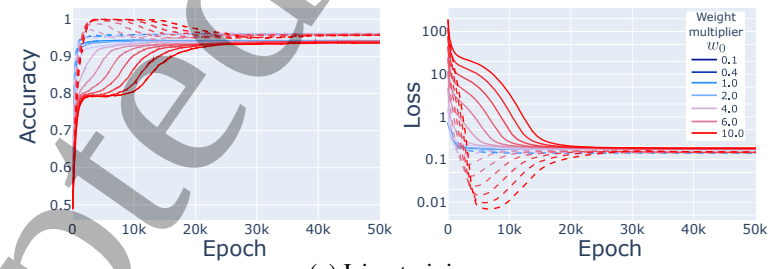
Figure 1: Snapshots of the 16 x 16 Ising lattice, where white and black squares indicate ± 1 spin assignments. As temperature increases, snapshots become more disordered, passing through a phase transition at temperature T_c.

2.2 Modular Addition Task

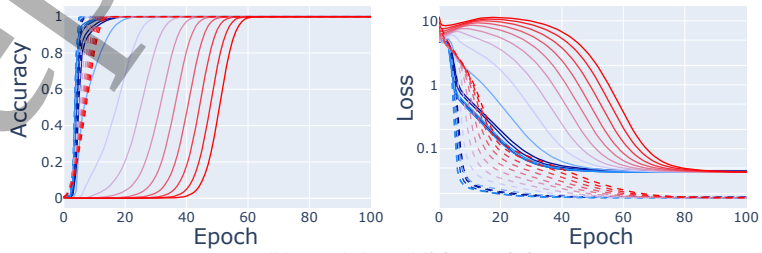
The second task is modular addition. Modular arithmetic models are trained to solve modular equations of the form $c = (a + b)\ \%\ P$. The input data is the set of P^2 two-hot encoded vectors corresponding to the pair a and b, i.e. the length $2P$ vectors $\mathbf{x}_{a,b} = (0_1, ..., 1_a, ..., 0_P, 0_{P+1}, ..., 1_{P+b}, ..., 0_{2P})$, and the output vectors are the one-hot encoded length P vectors corresponding to c, $\mathbf{y_c} = (0_1, ..., 1_c, ..., 0_P)$. A fully connected network with a single hidden layer and ReLU activation is trained, minimizing the cross-entropy loss using the Adam optimizer for $P = 113$. The training data is a fixed fraction (70% in the main text) of randomly chosen samples from the set of all possible pairs. The full details of the model and data for both tasks is in table 1.

2.3 Grokking and Learning

We first show that we can tune between grokking and conventional learning regimes while maintaining the same training data, architecture, and regularization. This allows us to generate comparable models in each regime which differ in whether they were trained via grokking or steady learning. In subsequent sections, we will compare the features, compression, and development of these models.



(a) Ising training curves



(b) Modular addition training curves

Figure 2: Ising and modular addition training curves, for varying weight multipliers at initialization (larger values are redder). The dashed lines indicate training accuracies/losses, and the solid lines test accuracies/losses.

A pragmatic definition of the *grokking time* is chosen, namely:

Definition 1 (Grokking time)**.** *The grokking time is the number of training epochs between when the model is within 5% of its maximum test accuracy, t_{test}, and when the model is within 5% of its maximum train accuracy, t_{train}:*

$$t_{grok} \equiv t_{test} - t_{train} \tag{3}$$

from which we define grokking as:

Definition 2 (Grokking)**.** *A training run is said to grok if $t_{grok} > t_{train}$. Otherwise, we say that the model 'steadily learns'.*

Following [3], we induce grokking by simply multiplying the initialization weights by a scale factor w_0, favouring overfitting during training. The resulting training curves for the Ising task are shown in fig. 2(a) and for the modular addition task in fig. 2(b). The curves are averaged over five and ten seeds (random initialization), respectively, and we summarize the distribution over individual seeds in appendix A. Increasing the weight initialization scale w_0 smoothly interpolates between the grokking and steady learning regimes. According to our definition, grokking sets approximately at $w_0 = 3$ for the Ising task, and $w_0 = 0.4$ for the modular addition task.

3 Results

A natural question is whether grokking is simply a slower route to the same endpoint as steady learning, or whether it produces qualitatively different models. If the endpoint features are similar, an immediate follow-up question is whether those features are encoded with different efficiency. Finally, even when terminal models appear similar, their training trajectories may still differ. Accordingly, we study three complementary aspects: learned features, compressibility, and dynamical development through training.

This section reports three empirical results. First, models trained via grokking and steady learning converge to the same dominant interpretable features (section 3.1). Second, the efficiency with which these features are encoded can differ substantially, including a compressive steady-learning regime in modular addition (section 3.2). Third, training dynamics through the grokking plateau are task-dependent, while information-geometric trajectory measures reveal additional structure in both tasks (section 3.3).

3.1 Shared Feature Learning Across Grokking and Steady Learning

Comparing learned features is a natural first test when evaluating whether two training paths are genuinely different or merely differ in optimization time. Endpoint accuracy alone cannot distinguish between alternative internal solutions, because different representations can yield similar performance. If grokking and steady learning solve the task through distinct mechanisms, those differences should appear directly in feature-level probes of the trained networks.

3.1.1 Ising Features: Energy and Magnetization

Models reached via grokking and 'steady' learning are found to learn the same dataset features. Previous studies have established that the key features learned by models trained on the Ising task are the energy (Eq. (1)) and the magnetization (Eq. (2)) [13, 14]. In these earlier works, it was shown that models (trained via the conventional steady learning trajectory) learned a mix of both features. To make a cleaner comparison between the terminal models from both regimes, it is helpful to have a single dominant learned feature. We thus deliberately design our setup to favour learning the energy. To achieve this, we include training snapshots that are in the ordered phase but have anomalously low magnetization. These snapshots are correctly classified by the energy but misclassified by the magnetization, and often appear as metastable states in local-update Monte Carlo simulations of the Ising model [21]. Moreover, we use a more powerful network, including a CNN component to detect the interfaces associated with calculating energy.

To understand which of the three features the model learns, we correlate the pre-activations for all neurons in the networks with the energy, the magnetization, and the absolute value of the magnetization. In fig. 3, we

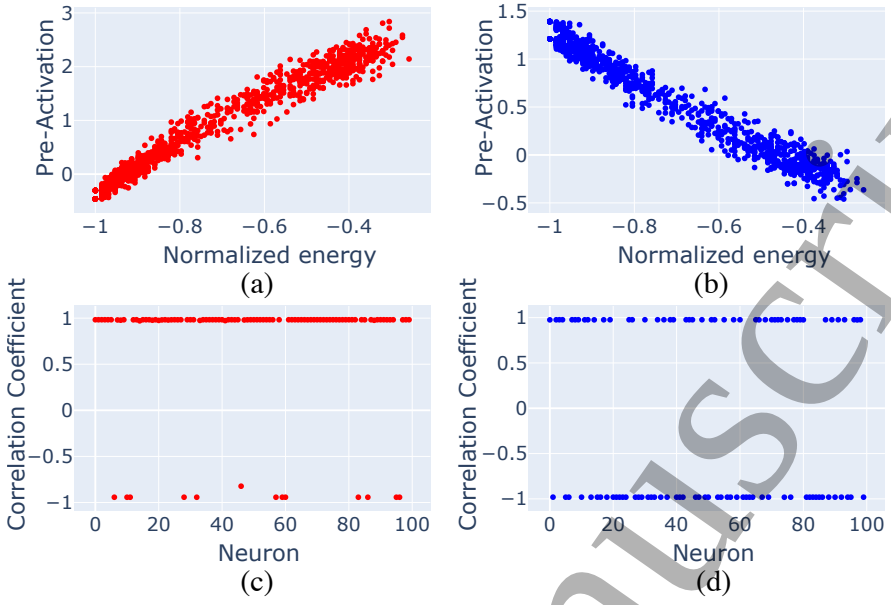


Figure 3: In the Ising task, the model learns the energy for both the grokking (a,c) and learning (b,d) regimes. In the top row, we show a scatter plot of the energy of input snapshots versus the resulting neuron pre-activations in the final layer. Data is shown for only most activated neuron. In the second row, we show that this high degree of correlation is typical of all neurons in the final layer.

compare the result for models reached via grokking and steady learning trajectories. We find that, in both grokking and steady learning, *every* neuron in the final layer is either perfectly correlated or anti-correlated with the energy of the input. In appendix C, we show how this develops in each layer of the model. For every seed, in both learning and grokking, the energy has the highest correlation of the three measures with the majority of neuron activations in the second CNN and in the fully connected layer. We hence conclude that both grokking and steady learning models classify snapshots by learning the same feature - the energy.

3.1.2 Modular Addition Features: Fourier Coefficients

To better understand the task dependence of our comparison between grokking and learning, we also study the modular addition task. A number of studies have convincingly shown that models trained on the modular addition task typically learn a Fourier representation of the problem [1, 9, 10]. For a two layer MLP with square activation, [9] showed that an ansatz of Fourier modes for the model weights solves the modular addition problem. Moreover, the inverse participation ratio (IPR) can be used as a measure of how well-localized the weights of the model are in the Fourier basis. This provides a quantitative comparison of the extent to which grokked and learned models learn the same features.

The distribution of neuron post-activations in the fully connected layer is shown in fig. 4(a-b). We see that each neuron learns the sine and cosine components of a single frequency. To quantify the localization to a given frequency in a given neuron k, we first Fourier transform the input (output) weight matrices in the embedding (unembedding) dimension, and then use Gromov's Inverse Participation Ratio measure [9] to quantify the localization to a given frequency:

$$IPR(k) = \sum_{\nu=1}^{D} \frac{|\tilde{W}_{\nu k}|^4}{(\sum_\nu \tilde{W}_{\nu k}^2)^2}, \tag{4}$$

where ν is the index of the embedding (unembedding) dimension of the input (output) matrices to the hidden layer, and $D = 2P$ ($D = P$) is its size. The results are shown in fig. 4(c-d). First, we notice that the average IPR is around $1/2$, consistent with each neuron predominantly processing the sine and cosine components of a given frequency. Second, we see comparable mean IPRs between the grokking and the steady learning

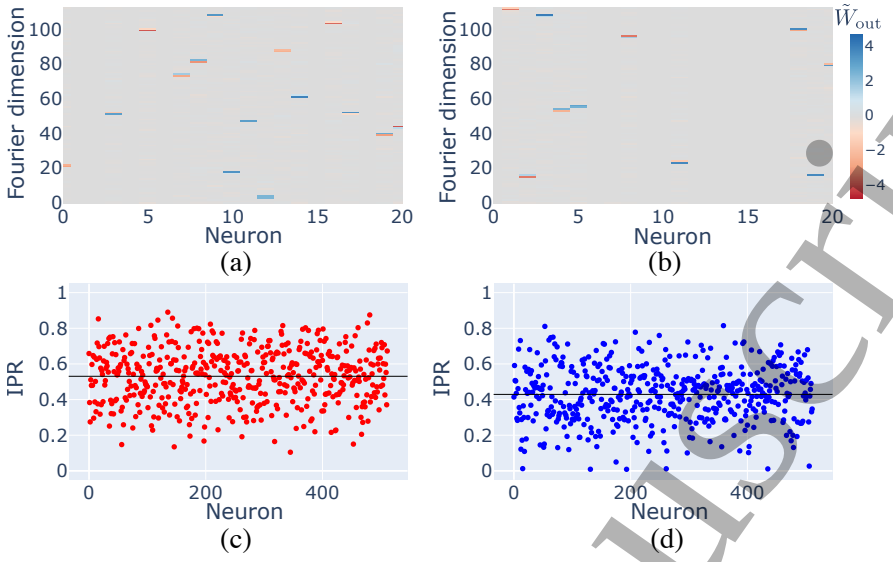


Figure 4: In the modular addition task, the model learns the Fourier modes of the input data. To see this, we Fourier transform the unembedding matrix and plot its coefficients, which correspond to different Fourier frequencies. The heatmaps for (a) grokking and (b) learning suggest that each neuron predominantly activates on the sine and cosine of a single frequency. This is confirmed by calculating the IPR for all neurons, shown in (c) for grokking and (d) for learning.

cases; 0.43 and 0.53, respectively. We therefore conclude that, as in the Ising task, grokked and steadily learned models learn the same dataset features.

3.2 Encoding Efficiency in Different Training Regimes

Given comparable endpoint features, compressibility provides a direct and operational proxy for how efficiently those features are encoded in parameter space. This is a natural next diagnostic: once representational content is similar, differences in parameter usage become the most immediate way to distinguish between terminal models. Magnitude pruning is therefore an appropriate measurement, because it directly quantifies how much of the trained network can be removed before performance degrades.

Although both grokking and steady learning produce similar learned features, substantial differences can nevertheless appear in encoding efficiency. This is quantified via magnitude pruning of terminal models in both tasks. The primary result in this section is the emergence, in modular addition, of an approximately linear trade-off between end-model *training* loss and compressibility (fig. 5). This parameter range is referred to as the 'compressive regime'. It provides a particularly clear example of the broader observation that models with similar learned features can differ markedly in compressibility.

We start by introducing our pruning scheme. In the main text, we use a global magnitude pruning scheme. In the appendix, we show the results of pruning each layer in the network individually. To account for differences in weight scales and layer architecture, we slightly modify naive global pruning. First, we rank all weights within each layer separately by their absolute values. Given a pruning fraction p, we set the smallest p fraction of the weights in each layer to zero, and then evaluate the model accuracy on the test set. To quantify model compressibility, we first integrate the accuracy $a(p)$ at each pruning fraction p with respect to p - i.e. we find the area bounded by the pruning curve and the x-axis in fig. 5(a-b). We then define the compressibility c as:

$$a \equiv \int a(p)dp, \;\; c \equiv \frac{1}{1-a}. \tag{5}$$

This is analogous to finding the size of the pruned model at a fixed accuracy, but averaged over the entire range of pruning.

