Scaling can lead to compositional generalization

Florian Redhardt[*] ETH Zurich

Yassir Akram[*] ETH Zurich

Simon Schug[†] Princeton University

Abstract

Can neural networks systematically capture discrete, compositional task structure despite their continuous, distributed nature? The impressive capabilities of large-scale neural networks suggest that the answer to this question is yes. However, even for the most capable models, there are still frequent failure cases that raise doubts about their compositionality. Here, we seek to understand what it takes for a standard neural network to generalize over tasks that share compositional structure. We find that simply scaling data and model size leads to compositional generalization. We show that this holds across different task encodings as long as the training distribution sufficiently covers the task space. In line with this finding, we prove that standard multilayer perceptrons can approximate a general class of compositional task families to arbitrary precision using only a linear number of neurons with respect to the number of task modules. Finally, we uncover that if networks successfully compositionally generalize, the constituents of a task can be linearly decoded from their hidden activations. We show that this metric correlates with failures of text-to-image generation models to compose known concepts.

Code available at `https://github.com/smonsays/scale-compositionality`

1 Introduction

The ability to understand and produce novel combinations from familiar constituents is a key faculty of intelligence. It has been debated for decades whether neural networks are ever able to truly achieve such compositional generalization [1]. Regardless of these theoretical considerations, scaling neural networks continues to result in increasingly capable models [2–4]. Naturally, as models are scaled up, their capacity to memorize grows, and it is perhaps unsurprising that as a result of training on ever larger datasets their ability to recall more information grows too [5]. However, the nature of compositionality is an exponential growth and ultimately any attempt to exhaustively capture this breadth by scaling the training data will be confronted with physical constraints.

Many works therefore advocate that neural network architectures should be explicitly endowed with compositional structure [e.g., 6–9] to allow making *infinite use of their finite means* [10, 11]. Capturing the underlying compositional procedure of the data is a more efficient pathway to generalize. In particular, the algorithmic complexity of this generalizing solution is much smaller than the complexity of the memorizing solution [12]. But does this mean that architectures need to explicitly factorize according to the data's underlying compositional mechanisms [9]? For instance, monolithic networks have been shown to discover modular subnetworks which may enable compositionality without specialized symbolic mechanisms [13]. Maybe simply scaling the data and size of neural networks is then enough to achieve compositionality. Here, we attempt to answer this question:

> Do neural networks compositionally generalize at scale?

[*]Equal contribution

[†]Correspondence to `sschug@princeton.edu`.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

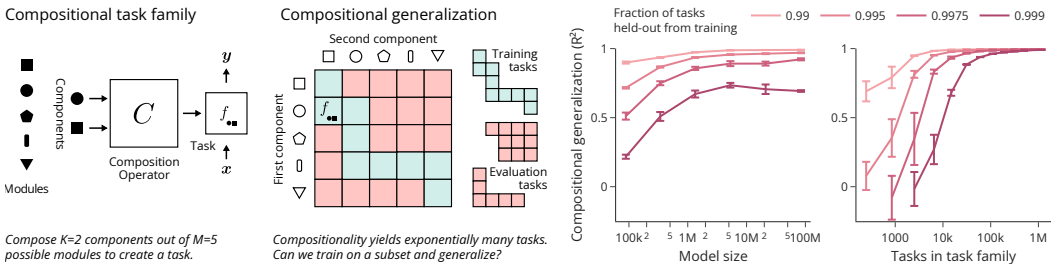

Figure 1: **Scaling can lead to compositional generalization.** We consider compositional task families that compose K out of M modules into tasks, each of which is modeled as a function. This gives rise to an exponential number of $\mathcal{O}(M^K)$ tasks. We train standard feedforward networks on a subset of tasks and evaluate compositional generalization on held-out tasks. We find that scaling the size of the model and the data leads to compositional generalization.

Our **main contributions** are as follows

- We demonstrate that standard multilayer perceptrons compositionally generalize on a variety of tasks as data and model size are scaled across task encodings if the training distribution sufficiently covers the task space.
- We prove that multilayer perceptrons can approximate a general class of compositional task families to arbitrary precision using only a linear number of neurons with respect to the number of task modules.
- We show that task constituents can be (linearly) decoded from the hidden activations of models that compositionally generalize, and demonstrate that this metric correlates with failures of image generation models to compose known concepts.

2 Compositionality and compositional generalization

We begin by formalizing compositionality and compositional generalization with the goal of capturing a variety of compositional data types including visual scenes, abstract reasoning and behavior policies.

2.1 Compositional task family

Specifically, we will consider *compositional task families* that specify a generative procedure over tasks with shared compositional structure. In a similar vein to [14], our definition uses algorithmic complexity theory, in particular the notion of Kolmogorov complexity, see [15] for a formal treatment. This definition is a modified version of the definition introduced by [16].[1]

Definition 2.1 (Compositional task family)**.** A compositional task family is a tuple $\mathcal{T} = (C, p : \boldsymbol{z} \mapsto p(\boldsymbol{z}), p : \boldsymbol{x} \mapsto p(\boldsymbol{x}))$, where:

- The task constituent space is a set $\mathcal{Z} \subseteq \{\boldsymbol{z} \in [0,1]^M : 1 \leq \|\boldsymbol{z}\|_0 \leq K \leq M\}$ with corresponding task distribution $p(\boldsymbol{z})$. K is the number of task components and M is the number of task modules.
- A *task* is a function $f_{\boldsymbol{z}} : \mathcal{X} \to \mathcal{Y}$ that labels data points $\boldsymbol{x} \sim p(\boldsymbol{x})$.
- The *composition operator* is a mapping $C : \mathcal{Z} \to (X \to Y)$ that takes as input task constituents $\boldsymbol{z} \in \mathcal{Z}$ and maps them to a task, $C(\boldsymbol{z}) \coloneqq f_{\boldsymbol{z}}$ for which the following conditions hold:
  (i) $C(\boldsymbol{z}) \neq C(\boldsymbol{z}')$ for all $\boldsymbol{z} \neq \boldsymbol{z}'$ with $\boldsymbol{z}, \boldsymbol{z}' \in \mathcal{Z}$, i.e. C is injective.
  (ii) The length of the shortest program that implements C as a function of K grows sub-exponentially in K.

In the discrete case, where $\mathcal{Z} \subseteq \{\boldsymbol{z} \in \{0,1\}^M : 1 \leq \|\boldsymbol{z}\|_0 \leq K \leq M\}$ is restricted to the set of binary, K-hot vectors, Definition 2.1 essentially states that a compositional task family compactly captures exponentially many tasks. Condition (i) ensures that all task components functionally

[1] For the sake of simplicity, we are stating the definition slightly informally. For the asymptotic behavior over K used in condition (ii) to be defined, we are technically considering a family of compositional task families.

enter the composition, while condition (ii) excludes the case where compositions are purely context-sensitive, ensuring that there is shared structure between tasks. For a more detailed discussion of this definition, please refer to [16].

The notion of a task is used in a general sense here and allows to capture different types of compositional data. For instance, a task could refer to a visual scene, where modules are the set of possible objects and the composition operator renders a selection of such objects into a scene. Similarly, a task could refer to a behavior policy, where modules consist of different reward functions, a subset of which is combined by the composition operator to induce an optimal policy.

2.2 Task encoding

Before we can continue to define compositional generalization using Definition 2.1, we must first specify how to present the model with information about its current task, as captured by the task constituents $\boldsymbol{z}$. In practice, such a task description might not be the task constituents themselves, but rather some encoding thereof. For example, a task could be described through a natural language instruction or by presenting example data points $(x_i, f_{\boldsymbol{z}}(x_i))_i$. To model this aspect, we define the task encoder as the mapping

$$\varphi : (\mathcal{Z}, \mathbb{N}) \to \mathcal{Z}'$$

that maps task constituents $\boldsymbol{z} \in \mathcal{Z}$ and a random seed $r \in \mathbb{N}$ to a task encoding. Throughout the paper, we mostly focus on settings where the task is unambiguously specified, i.e. where the task encoding φ is information-preserving and therefore injective.

2.3 Compositional generalization

With Definition 2.1 at hand, we can now formalize compositional generalization for a model that learns to perform tasks from a compositional task family, given a task encoding φ. This definition is a slightly modified version of the definition presented in [16].

Definition 2.2 (Compositional generalization)**.** A model parameterized by $\boldsymbol{\theta}$ is said to compositionally generalize with respect to the compositional task family $\mathcal{T} = (\mathcal{Z}, C, p(\boldsymbol{z}), p(\boldsymbol{x}))$ if there exists a discrete training distribution $\boldsymbol{z} \mapsto p^{\text{train}}(\boldsymbol{z})$ with finite support such that the number of points in the support grows sub-exponentially in K and it holds that

$$\begin{aligned} \boldsymbol{\theta}^* &\in \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{z}\sim p^{\text{train}}(\boldsymbol{z})} \left[\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \left[l(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{z})\right]\right] \\ \Rightarrow \boldsymbol{\theta}^* &\in \arg\min_{\boldsymbol{\theta}} \quad \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \left[\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \left[l(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{z})\right]\right], \end{aligned}$$

where $l(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{z}) = l\left(f_{\boldsymbol{z}}(\boldsymbol{x}), g_{\boldsymbol{\theta}}(\boldsymbol{x}, \varphi(\boldsymbol{z}))\right.$ is a loss function that measures the discrepancy between model predictions and the outputs of a task $f_{\boldsymbol{z}}(\boldsymbol{x})$ for a given datum $\boldsymbol{x}$ and task encoding $\varphi(\boldsymbol{z})$.

Note, that we are not considering fixed-size datasets but are sampling directly from the data distribution which reflects the nowadays common single-epoch training regime of large-scale foundation models [e.g., 3].

2.4 Hyperteachers: A general class of compositional task families

For the purpose of this study, it will be useful to instantiate a concrete but nevertheless general class of compositional task families according to Definition 2.1. Given that neural networks are flexible function approximators and thus able to cover a wide range of behaviors, we parameterize both the composition operator as well as the task functions using neural networks. The resulting system, a composable neural network that generates another neural network, can be interpreted as a hypernetwork [17]. Indeed, hypernetworks have previously been used to study compositional generalization [18].

Following [18], we consider a linear hypernetwork that sums K out of M possible weight matrices to parameterize a single hidden layer task network. We define the task constituent space to be $\mathcal{Z} \subseteq \{\boldsymbol{z} \in \{0.5, 0.6, \ldots, 1.0\}^M : 1 \leq \|\boldsymbol{z}\|_0 \leq K \leq M\}$ and the composition operator as

$$C(\boldsymbol{z}) = \boldsymbol{x} \mapsto \boldsymbol{\Omega}\, \text{ReLU}\left(\sum_{k:z_k\neq 0} z_k \boldsymbol{\Theta}_k \boldsymbol{x}\right), \tag{1}$$

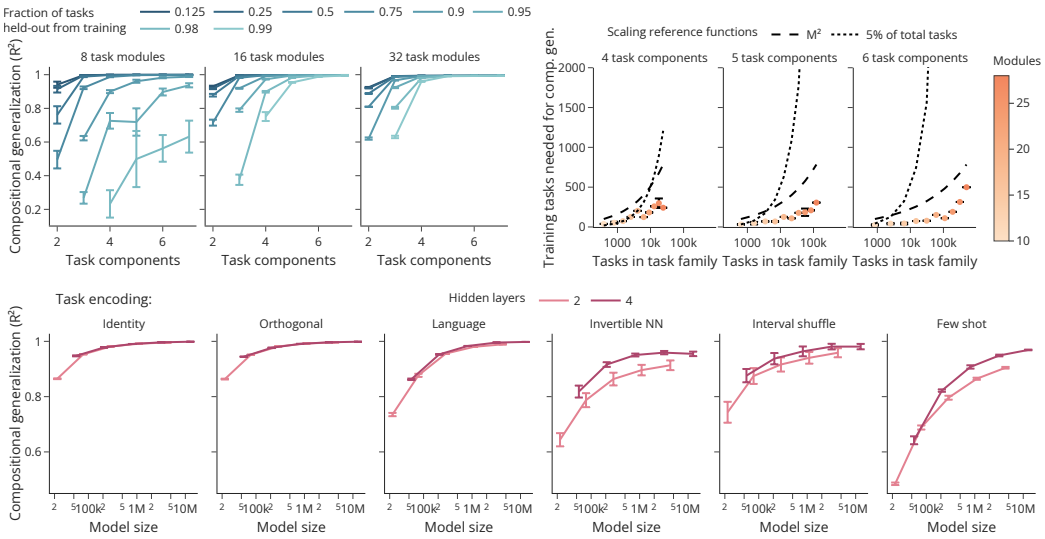

Figure 2: **Scaling data and model size leads to compositional generalization.** ***Top-left*** Scaling the number of training tasks by increasing the number of modules, task components or decreasing the fraction of tasks held-out from training leads to compositional generalization on the hyperteacher task family. ***Top-right*** The number of training tasks required to achieve compositional generalization, here defined as a $R^2 > 0.95$, scales sub-exponentially as the number of total tasks in the task family grows. ***Bottom*** Scaling model size by increasing the number of hidden neurons and the number of hidden layers leads to compositional generalization on the hyperteacher across different task encodings. Error bars denote SEM over three seeds.

where we have M sets of neural network parameters $\{\mathbf{\Theta}_m \in \mathbb{R}^{I \times H}\}_{m=1}^{M}$ with I input neurons and H hidden neurons, $\mathbf{\Omega} \in \mathbb{R}^{H \times O}$ is a shared readout projection with O output neurons and we sample $\boldsymbol{x} \in \mathbb{R}^I$ from the uniform distribution $p(\boldsymbol{x}) = \mathcal{U}[-1, 1]^I$. Each module $\mathbf{\Theta}_m$ also has an associated bias vector which we omit here for conciseness. The resulting task functions, $f_{\boldsymbol{z}} : \mathbb{R}^I \to \mathbb{R}^O$, are thus single hidden layer ReLU networks.

