# Understanding Simplicity Bias towards Compositional Mappings via Learning Dynamics

**Yi Ren**
University of British Columbia
renyi.joshua@gmail.com

**Danica J. Sutherland**
University of British Columbia & Amii
dsuth@cs.ubc.ca

## Abstract

Obtaining compositional mappings is important for the model to generalize well compositionally. To better understand when and how to encourage the model to learn such mappings, we study their uniqueness through different perspectives. Specifically, we first show that the compositional mappings are the simplest bijections through the lens of coding length (i.e., an upper bound of their Kolmogorov complexity). This property explains why models having such mappings can generalize well. We further show that the simplicity bias is usually an intrinsic property of neural network training via gradient descent. That partially explains why some models spontaneously generalize well when they are trained appropriately.

## 1 Introduction

There is a general belief that having more compositional representations is the key to improving compositional generalization [14]. Although there are many specifically designed algorithms (e.g., [19]) and network structures (e.g., [13, 24]) with this goal, methods to reliably obtain such representations in a variety of settings remain elusive. On the other hand, it has been repeatedly shown that compositional generalization ability can spontaneously emerge from standard supervised learning tasks [e.g. 18] or under repeated self-distillation training [e.g. 22]. In a recent position paper, Huh et al. [10] argued based on a variety of previous results that *deep networks naturally adhere to Occam's razor, implicitly favoring simple solutions that fit the data*.

To help understand the relationships between compositional mappings and the network's inherent bias, we first argue that compositional mappings are generally the simplest bijections to learn. Specifically, assuming the data-generating process is compositional, compositional mappings are the simplest (i.e. lowest-complexity). Next, we demonstrate experimentally that neural networks naturally favor simpler mappings through the process of gradient descent. This simplicity bias can be intuitively explained by the mutual influence of learning different samples, building on prior analyses [6, 23]. Although our experiments and discussions are restricted to a simplified setting (i.e., toyish input signals and only two factors), and in practice, some combinations of features are harder to learn [17], we believe the framework of this paper can help pave the way toward analyzing why some networks naturally achieve great compositional generalization ability under suitable learning tasks, and hopefully, help inspire more effective algorithms to exploit this simplicity bias for more effective generalization. The code for all experiments can be found in https://github.com/Joshua-Ren/simplicity_bias_learning_dynamics.

## 2 Compositional Mappings are the Simplest Bijections

In this section, we first specify that the mapping from the ground-truth characters of the input signal to the learned representation is the key to analyzing the compositional generalization problem. We

NeurIPS 2024 Workshop on Compositional Learning: Perspectives, Methods, and Paths Forward.

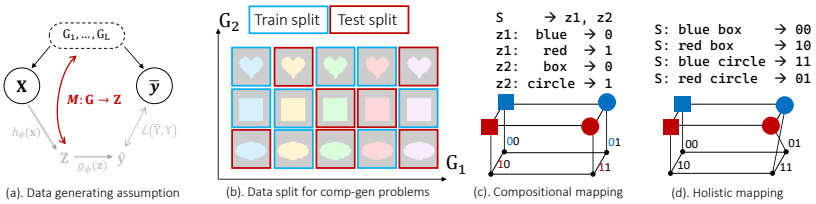



Figure 1: The compositional generalization problem (**a, b**) and two types of bijections (**c, d**).

then describe the uniqueness of compositional mappings by analyzing their Kolmogorov complexity. In short, we show that compositional mappings are simpler than non-compositional bijections.

We start with a typical compositional generalization problem, where the input signal $\mathbf{x}$ and label(s) $\bar{\mathbf{y}}$ are determined by several ground-truth generating factors $\mathbf{G} \triangleq [G_1, \dots, G_L] \in \mathcal{G}$, as in Figure 1-(a). All $G_i$ are discrete variables with $V$ possible values.[1] In this problem, the model only learns from data samples generated from a subset of all possible $\mathbf{G}$ and tries to generalize to unseen $\mathbf{G}$. Consider a colored-dSprite [16] example provided in Figure 1-(b), where $G_1$ and $G_2$ are the shape and color of the objects. With the provided train/test split, the model must first learn the concepts of "red" and "box" separately, and then compositionally combine them to make a correct prediction on the "red box" in the test set. Suppose the model uses a deep neural network $h_\theta : \mathcal{X} \to \mathcal{Z}$ to extract features from the input signals, where the representation space $\mathcal{Z}$ is also an $L \times V$ discrete space. Assuming each $\mathbf{x}$ has a unique $\mathbf{G}$, we can then define the mapping between ground-truth factors and the extracted representations as $M : \mathcal{G} \to \mathcal{Z}$. In order to generalize well, a good mapping $M$ should satisfy:

- $M$ should be a bijection, otherwise, two $\mathbf{x}$ with different $\mathbf{G}$ will be mapped to the same $\mathbf{z}$, which makes it impossible for the task head to separate these two $\mathbf{x}$ from their $\mathbf{z}$;
- $M$ should ensure different $z_i$ consistently encode different $G_j$ in a non-overlapping way, so the model can generalize appropriately to novel combinations.