We show the results of pruning across weight multipliers for modular addition in fig. 5(a). In the steady learning regime $w_0 \leq 0.4$, model compressibility improves systematically with reducing weight multiplier. In

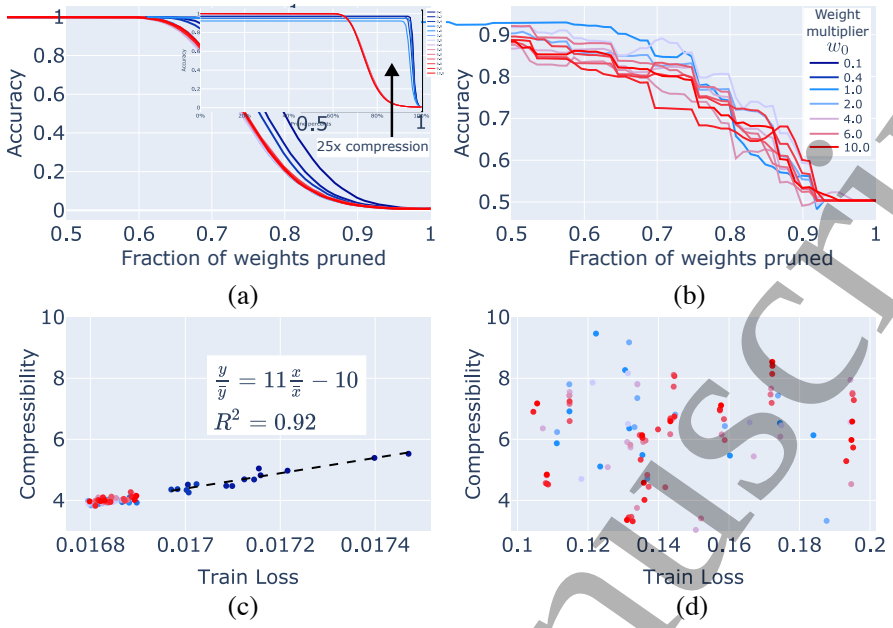


Figure 5: Pruning curves for (a) modular addition and (b) Ising. In the modular addition task, there is a linear trade-off between the end-loss and the compressibility c, shown in (c), while no such relationship for the Ising task, shown in (d). Data in the top row is averaged over seeds, while the bottom row shows all seeds.

fig. 5(c), we see that once the model has transitioned into this parameter regime there emerges a striking linear trade-off between encoding efficiency and end model training loss. In appendix D, we provide data for the relationship between compressibility and a range of other network measures. In particular, we emphasize that there is no correlation between compressibility and the *test* loss. Surprisingly, we also find no relationship between compressibility and model degeneracy, as measured by the local learning coefficient [22–24]. Although the more compressible models are more efficient since they can be specified with fewer parameters, improvements in speed are heavily dependent on architecture; it is not true *a priori* that, in practical applications, a more compressible model exhibits better inference performance.

In the grokking regime of modular addition, we see that there is no significant dependence of model compressibility on w_0 - the red curves are on top of each other. Hence, although models trained in both grokking and steady learning learn the Fourier components of the input data, models trained in the compressive regime of steady learning are significantly more compressible. We emphasize that the existence of a compressive regime in steady learning is not a generic feature, but is parameter-dependent. In appendix D, we show that this regime can be eliminated by increasing the batch size from 64 as used in the main text to 200. The compressive regime is thus a novel parameter regime of steady learning, rather than a generic feature. To our knowledge, an analogous compressive regime has not been observed in larger models [25]. In this sense, the initial weight scale is a critical control parameter in our setup, tuning both whether grokking occurs and, in the steady-learning range, the effective complexity/compressibility of the final model. At the same time, the critical range is not universal and depends on other hyperparameters, as we show explicitly for batch size and weight decay in the appendices. In other words, the parameter sensitivity clarifies the role of initialization scale in practice. In our experiments, w_0 can be used as a knob to move along a complexity–performance trade-off, but only after task-specific calibration with other training hyperparameters. We therefore view weight scale as a useful control variable, as opposed to a universal recipe.

In the inset of fig. 5(a), we show that we can achieve extremely high compressions for a higher weight decay (3×10^{-4} vs 3×10^{-5}). Pairing this with a low weight multiplier, we are able to *cut 95% of the weights* in the model for a 5% drop in accuracy, a compressibility 25x that of the original model, and 5x the highest compression achieved at the lower weight multiplier used in the rest of the main text. In appendix D.1, we explore this regime. We show that in this extremely compressed regime, the mean IPR, $I\tilde{P}R$, falls by close

to an order of magnitude, indicating that the model's Fourier representation is breaking down. We also note that the model is itself close to breakdown - models with a 30% larger weight decay no longer learn at all.

One possible lens through which to interpret these findings is compressibility as a proxy for robustness. Highly compressible models, by construction, admit substantial parameter perturbations without large degradation in performance, suggesting that their learned representations are redundantly encoded and stable to structural noise. This robustness may reflect an underlying simplicity or regularity in the function the model has learned, consistent with classical notions linking flat minima and generalisation. However, our results complicate this picture: despite the intuitive connection between compressibility and robustness, we observe no direct correlation with test loss, indicating that robustness to parameter removal is not, on its own, sufficient to guarantee improved generalisation. In the context of grokking, one speculative explanation is that grokked models, while not maximally compressible, may instead organise their representations in a more globally coherent or algorithmic fashion, enabling better extrapolation even in the absence of redundancy. By contrast, models in the compressive regime of steady learning may achieve robustness through redundancy rather than abstraction, leading to high compressibility but not necessarily superior generalisation. This suggests that there may be multiple, qualitatively distinct routes to robustness—only some of which align with improved out-of-distribution performance.

Given this, we conjecture that there are at least two qualitatively different routes to robustness and eventual generalisation: (i) a redundancy-driven route, in which features are encoded with high parameter redundancy and hence high pruneability/weight-noise stability (high compressibility), and (ii) an abstraction-driven route, in which representations become more globally coherent/algorithmic but not necessarily highly redundant (lower compressibility).

Finally, we show the model compressibility for the Ising task in fig. 5(b,d). Here we see no clear relationship between compressibility and weight multiplier, and therefore no compressive phase. Furthermore, there is no meaningful trade-off between model compressibility and the end loss. This reinforces the view that the compressive phase found in the modular addition task is a distinct regime of training dynamics.

3.3 Task Dependence of Training Dynamics

Endpoint comparisons alone cannot determine whether grokking and steady learning follow the same developmental route during training. Two runs may finish with similar accuracy and features while still traversing qualitatively different regions of model space. It is therefore natural to track feature-formation and information-geometric trajectory measures over time, as these time-resolved diagnostics can reveal mechanistic differences that are hidden by similar final performance.

Using the feature and compressibility measures defined above, development over training can be compared across tasks and regimes, we present two results. First, behavior in the pre-grokking plateau differs between Ising and modular addition tasks. In modular addition, consistent with prior 'progress measure' analyses [9, 10], representation quality continues to improve through the plateau despite flat accuracy. In Ising, by contrast, analogous feature development is not observed before the transition. Second, FIM-inspired trajectory diagnostics provide complementary evidence for these differences in model development.

3.3.1 Feature Development

To measure the development of model features, we first define layer-wide summary averages for our interpretability measures. For the Ising model, we define a pre-activation weighted correlation coefficient $\tilde{r}$ for a layer L as the sum of the (absolute value of) the correlation coefficients for each neuron k in that layer, weighted by that neuron's share of the absolute pre-activation, w_k

$$w_k \equiv \frac{\sum_{i\in\mathcal{D}} |z_k^i|}{\sum_{i\in\mathcal{D}}\sum_{k\in L} |z_k^i|}, \quad \tilde{r} \equiv \frac{1}{|L|}\sum_{k\in L} w_k |r_k|, \tag{6}$$

where $z_{k,L}^i$ is the pre-activation of the k-th neuron in layer L on the input vector i, $|\mathcal{D}|$ is the number of elements in the test dataset and $|L|$ is the number of neurons in the layer. For modular addition, we simply take the mean of the IPR across neurons k in the hidden layer to define $I\tilde{P}R$ as:

$$I\tilde{P}R \equiv \frac{1}{|L|}\sum_{k\in L} IPR(k), \tag{7}$$

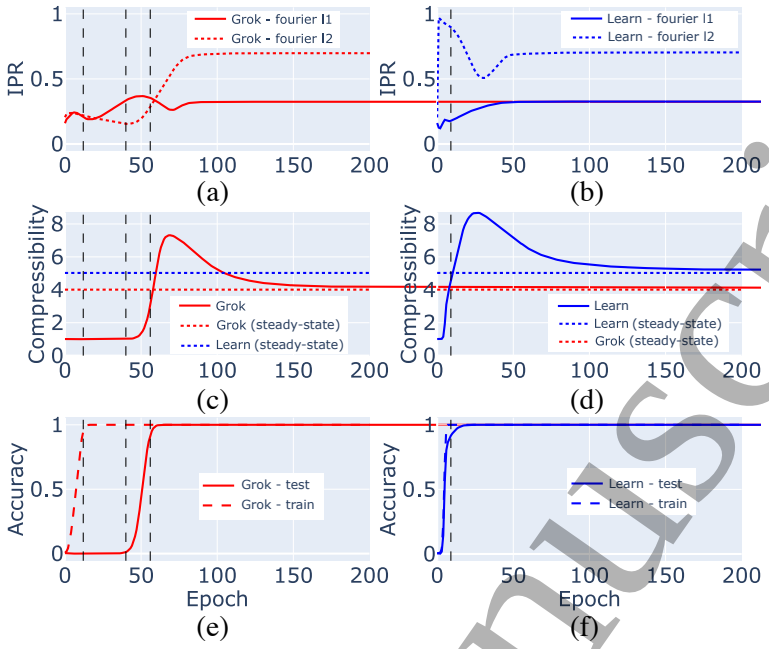


Figure 6: Dynamical measures comparing learning and grokking for modular addition. From left to right, the vertical dashed lines represent the training time, the end of the grokking plateau, and the test time. The model appears to be developing before and during the grokking transition, We also see a transient compressibility spike in the pruning dynamics.

We summarize the behaviour of these measures during training for the modular addition task in fig. 6 and for the Ising task in fig. 7, and additional data is provided in appendix F. The two tasks have markedly different behaviour in the grokking plateau. In the modular addition case, the model improves its localization in the Fourier basis throughout the plateau despite no improvement in model accuracy or loss. This is consistent with the 'circuit formation' picture described for a transformer in Nanda et al.[10] and for a two layer fully-connected network by Gromov [9]. The model trained on the Ising task, however, does not improve its representation of the energy in the plateau - all three metrics are roughly constant across all layers in the plateau. Whether or not the model is developing features in the grokking plateau is hence task dependent. Interestingly, in both grokking and steady learning, the model continues to improve its representation of the dominant feature well after the accuracy and loss are saturated.

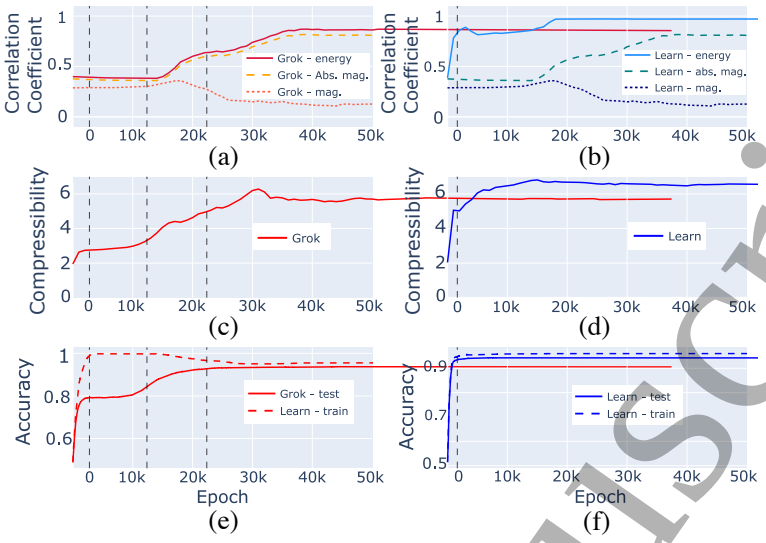


Figure 7: Dynamical measures comparing learning and grokking for the Ising task. Unlike modular addition, the model shows no sign of developing through the grokking plateau.

3.3.2 Dynamics of Compressibility

We also consider the dynamics of model compressibility. The results for the Ising task are given in the middle row of fig. 7(c-d). In the grokking case, we see that the compressibility peaks a short time after the accuracy saturates and then declines slightly. The results for the modular addition task are given in fig. 6(c-d). Here we see a transient 50% larger than the steady state compressibility for the steady learning trajectory, and 67% larger in the grokking trajectory. Comparing to the model formation measures in fig. 6(a-b), we note that after the compressibility peak, the compressibility *decreases* as the neuron IPR $\tilde{r}$ increases. We also note that we observed the transient peak in the compressibility in all our modular addition runs - even at larger batch sizes for which there is no compressive phase. This suggests that the transient behaviour in the compressibility is a distinct phenomenon from terminal model compression.

3.3.3 Information-Geometric Trajectory Analysis

Once a model architecture has been set, the space of parameters defines the space of functions the model can represent, known as the model space. This model space may have a non-trivial geometry, which can be measured by the Fisher Information Metric (FIM). We give a fuller account of this connection in appendix E. The FIM can then be used to measure how the information geometry of a model changes in training.

We use this connection to introduce measures of model 'speed' and 'direction' in model space through training. The measures are the information-geometric generalisation of step magnitude and cosine similarity between consecutive update steps along the trajectory. Given a model space position θ^e at epoch e, the model space step is defined as $s^e := \theta^{e+f} - \theta^e$, where f is the frequency between samples which is 200 (1) for the Ising (modular addition) task. From this the step magnitude and cosine similarity between two steps s and s' are given by

$$|s|_{FIM} := \sqrt{s_i \cdot g_{ij}^{FIM} \cdot s_j} \qquad \in [0, \infty) \,, \tag{8}$$

$$S_{C-FIM}(s, s') := \frac{s_i \cdot g_{ij}^{FIM} \cdot s'_j}{|s|_{FIM} |s'|_{FIM}} \qquad \in [-1, 1] \,, \tag{9}$$

where repeated indices are summed. Focus is placed on the cosine similarity, where values close to one indicate steps that are approximately parallel, while values close to zero indicate orthogonal steps. We track the step cosine similarities, averaged over the 10 seed runs, for both the Ising and modular addition tasks,

and for both the grokking and steady learning. In particular, we compute the cosine similarity of consecutive steps, s^e and s^{e+1}, and show the results in fig. 8(a-b).