3 Scaling can lead to compositional generalization

In the following, we will investigate if and under what circumstances standard multilayer perceptrons compositionally generalize. We will show that simply scaling the number of tasks in the training distribution as well as scaling model size leads to compositional generalization. The results presented in this section focus on various parameterizations of the hyperteacher compositional task family presented in Section 2.4. **We reproduce all findings of this section on the compositional preference task family introduced by [18].** This task family requires learning optimal policies in a grid world with compositional reward functions. We present the corresponding results in Appendix B.1 in Figures 7, 8, 9 and Table 2. For additional experimental details, please refer to Appendix C.

While there exist various architectures specialized for compositionality, here we are interested in understanding if a standard fully connected neural network, a multilayer perceptron, can compositionally generalize. Fully connected layers are common building blocks of virtually all standard architectures, including transformers or recurrent neural networks. Specifically, we consider a multilayer perceptron with ReLU nonlinearities that accepts the concatenation of $\boldsymbol{x}$ and $\varphi(\boldsymbol{z}, r)$ as input.

In order to measure a model's ability to compositionally generalize, we will hold out tasks from training and evaluate the model's performance on these tasks.

3.1 Scaling the number of compositional tasks leads to compositional generalization

To investigate the main question of how scale affects compositional generalization, we will vary both data and model size. The former can be accomplished along two different dimensions: We can vary the total number of tasks in the compositional task family by changing both the number of modules M and the number of components K, and we can vary the fraction of distinct tasks that are held-out from training. Since the number of possible tasks grows exponentially, $\mathcal{O}(M^K)$, the compositional task families we consider can easily contain a very large number of distinct tasks. The top-left of Figure 2 shows that as we scale the number of tasks, compositional generalization improves. Notably, the required number of training tasks to achieve compositional generalization grows more slowly than the total number of tasks as shown on the top-right of Figure 2. This implies that compositional generalization with a sub-exponential number of tasks, as in Definition 2.2, is indeed achievable at scale. In Appendix B, we further demonstrate that this scaling relationship is even more favorable for transformers and similarly holds for the compositional preference task family as well as a hyperteacher with a deep target network.

In addition to scaling the data, we would like to understand how scaling model size affects compositional generalization. Shown at the bottom of Figure 2, we vary both the number of hidden layers and the size of the hidden layers for various possible task encodings $\varphi(\boldsymbol{z}, r)$ (more details on the task encodings will follow in Section 3.3). We find that, given sufficient data, increasing model size consistently improves compositional generalization. This is noteworthy, given that increasing model size in principle increases the capacity to memorize training tasks without capturing the underlying compositional structure required for compositional generalization. As we will argue in the following however, these results can be interpreted as evidence that deep neural networks tend to prefer solutions of low algorithmic complexity [19].

3.2 Complexity of generalizing solution dominates memorizing solution asymptotically

Memorizing all tasks of a compositional task family by definition requires exponential network capacity. Intuitively, a solution that captures the underlying compositional structure and thus generalizes should be more efficient. A priori, it is however not clear whether such a solution exists for a finite-sized, multilayer perceptron. As we have argued before, hyperteachers can be regarded as a general class of compositional task families. It is therefore instructive to consider whether a finite-sized multilayer perceptron can implement any hyperteacher without having to memorize the exponential number of possible tasks. The following theorem answers this question in the affirmative.

Theorem 3.1. *Let $\left(\boldsymbol{\Theta}_m \in \mathbb{R}^{I \times H}\right)$ be a sequence of uniformly bounded matrices. Then, for any $M \in \mathbb{N}$, $\varepsilon > 0$, and on any compact input set, $\mathcal{X} \times \mathcal{Z}$ with $\mathcal{Z} = \{\boldsymbol{z} : \|\boldsymbol{z}\|_1 \leq 1\}$, there exists a ReLU multilayer perceptron that approximates a hyperteacher to within ε error in the $\|\cdot\|_\infty$ norm using $\mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}} + M\right)$ neurons.*

The corresponding constructive proof is presented in Appendix A.2 along with an extension to hyperteachers with multiple layers. Theorem 3.1 notably states that the number of neurons required for the generalizing solution scales linearly in the number of modules M. Consistent with our experimental findings, this means that as M grows, the simplicity of the fully generalizing solution will increasingly dominate the naive memorizing solution.

3.3 Compositional generalization emerges across task encodings

We now turn to the question, to what extent the way in which the task is specified to the model matters for its ability to compositionally generalize. Specifically, one might suspect that certain ways of encoding a task are better suited to leveraging compositional structure than others. For instance, [20] argue that language in particular encodes task structure in a way that is beneficial for learning compositional representations. To study this question in the context of our setup, we experiment with different task encodings $\varphi(\boldsymbol{z}, r)$ illustrated in Figure 3.

Generally, we find that all task encodings lead to compositional generalization, including nonlinear encodings, although some require more model capacity as shown at the bottom of Figure 2 and Table 1 (also see Table 2 in the appendix for corresponding results on the compositional preference task

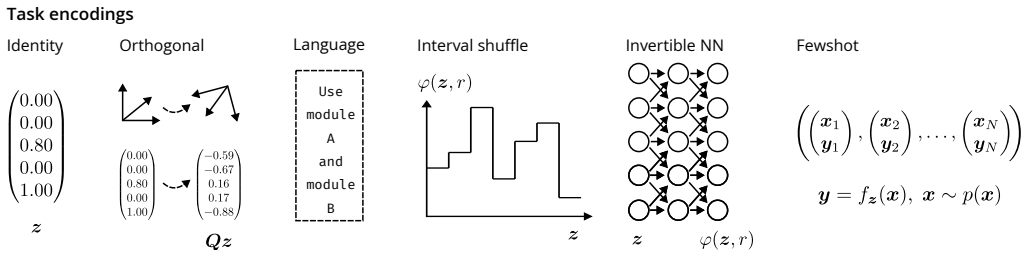

Figure 3: **Task encodings.** Illustration of the different task encodings $\varphi(\boldsymbol{z}, r)$ used in Table 1. The first three encodings are linear with respect to the task constituents while the last three are nonlinear.

Table 1: **Compositional generalization emerges across task encodings.** Comparison of the decodability of the task constituents from the hidden activations of the third layer (Task decoder) and compositional generalization performance for various task encodings for a hyperteacher with $M = 16$, $K = 3$. We additionally show the linear decodability of the task constituents directly from the task encoding itself (Input decoder), which allows to distinguish linear from nonlinear task encodings. $\pm$ SEM over three seeds.

Task encoding	Task decoder (R^2)	Input decoder (R^2)	Comp. gen. (R^2)
Identity	0.95 ± 0.012	1.00 ± 0.000	1.00 ± 0.000
Orthogonal	0.96 ± 0.002	1.00 ± 0.000	1.00 ± 0.000
Language	0.99 ± 0.000	1.00 ± 0.000	1.00 ± 0.000
Invertible NN	0.94 ± 0.001	0.56 ± 0.000	0.95 ± 0.000
Interval shuffle	0.96 ± 0.011	0.73 ± 0.082	0.98 ± 0.010
Few shot	0.90 ± 0.004	-0.23 ± 0.008	0.97 ± 0.001

family). Specifically, we observe no benefit of directly using the identity task encoding $\varphi(\boldsymbol{z}, r) = \boldsymbol{z}$, over a random but fixed orthogonal projection $\varphi(\boldsymbol{z}, r) = \boldsymbol{Q}\boldsymbol{z}$ where $\boldsymbol{Q} \in \mathbb{R}^{M \times M}$ is an orthogonal matrix. In the same way, encoding each task through a language instruction poses no issues. The latter is consistent with the observation that the task constituents can be linearly decoded from such instructions. Interestingly, even nonlinear encodings such as specifying the task through examples (denoted fewshot), via an invertible neural network or from the highly nonlinear interval shuffle function (see Algorithm 1 in the appendix for a definition) lead to compositional generalization if the model size is sufficiently large. One possible explanation for these findings is that regardless of the task encoding, the model internally infers the task constituents up to a linear transformation after which Theorem 3.1 guarantees that a generalizing solution that scales linearly in the number of modules M exists.

To verify this hypothesis, we train a linear decoder to predict the task constituents based on the hidden activations of the model solving the task. We report the ability of this task decoder to predict the task constituents on the held-out compositional generalization tasks in Table 1. Indeed, we find that also for nonlinear task encodings the task constituents can be linearly decoded from the hidden activations providing evidence that the models internally linearize the task constituents to achieve compositional generalization. We will expand on this finding in Section 4.

3.4 The support of the training distribution needs to sufficiently cover the task space

In principle, it is easy to come up with degenerate training distributions that make compositional generalization impossible. For instance, if a module is consistently absent from all training tasks, the model has no opportunity to learn this module and will generally fail to generalize to tasks that contain this module. In this sense, the support of the training distribution $p^{\text{train}}(\boldsymbol{z})$ needs to sufficiently cover the full constituent space for compositional generalization to succeed. In this section we investigate how various conditions over the training support affect compositional generalization.

Prior work has studied how the training distribution affects compositional generalization [e.g., 18, 21, 22]. Following [21] we refer to the condition of having non-zero support for each module in the training distribution as *compositional support* . For the class of hyperteacher task families,

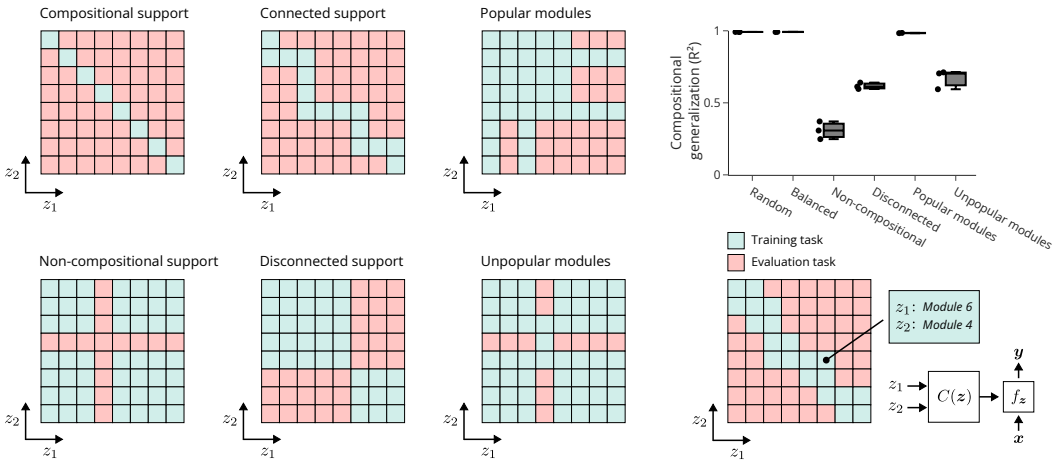

Figure 4: **The support of the training distribution needs to sufficiently cover the task space.**: ***Left*** Illustration of the different types of task support for the special case of a compositional task family with two components. Turquoise tiles denote module combinations that are part of the training distribution, red tiles are reserved for evaluation. ***Right*** Compositional generalization as a function of the different types of task support on the hyperteacher for $M = 16$ and $K = 3$.

the additional condition of *connected support* needs to be satisfied, which states that no subset of modules should appear solely in isolation from the rest. [18] show that in a teacher-student setting where both the teacher and the student are hypernetworks, this condition is required to guarantee compositional generalization. We can extend this result to the more general case of any kind of student, including the ReLU multilayer perceptron. The proof follows immediately by constructing examples of multiple different hyperteachers that have an identical training distribution if the training support is not connected. Please refer to Appendix A.1 for more details.

Figure 4 illustrates the different types of training support we consider here as well as their effect on compositional generalization in the hyperteacher. For a more detailed description for how each training support is constructed, please refer to Appendix C.3. Our findings empirically confirm that violating compositional and connected support interferes with compositional generalization. Interestingly, having a small set of popular modules who appear more frequently poses no issue for achieving compositional generalization. The converse of having a small set of unpopular modules that are rarely encountered however does lead to a noticeable drop. These findings are consistent with prior work that find that module imbalance hampers compositional generalization [23, 24]. However, our experiments further suggest that such failures are due to underrepresentation of certain modules and not simply due to an asymmetry in module popularity. Intuitively, if a module is seen only a constant number of times during training, it can be memorized with constant capacity.

4 Task constituents are linearly decodable in models that compositionally generalize

Section 3.3 has revealed that in cases where compositional generalization succeeds, the task constituents can be linearly decoded from the hidden activations. This prompts the question whether models that succeed to compositionally generalize typically form an internal linear representation of the task constituents. Generally speaking, we can show that for any model that compositionally generalizes to most tasks, the task constituents must be decodable.