To get a clearer picture of this second requirement, we consider a "Toy256" example, where $G_1 = \{\texttt{blue},\texttt{red}\}$, $G_2 = \{\texttt{box},\texttt{circle}\}$, and $\mathbf{z} = \{00, 01, 10, 11\}$. There are $4^4 = 256$ possible mappings for $M$, of which $4! = 24$ are bijections. Among all these bijections, only $2! \times 2^2 = 8$ are compositional. To get such a mapping, we should first assign color and shape to different $z_i$, and then separately assign 0 or 1 to represent different values for each attribute. Following Ren et al. [22], we call those non-compositional bijections "holistic mappings". As demonstrated in Figure 1-(c, d), compositional mappings can be decomposed into some shared rules while holistic mappings cannot, as illustrated by corresponding CFGs (context-free grammars) at the top of the figure.

Since the compositional mappings are generated in a systematic way, intuitively they are simpler and can be described with less effort. We can use group theory to define the simplicity of a bijection more formally. When generating a compositional mapping, we first select $z_i$ for each $G_j$ in a non-overlapping way. Such a process can be represented by an element in a symmetry group $S_L$. We then build an injection from the paired $G_j$ to $z_i$, by which each possible value of the $j$-th attribute is encoded by different "words" in $z_i$. In short, assuming both $\mathbf{G}$ and $\mathbf{z}$ are $L \times V$ grid spaces, any compositional mapping can be described by an element in the group $S_V^L \rtimes S_V \in S_{V^L}$, where $\rtimes$ is the semidirect product in group theory.

This implies why a compositional mapping has a lower Kolmogorov complexity upper bound[2] than an arbitrary non-compositional one among all bijections. From the definition of the symmetry group, we know each element in $S_{V^L}$ can be represented by a permutation matrix of size $V^L$. As there is only one "1" in each row and column of a permutation matrix, any permutation matrix can be uniquely represented by a permuted sequence of length $V^L$. Specifically, assume we have a sequence of natural numbers $\{1, 2, ..., V^L\}$, each permuted sequence $\mathsf{Perm}(\{1, 2, ..., V^L\})$ represents a distinct

[1] Continuous $G_i$ can be quantized to fit our analysis; factors with fewer than $V$ values can randomly assign features or pad zeros.

[2] While it is not possible to lower-bound the Kolmogorov complexity of any particular mapping without fixing the underlying Turing machine, a counting argument shows that *most* non-compositional bijections must have higher complexity. The complexity we described here is actually the $\mathcal{K}(f)$ discussed in [4], which plays an important role of defining the representational compositionality.

permutation matrix, and hence represents a distinct bijection from $\mathbf{G}$ to $\mathbf{z}$. In other words, we can encode one bijection from $\mathbf{G}$ to $\mathbf{z}$ using a sequence of length $V^L$, i.e., $\mathsf{Perm}(\{1, 2, ..., V^L\})$, and bound the corresponding Kolmogorov complexity (in bits) as

$$\mathcal{K}(\text{bijection}) \leq V^L \cdot \log_2 V^L = V^L \cdot L \cdot \log_2 V, \tag{1}$$

As an arbitrary bijection from $\mathbf{G}$ to $\mathbf{z}$ doesn't have any extra information to improve the coding efficiency, Equation (1) provides an upper bound of the minimal Kolmogorov complexity.

On the contrary, as each compositional mapping can be represented by an element in $S_V^L \rtimes S_L$, we can encode the mappings more efficiently. Specifically, we need to first use $L$ sequences with length $V$, i.e., $\mathsf{Perm}(\{1, 2, ..., V\})$, to represent the assignment of "words" for each $z_i$. After that, we need one sequence of length $L$, i.e., $\mathsf{Perm}(\{1, 2, ..., L\})$ to encode the assignment between $z_i$ and $G_j$. Ignoring the necessary separators for these $L + 1$ sub-sequences, the corresponding Kolmogorov complexity is then bounded as

$$\mathcal{K}(\text{comp}) \leq V \cdot \log_2 V + L \cdot \log_2 L. \tag{2}$$

To compare the Kolmogorov complexity of different mappings, we can define a ratio as $\gamma \triangleq \frac{\mathcal{K}(\text{bijection})}{\mathcal{K}(\text{comp})}$. Obviously, when $L \leq V$, $\gamma \geq \frac{V^{L-1} \cdot L}{2}$, which is larger than 1 as long as $L, V \geq 2$. When $L > V$, $\gamma \geq \frac{V^L \log_2 V}{2 \log_2 L}$, which is also larger than 1 when $L, V \geq 2$.

Note that there might be some partially compositional mappings. For example, we can have a mapping with $z_{i \leq 10}$ sharing the reused rules while other $z_{i>10}$ doesn't. Then this type of mapping can be represented by an element in $S_V^{10} \rtimes S_{10} \rtimes S_{V^{L-10}}$. As a mapping in this subset shares 10 common rules, its Kolmogorov complexity is between $\mathcal{K}(\text{bijection})$ and $\mathcal{K}(\text{comp})$. Intuitively, *for all bijections*, smaller $\mathcal{K}(\cdot)$ means higher compositionality.

