# What Drives Compositional Generalization in Visual Generative Models? The Importance of Continuous Training Objectives

**Karim Farid**[*1] **Rajat Sahay**[*1,2] **Yumna Ali**[*1] **Simon Schrodi**[1]

**Volker Fischer**[2] **Cordelia Schmid**[3] **Thomas Brox**[1]

[1]University of Freiburg [2]Bosch Center for Artificial Intelligence
[3]Inria, École Normale Supérieure, CNRS, PSL Research University

{faridk, sahayr, aliy}@cs.uni-freiburg.de

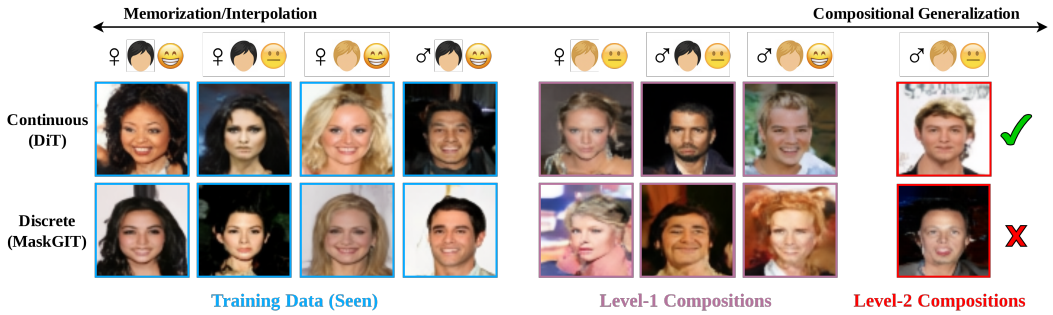



Figure 1: **Compositional Generalization Analysis.** We compare discrete masking (MaskGIT) and continuous diffusion (DiT) on novel compositions of three binary CelebA factors: gender, hair color, and smile. Models are trained on four observed combinations (blue) and evaluated on held-out Level-1 (pink, one-factor change) and Level-2 (red, two-factor change) compositions. DiT (row 1) generalizes compositionally, while MaskGIT (row 2) struggles.

## Abstract

Compositional generalization, the ability to generate novel combinations of known concepts, is a key ingredient for visual generative models. Yet, not all mechanisms that enable or inhibit it are fully understood. In this work, we conduct a systematic study of which design choices critically determine compositional generalization in image and video generation. By isolating independent design axes, we identify two key factors strongly associated with compositional success: (i) whether the training objective operates on a discrete or continuous distribution, and (ii) the completeness of conditioning information about constituent factors during training. We also show that relaxing the discrete loss with an auxiliary continuous latent objective can partially recover compositional performance in discrete models like MaskGIT. Our findings, corroborated by diverse compositional tasks and preliminary evidence in world models and LLMs, motivate a shift toward continuous objectives for compositional generalization.

[*]Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

# 1 Introduction

Visual generative models, such as diffusion models [33, 42, 68] and generative transformers [59, 32, 27], can generate high-fidelity images and videos [58, 26, 29], especially when trained on large amounts of data. However, are these models capable of robust *compositional generalization* or are they merely "interpolating their training data?" That is, can they systematically decompose the observed data into its underlying factors or "concepts" (e.g., objects, attributes) and synthesize novel combinations not seen during training [72, 63, 18]?

Despite significant progress in generative modeling, compositional generalization in current models remains inconsistent. While some studies report successes on targeted compositional tasks [65, 20, 50], others reveal notable limitations, particularly when trying to generate complex scenes or novel combinations of known elements [1, 31]. The factors that drive (or hamper) compositional generalization are not yet clearly identified. Hence the field operates inefficiently. For example, significant resources are currently partitioned between continuous diffusion [22, 7] and discrete masked modeling [38, 60, 10] families, a paradigm that has been bolstered by the scaling successes of discrete Large Language Models (LLMs). Yet, which representation generalizes "better" out of distribution (OOD) remains open.