Theorem 4.1 (Decodability under compositional generalization)**.** *For any $\delta > 0$, assume we have a student g that predicts labels $f_{\boldsymbol{z}}(\boldsymbol{x}) = \boldsymbol{y}$ given a task encoding $\varphi(\boldsymbol{z}, r)$ and inputs $\boldsymbol{x}$. Then there exists a decoder map ϕ that decodes $\varphi(\boldsymbol{z}, r)$ to $\boldsymbol{z}$ with probability at least $1 - \sqrt{\delta}$, if:*

(i) $\mathbb{P}_{\boldsymbol{z},\boldsymbol{x}}[g(\varphi(\boldsymbol{z}, r), \boldsymbol{x}) = C(\boldsymbol{z})(\boldsymbol{x})] > 1 - \delta$

(ii) For each $\boldsymbol{z} \neq \boldsymbol{z}'$, $\mathbb{P}_{\boldsymbol{x}}[C(\boldsymbol{z})(\boldsymbol{x}) = C(\boldsymbol{z}')(\boldsymbol{x})] < 1 - 2\sqrt{\delta}$

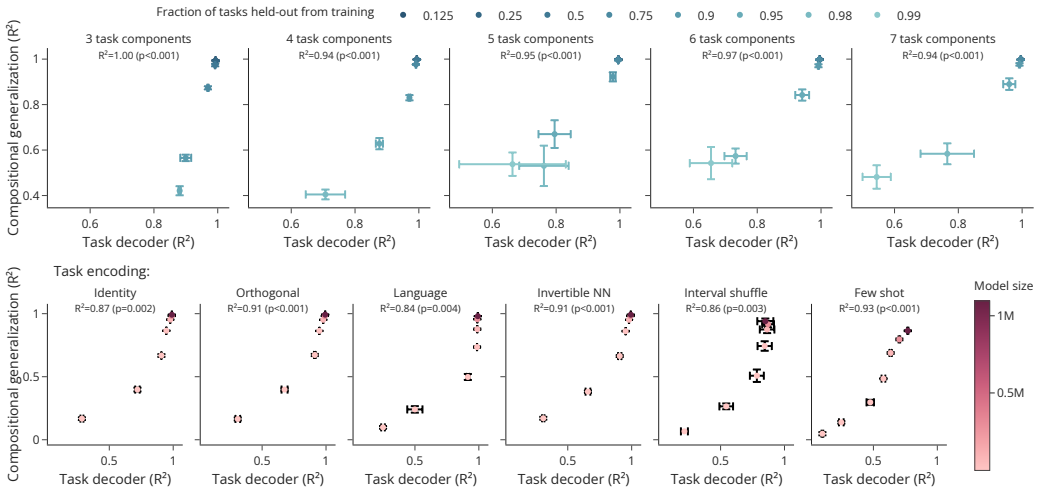

Figure 5: **Compositional generalization correlates with linear decodability of task constituents.** ***Top*** Relationship between linear decodability of the task constituents and compositional generalization across hyperteachers with $M = 8$ modules and varying K. ***Bottom*** Relationship between linear decodability of the task constituents and compositional generalization across different task encodings for varying model sizes on the hyperteacher with $M = 16$, $K = 3$. Error bars denote SEM over three seeds. ***Top/Bottom*** We report the R^2 and corresponding p-value for an ordinary least square estimator in the facet titles.

We provide the proof in Appendix A.3. In practice, we observe an even stronger version of this statement, namely that the task constituents are *linearly* decodable from the hidden activations in multilayer perceptrons that successfully compositionally generalize.

4.1 Compositional generalization correlates with linear decodability of task constituents

To further illuminate the connection between the observed linear decodability of the task constituents and compositional generalization, we attempt to decode the task constituents in models that (partially) fail to fully compositionally generalize. Figure 5 shows a remarkably clear correlation between decodability and compositional generalization across different data scales (top) and model sizes and task encodings (bottom). Particularly interesting is the case where the unmodified task constituents are provided to the model, i.e. where the task encoding is the identity. In this case, the task constituents are of course trivially linearly decodable from the input. However, training the decoder on the hidden activations of deeper layers, this information is lost in networks that do not compositionally generalize. In line with previous research, this implies that having access to a disentangled task representation is by itself not sufficient to achieve compositional generalization [25, 26].

4.2 Task constituents can be linearly decoded when image composition succeeds

Finally, can we leverage the decodability of task constituents to gain insights into the successes and failures of image generation models at composing scenes from text prompts? [2] Image generation models have come a long way, displaying impressive abilities in creating novel compositions of known concepts [24]. Nevertheless, there are still systematic failure cases [27, 28]. Can such failures be related to the ability to infer the task constituents from the model's hidden activations?

To answer this question, we construct a large number of compositional tasks that require composing several concepts. Two examples are shown in the top-left of Figure 6. For an extensive list of all the tasks we consider, as well as more details on the task construction, see Appendix C.7. Using these tasks, we evaluate the ability of several diffusion-based image generation models to systematically generate image compositions [29, 30]. We use a vision-language model to label whether a given generation was successful and train a linear decoder to decode the task constituents based on the

[2] Code for image composition experiments available at `https://github.com/florian-toll/compgen-vision`

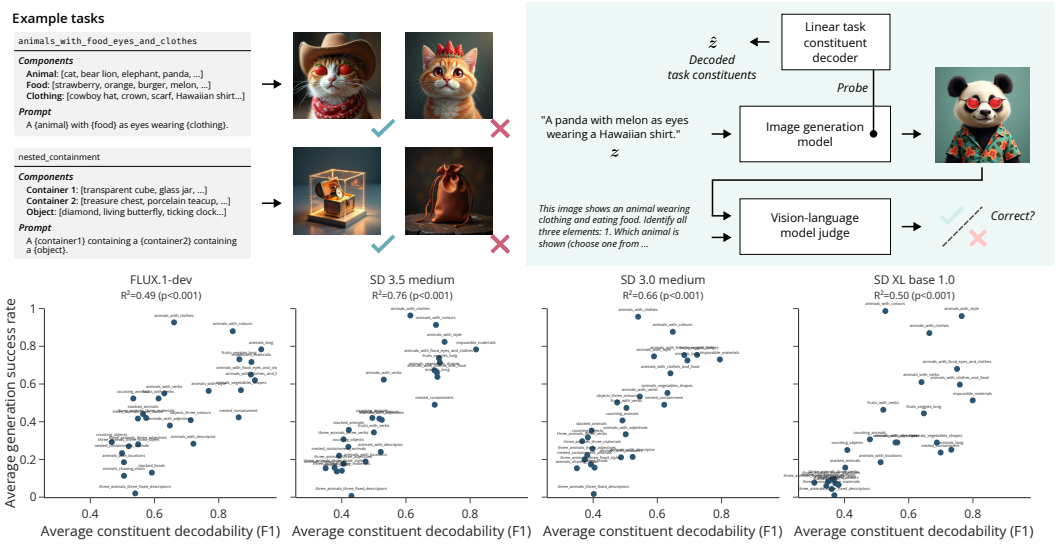

Figure 6: **Task constituents can be linearly decoded when image composition succeeds.** ***Top*** Example tasks to test compositionality of image generation models as well as an overview of the pipeline used to decode task constituents and judge model outputs. ***Bottom*** Relationship between average constituent decodability and image generation success rate across image composition task families for four different image generation models. We report the R^2 and corresponding p-value for an ordinary least square estimator in the facet titles.

model's hidden activations. The full pipeline is shown in the top-right of Figure 6. Please refer to Appendix C.7 for further details. The results of this analysis are shown at the bottom of Figure 6. We observe a clear correlation between the average task constituent decodability and the average generation success rate across models, with the relative task difficulty being similar across models, as shown in Figure 12 of the Appendix. This provides evidence that models which succeed at systematically composing known concepts into scenes form an internal representation of the task constituents.

5 Related work

The study of compositional generalization in neural networks has a long and rich history, with early critiques highlighting the challenges of connectionist models to exhibit systematicity and compositionality [e.g., 1, 31–33] and numerous work that in response explored mechanisms for representing and processing structured information using distributed representations [e.g., 34, 35]. In recent years, theoretical progress has been made in showing that compositional generalization can provably be achieved with neural networks in specific settings [18, 22, 36–39]. This typically requires constraining the model architecture and the data generating process. Consistent with our results, the statistics of the training data play a crucial rule in enabling compositional generalization [18, 21, 22, 39].

We aim to complement this work by showing how scaling generic neural networks can lead to compositional generalization in the absence of stronger architectural constraints. This is motivated by the finding that scaling neural networks can break the curse of dimensionality [40], and consistently results in improvements in model performance [2, 41] with new capabilities emerging as models are scaled up [3, 4, 42]. Compositional abilities in particular have therefore seemingly moved within grasp in practice [43–48]. However, even at larger scales, models often display a compositional generalization gap which does not close as scale increases [49–57], despite standard transformers showing compositionality in controlled settings [58–60].

Image generation models in particular have made impressive leaps in their ability to create novel image compositions [29, 30, 61, 62]. Compositional abilities have been shown to emerge in such diffusion-based models on synthetic tasks in an order determined by the underlying data processes and with performance showing sudden emergence due to multiplicative dependencies [24, 63]. Consistent

with our finding that the task constituents are linearly decodable in models that successfully compositionally generalize, [64] find that diffusion image generation models learn factorized representations on a number of synthetic tasks. However, as the number of concepts that need to be composed grows, the performance of image generation models starts to deteriorate, showing the limits of their productivity to arbitrarily complex compositions [27, 28].

6 Discussion

We have shown that simply scaling standard multilayer perceptrons can lead to compositional generalization challenging the position that stronger architectural priors are necessary to endow neural networks with compositionality [6–9]. That being said, architectural priors do matter when it comes to data efficiency, as highlighted by the improved scaling of transformers over multilayer perceptrons (see Appendix B). Our findings also emphasize that the particular structure of the training data plays a critical role, demonstrating that not any type of scaling will lead to compositional generalization.

Interestingly, we find that when the training distribution is appropriately chosen, a wide range of task encodings support compositional generalization, including language but crucially also nonlinear task encodings such as specifying each task through examples. Prior work posits that language compositionally structures neural representations and thereby aids rapid adaptation to novel compositional tasks [20]. Our results indicate that in fact any information-preserving mapping of the underlying task constituents suffices to achieve compositional generalization. This might help explain why animals that do not use sophisticated languages can compositionally generalize [e.g., 65].

Surprisingly, regardless of whether task encodings are linear or nonlinear, we consistently find that the degree to which a model compositionally generalizes is related to the task constituents being linearly decodable from its hidden activations. In particular, we show that this metric correlates with the success rate of image generation models at composing novel scenes from known constituents. This finding suggests that, even though nonlinear representations would in principle be possible, successful compositional generalization in neural networks depends on linear representations of compositional structure.

Limitations While Theorem3.1 reveals the existence of a solution that enables compositional generalization without requiring an exponential number of neurons, identifying the theoretical conditions under which one can show that this solution is guaranteed to be discovered by stochastic gradient descent is an open question. Prior results indicate that deep neural networks trained with stochastic gradient descent often display a preference towards simple solutions [19, 66–68]. In the context of compositional generalization, the complexity of the memorizing solution is by definition exponential in the number of modules which might help explain why empirically we observe the discovery of the generalizing solution which in contrast scales linearly in the number of modules.

We have focused on settings, where the task is fully specified through the task encoding. In practice, a task specification might be ambiguous or incomplete, requiring models to handle uncertainty and infer the task at hand. The strong in-context learning abilities of transformers that might allow extracting localized, and even cross-modal task vectors could play an important role in this context [69–71].

More generally, scaling the data in a way that precisely controls the coverage of the training distribution requires an understanding of the true generative process underlying the compositional data. For this reason, our scaling experiments rely on synthetic data generation that gives us the required experimental control. On real-world data, the underlying generative process is mostly unknown and accordingly identifying training distributions that can support compositional generalization remains an open question.

Broader impacts This paper conducts foundational research aiming to illuminate under what circumstances neural networks compositionally generalize. While we foresee no immediate negative societal impact, we hope that it may improve our understanding of this widely deployed technology.

Acknowledgments We would like to thank Seijin Kobayashi, Angelika Steger, Jack Brady and Brenden Lake for vital discussions and fruitful feedback. Simon Schug was supported by a Postdoc.Mobility grant (P500PT_225369) from the Swiss National Science Foundation.

References

[1] Jerry A. Fodor and Zenon W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, March 1988. ISSN 00100277. doi: 10.1016/0010-0277(88)90031-5. URL https://linkinghub.elsevier.com/retrieve/pii/0010027788900315.

[2] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, January 2020. URL http://arxiv.org/abs/2001.08361. arXiv:2001.08361 [cs].

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pages 1877–1901, Red Hook, NY, USA, December 2020. Curran Associates Inc. ISBN 978-1-7138-2954-6.

[4] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=yzkSU5zdwD.

[5] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the Limits of Weakly Supervised Pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[6] Akhilan Boopathy, Sunshine Jiang, William Yue, Jaedong Hwang, Abhiram Iyer, and Ila R. Fiete. Breaking Neural Network Scaling Laws with Modularity. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=5Qxx5KpFms.

[7] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks, October 2018. URL http://arxiv.org/abs/1806.01261. arXiv:1806.01261 [cs, stat].

[8] Jacob Russin, Randall C. O'Reilly, Jason Jo, and Yoshua Bengio. Systematicity in a Recurrent Neural Network by Factorizing Syntax andSemantics. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 42(0), 2020. URL https://escholarship.org/uc/item/3c0462ph.

[9] Yilun Du and Leslie Pack Kaelbling. Compositional Generative Modeling: A Single Model is Not All You Need. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=SoNexFx8qz.

[10] Wilhelm Von Humboldt. *Humboldt:'On language': On the diversity of human language construction and its influence on the mental development of the human species*. Cambridge University Press, 1836.

[11] Noam Chomsky. *Aspects of the Theory of Syntax*. Number 11. MIT press, 1965.

[12] Yi Ren and Danica J. Sutherland. Understanding Simplicity Bias towards Compositional Mappings via Learning Dynamics. In *NeurIPS 2024 Workshop on Compositional Learning: Perspectives, Methods, and Paths Forward*, 2024. URL https://openreview.net/forum?id=TOEv2N7RuG.

[13] Michael Lepori, Thomas Serre, and Ellie Pavlick. Break It Down: Evidence for Structural Compositionality in Neural Networks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 42623–42660. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/85069585133c4c168c865e65d72e9775-Paper-Conference.pdf.

[14] Eric Elmoznino, Thomas Jiralerspong, Yoshua Bengio, and Guillaume Lajoie. A Complexity-Based Theory of Compositionality, February 2025. URL http://arxiv.org/abs/2410.14817. arXiv:2410.14817 [cs].

[15] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Texts in Computer Science. Springer, Cham, 4th ed. 2019 edition, 2019. ISBN 978-3-030-11298-1. doi: 10.1007/978-3-030-11298-1.

[16] Simon Schug. *Meta-Learning & Compositional Generalization in Neural Networks*. Doctoral Thesis, ETH Zurich, 2025. URL https://www.research-collection.ethz.ch/handle/20.500.11850/738789.

[17] David Ha, Andrew M. Dai, and Quoc V. Le. HyperNetworks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

[18] Simon Schug, Seijin Kobayashi, Yassir Akram, Maciej Wolczyk, Alexandra Maria Proca, Johannes von Oswald, Razvan Pascanu, Joao Sacramento, and Angelika Steger. Discovering modular solutions that generalize compositionally. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=H98CVcX1eh.