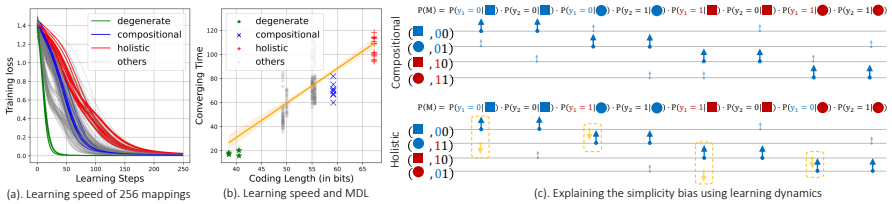
## 3 Simpler Mappings are Learned Faster

The analysis above links the concepts of compositionally, simplicity, and Kolmogorov complexity under an idealized setting, which also aligns well with many related works. For example, Huh et al. [10] claim that simplicity bias is the key for the models in different modalities converging to a shared representation space that is similar to the ground truth. Goldblum et al. [5] also link Kolmogorov complexity to PAC-Bayes generalization bounds. This supports the idea that having more compositional mappings greatly benefits the model's generalization ability. This section further demonstrates that a neural network naturally favors such simpler mappings. We will first verify this claim by experiments under manual settings, and then provide a detailed explanation of why such a tendency exists using learning dynamics.

Specifically, we claim that simpler mappings are learned faster by a neural network trained using GD. To verify this, we consider a multi-label classification problem and create 256 different datasets (each only contains 4 examples) for each $M$ in our "Toy256" setting. For example, the dataset for the mapping in Figure 1-(c) should be {(`blue box`, 00), (`blue circle`, 01), (`red box`, 10), (`red circle`, 11)}, where the label "01" means $\bar{y}_1 = 0$ and $\bar{y}_2 = 1$. We then randomly initialize a neural network as our $h_\theta$ and concatenate two Sigmoid functions as our $g_\phi$. With the same initialization and all hyper-parameters, we train the network to convergence for each dataset. We also consider different input signals (images and dense random vectors), network structures (MLP and ResNet), loss functions (cross-entropy and mean square error), and optimizers (standard SGD and Adam). Please refer to Appendix C for more details.

Figure 2-(a) shows the training curves of 256 runs in our default setting. Since the only difference among these runs is the dataset generated by different mappings, it is safe to conclude that the difference in their learning speed is caused by *the inherent bias of the model's learning behavior* on this problem. From this figure, we see compositional mappings are learned faster than holistic ones. However, some mappings are learned even faster, which makes sense because those non-bijection mappings contain degenerate components, i.e., two or more objects are mapped to the same $\mathbf{z}$, which means they are simpler. That also explains why the four degenerate mappings, which map all four objects to the same $\mathbf{z}$, are learned fastest among all 256 mappings.

To further verify this, we quantify the learning speed using the concept of "convergence time," i.e., the area under the learning curve. A smaller convergence time means the mapping is learned faster. This




(a). Learning speed of 256 mappings (b). Learning speed and MDL (c). Explaining the simplicity bias using learning dynamics

Figure 2: The evidence and explanations of the claim that *simpler mappings are learned faster*. The blue arrows in the last panel mean when learning the given example, the model increases its confidence in the corresponding prediction. The increase of the darker arrows is stronger than that of lighter ones because the confidence changes for the lighter ones are indirectly caused by the "elasticity" of the neural network [7]. For holistic mapping, the yellow arrows pointing down mean the corresponding confidence decreases after learning the example.

metric is similar to the C-score of Jiang et al. [11], which describes a training example's difficulty. Also, if the model is trained with cross-entropy loss and all examples only appear once, this metric is the compression rate for the entire dataset [20]. These works also hint that learning speed is deeply related to compression, simplicity, and generalization ability.

Another quantity we want to explicitly calculate is bounds on each mapping's Kolmogorov complexity. Since the mapping space studied in our Toy256 setting is simple enough, we can first create the CFGs for each mapping and then convert them to a piece of description sequence using the method provided by Kirby et al. [12]. After that, we can use Huffman coding [3] and calculate the coding length in bits for each mapping. Please refer to Appendix B for more details. In short, a smaller coding length means the mapping is simpler. It is clear in Figure 2-(b) that compositional mappings are the simplest bijections. Note that all the non-bijection mappings contain degenerate components, hence are simpler than bijections. The figure also clearly demonstrates that the learning speed is strongly correlated to the coding length (with $\rho > 0.65$ and $p < 10^{-30}$), matching our hypothesis well. This trend is consistent across various settings, as demonstrated in Appendix D.