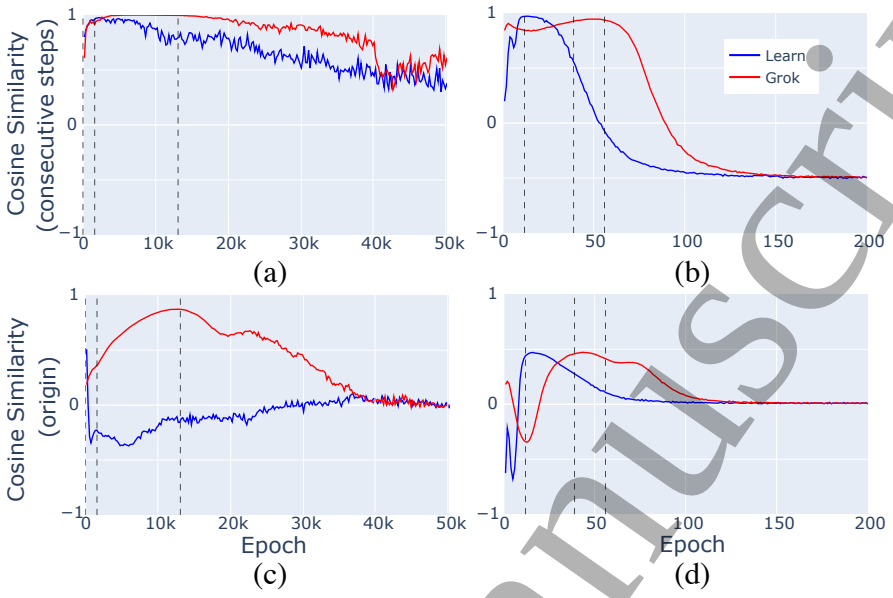


Figure 8: FIM-cosine similarities, $S_{C-FIM}(s, s')$, of the model trajectories through model space during training (averaged over 10 runs). The similarity between consecutive steps, (s^e, s^{e+1}), is shown in (a) for the Ising task and in (b) for modular addition. The similarity between each step and a trajectory pointing to the origin of model, (s^e, s^{OT}), is shown in (c) for the Ising task and in (d) for modular addition.

In both tasks and both regimes, the cosine similarity initially increases, reaches a peak around the learning time, and then decreases to near zero. These results reflect the training dynamics: once the model finds a good trajectory, it continues updating in roughly the same direction until it reaches its minimum loss. After learning, the model updates are random fluctuations around the loss minimum, and consecutive steps become uncorrelated. The grokking regime shows peaks later in the training, and for the Ising task the peak is noticeably higher than in the steady learning regime. In fact, the peak cosine similarity is nearly one, indicating that the model evolves along a straight line in the high-dimensional parameter space.

Previous work suggests that the dynamics before grokking is dominated by a decay of extraneous parameters [10], a hypothesis corroborated by the exponential fitting of the total weight norm in [3]. During uniform weight decay, the trajectory of a model points towards the origin of model space. This is a particularly special path, and may explain the unusual directness of the grokking trajectory for the Ising task.

To understand whether this is actually the case, we compute the cosine similarity between each step and the negative of the model position at the start of that step (which points to the origin). The values of this cosine similarity have a particularly natural interpretation: when they are positive the FIM-weighted average weight is increasing, and for negative values it is decreasing. In the limits of $S_{C-FIM} \approx \pm 1$, uniform weight decay/growth is the dominant feature. These results, for both the Ising and modular addition tasks, and for both the grokking and steady learning regimes, are shown in fig. 8(c-d). At late times, all cases exhibit a cosine similarity of zero, indicating neither weight decay or growth once the terminal model is reached.

For the Ising task, the grokking regime initially follows a path close to the origin trajectory with the cosine similarity approaching one. The opposite occurs in the learning case: the negative similarity score corresponds to network parameters marginally increasing. These results provide further evidence that, in the Ising task, the dominant effect before the grokking transition is weight decay. Furthermore, such a trajectory also explains why feature development does not occur until the grokking transition. In the loss plateau prior to the transition, the dynamics are almost entirely weight-decay driven, and there appears to be no 'hidden' model formation.

For the modular addition task, the cosine similarity to the origin trajectory is somewhat different. At the start of training, both grokking and steady learning regimes exhibit an early regime where the model

seems to be moving slightly away from the origin, which later gives way to a regime of slightly origin-oriented updates approaching the learning transition. However, the cosine similarity is significantly lower than in the Ising task, indicating a trajectory that is less dominated by weight decay. This corroborates the fact that for modular addition, the grokking trajectory also develops features before the grokking transition, and thus the dynamics cannot be simply a uniform decay of weights.

Overall, these novel information-geometric inspired measures provide a new means of analysing machine learning training dynamics by considering the geometry of the model space implicitly. By tracking the direction of the trajectory in model space, we have provided evidence that grokking dynamics is dominated by weight decay for the Ising task, but not for modular addition. Besides capturing the directions of model updates, the FIM can also be used to track their magnitude. An analysis of update step magnitude is left to the appendix, but we note that it displays unique gradient changes at the start and end points of the grokking plateau, and hence suggests a novel method of identifying grokking. Unlike the task-specific feature probes used above, these information-geometric quantities do not require hand-crafting observables for each dataset, which makes them a promising candidate for transfer to larger settings (subject to scalable FIM approximations) [26].

4 Conclusion

To compare the features learned in grokking and learning as closely as possible, we studied toy models with clearly measurable features, and where hyperparameter sweeps could be feasibly performed.

To begin, in section 3.1, we found that for the two tasks considered, grokking and steady learning converged to the same dominant features, while steady learning often produced more compressible models. On this basis, no intrinsic advantage for grokking was observed in the tasks studied. At the same time, these results should not be taken to imply that grokking is generally unhelpful: other tasks or architectures may still benefit from delayed transitions, or from properties not assessed in this study, such as robustness to distribution shift [6] or improved circuit efficiency [27].

Additionally, in section 3.2 we discovered a compressive regime in ordinary learning and not in grokking. The very large compression achieved raises the interesting possibility of using such a training scheme to obtain more compressible models in general. Moreover, the compressive regime allows one to tune the size of the effective model by changing the initial weight multiplier. Since models are known to compress features by superposition [28], it would be particularly interesting to investigate in further work whether models in this regime display significantly higher superposition than models trained outside of it. The compressive regime suggests a route to reducing effective model size, and could in certain circumstances provide a means to improve inference performance.

Furthermore, in section 3.3, we examined how information geometry may be used to help scale interpretability. Our interpretability measures allowed us to follow model development through training. Although this provided progress measures in which the model develops smoothly in the modular addition task, it does not do so for the Ising task, where generalisation still emerges suddenly. Crucially, these measures had to be constructed by hand for each task. By contrast, our information geometric measures were general, and identified signatures of the absence of model development in the Ising task as well as the non-trivial pre-grokking dynamics of the modular addition task. Similar analysis on larger models would ideally employ approximate FIM estimators [26].

Finally, we propose some next steps for further work. In this note, we focussed on a pair of toy models with interpretable features. Whilst this choice is beneficial from an interpretability standpoint, it may limit direct extrapolation to larger and more complex systems. As such, future work may examine feature development, for example, in language models undergoing the grokking transition.

Data Availability

The code and datasets used in this work are available here: `https://github.com/xand-stapleton/grokking_vs_learning`.

Acknowledgments

We gratefully acknowledge extremely helpful discussions with Marc S. Klinger. Marc brought together the initial collaboration, and contributed important insights on information geometry. The work of B.B. was supported by the US National Science Foundation under Grant No. DMR-1945058. A.G.S, D.S.B, & E.H acknowledge support from Pierre Andurand over the course of this research. G.D.T. acknowledges the support from the EPiQS Program of the Gordon and Betty Moore Foundation. Dmitry Manning-Coe gratefully acknowledges the funding support and hospitality of the Machine Learning Alignment and Theory Scholars Program (MATS), where part of this research was conducted. This work made use of the Illinois Campus Cluster, a computing resource that is operated by the Illinois Campus Cluster Program (ICCP) in conjunction with the National Center for Supercomputing Applications (NCSA) and which is supported by funds from the University of Illinois at Urbana-Champaign. This research was supported in part by the Illinois Computes project which is supported by the University of Illinois Urbana-Champaign and the University of Illinois System. This research also utilised Queen Mary's Apocrita HPC facility [29], supported by QMUL Research-IT.

References

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

[2] T. Agrawal and M. Kumar, *Analysis of grokking-induced rigidity in neural networks using transfer learning*, in *Proceedings of Data Analytics and Management*, A. Swaroop, B. Virdee, S.D. Correia and Z. Polkowski, eds., (Cham), pp. 422–433, Springer Nature Switzerland, 2026.

[3] Z. Liu, E.J. Michaud and M. Tegmark, *Omnigrok: Grokking beyond algorithmic data*, 2023.

[4] V. Thilak, E. Littwin, S. Zhai, O. Saremi, R. Paiss and J. Susskind, *The slingshot mechanism: An empirical study of adaptive optimizers and the grokking phenomenon*, 2022.

[5] T. Kumar, B. Bordelon, S.J. Gershman and C. Pehlevan, *Grokking as the transition from lazy to rich training dynamics*, 2024.

[6] Z. Tan and W. Huang, *Understanding grokking through a robustness viewpoint*, 2024.

[7] A. Imtiaz Humayun, R. Balestriero and R. Baraniuk, *Deep Networks Always Grok and Here is Why*, *arXiv e-prints* (2024) arXiv:2402.15555 [2402.15555].

[8] S.V. Kozyrev, *How to explain grokking*, *arXiv e-prints* (2024) arXiv:2412.18624 [2412.18624].

[9] A. Gromov, *A simple and interpretable model of grokking modular arithmetic tasks*, 2024.

[10] N. Nanda, L. Chan, T. Lieberum, J. Smith and J. Steinhardt, *Progress measures for grokking via mechanistic interpretability*, 2023.

[11] E. Ising, *Beitrag zur theorie des ferromagnetismus*, *Zeitschrift für Physik* **31** (1925) 253.

[12] R. Baxter, *Exactly Solved Models in Statistical Mechanics*, Dover books on physics, Dover Publications (2007).

[13] W. Hu, R.R.P. Singh and R.T. Scalettar, *Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination*, *Phys. Rev. E* **95** (2017) 062122 [1704.00080].

[14] P. Suchsland and S. Wessel, *Parameter diagnostics of phases and phase transition learning by neural networks*, *Phys. Rev. B* **97** (2018) 174435 [1802.09876].

[15] D. Doshi, A. Das, T. He and A. Gromov, *To grok or not to grok: Disentangling generalization and memorization on corrupted algorithmic datasets*, 2024.

[16] S.-i. Amari, R. Karakida and M. Oizumi, *Fisher information and natural gradient learning in random deep networks*, in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, K. Chaudhuri and M. Sugiyama, eds., vol. 89 of *Proceedings of Machine Learning Research*, pp. 694–702, PMLR, 16–18 Apr, 2019, https://proceedings.mlr.press/v89/amari19a.html [1808.07172].

[17] D.S. Berman and M.S. Klinger, *The Inverse of Exact Renormalization Group Flows as Statistical Inference*, *Entropy* **26** (2024) 389 [2212.11379].

[18] D.S. Berman, M.S. Klinger and A.G. Stapleton, *Bayesian renormalization*, *Mach. Learn. Sci. Tech.* **4** (2023) 045011 [2305.10491].

[19] D.S. Berman, M.S. Klinger and A.G. Stapleton, *NCoder – A Quantum Field Theory approach to encoding data*, 2, 2024.

[20] D.P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

[21] P. Suchsland and S. Wessel, *Parameter diagnostics of phases and phase transition learning by neural networks*, *Phys. Rev. B* **97** (2018) 174435.

[22] J. Hoogland, G. Wang, M. Farrugia-Roberts, L. Carroll, S. Wei and D. Murfet, *The developmental landscape of in-context learning*, 2024.

[23] Z. Chen, E. Lau, J. Mendel, S. Wei and D. Murfet, *Dynamical versus bayesian phase transitions in a toy model of superposition*, 2023.

[24] E. Lau, Z. Furman, G. Wang, D. Murfet and S. Wei, *The local learning coefficient: A singularity-aware complexity measure*, 2024.

[25] X. Zhu, Y. Fu, B. Zhou and Z. Lin, *Critical data size of language models from a grokking perspective*, 2024.

[26] J. Halverson, T.R. Harvey and M. Nee, *Naturalness and fisher information*, 2026.

[27] V. Varma, R. Shah, Z. Kenton, J. Kramár and R. Kumar, *Explaining grokking through circuit efficiency*, 2023.

[28] N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec et al., *Toy models of superposition*, *Transformer Circuits Thread* (2022) .

[29] T. King, S. Butcher and L. Zalewski, *Apocrita - High Performance Computing Cluster for Queen Mary University of London*, March, 2017. 10.5281/zenodo.438045.

[30] N. Panickserry and D. Vaintrob, *devinterp*, 2024.

[31] S.-i. Amari, *Information geometry and its applications*, vol. 194, Springer (2016).

[32] G. Alain and Y. Bengio, *Understanding intermediate layers using linear classifier probes*, 2018.

[33] D.S. Berman, J.J. Heckman and M. Klinger, *On the Dynamics of Inference and Learning*, 4, 2022.

[34] G. Blanc, N. Gupta, G. Valiant and P. Valiant, *Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process*, 2020.

[35] H. Cheng, M. Zhang and J.Q. Shi, *A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations*, 2024.

[36] G.C. Marinó, A. Petrini, D. Malchiodi and M. Frasca, *Deep neural networks compression: A comparative survey and choice recommendations*, *Neurocomputing* **520** (2023) 152.

[37] D. Hwang, *Fadam: Adam is a natural gradient optimizer using diagonal empirical fisher information*, 2024.

[38] B. DeMoss, S. Sapora, J. Foerster, N. Hawes and I. Posner, *The Complexity Dynamics of Grokking*, *arXiv e-prints* (2024) arXiv:2412.09810 [2412.09810].

[39] S. Fan, R. Pascanu and M. Jaggi, *Deep grokking: Would deep neural networks generalize better?*, 2024.

[40] T. George, *NNGeometry: Easy and Fast Fisher Information Matrices and Neural Tangent Kernels in PyTorch*, Feb., 2021. 10.5281/zenodo.4532597.

[41] R.J. Glauber, *Time-Dependent Statistics of the Ising Model*, *J. Math. Phys.* **4** (1963) 294.

[42] J.N. Howard, M.S. Klinger, A. Maiti and A.G. Stapleton, *Bayesian rg flow in neural network field theories*, 2024.

[43] A. Jeffares, A. Curth and M. van der Schaar, *Deep learning through a telescoping lens: A simple model provides empirical insights on grokking, gradient boosting & beyond*, 2024.