[19] Andrew Gordon Wilson. Deep Learning is Not So Mysterious or Different, March 2025. URL http://arxiv.org/abs/2503.02113. arXiv:2503.02113 [cs].

[20] Reidar Riveland and Alexandre Pouget. Natural language instructions induce compositional generalization in networks of neurons. *Nature Neuroscience*, pages 1–12, March 2024. ISSN 1546-1726. doi: 10.1038/s41593-024-01607-5. URL https://www.nature.com/articles/s41593-024-01607-5. Publisher: Nature Publishing Group.

[21] Thaddäus Wiedemer, Prasanna Mayilvahanan, Matthias Bethge, and Wieland Brendel. Compositional Generalization from First Principles. November 2023. URL https://openreview.net/forum?id=LqOQ1uJmSx.

[22] Samuel Lippl and Kim Stachenfeld. When does compositional structure yield compositional generalization? A kernel theory. October 2024. URL https://openreview.net/forum?id=FPBce2P1er&name=pdf.

[23] Yuren Cong, Martin Renqiang Min, Li Erran Li, Bodo Rosenhahn, and Michael Ying Yang. Attribute-Centric Compositional Text-to-Image Generation, January 2023. URL http://arxiv.org/abs/2301.01413. arXiv:2301.01413 [cs].

[24] Maya Okawa, Ekdeep S Lubana, Robert Dick, and Hidenori Tanaka. Compositional Abilities Emerge Multiplicatively: Exploring Diffusion Models on a Synthetic Task. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 50173–50195. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/9d0f188c7947eacb0c07f709576824f6-Paper-Conference.pdf.

[25] Milton Montero, Jeffrey Bowers, Rui Ponte Costa, Casimir Ludwig, and Gaurav Malhotra. Lost in Latent Space: Examining failures of disentangled models at combinatorial generalisation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 10136–10149. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/41ca8a0eb2bc4927a499b910934b9b81-Paper-Conference.pdf.

[26] Seijin Kobayashi, Simon Schug, Yassir Akram, Florian Redhardt, Johannes von Oswald, Razvan Pascanu, Guillaume Lajoie, and João Sacramento. When can transformers compositionally generalize in-context?, July 2024. URL http://arxiv.org/abs/2407.12275. arXiv:2407.12275 [cs].

[27] Martha Lewis, Nihal Nayak, Peilin Yu, Jack Merullo, Qinan Yu, Stephen Bach, and Ellie Pavlick. Does CLIP Bind Concepts? Probing Compositionality in Large Image Models. In Yvette Graham and Matthew Purver, editors, *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1487–1500, St. Julian's, Malta, March 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-eacl.101/.

[28] Xindi Wu, Dingli Yu, Yangsibo Huang, Olga Russakovsky, and Sanjeev Arora. ConceptMix: A Compositional Image Generation Benchmark with Controllable Difficulty. November 2024. URL https://openreview.net/forum?id=MU2s9wwWLo#discussion.

[29] Black Forest Labs. black-forest-labs/FLUX.1-dev · Hugging Face, April 2025. URL https://huggingface.co/black-forest-labs/FLUX.1-dev.

[30] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling Rectified Flow Transformers for High-Resolution Image Synthesis, March 2024. URL http://arxiv.org/abs/2403.03206. arXiv:2403.03206 [cs].

[31] Paul Smolensky. Connectionism, Constituency and the Language of Thought. In Barry M. Loewer and Georges Rey, editors, *Meaning in Mind: Fodor and His Critics*. Blackwell, 1991.

[32] Robert F Hadley. Systematicity in connectionist language learning. *Mind & Language*, 9(3): 247–272, 1994. Publisher: Wiley Online Library.

[33] Steven Phillips. *Connectionism and the problem of systematicity*. PhD Thesis, University of Queensland, 1995.

[34] Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1):159–216, November 1990. ISSN 0004-3702. doi: 10.1016/0004-3702(90)90007-M. URL https://www.sciencedirect.com/science/article/pii/000437029090007M.

[35] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural Module Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[36] Devon Jarvis, Richard Klein, Benjamin Rosman, and Andrew M. Saxe. On The Specialization of Neural Modules. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Fh97BDaR6I.

[37] Thaddäus Wiedemer, Jack Brady, Alexander Panfilov, Attila Juhos, Matthias Bethge, and Wieland Brendel. Provable Compositional Generalization for Object-Centric Learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=7VPTUWkiDQ.

[38] Jin Hwa Lee, Stefano Sarao Mannelli, and Andrew M Saxe. Why Do Animals Need Shaping? A Theory of Task Composition and Curriculum Learning. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 26837–26855. PMLR, July 2024. URL https://proceedings.mlr.press/v235/lee24r.html.

[39] Jack Brady, Julius von Kügelgen, Sebastien Lachapelle, Simon Buchholz, Thomas Kipf, and Wieland Brendel. Interaction Asymmetry: A General Principle for Learning Composable Abstractions. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=cCl10IU836.

[40] Francesco Cagnetta, Leonardo Petrini, Umberto M. Tomasini, Alessandro Favero, and Matthieu Wyart. How Deep Neural Networks Learn Compositional Data: The Random Hierarchy Model. *Physical Review X*, 14(3):031001, July 2024. doi: 10.1103/PhysRevX.14.031001. URL https://link.aps.org/doi/10.1103/PhysRevX.14.031001. Publisher: American Physical Society.

[41] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pages 30016–30030, Red Hook, NY, USA, November 2022. Curran Associates Inc. ISBN 978-1-7138-7108-8.

[42] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Johan Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew M. Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bart\lomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, Cesar Ferri, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Christopher Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, C. Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodolà, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Xinyue Wang, Gonzalo Jaimovitch-Lopez, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Francis Anthony Shevlin, Hinrich Schuetze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Kocon, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh Dhole, Kevin Gimpel, Kevin Omondi, Kory Wallace Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros-Colón, Luke Metz, Lütfi Kerem Senel, Maarten Bosma, Maarten Sap, Maartje Ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramirez-Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias

Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael Andrew Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michal Swedrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T, Nanyun Peng, Nathan Andrew Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter W. Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Milkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan Le Bras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Russ Salakhutdinov, Ryan Andrew Chi, Seungjae Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel Stern Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima Shammie Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven Piantadosi, Stuart Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsunori Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Venkatesh Ramasesh, vinay uday prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj.

[43] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WZH7099tgfM.

[44] A. Emin Orhan. Compositional generalization in semantic parsing with pretrained transformers, December 2022. URL http://arxiv.org/abs/2109.15101. arXiv:2109.15101 [cs].

[45] Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. Compositional Generalization in Semantic Parsing: Pre-training vs. Specialized Architectures, September 2021. URL http://arxiv.org/abs/2007.08970. arXiv:2007.08970 [cs].

[46] Róbert Csordás, Kazuki Irie, and Juergen Schmidhuber. The Devil is in the Detail: Simple Tricks Improve Systematic Generalization of Transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 619–634, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.49. URL https://aclanthology.org/2021.emnlp-main.49.

[47] Linlu Qiu, Peter Shaw, Panupong Pasupat, Tianze Shi, Jonathan Herzig, Emily Pitler, Fei Sha, and Kristina Toutanova. Evaluating the Impact of Model Scale for Compositional Generalization in Semantic Parsing. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings*

of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 9157–9179, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.624. URL https://aclanthology.org/2022.emnlp-main.624/.

[48] Jackson Petty, Sjoerd Steenkiste, Ishita Dasgupta, Fei Sha, Dan Garrette, and Tal Linzen. The Impact of Depth on Compositional Generalization in Transformer Language Models. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7239–7252, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.402. URL https://aclanthology.org/2024.naacl-long.402/.

[49] Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and Fate: Limits of Transformers on Compositionality, June 2023. URL http://arxiv.org/abs/2305.18654. arXiv:2305.18654 [cs].

[50] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and Narrowing the Compositionality Gap in Language Models, May 2023. URL http://arxiv.org/abs/2210.03350. arXiv:2210.03350 [cs].

[51] Arian Hosseini, Alessandro Sordoni, Daniel Kenji Toyama, Aaron Courville, and Rishabh Agarwal. Not All LLM Reasoners Are Created Equal. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024. URL https://openreview.net/forum?id=RcqAmkDJfI.

[52] Zhuoyan Xu, Zhenmei Shi, and Yingyu Liang. Do Large Language Models Have Compositional Ability? An Investigation into Limitations and Scalability. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024. URL https://openreview.net/forum?id=4XPeF0SbJs.

[53] Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=AjXkRZIvjB.

[54] Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokking of Implicit Reasoning in Transformers: A Mechanistic Journey to the Edge of Generalization. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 95238–95265. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ad217e0c7fecc71bdf48660ad6714b07-Paper-Conference.pdf.

[55] Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do Large Language Models Latently Perform Multi-Hop Reasoning? In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10210–10229, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.550. URL https://aclanthology.org/2024.acl-long.550/.

[56] Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The Reversal Curse: LLMs trained on "A is B" fail to learn "B is A". In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=GPKTIktA0k.

[57] Apoorv Khandelwal and Ellie Pavlick. How Do Language Models Compose Functions?, October 2025. URL http://arxiv.org/abs/2510.01685. arXiv:2510.01685 [cs].

[58] Brenden M. Lake and Marco Baroni. Human-like systematic generalization through a meta-learning neural network. *Nature*, 623(7985):115–121, November 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06668-3. URL https://www.nature.com/articles/s41586-023-06668-3. Publisher: Nature Publishing Group.

[59] Simon Schug, Seijin Kobayashi, Yassir Akram, Joao Sacramento, and Razvan Pascanu. Attention as a Hypernetwork. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=V4K9h1qNxE.

[60] Ryoma Kumon and Hitomi Yanaka. Analyzing the Inner Workings of Transformers in Compositional Generalization. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8529–8540, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. URL https://aclanthology.org/2025.naacl-long.432/.

[61] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, and others. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

[62] Imagen-Team-Google, Jason Baldridge, Jakob Bauer, Mukul Bhutani, Nicole Brichtova, Andrew Bunner, Lluis Castrejon, Kelvin Chan, Yichang Chen, Sander Dieleman, Yuqing Du, Zach Eaton-Rosen, Hongliang Fei, Nando de Freitas, Yilin Gao, Evgeny Gladchenko, Sergio Gómez Colmenarejo, Mandy Guo, Alex Haig, Will Hawkins, Hexiang Hu, Huilian Huang, Tobenna Peter Igwe, Christos Kaplanis, Siavash Khodadadeh, Yelin Kim, Ksenia Konyushkova, Karol Langner, Eric Lau, Rory Lawton, Shixin Luo, Soňa Mokrá, Henna Nandwani, Yasumasa Onoe, Aäron van den Oord, Zarana Parekh, Jordi Pont-Tuset, Hang Qi, Rui Qian, Deepak Ramachandran, Poorva Rane, Abdullah Rashwan, Ali Razavi, Robert Riachi, Hansa Srinivasan, Srivatsan Srinivasan, Robin Strudel, Benigno Uria, Oliver Wang, Su Wang, Austin Waters, Chris Wolff, Auriel Wright, Zhisheng Xiao, Hao Xiong, Keyang Xu, Marc van Zee, Junlin Zhang, Katie Zhang, Wenlei Zhou, Konrad Zolna, Ola Aboubakar, Canfer Akbulut, Oscar Akerlund, Isabela Albuquerque, Nina Anderson, Marco Andreetto, Lora Aroyo, Ben Bariach, David Barker, Sherry Ben, Dana Berman, Courtney Biles, Irina Blok, Pankil Botadra, Jenny Brennan, Karla Brown, John Buckley, Rudy Bunel, Elie Bursztein, Christina Butterfield, Ben Caine, Viral Carpenter, Norman Casagrande, Ming-Wei Chang, Solomon Chang, Shamik Chaudhuri, Tony Chen, John Choi, Dmitry Churbanau, Nathan Clement, Matan Cohen, Forrester Cole, Mikhail Dektiarev, Vincent Du, Praneet Dutta, Tom Eccles, Ndidi Elue, Ashley Feden, Shlomi Fruchter, Frankie Garcia, Roopal Garg, Weina Ge, Ahmed Ghazy, Bryant Gipson, Andrew Goodman, Dawid Górny, Sven Gowal, Khyatti Gupta, Yoni Halpern, Yena Han, Susan Hao, Jamie Hayes, Jonathan Heek, Amir Hertz, Ed Hirst, Emiel Hoogeboom, Tingbo Hou, Heidi Howard, Mohamed Ibrahim, Dirichi Ike-Njoku, Joana Iljazi, Vlad Ionescu, William Isaac, Reena Jana, Gemma Jennings, Donovon Jenson, Xuhui Jia, Kerry Jones, Xiaoen Ju, Ivana Kajic, Christos Kaplanis, Burcu Karagol Ayan, Jacob Kelly, Suraj Kothawade, Christina Kouridi, Ira Ktena, Jolanda Kumakaw, Dana Kurniawan, Dmitry Lagun, Lily Lavitas, Jason Lee, Tao Li, Marco Liang, Maggie Li-Calis, Yuchi Liu, Javier Lopez Alberca, Matthieu Kim Lorrain, Peggy Lu, Kristian Lum, Yukun Ma, Chase Malik, John Mellor, Thomas Mensink, Inbar Mosseri, Tom Murray, Aida Nematzadeh, Paul Nicholas, Signe Nørly, João Gabriel Oliveira, Guillermo Ortiz-Jimenez, Michela Paganini, Tom Le Paine, Roni Paiss, Alicia Parrish, Anne Peckham, Vikas Peswani, Igor Petrovski, Tobias Pfaff, Alex Pirozhenko, Ryan Poplin, Utsav Prabhu, Yuan Qi, Matthew Rahtz, Cyrus Rashtchian, Charvi Rastogi, Amit Raul, Ali Razavi, Sylvestre-Alvise Rebuffi, Susanna Ricco, Felix Riedel, Dirk Robinson, Pankaj Rohatgi, Bill Rosgen, Sarah Rumbley, Moonkyung Ryu, Anthony Salgado, Tim Salimans, Sahil Singla, Florian Schroff, Candice Schumann, Tanmay Shah, Eleni Shaw, Gregory Shaw, Brendan Shillingford, Kaushik Shivakumar, Dennis Shtatnov, Zach Singer, Evgeny Sluzhaev, Valerii Sokolov, Thibault Sottiaux, Florian Stimberg, Brad Stone, David Stutz, Yu-Chuan Su, Eric Tabellion, Shuai Tang, David Tao, Kurt Thomas, Gregory Thornton, Andeep Toor, Cristian Udrescu, Aayush Upadhyay, Cristina Vasconcelos, Alex Vasiloff, Andrey Voynov, Amanda Walker, Luyu Wang, Miaosen Wang, Simon Wang, Stanley Wang, Qifei Wang, Yuxiao Wang, Ágoston Weisz, Olivia Wiles, Chenxia Wu, Xingyu Federico Xu, Andrew Xue, Jianbo Yang, Luo Yu, Mete Yurtoglu, Ali Zand, Han Zhang, Jiageng Zhang, Catherine Zhao, Adilet Zhaxybay, Miao Zhou, Shengqi Zhu, Zhenkai Zhu, Dawn Bloxwich, Mahyar Bordbar, Luis C. Cobo, Eli Collins, Shengyang Dai, Tulsee Doshi, Anca Dragan, Douglas Eck, Demis Hassabis, Sissie Hsiao, Tom Hume, Koray Kavukcuoglu, Helen King, Jack Krawczyk, Yeqing Li, Kathy Meier-Hellstern, Andras Orban, Yury Pinsky, Amar Subramanya, Oriol Vinyals, Ting Yu, and Yori Zwols. Imagen 3, December 2024. URL http://arxiv.org/abs/2408.07009. arXiv:2408.07009 [cs].