The results above bridge the simplicity bias to the model's learning behavior, where the latter can be further explained using learning dynamics [21, 23]. Remember our model generates probabilistic predictions on both $y_1$ and $y_2$ using Sigmoid functions. Then, we can directly write down the predicted probability of each mapping as a product of eight terms, as in the top line of Figure 2-(c). In this figure, we demonstrate how the model's confidence of different $M$ is updated when learning specific training samples. For example, in the first row of the compositional mapping in the figure, the model learns $(\texttt{blue box}, 00)$. Then the corresponding $P(y_1 = 0 \mid \texttt{blue box})$ and $P(y_2 = 0 \mid \texttt{blue box})$ are significantly improved, since they are directly updated by learning this example. Furthermore, as the neural network has local elasticity [7], the model's predictions on those "similar" (measured using Hamming distance) input examples would also be indirectly updated (represented by the small arrows in the figure). As a result, the model's confidence on `blue circle` and `red box`, which share one attribute with the learned `blue box`, are influenced more by this update. Furthermore, since a compositional mapping always utilizes consistent values to represent the same attribute (e.g., $z_1 = 0$ always encodes the blue color), all the direct and indirect updates align well with the compositional mapping. That is why the training loss of such mappings decreases faster. On the contrary, for a holistic mapping, we observe several contradictions between the direct and indirect updates: learning it requires the model to use more updates to counteract those negative indirect influences. That explains why compositional mappings are usually learned faster than holistic ones by a neural network.

## 4 Conlusion

This paper first shows that compositional mappings are the simplest bijections in terms of Kolmogorov complexity or coding length. Then, using experiments and analysis from the learning dynamics perspective, the paper claims that the simplicity bias (in terms of coding length) is inherent in neural network training for typical architectures using gradient descent. Although the settings in the paper are simple, the theoretical formulation and analysis have the potential to be extended to more practical problems.

## References


[1] Jacob Andreas. "Measuring Compositionality in Representation Learning." *International Conference on Learning Representations*. 2019.

[2] Henry Brighton and Simon Kirby. "Understanding linguistic evolution by visualizing the emergence of topographic mappings." *Artificial life* 12.2 (2006), pages 229–242.

[3] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[4] Eric Elmoznino, Thomas Jiralerspong, Yoshua Bengio, and Guillaume Lajoie. "A Complexity-Based Theory of Compositionality." *arXiv preprint arXiv:2410.14817* (2024).

[5] Micah Goldblum, Marc Finzi, Keefer Rowan, and Andrew Gordon Wilson. *The No Free Lunch Theorem, Kolmogorov Complexity, and the Role of Inductive Biases in Machine Learning*. 2023. arXiv: 2304.05366 [cs.LG].

[6] Shangmin Guo, Yi Ren, Stefano V Albrecht, and Kenny Smith. "lpNTK: Better Generalisation with Less Data via Sample Interaction During Learning." *The Twelfth International Conference on Learning Representations*. 2024.

[7] Hangfeng He and Weijie Su. "The Local Elasticity of Neural Networks." *International Conference on Learning Representations*. 2020.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pages 770–778.

[9] Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. *Towards a definition of disentangled representations*. 2018. arXiv: 1812.02230 [cs.LG].

[10] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. "The platonic representation hypothesis." *International Conference on Machine Learning* (2024).

[11] Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and Michael C Mozer. "Characterizing Structural Regularities of Labeled Data in Overparameterized Models." *International Conference on Machine Learning*. PMLR. 2021, pages 5034–5044.

[12] Simon Kirby, Monica Tamariz, Hannah Cornish, and Kenny Smith. "Compression and communication in the cultural evolution of linguistic structure." *Cognition* 141 (2015), pages 87–102.

[13] Yen-Ling Kuo, Boris Katz, and Andrei Barbu. "Compositional networks enable systematic generalization for grounded language understanding." *arXiv preprint arXiv:2008.02742* (2020).

[14] Ziming Liu, Ouail Kitouni, Niklas S Nolte, Eric Michaud, Max Tegmark, and Mike Williams. "Towards understanding grokking: An effective theory of representation learning." *Advances in Neural Information Processing Systems* 35 (2022), pages 34651–34663.

[15] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[16] Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. *dSprites: Disentanglement testing Sprites dataset*. https://github.com/deepmind/dsprites-dataset/. 2017.

[17] Milton Montero, Jeffrey Bowers, Rui Ponte Costa, Casimir Ludwig, and Gaurav Malhotra. "Lost in Latent Space: Examining failures of disentangled models at combinatorial generalisation." *Advances in Neural Information Processing Systems* 35 (2022), pages 10136–10149.

[18] 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).

[19] Linlu Qiu, Peter Shaw, Panupong Pasupat, Paweł Krzysztof Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. "Improving compositional generalization with latent structure and data augmentation." *arXiv preprint arXiv:2112.07610* (2021).

[20] Jack Rae. *Compression for AGI*. 2023. URL: https://www.youtube.com/watch?v=dO4TPJkeaaU.

[21] Yi Ren, Shangmin Guo, Wonho Bae, and Danica J. Sutherland. "How to prepare your task head for finetuning." *The Eleventh International Conference on Learning Representations*. 2023.