[44] Y. LeCun, J. Denker and S. Solla, *Optimal brain damage*, *Advances in neural information processing systems* **2** (1989) .

[45] A. Lewkowycz and G. Gur-Ari, *On the training dynamics of deep networks with l_2 regularization*, 2021.

[46] Z. Liu, O. Kitouni, N. Nolte, E.J. Michaud, M. Tegmark and M. Williams, *Towards Understanding Grokking: An Effective Theory of Representation Learning*, May, 2022. 10.48550/arXiv.2205.10343.

[47] Z. Liu, Z. Zhong and M. Tegmark, *Grokking as compression: A nonlinear complexity perspective*, 2023.

[48] G. Minegishi, Y. Iwasawa and Y. Matsuo, *Bridging lottery ticket and grokking: Is weight norm sufficient to explain delayed generalization?*, 2024.

[49] M.A. Mohamadi, Z. Li, L. Wu and D.J. Sutherland, *Why do you grok? a theoretical analysis of grokking modular addition*, 2024.

[50] Z. Liu, O. Kitouni, N.S. Nolte, E. Michaud, M. Tegmark and M. Williams, *Towards understanding grokking: An effective theory of representation learning*, 2022.

[51] L. Prieto, M. Barsbey, P.A.M. Mediano and T. Birdal, *Grokking at the edge of numerical stability*, 2025.

[52] H. Tessier, V. Gripon, M. Léonardon, M. Arzel, T. Hannagan and D. Bertrand, *Rethinking weight decay for efficient neural network pruning*, *Journal of Imaging* **8** (2022) 64 [2011.10520].

[53] N. Rubin, I. Seroussi and Z. Ringel, *Grokking as a first order phase transition in two layer networks*, 2024.

[54] I. Seroussi, G. Naveh and Z. Ringel, *Separation of scales and a thermodynamic description of feature learning in some cnns*, 2022.

[55] D. Wright, C. Igel and R. Selvan, *Bmrs: Bayesian model reduction for structured pruning*, 2024.

A Model Parameters and Seed Distribution

In this work, we control grokking solely by the modification of the weight multiplier. For convenience, table 1 provides all other model parameters for the data generated in the main text.

Table 1: Model Architecture and Training Parameters

Parameter	**Ising CNN**	**Modular addition MLP**
Weight decay	0.1	3×10^{-5}
Learning rate	10^{-4}	5×10^{-3}
Learning rate scheduler	10^{-4}	5×10^{-3}
Epochs	100,000	1000
Loss	Cross Entropy	Cross Entropy
Optimizer	Adam	Adam
Activation	ReLU	ReLU
Modulation (P)	–	113
Train fraction	–	70%
Train dataset size	300	8938
Batch Size	300	64
Test dataset size	1000	3831
Fully connected layer size	100	512
Convolutional channels	2,4	–
Convolutional stride	2,2	–
Kernel size	2,2	–

We also provide summary data for the individual runs. In fig. 9 we provide the distribution over individual seeds for Ising runs. We see that final model accuracy and loss are consistent as we tune from grokking into

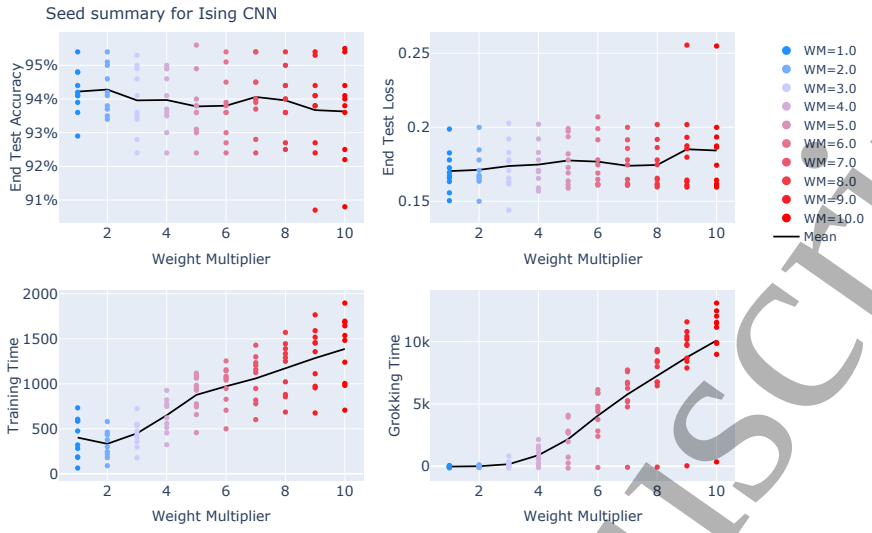


Figure 9: A summary of the outcomes of the runs for the 10 different seeds used for each weight multiplier. We see that all bar one of the runs groks for the higher weight multiplier regime of the plot, and that for the majority of seeds the final test accuracies and losses for grokking and learning are within a few percent of each other.

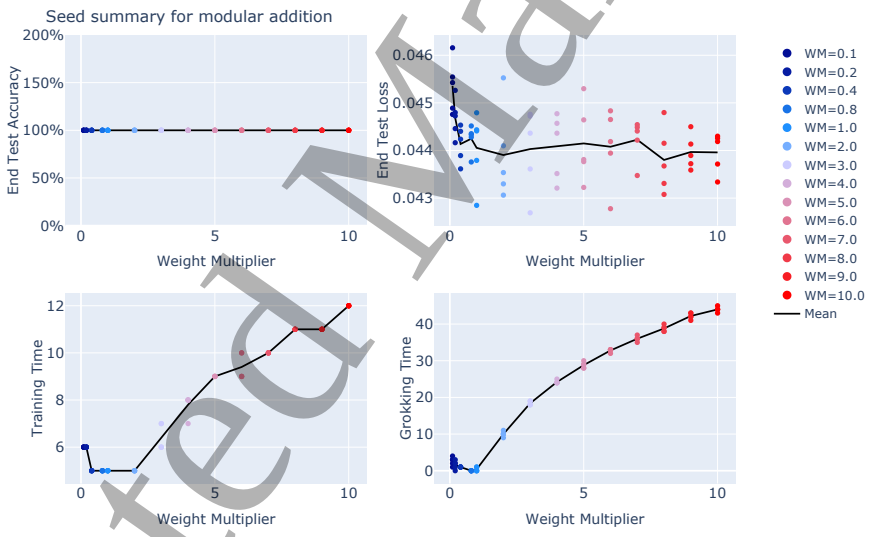


Figure 10: A summary of the outcomes of the runs for the 5 different seeds used for each weight multiplier in modular addition.

learning. We also see a slight increase in seed variability as we tune deeper into the grokking regime, with one seed persistently failing to grok.

For modular addition, we provide the summary data over seeds in fig. 10. We see that the modular addition task is significantly less seed dependent than the Ising task. All seeds reach 100% accuracy, and the end losses are within 5% of each other, whereas for the Ising tasks, end accuracies are within 5% and end losses can vary by more than 50%. We also note that, interestingly, both modular addition and the Ising task show an increase in training time as they transition from the grokking to the learning regime.

B The Ising Classification Problem

This section gives a more formal definition of the Ising model used in the main text. Generically, the Ising model is defined on a graph G (with a vertex set V and an edge set E), where each node is associated with a discrete variable $\sigma_x \in \{-1, 1\}$, called a spin. The space of spin configurations (snapshots) is given by all possible combinations $\Omega = \{-1, 1\}^V$. The Ising model can be defined as a probability distribution over this space of snapshots Ω. More precisely, a snapshot $\{\sigma_x\}_{x\in V}$ is drawn from the Boltzmann distribution:

$$p(\{\sigma_x\}) = \frac{e^{-E(\{\sigma_i\})/T}}{Z(T)}, \tag{10}$$

where T is the temperature of the system, $Z(T)$ is a normalization factor (the partition function), and $E(\{\sigma_x\})$ is the energy (cost) function:

$$E(\{\sigma_x\}) = -J \sum_{(i,j)\in E} \sigma_i \sigma_j. \tag{11}$$

Here the sum is over edges of the graph, and $J > 0$ is the interaction strength between spins, which in the main text is set to one for simplicity. The phase transition is determined by the competition between the interaction J, which favours ordering by aligning the spins to minimize the energy in eq. (11), and the temperature T, which enlarges the width of the probability distribution in eq. (10), and thus favours disorder by exploring spin configurations with random spin alignments.

In the main text, we consider the case in which the graph G is the square two-dimensional ($2D$) lattice of linear size $L = 16$ ($|V| = L^2$). In the limit of large linear size ($L \to \infty$), the model presents a phase transition at $T_c = \frac{2J}{\log(1+\sqrt{2})}$, separating an ordered phase at $T < T_c$ and a disordered phase at $T > T_c$.

Formally, these two phases are uniquely characterized by the averaged magnetization density (the local order parameter):

$$\langle M \rangle(T) := \lim_{h\to 0} \lim_{L\to\infty} \langle \frac{1}{L^2} \sum_{i\in V} \sigma_i \rangle_h, \tag{12}$$

where $\langle\cdot\rangle_h$ indicates the average over the Boltzmann probability distribution with an external field, $p_h(\sigma_x) = \frac{e^{-E_h(\sigma_i)/T}}{Z(T,h)}$, with $E_h(\sigma_x) = E(\sigma_x) - h\sum_{i\in V}\sigma_i$, and $Z(T,h)$ a normalization factor. The term $-h\sum_{i\in V}\sigma_i$ adds an energy penalty for having $\sigma_x = -1$, breaking the energy degeneracy $E(\sigma_x) = E(-\sigma_x)$. In the ordered phase, $\langle M \rangle$ is positive, while in the disordered phase, it is zero in the limit $L \to \infty$. In particular, for the two-dimensional case:

$$\langle M \rangle(T) = \begin{cases} \left(1 - \frac{1}{\sinh^4 2J/T}\right)^{1/8}, & T < T_c \\ 0, & T > T_c. \end{cases} \tag{13}$$

From a concrete point of view, spin configurations drawn from Eq. (10) for a finite L, which can be seen simply as binary images, look very different for T smaller or larger than T_c. For $T = 0$, the Boltzmann probability distribution is concentrated with equal weight in only two configurations: all spins $+1$ or all spins -1, and the absolute value of the magnetization, $|\sum_{x\in V}\sigma_x|$, equals $|V|$. As the temperature increases within the ordered phase, thermal fluctuations induce small clusters of spins with a sign opposing the bulk magnetization. Nevertheless, the magnetization remains extensive in absolute value. In the opposite limit of $T \to \infty$, deep in the disordered phase, $p(\{\sigma_x\}) = 1/2^{|V|}$, and all configurations are equally probable. Consequently, typical configurations consist of random spins, and the absolute value of the magnetization is small compared to $|V|$. These qualitatively different spin arrangements make detecting the phase of a finite-size snapshot similar to the problem of image recognition.

The snapshot configurations used to train our convolutional neural network are generated using local-update Monte Carlo simulations. Specifically, we use the Glauber dynamics algorithm [41], which we describe below.

To sample a spin configuration from the Boltzmann probability distribution at fixed T, one starts with a configuration at infinite temperature, $T \to \infty$. In other words, one samples a spin configuration $\{\sigma_x\}_{x\in V}$ from the uniform probability distribution, $p(\{\sigma_x\}) = 1/2^{|V|}$. Next, one updates individual spins to obtain

a typical configuration drawn from Eq. (10) and approach the target average energy cost function. Each update is called a Monte Carlo step.

For concreteness, we consider the case in which the graph G is a two-dimensional grid of linear size L, and each node is specified by two-dimensional coordinates $1 \leq x, y \leq L$. We take periodic boundary conditions, e.g., $\sigma_{L+1,y} = \sigma_{1,y}$. The procedure is as follows:

1. We sample a spin configuration (snapshot), $\sigma_{x,y}$, from the uniform probability distribution and calculate its energy, $E_0(\sigma_{x,y})$.

2. We randomly choose a node (x, y) and compute the energy, E_t, of a new configuration in which the spin at that node is flipped: $\sigma_{x,y} \to -\sigma_{x,y}$.

3. We choose to flip the spin and update the spin configuration with probability $p_{\text{flip}} = 1/(1 + e^{\Delta E/T})$, where $\Delta E = E_t - E_0$ is the energy difference between the two spin configurations. If the spin-flip is accepted, we set $E_0 = E_t$.

4. We repeat steps 2 and 3, updating the spin configurations, for a total of N Monte Carlo steps. We take $N \gg L^2$ to ensure at least partial equilibration before saving the resulting snapshot.

To generate the training data for our classification task, we repeat this procedure 1000 times for 50 temperatures in the interval $[T_c - 1, T_c + 1]$ while fixing $J = 1$. We then index the (finite-size) snapshots by their temperature, and their ultimate phase label (ordered or disordered) is based on the phase of an infinite system with that temperature. This procedure is guaranteed to generate snapshots of the Ising model in equilibrium if it is run for infinite time. In equilibrium, snapshots are completely characterized by their magnetization, and the classification task is almost trivial.

We therefore choose to run the simulation for a large but fixed number of steps N so that a few snapshots of the training data are not fully equilibrated. In particular, local-update Monte Carlo simulations like ours often get 'stuck' in metastable configurations before reaching equilibrium. By fixing a stopping point for our simulations, we necessarily include some metastable snapshots in our data. More specifically, we include so-called extended domain wall configurations, which are ordered snapshots that have an anomalously low magnetization [21]. These snapshots are correctly classified as ordered by looking at their energy, but are misclassified if our network only learns to discern magnetization. A few examples of these extended domain wall configurations are shown in fig. 11. By including such snapshots in the training data, we allow our network to learn both the magnetization and the energy.

Another important difference between the theoretical Ising model and our snapshots is the finite size of our grid. The main effect of finite system size is that the critical point at T_c extends to a critical region surrounding T_c. In an infinite system, the correlation length [1] between spins diverges precisely at T_c. In our finite lattice, however, as soon as the correlation length is larger than the system size ($L = 16$), the system behaves critically. In training our model, this finite-size effect actually provides a useful test of whether the model overfits. Roughly 10% of our training snapshots are in this 'critical region', and therefore cannot be classified using any physical quantities. If the model's accuracy on the training data is near 100%, then, it means that it must be memorizing the phase labels of these ambiguous points. On the other hand, after the model has learned, the test accuracy drops to around 95%, which is the best it can do using physical features of the snapshots.