[63] Yongyi Yang, Core Francisco Park, Ekdeep Singh Lubana, Maya Okawa, Wei Hu, and Hidenori Tanaka. Swing-by Dynamics in Concept Learning and Compositional Generalization, March 2025. URL http://arxiv.org/abs/2410.08309. arXiv:2410.08309 [cs].

[64] Qiyao Liang, Ziming Liu, Mitchell Ostrow, and Ila Fiete. How Diffusion Models Learn to Factorize and Compose. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 15121–15148. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/1b60893887328a6a50dd00ae0d5ed51a-Paper-Conference.pdf.

[65] Lucas Y. Tian, Kedar U. Garzón, Adam G. Rouse, Mark A. G. Eldridge, Marc H. Schieber, Xiao-Jing Wang, Joshua B. Tenenbaum, and Winrich A. Freiwald. Neural representation of action symbols in primate frontal cortex, March 2025. URL http://biorxiv.org/lookup/doi/10.1101/2025.03.03.641276.

[66] Pedro Savarese, Itay Evron, Daniel Soudry, and Nathan Srebro. How do infinite width bounded norm networks look in function space? In *Proceedings of the Thirty-Second Conference on Learning Theory*, pages 2667–2690. PMLR, June 2019. URL https://proceedings.mlr.press/v99/savarese19a.html. ISSN: 2640-3498.

[67] Lénaïc Chizat and Francis Bach. Implicit Bias of Gradient Descent for Wide Two-layer Neural Networks Trained with the Logistic Loss. In *Proceedings of Thirty Third Conference on Learning Theory*, pages 1305–1338. PMLR, July 2020. URL https://proceedings.mlr.press/v125/chizat20a.html. ISSN: 2640-3498.

[68] Etienne Boursier, Loucas Pillaud-Vivien, and Nicolas Flammarion. Gradient flow dynamics of shallow ReLU networks for square loss and orthogonal inputs. *Advances in Neural Information Processing Systems*, 35:20105–20118, December 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/7eeb9af3eb1f48e29c05e8dd3342b286-Abstract-Conference.html.

[69] Roee Hendel, Mor Geva, and Amir Globerson. In-Context Learning Creates Task Vectors. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.624. URL https://aclanthology.org/2023.findings-emnlp.624/.

[70] Grace Luo, Trevor Darrell, and Amir Bar. Vision-Language Models Create Cross-Modal Task Representations, May 2025. URL http://arxiv.org/abs/2410.22330. arXiv:2410.22330 [cs].

[71] Liu Yang, Ziqian Lin, Kangwook Lee, Dimitris Papailiopoulos, and Robert Nowak. Task Vectors in In-Context Learning: Emergence, Formation, and Benefit, January 2025. URL http://arxiv.org/abs/2501.09240. arXiv:2501.09240 [cs].

[72] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. February 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

[73] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, January 2019. URL http://arxiv.org/abs/1711.05101. arXiv:1711.05101 [cs, math].

[74] Stability AI. stabilityai/stable-diffusion-xl-base-1.0 · Hugging Face, January 2025. URL https://huggingface.co/stabilityai/stable-diffusion-xl-base-1.0.

[75] Stability AI. stabilityai/stable-diffusion-3-medium · Hugging Face, January 2025. URL https://huggingface.co/stabilityai/stable-diffusion-3-medium.

[76] Stability AI. stabilityai/stable-diffusion-3.5-medium · Hugging Face, January 2025. URL https://huggingface.co/stabilityai/stable-diffusion-3.5-medium.

[77] Google Deepmind. Gemini 2.0 Flash, 2025. URL https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-0-flash.

[78] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[79] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2023. URL http://github.com/google/flax.

[80] Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL http://github.com/deepmind.

[81] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

[82] Simon Willison. llm: CLI utility and Python library for interacting with Large Language Models from organizations like OpenAI, Anthropic and Gemini plus local models installed on your own machine., 2023. URL https://github.com/simonw/llm.

[83] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[84] Lukas Biewald. Experiment Tracking with Weights and Biases, 2020. URL https://www.wandb.com/.

[85] Plotly Technologies Inc. Collaborative data science, 2015. URL https://plot.ly. Place: Montreal, QC Publisher: Plotly Technologies Inc.

[86] Charlie Marsh. uv: An extremely fast Python package and project manager, written in Rust., 2024. URL https://pypi.org/project/uv/.

Appendix

Table of Contents

A Proofs

A.1 Connected support

We briefly state the definition of connected support as introduced by [18] using our notation.

Definition A.1 (Connected support)**.** Let $\mathcal{T} = (C, p : \boldsymbol{z} \mapsto p(\boldsymbol{z}), p : \boldsymbol{x} \mapsto p(\boldsymbol{x}))$ be a compositional task family, and let $\boldsymbol{z} \mapsto p^{\text{train}}(\boldsymbol{z})$ be a distribution over $\mathcal{Z}$ with discrete support $\text{supp}(p^{\text{train}}) \subseteq \mathcal{Z}$. We say that $\text{supp}(p^{\text{train}})$ is connected if the graph $G = (V, E)$ is connected, where $V = \{\boldsymbol{z} \in \mathcal{Z} \mid p^{\text{train}}(\boldsymbol{z}) > 0\}$ is the set of vertices and $E = \{(\boldsymbol{z}, \boldsymbol{z}') \in V \times V \mid \exists i \in \{1, 2, \ldots, M\} \text{ such that } z_i = z'_i\}$ is the set of edges.

In other words, two training task constituents are connected by an edge if and only if they share at least one element at the same position. Section A.3 of [18] provides an example that demonstrates how a student hypernetwork can perfectly fit the training distribution of a different teacher hypernetwork if the task support is compositional but disconnected. This implies that there exist two distinct hyperteachers that have the same training distribution. In such cases, any student, including the multilayer perceptron considered here, is not guaranteed to generalize even when perfectly fitting the training tasks. To avoid those cases, connected support is generally required for the hyperteacher.

A.2 Proof of Theorem 3.1

In the following, we will show that a multilayer perceptron can approximate a hyperteacher using a linear number of neurons in the number of task modules. We first state the result for the single layer hyperteacher we primarily consider in the main text, before extending the result to hyperteachers with multiple layers.

A.2.1 Single layer hyperteacher

Let us recall the definition of the hyperteacher in Equation 1 with M modules, I input neurons, H hidden neurons and O output neurons,

$$(\boldsymbol{x}, \boldsymbol{z}) \mapsto \boldsymbol{\Omega}\, \text{ReLU}\left(\sum_{m=1}^{M}\sum_{i=1}^{I} \boldsymbol{\Theta}_{m,i} z_m x_i\right),$$

where $\{\boldsymbol{\Theta}_m \in \mathbb{R}^{I\times H}\}_{m=1}^{M}$ are the modules, $\boldsymbol{\Omega} \in \mathbb{R}^{H\times O}$ is a readout projection and $\boldsymbol{z} \in [0,1]^M$ are the task constituents. We are restating Theorem 3.1 here for convenience.

Theorem 3.1. *Let $\left(\boldsymbol{\Theta}_m \in \mathbb{R}^{I\times H}\right)$ be a sequence of uniformly bounded matrices. Then, for any $M \in \mathbb{N}$, $\varepsilon > 0$, and on any compact input set, $\mathcal{X} \times \mathcal{Z}$ with $\mathcal{Z} = \{\boldsymbol{z} : \|\boldsymbol{z}\|_1 \leq 1\}$, there exists a ReLU multilayer perceptron that approximates a hyperteacher to within ε error in the $\|\cdot\|_\infty$ norm using $\mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}} + M\right)$ neurons.*

Proof. We will prove the statement, by providing an explicit construction of a ReLU multilayer perceptron that approximates a hyperteacher. The construction relies on two key lemmas: in Lemma A.2, we show that the square function can be approximated in a ReLU multilayer perceptron, which we use in Lemma A.3 to show how the multiplication of two numbers can be approximated. Based on these building blocks, we then provide the construction of the full hyperteacher approximation.

Lemma A.2. *For any $\varepsilon > 0$, there exists a ReLU multilayer perceptron that approximates*

$$\text{square} := \left\{ \begin{array}{ccc} [0,1] & \to & \mathbb{R} \\ x & \mapsto & x^2 \end{array} \right.$$

to within ε error in the $\|\cdot\|_\infty$ norm using $\mathcal{O}(1/\varepsilon^2)$ neurons.

Proof. Let us first consider how any function f can be approximated by a piecewise linear function L matching f on $x_i := \frac{i}{n}$ for $i \in \{0, 1, \ldots, n\}$, where n is a fixed integer. We define

$$L : x \mapsto f(0) + \sum_i \Big(\frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}\text{ReLU}(x - x_i) - \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}\text{ReLU}(x - x_{i-1})\Big)$$

where $\frac{f(x_0)-f(x_{-1})}{x_0-x_{-1}}$ is set to 0 by convention.

It can easily be verified that L is linear on any interval $[x_i, x_{i+1}]$ for $i \in \{0, 1, \ldots, n\}$, and coincides with f on $\left\{\frac{i}{n}, i \in \{0, 1, \ldots, n\}\right\}$. We now want to bound $\|f - L\|_\infty$. While more general results can be shown for $f \in \mathcal{C}^2$, for $f = \text{square}$ we can derive our result in the following simple way: for any x on any interval $[x_i, x_{i+1}]$,

$$L(x) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) + f(x_i)$$

so

$$\begin{aligned} |L(x) - f(x)| &= \left| \frac{x_{i+1}^2 - x_i^2}{x_{i+1} - x_i}(x - x_i) + x_i^2 - x^2 \right| \\ &= (x - x_i)(x_{i+1} - x) \\ &\leq \varepsilon \qquad \text{for } n \geq 1/\sqrt{\varepsilon} \end{aligned}$$

□

An immediate corollary is that this result holds on any fixed bounded set. We now show how to multiply two numbers using a ReLU multilayer perceptron:

Lemma A.3. *For any $\varepsilon > 0$, there exists a ReLU multilayer perceptron that approximates*

$$\text{multiply} := \left\{ \begin{array}{ccc} [0,1]^2 & \to & \mathbb{R} \\ (x, y) & \mapsto & xy \end{array} \right.$$

to within ε error in the $\|\cdot\|_\infty$ norm using $\mathcal{O}(1/\varepsilon^2)$ neurons.

Proof. Using the polarization identity $xy = \frac{(x+y)^2-(x-y)^2}{4}$, we can approximate $\text{multiply}(x, y)$ with a ReLU multilayer perceptron as follows:

$$(x, y) \mapsto (x + y, x - y) \mapsto ((x + y)^2, (x - y)^2) \mapsto xy$$

□

Again, this result holds on any fixed bounded set.

We can now state a construction for a multilayer perceptron that approximates the preactivation of the hyperteacher, namely $(\boldsymbol{x}, \boldsymbol{z}) \mapsto \sum_{m=1}^{M} \sum_{i=1}^{I} \boldsymbol{\Theta}_{m,i} z_m x_i$ using a linear number of neurons in the number of task modules.

Lemma A.4. *Let $\left(\boldsymbol{\Theta}_m \in \mathbb{R}^{I \times H}\right)$ be a sequence of uniformly bounded matrices. Then, for any $M \in \mathbb{N}$, $\varepsilon > 0$, and on any compact input set, $\mathcal{X} \times \mathcal{Z}$ with $\mathcal{Z} = \{\boldsymbol{z} : \|\boldsymbol{z}\|_1 \leq 1\}$, there exists a ReLU multilayer perceptron that approximates $(\boldsymbol{x}, \boldsymbol{z}) \mapsto \sum_{m=1}^{M} \sum_{i=1}^{I} \boldsymbol{\Theta}_{m,i} z_m x_i$ to within ε error in the $\|\cdot\|_\infty$ norm using $\mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}} + M\right)$ neurons.*

Proof. We construct our multilayer perceptron layer by layer, tracking the number of neurons required for each layer and the error it incurs in the approximation:

- The first layer copies the input $\boldsymbol{x}$ and computes $(\boldsymbol{x}, \boldsymbol{z}) \mapsto (\sum_{m=1}^{M} z_m \Theta_{m,i,h})_{i,h}$ for each $i \in \{1, \ldots, I\}$ and $h \in \{1, \ldots, O\}$, canceling the ReLUs by adjusting the biases accordingly given that the input is bounded. This requires $\mathcal{O}(M)$ neurons.
- Using Lemma A.3 with error $\frac{\varepsilon}{I}$, the next three layers multiply $\sum_{m=1}^{M} z_m \Theta_{m,i,h}$ by x_i for each $i \in \{1, \ldots, I\}$ and $h \in \{1, \ldots, O\}$, using $\mathcal{O}(1/\sqrt{\varepsilon})$ neurons.
- The next layer sums over i for each $h \in \{1, \ldots, O\}$, again canceling the ReLU nonlinearity by adjusting the biases. This requires M neurons and incurs a final error of at most ε since each output neuron sums the error coming from I neurons.