[22] Yi Ren, Shangmin Guo, Matthieu Labeau, Shay B. Cohen, and Simon Kirby. "Compositional languages emerge in a neural iterated learning model." *International Conference on Learning Representations*. 2020.

[23] Yi Ren, Shangmin Guo, and Danica J. Sutherland. "Better Supervisory Signals by Observing Learning Paths." *International Conference on Learning Representations*. 2022.

[24] Yi Ren, Samuel Lavoie, Michael Galkin, Danica J Sutherland, and Aaron C Courville. "Improving compositional generalization using iterated learning and simplicial embeddings." *Advances in Neural Information Processing Systems* 36 (2024).

[25] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. "Understanding deep learning (still) requires rethinking generalization." *Communications of the ACM* 64.3 (2021), pages 107–115.

# A Compositional Representation and Platonic Representation Hypothesis

This appendix tries to uncover the implicit relationship between compositional representation learning (usually studied in a manually toyish setting) and the Platonic representation hypothesis (proposed in [10], experimentally verified on many SOTA large vision and language models). Specifically, we focus on the following three aspects: 1.) the underlying assumption of the existence of $\mathbf{G}$; 2.) the measuring metrics; 3.) the converging pressures. Our analysis hints that more advanced compositional generalization ability could also be achieved if we design appropriate learning systems following the fundamental principles demonstrated in Huh et al. [10].

## A.1 The Underlying Assumption of the Ground-truth Generating Mechanism

The main claim of Huh et al. [10] is that there exists a unique ground-truth idealized world (i.e., the $\mathbf{G}$ in our paper), from which, all observations in different modalities are its projections. A deep learning system, which learns from these projections and then generalizes to related tasks, are trying to uncover such ground truth. As the models in different modalities become stronger, their representations (i.e., $\mathbf{z} \in \mathcal{Z}$ in our paper) are more aligned, because they all tend to converge to the ground truth $\mathbf{G}$.

In our paper, we also assume the existence of $\mathbf{G}$ and consider both input signal $\mathbf{x}$ and labels $\bar{y}$ are determined by it. By treating the mapping from $\mathcal{G} \to \mathcal{Y}$ as a special projection for a specific modality, our Figure 1-(a) becomes the upper part of the Figure 1 in [10]. The goal of a compositional representation learning task is to recover a good representation space that is similar to the ground truth, and hence generalize well to unseen combinations of attributes. This also aligns with the claim that "models generalize better on different modalities align better to the ground-truth" in [10].

## A.2 Measuring Metrics: Kernel Alignment, Disentanglement, and Topological Similarity

To mathematically describe the representation's convergence, Huh et al. [10] use three steps to define a metric called *Kernel Alignment* to measure the similarity between two representation spaces.

1. A *representation*, which maps the input signal to a dense representation space, is a function $f : \mathcal{X} \to \mathbb{R}^n$. Note that our $h_\theta(\mathbf{x})$ plays a similar role;
2. A *kernel*, $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, characterize the similarity between two elements in $\mathcal{X}$. In a dense representation case, the inner product is usually applied, i.e., $K(x_i, x_j) = \langle f(x_i), f(x_j) \rangle, K \in \mathcal{K}$. Our paper consider Hamming distance, because our $\mathbf{G}$ and $\mathbf{z}$ are all categorical variables;
3. A *kernel-alignment metric* $m : \mathcal{K} \times \mathcal{K} \to \mathbb{R}$, measures the similarity between two kernels.

For the third-level measurement, Huh et al. [10] use Centered Kernel Distance (CKA), a kernel-alignment metric throughout their paper. Actually, many related works in compositional generalization also have similar measurements, e.g., the topological similarity proposed in [2]:

$$\mathsf{Topsim}(\mathbf{G}, \mathbf{z}) \triangleq \mathsf{Corr}\left(d_G(\mathbf{G}^{(i)}, \mathbf{G}^{(j)}), d_z(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})\right), \tag{3}$$

This definition also follows three steps: $\mathbf{z}$ are representations generated by feeding $\mathbf{x}$ to $h_\theta$, $d_G$ and $d_z$ are distance measurements (or kernel in the second step above) for space $\mathcal{G}$ and $\mathcal{Z}$, and $\mathsf{Corr}(\cdot, \cdot)$ is the Spearman's correlation measuring the relationship between the output of two functions (kernels). In short, *higher topological similarity means similar objects in $\mathcal{G}$ are mapped to similar positions in $\mathcal{Z}$*. If we consider $\mathcal{G}$ as another modality of the projected ground truth, the topological similarity is just a special kernel-alignment metric used in [10]. Also, some other measurements of compositionality like TRE (Tree Reconstruction Error, [1]), representation disentanglement [9], etc., also follow this principle generally. In summary, since the main measurement of the Platonic representation hypothesis and the compositional representation learning are essentially identical, we can draw more parallels between these two seemingly distinct fields in the future.

