Towards Understanding Grokking: An Effective Theory of Representation Learning

Ziming Liu, Ouail Kitouni, Niklas Nolte, Eric J. Michaud, Max Tegmark, Mike Williams
Department of Physics, Institute for AI and Fundamental Interactions, MIT
{zmliu,kitouni,nnolte,ericjm,tegmark,mwill}@mit.edu

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. We find on transformers the grokking phase stays closer to the memorization phase (compared to the comprehension 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, raising three key puzzles:

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 input embeddings, i.e., a representation that has the appropriate structure for the task and which can be predicted from the theory in Section 3. 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).

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

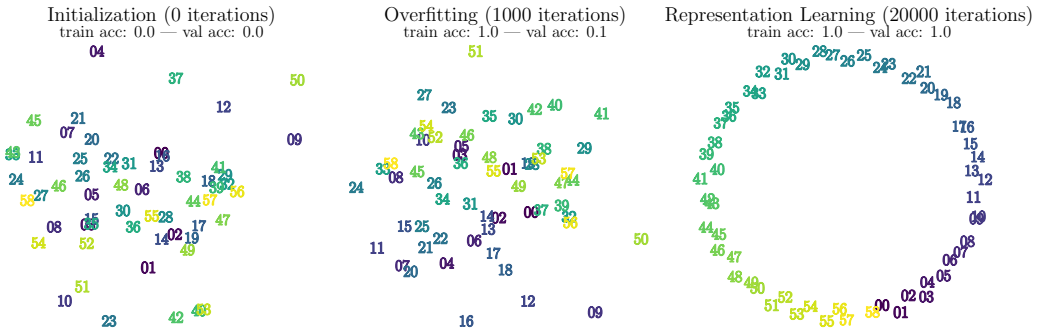


Figure 1: Visualization of the first two principal components of the learned input embeddings at different training stages of a transformer learning modular addition. We observe that generalization coincides with the emergence of structure in the embeddings. See Section 4.2 for the training details.

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 6.

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]

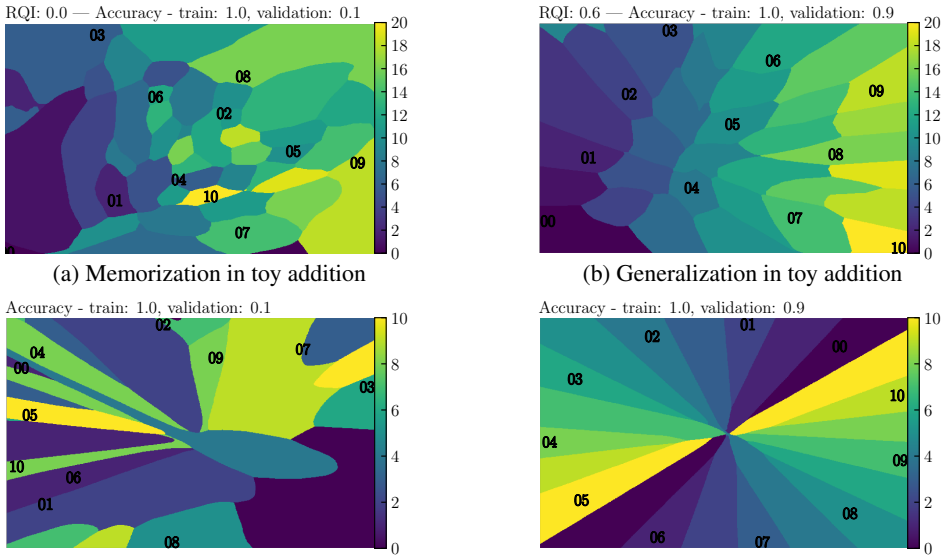
2 Problem Setting

Power et al. [1] observe grokking on a less common task – learning "algorithmic" binary operations. Given some binary operation $\circ$, a network is tasked with learning the map $(a, b) \mapsto c$ where $c = a \circ b$. 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 (mean-squared error) and classification (cross-entropy). 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 B). 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 H.

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: https://github.com/ejmichaud/grokking-squared



(a) Memorization in toy addition (b) Generalization in toy addition

(c) Memorization in toy modular addition (d) Generalization in toy modular addition

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 Occurs: 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 (a, b) shows parallelograms in toy addition, and (c, 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. The concept of an effective theory in physics is similar to model reduction in computational methods in that it aims to describe complex phenomena with simple yet intuitive pictures. 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 in embedding space 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.

Proposition 1. *When the training loss is zero, any parallelogram (i, j, m, n) in representation $\mathbf{R}$ satisfies $i + j = m + n$.*

Proof. 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. □

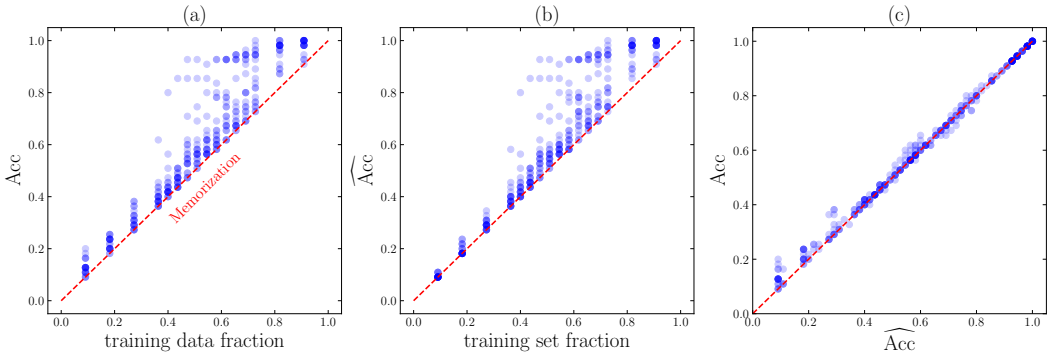


Figure 3: We compute accuracy (of the full dataset) either measured empirically Acc, or predicted from the representation of the embeddings $\widehat{\text{Acc}}$. These two accuracies as a function of training data fraction are plotted in (a)(b), and their agreement is shown in (c).

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 D 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 structured representation of input embeddings leads to generalization. We use an example to illustrate the origin of generalization here. In the setup of Figure 2 (b), 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 dynamics of embedding vectors

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 2 provides a mechanism for the formation of parallelograms.

[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 E for details.

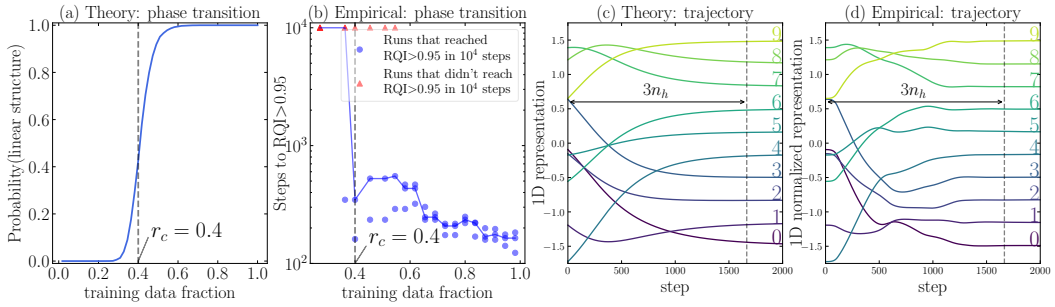


Figure 4: (a) The effective theory predicts a phase transition in the probability of obtaining a linear representation around $r_c = 0.4$. (b) Empirical results display a phase transition of RQI around $r_c = 0.4$, in agreement with the theory (the blue line shows the median of multiple random seeds). 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.

Proposition 2. *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.*

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

The dynamics of the trained embedding vectors 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 (in the physics effective theory sense). We will assume that the normalized embedding vectors obey a gradient flow for an effective loss function of the form

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

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

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 F:

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

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 $\mathrm{Nullity}(A(P_0)) = 2$ 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 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 $\mathrm{RQI} > 0.95$ 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 $\ell_{\text{eff}} = \frac{1}{2}\mathbf{R}^T\mathbf{H}\mathbf{R}$, $\mathbf{R} = [\mathbf{E}_0, \mathbf{E}_1, \cdots, \mathbf{E}_{p-1}]$, 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 $\mathbf{R}(t = 0) = \sum_i a_i \bar{\mathbf{v}}_i$, then we have $\mathbf{R}(t) = \sum_i a_i \bar{\mathbf{v}}_i e^{-\lambda_i t}$. 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. We call λ_3 the *grokking rate*. 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 take infinitesimal step sizes.

Dependence of grokking on data size Note that ℓ_{eff} involves averaging over parallelograms in the training set, it is dependent on training data size, so is λ_3. In Figure 5 (a), we plot the dependence of λ_3 on training data fraction. There are many datasets with the same data size, so λ_3 is a probabilistic function of data size.