C Feature Development in the Ising CNN

In fig. 12 and fig. 13 we provide the by-layer breakdown, across all seeds, of the features learned by the model trained on the Ising task for the grokking and learning cases, respectively. We note first that features become better defined as we step through the network. In the first CNN layer, there appears to be no information about the energy or absolute magnetization. In the second CNN layer this information emerges, and then is processed by the hidden layer, where the absolute value of the correlation to the energy in each neuron is around 0.99 and to the absolute magnetization around 0.9. We note in passing that the fully

[1] The correlation length in the Ising model roughly corresponds to the scale on which spins are correlated with each other in a snapshot.

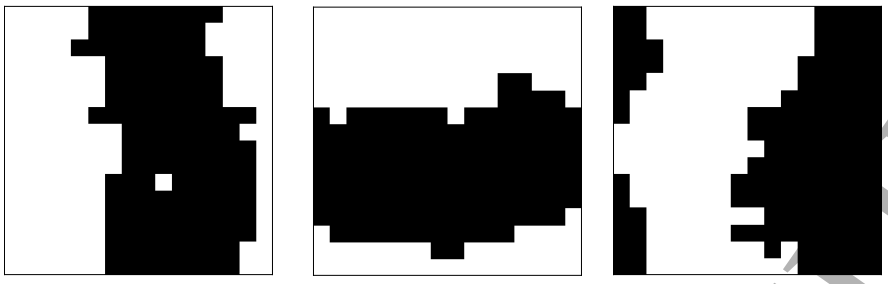


Figure 11: Three extended domain wall configurations of the Ising model. These snapshots are correctly classified as ordered by the energy, but are misclassified if the network only learns its magnetization. Such configurations are common in local update Monte Carlo simulations.

connected layer can be considered to be acting as a non-linear probe on the second convolutional layer [32], which supports the view that the information about the energy is already formed at this point in the network.

We also note that the representation is more noisy in the grokking than in the learning regime. In the learning regime fig. 13, every seed has almost all of its fully connected layer neurons close to perfectly correlated or anti-correlated with the energy. In the grokking regime fig. 12, three of the seeds have significant deviation from perfect correlation with the energy, and instead correlate to the *raw* value of the magnetization.

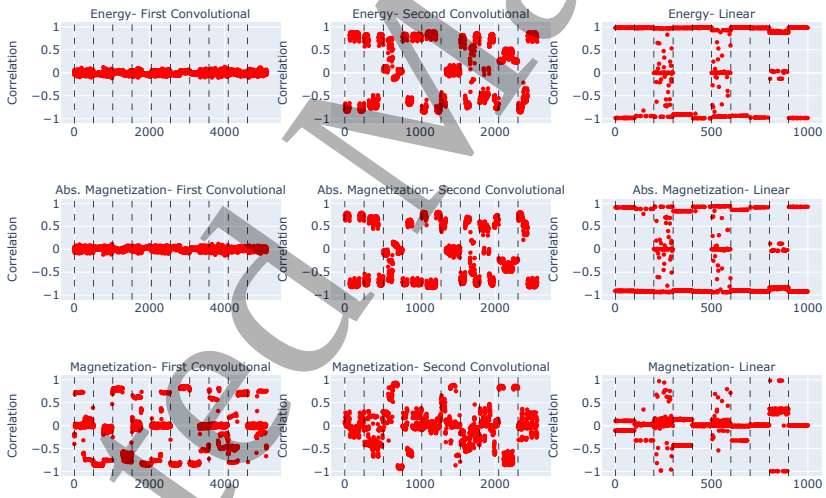


Figure 12: A summary of the correlation coefficients for the different measures across seeds and all layers for the grokking case in Ising. Vertical dashed lines separate different seeds. We can see that although the dominant behaviour is still to learn the magnetization, the grokking case is noisier than the learning case; in some seeds the model learns the magnetization.

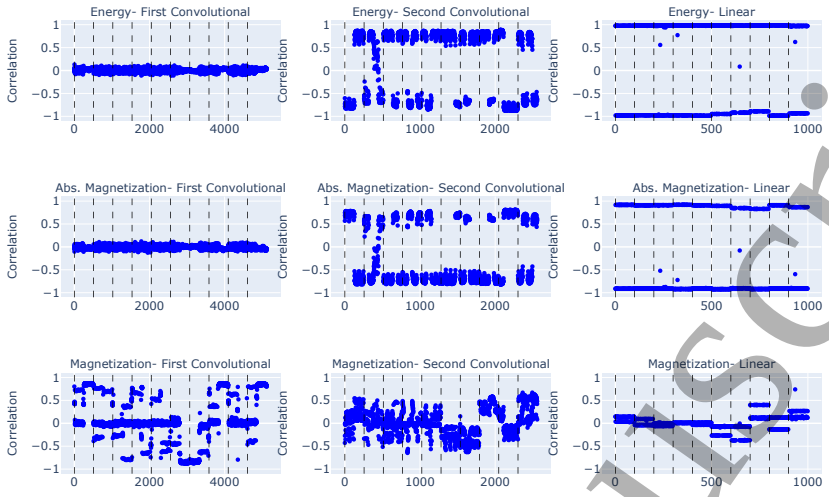


Figure 13: A summary of the correlation coefficients for the different measures across seeds and all layers for the learning case in Ising. Vertical dashed lines separate different seeds.

D Compressibility

We present additional results for compressibility. In the modular addition task, we identified a 'compressive regime' which is characterized by a linear trade-off between end model loss and compressibility. Whether or not this regime exists depends on a wider range of model parameters. Although we have not done an exhaustive parameter search for the compressive regime, we have found that it is increasing the batch size to 200 from 64 in the main text eliminates the compressive effect, as shown in fig. 14.

To better understand the nature of the compressive regime, we also provide data for the correlation between model compressibility and a range of other measures in fig. 15. We also provide equivalent data for the non-compressive, batch size 200, runs in fig. 16. We first note that, despite the linear relationship with the *train* loss in the compressive regime, there is no relationship with the *test* loss in either the compressive or non-compressive regime.

Second the compressive regime is associated with a significant reduction in the localization of neurons to a particular frequency. We see in the rightmost plots of the bottom row that for both input and output weight matrices to the hidden layer, the compressive regime leads to an improvement in compressibility through a decrease in Fourier basis localization. In the extremely compressed regime that we discuss in appendix D.1, this effect becomes very dramatic. The low weight multiplier runs are associated to an almost complete breakdown of Fourier basis localization (fig. 15). It is important to point out that the reduction in the Fourier basis localization is a property of the compressive regime and not a difference between grokking and learning generally. In fig. 16 we provide data for the same measures, but at the higher batch size. Here, there is no significant difference between the IPRs of grokking and learning.

Third, unsurprisingly, more compressible models have higher weight Gini coefficients. In addition to the Gini coefficients of the weight magnitudes, we also measure the Gini coefficient of the diagonal entry corresponding to a given weight in the Fisher Information Metric as discussed in appendix E. The Gini coefficients of the Fisher diagonals are substantially higher than the Gini coefficients of the raw weights. This should be unsurprising, since the Fisher diagonals should give a more refined measure of the importance of a given weight. Given this observation, is it puzzling why pruning on the basis of the Fisher diagonals gives worse performance than magnitude pruning, as we shown in appendix E.1. Resolving this tension could help us to understand how the FIM can be used in ablations and other in other interpretability contexts.

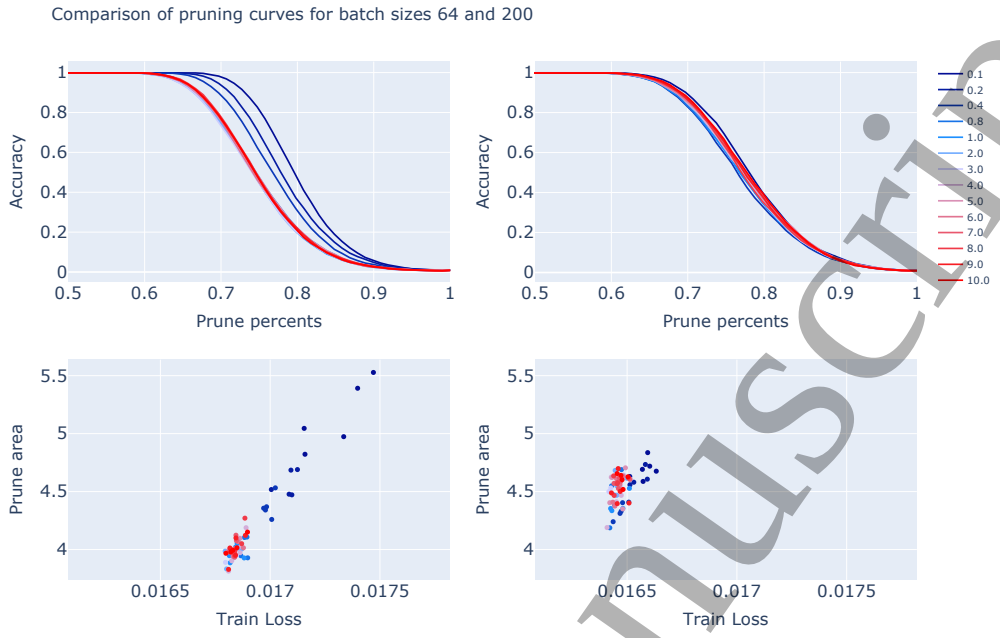


Figure 14: Pruning curves for batch size 64 (left) as in the main text, and 200 (right). All other parameters are as in table 1. We see that changing the batch size eliminates the compressive regime. At batch size 200 there is no significant increase in compressibility at lower weight multiplier.

Finally, we also compute the 'local learning coefficient' (LLC) for all runs, using code from [30]. The LLC is intended to be a measure of a model degeneracy. Intuitively, models with higher degeneracy should be expected to be more compressible. Surprisingly, there have not been many studies that investigate the relationship between the LLC and pruning in a controlled way. Since the compressive regime provides a way to tune the model compressibility we are able to do so in this context. Interestingly, we find no relationship between the LLC and the model's compressibility. A deeper understanding of this observation could open the way towards developing the LLC as a tool for model compression.

We note that that since there is no compressive phase in the Ising problem, there is no trade-off between the loss and level of model compression from the weight multiplier, shown in fig. 17.

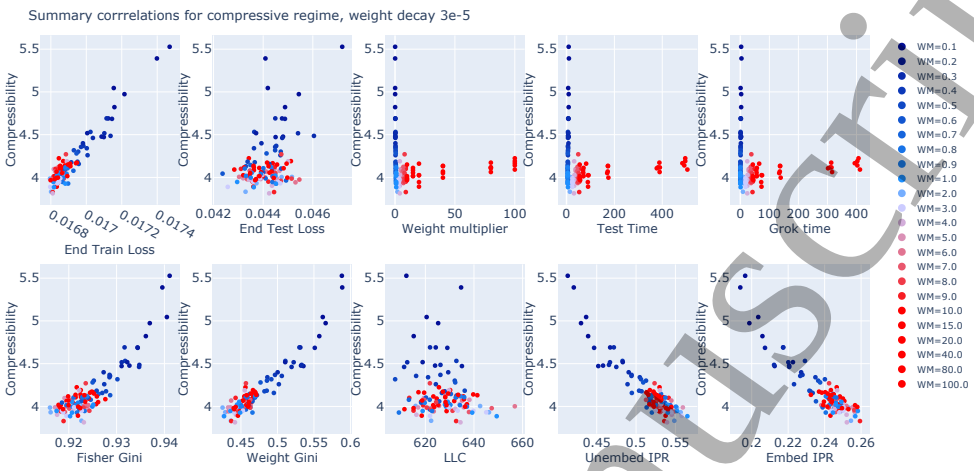


Figure 15: Correlations between model compressibility and other model properties for the parameters considered in the main text. We note that the although the train loss trades off linearly with the compressibility, there is no relationship with the magnetization. In the compressive regime, we also see a reduction in localization in the Fourier basis, as measured by the IPR.

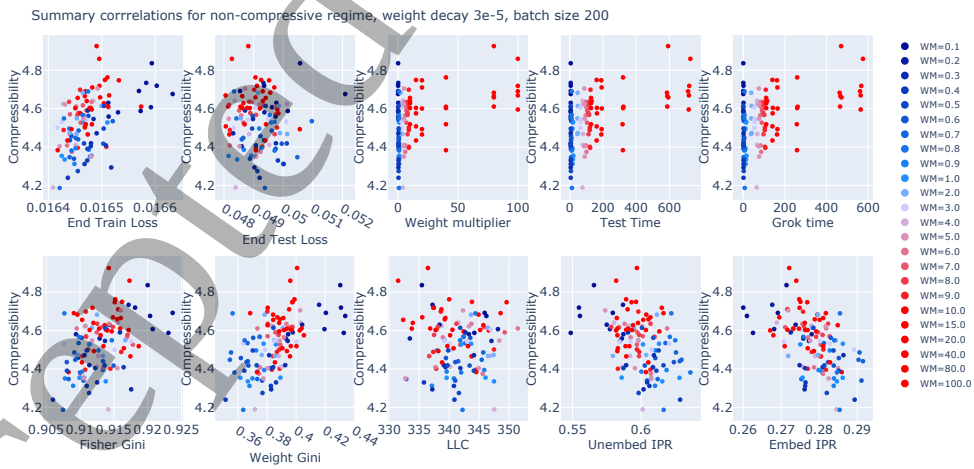


Figure 16: Correlations between model compressibility and other model properties for batch size 200, and remaining parameters as in the main text. Note that here, there is no relationship between grokking and the IPRs.

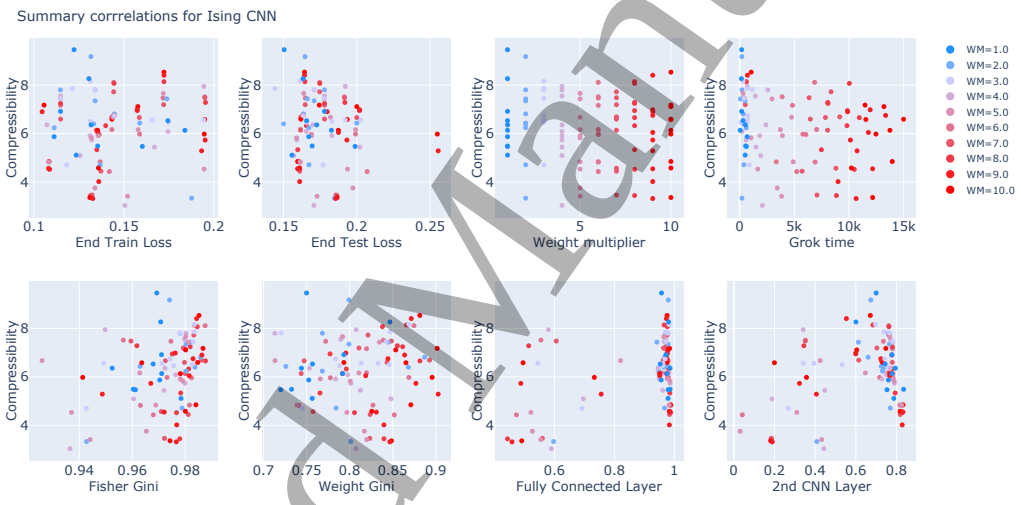


Figure 17: In the Ising case, there is no compressive phase, and so there is no trade-off between the end-loss and the model compression accessible by tuning the weight multiplier.