## A.3 The Converging Pressures

Section 3 of [10] proposes three pressures that lead the model's representations to converge to the ground truth. We could also find some counterparts in the field of compositional representation

learning. The first one is task generality, which means requiring the model to solve more tasks using the same representations leads to better convergence to the ground truth. We can also draw a similar conclusion from the experiments in [24], in which the authors show that the quantity and diversity of the learning tasks play an important role in achieving good systematic representations.

The second pressure is the model's capacity. Because bigger models are more likely to converge to a shared representation than smaller ones [10]. This partially aligns with our "unambiguous" requirement on the mapping $M : \mathcal{G} \to \mathcal{Z}$ discussed in Section 2. This requires the model to be capable enough to have perfect training accuracy, otherwise, important information about the task labels would be lost. However, Zhang et al. [25] demonstrates that achieving perfect training accuracy (even if the label is purely random noise) is not a hard task for a deep neural network, while the optimal mapping can be even simpler than the noisy-label dataset. As a result, it might be more useful to consider the influence of the model's capacity from a dynamical perspective (e.g., optimization), which is usually neglected when studying the relationship between generalization and model capacity.

The last pressure, i.e., the simplicity bias, is the one we discussed most in our paper. We showed that such simplicity can be understood as the lower bound of Kolmogorov complexity, which also measures how compressible the mapping is. Requiring simpler mappings also aligns well with Occam's Razor, which might be an important direction for our future exploration. The explanation of learning dynamics is a good starting point for combining this model-agnostic measurement (i.e., Kolmogorov complexity) with the model's inherent bias. Actually, Figure 1 of Zhang et al. [25] shows that under the same setting, the model learns the random noise labels slower than the ground truth labels. This phenomenon could also be explained by the interactions between training examples used in this paper. Since the random noisy-label dataset would have two semantically similar $\mathbf{x}$ with very different labels, the direct and indirect updates would have more contradictions, similar to the holistic updating case in Figure 2.

## B Coding Length and Topological Similarity for the Mappings in Toy256

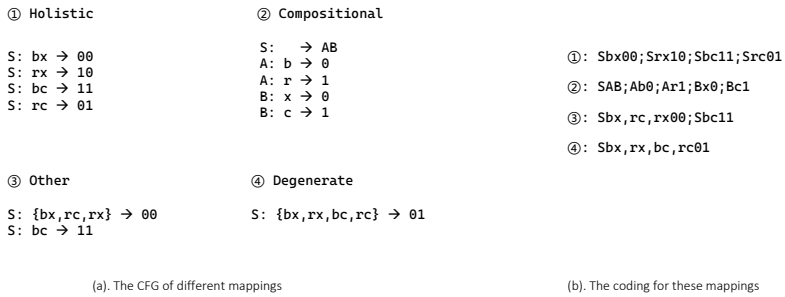



Figure 3: Four typical mappings studied in this paper and their coding strings.

The main target of this paper is to show that the simplicity bias is inherent in neural network training. Inspired by many related works on compositional generalization, we believe the Kolmogorov complexity is a perfect measurement for the simplicity of a mapping. However, it is well known that Kolmogorov complexity is usually hard to calculate and people typically use the minimum description length (MDL) under specific constraints as its approximation [15].

To experimentally show the correlation between learning speed and simplicity, we use the coding method provided in Kirby et al. [12] to calculate the coding length for all 256 mappings. Note that such a coding mechanism might not be optimal (i.e., its length is not the MDL). Hence we only call this measurement coding length (CL) throughout the paper.

The calculation of CL involves three steps. First, we convert all mappings using a *compressed* CFG, as illustrated in Figure 3-(a). As all the attributes studied here are categorical variables, we use `b, x, r, c` to represent `blue, box, red, circle`, respectively. Then, we delete all the redundant characters and generate the unique coding sequence for each CFG, as in Figure 3-(b). The special characters "S" and "`;`" denote the starting and ending of one piece of rule and "`,`" is used to separate different objects sharing the same message. Finally, the coding length in bits of a mapping $M$ is

calculated using

$$\mathsf{CL}(M) = -\sum_{i=1}^{|\mathsf{Seq}(M)|} \log_2 p(s_i); \quad p(s_i) = \frac{\mathsf{Cnt}(s_i)}{|\mathsf{Seq}(M)|} \tag{4}$$

where $\mathsf{Seq}(M)$ converts the mapping to a coding sequence and $p(s_i)$ is the probability of the $i$-th character in this code squence. For example, for the degenerate sequence `[Sbx,rx,bc,rc01]`, $p(s_1) = p(s_7) = \frac{\mathsf{Cnt(b)}}{|\mathsf{Seq}(M)|} = \frac{2}{14}$.