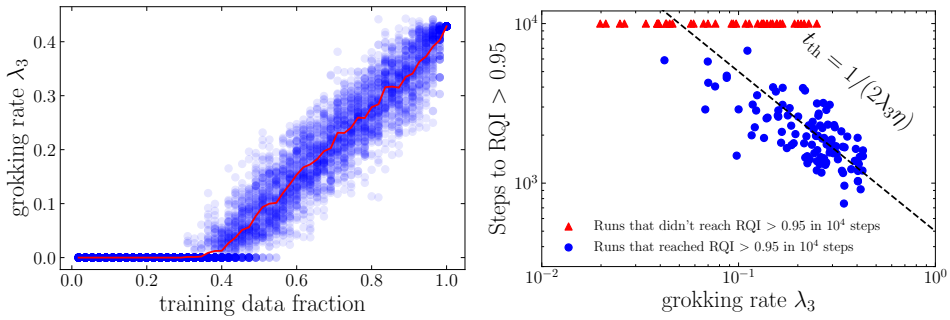
Two insights on grokking can be extracted from this plot: (i) When the data fraction is below some threshold (around 0.4), λ_3 is zero with high probability, corresponding to no generalization. This again verifies our critical point in Figure 4. (ii) When data size is above the threshold, λ_3 (on average) is an increasing function of data size. This implies that grokking time $t \sim 1/\lambda_3$ decreases as training data size becomes larger, an important observation from [1].

To verify our effective theory, we compare the grokking steps obtained from real neural network training (defined as steps to $\mathrm{RQI} > 0.95$), and those predicted by our theory $t_{\text{th}} \sim \frac{1}{\lambda_3 \eta}$ (η is the embedding learning rate), shown in Figure 5 (b). The theory agrees qualitatively with neural networks, showing the trend of decreasing grokking steps as increasing data size. The quantitative differences might be explained as the gap between our effective loss and actual loss.

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 (d) 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



(a) (b)

Figure 5: Effective theory explains the dependence of grokking time on data size, for the addition task. (a) Dependence of λ_3 on training data fraction. Above the critical data fraction (around 0.4), as data size becomes larger, λ_3 increases hence grokking time $t \sim 1/\lambda_3$ (predicted by our effective theory) decreases. (b) Comparing grokking steps (defined as RQI > 0.95) predicted by the effective theory with real neural network results. $\eta = 10^{-3}$ is the learning rate of the embeddings.

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 depends 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.

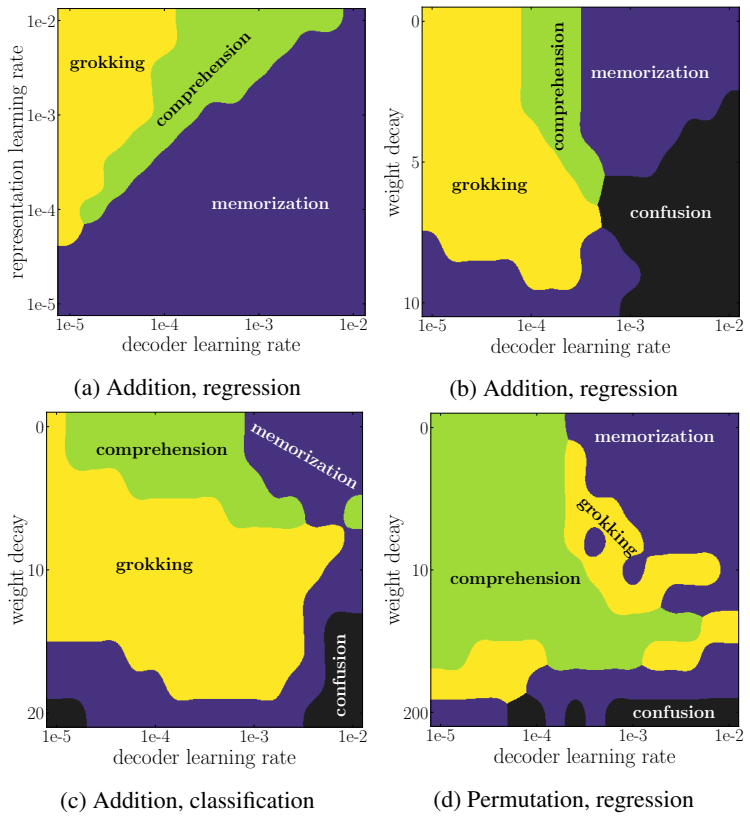
4.1 Phase diagram of a toy model

Training details We update the representation and the decoder with different optimizers. For the 1D embeddings, we use the Adam optimizer with learning rate $[10^{-5}, 10^{-2}]$ and zero weight decay. For the decoder, we use an AdamW optimizer with the learning rate in $[10^{-5}, 10^{-2}]$ and the weight decay in $[0, 10]$ (regression) or $[0, 20]$ (classification). For training/validation spliting, we choose 45/10 for non-modular addition ($p = 10$) and 24/12 for the permutation group S_3. We hard-code addition or matrix multiplication (details in Appendix H) 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 in Appendix A. 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 6 shows the phase diagrams for the addition group and the permutation group. They display quite rich phenomena.

Competition between representation learning and decoder overfitting In the regression setup of the addition dataset, we show how the competition between representation learning and decoder learning (which depend on both learning rate and weight decay, among other things) lead to different learning phases in Figure 6 (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, the dynamics of the particles are described *only* by their relative positions, in that sense, structure forms mainly due to inter-particle interactions (in reality, these interactions are mediated by the decoder and the loss). The decoder plays the role of an environment exerting external forces on the embeddings. If the magnitude of the external forces are small/large one can expect better/worse representations.



(a) Addition, regression

(b) Addition, regression

(c) Addition, classification

(d) Permutation, regression

Figure 6: 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. In (b)(c)(d), grokking is sandwiched between comprehension and memorization.

Universality of phase diagrams We fix the embedding learning rate to be 10^{-3} and sweep instead decoder weight decay in Figure 6 (b)(c)(d). The phase diagrams correspond to addition regression (b), addition classification (c) and permutation regression (d), respectively. Common phenomena 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.

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 7).

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

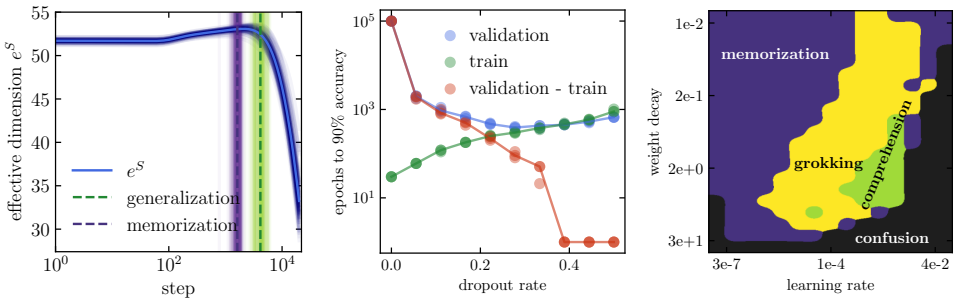


Figure 7: Left: Evolution of the effective dimension of the embeddings (defined as the exponential of the entropy) during training and evaluated over 100 seeds. Center: Effect of dropout on speeding up generalization. Right: Phase diagram of the transformer architecture. A scan is performed over the 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).

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 7, 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. Another interesting insight appears when we project the embeddings at initialization onto the principal axes at the end of training. Some of the structure required for generalization exists before training hinting at a connection with the Lottery Ticket Hypothesis. See Appendix K for more details.

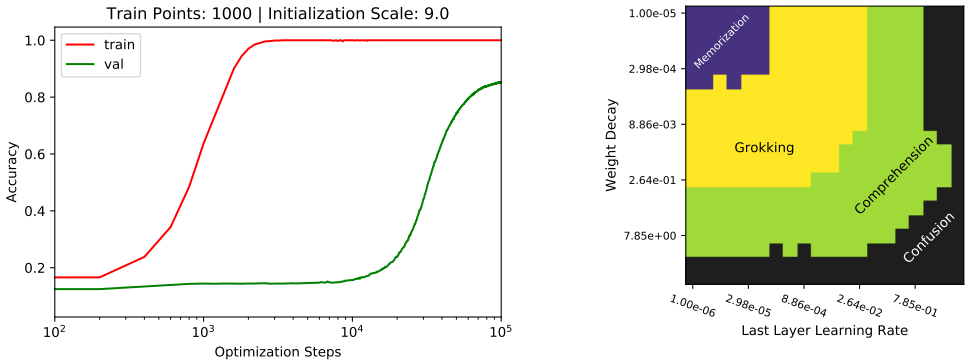
In Figure 7 (right), we show a comparable phase diagram to Figure 6 evaluated now in the transformer setting. Note that, as opposed to the setting in [1], weight decay has only been applied to the decoder and not to the embedding layer. Contrary to the toy model, a certain amount of weight decay proves 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. This is also observed in an image classification task where we were able to induce grokking. See Appendix J for more details.