D.1 25x Compression in Modular Addition

Conducting a rough parameter search, we found a range with extremely compressed models for weight decay 3×10^{-4}, an order of magnitude larger than that considered in the main text. In fig. 18, we show the pruning curves for the same parameters as in the main text, but at the higher weight decay. We see a very large improvement in the pruning, that occurs suddenly as we transition into the compressive regime. Model compressibility increases suddenly from around 32% of the weights needed to maintain an accuracy of 95% at $w_0 = 2$ to only 5% of the weights needed to maintain the same accuracy at $w_0 = 1$. This corresponds to a compressibility of **25x the base model, and 5x the compression achieved in the main text.**

To better understand this highly compressed regime, we again provide data for the same measures as considered earlier in this section in fig. 19. We see a very stark difference between the compressive regime and the other weight multipliers. Most strikingly, we see in the two rightmost sub-figures of the bottom row that the Fourier basis localization has almost completely broken down. Interestingly, in this regime, model train and test loss are negatively correlated to the compressibility.

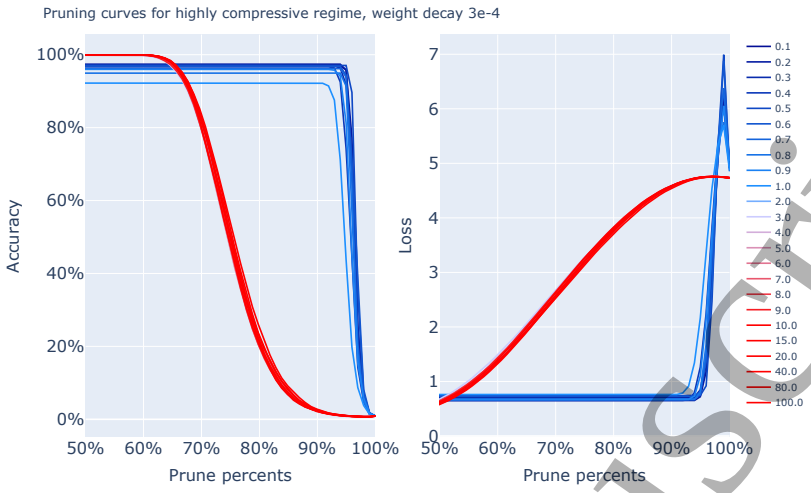


Figure 18: Pruning curves for the highly compressed curves in the top left of the modular addition pruning areas curve. There is a very sharp transition to very highly compressed models between weight multipliers $w_0 = 2$ and $w_0 = 1$.

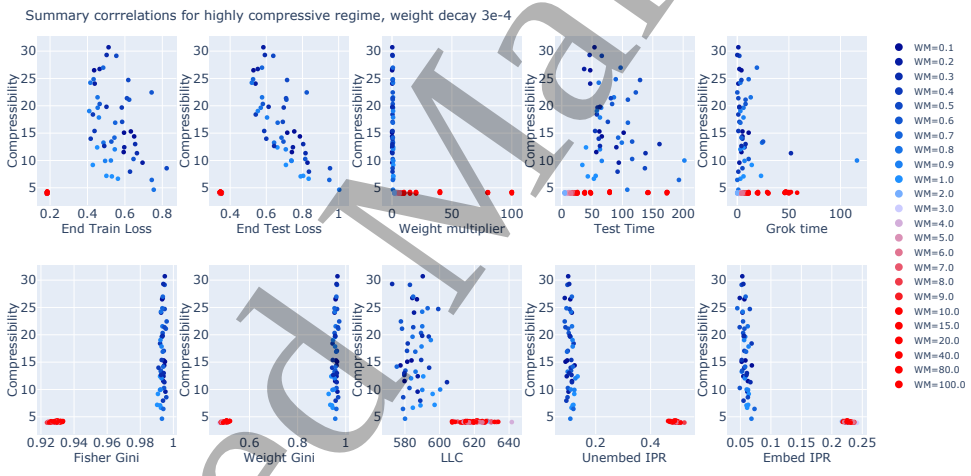


Figure 19: Correlations between the compressibility and measures of training dynamics and model features in the highly compressive regime (weight decay 3×10^{-4}).

E Fisher Information Geometry

This section provides a whistle-stop tour of the basics of information geometry. For a more complete treatment, see for example [31].

Given an architecture function, which for us we take to be a neural network, $f_{NN}(x_{\text{input}})$, with parameters θ_i, the space of neural networks with this architecture is spanned by the allowed values of the parameters. Each weight and bias may take any real value, so this model space is homeomorphic to $\mathbb{R}^N$, where there are N parameters. Each set of parameter values defines a point on the model space, and as the parameters change smoothly[2] through training a path on the model space is traced out, forming a trajectory.

To define a notion of distances on the model space we require a metric. Where the task we consider is a classification problem (as in the tasks considered in this work), the trained function output is a normalised

[2]Practically the updates are discrete and hence the path is not smooth, but we can consider the infinitesimal update limit for this theoretical illustration.

probability distribution (often enforced by a final softmax activation). For this we first assume the model space is a manifold, and then consider the tangent space, traditionally spanned by a basis $\{\frac{\partial}{\partial\theta_i}\}$ but may also suitable by spanned by the basis of score vectors[3] $\ell_i = \frac{\partial}{\partial\theta_i}\ln(f_{NN}(x|\theta))$ which is the basis we choose $\{\ell_i\}$. The Fisher metric is then defined on this tangent space basis as:

$$\begin{aligned} g_{ij}^{FIM} &:= \mathbb{E}_x(\ell_i\ell_j)\ , \\ &= \mathbb{E}_x\left(\frac{\partial\ln(f_{NN}(x|\theta))}{\partial\theta_i}\frac{\partial\ln(f_{NN}(x|\theta))}{\partial\theta_j}\right)\ . \end{aligned} \tag{14}$$

The expectation is theoretically an integral over the entire data space x, however practically we compute a discrete expectation as an average over samples from the training data. The metric changes depending on the position in model space since f_{NN} is a non-linear function of θ, and this is what defines the non-flat geometry of the space. The discrete derivatives are computed using `pytorch` functionality, via the `nngeometry` package [40].

The Fisher information metric provides a measure of the similarity between two models, well illustrated by the connection to the KL-divergence, D_{KL}, which measures the relative entropy between probability distributions $p(x|\theta)$

$$D_{KL}(\theta'|\theta) = \mathbb{E}_x\left(\ln\left(\frac{p(x|\theta')}{p(x|\theta)}\right)\right)\ , \tag{15}$$

which has a unique global minimum of 0 when $\theta' = \theta$, and in the neighbourhood of this minimum the KL-divergence expands as

$$D_{KL}(\theta+\delta\theta|\theta) = \frac{1}{2}g_{ij}^{FIM}(\theta)\delta\theta_i\delta\theta_j + \mathcal{O}(\delta\theta^3)\ . \tag{16}$$

Therefore the FIM acts as an infinitesimal measure of distance in the model space.

In the Ising task, the neural network architecture used consists of 6798 parameters, whilst for the modular addition task the architecture has 174193 parameters. This leads to real symmetric non-degenerate FIM matrices of respective sizes $g^{FIM}_{6798\times6798}$, and $g^{FIM}_{174193\times174193}$. However, since the number of parameters is large, one can use the assumption that these matrices are approximately diagonal [16] to ease computational cost; thus we use a vector of entries g_{ii}^{FIM} to represent the FIM diagonal for each of the parameters θ_i.

During the training the network parameters (i.e. the positions in model space) are recorded at regular intervals. Equivalently the FIM diagonal is also computed at these same intervals. For the Ising task these were at intervals of 200 epochs for the first 100000 epochs, and for the modular addition task at intervals of 1 epoch for the first 500 epochs; after each of these periods the behaviour between the steady learning and grokking regimes were deemed equivalent.

For the neural network models trained on the tasks considered in this work, the final FIM diagonal spectra after training, averaged over the 10 seed runs, can be computed and plotted (in importance increasing order for each parameter); this is shown in fig. 20. These spectra show gaps in all cases, which are approximately at the same positions, and the dashed line marks the 0.1 value selected as the threshold for the stiff × sloppy splitting of the model space.

As described by the works [18], one may consider the FIM diagonal as a measure of the relative importance of parameters to the model.

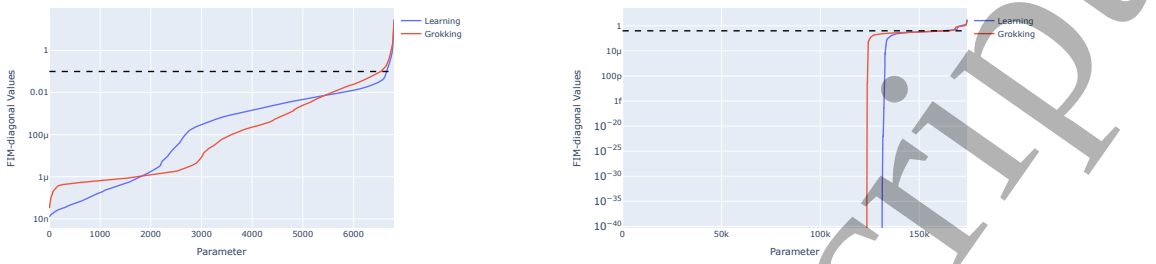
E.1 Fisher Pruning

As an alternative to magnitude pruning, this section applies the Fisher pruning (BRG) scheme [17, 18, 33, 42] to the models studied throughout the note.

The geometry of model space is often assumed to be flat, such that a Euclidean metric[4] can be used to compute distances. However one can compute a more natural information geometry metric which directly depends on the architecture: the Fisher Information Metric (FIM). The FIM, g_{ij}^{FIM}, as introduced in appendix appendix E, enables the intrinsic geometry of the model space to be probed.

[3] Note the classification nature of the task is essential here, as non-negativity of the outputs is required for logarithms to be well defined over the real numbers.

[4] Where the Euclidean metric is, for practical purposes, the identity matrix.



(a) Ising (b) ModAdd

Figure 20: FIM diagonal values for the neural network model after training, sorted increasing values, and averaged over the 10 seed runs. Plots showing both the learning and grokking regimes; for both the Ising (a), and ModAdd (b) tasks. The dashed line represent the 0.1 value set as the threshold for the subspace splitting, positioned at the spectrum mass gaps.

Fisher pruning is an information-theoretic network pruning scheme that generalises the principles of the exact renormalization group (ERG) studied in high-energy physics to arbitrarily parameterized probability distributions, including those of NNs. Pruning is performed in parameter space with respect to an information-theoretic distinguishability scale set by the Fisher information metric. Fisher pruning removes neural network parameters which covary weakly with the output of the model[5]. After specifying an architecture for a neural network, one may treat the parameters as a natural set of coordinates for the space of models (associated with the given architecture). Initializing the network involves sampling a random point in model space. During training, the parameters change smoothly, and the position in model space updates tracing out a trajectory.

In the works [17, 18, 33] the eigenspectrum of the FIM is argued to provide a parameter importance measure, allowing the model function to be analysed and pruned in detail. The eigenvalues give this measure of relative importance for linear combinations of parameters (defined by the eigenvectors) to the model. In the case where the metric is approximately diagonal, valid for high-dimensional model spaces [16], the eigenvalues are equal to the diagonal entries and these give a relative importance of each parameter to the model.

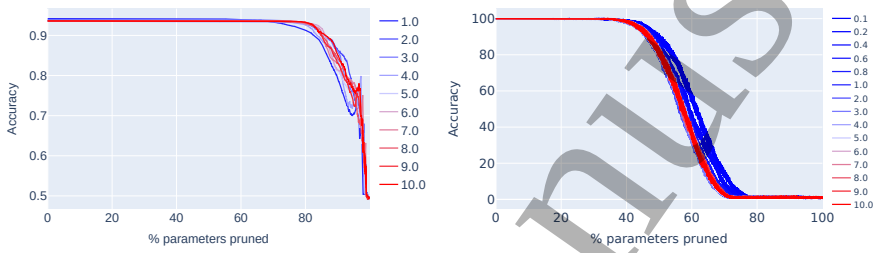
Intuitively one may believe magnitude pruning to be sufficient to determine the compressibility of a model [44]; this is typically not the case[6]. In practice, deep networks are typically highly non-linear (which is at least partly responsible for their great ability to express complex functions). Moreover, modern networks have built-in non-linearities in the form of activation functions. In the case that the activation function $\sigma(x)$ does not monotonically increase, i.e. there exists an x such that $\sigma(x+\delta) \leq \sigma(x)$, for $\delta > 0$, magnitude pruning is poorly motivated. An example of a commonly used activation function possessing this feature is the GELU activation. A novel pruning technique, immune to such restrictions, is Fisher pruning scheme [17, 18]. Fisher pruning is performed hierarchically: parameters are pruned such that those with the smallest associated diagonal Fisher information metric components are removed first. The advantages of the Fisher pruning scheme are most manifest when pruning the model holistically, i.e. each layer simultaneously.

In fig. 21, we present loss and accuracy plots as a function of number of parameters pruned for both the Ising and modular arithmetic problems. In agreement with the magnitude pruning analysis, we conclude hyperparameters conducive to grokking do not consistently result in a pruning advantage. The Ising network presents an interesting property: there does not exists a clear pruning advantage in the grokked regime when compared the steady-learning phase (for a batch size of 200). While one might argue that the chosen weight multiplier (10) to represent 'grokking' provides a pruning advantage for the archetypical value selected for steady learning (0.1), it is evident that other weight multiplier values resulting in learning produce the opposite effect. We thus conclude that there is not a definitive compressibility behaviour that can be attributed to grokking in the Ising model.

[5]This technique is also known as BRG [42].