Furthermore, we also verify the correlation between the learning speed and topological similarity defined in Equation (3). Generally, the Topsim for perfect compositional mappings equals one. Note that the Topsim of pure degenerate mappings that maps every object to the same $\mathbf{z}$ is not well defined in Equation (3), because the Spearman's correlation is calculated by $r_s = \frac{\mathsf{Cov}(R[\mathbf{z}],R[\mathbf{G}])}{\sigma_{R[\mathbf{z}]}\sigma_{R[\mathbf{G}]}}$. In a degenerate mapping, the six pair-wise distances of all four possible $\mathbf{z}$ are all zeros, which means $R[\mathbf{z}] =$ `[0,0,0,0,0,0]`. Hence the corresponding $r_s$ becomes $\frac{0}{0}$. However, following the definition of topological similarity that high Topsim mappings tend to assign similar $\mathbf{z}$ to $\mathbf{x}$ with similar $\mathbf{G}$, we just define the Topsim of those degenerate mappings as one.

## C Experimental Settings

We consider various settings for the Toy256 examples to verify that the simplicity bias discussed in this paper is general enough. For the input signals, we first consider two types of one-hot concatenation vectors. One is the concatenation of two 2-dim vectors (`OHT2` for short). For example, `blue box` and `red circle` are encoded as `[0101]` $\cdot W_{4\times d}$ and `[1010]` $\cdot W_{4\times d}$, respectively, where $W_{4\times d}$ is a randomly initialized matrix fixed for all 256 mappings. Another setting is `OHT3`, which considers a redundant dimension for each attribute, where `blue box` and `red circle` are encoded as `[010010]` $\cdot W_{6\times d}$ and `[100100]` $\cdot W_{6\times d}$, respectively. We also consider the vision input, where the image is sampled and colored from the dSprite dataset, as illustrated in Figure 1.

We consider different network structures for different input modalities. For the one-hot input, we use an MLP with three hidden layers with a width of 128. For the image input, we consider both a 3-layer MLP and a ResNet9 [8] with narrower hidden layers. The task heads for all the networks are identical: we add two separate linear projection layers with size $h \times 2$ on the output of the backbone. After that, we take Softmax on each of these outputs to generate probabilistic predictions. When calculating the loss function, we consider both cross-entropy (CE) loss and a mean square error (L2) loss, where the latter is calculated between the predicting probability vector and a one-hot distribution of the ground truth labels. When optimizing the network, we consider both stochastic gradient descent (SGD) and Adam. Unless otherwise stated, the learning rate is set to $10^{-3}$, weight decay is $5 * 10^{-4}$, and all other parameters are set to be the default values. Note that all hyper-parameters (including the initialization of the network) are shared for all 256 experiments in each group.

## D More Experimental Results

To visualize the relationship between learning speed and the simplicity of each mapping, we provide three types of visualizations in Figure 4 and Figure 5. The first one is the learning curves of all 256 mappings. It is clear that under most settings, the blue curves (i.e., those for compositional mappings) decay faster than the red ones (the non-compositional bijections). The second one is the scatter plots showing the correlation between the converging time (i.e., the integral under the learning curve) and CL in Equation (4). The third one is the scatter plots showing the correlation between the converging time and topological similarity defined in Equation (3). We also calculate the Pearson correlation in the latter two cases in Table 1. It is clear that in most cases, simpler mappings are indeed learned faster under different settings. One exceptional case is training a ResNet with image input using Adam optimizer. The simplicity bias is even reversed compared with the results using SGD. This phenomenon hints that the bias in DNN's learning is also influenced by the inherent bias of specific network structures and optimizers. We left this for our future work.

Table 1: The statistical correlation between learning speed and simplicity.

| Input | Optim. | Loss | CL-Conv.Time | | Topsim-Conv.Time | | Input | Optim. | Loss | CL-Conv.Time | | Topsim-Conv.Time | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $\rho$ | $p$ | $\rho$ | $p$ | | | | $\rho$ | $p$ | $\rho$ | $p$ |
| **OHT2 MLP** | **SGD** | CE | 0.6475 | 8.1*1e-32 | -0.7101 | 1.4*1e-40 | **Image MLP** | **SGD** | CE | 0.6866 | 5.0*1e-37 | -0.5911 | 1.6*1e-25 |
| | | L2 | 0.5793 | 2.4*1e-24 | -0.7817 | 5.3*1e-54 | | | L2 | 0.5932 | 1.1*1e-25 | -0.6057 | 5.1*1e-27 |
| | **Adam** | CE | 0.6598 | 2.3*1e-33 | -0.5731 | 9.4*1e-24 | | **Adam** | CE | 0.5403 | 8.4*1e-21 | -0.6720 | 5.5*1e-35 |
| | | L2 | 0.5378 | 1.4*1e-20 | -0.7223 | 1.2*1e-42 | | | L2 | 0.4433 | 9.5*1e-14 | -0.6585 | 3.4*1e-33 |
| **OHT3 MLP** | **SGD** | CE | 0.5976 | 3.5*1e-26 | -0.7963 | 2.2*1e-57 | **Image ResNet** | **SGD** | CE | 0.6711 | 7.4*1e-35 | -0.2619 | 2.2*1e-5 |
| | | L2 | 0.6386 | 9.8*1e-31 | -0.7311 | 4.5*1e-44 | | | L2 | -0.015 | 0.8159 | -0.0297 | 0.6358 |
| | **Adam** | CE | 0.5672 | 3.4*1e-23 | -0.6418 | 4.1*1e-31 | | **Adam** | CE | **-0.5115** | **1.8*1e-18** | 0.0423 | 0.4999 |
| | | L2 | 0.5582 | 2.3*1e-22 | -0.7026 | 2.1*1e-39 | | | L2 | **-0.2876** | **2.8*1e-06** | 0.0197 | 0.7538 |