4.2.1 Grokking Experiment on MNIST

We now demonstrate, for the first time, that grokking (significantly delayed generalization) is a more general phenomenon in machine learning that can occur not only on algorithmic datasets, but also on mainstream benchmark datasets. In particular, we exhibit grokking on MNIST in Figure 8 and demonstrate that we can control grokking by varying optimization hyperparameters. More details on the experimental setup are in Appendix J.

5 Related work

Relatively few works have analyzed the phenomenon of grokking. [2] describe the circuit that transformers use to perform modular addition, track its formation over training, and broadly suggest that grokking is related to the phenomenon of "phase changes" in neural network training. [3, 4]



(a) (b)

Figure 8: Left: Training curves for a run on MNIST, in the setting where we observe grokking. Right: Phase diagram with the four phases of learning dynamics on MNIST.

provided earlier speculative, informal conjectures on grokking [3, 4]. Our work is related to the following broad research directions:

Learning mathematical structures [5] 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 [6], knot theory [7] and group theory [8].

Double descent Grokking is somewhat reminiscent of the phenomena of "epoch-wise" *double descent* [9], where generalization can improve after a period of overfitting. [10] 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 [11–14]. 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 proved to be useful in understanding deep learning from a theoretical perspective. These tools include effective theories [15, 16], conservation laws [17] and free energy principle [18]. In addition, statistical physics has been identified as a powerful tool in studying generalization in neural networks [19–22]. Our work connects a low-level understanding of models with their high-level performance. In a recent work, researchers at Anthropic [23], 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.

6 Conclusion

We have shown how, in both toy models and 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 four learning phases (comprehension, grokking, memorization and confusion) which depend on the decoder capacity and learning speed (given by, among other things, learning rate and weight decay) 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*, connecting 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 [24], to be a rich area for further work. The broader impact of such work, if successful, could be to make models more transparent and predictable [23, 25, 26], 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] Neel Nanda and Tom Lieberum. A mechanistic interpretability analysis of grokking, 2022. URL https://www.alignmentforum.org/posts/N6WM6hs7RQMKDhYjB/a-mechanistic-interpretability-analysis-of-grokking.

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

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

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

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

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

[8] 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.

[9] 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.

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

[11] 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.

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

[13] 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.

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

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

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

[17] 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.

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

[19] Federica Gerace, Bruno Loureiro, Florent Krzakala, Marc Mézard, and Lenka Zdeborová. Generalisation error in learning with random features and the hidden manifold model. In *International Conference on Machine Learning*, pages 3452–3462. PMLR, 2020.

[20] Mohammad Pezeshki, Amartya Mitra, Yoshua Bengio, and Guillaume Lajoie. Multi-scale feature learning dynamics: Insights for double descent. In *International Conference on Machine Learning*, pages 17669–17690. PMLR, 2022.

[21] Sebastian Goldt, Bruno Loureiro, Galen Reeves, Florent Krzakala, Marc Mezard, and Lenka Zdeborova. The gaussian equivalence of generative models for learning with shallow neural networks. In Joan Bruna, Jan Hesthaven, and Lenka Zdeborova, editors, *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*, volume 145 of *Proceedings of Machine Learning Research*, pages 426–471. PMLR, 16–19 Aug 2022. URL https://proceedings.mlr.press/v145/goldt22a.html.

[22] R Kuhn and S Bos. Statistical mechanics for neural networks with continuous-time dynamics. *Journal of Physics A: Mathematical and General*, 26(4):831, 1993.

[23] 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.

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

[25] 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.

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

[27] 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.

[28] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117 (40):24652–24663, 2020.

[29] Wikipedia contributors. Thomson problem — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Thomson_problem&oldid=1091431454, 2022. [Online; accessed 29-July-2022].

[30] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

[31] Zhi-Qin John Xu, Yaoyu Zhang, and Yanyang Xiao. Training behavior of deep neural network in frequency domain. In *International Conference on Neural Information Processing*, pages 264–274. Springer, 2019.

[32] Yaoyu Zhang, Zhi-Qin John Xu, Tao Luo, and Zheng Ma. A type of generalization error induced by initialization in deep neural networks. In *Mathematical and Scientific Machine Learning*, pages 144–164. PMLR, 2020.

[33] Ziming Liu, Eric J. Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data, 2022.

[34] Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 3635–3673. PMLR, 09–12 Jul 2020. URL https://proceedings.mlr.press/v125/woodworth20a.html.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (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]
   (b) Did you include complete proofs of all theoretical results? [Yes]
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]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (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)? [Yes] All experiments were run on a workstation with two NVIDIA A6000 GPUs within a few days.
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? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (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 Definitions of the phases of learning

Table 1: Definitions of the four phases of learning

Phase	criteria		
	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

B 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 [27] 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 $\phi(x) = \ln(x)$.

C An illustrative example

We use a concrete case to illustrate why parallelograms lead to generalization (see Figure 9). For the purpose of illustration, we exploit a curriculum learning setting, where a neural network is fed with a few new samples each time. We will illustrate that, as we have more samples in the training set, the ideal model $\mathcal{M}^*$ (defined in Section 3.2) will arrange the representation $\mathbf{R}^*$ in a more structured way, i.e., more parallelograms are formed, which helps generalization to unseen validation samples. For simplicity we choose $p = 6$.

- $D_1 = (0, 4)$ and $D_2 = (1, 3)$ have the same label, so $(0, 4, 1, 3)$ becomes a parallelogram such that $\mathbf{E}_0 + \mathbf{E}_4 = \mathbf{E}_1 + \mathbf{E}_3 \to \mathbf{E}_3 - \mathbf{E}_0 = \mathbf{E}_4 - \mathbf{E}_1$. $D_3 = (1, 5)$ and $D_4 = (2, 4)$ have

the same label, so $(1, 5, 2, 4)$ becomes a parallelogram such that $\mathbf{E}_1 + \mathbf{E}_5 = \mathbf{E}_2 + \mathbf{E}_4 \to \mathbf{E}_4 - \mathbf{E}_1 = \mathbf{E}_5 - \mathbf{E}_2$. We can derive from the first two equations that $\mathbf{E}_5 - \mathbf{E}_2 = \mathbf{E}_3 - \mathbf{E}_0 \to \mathbf{E}_0 + \mathbf{E}_5 = \mathbf{E}_2 + \mathbf{E}_3$, which implies that $(0, 5, 2, 3)$ is also a parallelogram (see Figure 9(a)). This means if $(0, 5)$ in training set, our model can predict $(2, 3)$ correctly.

- $D_5 = (0, 2)$ and $D_6 = (1, 1)$ have the same label, so $\mathbf{E}_0 + \mathbf{E}_2 = 2\mathbf{E}_1$, i.e., 1 is the middle point of 0 and 2 (see Figure 9(b)). Now we can derive that $2\mathbf{E}_4 = \mathbf{E}_3 + \mathbf{E}_5$, i.e., 4 is the middle point of 3 and 5. If $(4, 4)$ is in the training data, our model can predict $(3, 5)$ correctly.
- Finally, $D_7 = (2, 4)$ and $D_8 = (3, 3)$ have the same label, so $2\mathbf{E}_3 = \mathbf{E}_2 + \mathbf{E}_4$, i.e., 3 should be placed at the middle point of 2 and 4, ending up Figure 9(c). This linear structure agrees with the arithmetic structure of $\mathbb{R}$.

In summary, although we have $p(p+1)/2 = 21$ different training samples for $p = 6$, we only need 8 training samples to uniquely determine the perfect linear structure (up to linear transformation). The punchline is: representations lead to generalization.

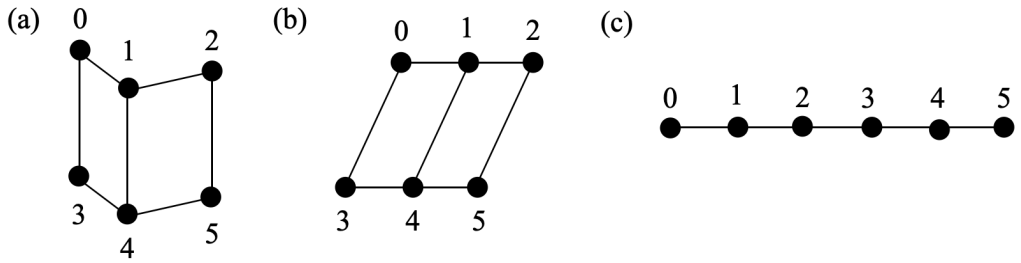


Figure 9: As we include more data in the training set, the (ideal) model is capable of discovering increasingly structured representations (better RQI), from (a) to (b) to (c).