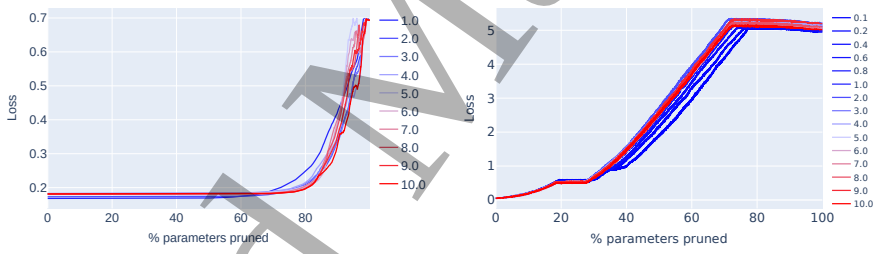
[6]There do exist some exceptions; for example, magnitude-pruning a very simple single-layer perceptron network, with monotonically increasing activation functions, trained subject to the maximisation of output vectors would be sufficient.

The modular arithmetic network (fig. 21) tells a similar story, hyperparameter choices resulting in grokking do not produce a more prunable model than those which do not under the Fisher pruning regime (unlike the magnitude pruning regime). This analysis clearly shows that a grokked model is not necessarily a more compressed model. Compressibility is attributed to the weight multiplier and not the presence of grokking. A natural question one may ask is: 'how does Fisher pruning act per-layer?' In order to facilitate such an analysis, we propose freezing splitting the model parameters from the whole model into two non-overlapping sets $\theta_a \sqcup \tilde{\theta}_a$, where a is the index of the layer to be examined. For each model, we compute the Fisher metric $\mathcal{I}_{ij}(\theta)$ and systematically remove θ_k^a in increasing order of their associated $\mathcal{I}_{ij}$ value. A per-layer breakdown of the Fisher pruning regime for the modular addition problem is provided in fig. 22, whilst an analogous plots for the Ising problem are presented in fig. 23.



(a) Ising – Fisher pruning (acc.)

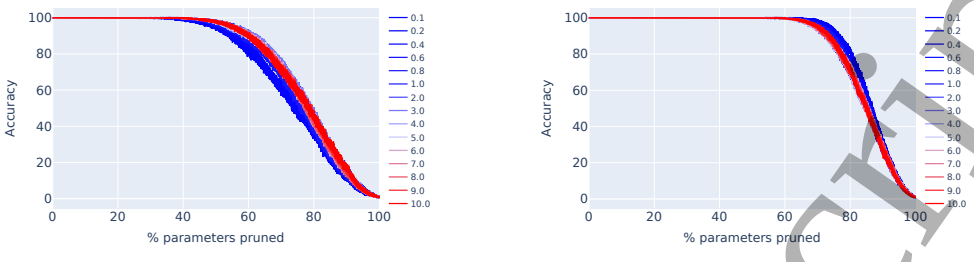
(b) ModAdd - Fisher pruning (acc.)



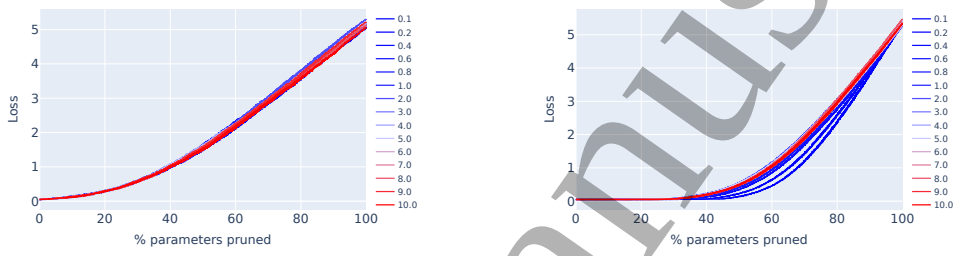
(c) Ising – Fisher pruning (loss)

(d) ModAdd - Fisher pruning (loss)

Figure 21: Accuracies and losses as a function of % parameters pruned for the grokked and steady-learning regimes (Ising and modular arithmetic networks).

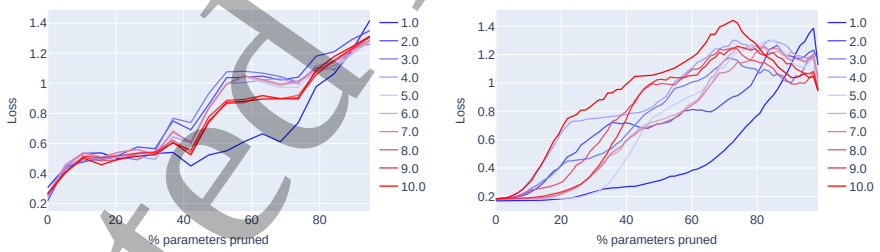


(a) Acc. Layer 1 (b) Acc. Layer 2

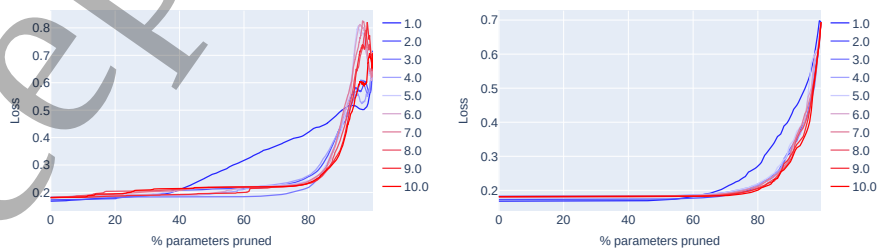


(c) Loss Layer 1 (d) Loss Layer 2

Figure 22: Layer-wise losses and accuracies of the modular addition network output as a function of parameters Fisher-pruned



(a) Loss Layer 1 (b) Loss Layer 2



(c) Loss Layer 3 (d) Loss Layer 4

Figure 23: Losses of Ising network output as a function of the number of parameters Fisher-pruned

E.2 Other Magnitude Pruning Schemes

In the main text, we chose to prune our networks using a particular magnitude pruning scheme in which the same percentage of weights are pruned in each layer. In appendix E.1, we discuss the results of several pruning schemes that are based on the Fisher metric instead of the (Euclidean) magnitude of the weights. Here, we show several other magnitude pruning schemes for comparison to the Fisher-based approach. In particular, we focus on global unstructured pruning (both with and without rescaling each weights by layer norms) and layer-resolved pruning.

In fig. 24, we show the results of the different global pruning schemes for both tasks. The pruning scheme in the main text (parallel pruning in each layer) is shown in the first column, naive global unstructured pruning is shown in the second column, and global unstructured pruning done after rescaling all weights by their respective layer norms is shown in the third column. Using naive global unstructured pruning, it appears that a larger fraction of the network can be pruned at no loss in accuracy compared to the other strategies. However, this is an artefact of the large number of insignificant weights in the last fully connected layer.

Since unstructured pruning removes the smallest weights independently of where they are in the network, it results in a behaviour very similar to simply pruning weights in the last layer, where the majority of the smallest weights reside. The network is relatively resilient to this kind of pruning, while pruning its initial convolutional layers, which have significantly fewer parameters, is more drastic. These observations can be verified by comparing fig. 24 to the results of pruning one layer at a time in the Ising network, shown in fig. 25. We show the same layer-resolved pruning scheme for modular addition in fig. 26.

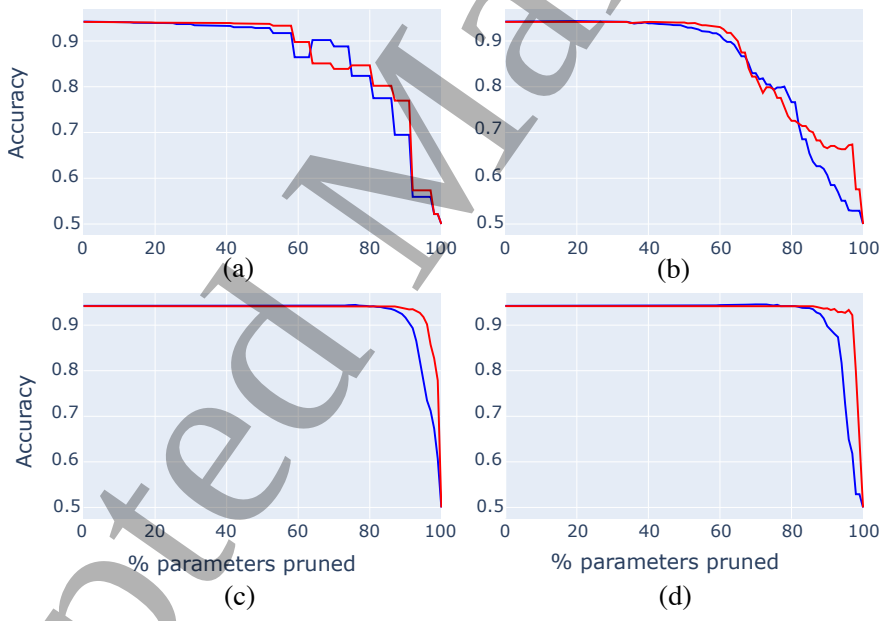


Figure 24: Different global magnitude pruning schemes for the two tasks, averaged over 8 seeds and using two weight multipliers. The results of pruning the Ising network are shown on the top row, while the analogous results for modular addition are on the bottom row. (a,d) Parallel pruning in all layers (as in main text) (b,e) Global unstructured pruning, and (c, f) Global unstructured pruning with each weight rescaled by the norm of all weights in its layer.

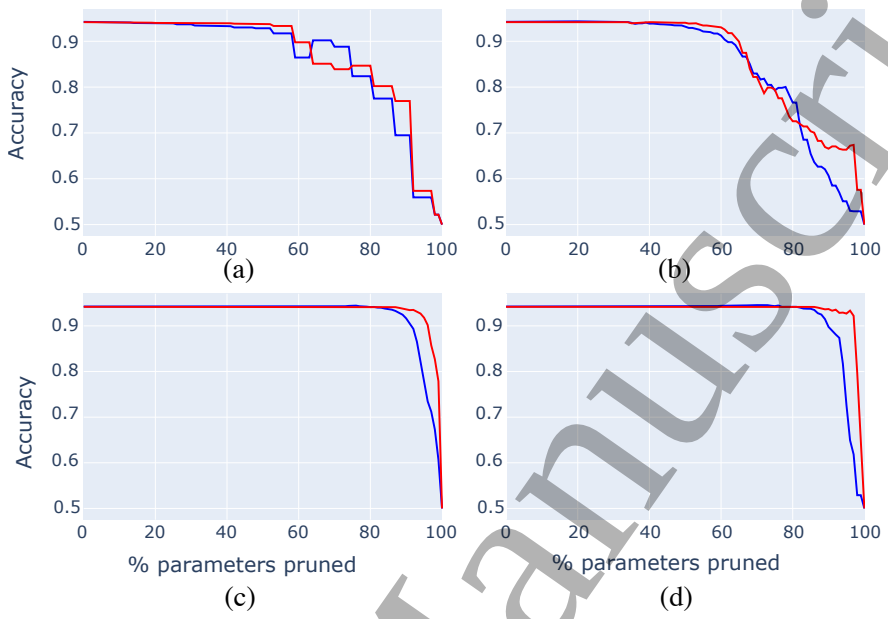


Figure 25: Layer-resolved magnitude pruning of the weights in the Ising network. Results of pruning only (a) The first CNN, (b) The second CNN, (c) The first fully connected layer, and (d) The second fully connected layer.

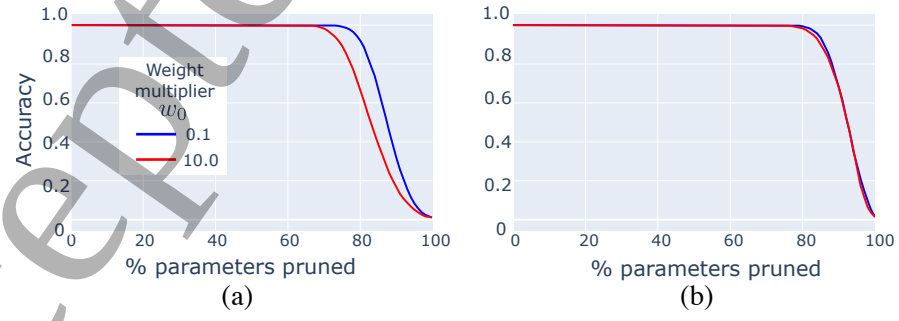


Figure 26: Layer-resolved magnitude pruning of the weights in the modular addition network. Results of pruning only (a) The first fully connected layer and (b) the second fully connected layer.

F Dynamics

F.1 Feature Development Through Interpretability

We provide here additional measures for feature development in the Ising task, across layers. In addition to the pre-activation weighted volume that we considered in the main text, we also consider the fraction of neurons which have their highest correlation coefficient with the each measure, and the share of the total pre-activation accounted for by each measure. We provide these measures for the first CNN layer in fig. 27, for the second CNN layer in fig. 28, and for the last layer in fig. 29.

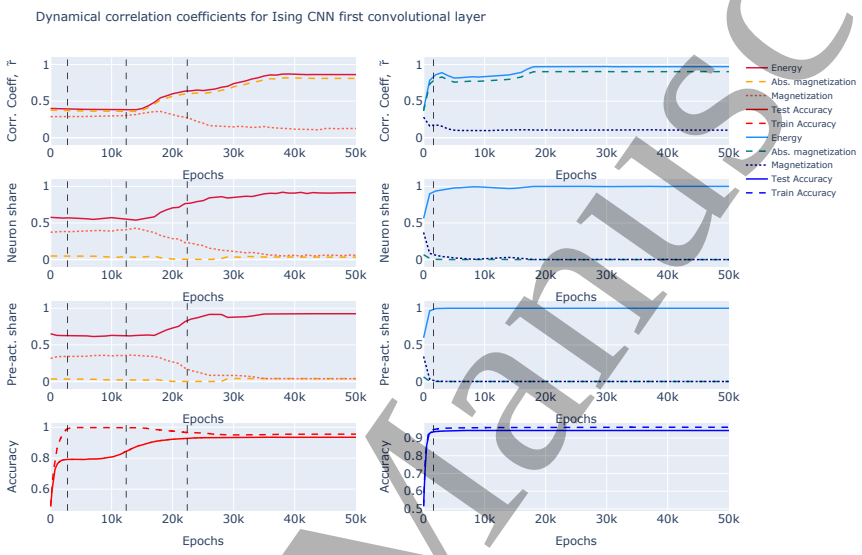


Figure 27: A summary of the correlation coefficients for the different measures across seeds and the first convolutional layer for the grokking case in Ising.

