Towards Understanding Grokking:
An Effective Theory of Representation Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We aim to understand *grokking*, a phenomenon where models generalize long after overfitting their training set. We present both a *microscopic* analysis anchored by an effective theory and a *macroscopic* analysis of phase diagrams describing learning performance across hyperparameters. We find that generalization originates from structured representations whose training dynamics and dependence on training set size can be predicted by our effective theory in a toy setting. We observe empirically the presence of four learning phases: *comprehension*, *grokking*, *memorization*, and *confusion*. We find representation learning to occur only in a "Goldilocks zone" (including comprehension and grokking) between memorization and confusion. Compared to the comprehension phase, the grokking phase stays closer to the memorization phase, leading to delayed generalization. The Goldilocks phase is reminiscent of "intelligence from starvation" in Darwinian evolution, where resource limitations drive discovery of more efficient solutions. This study not only provides intuitive explanations of the origin of grokking, but also highlights the usefulness of physics-inspired tools, e.g., effective theories and phase diagrams, for understanding deep learning.

1 Introduction

Perhaps *the* central challenge of a scientific understanding of deep learning lies in accounting for neural network generalization. Power et al. [1] recently added a new puzzle to the task of understanding generalization with their discovery of *grokking*. Grokking refers to the surprising phenomenon of *delayed generalization* where neural networks, on certain learning problems, generalize long after overfitting their training set. It is a rare albeit striking phenomenon that violates common machine learning intuitions. We attempt to answer the following three questions:

Q1 *The origin of generalization*: When trained on the algorithmic datasets where grokking occurs, how do models generalize at all?

Q2 *The critical training size*: Why does the training time needed to "grok" (generalize) diverge as the training set size decreases toward a critical point?

Q3 *Delayed generalization*: Under what conditions does delayed generalization occur?

We provide evidence that representation learning is central to answering each of these questions. Our answers can be summarized as follows:

A1 Generalization can be attributed to learning a good representation of the inputs, i.e. a representation that has the appropriate structure for the task. See Figures 1 and 2.

A2 The critical training set size corresponds to the least amount of training data that can determine such a representation (which, in some cases, is unique up to linear transformations).
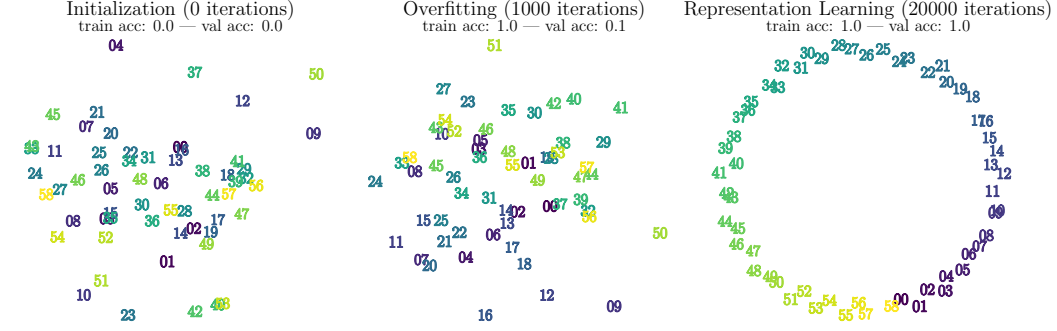
Figure 1: Visualization of the first two principal components of the learned input embeddings at different training stages of a transformer learning modular addition. We observe that generalization coincides with the emergence of structure in the embeddings. See Section 4.2 for the training details.

A3 Grokking is a phase between "comprehension" and "memorization" phases and it can be remedied with proper hyperparmeter tuning, as illustrated by the phase diagrams in Figure 5.

This paper is organized as follows: In Section 2, we introduce the problem setting and build a simplified toy model. In Section 3, we will use an *effective theory* approach, a useful tool from theoretical physics, to shed some light on questions **Q1** and **Q2** and show the relationship between generalization and the learning of structured representations. In Section 4, we explain **Q3** by displaying phase diagrams from a grid search of hyperparameters and show how we can "de-delay" generalization by following intuition developed from the phase diagram. We discuss related work in Section 5, followed by conclusions in Section 6.[1]

2 Problem Setting

Power et al. [1] observe grokking on an unusual task – learning an "algorithmic" binary operation. Given some binary operation $\circ$, a network is tasked with learning the map $(a, b) \mapsto c$ where $a \circ b = c$. They use a decoder-only transformer to predict the second to last token in a tokenized equation of the form "<lhs> <op> <rhs> <eq> **<result>** <eos>". Each token is represented as a 256-dimensional embedding vector. The embeddings are learnable and initialized randomly. After the transformer, a final linear layer maps the output to class logits for each token.

Toy Model We primarily study grokking in a simpler toy model, which still retains the key behaviors from the setup of [1]. Although [1] treated this as a classification task, we study both regression and classification problems. The basic setup is as follows: our model takes as input the symbols a, b and maps them to trainable embedding vectors $\mathbf{E}_a, \mathbf{E}_b \in \mathbb{R}^{d_{\text{in}}}$. It then sums $\mathbf{E}_a, \mathbf{E}_b$ and sends the resulting vector through a "decoder" MLP. The target output vector, denoted $\mathbf{Y}_c \in \mathbb{R}^{d_{\text{out}}}$ is a fixed random vector (regression task) or a one-hot vector (classification task). Our model architecture can therefore be compactly described as $(a, b) \mapsto \text{Dec}(\mathbf{E}_a + \mathbf{E}_b)$, where the embeddings $\mathbf{E}_*$ and the decoder are trainable. Despite its simplicity, this toy model can generalize to all abelian groups (discussed in Appendix A). In sections 3-4.1, we consider only the binary operation of addition. We consider modular addition in Section 4.2 to generalize some of our results to a transformer architecture and study general non-abelian operations in Appendix F.

Dataset In our toy setting, we are concerned with learning the addition operation. A data sample corresponding to $i + j$ is denoted as (i, j) for simplicity. If $i, j \in \{0, \ldots, p - 1\}$, there are in total $p(p + 1)/2$ different samples since we consider $i + j$ and $j + i$ to be the same sample. A dataset D is a set of non-repeating data samples. We denote the full dataset as D_0 and split it into a training dataset D and a validation dataset D', i.e., $D \bigcup D' = D_0$, $D \bigcap D' = \emptyset$. We define *training data fraction* $= |D|/|D_0|$ where $|\cdot|$ denotes the cardinality of the set.

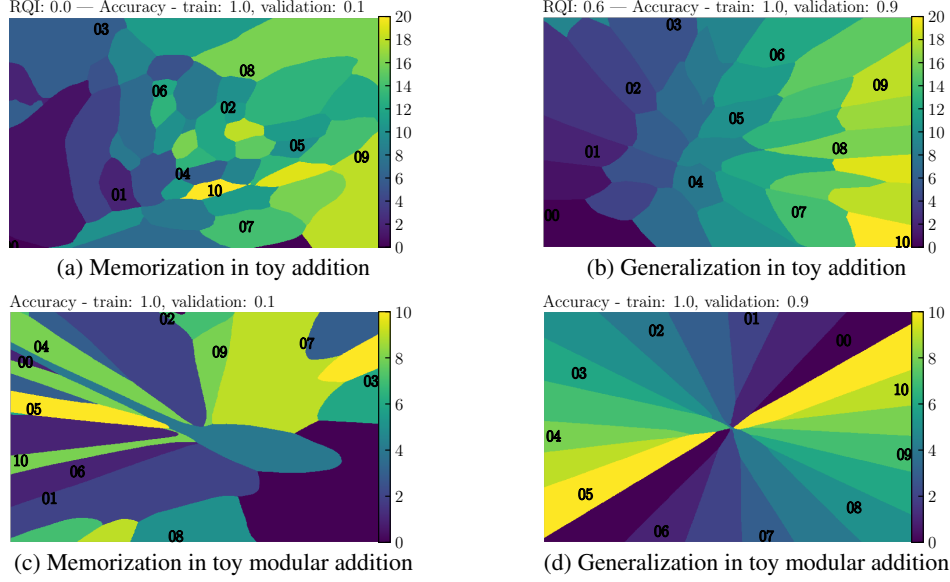[1]Project code can be found at: Removed to maintain anonymity of review.

Figure 2: Visualization of the learned set of embeddings ($p = 11$) and the decoder function associated with it for the case of 2D embeddings. Axes refer to each dimension of the learned embeddings. The decoder is evaluated on a grid of points in embedding-space and the color at each point represents the highest probability class. For visualization purposes, the decoder is trained on inputs of the form $(\mathbf{E}_i + \mathbf{E}_j)/2$. One can read off the output of the decoder when fed the operation $i \circ j$ from this figure simply by taking the midpoint between the respective embeddings of i and j.

3   Why Generalization: Representations and Dynamics

We can see that generalization appears to be linked to the emergence of highly-structured embeddings in Figure 2. In particular, Figure 2 (b) shows parallelograms in toy addition, and (d) shows a circle in toy modular addition. We now restrict ourselves to the toy addition setup and formalize a notion of representation quality and show that it predicts the model's performance. We then develop a physics-inspired *effective* theory of learning which can accurately predict the critical training set size and training trajectories of representations. In physics, effective theories aim to describe observed phenomena without implying that the mechanisms used in the description necessarily have an exact counterpart in the true underlying causes. In our effective theory, we will model the dynamics of representation learning not as gradient descent of the true task loss but rather a simpler effective loss function ℓ_{eff} which depends only on the representations and not on the decoder.

3.1   Representation quality predicts generalization for the toy model

A rigorous definition for *structure* in the learned representation is necessary. We propose the following definition,

Definition 1. (i, j, m, n) *is a δ-parallelogram in the representation* $\mathbf{R} \equiv [\mathbf{E}_0, \cdots, \mathbf{E}_{p-1}]$ *if*

$$|(\mathbf{E}_i + \mathbf{E}_j) - (\mathbf{E}_m + \mathbf{E}_n)| \leq \delta.$$

In the following derivations, we can take δ, which is a small threshold to tolerate numerical errors, to be zero.