D 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 $\mathrm{Dec}(\mathbf{E}_\mathrm{m} + \mathbf{E}_\mathrm{n}) = \mathbf{Y}_{\mathrm{m+n}}$, holds when (m, n) is in the 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 $\mathrm{RQI} \propto |P|$ 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:

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

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 $\mathrm{Acc} \approx \widehat{\mathrm{Acc}}$ in a toy setup (addition dataset $p = 10$, 1D embedding space, hard code addition), as shown in Figure 3 (c). Figure 3 (a)(b) show Acc and $\widehat{\mathrm{Acc}}$ 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 vectors, we do not expect such agreement can trivially extend to high dimensional embedding vectors. 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 dimension and drop all other $N-1$ useless dimensions. It would be interesting to investigate how to define an RQI that takes into account the role of decoder in future works.

E 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 Figure 10.

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_1B_1 is equal and parallel to A_2B_2, and A_2B_2 is equal and parallel to A_3B_3, then we can deduce that A_1B_1 is equal and parallel to A_3B_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

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

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.,

$$P(R) \subseteq P(D). \quad (15)$$

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:

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

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

$$\widehat{\mathrm{Acc}} \equiv \widehat{\mathrm{Acc}}(D, P(R)) \leq \widehat{\mathrm{Acc}}(D, P(D)) \equiv \overline{\mathrm{Acc}}. \quad (17)$$

In Figure 10 (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 10 (a)(b)(d)(e), four quantities (RQI, $\overline{\mathrm{RQI}}$, Acc, $\overline{\mathrm{Acc}}$) 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 $\overline{\mathrm{RQI}} = 1$ 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.

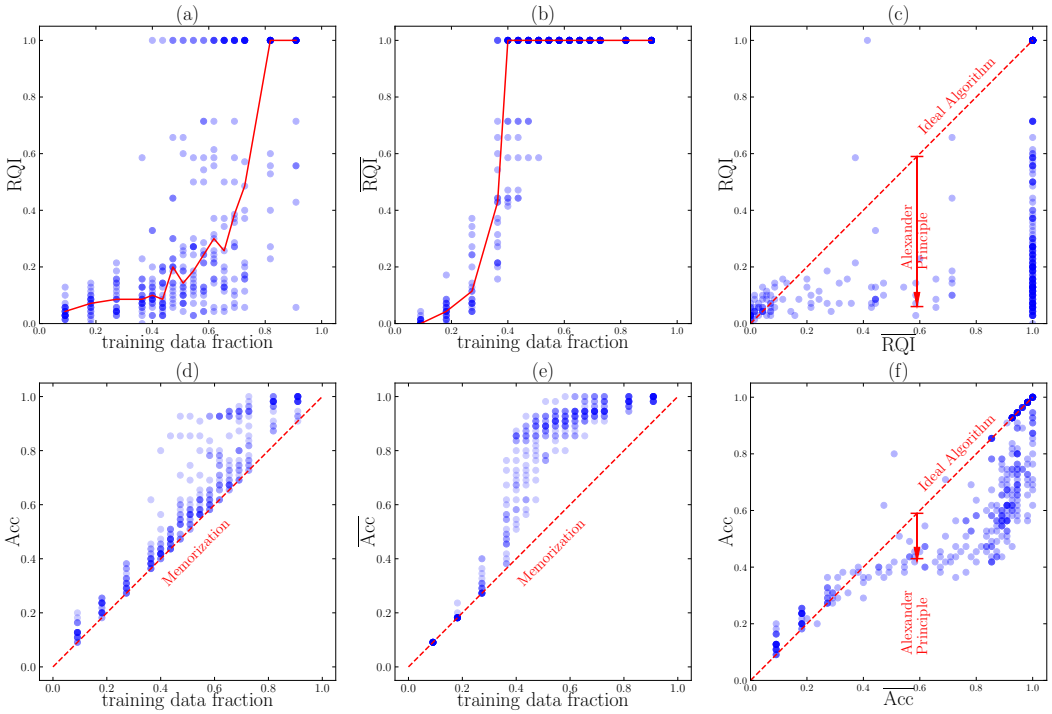


Figure 10: We compare RQI and Acc for an ideal algorithm (with bar) and a realistic algorithm (without bar). In (a)(b)(d)(e), four quantities (RQI, $\overline{\text{RQI}}$, Acc, $\overline{\text{Acc}}$) as functions of training data fraction are shown. In (c)(f), RQI and Acc of the ideal algorithm sets upper bounds for those of the realistic algorithm.

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 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 11 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 12, 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 12 left), and (2) relatively slow decoder $(\eta_1, \eta_2) = (10^{-2}, 10^{-3})$ (Figure 12) 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.

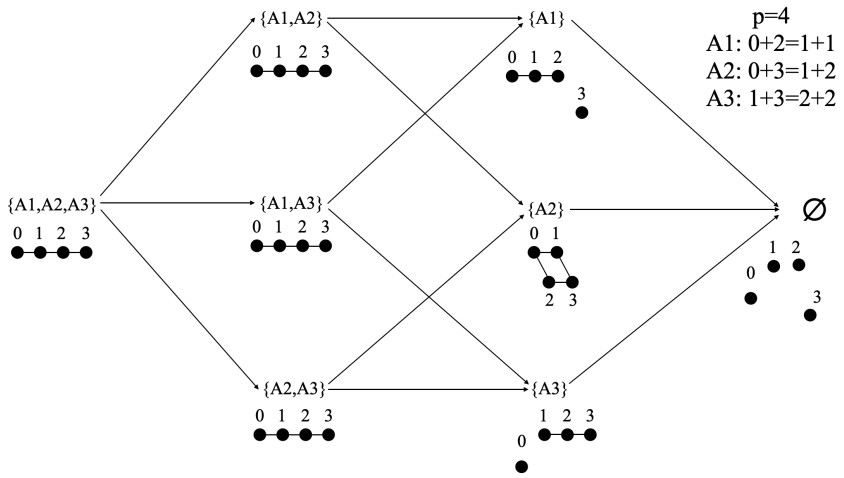


Figure 11: $p = 4$ case. Equation set A (or geometrically, representation) has a hierarchy: $a \to b$ means a is a parent of b, and b is a child of a. A realistic model can only generate representations that are descendants of the representation generated by an ideal model.

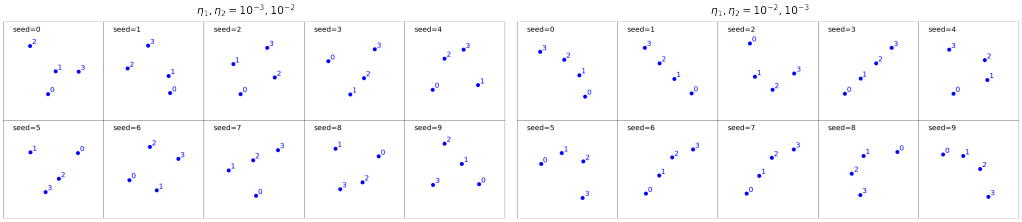


Figure 12: $p = 4$ case. Representations obtained from training neural networks are displayed. η_1 and η_2 are learning rates of the representation and the decoder, respectively. As described in the main text, $(\eta_1, \eta_2) = (10^{-2}, 10^{-3})$ (right) is more ideal than $(\eta_1, \eta_2) = (10^{-3}, 10^{-2})$ (left), thus producing representations containing more parallelograms.

F Conservation laws of the effective theory

Recall that the effective loss function

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

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 $\mathbf{E}'_i = a\mathbf{E}_i + \mathbf{b}$. We will ignore the $1/|P_0(D)|$ factor in ℓ_0 since having it is equivalent to rescaing time, which does not affect conservation laws. 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

$$\begin{aligned}\frac{dZ_0}{dt} &= 2\sum_i \mathbf{E}_k \cdot \frac{d\mathbf{E}_k}{dt} \\ &= \frac{2}{Z_0^2}\sum_i \mathbf{E}_i \cdot (-Z_0 \frac{\partial \ell_0}{\partial \mathbf{E}_k} + 2\ell_0 \mathbf{E}_k) \\ &= \frac{2}{Z_0}(-\sum_k \frac{\partial \ell_0}{\partial \mathbf{E}_k} \cdot \mathbf{E}_k + 2\ell_0) \\ &= 0.\end{aligned} \tag{26}$$

where the last equation uses the fact that

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

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

$$\begin{aligned}\frac{d\mathbf{C}}{dt} &= \sum_k \frac{d\mathbf{E}_k}{dt} \\ &= -\frac{1}{Z_0}\sum_k \frac{\partial \ell_0}{\partial \mathbf{E}_k} \\ &= -\frac{2}{Z_0}\sum_k \sum_{(i,j,m,n)\in P_0(D)} (\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n)(\delta_{ik} + \delta_{jk} - \delta_{mk} - \delta_{nk}) \\ &= \mathbf{0}.\end{aligned} \tag{27}$$

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

G More phase diagrams of the toy setup

We study another three hyperparameters in the toy setup by showing phase diagrams similar to Figure 6. 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 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 13, 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