In total, this construction uses $\mathcal{O}(M) + \mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}}\right) + \mathcal{O}(M) = \mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}} + M\right)$ neurons. □

Since ReLU is a contracting function, the output error after applying the ReLU activation is also ε. Finally, we can straightforwardly apply the output projection $\boldsymbol{\Omega}$ using a constant number of neurons, scaling the error by a constant. □

A.2.2 Multilayer hyperteacher

We now consider a multilayer hyperteacher, where a linear hypernetwork configures a multilayer perceptron with L layers, H_l neurons for layer l, $H_0 = I$ input neurons and $H_L = O$ output neurons. In this case, we have $C(\boldsymbol{z}) \coloneqq \boldsymbol{\Omega}\boldsymbol{h}_L$, a sequence of hidden layers,

$$\boldsymbol{h}_{l+1} = \mathrm{ReLU}\Big(\underbrace{\sum_{m=1}^{M} z_m \boldsymbol{\Theta}_{l,m}}_{=:\boldsymbol{W}_l(\boldsymbol{z})} \boldsymbol{h}_l\Big),$$

with input $\boldsymbol{h}_0 = \boldsymbol{x}$, a sequence of modules for each layer, $\left\{\boldsymbol{\Theta}_{l,m} \in \mathbb{R}^{H_l \times H_{l+1}}\right\}_{\substack{l=1,\dots,L \\ m=1,\dots,M}}$, and output projection $\boldsymbol{\Omega} \in \mathbb{R}^{H_{L-1} \times O}$.

Theorem A.5. *Let $\left(\boldsymbol{\Theta}_{l,m} \in \mathbb{R}^{H_l \times H_{l+1}}\right)$ be a sequence of uniformly bounded matrices. Then, for any $M \in \mathbb{N}$, $\varepsilon > 0$, fixed L and on any compact input set, $\mathcal{X} \times \mathcal{Z}$ with $\mathcal{Z} = \{\boldsymbol{z} : \|\boldsymbol{z}\|_1 \leq 1\}$, there exists a ReLU multilayer perceptron that approximates an L-layer hyperteacher to within ε error in the $\|\cdot\|_\infty$ norm using $\mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}} + M\right)$ neurons.*

Proof. First, observe that we can copy the task constituents $\boldsymbol{z}$ to each layer using M neurons per layer. Because of this, we will assume that we have access to the task constituents at each layer. The proof then follows a similar approach to the proof of the single layer case of Theorem 3.1 but we must now consider how the error propagates through each layer. For the sake of simplicity, we will ignore the readout projection (i.e. assume $\boldsymbol{\Omega} = \boldsymbol{I}$), since this part of the proof is identical.

We prove by induction on the number of layers L that we can approximate $\boldsymbol{h}_L$ to within ε error in the $\|\cdot\|_\infty$ norm with $\mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}} + M\right)$ neurons. The base case for $L = 0$ holds by definition, i.e. $\boldsymbol{h}_0 = \boldsymbol{x}$. Now, assume the result holds for an L-layer hyperteacher. Let $\varepsilon > 0$ and $\mathcal{X}$, $\mathcal{Z}$ be compact sets with $\mathcal{Z} = \{\boldsymbol{z} : \|\boldsymbol{z}\|_1 \leq 1\}$. By the induction hypothesis, let $\hat{h}_L$ be a multilayer perceptron that approximates $\boldsymbol{h}_L$ on the compact set $\mathcal{X} \times \mathcal{Z}$, up to error ε in the $\|\cdot\|_\infty$ norm.

By Lemma A.4, let g be a multilayer perceptron that approximates $(\boldsymbol{x}, \boldsymbol{z}) \mapsto \sum_{m=1}^{M} z_m \boldsymbol{\Theta}_{L+1,m} \boldsymbol{x}$ on $\bar{B}_\varepsilon \times \mathcal{Z}$, where $\bar{B}_\varepsilon$ is the closed ε-ball around $\hat{h}_L(\mathcal{X} \times \mathcal{Z})$. Since the image of a compact set formed by a continuous map is compact and the closed epsilon ball around a compact set in a finite-dimensional space is compact, $\bar{B}_\varepsilon$ is also a compact set.

Let us now show that $(\boldsymbol{x}, \boldsymbol{z}) \mapsto \mathrm{ReLU}\left(g(\hat{h}_L(\boldsymbol{x}, \boldsymbol{z}), \boldsymbol{z})\right)$ approximates $\boldsymbol{h}_{L+1}$ on $\mathcal{X} \times \mathcal{Z}$ up to error $\mathcal{O}(\varepsilon)$. We define $\boldsymbol{\Delta}_L \coloneqq \hat{h}_L(\boldsymbol{x}, \boldsymbol{z}) - \boldsymbol{h}_L$ for L layers. Then, for $L + 1$ layers it holds that,

$$\begin{aligned}
\|\boldsymbol{\Delta}_{L+1}\|_\infty &= \|\mathrm{ReLU}(g(\boldsymbol{h}_L + \boldsymbol{\Delta}_L, \boldsymbol{z})) - \boldsymbol{h}_{L+1}\|_\infty \\
&= \|g(\boldsymbol{h}_L + \boldsymbol{\Delta}_L, \boldsymbol{z}) - \boldsymbol{W}_L(\boldsymbol{z})\boldsymbol{h}_L\|_\infty \qquad \text{(since ReLU is contracting)} \\
&= \|g(\boldsymbol{h}_L + \boldsymbol{\Delta}_L, \boldsymbol{z}) - \boldsymbol{W}_L(\boldsymbol{z})(\boldsymbol{h}_L + \boldsymbol{\Delta}_L) + \boldsymbol{W}_L(\boldsymbol{z})\boldsymbol{\Delta}_L\|_\infty \\
&\leq \|g(\boldsymbol{h}_L + \boldsymbol{\Delta}_L, \boldsymbol{z}) - \boldsymbol{W}_L(\boldsymbol{z})(\boldsymbol{h}_L + \boldsymbol{\Delta}_L)\|_\infty + \|\boldsymbol{W}_L(\boldsymbol{z})\|_{\infty\to\infty}\|\boldsymbol{\Delta}_L\|_\infty \\
&= \varepsilon + \mathcal{O}(\varepsilon) = \mathcal{O}(\varepsilon),
\end{aligned}$$

where $\|\cdot\|_{\infty\to\infty}$ is the operator norm induced by the $\|\cdot\|_\infty$ norm. The final error can be reduced to be at most ε by adjusting the number of neurons by a constant factor.

Finally, let us bound the number of neurons required for the full construction. We used $\mathcal{O}(M)$ neurons for copying $\boldsymbol{z}$ to each layer, $\mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}} + M\right)$ neurons for the construction of $\hat{h}_L$, and $\mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}} + M\right)$ neurons for the construction of g, which sums up to $\mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}} + M\right)$ neurons. □

A.3 Proof of Theorem 4.1

Proof. Let X, Z, R be independent random variables sampled according to their respective distributions. We define the event made of the set of pairs of task constituents $\boldsymbol{z}$ and seeds r such that the accuracy is higher than $1 - \sqrt{\delta}$:

$$\begin{aligned} A &\coloneqq \left[\mathbb{E}\left[\mathbb{1}\{g(\varphi(Z,R),X) = C(Z)(X)\}|Z,R\right] \geq 1-\sqrt{\delta}\right] \\ &= \left\{(\boldsymbol{z},r) \in \mathcal{Z}\times\mathbb{N} \mid \mathbb{P}[g(\varphi(\boldsymbol{z},r),X) = C(\boldsymbol{z})(X)] \geq 1-\sqrt{\delta}\right\} \end{aligned}$$

where the expectation is taken over X. By Markov's inequality, and the first assumption, this set can't be too small. Indeed,

$$\begin{aligned} \mathbb{P}[\neg A] &< \frac{\mathbb{E}\left[\mathbb{1}\{g(\varphi(Z,R),X) \neq C(Z)(X)\}\right]}{\sqrt{\delta}} && \text{by Markov's inequality} \\ &< \sqrt{\delta} && \text{by the first assumption} \end{aligned}$$

We now show that for all $(\boldsymbol{z},r),(\boldsymbol{z'},r') \in A \subset \mathcal{Z}\times\mathbb{N}$

$$\varphi(\boldsymbol{z},r) = \varphi(\boldsymbol{z'},r') \implies \boldsymbol{z} = \boldsymbol{z'}.$$

Indeed, let $(\boldsymbol{z},r),(\boldsymbol{z'},r') \in A \subset \mathcal{Z}\times\mathbb{N}$, with $\hat{\boldsymbol{z}} \coloneqq \varphi(\boldsymbol{z},r) = \varphi(\boldsymbol{z'},r')$.

We have, by definition of A,

$$\begin{aligned} \mathbb{P}[g(\hat{\boldsymbol{z}},X) = C(\boldsymbol{z})(X)] \geq 1-\sqrt{\delta} \\ \mathbb{P}[g(\hat{\boldsymbol{z}},X) = C(\boldsymbol{z'})(X)] \geq 1-\sqrt{\delta} \end{aligned}$$

so by union-bound, it holds

$$\mathbb{P}[C(\boldsymbol{z})(X) = C(\boldsymbol{z'})(X)] \geq 1-2\sqrt{\delta}$$

which is only possible if $\boldsymbol{z} = \boldsymbol{z'}$ given the second assumption.

We now define

$$\phi : \begin{cases} \mathcal{Z}' & \to & \mathcal{Z} & & \\ \hat{\boldsymbol{z}} & \mapsto & \boldsymbol{z} & \text{for any } (\boldsymbol{z},r) \in \varphi^{-1}(\hat{\boldsymbol{z}}) & \text{if } \hat{\boldsymbol{z}} \in \varphi(A) \\ \hat{\boldsymbol{z}} & \mapsto & \mathbf{0} & & \text{otherwise} \end{cases}$$

ϕ is uniquely defined because of the previous property.

We have

$$\mathbb{P}[\phi(\varphi(Z,R)) = Z] \geq \mathbb{P}[(Z,R) \in A] \geq 1-\sqrt{\delta}$$

which proves our statement. □

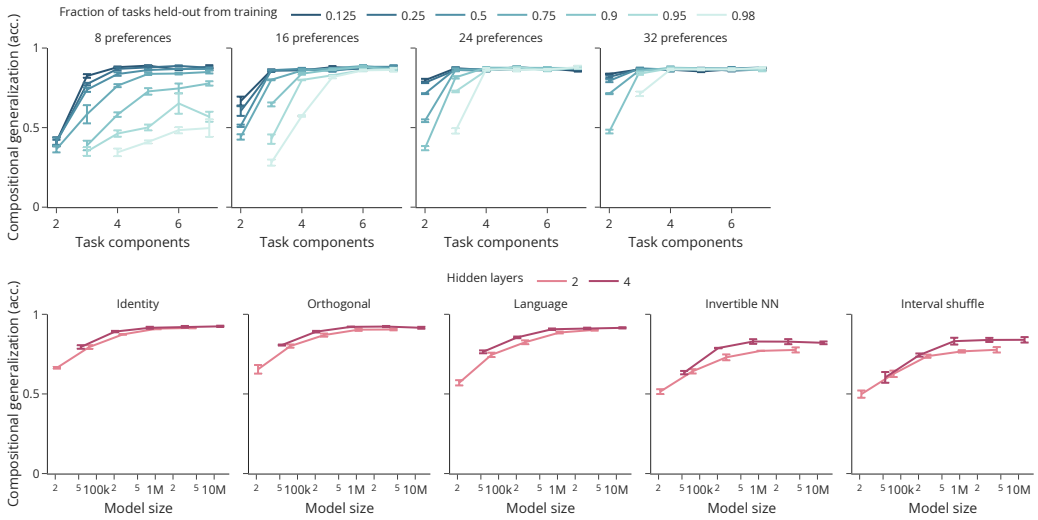

Figure 7: **Scaling data and model size leads to compositional generalization.** ***Top*** Scaling the number of training tasks by increasing the number of modules, task components or decreasing the fraction of tasks held-out from training leads to compositional generalization on the compositional preference task family. ***Bottom*** Scaling model size by increasing the number of hidden neurons or the number of hidden layers leads to compositional generalization on the compositional preference task family across different task encodings. Error bars denote SEM over three seeds.

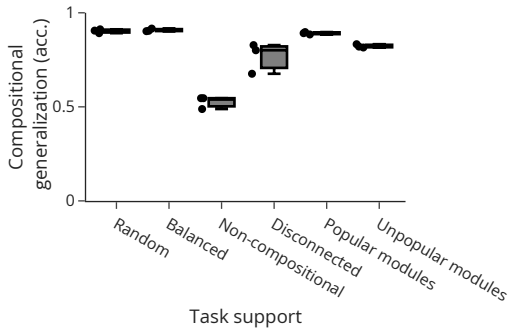

Figure 8: **The support of the training distribution needs to sufficiently cover the task space.**: Compositional generalization as a function of the different types of task support on the compositional preference task family for $M = 16$ and $K = 3$.

B Additional results

B.1 Compositional generalization on the preference grid

The findings in Section 3 and Section 4.1 focus on the hyperteacher task family. In the following, we show that all findings can be reproduced on the compositional preference family introduced by [18].