As long as training loss is effectively zero, any parallelogram (i, j, m, n) should satisfy $i+j = m+n$. Suppose that this is not the case, i.e., suppose $\mathbf{E}_i + \mathbf{E}_j = \mathbf{E}_m + \mathbf{E}_n$ but $i + j \neq m + n$, then $\mathbf{Y}_{i+j} = \text{Dec}(\mathbf{E}_i + \mathbf{E}_j) = \text{Dec}(\mathbf{E}_m + \mathbf{E}_n) = \mathbf{Y}_{m+n}$ where the first and last equalities come from the zero training loss assumption. However, since $i + j \neq m + n$, we have $\mathbf{Y}_{i+j} \neq \mathbf{Y}_{n+m}$ (almost surely in the regression task), a contradiction.
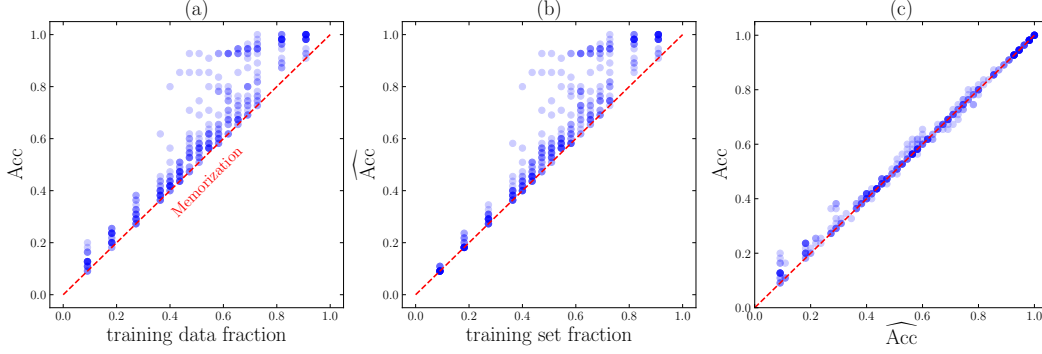
Figure 3: We compute accuracy (of the full dataset) either measured empirically Acc, or predicted from representation $\widehat{\text{Acc}}$. These two accuracies as a function of training data fraction are plotted in (a)(b), and their agreement is shown in (c).

It is convenient to define the permissible parallelogram set associated with a training dataset D ("permissible" means consistent with 100% training accuracy) as

$$P_0(D) = \{(i, j, m, n) | (i, j) \in D, (m, n) \in D, i + j = m + n\}. \tag{1}$$

For simplicity, we denote $P_0 \equiv P_0(D_0)$. Given a representation $\mathbf{R}$, we can check how many permissible parallelograms actually exist in $\mathbf{R}$ within error δ, so we define the parallelogram set corresponding to $\mathbf{R}$ as

$$P(\mathbf{R}, \delta) = \{(i, j, m, n) | (i, j, m, n) \in P_0, |(\mathbf{E}_i + \mathbf{E}_j) - (\mathbf{E}_m + \mathbf{E}_n)| \leq \delta\}. \tag{2}$$

For brevity we will write $P(\mathbf{R})$, suppressing the dependence on δ. We define the representation quality index (RQI) as

$$\text{RQI}(\mathbf{R}) = \frac{|P(\mathbf{R})|}{|P_0|} \in [0, 1]. \tag{3}$$

We will use the term *linear representation* or *linear structure* to refer to a representation whose embeddings are of the form $\mathbf{E}_k = \mathbf{a} + k\mathbf{b}$ ($k = 0, \cdots, p - 1; \mathbf{a}, \mathbf{b} \in \mathbb{R}^{d_{\text{in}}}$). A linear representation has $\text{RQI} = 1$, while a random representation (sampled from, say, a normal dstribution) has $\text{RQI} = 0$ with high probability.

Quantitatively, we denote the "predicted accuracy" $\widehat{\text{Acc}}$ as the accuracy achievable on the whole dataset given the representation $\mathbf{R}$ (see Appendix B for the full details). In Figure 3, we see that the predicted $\widehat{\text{Acc}}$ aligns well with the true accuracy Acc, establishing good evidence that representation leads to generalization. We use an example to illustrate the origin of generalization here. In the setup of Figure 2 (a), suppose the decoder can achieve zero training loss and $\mathbf{E}_6 + \mathbf{E}_8$ is a training sample hence $\text{Dec}(\mathbf{E}_6 + \mathbf{E}_8) = \mathbf{Y}_{14}$. At validation time, the decoder is tasked with predicting a validation sample $\mathbf{E}_5 + \mathbf{E}_9$. Since $(5, 9, 6, 8)$ forms a parallelogram such that $\mathbf{E}_5 + \mathbf{E}_9 = \mathbf{E}_6 + \mathbf{E}_8$, the decoder can predict the validation sample correctly because $\text{Dec}(\mathbf{E}_5 + \mathbf{E}_9) = \text{Dec}(\mathbf{E}_6 + \mathbf{E}_8) = \mathbf{Y}_{14}$.

3.2 The critical training set size

Suppose that we have an ideal model $\mathcal{M}^* = (\text{Dec}^*, \mathbf{R}^*)$ such that:[2]

- (1) $\mathcal{M}^*$ can achieve zero training loss;
- (2) $\mathcal{M}^*$ has an injective decoder, i.e., $\text{Dec}^*(\mathbf{x}_1) \neq \text{Dec}^*(\mathbf{x}_2)$ for any $\mathbf{x}_1 \neq \mathbf{x}_2$.

Then Proposition 1 provides a mechanism for the formation of parallelograms.

Proposition 1. *If a training set D contains two samples (i, j) and (m, n) with $i + j = m + n$, then $\mathcal{M}^*$ learns a representation $\mathbf{R}^*$ such that $\mathbf{E}_i + \mathbf{E}_j = \mathbf{E}_m + \mathbf{E}_n$, i.e., (i, j, m, n) forms a parallelogram.*

---

[2]One can verify a posteriori if a trained model $\mathcal{M}$ is close to being an ideal model $\mathcal{M}^*$. Please refer to Appendix C for details.
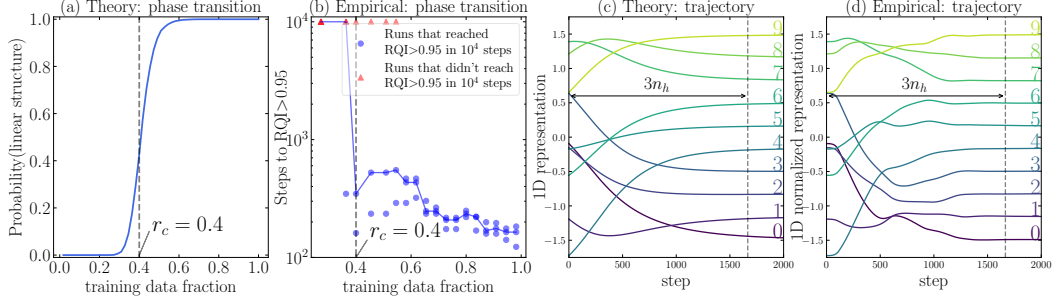
Figure 4: (a) The effective theory predicts a phase transition in the probability of obtaining a linear representation around $r_c = 0.4$. (b) Empirical results display a phase transition of RQI around $r_c = 0.4$, in agreement with the theory (the blue line shows the median of multiple random seeds). The evolution of 1D representations predicted by the effective theory or obtained from neural network training (shown in (c) and (d) respectively) agree creditably well.

Proof. Due to the zero training loss assumption, we have $\text{Dec}^*(\mathbf{E}_i + \mathbf{E}_j) = \mathbf{Y}_{i+j} = \mathbf{Y}_{m+n} = \text{Dec}^*(\mathbf{E}_m + \mathbf{E}_n)$. Then the injectivity of Dec^* implies $\mathbf{E}_i + \mathbf{E}_j = \mathbf{E}_m + \mathbf{E}_n$. $\qquad\square$

The dynamics of the trained embedding parameters are determined by various factors interacting in complex ways, for instance: the details of the decoder architecture, the optimizer hyperparameters, and the various kinds of implicit regularization induced by the training procedure. We will see that the dynamics of normalized quantities, namely, the normalized embeddings at time t, defined as $\tilde{\mathbf{E}}_k^{(t)} = \frac{\mathbf{E}_k^{(t)} - \mu_t}{\sigma_t}$, where $\mu_t = \frac{1}{p} \sum_k \mathbf{E}_k^{(t)}$ and $\sigma_t = \frac{1}{p} \sum_k |\mathbf{E}_k^{(t)} - \mu_t|^2$, can be qualitatively described by a simple effective loss. We will assume that the normalized embedding vectors obey a gradient flow for an effective loss function of the form

$$\frac{d\tilde{\mathbf{E}}_i}{dt} = -\frac{\partial \ell_{\text{eff}}}{\partial \tilde{\mathbf{E}}_i}, \tag{4}$$

$$\ell_{\text{eff}} = \frac{\ell_0}{Z_0}, \quad \ell_0 \equiv \sum_{(i,j,m,n) \in P_0(D)} |\tilde{\mathbf{E}}_i + \tilde{\mathbf{E}}_j - \tilde{\mathbf{E}}_m - \tilde{\mathbf{E}}_n|^2, \quad Z_0 \equiv \sum_k |\tilde{\mathbf{E}}_k|^2, \tag{5}$$

where $|\cdot|$ denotes Euclidean vector norm. Note that the embeddings do not collapse to the trivial solution $\mathbf{E}_0 = \cdots = \mathbf{E}_{p-1} = 0$ unless initialized as such, because two conserved quantities exist, as proven in Appendix D:

$$\mathbf{C} = \sum_k \mathbf{E}_k, \quad Z_0 = \sum_k |\mathbf{E}_k|^2. \tag{6}$$

We shall now use the effective dynamics to explain empirical observations such as the existence of a critical training set size for generalization.