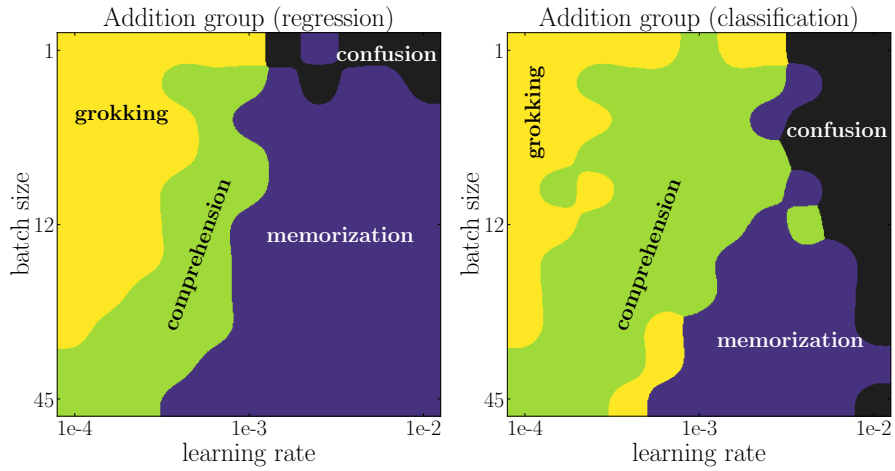


Figure 13: Phase diagrams of decoder learning rate (x axis) and batch size (y axis) for the addition group (left: regression; right: classification). Small decoder leanrning rate and large batch size (bottom left) lead to comprehension.

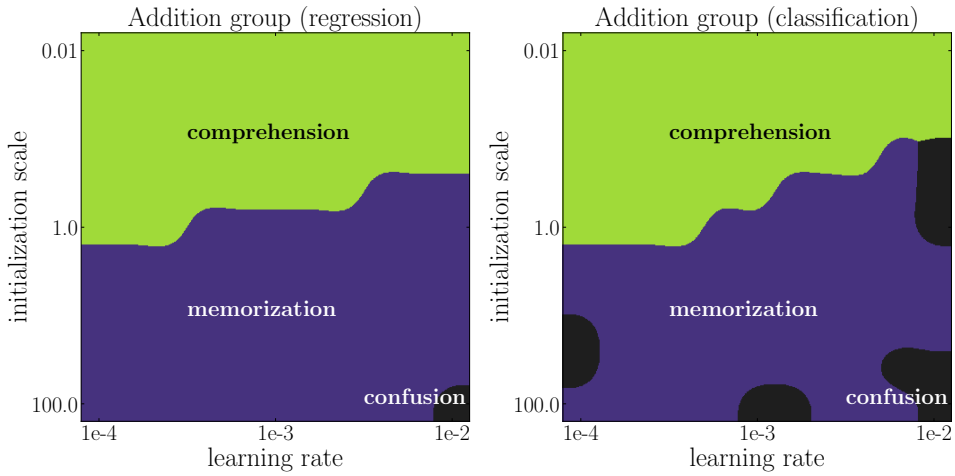


Figure 14: Phase diagrams of decoder learning rate (x axis) and initialization (y axis) for the addition group (left: regression; right: classification). Small intialization scale (top) leads to comprehension.

is called the initialization scale. Shown in Figure 14, 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 15, we see the representation weight decay in general does not affect model performance much.

H General groups

H.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.

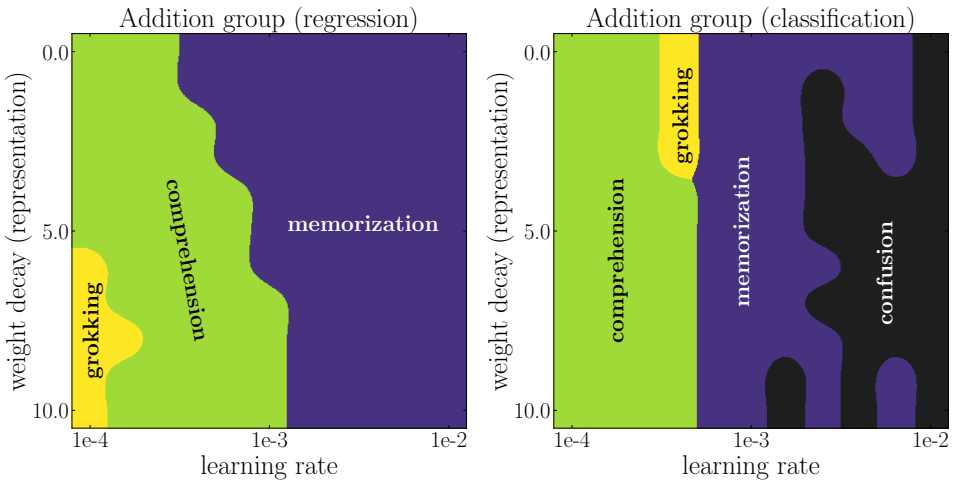


Figure 15: Phase diagrams of decoder learning rate (x axis) and representation weight decay (y axis) for the addition group (left: regression; right: classification). Representation weight decay does not affect model performance much.

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

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

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 $||\mathbf{E}_a\mathbf{E}_b - \mathbf{E}_c\mathbf{E}_d||_F^2 \le \delta$, 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.

Deduction of parallelograms We first recall the case of the Abelian group (e.g., addition group). As shown in Figure 16, 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

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

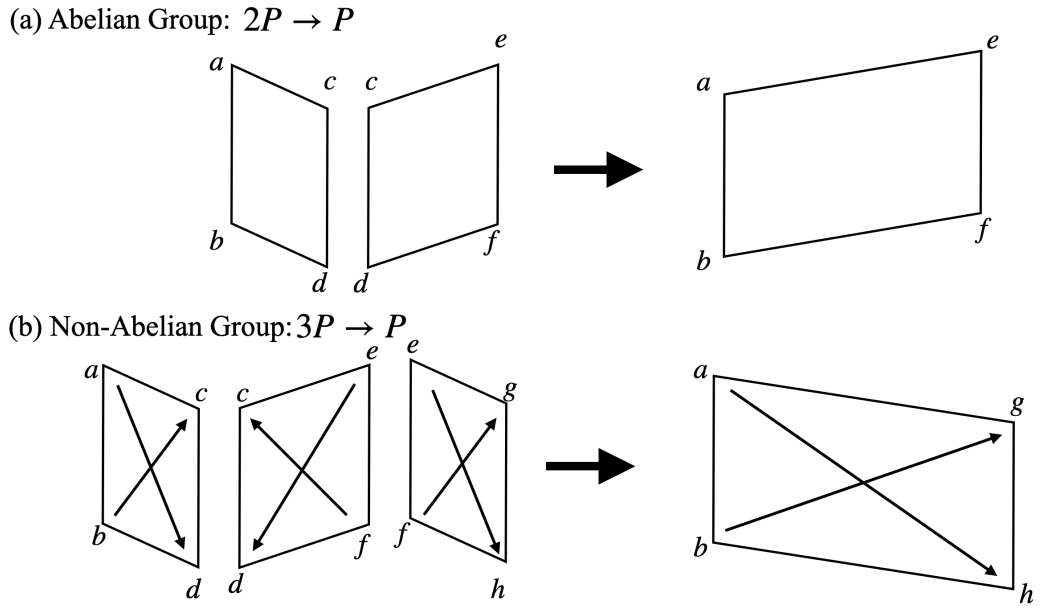


Figure 16: Deduction of parallelograms

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.

H.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 $\widehat{\mathrm{Acc}}$ from representation (as in appendix D). We also compute the number of steps needed to reach $\mathrm{RQI} = 0.95$.

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 17. On the plane of $\mathrm{PC1}$ and $\mathrm{PC3}$, the six points are organized as a hexagon.

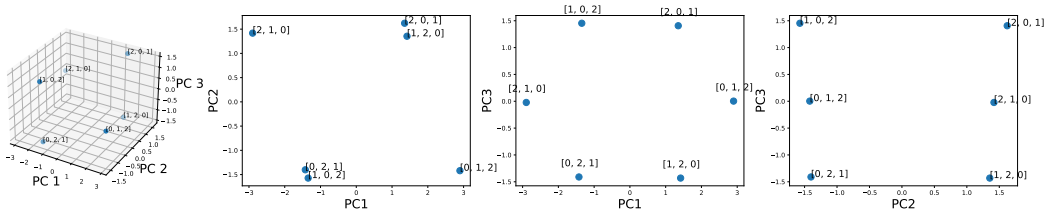


Figure 17: Permuation group S_3. First three principal components of six embedding matrices $\mathbb{R}^{3\times 3}$.

RQI In Figure 18 (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 18 (b), we whow the steps to reach $\mathrm{RQI} > 0.95$ 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).