In Figure 7 we show, that scaling data and model size leads to compositional generalization on the compositional preference task family. This holds for various linear and nonlinear task encodings as further shown in Table 2. Note, that we are not able to evaluate the fewshot task encoding of the main text since the resulting input dimension is prohibitively large. Figure 8 shows that non-compositional and disconnected task support as well as rarely encountering a few modules interferes with compositional generalization. And finally, Figure 9 demonstrates that also on the compositional preference task family, compositional generalization and linear decodability of the task constituents from the hidden activations of the model are correlated.

Table 2: **Compositional generalization emerges across task encodings.** Comparison of the decodability of the task constituents from the hidden activations (Task decoder) and compositional generalization performance for linear and nonlinear task encodings on the compositional preference task family with $M = 16$, $K = 3$. For the linear Identity, Orthogonal and Language task encoding, we report the decodability of task constituents from the first layer whereas for the nonlinear Invertible NN (with 2 layers) and Interval shuffle encodings we report it for the second layer. As opposed to linear task encodings, for nonlinear task encodings the decodability of task constituents is higher in the second layer compared to the first layer suggesting that the first layer is used to linearize the task constituents. We additionally show the linear decodability of the task constituents directly from the task encoding itself (Input decoder), which allows to distinguish linear from nonlinear task encodings. $\pm$ SEM over three seeds.

Task encoding	Task decoder (R^2)	Embedding decoder (R^2)	Comp. gen. (acc.)
Identity	0.95 ± 0.012	1.00 ± 0.000	0.92 ± 0.004
Orthogonal	0.96 ± 0.008	1.00 ± 0.000	0.92 ± 0.006
Language	0.98 ± 0.004	1.00 ± 0.000	0.92 ± 0.004
Invertible NN	0.93 ± 0.005	0.74 ± 0.018	0.82 ± 0.009
Interval shuffle	0.90 ± 0.022	0.72 ± 0.083	0.84 ± 0.017

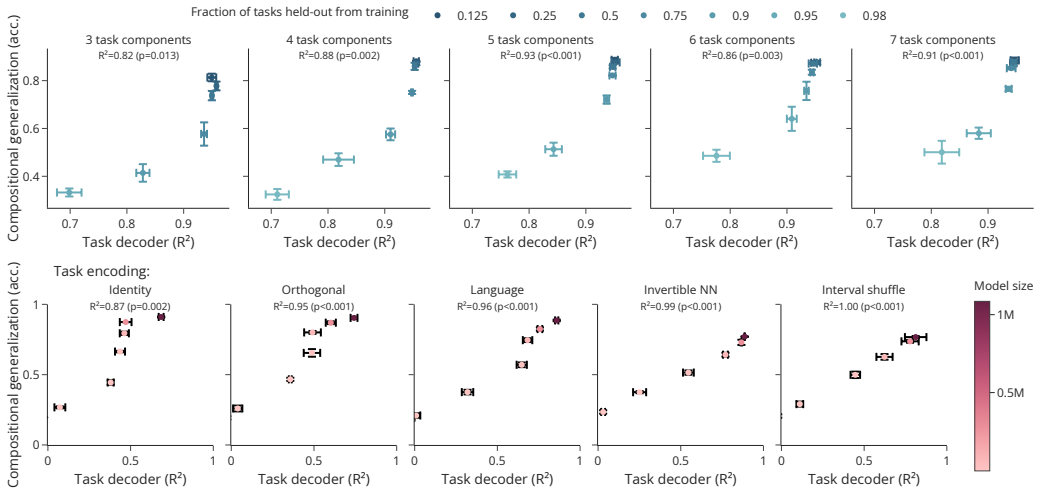

Figure 9: **Compositional generalization correlates with linear decodability of task constituents.** ***Top*** Relationship between linear decodability of the task constituents and compositional generalization across instantiations of the compositional preference task family with $M = 8$ preferences and varying K. ***Bottom*** Relationship between linear decodability of the task constituents and compositional generalization across different task encodings for varying model sizes on the compositional preference task family with $M = 16$, $K = 3$. Error bars denote SEM over three seeds. ***Top/Bottom*** We report the R^2 and corresponding p-value for an ordinary least square estimator in the facet titles.

B.2 Scaling data leads to compositional generalization in transformers

In addition to using multilayer perceptrons as done in the main text, we investigate whether the widely used transformer architecture similarly achieves compositional generalization through scale. For this purpose, we train a decoder-only transformer that takes a sequence consisting of the task constituents followed by the task inputs as input and predicts the corresponding labels. Given that the transformer contains multilayer perceptrons in the feedforward blocks, we expect it to be similarly capable of compositional generalization at scale.

The top of Figure 10 confirms this, showing that scaling data similarly leads to compositional generalization in transformers. Compared to the multilayer perceptron, the transformer requires less

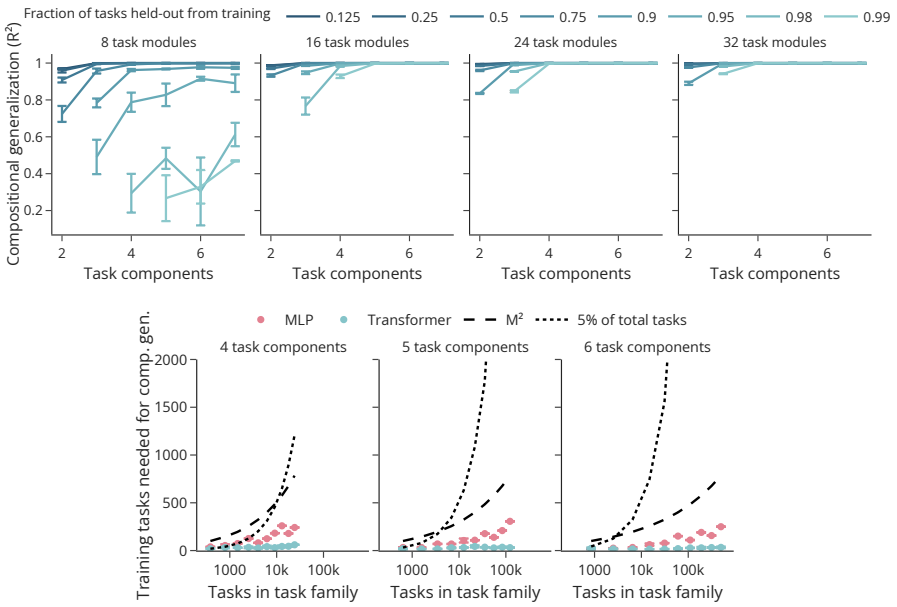

Figure 10: **Scaling data leads to compositional generalization in transformers.** ***Top*** Scaling the number of training tasks by increasing the number of modules, task components or decreasing the fraction of tasks held-out from training leads to compositional generalization in transformers on the hyperteacher task family. ***Bottom*** Transformers require less training tasks to achieve compositional generalization ($R^2 > 0.95$) compared to multilayer perceptrons (MLP).

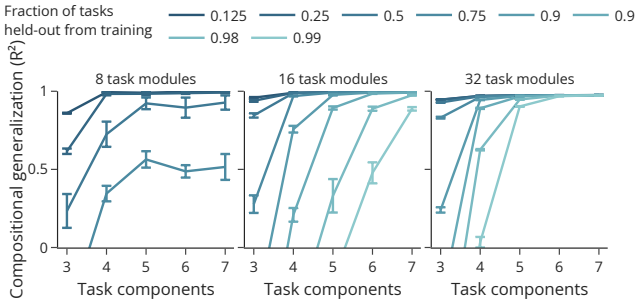

Figure 11: **Scaling data leads to compositional generalization on a deep hyperteacher.** Scaling the number of training tasks by increasing the number of modules, task components or decreasing the fraction of tasks held-out from training leads to compositional generalization on the hyperteacher task family with a deep task network with three hidden layers.

distinct training tasks to achieve compositional generalization, as can be observed at the bottom of Figure 10.

B.3 Scaling data leads to compositional generalization in a deep hyperteacher

In the main text, we focus on a hyperteacher with a task network with a single hidden layer. A natural question is how increasing the difficulty of the hyperteacher by equipping the task network with multiple hidden layers affects our results. For this purpose, we reproduce the data scaling plot shown in Figure 2 of Section 3 for a hyperteacher with three hidden layers, each with 16 hidden neurons. Figure 11 demonstrates that while this makes the task noticeably more difficult, it reproduces our finding that scaling data leads to compositional generalization.

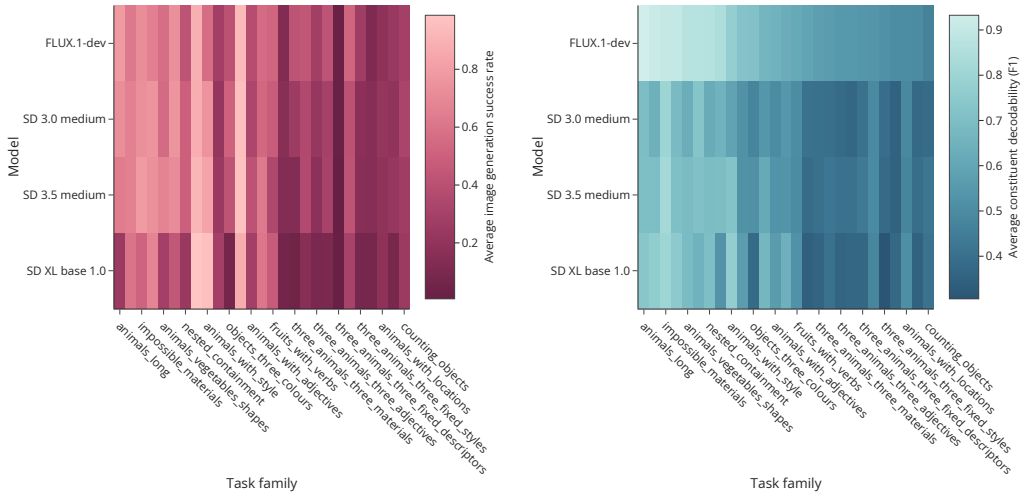

Figure 12: **Image composition task difficulties sorted by linear decodability of task constituents. Left** Average image generation success for each image composition task family for different text-to-image generation models. **Left** Average linear decodability of constituents from hidden activations for each image composition task family for different text-to-image generation models.

B.4 Difficulty of image composition task family

To complement Figure 6 of the main text, Figure 12 shows the task difficulty as well as the average linear decodability of the task constituents from the hidden activations for all the image composition task families and text-to-image generation models we consider.

C Experimental details

C.1 Task families

We create the hyperteacher task family described in Section 2.4 with $I = 16$ input neurons, $H = 16$ hidden neurons and $O = 16$ output neurons to create a family of compositional regression tasks to be learned by a student. We sample teacher modules from a truncated normal distribution with zero mean and standard deviation $\sqrt{\frac{\sqrt{3}}{I}}$. Each teacher module also has a bias term that is sampled from the uniform distribution over the interval $[0, 0.5]$. We sample the fixed readout matrix shared by all tasks from a truncated normal distribution with zero mean and standard deviation $\sqrt{\frac{\sqrt{3}}{H}}$. These values have been picked as to ensure that the hyperteacher creates tasks of sufficient diversity and difficulty.

For the definition of the compositional preference task family, please refer to [18]. Following the notation of the hyperteacher, we denote the number of possible preferences as M and the maximum number of preferences combined into a task as K. [18] uses $M = 8$ number of possible preferences throughout. We increase the difficulty of the tasks by increasing the number of preference features to $M = 16$ in Figure 7 bottom, Figure 8 and Figure 9 and to $M = 32$ in Figure 7 top.

C.2 Task encoding

We employ the invertible neural network introduced by [72] with 5 layers (3 for the compositional preference task family) that consists of a series of bijective transformations, to obtain a nonlinear but information-preserving encoding of the task constituents. In addition, we use the Interval Shift function defined in Algorithm 1 as another example of a nonlinear but bijective task encoding.

Algorithm 1 Interval Shift

Number of intervals: N
function $\varphi(z, r)$
    $\pi \leftarrow \text{GeneratePermutation}(N, r)$ ▷ Generate permutation of [0,N-1] using seed r
    $i \leftarrow \lfloor N \cdot z \rfloor$ ▷ Determine which interval z belongs to
    $\alpha \leftarrow N \cdot z - i$ ▷ Fractional position within interval
    $j \leftarrow \pi(i)$ ▷ Apply permutation
    return $\frac{j+\alpha}{N}$ ▷ Preserve relative position within new interval
end function

C.3 Training task support

We investigate the effect of various procedures to construct the training task support on compositional generalization, which we describe in the following. For this purpose, consider the graph $G = (V, E)$ where $V = \{\boldsymbol{z} \in \mathcal{Z} \mid p^{\text{train}}(\boldsymbol{z}) > 0\}$ is the set of vertices and $E = \{(\boldsymbol{z}, \boldsymbol{z}') \in V \times V \mid \exists i \in \{1, 2, \ldots, K\}$ such that $z_i = z_i'\}$ is the set of edges.

- **Random**: Samples a random subset of vertices until a graph that is both compositional and connected is found.
- **Balanced**: Similar to Random, but ensures that each module appears with equal frequency in the training distribution using a greedy search over vertices.
- **Non-compositional**: Holds out all tasks that contain one random but fixed module.
- **Disconnected**: Divides modules into two disjoint subsets and only uses vertices that use modules from either subset but do not contain modules from both subsets.
- **Popular modules**: Defines a set of P popular modules and only includes vertices that contain at least one popular module. P is determined such that the fraction of held-out tasks can be satisfied as specified. If it cannot be exactly satisfied, one module that is not in the set of popular modules receives additional vertices. Additionally ensures that the resulting set is compositional and connected.
- **Unpopular modules**: Defines a set of U unpopular modules and includes all vertices that do not contain any unpopular module. For each unpopular module, one vertex that includes the unpopular module and otherwise only not unpopular modules is added. U is determined such that the fraction of held-out tasks can be satisfied as specified. If it cannot be exactly satisfied, one unpopular module receives additional vertices. Additionally ensures that the resulting set is compositional and connected.