Degeneracy of ground states (loss optima) We define ground states as those representations satisfying $\ell_{\text{eff}} = 0$, which requires the following linear equations to hold:

$$A(P) = \{\mathbf{E}_i + \mathbf{E}_j = \mathbf{E}_m + \mathbf{E}_n | (i,j,m,n) \in P\} \tag{7}$$

Since each embedding dimension obeys the same set of linear equations, we will assume, without loss of generality, that $d_{\text{in}} = 1$. The dimension of the null space of $A(P)$, denoted as n_0, is the number of degrees of freedom of the ground states. Given a set of parallelograms implied by a training dataset D, the nullity of $A(P(D))$ could be obtained by computing the singular values $0 \le \sigma_1 \le \cdots \le \sigma_p$. We always have $n_0 \ge 2$, i.e., $\sigma_1 = \sigma_2 = 0$ because the nullity of $A(P_0)$, the set of linear equations given by all possible parallelograms, is $\text{Nullity}(A(P_0)) = 2$ which can be attributed to two degrees of freedom (translation and scaling). If $n_0 = 2$, the representation is unique up to translations and scaling factors, and the embeddings have the form $\mathbf{E}_k = \mathbf{a} + k\mathbf{b}$. Otherwise, when $n_0 > 2$, the representation is not constrained enough such that all the embeddings lie on a line.

We present theoretical predictions alongside empirical results for addition ($p = 10$) in Figure 4. As shown in Figure 4 (a), our effective theory predicts that the probability that the training set implies a

5

unique linear structure (which would result in perfect generalization) depends on the training data fraction and has a phase transition around $r_c = 0.4$. Empirical results from training different models are shown in Figure 4(b). The number of steps to reach RQI > 0.95 is seen to have a phase transition at $r_c = 0.4$, agreeing with the proposed effective theory and with the empirical findings in [1].

Time towards the linear structure We define the Hessian matrix of ℓ_0 as

$$\mathbf{H}_{ij} = \frac{1}{Z_0}\frac{\partial^2 \ell_0}{\partial \mathbf{E}_i \partial \mathbf{E}_j},\tag{8}$$

Note that $\ell_0 = \frac{1}{2}\mathbf{R}^T\mathbf{H}\mathbf{R}$, $\mathbf{R} = [\mathbf{E}_0, \mathbf{E}_1, \cdots, \mathbf{E}_{p-1}]$, so the gradient descent is linear, i.e.,

$$\frac{d\mathbf{R}}{dt} = -\mathbf{H}\mathbf{R}.\tag{9}$$

If $\mathbf{H}$ has eigenvalues $\lambda_i = \sigma_i^2$ (sorted in increasing order) and eigenvectors $\bar{\mathbf{v}}_i$, and we have the initial condition $\mathbf{R}(t = 0) = a_i\bar{\mathbf{v}}_i$, then we have $\mathbf{R}(t) = a_i\bar{\mathbf{v}}_i e^{-\lambda_i t}$. The first two eigenvalues vanish and $t_h = 1/\lambda_3$ determines the timescale for the slowest component to decrease by a factor of e. When the step size is η, the corresponding number of steps is $n_h = t_h/\eta = 1/(\lambda_3\eta)$.

We verify the above analysis with empirical results. Figure 4 (c)(d) show the trajectories obtained from the effective theory and from neural network training, respectively. The 1D neural representation in Figure 4(d) are manually normalized to zero mean and unit variance. The two trajectories agree qualitatively, and it takes about $3n_h$ steps for two trajectories to converge to the linear structure. The quantitative differences might be due to the absence of the decoder in the effective theory, which assumes the decoder to be *adiabatic* in the sense that it takes infinitesimal step sizes.

Limitations of the effective theory While our theory defines an effective loss based on the Euclidean distance between embeddings $\mathbf{E}_i + \mathbf{E}_j$ and $\mathbf{E}_n + \mathbf{E}_m$, one could imagine generalizing the theory to define a broader notion of parallogram given by some other metric on the representation space. For instance, if we have a decoder like in Figure 2(c) then the distance between distinct representations within the same "pizza slice" is low, meaning that representations arranged not in parallelograms w.r.t. the Euclidean metric may be parallelograms with respect to the metric defined by the decoder.

4 Delayed Generalization: A Phase Diagram

So far, we have (1) observed empirically that generalization on algorithmic datasets corresponds with the emergence of well-structured representations, (2) defined a notion of representation quality in a toy setting and shown that it predicts generalization, and (3) developed an effective theory to describe the learning dynamics of the representations in the same toy setting. We now study how optimizer hyperparameters affect high-level learning performance. In particular, we develop phase diagrams for how learning performance depend on the representation learning rate, decoder learning rate and the decoder weight decay. These parameters are of interest since they most explicitly regulate a kind of *competition* between the encoder and decoder, as we elaborate below.

4.1 Phase diagram of a toy model

Training details We update the representation and the decoder with different optimizers. For the 1D embeddings, we use the Adam optimizer with learning rate $[10^{-4}, 10^{-2}]$ and zero weight decay. For the decoder, we use an AdamW optimizer with the learning rate in $[10^{-4}, 10^{-2}]$ and the weight decay in $[0, 10]$ (regression) or $[0, 20]$ (classification). For training/validation spliting, we choose 45/10 for non-modulo addition ($p = 10$) and 24/12 for the permutation group S_3. We hard-code addition or matrix multiplication (details in Appendix F) in the decoder for the addition group and the permutation group, respectively.

For each choice of learning rate and weight decay, we compute the number of steps to reach high (90%) training/validation accuracy The 2D plane is split into four phases: *comprehension*, *grokking*, *memorization* and *confusion*, defined in Table 1. Both comprehension and grokking are able to generalize (in the "Goldilocks zone"), although the grokking phase has delayed generalization. Memorization is also called overfitting, and confusion means failure to even memorize training data. Figure 5 shows the phase diagrams for the addition group and the permutation group. They display quite rich phenomenon.

6

(a) Addition, regression (b) Addition, regression

(c) Addition, classification (d) Permutation, regression
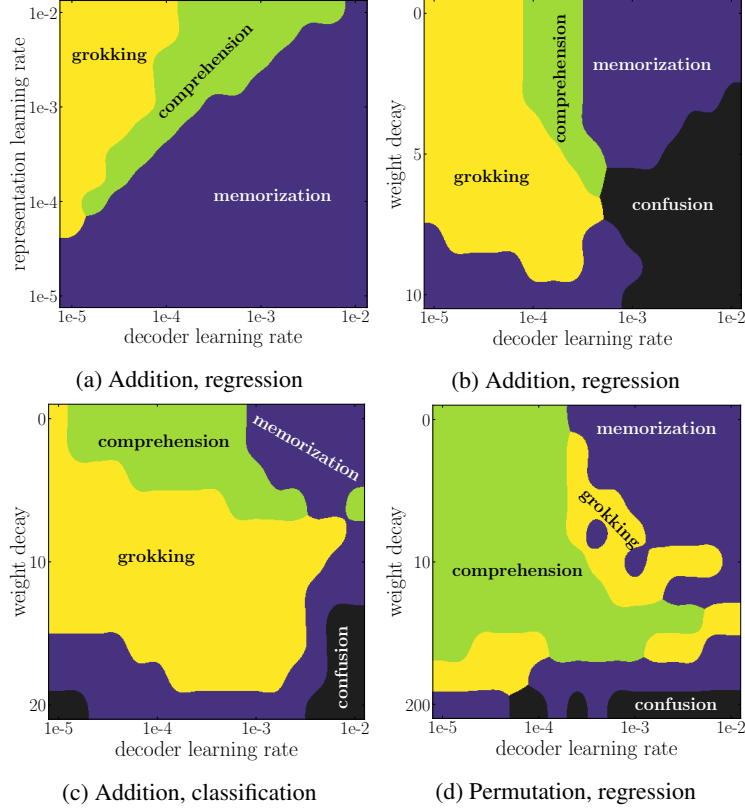
Figure 5: Phase diagrams of learning for the addition group and the permutation group. (a) shows the competition between representation and decoder. (b)(c)(d): each phase diagram contains four phases: comprehension, grokking, memorization and confusion, defined in Table 1. Grokking is sandwiched between comprehension and memorization.

Table 1: Definitions of the four phases of learning

| | criteria | | |
Phase	training acc > 90% within 10^5 steps	validation acc > 90% within 10^5 steps	step(validation acc>90%) $-$step(training acc>90%)<10^3
Comprehension	Yes	Yes	Yes
Grokking	Yes	Yes	No
Memorization	Yes	No	Not Applicable
Confusion	No	No	Not Applicable

Competition between representation learning and decoder overfitting In the regression setup of the addition dataset, we show how the competition between representation learning rate and decoder learning rate lead to different learning phases in Figure 5 (a). As expected, a fast decoder coupled with slow representation learning (bottom right) lead to memorization. In the opposite extreme, although an extremely slow decoder coupled with fast representation learning (top left) will generalize in the end, the generalization time is long due to the inefficient decoder training. The ideal phase (comprehension) requires representation learning to be faster, but not too much, than the decoder.

Drawing from an analogy to physical systems, one can think of embedding vectors as a group of particles. In our effective theory from Section 3.2, internal particle interactions promote the formation of structure, while external forces inhibit it. The decoder plays the role of an environment exerting external forces on the embeddings. If the decoder changes slowly/fast, the magnitude of the external forces are small/large, leading to better/worse representations.

7

Universality of phase diagrams We fix representation learning to be 10^{-3} and sweep instead decoder weight decay in Figure 5 (b)(c)(d). The phase diagrams correspond to addition regression (b), addition classification (c) and permutation regression (d), respectively. Common phenomenon emerge from these different tasks: (i) they all include four phases; (ii) The top right corner (a fast and capable decoder) is the memorization phase; (iii) the bottom right corner (a fast and simple decoder) is the confusion phase; (iv) grokking is sandwiched between comprehension and memorization, which seems to imply that it is an undesirable phase that stems from improperly tuned hyperparameters.

Due to limited space, we include phase diagrams of other hyperparameters (batch size, intialization scale and weight decay on representation) in Appendix E.

4.2 Beyond the toy model

We conjecture that many of the principles which we saw dictate the training dynamics in the toy model also apply more generally. Below, we will see how our framework generalizes to transformer architectures for the task of addition modulo p, a minimal reproducible example of the original grokking paper [1].