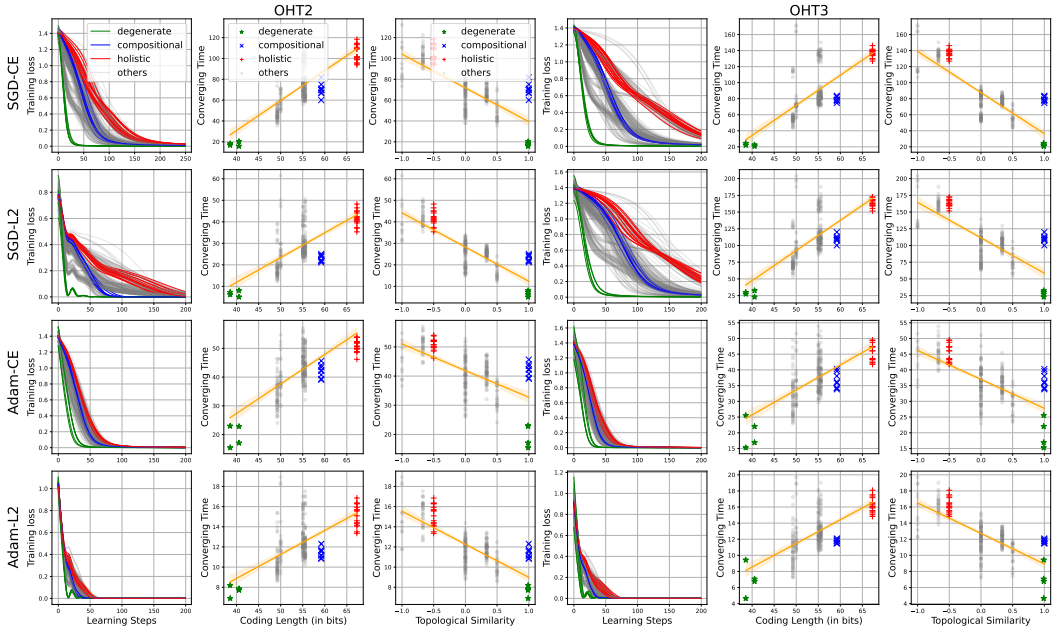



Figure 4: Experiments for the one-hot vector inputs.

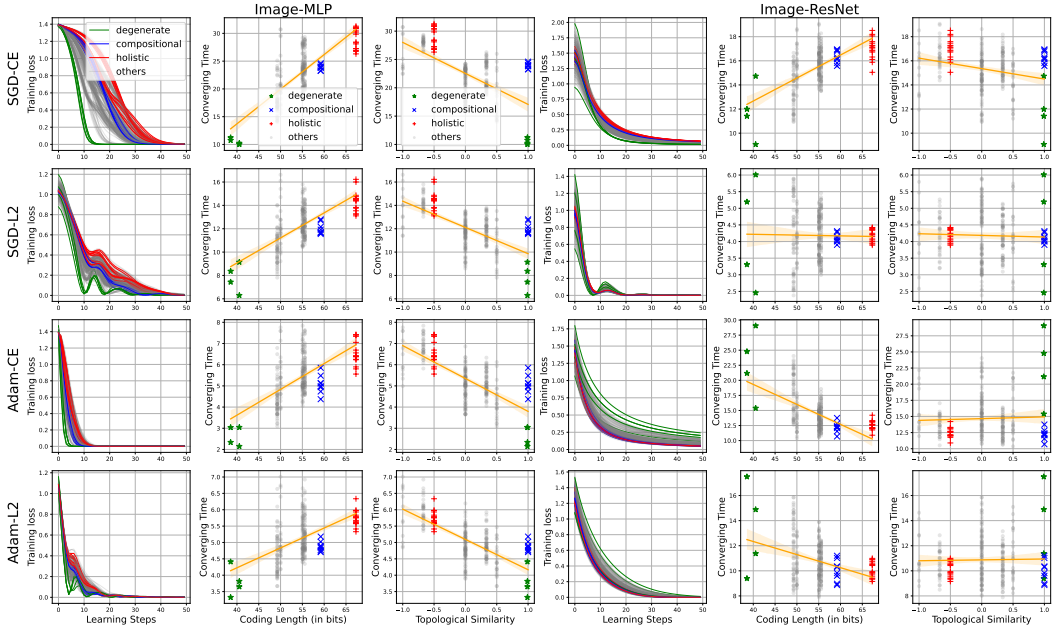



Figure 5: Experiments for the vision inputs.