C.4 Task constituent decoding

On the hyperteacher and compositional preference task family, we fit a linear decoder using ridge regression to predict the task constituents given the hidden activations of a particular layer of the multilayer perceptron solving the task. For this purpose, we train on pairs of hidden activations and ground truth task constituents from the training distribution, $p^{\text{train}}(\boldsymbol{z})$, and evaluate the performance of the decoder on held-out tasks, reporting the coefficient of determination (R^2 score). Throughout, we employ a regularizer of $\lambda = 1.0$ for the ridge regression.

C.5 Architecture

Unless specified otherwise, we use a multilayer perceptron with four hidden layers with 1024 hidden neurons each for the hyperteacher and with two hidden layers for the compositional preference task family.

The transformer in Section B is causally masked and consists of 4 layers with 4 attention heads, a model dimension of 256, a feedforward dimension of 1024 and separate projection matrices for the task constituents, inputs and the output.

C.6 Hyperparameters

Throughout our experiments, we use the AdamW optimizer [73] with a batch size of 128. On the hyperteacher task family, we use a mean-squared error loss, on the compositional preference task family, we use a cross-entropy loss. We performed an initial grid search over the learning rate and weight decay to find a common set of hyperparameters for all experiments on the hyperteacher task family and a common set of hyperparameters for all experiments on the compositional preference tasks respectively. We report the search grid in Table 3.

Table 3: Hyperparameters for experiments on the hyperteacher task family and compositional preference task family. Lists of values denote parameters explored via grid search with a bold number indicating the value found to perform best and used throughout the experiments.

Parameter	Hyperteacher	Compositional preferences
`learning_rate`	$[0.001, 0.003, 0.0001, \mathbf{0.0003}]$	$[0.001, 0.003, \mathbf{0.0001}, 0.0003]$
`weight_decay`	$[\mathbf{0.003}, 0.001, 0.0003]$	$[0.003, \mathbf{0.001}, 0.0003]$

C.7 Compositional text-to-image generation

Models We compare four open-weight text-to-image generation models: `FLUX.1-dev` [29, FLUX.1 dev Non-Commercial License], `SD XL base 1.0` [74, Open RAIL++-M License], `SD 3.0 medium` [75, Stability AI Community License] `SD 3.5 medium` [76, Stability AI Community License].

Experimental setup In the following, we describe the experimental setup illustrated in Figure 6. For each image composition task family listed below, we prompt each model on all possible image combinations to generate an image using 40 inference steps. During this process, we collect their hidden activations to be used by the image constituent decoder (see below). We then prompt a VLM to judge whether the image constituents have been correctly combined into a coherent image by asking it to readout the image constituents given the image and the full set of possible constituents. We count an image generation as successful if the image constituents generated by the VLM exhaustively match the ground truth constituents. Here, we report results using Gemini 2.0 Flash [77] as the judge.

Image constituent decoder We train standard logistic regression classifiers to predict image constituents given the hidden activations of the text-to-image generation models and report their F1 scores on held-out image compositions. Not all layers of the respective models contain this information and we need to carefully choose layers depending on the model architecture. In particular, the image constituents can trivially be linearly decoded from early layers of the model that encode the text prompt, as well as any layers that are directly connected to the text encoding either via residual connections or via cross-attention. For this reason, we select layers that require the model to explicitly learn to pass information about image constituents to. In the transformer-based models `FLUX.1-dev`, `SD 3.5 medium` and `SD 3.0 medium`, we probe the hidden activation of the penultimate transformer block's MLP. `SD XL base 1.0` contains a bottleneck block, so we probe the hidden activations at the final layer of the bottleneck. Since the models we are considering are diffusion models that run inference over several time steps, we must specify at what time during the inference process to collect the hidden activations. In our experiments, we run 30 inference steps and collect hidden activations at time steps 9, 15, 21, and 30, as well as the average hidden activation over all time steps. We concatenate all hidden activations for a given model and image and use them as the input to the linear decoder.

Image composition task families We create a variety of image composition task families that each consist of a combinatorial space of objects and attributes that need to be combined into a coherent image. Specifically, each task has the following structure with a corresponding prompt template:

- `task_name`: *Prompt template specifying how to compose {component1} and {component2}.*
  - **component1**: module1, module2, ...
  - **component2**: module1, module2, ...

We list all the tasks in the following:

- `animals_with_colours`: *A {colour} {animal}*
  - **animal**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **colour**: red, blue, green, yellow, orange, purple, pink, brown, black, white
- `animals_with_style`: *A {animal} illustrated in {style} style*
  - **animal**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **style**: watercolor, pixel art, oil painting, sketch, cartoon, origami, stained glass, pop art, charcoal, clay sculpture
- `animals_with_locations`: *A {animal} in the {location}*
  - **animal**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **location**: top left, top center, top right, middle left, center, middle right, bottom left, bottom center, bottom right
- `animals_with_descriptor`: *A {descriptor} {animal}*
  - **animal**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **descriptor**: furry, scaly, feathered, leathery, smooth, wrinkled, young, old, three-legged, spotted
- `animals_with_adjectives`: *A image of a {adjective} {animal}*
  - **adjective**: happy, sad, angry, sleepy, curious, playful, scared, proud, surprised, bored
  - **animal**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
- `fruits_with_verbs`: *A {fruit} is {verb}*
  - **fruit**: apple, banana, orange, grape, strawberry, watermelon, pineapple, mango, blueberry, peach
  - **verb**: dancing, flying, bouncing, sleeping, swimming, rolling, climbing, stretching, spinning, hiding
- `counting_animals`: *An image with exactly {number} {animal}*
  - **number**: one, two, three, four, five, six, seven, eight, nine, ten
  - **animal**: lions, elephants, giraffes, crocodiles, bears, snakes, eagles, cows, zebras, tigers
- `animals_with_verbs`: *A {animal} is {verb}*
  - **animal**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **verb**: eating, sleeping, running, jumping, flying, swimming, climbing, dancing, playing, hiding
- `counting_objects`: *An image with exactly {number} {object}*
  - **number**: one, two, three, four, five, six, seven, eight, nine, ten, eleven, twelve, thirteen, fourteen, fifteen, sixteen, seventeen, eighteen, nineteen, twenty
  - **object**: tomatoes, onions, oranges, wolves, bears, apples, bananas, carrots, cucumbers, strawberries, lemons, cherries, grapes, peaches, pears, foxes, rabbits, cats, dogs, sheep
- `nested_containment_animals`: *A {animal_outer} containing a {animal_middle} containing an {animal_inner}*
  - **animal_outer**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal_middle**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal_inner**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
- `three_animals_three_materials`: *{animal_fire} made of fire, {animal_ice} made of ice, and {animal_wood} made of wood*
  - **animal_fire**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal_ice**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal_wood**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
- `three_animals_three_verbs`: *{animal_singing} singing, {animal_eating} eating, and {animal_sleeping} sleeping*
  - **animal_singing**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal_eating**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal_sleeping**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger

- `three_animals_three_adjectives`: *A sad {animal_happy}, a happy {animal_sad}, and an angry {animal_angry}*
  - **animal_happy**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal_sad**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal_angry**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
- `animals_with_clothes`: *{animal} wearing {clothing}*
  - **animal**: cat, dog, bear, lion, elephant, giraffe, monkey, zebra, tiger, panda
  - **clothing**: hat, pair of sunglasses, scarf, bowtie, jacket, crown, tie, cape, sweater, necklace
- `animals_with_clothes_and_food`: *{animal} wearing {clothing} eating {food}*
  - **animal**: cat, dog, bear, lion, elephant, giraffe, monkey, zebra, tiger, panda
  - **clothing**: hat, pair of sunglasses, scarf, bowtie, jacket, crown, tie, cape, pair of pants, pair of boots
  - **food**: pizza, banana, ice cream, cake, hamburger, apple, watermelon, donut, sandwich, salad
- `animals_with_food_eyes_and_clothes`: *A {animal} with {food} as eyes wearing {clothing}*
  - **animal**: cat, bear, lion, elephant, giraffe, monkey, zebra, panda, wolf, rabbit
  - **food**: strawberries, oranges, burgers, watermelons, donuts, cookies, cupcakes, pizza, lemons, tomatoes
  - **clothing**: crown, cowboy hat, scarf, cape, hawaiian shirt, leather jacket, pair of pants, tuxedo, raincoat, sunglasses
- `stacked_foods`: *{food_top} on top of {food_middle} on top of {food_bottom}*
  - **food_top**: burger, pizza, salad, Sushi, taco, donut, Ice cream, pancake, Spaghetti, sandwich
  - **food_middle**: burger, pizza, salad, sushi, taco, donut, ice cream, pancake, spaghetti, sandwich
  - **food_bottom**: burger, pizza, salad, sushi, taco, donut, ice cream, pancake, spaghetti, sandwich
- `stacked_animals`: *{animal_top} on top of {animal_middle} on top of {animal_bottom}*
  - **animal_top**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal_middle**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal_bottom**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
- `animals_chasing_chain`: *{animal1} chasing {animal2} chasing {animal3}*
  - **animal1**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal2**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal3**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
- `nested_containment`: *A {container1} containing a {container2} containing a {object}*
  - **container1**: transparent cube, glass jar, wooden box, metal safe, woven basket, leather bag, ceramic pot, copper kettle, crystal sphere, rubber ball
  - **container2**: small treasure chest, porcelain teacup, silk pouch, stone bowl, paper envelope, cardboard tube, tin can, shell, gold locket, velvet case
  - **object**: diamond, living butterfly, ticking clock, miniature planet, flickering flame, drop of mercury, hologram, glowing ember, snowflake, single cell organism
- `impossible_materials`: *A {object} made entirely of {material} sitting on a {surface}*
  - **object**: chair, bicycle, bookshelf, piano, computer, refrigerator, watch, umbrella, camera, guitar
  - **material**: liquid water, fire, smoke, mirrors, ice, tree bark, glass noodles, gelatin, paper, soap bubbles
  - **surface**: clouds, ocean waves, melting ice, sand dunes, moss, broken glass, spiderwebs, lily pads, autumn leaves, foam
- `three_animals_three_fixed_styles`: *A {animal_pixel} in pixel art, a {animal_oil} in oil painting, and a {animal_cartoon} in cartoon style*
  - **animal_pixel**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal_oil**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal_cartoon**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger

- `three_animals_three_fixed_descriptors`: *A {animal_furry} with fur, a {animal_scaly} with scales, and a {animal_feathered} with feathers*
  - **animal_furry**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal_scaly**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
  - **animal_feathered**: lion, elephant, giraffe, crocodile, bear, snake, eagle, cow, zebra, tiger
- `animals_long`: *An image with {one {animal}}*[1,2,3]
  - **animals**: giraffe, lion, elephant, crocodile, bear, snake, eagle, cow, zebra, tiger, rhino, hippo, wolf, fox, deer, monkey, panda, koala, kangaroo, penguin
- `objects_three_colours`: *An image with {one {object}}*[1,2,3]
  - **objects**: red cube, green cube, blue cube, blue sphere, orange sphere, red sphere, green cylinder, purple cylinder, yellow cylinder, yellow cone, blue cone, green cone, purple pyramid, red pyramid, orange pyramid, orange cuboid, yellow cuboid, blue cuboid
- `fruits_veggies_long`: *An image with {one {fruit_veggie}}*[1,2,3]
  - **fruits_veggies**: apple, banana, orange, grape, strawberry, watermelon, pineapple, mango, blueberry, peach, lemon, cherry, carrot, broccoli, tomato, cucumber, potato, onion, pepper, lettuce
- `animals_vegetables_shapes`: *An image with {one {element}}*[1,2,3]
  - **animals_vegetables_shapes**: lion, elephant, giraffe, tiger, bear, zebra, monkey, carrot, broccoli, potato, tomato, cucumber, onion, pepper, cube, sphere, cylinder, cone, pyramid, triangular prism

D Additional details

D.1 Compute resources

We used a Linux workstation with two Nvidia RTX 3090 GPUs with 24GB of memory each for development and conducted hyperparameter searches and experiments on an internal Slurm cluster using Nvidia RTX 4090 GPUs and Nvidia A100 GPUs.

A single run of a hyperteacher experiment takes 4-10 minutes on a RTX 4090 depending on model size, a single run of a compositional preference experiment takes 50-100 minutes. In total, reproducing all experiments reported for the hyperteacher task family with around 1000 distinct runs takes about 7 GPU days and reproducing all corresponding experiments for the compositional preference task family takes about 70 GPU days.

Generating the images for one of the image composition task families takes 8 GPU hours on an A100 for the `Flux.1-dev` model and 4 GPU hours on an RTX 4090 for each of the `SD` models. Running all 27 tasks for all models takes a total of 23 GPU days.

D.2 Software and libraries

For the results obtained in this paper we built on free and open-source software. We implemented our experiments in Python using JAX [78, Apache License 2.0], Flax [79, Apache License 2.0], the Deepmind Jax Ecosystem [80, Apache License 2.0], PyTorch [BSD-style license 81], LLM [82, Apache License 2.0] and Scikit-learn [83, BSD 3-Clause License]. We utilized WandB [84, MIT license] to monitor the progress and results of experiments, and Plotly [85, MIT license] for generating the plots. We use uv for Python project dependency management [86, MIT License].

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes the abstract includes all relevant claims and no unsupported claims.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the discussion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Yes for all proof assumptions are stated.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes] ,

   Justification: All experiments are explained and the code will be made public.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: we will submit the code which allows one to reproduce the results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental details can be found in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Where applicable at least three seeds with error bars were run.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The details about compute ressources can be found in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes we conformed with the Code of Ethics

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: As part of the discussion.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: There is no meaningful risk associated with the submission.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Licensees are respected and listed in the appendix.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The code is provide and documented.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No human subjects or croudsourcing was used.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: No humand participants were part of the study.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Yes we describe how we use LLMs to evaluate image generation success.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.