We first encode $p = 53$ integers into 256D learnable embeddings, then pass two integers to a decoder-only transformer architecture. For simplicity, we do not encode the operation symbols here. The outputs from the last layer are concatenated and passed to a linear layer for classification. Training both the encoder and the decoder with the same optimizer (i.e., with the same hyperparameters) leads to the grokking phenomenon. Generalization appears much earlier once we lower the effective decoder capacity with weight decay (full phase diagram in Figure 6).
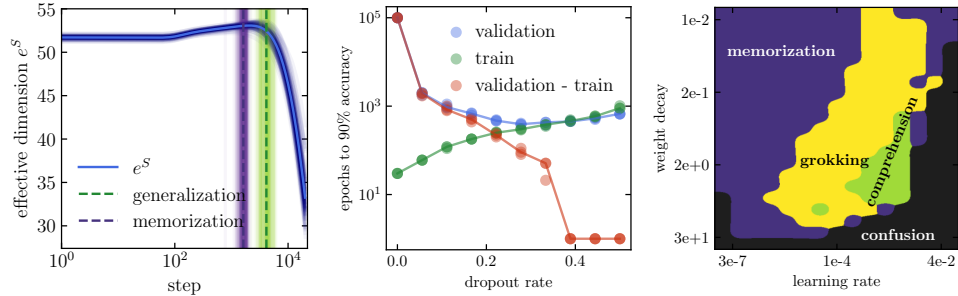

Figure 6: Left: Evolution of the effective dimension of the embeddings (defined as the exponential of the entropy) during training and evaluated over 100 seeds. Center: Effect of dropout on speeding up generalization. Right: Phase diagram of the transformer architecture. A scan is performed over the weight decay and learning rate of the decoder while the learning rate of the embeddings is kept fixed at 10^{-3} (with zero weight decay).

Early on, the model is able to perfectly fit the training set while having no generalization. We study the embeddings at different training times and find that neither PCA (shown in Figure 1) nor t-SNE (not shown here) reveal any structure. Eventually, validation accuracy starts to increase, and perfect generalization coincides with the PCA projecting the embeddings into a circle in 2D. Of course, no choice of dimensionality reduction is guaranteed to find any structure, and thus, it is challenging to show explicitly that generalization only occurs when a structure exists. Nevertheless, the fact that, when coupled with the implicit regularization of the optimizer for sparse solutions, such a clear structure appears in a simple PCA so quickly at generalization time suggests that our analysis in the toy setting is applicable here as well. This is also seen in the evolution of the entropy of the explained variance ratio in the PCA of the embeddings (defined as $S = -\sum_i \sigma_i \log \sigma_i$ where σ_i is the fractional variance explained by the ith principal component). As seen in Figure 6, the entropy increases up to generalization time then decreases drastically afterwards which would be consistent with the conjecture that generalization occurs when a low-dimensional structure is discovered. The decoder then primarily relies on the information in this low-dimensional manifold and essentially "prunes" the rest of the high-dimensional embedding space.

In Figure 6 (right), we show a comparable phase diagram to Figure 5 evaluated now in the transformer setting. Note that, as opposed to the setting in [1], weight decay has only been applied to the decoder and not to the embedding layer. Contrary to the toy model, a certain amount of weight decay proves

8

beneficial to generalization and speeds it up significantly. We conjecture that this difference comes from the different embedding dimensions. With a highly over-parameterized setting, a non-zero weight decay gives a crucial incentive to reduce complexity in the decoder and help generalize in fewer steps. This is subject to further investigation. We also explore the effect of dropout layers in the decoder blocks of the transformer. With a significant dropout rate, the generalization time can be brought down to under 10^3 steps and the grokking phenomenon vanishes completely. The overall trend suggests that constraining the decoder with the same tools used to avoid overfitting reduces generalization time and can avoid the grokking phenomenon.

5 Related work

We are not aware of other formal attempts to understand grokking, though some authors have provided speculative, informal accounts. For instance, [2] describes training as a random walk over parameters, and draws connections to Solomonoff induction. Separately, [3] suggest that simpler functions which generalize achieve lower loss than complicated functions which memorize, providing a training signal encouraging generalization. More broadly, our work is related to the following research directions:

Learning mathematical structures [4] trains a neural network to learn arithmetic operation from pictures of digits, but they do not observe grokking due to their abundant training data. Beyond arithmetic relations, machine learning has been applied to learn other mathematical structures, including geometry [5], knot theory [6] and group theory [7].

Double descent Grokking is somewhat reminiscent of the phenomena of "epoch-wise" *double descent* [8], where generalization can improve after a period of overfitting. [9] find that regularization can mitigate double descent, similar perhaps to how weight decay influences grokking.

Representation learning Representation learning lies at the core of machine learning [10–13]. Representation quality is usually measured by (perhaps vague) semantic meanings or performance on downstream tasks. In our study, the simplicity of arithmetic datasets allows us to define representation quality and study evolution of representations in a quantitative way.

Physics of learning Physics-inspired tools have been proved useful to understand machine learning from a theoretical perspective. These tools include effective theories [14, 15], conservation laws [16] and free energy principle [17]. However, none of these works aim to develop an effective theory of representation learning.

We have argued that grokking occurs when models learn a particularly structured representation of their inputs. This connects a low-level understanding of the internal operation of our models with a high-level phenomenon: an increase in validation accuracy. In a recent work, researchers at Anthropic [18], connect a sudden decrease in loss during training with the emergence of *induction heads* within their models. They analogize their work to *statistical physics*, since it bridges a "microscopic", mechanistic understanding of networks with "macroscopic" facts about overall model performance. We view our work in the same vein, but we make the additional contribution that we develop, in a toy setting, an **analytic model of low-level representation learning dynamics**.

6 Conclusion

We have shown how, in both toy models and more general settings, that representation enables generalization when it reflects structure in the data. We developed an effective theory of representation learning dynamics (in a toy setting) which predicts the critical dependence of learning on the training data fraction. We then presented phase transitions between comprehension, grokking, memorization and confusion where the learning "phase" depends on the decoder capacity and learning rate in decoder-only architectures. While we have mostly focused on a toy model, we find preliminary evidence that our results generalize to the setting of [1].

Our work can be viewed as a step towards a *statistical physics of deep learning*, a theory which connects the "microphysics" of low-level network dynamics with the "thermodynamics" of high-level model behavior. We view the application of theoretical tools from physics, such as effective theories [19], to be a rich area for further work. The broader impact of such work, if successful, would be to make models more transparent and predictable [18, 20, 21], crucial to the task of ensuring the safety of advanced AI systems.

References

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

[2] Beren Millidge. Grokking 'grokking'. https://beren.io/2022-01-11-Grokking-Grokking/, 2022.

[3] Rohin Shah. Alignment Newsletter #159. https://www.alignmentforum.org/posts/zvWqPmQasssaAWkrj/an-159-building-agents-that-know-how-to-experiment-by#DEEP_LEARNING_, 2021.

[4] Yedid Hoshen and Shmuel Peleg. Visual learning of arithmetic operation. In *AAAI*, 2016.

[5] Yang-Hui He. Machine-learning mathematical structures. *arXiv preprint arXiv:2101.06317*, 2021.

[6] Sergei Gukov, James Halverson, Fabian Ruehle, and Piotr Sułkowski. Learning to unknot. *Machine Learning: Science and Technology*, 2(2):025035, 2021.

[7] Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600(7887):70–74, 2021.

[8] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

[9] Preetum Nakkiran, Prayaag Venkat, Sham Kakade, and Tengyu Ma. Optimal regularization can mitigate double descent. *arXiv preprint arXiv:2003.01897*, 2020.

[10] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

[11] Yassine Ouali, Céline Hudelot, and Myriam Tami. An overview of deep semi-supervised learning. *arXiv preprint arXiv:2006.05278*, 2020.

[12] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

[13] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934, 2020.

[14] James Halverson, Anindita Maiti, and Keegan Stoner. Neural networks and quantum field theory. *Machine Learning: Science and Technology*, 2(3):035002, 2021.

[15] Daniel A Roberts, Sho Yaida, and Boris Hanin. The principles of deep learning theory. *arXiv preprint arXiv:2106.10165*, 2021.

[16] Daniel Kunin, Javier Sagastuy-Brena, Surya Ganguli, Daniel LK Yamins, and Hidenori Tanaka. Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics. *arXiv preprint arXiv:2012.04728*, 2020.

[17] Yansong Gao and Pratik Chaudhari. A free-energy principle for representation learning. In *International Conference on Machine Learning*, pages 3367–3376. PMLR, 2020.

[18] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[19] Daniel A. Roberts, Sho Yaida, and Boris Hanin. *The Principles of Deep Learning Theory*. Cambridge University Press, 2022. https://deeplearningtheory.com.

[20] Deep Ganguli, Danny Hernandez, Liane Lovitt, Nova DasSarma, Tom Henighan, Andy Jones, Nicholas Joseph, Jackson Kernion, Ben Mann, Amanda Askell, et al. Predictability and surprise in large generative models. *arXiv preprint arXiv:2202.07785*, 2022.

[21] Jacob Steinhardt. Future ML Systems Will Be Qualitatively Different. https://www.lesswrong.com/s/4aARF2ZoBpFZAhbbe/p/pZaPhGg2hmmPwByHc, 2022.

[22] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Checklist

1. For all authors...

    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We lay out our "microscopic" analysis and effective theory in Section 3. For a toy model, we establish the connection between well-structured representations and generalization in 3.1 and describe an effective theory of the learning dynamics of these representations in 3.2, which we find predicts the critical training set size. We perform our "macroscopic" analysis of phase diagrams in Section 4.

    (b) Did you describe the limitations of your work? [Yes] Limitations are described throughout the text where applicable, and also explicitly in the Appendix.

    (c) Did you discuss any potential negative societal impacts of your work? [No] Our work is theoretical. While some of our insights may eventually lead to more capable models, which could be misused, we don't think our work introduces any specific risks for misuse. On the other hand, work like ours trying to gain understanding of deep learning phenomena could prove useful for AI safety.