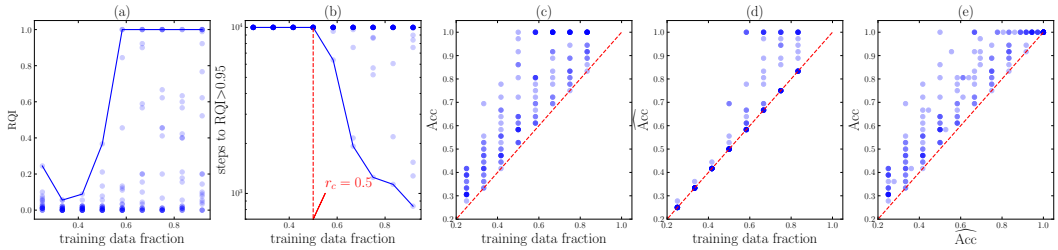


Figure 18: Permutation group S_3. (a) RQI increases as training set becomes larger. Each scatter point is a random seed, and the blue line is the highest RQI obtained with a fixed training set ratio; (b) steps to reach $\mathrm{RQI} > 0.95$. The blue line is the smallest number of steps required. There is a phase transition around $r_c = 0.5$. (c) real accuracy Acc; (d) predicted accuracy $\widehat{\mathrm{Acc}}$; (e) comparison of Acc and $\widehat{\mathrm{Acc}}$: $\widehat{\mathrm{Acc}}$ serves as a lower bound of Acc.

Accuracy The real accuracy Acc is shown in Figure 18 (c), while the predicted accuracy $\widehat{\mathrm{Acc}}$ (calculated from RQI) is shown in Figure 18 (d). Their comparison is shown in (e): $\widehat{\mathrm{Acc}}$ 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 6 (d).

I Effective theory for image classification

In this section, we show our effective theory proposed in Section 3.2 can generalize beyond algorithmic datasets. In particular, we will apply the effective theory to image classifications. We find that: (i) The effective theory naturally gives rise to a novel self-supervised learning method, which can provably avoid mode collapse without contrastive pairs. (ii) The effective theory can shed light on the neural collapse phenomenon [28], in which same-class representations collapse to their class-means.

We first describe how the effective theory applies to image classification. The basic idea is again that, similar to algorithmic datasets, neural networks try to develop a structured representation of the inputs based on the relational information between samples (class labels in the case of image classification, sum parallelograms in the case of addition, etc.). The effective theory has two ingredients: (i) samples with the same label are encouraged to have similar representations; (ii) the effective loss function is scale-invariant to avoid all representations collapsing to zero (global collapse). As a result, an effective loss for image classification has the form

$$\ell_{\text{eff}} = \frac{\ell}{Z}, \quad \ell = \sum_{(\mathbf{x},\mathbf{y})\in P} |\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})|^2, \quad Z = \sum_{\mathbf{x}} |\mathbf{f}(\mathbf{x})|^2 \tag{32}$$

where $\mathbf{x}$ is an image, $\mathbf{f}(\mathbf{x})$ is its representation, $(\mathbf{x}, \mathbf{y}) \in P$ refers to unique pairs $\mathbf{x}$ and $\mathbf{y}$ that have the same label. Scale invariance means the loss function ℓ_{eff} does not change under the linear scaling $\mathbf{f}(\mathbf{x}) \to a\mathbf{f}(\mathbf{x})$.

Relation to neural collapse It was observed in [28] that image representations in the penultimate layer of the model have some interesting features: (i) representations of same-class images collapse to their class-means; (ii) class-means of different classes develop into an equiangular tight frame. Our effective theory is able to predict the same-class collapse, but does not necessarily put class-means into equiangular tight frames. We conjecture that little explicit repulsion among different classes can help class-means develop into an equiangular tight frame, similar to electrons developing into lattice structures on a sphere under repulsive Coulomb forces (the Thomson problem [29]). We would like to investigate this modification of the effective theory in the future.

Experiment on MNIST We directly apply the effective loss Eq. (32) to the MNIST dataset. Firstly, each image $\mathbf{x}$ is randomly encoded to a 2D embedding $\mathbf{f}(\mathbf{x})$ via the same encoder MLP whose weights are randomly initialized. We then train these embeddings by minimizing the effective loss ℓ_{eff} with

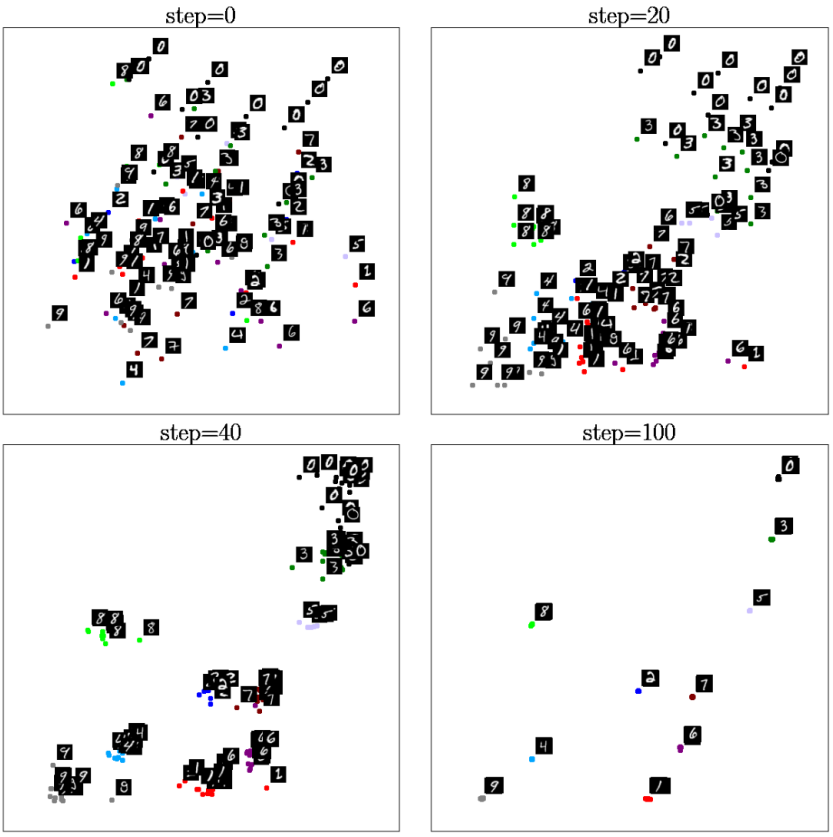


Figure 19: Our effective theory applies to MNIST image classifications. Same-class images collapse to their class-means, while class-means of different classes stay separable. As such, the effective theory serves as a novel self-supervised learning method, as well as shed some light on neural collapse. Please see texts in Appendix I.

an Adam optimizer (10^{-3} learning rate) for 100 steps. We show the evolution of these embeddings in Figure 19. Images of the same class collapse to their class-means, and different class-means do not collapse. This means that our effective theory can give rise to a good representation learning method which only exploits non-contrastive relational information in datasets.

Link to self-supervised learning Note that ℓ itself is vulnerable to global collapse, in the context of Siamese learning without contrastive pairs. Various tricks (e.g., decoder with momentum, stop gradient) [13, 30] have been proposed to avoid global collapse. However, the reasons why these tricks can avoid global collapse are unclear. We argue ℓ fails simply because $\ell \to a^2\ell$ under scaling $\mathbf{f}(\mathbf{x}) \to a\mathbf{f}(\mathbf{x})$ so gradient descent on ℓ encourage $a \to 0$. Based on this picture, our effective theory provides another possible fix: make the loss function ℓ scale-invariant (by the normalized loss ℓ_{eff}), so the gradient flow has no incentive to change representation scales. In fact, we can prove that the gradient flow on ℓ_{eff} preserve Z (variance of representations) so that global collapse is avoided provably:

$$
\begin{aligned}
\frac{\partial \ell_{\text{eff}}}{\partial \mathbf{f}(\mathbf{x})} &= \frac{1}{Z}\frac{\partial \ell}{\partial \mathbf{f}(\mathbf{x})} - \frac{l}{Z^2}\frac{\partial Z}{\partial \mathbf{f}(\mathbf{x})} = \frac{2}{Z}\sum_{\mathbf{y}\sim\mathbf{x}}(\mathbf{f}(\mathbf{x})-\mathbf{f}(\mathbf{y})) - \frac{2\ell}{Z^2}\mathbf{f}(\mathbf{x}), \\
\frac{dZ}{dt} &= 2\sum_{\mathbf{x}}\mathbf{f}(\mathbf{x})\cdot\frac{d\mathbf{f}(\mathbf{x})}{dt} = 2\sum_{\mathbf{x}}\mathbf{f}(\mathbf{x})\cdot\frac{\partial \ell_{\text{eff}}}{\partial \mathbf{f}(\mathbf{x})} \\
&= \frac{4}{Z}\sum_{\mathbf{x}}\mathbf{f}(\mathbf{x})\cdot(\sum_{\mathbf{y}\sim\mathbf{x}}(\mathbf{f}(\mathbf{x})-\mathbf{f}(\mathbf{y})) - \frac{\ell}{Z}\mathbf{f}(\mathbf{x})) \\
&= \frac{4}{Z}\big[\sum_{\mathbf{x}}\mathbf{f}(\mathbf{x})\cdot\sum_{\mathbf{y}\sim\mathbf{x}}(\mathbf{f}(\mathbf{x})-\mathbf{f}(\mathbf{y})) - \sum_{\mathbf{x}}\frac{\ell}{Z}|\mathbf{f}(\mathbf{x})|^2\big] \\
&= 0.
\end{aligned} \tag{33}
$$