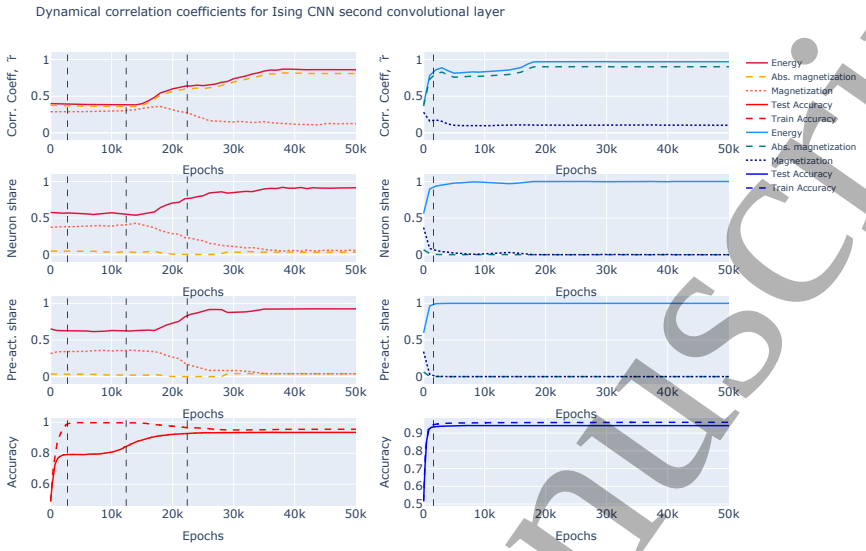


Figure 28: A summary of the correlation coefficients for the different measures across seeds and the second convolutional layer for the grokking case in Ising.

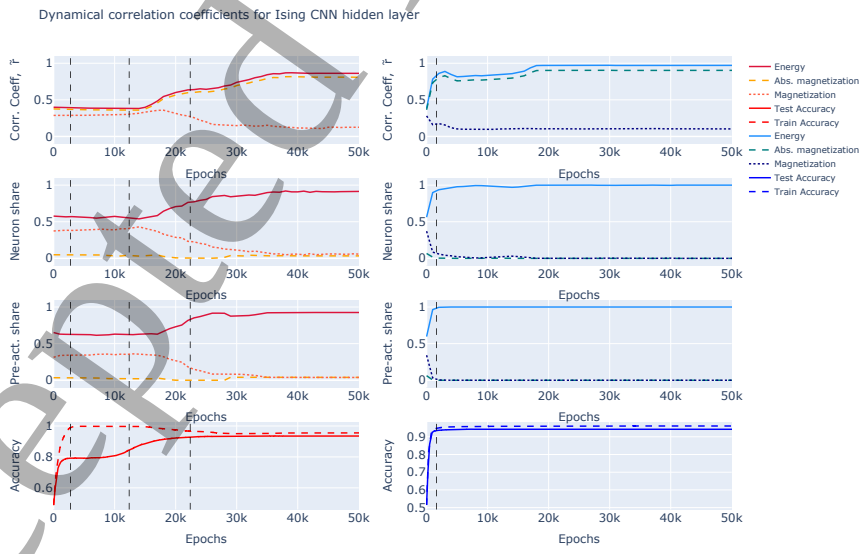


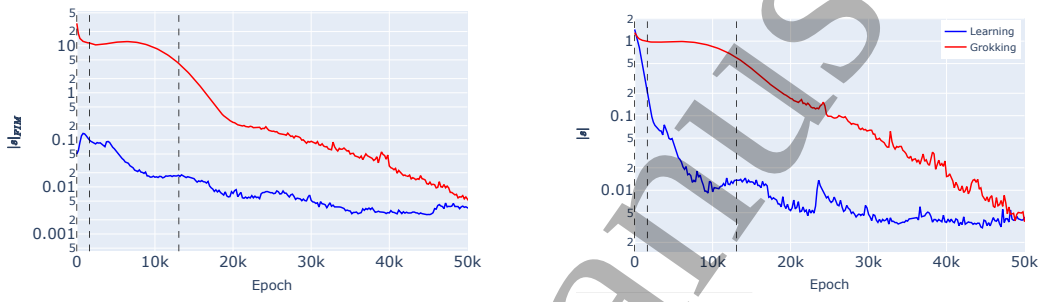
Figure 29: A summary of the correlation coefficients for the different measures across seeds and the fully connected layer for the grokking case in Ising.

F.2 Comparing Fisher and Euclidean Model Space Geometry

One may wonder how the novel FIM-inspired dynamical measures would act in the limit that the model space is assumed Euclidean, such that the FIM is not computed or used.

Across the presented plots related to these measures each were also regenerated using a Euclidean version of the measure[7]. Overall, the observed behaviour was similar, with the FIM versions of the measures showing the analysed features to be more pronounced. Motivating their use as improvements on more traditional assumptions, and better theoretically motivated for the analysis of model space.

As an example, in fig. 30, the step magnitudes for the Ising task are shown using the FIM-magnitude measure and also with the simplified Euclidean version. One may observe the FIM measure better distinguishes between the grokking and steady Learning regimes, and the 'bump' feature as grokking begins is more pronounced.



(a) FIM Measure (b) Euclidean Measure

Figure 30: Step magnitudes of the models' trajectories through model space during training. Measures computed using the FIM $|s|_{FIM}$, or simply with the Euclidean assumption $|s|$, and averaged over the 10 runs.

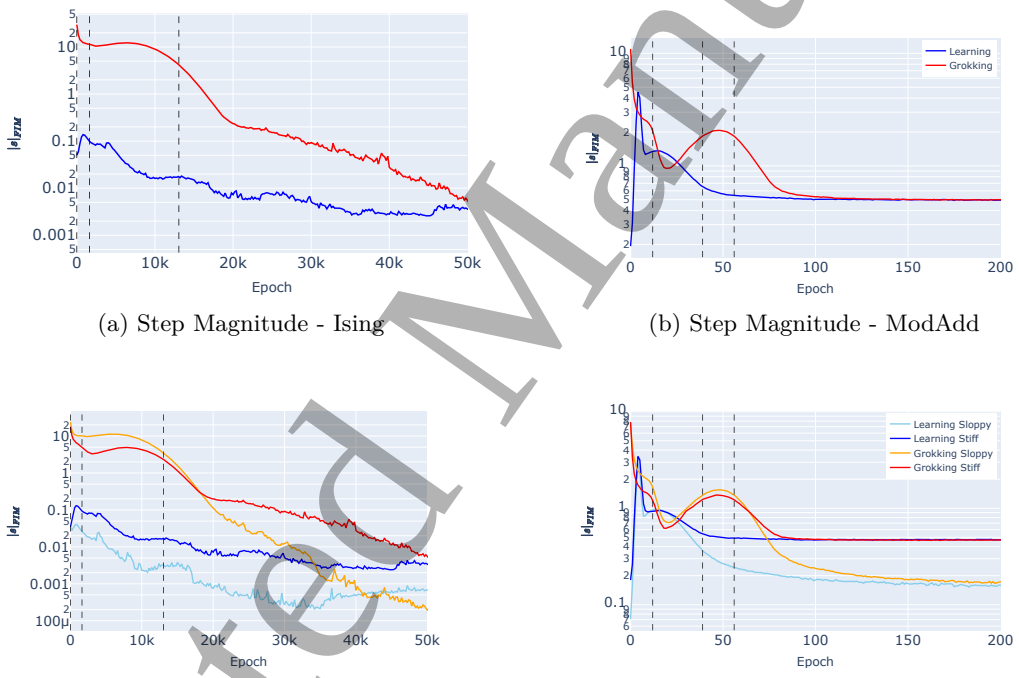
[7] In fact, this paper's respective codebase allows full optimised functionality for this should one prefer to use it.

F.3 Fisher Step Magnitudes

Considering the spectrum of diagonal entries averaged over the 10 seed runs, shown in appendix E, the spectra exhibit a common mass gap, such that a subset of the parameters have relatively negligible importance. This subset of parameters are labelled 'sloppy', whilst the complement subset are labelled 'stiff'. Motivated by this mass gap, it is interesting to consider splitting the model space into two subspaces[8], one parameterised by the stiff parameters and one by the sloppy. The model space positions and FIMs can be split onto these subspaces, and the equivalent novel trajectory analysis measures reduced to be defined on each of these subspaces also.

In both tasks a FIM diagonal threshold of 0.1 is used to discern whether a network parameter is stiff or sloppy. The FIM diagonal is considered for the final model after training, and parameters are assigned to be stiff (or sloppy) when $g_{ii}^{FIM} \geq 0.1$ (or < 0.1). These assignments define the subspaces, which are then fixed and tracked back through the training.

The first of the two novel dynamical measures introduced in this work considers the training step FIM-magnitudes. Averaged over the 10 seed runs, these are shown for both the Ising and modular addition tasks, with and without the stiff-sloppy splitting, and for both the grokking and steady learning regimes, in fig. 31.



(a) Step Magnitude - Ising

(b) Step Magnitude - ModAdd

(c) Split Step Magnitude - Ising

(d) Split Step Magnitude - ModAdd

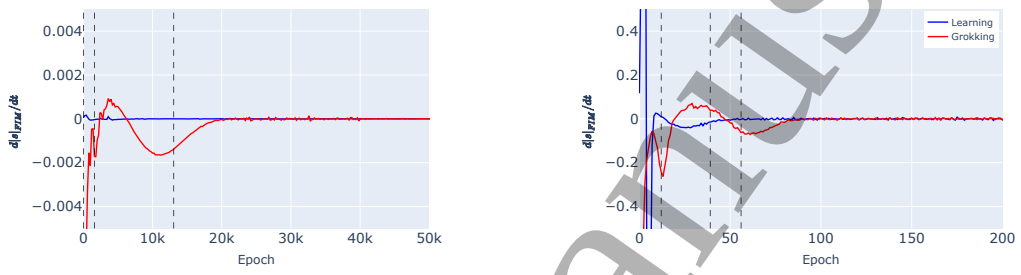
Figure 31: Step magnitudes, $|s|_{FIM}$, of the models' trajectories through model space during training. Measures computed using the FIM, averaged over the 10 runs. Plots show the behaviour of the full model (a) & (b), as well as the sloppy and stiff submodels (c) & (d); for both the learning and grokking regimes; for both the Ising (a) & (c), and ModAdd (b) & (d) investigations.

In both tasks and both regimes the step magnitude generally decreases over the training as the model improves the loss gradient drops and the steps become smaller. The grokking regimes start with higher step magnitude values since their initialisation uses larger weight multipliers, but towards the end of training the step magnitudes are of the same order between the grokking and learning regimes. The noticeable difference is that in the grokking regime the step magnitude has a significant bump in value at the start of the grokking period, where the step magnitude continues to increase during this period. Where a similar behaviour seems to occur earlier in the Learning regime this is much more pronounced in the grokking case, and may be

[8]Here the full model space is treated as a product space and with the stiff × sloppy subspaces independent.

attributed to the optimizer finding an optimal trajectory for improvement which allows the momentum to grow and step magnitude to increase.

From the previous observations, the bump in the FIM step magnitude appears to be a good indicator for the grokking period, this we can make more explicit by plotting the discrete gradient of this graph and considering the roots of the line, as shown in fig. 32. Both tasks show 2 significant roots where the plot crosses gradients of zero before plateauing towards zero gradient, which may be used to define the start and end of the grokking regimes. For the Ising task the first root is between the epoch samples 2200 and 2400, whilst the second root is between 6200 and 6400. For the modular arithmetic task the first root is between epochs 19 and 20, whilst the second root is between epochs 46 and 47. These epoch ranges for the grokking period roughly agree with those specified previously with the measure defined in this paper, supporting this as an alternative mechanism to define the grokking period.



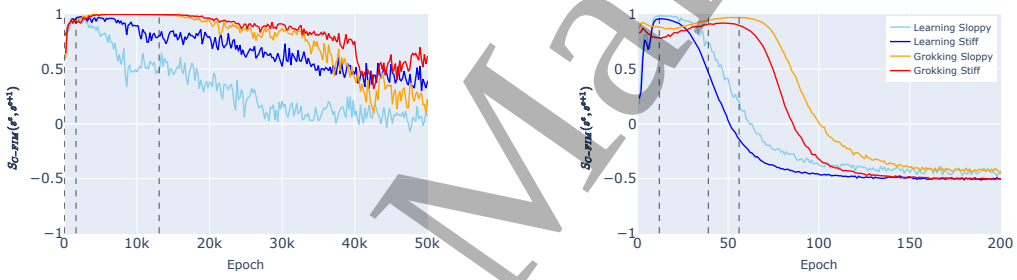
(a) Ising (b) ModAdd

Figure 32: Discrete gradients of FIM steps, $d|s|_{FIM}/dt$, along the model space learning trajectory, averaged over the 10 seed runs. Plots showing both the learning and grokking regimes; for both the Ising (a), and ModAdd (b) investigations. The axes are restricted to highlight the roots.

Considering how the step magnitude analysis splits into the stiff and sloppy subspaces, on both tasks the stiff step magnitudes converge to higher values than the sloppy step magnitudes. Indicating that the optimizer identifies these stiff directions and directs the parameter updates to only change those which have a significant effect on the model. The bump increase in step magnitude during the grokking period is exhibited in both the stiff and sloppy subspaces indicating this is likely a universal feature of grokking across the parameters. Furthermore, in the grokking regime the stiff and sloppy lines cross, indicating that when a model is set up to grok the optimizer will first prioritize changing sloppy features (where this line starts higher) and then will swap after the grokking period has finished to prioritize stiff features.

The second novel dynamical measure, considered primarily in the main body of the text, may also be split to study the trajectories on the sloppy and stiff submanifolds. Where here we only consider $S_{C-FIM}(s^e, s^{e+1})$ for step vectors at consecutive epoch samples, as in the main body but without the origin trajectory comparison. These equivalent plots are shown, averaged over the 10 seed runs, for both the Ising and modular addition tasks, with the stiff-sloppy splitting, and for both the grokking and steady learning regimes, in fig. 33.

In the grokking regime, the behaviour is similar between the subspaces, whereas for the learning regime in the Ising task, the stiff subspace has consistently higher values - indicating that the grokking regime treats the model parameters more symmetrically in the updates.



(a) Ising (b) ModAdd

Figure 33: Split FIM-cosine similarities between consecutive steps, $S_{C-FIM}(s^e, s^{e+1})$, for the sloppy and stiff sub-models. Plots showing both the learning and grokking regimes; for both the Ising (a), and ModAdd (b) investigations.