    (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

    (a) Did you state the full set of assumptions of all theoretical results? [Yes] The theory we develop in this paper is, in the language of physics, an *effective* theory. It is simpler than the full setup where grokking was initially observed, but this simplicity allows us to develop a precise mathematical description of representation quality and learning dynamics. We make clear that our theoretical analysis is specific to our toy model.

    (b) Did you include complete proofs of all theoretical results? [Yes] We have relatively few theoretical results, contained within our discussion of our effective theory on the toy model 3. We provided short proofs of those results.

3. If you ran experiments...

    (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In the final (non-anonymized) version of the paper we will include a working link to a public repository containing code for reproducing our results and figures.

    (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We discuss hyperparameters and data splits in Section 4.

11

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] It is difficult to visualize uncertainty for our phase diagrams.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] Our experiments can be reproduced on a single GPU within a week. Since this is not a substantial amount of compute, and our study is about gaining scientific understanding rather than achieving SOTA, we did not deem it important to report this in the text.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] Some of our code was based off of code released by the authors of the original paper on grokking [1].

(b) Did you mention the license of the assets? [No] The code for reproducing our experiments will be made available on GitHub under the MIT license. We release no other assets.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] In the full, non-anonymized publication, a working link will be included to the code for reproducing our experiments.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Applicability of our toy setting

In the main paper, we focused on the toy setting with (1) the addition dataset and (2) the addition operation hard coded in the decoder. Although both simplifications appear to have quite limited applicability, we argue below that the analysis of the toy setting can actually apply to all Abelian groups.

The addition dataset is the building block of all Abelian groups A cyclic group is a group that is generated by a single element. A finite cyclic group with order n is $C_n = \{e, g, g^2, \cdots, g^{n-1}\}$ where e is the identify element and g is the generator and $g^i = g^j$ whenever $i = j \pmod{n}$. The modulo addition and $\{0, 1, \cdots, n-1\}$ form a cyclic group with $e = 0$ and g can be any number q coprime to n such that $(q, n) = 1$. Since algorithmic datasets contain only symbolic but no arithmetic information, the datasets of modulo addition could apply to all other cyclic groups, e.g., modulo multiplication and discrete rotation groups in 2D.

Although not all Abelian groups are cyclic, a finite Abelian group G can be always decomposed into a direct product of k cyclic groups $G = C_{n_1} \times C_{n_2} \cdots C_{n_k}$. So after training k neural networks with each handling one cyclic group separately, it is easy to construct a larger neural network that handles the whole Abelian group.

The addition operation is valid for all Abelian groups It is proved in [22] that for a permutation invariant function $f(x_1, x_2, \cdots, x_n)$, there exists ρ and ϕ such that

$$f(x_1, x_2, \cdots, x_n) = \rho[\sum_{i=1}^{n} \phi(x_i)], \tag{10}$$

or $f(x_1, x_2) = \rho(\phi(x_1) + \phi(x_2))$ for $n = 2$. Notice that $\phi(x_i)$ corresponds to the embedding vector $\mathbf{E}_i$, ρ corresponds to the decoder. The addition operator naturally emerges from the commutativity of the operator, not restricting the operator itself to be addition. For example, multiplication of two numbers x_1 and x_2 can be written as $x_1 x_2 = \exp(\ln(x_1) + \ln(x_2))$ where $\rho(x) = \exp(x)$ and $\phi(x) = \ln(x)$.
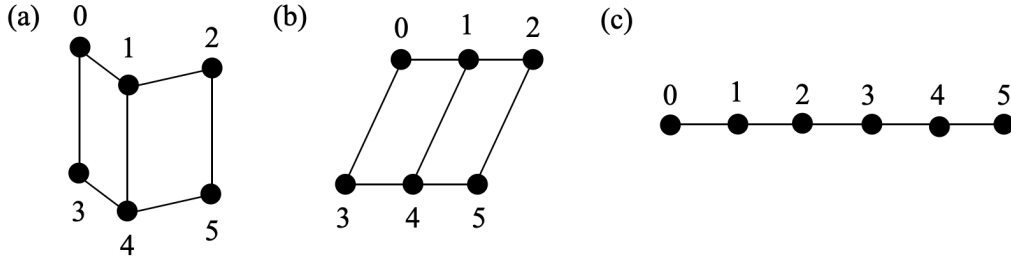

Figure 7: As we include more data in the training set, the (ideal) model have a more structured representation, from (a) to (b) to (c).

B Definition of $\widehat{\mathrm{Acc}}$

Given a training set D and a representation $\mathbf{R}$, if (i, j) is a validation sample, can the neural network correctly predict its output, i.e., $\mathrm{Dec}(\mathbf{E}_i + \mathbf{E}_j) = \mathbf{Y}_{i+j}$? Since neural network has never seen (i, j) in the training set, one possible mechanism of induction is through

$$\mathrm{Dec}(\mathbf{E}_i + \mathbf{E}_j) = \mathrm{Dec}(\mathbf{E}_m + \mathbf{E}_n) = \mathbf{Y}_{m+n}(= \mathbf{Y}_{i+j}). \tag{11}$$

The first equality $\mathrm{Dec}(\mathbf{E}_i + \mathbf{E}_j) = \mathrm{Dec}(\mathbf{E}_m + \mathbf{E}_n)$ holds only when $\mathbf{E}_i + \mathbf{E}_j = \mathbf{E}_m + \mathbf{E}_n$ (i.e., (i, j, m, n) is a parallelogram). The second equality $\mathrm{Dec}(\mathbf{E}_m + \mathbf{E}_n) = \mathbf{Y}_{m+n}$, holds when (m, n) in training set, i.e., $(m, n) \in D$, under the zero training loss assumption. Rigorously, given a training

set D and a parallelogram set P (which can be calculated from $\mathbf{R}$), we collect all zero loss samples in an *augmented* training set $\overline{D}$

$$\overline{D}(D, P) = D \bigcup \{(i, j) | \exists (m, n) \in D, (i, j, m, n) \in P\}. \tag{12}$$

Keeping D fixed, a larger P would probably produce a larger $\overline{D}$, i.e., if $P_1 \subseteq P_2$, then $\overline{D}(D, P_1) \subseteq \overline{D}(P, P_2)$, which is why in Eq. (3) our defined RQI $\propto |P|$ gets its name "representation quality index", because higher RQI normally means better generalization. Finally, the expected accuracy from a dataset D and a parallelogram set P is:

$$\widehat{\text{Acc}} = \frac{|\overline{D}(D, P)|}{|D_0|}, \tag{13}$$

which is the estimated accuracy (of the full dataset), and $P = P(\mathbf{R})$ is defined on the representation after training. On the other hand, accuracy Acc can be accessed empirically from trained neural network. We verified $\text{Acc} \approx \widehat{\text{Acc}}$ in a toy setup (addition dataset $p = 10$, 1D embedding space, hard code addition), as shown in FIG. 3 (c). FIG. 3 (a)(b) show Acc and $\widehat{\text{Acc}}$ as a function of training set ratio, with each dot corresponding to a different random seed. The dashed red diagonal corresponds to memorization of the training set, and the vertical gap refers to generalization.

Although the agreement is good for 1D embedding space, we do not expect such agreement can trivially extend to high dimensional embedding space. In high dimensions, our definition of RQI is too restrictive. For example, suppose we have an embedding space with N dimensions. Although the representation may form a linear structure in the first dimension, the representation can be arbitrary in other $N - 1$ dimensions, leading to RQI ≈ 0. However, the model may still generalize well if the decoder learns to keep only the useful information (the 1st dimension) and drop all other useless information (other 31 dimensions). It would be interesting to investigate how to define an RQI that takes into account the role of decoder in future works.

C The gap of a realistic model $\mathcal{M}$ and the ideal model $\mathcal{M}^*$

Realistic models $\mathcal{M}$ usually form fewer number of parallelograms than ideal models $\mathcal{M}^*$. In this section, we analyze the properties of ideal models and calculated ideal RQI and ideal accuracy, which set upper bounds for empirical RQI and accuracy. The upper bound relations are verified via numerical experiments in Fig. 8.

Similar to Eq. (12) where some validation samples can be derived from training samples, we demonstrate how *implicit parallelograms* can be 'derived' from explicit ones in $P_0(D)$. The so-called derivation follows a simple geometric argument that: if $A_1 B_1$ is equal and parallel to $A_2 B_2$, and $A_2 B_2$ is equal and parallel to $A_3 B_3$, then we can deduce that $A_1 B_1$ is equal and parallel to $A_3 B_3$ (hence (A_1, B_2, A_2, B_1) is a parallelogram).