where we use the fact that

$$
\sum_{\mathbf{x}}\mathbf{f}(\mathbf{x})\cdot\sum_{\mathbf{y}\sim\mathbf{x}}(\mathbf{f}(\mathbf{x})-\mathbf{f}(\mathbf{y})) = \sum_{(\mathbf{x},\mathbf{y})\in P}(\mathbf{f}(\mathbf{x})-\mathbf{f}(\mathbf{y}))\cdot(\mathbf{f}(\mathbf{x})-\mathbf{f}(\mathbf{y})) = \ell \tag{34}
$$

J Grokking on MNIST

To induce grokking on MNIST, we make two nonstandard decisions: (1) we reduce the size of the training set from 50k to 1k samples (by taking a random subset) and (2) we increase the scale of the weight initialization distribution (by multiplying the initial weights, sampled with Kaiming uniform initialization, by a constant > 1).

The choice of large initializations is justified by [31–33] which find large initializations overfit data easily but prone to poor generalization. Relevant to this, initialization scale is found to regulate "kernel" vs "rich" learning regimes in networks [34].

With these modifications to training set size and initialization scale, we train a depth-3 width-200 MLP with ReLU activations with the AdamW optimizer. We use MSE loss with one-hot targets, rather than cross-entropy. With this setup, we find that the network quickly fits the train set, and then much later in training validation accuracy improves, as shown in Figure 8a. This closely follows the stereotypical grokking learning, first observed in algorithmic datasets.

With this setup, we also compute a phase diagram over the model weight decay and the last layer learning rate. See Figure 8b. While in MLPs it is less clear what parts of the network to consider the "encoder" vs the "decoder", for our purposes here we consider the last layer to be the "decoder" and vary its learning rate relative to the rest of the network. The resulting phase diagram has some similarity to Figure 7. We observe a "confusion"phase in the bottom right (high learning rate and high weight decay), a "comprehension" phase bordering it, a "grokking" phase as one decreases weight decay and decoder learning rate, and a "memorization" phase at low weight decay and low learning rate. Instead of an accuracy threshold of 95%, we use a threshold of 60% here for validation accuracy for runs to count as comprehension or grokking. This phase diagram demonstrates that with sufficient regularization, we can again "de-grok" learning.

We also investigate the effect of training set size on time to generalization on MNIST. We find a result similar to what Power et al. [1] observed, namely that generalization time increases rapidly once one drops below a certain amount of training data. See Figure 20.

K Lottery Ticket Hypothesis Connection

In Figure 21, we show the projection of the learned embeddings after generalization to their first two principal components. Compared to the projection at initialization, structure clearly emerges in embedding space when the neural network is able to generalize ($> 99\%$ validation accuracy). What is intriguing is that the projection of the embeddings at initialization to the principal components of the embeddings at generalization seem to already contain much of that structure. In this sense, the structured representation necessary for generalization already existed (partially) at initialization. The training procedure essentially prunes other unnecessary dimensions and forms the required parallelograms for generalization. This is a nonstandard interpretation of the lottery ticket hypothesis

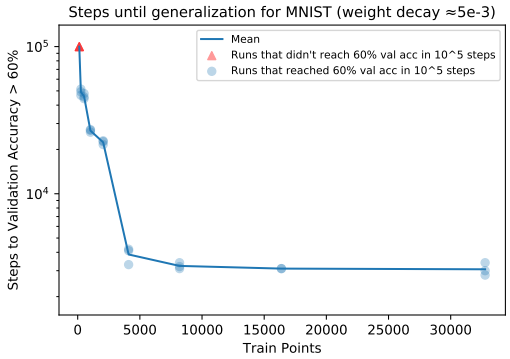


Figure 20: Time to generalize as a function of training set size, on MNIST.

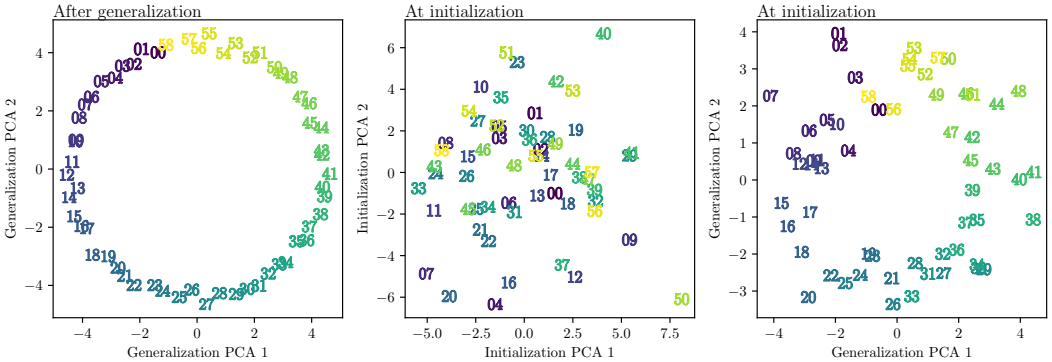


Figure 21: **(Left)** Input embeddings after generalization projected on their first 2 principal components.**(Center)** Input embeddings at initialization projected on their first 2 principal components. **(Right)** Input embeddings at initialization projected on the first 2 principal components of the embeddings after generalization at the end of training (same PCA as the left figure).

where the winning tickets are not weights or subnetworks but instead particular axes or linear combinations of the weights (the learned embeddings).

In Figure 22, we show the original training curves (dashed lines). In solid lines, we recompute accuracy with models which use embeddings that are projected onto the n principal components of the embeddings at the end of training (and back). Clearly, the first few principal components contain enough information to reach 99% accuracy. The first few PCs explain the most variance by definition, however, we note that this is not necessarily the main reason for why they can generalize so well. In fact, embeddings reconstructed from the PCA at the end of training (solid lines) perform better than current highest variance axes (dotted line). This behavior is consistent across seeds.