Recall that a parallelogram (i, j, m, n) is equivalent to $\mathbf{E}_i + \mathbf{E}_j = \mathbf{E}_m + \mathbf{E}_n$ ($*$). So we are equivalently asking if equation ($*$) can be expressed as a linear combination of equations in $A(P_0(D))$. If yes, then ($*$) is dependent on $A(P_0(D))$ (defined in Eq. (7)), i.e., $A(P_0(D))$ and $A(P_0(D) \bigcup (i, j, m, n))$ should have the same rank. We augment $P_0(D)$ by adding implicit parallelograms, and denote the augmented parallelogram set as

$$P(D) = P_0(D) \bigcup \{q \equiv (i, j, m, n) | q \in P_0, \text{rank}(A(P_0(D))) = \text{rank}(A(P_0(D) \bigcup q))\}. \tag{14}$$

We need to emphasize that an assumption behind Eq. (14) is that we have an ideal model $\mathcal{M}^*$. When the model is not ideal, e.g., when the injectivity of the encoder breaks down, fewer parallelograms are expected to form, i.e.,

$$P(R) \subseteq P(D). \tag{15}$$

The inequality is saying, whenever a parallelogram is formed in the representation after training, the reason is hidden in the training set. This is not a strict argument, but rather a belief that today's neural networks can only copy what datasets (explicitly or implicitly) tell it to do, without any autonomous creativity or intelligence. For simplicity we call this belief *Alexander Principle*. In very rare cases when something lucky happens (e.g., neural networks are initialized at approximate correct weights), Alexander principle may be violated. Alexander principle sets an upper bound for RQI:

$$\text{RQI}(R) \leq \frac{|P(D)|}{|P_0|} \equiv \overline{\text{RQI}}, \tag{16}$$

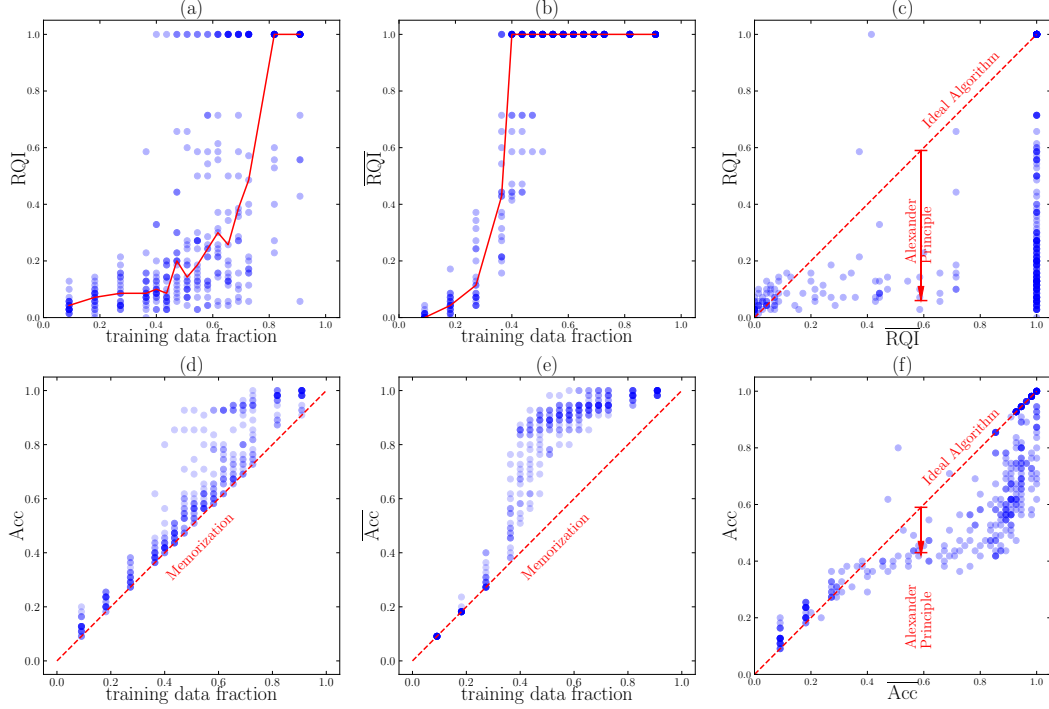Figure 8: We compare RQI and Acc for an ideal algorithm (with bar) and a realistic algorithm (without bar). In (a)(b)(d)(e), four quantities (RQI, $\overline{\text{RQI}}$, Acc, $\overline{\text{Acc}}$) as functions of training data fraction are shown. In (c)(f), RQI and Acc of the ideal algorithm sets upper bounds for those of the realistic algorithm.

and sets an upper bound for $\widehat{\text{Acc}}$:

$$\widehat{\text{Acc}} \equiv \widehat{\text{Acc}}(D, P(R)) \leq \widehat{\text{Acc}}(D, P(D)) \equiv \overline{\widehat{\text{Acc}}}, \tag{17}$$

In Figure 8 (c)(f), we verify Eq. (16) and Eq. (17). We choose $\delta = 0.01$ to compute RQI(R,δ). We find the trained models are usually far from being ideal, although we already include a few useful tricks proposed in Section 4 to enhance representation learning. It would be an interesting future direction to develop better algorithms so that the gap due to Alexander principle can be reduced or even closed. In Figure 8 (a)(b)(d)(e), four quantities (RQI, $\overline{\text{RQI}}$, Acc, $\overline{\text{Acc}}$) as functions of the training data fraction are shown, each dot corresponding to one random seed. It is interesting to note that it is possible to have $\overline{\text{RQI}} = 1$ only with $< 40\%$ training data, i.e., $55 \times 0.4 = 22$ samples, agreeing with our observation in Section 3.

Realistic representations Suppose an ideal model $\mathcal{M}^*$ and a realistic model $\mathcal{M}$ which train on the training set D give the representation R^* and R, respectively. What is the relationship between R and R_*? Due to the Alexander principle we know $P(R) \subseteq P(D) = P(R^*)$. This means R^* has more parallelograms than R, hence R^* has fewer degrees of freedom than R.

We illustrate with the toy case $p = 4$. The whole dataset contains $p(p+1)/2 = 10$ samples, i.e.,

$$D_0 = \{(0,0), (0,1), (0,2), (0,3), (1,1), (1,2), (1,3), (2,2), (2,3), (3,3)\}. \tag{18}$$

The parallelogram set contains only three elements, i.e.,

$$P_0 = \{(0,1,1,2), (0,1,2,3), (1,2,2,3)\}, \tag{19}$$

Or equivalently the equation set

$$A_0 = \{\text{A1} : \mathbf{E}_0 + \mathbf{E}_2 = 2\mathbf{E}_1, \text{A2} : \mathbf{E}_0 + \mathbf{E}_3 = \mathbf{E}_1 + \mathbf{E}_2, \text{A3} : \mathbf{E}_1 + \mathbf{E}_3 = 2\mathbf{E}_2\}. \tag{20}$$

Pictorially, we can split all possible subsets $\{A|A \subseteq A_0\}$ into different levels, each level defined by $|A|$ (the number of elements). A subset A_1 in the ith level points an direct arrow to another subset
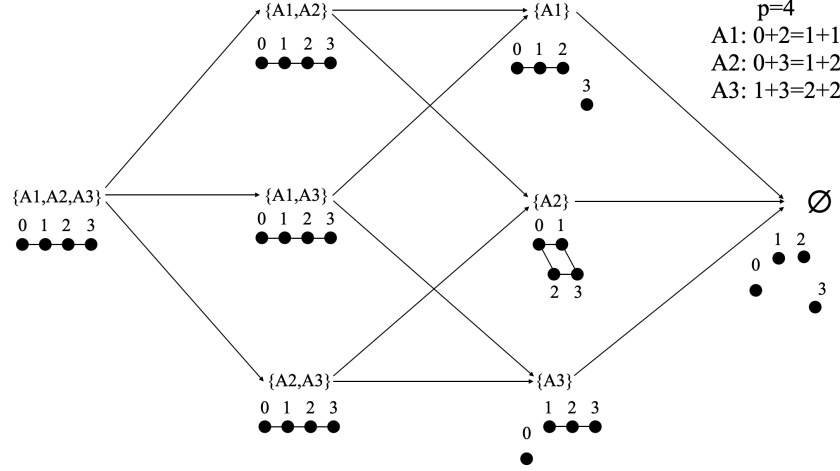
Figure 9: $p = 4$ case. Equation set A (or geometrically, representation) has a hierarchy: $a \rightarrow b$ means a is a parent of b, and b is a child of a. A realistic model can only generate representations that are descendants of the representation generated by an ideal model.
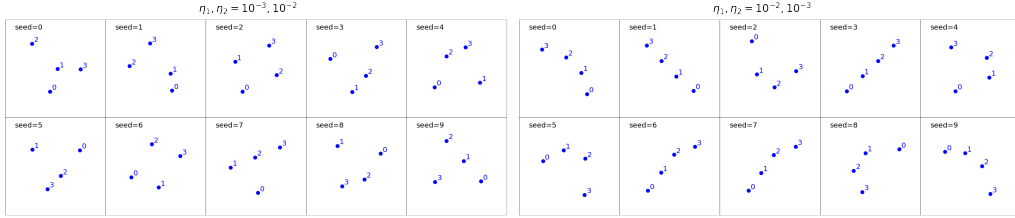

Figure 10: $p = 4$ case. Representations obtained from training neural networks are displayed. η_1 and η_2 are learning rates of the representation and the decoder, respectively. As described in the main text, $(\eta_1, \eta_2) = (10^{-2}, 10^{-3})$ (right) is more ideal than $(\eta_1, \eta_2) = (10^{-3}, 10^{-2})$ (left), thus producing representations containing more parallelograms.

A_2 in the $(i + 1)^{\text{th}}$ level if $A_2 \subset A_1$, and we say A_2 is a child of A_1, and A_1 is a parent of A_2. Each subset A can determine a representation R with $n(A)$ degrees of freedom. So R should be a descendant of R_*, and $n(R_*) \leq n(R)$. Numerically, $n(A)$ is equal to the dimension of the null space of A.

Suppose we have a training set

$$D = \{(0, 2), (1, 1), (0, 3), (1, 2), (1, 3), (2, 2)\}, \tag{21}$$

and correspondingly $P(D) = P_0, A(P) = A_0$. So an ideal model $\mathcal{M}_*$ will have the linear structure $\mathbf{E}_k = \mathbf{a} + k\mathbf{b}$ (see Figure 9 leftmost). However, a realistic model $\mathcal{M}$ may produce any descendants of the linear structure, depending on various hyperparameters and even random seeds.

In Figure 10, we show our algorithms actually generates all possible representations. We have two settings: (1) fast decoder $(\eta_1, \eta_2) = (10^{-3}, 10^{-2})$ (Figure 10 left), and (2) relatively slow decoder $(\eta_1, \eta_2) = (10^{-2}, 10^{-3})$ (Figure 10) right). The relatively slow decoder produces better representations (in the sense of higher RQI) than a fast decoder, agreeing with our observation in Section 4.

D Conservation laws of the effective theory

Recall that the effective loss function

$$\ell_{\text{eff}} = \frac{\ell_0}{Z_0}, \quad \ell_0 \equiv \sum_{(i,j,m,n) \in P_0(D)} |\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n|^2, \quad Z_0 \equiv \sum_k |\mathbf{E}_k|^2 \tag{22}$$

16

where ℓ_0 and Z_0 are both quadratic functions of $R = \{\mathbf{E}_0, \cdots, \mathbf{E}_{p-1}\}$, and $\ell_{\text{eff}} = 0$ remains zero under rescaling and translation $\mathbf{E}'_i = a\mathbf{E}_i + \mathbf{b}$. The representation vector $\mathbf{E}_i$ evolves according to the gradient descent

$$\frac{d\mathbf{E}_i}{dt} = -\frac{\partial \ell_{\text{eff}}}{\partial \mathbf{E}_i}. \tag{23}$$

We will prove the following two quantities are conserved:

$$\mathbf{C} = \sum_k \mathbf{E}_k, \quad Z_0 = \sum_k |\mathbf{E}_k|^2 \tag{24}$$

Eq. (22) and Eq. (23) give

$$\frac{d\mathbf{E}_i}{dt} = -\frac{\ell_{\text{eff}}}{\partial \mathbf{E}_i} = -\frac{\partial(\frac{\ell_0}{Z_0})}{\partial \mathbf{E}_i} = -\frac{1}{Z_0}\frac{\partial \ell_0}{\partial \mathbf{E}_i} + \frac{\ell_0}{Z_0^2}\frac{\partial Z_0}{\partial \mathbf{E}_i}. \tag{25}$$

Then

$$\frac{dZ_0}{dt} = 2\sum_i \mathbf{E}_k \cdot \frac{d\mathbf{E}_k}{dt} \tag{26}$$

$$= \frac{2}{Z_0^2}\sum_i \mathbf{E}_i \cdot (-Z_0\frac{\partial \ell_0}{\partial \mathbf{E}_k} + 2\ell_0\mathbf{E}_k)$$

$$= \frac{2}{Z_0}(-\sum_k \frac{\partial \ell_0}{\partial \mathbf{E}_k} \cdot \mathbf{E}_k + 2\ell_0)$$

$$= 0.$$

where the last equation uses the fact that

$$\sum_k \frac{\partial \ell_0}{\partial \mathbf{E}_k} \cdot \mathbf{E}_k = 2\sum_k \sum_{(i,j,m,n)\in P_0(D)} (\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n)(\delta_{ik} + \delta_{jk} - \delta_{mk} - \delta_{nk}) \cdot \mathbf{E}_k$$

$$= 2\sum_{(i,j,m,n)\in P_0(D)} (\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n)\sum_k (\delta_{ik} + \delta_{jk} - \delta_{mk} - \delta_{nk}) \cdot \mathbf{E}_k$$

$$= \sum_{(i,j,m,n)\in P_0(D)} (\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n) \cdot (\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n)$$

$$= 2\ell_0$$

The conservation of Z_0 prohibits the representation from collapsing to zero. Now that we have demonstrated that Z_0 is a conserved quantity, we can also show

$$\frac{d\mathbf{C}}{dt} = \sum_k \frac{d\mathbf{E}_k}{dt} \tag{27}$$

$$= -\frac{1}{Z_0}\sum_k \frac{\partial \ell_0}{\partial \mathbf{E}_k}$$

$$= -\frac{2}{Z_0}\sum_k \sum_{(i,j,m,n)\in P_0(D)} (\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n)(\delta_{ik} + \delta_{jk} - \delta_{mk} - \delta_{nk})$$

$$= \mathbf{0}.$$

The last equality holds because the two summations can be swapped and $\sum_k (\delta_{ik} + \delta_{jk} - \delta_{mk} - \delta_{nk}) = 0$.

E More phase diagrams of the toy setup

We study another three hyperparameters in the toy setup by showing phase diagrams similar to Figure 5. The toy setup is: (1) addition without modulo ($p = 10$); (2) training/validation is split into 45/10; (3) hard code addition; (4) 1D embedding. In the following experiments, the decoder is an
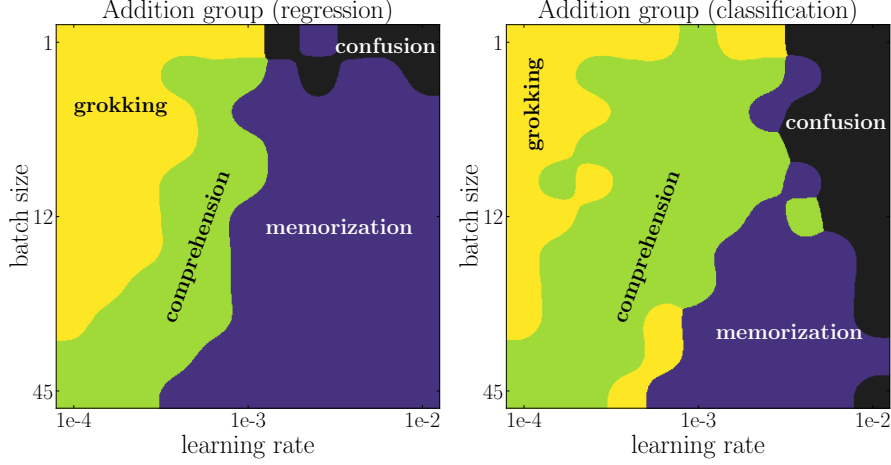
17

Figure 11: Phase diagrams of decoder learning rate (x axis) and batch size (y axis) for the addition group (left: regression; right: classification). Small decoder leanrning rate and large batch size (bottom left) lead to comprehension.

MLP with size 1-200-200-30. The representation and the encoder are optimized with AdamW with different hyperparameters. The learning rate of the representation is 10^{-3}. We sweep the learning rate of the decoder in range $[10^{-4}, 10^{-2}]$ as the x axis, and sweep another hyperparameter as the y axis. By default, we use full batch size 45, initialization scale $s = 1$ and zero weight decay of representation.

Batch size controls the amount of noise in the training dynamics. In Figure 11, the grokking region appears at the top left of the phase diagram (small decoder learning rate and small batch size). However, large batch size (with small learning rate) leads to comprehension, implying that smaller batch size seems harmful. This makes sense since to get crystals (good structures) in experiments, one needs a freezer which gradually decreases temperature, rather than something perturbing the system with noise.

Initialization scale controls distances among embedding vectors at initialization. We initialize components of embedding vectors from independent uniform distribution $U[-s/2, s/2]$ where s is called the initialization scale. Shown in Figure 12, it is beneficial to use a smaller initialization scale. This agrees with the physical intuition that closer particles are more likely to interact and form structures. For example, the distances among molecules in ice are much smaller than distances in gas.

Representation weight decay controls the magnitude of embedding vectors. Shown in Figure 13, we see the representation weight decay in general does not affect model performance much.

F   General groups

F.1   Theory

We focused on Abelian groups for the most part of the paper. This is, however, simply due to pedagogical reasons. In this section, we show that it is straight-forward to extend definitions of parallelograms and representation quality index (RQI) to general non-Abelian groups. We will also show that most (if not all) qualitative results for the addition group also apply to the permutation group.

Matrix representation for general groups Let us first review the definition of group representation. A representation of a group G on a vector space V is a group homomorphism from G to $\mathrm{GL}(V)$, the general linear group on V. That is, a representation is a map $\rho : G \to \mathrm{GL}(V)$ such that

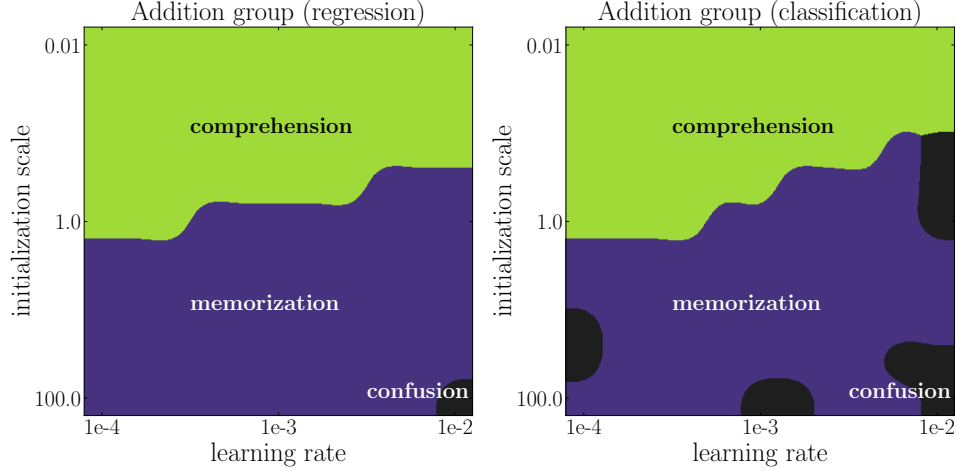$$\rho(g_1 g_2) = \rho(g_1)\rho(g_2), \quad \forall g_1, g_2 \in G. \tag{28}$$

Figure 12: Phase diagrams of decoder learning rate (x axis) and initialization (y axis) for the addition group (left: regression; right: classification). Small intialization scale (top) leads to comprehension.
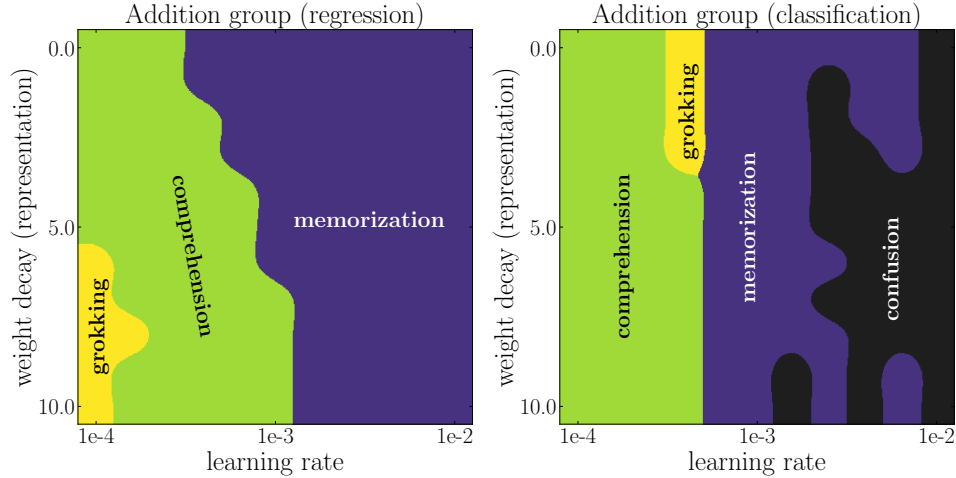

Figure 13: Phase diagrams of decoder learning rate (x axis) and representation weight decay (y axis) for the addition group (left: regression; right: classification). Representation weight decay does not affect model performance much.