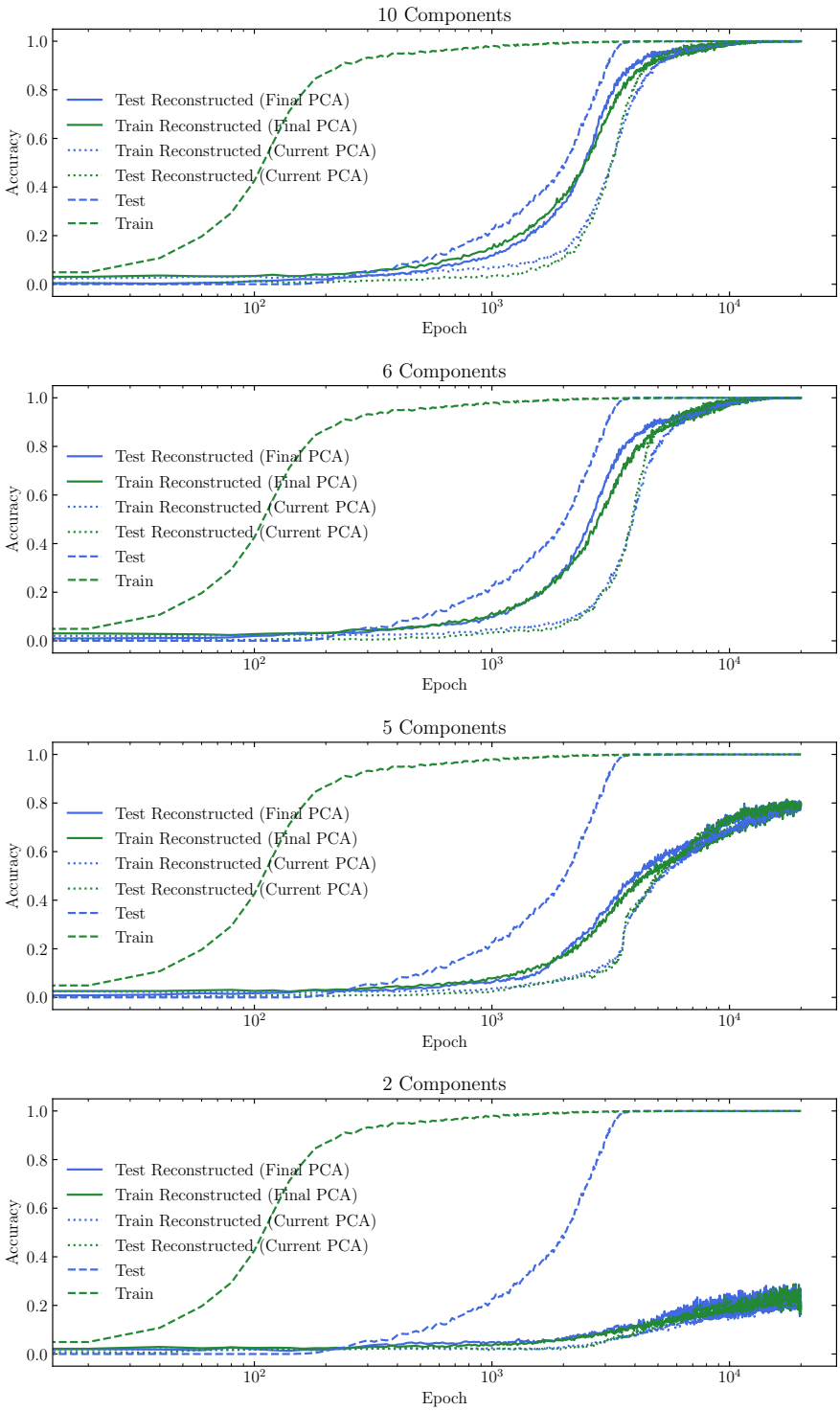


Figure 22: Train and test accuracy computed while using actual embeddings (dashed line) and embeddings projected onto and reconstructed from their first n principal components (dotted lines) and, finally, using embeddings projected onto and reconstructed from the first n PCs of the embeddings at the end of training (solid lines).

L Derivation of the effective loss

In this section, we will further motivate the use of our effective loss to study the dynamics of representation learning by deriving it from the gradient flow dynamics on the actual MSE loss in linear regression. The loss landscape of a neural network is in general nonlinear, but the linear case may shed some light on how the effective loss can be derived from actual loss. For a sample $\mathbf{r}$ (which is the sum of two embeddings $\mathbf{E}_i$ and $\mathbf{E}_j$), the prediction of the linear network is $D(\mathbf{r}) = \mathbf{A}\mathbf{r} + \mathbf{b}$. The loss function is ($\mathbf{y}$ is its corresponding label):

$$\ell = \underbrace{\frac{1}{2}|\mathbf{A}\mathbf{r} + \mathbf{b} - \mathbf{y}|^2}_{\ell_{\text{pred}}} + \underbrace{\frac{\gamma}{2}||\mathbf{A}||_F^2}_{\ell_{\text{reg}}}, \tag{35}$$

where the first and the second term are prediction error and regularization, respectively. Both the model $(\mathbf{A}, \mathbf{b})$ and the input $\mathbf{r}$ are updated via gradient flow, with learning rate η_A and η_x, respectively:

$$\frac{d\mathbf{A}}{dt} = -\eta_A \frac{\partial \ell}{\partial \mathbf{A}}, \frac{d\mathbf{b}}{dt} = -\eta_A \frac{\partial \ell}{\partial \mathbf{b}}, \frac{d\mathbf{r}}{dt} = -\eta_x \frac{\partial \ell}{\partial \mathbf{r}}. \tag{36}$$

Inserting ℓ into the above equations, we obtain the gradient flow:

$$\begin{aligned} \frac{d\mathbf{A}}{dt} &= -\eta_A \frac{\partial \ell}{\partial \mathbf{A}} = -\eta_A[\mathbf{A}(\mathbf{r}\mathbf{r}^T + \gamma) + (\mathbf{b} - \mathbf{y})\mathbf{r}^T], \\ \frac{d\mathbf{b}}{dt} &= -\eta_A \frac{\partial \ell}{\partial \mathbf{b}} = -\eta_A(\mathbf{A}\mathbf{r} + \mathbf{b} - \mathbf{y}) \\ \frac{d\mathbf{r}}{dt} &= -\eta_x \frac{\partial \ell}{\partial \mathbf{r}} = -\eta_x \mathbf{A}^T(\mathbf{A}\mathbf{r} + \mathbf{b} - \mathbf{y}). \end{aligned} \tag{37}$$

For the $d\mathbf{b}/dt$ equation, after ignoring the $\mathbf{A}\mathbf{r}$ term and set the initial condition $\mathbf{b}(0) = \mathbf{0}$, we obtain analytically $\mathbf{b}(t) = (1 - e^{-2\eta_A t})\mathbf{y}$. Inserting this into the first and third equations, we have

$$\begin{aligned} \frac{d\mathbf{A}}{dt} &= -\eta_A[\mathbf{A}(\mathbf{r}\mathbf{r}^T + \gamma) - e^{-2\eta_A t}\mathbf{y}\mathbf{r}^T], \\ \frac{d\mathbf{r}}{dt} &= \underbrace{-\eta_x \mathbf{A}^T\mathbf{A}\mathbf{r}}_{\text{internal interaction}} + \underbrace{\eta_x e^{-2\eta_A t}\mathbf{A}^T\mathbf{y}}_{\text{external force}}. \end{aligned} \tag{38}$$

For the second equation on the evolution of $d\mathbf{r}/dt$, we can artificially decompose the right hand side into two terms, based on whether they depend on the label $\mathbf{y}$. In this way, we call the first term "internal interaction" since it does not depend on $\mathbf{y}$, while the second term "external force". Note this distinction seems a bit artificial from a mathematical perspective, but it can be conceptually helpful from a physics perspective. We will show below the internal interaction term is important for representations to form. Because we are interested in how two samples interact, we now consider another sample at $\mathbf{r}'$, and the evolution becomes

$$\begin{aligned} \frac{d\mathbf{A}}{dt} &= -\eta_A[\mathbf{A}(\mathbf{r}\mathbf{r}^T + \mathbf{r}'\mathbf{r}'^T + 2\gamma) - e^{-2\eta_A t}\mathbf{y}(\mathbf{r} + \mathbf{r}')^T], \\ \frac{d\mathbf{r}}{dt} &= -\eta_x \mathbf{A}^T\mathbf{A}\mathbf{r} + \eta_x e^{-2\eta_A t}\mathbf{A}^T\mathbf{y}, \\ \frac{d\mathbf{r}'}{dt} &= -\eta_x \mathbf{A}^T\mathbf{A}\mathbf{r}' + \eta_x e^{-2\eta_A t}\mathbf{A}^T\mathbf{y}. \end{aligned} \tag{39}$$

Subtracting $d\mathbf{r}/dt$ by $d\mathbf{r}'/dt$ and setting $\mathbf{r}' = -\mathbf{r}$, the above equations further simply to

$$\begin{aligned} \frac{d\mathbf{A}}{dt} &= -2\eta_A \mathbf{A}(\mathbf{r}\mathbf{r}^T + \gamma), \\ \frac{d\mathbf{r}}{dt} &= -\eta_x \mathbf{A}^T\mathbf{A}\mathbf{r}. \end{aligned} \tag{40}$$

The second equation implies that the pair of samples interact via a quadratic potential $U(\mathbf{r}) = \frac{1}{2}\mathbf{r}^T\mathbf{A}^T\mathbf{A}\mathbf{r}$, leading to a linear attractive force $f(r) \propto r$. We now consider the adiabatic limit where $\eta_A \to 0$.

The adiabatic limit Using the standard initialization (e.g., Xavier initialization) of neural networks, we have $\mathbf{A}_0^T \mathbf{A}_0 \approx \mathbf{I}$. As a result, the quadratic potential becomes $U(\mathbf{r}) = \frac{1}{2}\mathbf{r}^T\mathbf{r}$, which is time-independent because $\eta_A \to 0$. We are now in the position to analyze the addition problem. For two samples $\mathbf{x}^{(1)} = \mathbf{E}_i + \mathbf{E}_j$ and $\mathbf{x}^{(2)} = \mathbf{E}_m + \mathbf{E}_n$ with the same label ($i + j = m + n$), they contribute to an interaction term

$$U(i, j, m, n) = \frac{1}{2}|\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n|_2^2. \tag{41}$$

Averaging over all possible quadruples in the training dataset D, the total energy of the system is

$$\ell_0 = \sum_{(i,j,m,n)\in P_0(D)} \frac{1}{2}|\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n|_2^2 / |P_0(D)|, \tag{42}$$

where $P_0(D) = \{(i, j, m, n)|i + j = m + n, (i, j) \in D, (m, n) \in D\}$. To make it scale-invariant, we define the normalized Hamiltonian Eq. (42) as

$$\ell_{\text{eff}} = \frac{\ell_0}{Z_0}, \quad Z_0 = \sum_i |\mathbf{E}_i|_2^2 \tag{43}$$

which is the effective loss we used in Section 3.2.