In the case V is of finite dimension n, it is common to identify $\mathrm{GL}(V)$ with n by n invertible matrices. The punchline is that: each group element can be represented as a matrix, and the binary operation is represented as matrix multiplication.

A new architecture for general groups Inspired by the matrix representation, we embed each group element a as a learnable matrix $\mathbf{E}_a \in \mathbb{R}^{d \times d}$ (as opposed to a vector), and manually do matrix multiplication before sending the product to the deocder for regression or classification. More concretly, for $a \circ b = c$, our architecture takes as input two embedding matrices $\mathbf{E}_a$ and $\mathbf{E}_b$ and aims to predict $\mathbf{Y}_c$ such that $\mathbf{Y}_c = \mathrm{Dec}(\mathbf{E}_a\mathbf{E}_b)$, where $\mathbf{E}_a\mathbf{E}_b$ means the matrix multiplication of $\mathbf{E}_a$ and $\mathbf{E}_b$. The goal of this simplication is to disentangle learning the representation and learning the arithmetic operation (i.e, the matrix multiplication). We will show that, even with this simplification, we are still able to reproduce the characteristic grokking behavior and other rich phenomenon.

Generalized parallelograms we define generalized parallelograms: (a, b, c, d) is a generalized parallelogram in the representation if $||\mathbf{E}_a\mathbf{E}_b - \mathbf{E}_c\mathbf{E}_d||_F^2 \leq \delta$, where $\delta > 0$ is a threshold to tolerate numerical errors. Before presenting the numerical results for the permutation group, we show an intuitive picture about how new parallelograms can be deduced from old ones for general groups, which is the key to generalization.

19

(a) Abelian Group: $2P \rightarrow P$
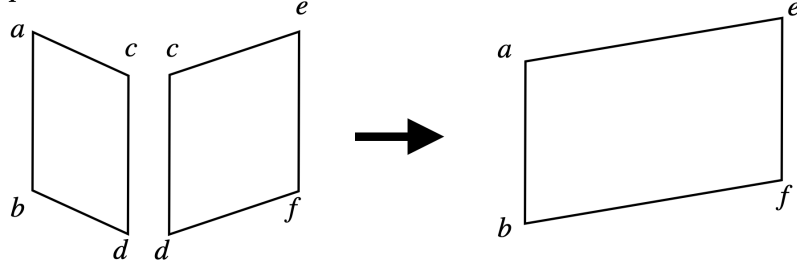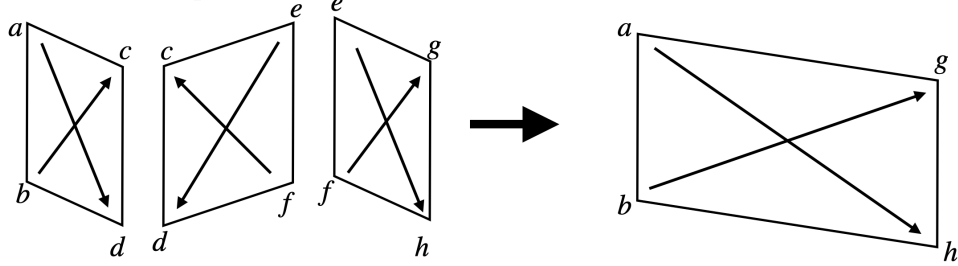

(b) Non-Abelian Group: $3P \rightarrow P$

Figure 14: Deduction of parallelograms

Deduction of parallelograms

We first recall the case of the Abelian group (e.g., addition group). As shown in Figure 14, when (a, d, b, c) and (c, f, d, e) are two parallelograms, we have

$$\begin{aligned} \mathbf{E}_a + \mathbf{E}_d &= \mathbf{E}_b + \mathbf{E}_c \\ \mathbf{E}_c + \mathbf{E}_f &= \mathbf{E}_d + \mathbf{E}_d \end{aligned} \tag{29}$$

We can derive that $\mathbf{E}_a + \mathbf{E}_f = \mathbf{E}_b + \mathbf{E}_e$ implying that (a, f, b, e) is also a parallelogram. That is, for Abelian groups, two parallelograms are needed to deduce a new parallelogram.

For the non-Abelian group, if we have only two parallelograms such that

$$\begin{aligned} \mathbf{E}_a \mathbf{E}_d &= \mathbf{E}_b \mathbf{E}_c \\ \mathbf{E}_f \mathbf{E}_c &= \mathbf{E}_e \mathbf{E}_d \end{aligned} \tag{30}$$

we have $\mathbf{E}_b^{-1}\mathbf{E}_a = \mathbf{E}_c\mathbf{E}_d^{-1} = \mathbf{E}_f^{-1}\mathbf{E}_e$, but this does not lead to something like $\mathbf{E}_f\mathbf{E}_a = \mathbf{E}_e\mathbf{E}_b$, hence useless for generalization. However, if we have a third parallelogram such that

$$\mathbf{E}_e \mathbf{E}_h = \mathbf{E}_f \mathbf{E}_g \tag{31}$$

we have $\mathbf{E}_b^{-1}\mathbf{E}_a = \mathbf{E}_c\mathbf{E}_d^{-1} = \mathbf{E}_f^{-1}\mathbf{E}_e = \mathbf{E}_g\mathbf{E}_h^{-1}$, equivalent to $\mathbf{E}_a\mathbf{E}_h = \mathbf{E}_b\mathbf{E}_g$, thus establishing a new parallelogram (a, h, b, g). That is, for non-Abelian groups, three parallelograms are needed to deduce a new parallelogram.

F.2 Numerical Results

In this section, we conduct numerical experiments on a simple non-abelian group: the permutation group S_3. The group has 6 group elements, hence the full dataset contains 36 samples. We embed each group element a into a learnable 3×3 embedding matrix $\mathbf{E}_a$. We adopt the new architecture described in the above subsection: we hard code matrix multiplication of two input embedding matrices before feeding to the decoder. After defining the generalized parallelogram in the last subsection, we can continue to define RQI (as in Section 3) and predict accuracy $\widehat{\text{Acc}}$ from representation (as in appendix B). We also compute the number of steps needed to reach $\text{RQI} = 0.95$.

Representation We flatten each embedding matrix into a vector, and apply principal component analysis (PCA) to the vectors. We show the first three principal components of these group elements in Figure 15. On the plane of PC1 and PC3, the six points are organized as a hexagon.
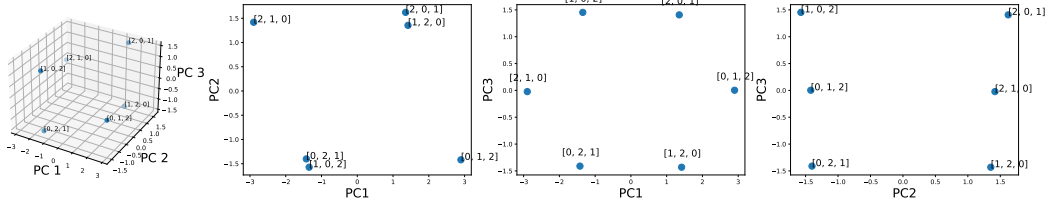
20

Figure 15: Permuation group S_3. First three principal components of six embedding matrices $\mathbb{R}^{3\times 3}$.
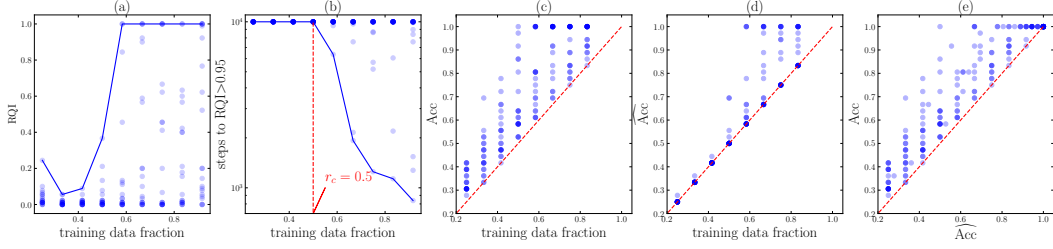

Figure 16: Permutation group S_3. (a) RQI increases as training set becomes larger. Each scatter point is a random seed, and the blue line is the highest RQI obtained with a fixed training set ratio; (b) steps to reach RQI>0.95. The blue line is the smallest number of steps required. There is a phase transition around $r_c = 0.5$. (c) real accuracy Acc; (d) predicted accuracy $\widehat{\text{Acc}}$; (e) comparison of Acc and $\widehat{\text{Acc}}$: $\widehat{\text{Acc}}$ serves as a lower bound of Acc.

RQI In Figure 16(a), we show RQI as a function of training data fraction. For each training data fraction, we run 11 random seeds (shown as scatter points), and the blue line corresponds to the highest RQI.

Steps to reach RQI= 0.95 In Figure 16(b), we whow the steps to reach $\text{RQI} > 0.95$ as a function of training data fraction, and find a phase transition at $r = r_c = 0.5$. The blue line corresponds to the best model (smallest number of steps).

Accuracy The real accuracy Acc is shown in Figure 16(c), while the predicted accuracy $\widehat{\text{Acc}}$ (calculated from RQI) is shown in Figure 16(d). Their comparison is shown in (e): $\widehat{\text{Acc}}$ is a lower bound of Acc, implying that there must be some generalization mechanism beyond RQI.

Phase diagram We investigate how the model performance varies under the change of two knobs: decoder learning rate and decoder weight decay. We calculate the number of steps to training accuracy ≥ 0.9 and validation accuracy ≥ 0.9, respectively, shown in Figure 5 (d).