Figure 1 illustrates a striking difference between visual generative model families. Although both models are trained on a subset of compositions (i.e., "non-smiling, women with blonde hair" and "smiling men with black hair"), a *continuous* DiT [48] generates a well-looking "non-smiling man with blonde hair", whereas a *discrete* MaskGIT [13] produces a severely distorted image. This brings us to our core research question:

*What factors drive compositional generalization in visual generative models?*

To answer this question, we develop a systematic framework for isolating the effects of individual design choices. We categorize modern visual generative models along three primary axes: (i) the *Tokenizer*, which defines the underlying representation space; (ii) the *Generative model*, which generates samples in this space guided by the conditioning signal (task), and (iii) the *Conditioning signal*, which specifies the factors to be generated. Note that the last is also related to scale/diversity of the training data, as large, diverse data can help identify the factors when they are not specified. To anchor our study, we select the simplest, most canonical representatives of each model family–MaskGIT (discrete/masked) and DiT (continuous/diffusion)–to establish a controlled baseline. We then systematically "interpolate" between them by varying individual design axes (e.g., tokenizer type, masking strategy, training objective) while holding others fixed. Thereby, we isolate the factors that affect their compositional generalization capability, independent of engineering refinements that could be applied to any model in any family. Adopting these architectural minimal pairs allows us to compare across multiple variants in different families without introducing additional confounders.

Concretely, we investigate the following research questions:

- **RQ1**: *Does the type of the tokenizer affect compositional generalization?* (VAE with KL regularization vs. VQ-VAE with quantization/commitment regularization.)
- **RQ2**: *How does the generative model design affect compositionality?* In particular, does it matter whether the modeled distribution is continuous or discrete (e.g., continuous latents vs. discrete tokens)? Is a denoising objective essential, or does a masking loss suffice?
- **RQ3**: *Can generative models, without being conditioned on the full data-generating factors during training, generalize compositionally?* Must conditioning have the exact factors, or can they rely on quantized or missing abstractions (e.g., "red" instead of the RGB value, or hiding "smile" in {"blond", "smile", "girl"})?
- **RQ4**: Guided by our findings, *can we intervene on non-compositional models to make them compositional?*

To shed light on these questions, we conduct controlled experiments across a range of architectural and training design choices. The results consistently indicate that models trained to learn a *continuous distribution* by their training objective exhibit stronger compositional capabilities than models trained to model a categorical distribution. Possibly less surprising, we also find that *providing full conditioning information of the generating factors during training* is important; quantized or partial

conditioning leads to weaker compositional generalization. On the other hand, training the tokenizer with a quantization bottleneck has no significant effect on compositional generalization.

Guided by our findings and to further examine why discrete modeling hinders compositional generalization, we test whether augmenting the categorical training objective with a continuous Joint Embedding Predictive Architecture (JEPA) objective [35] can improve compositional performance. This modification introduces continuous latent targets and yields clear improvements on held-out factor combinations. At the same time, models trained with this auxiliary objective exhibit more disentangled intermediate representations, suggesting that predictive continuous objectives can shape the internal structure to support compositionality in discrete models.

**Summary of contributions**: Through controlled experiments on transformer-based visual generative models, we show that (i) models trained with continuous output objectives exhibit substantially stronger compositional generalization than models trained with discrete categorical objectives across the design choices we study; (ii) complete and precise conditioning on generative factors substantially improves compositional generalization; and (iii) augmenting discrete models with a continuous auxiliary objective partially recovers compositionality.

## 2 Related Works

Factorization and compositional generalization have been extensively studied across a range of architectures [66, 36, 69], as they are central to building models that can capture and recombine the underlying factors of variation in data. Prior work has approached this challenge from multiple angles. On the theoretical side, several studies seek first-principles characterizations of compositionality based on the data-generating process [63] or provide provable guarantees through structural constraints such as object-centric autoencoders [64]. Our work instead takes an empirical perspective, systematically probing how architectural and training choices shape compositional generalization in visual generative models.

Complementary to these theoretical accounts, recent empirical investigations highlight how large-scale models such as CLIP [49] exhibit degrees of compositionality, often tied to the properties of their training data [30, 65]. Our focus in this work is not on data, but on architectural, training, and conditioning choices that enhance compositionality.

For generative models, Okawa et al. [43] studied compositional generalization solely for diffusion models. We extend this work along two dimensions: (i) by considering a broader set of generative frameworks, including both continuous and discrete models, and (ii) by examining additional data modalities such as video. Our work also connects to the rich literature on disentangled representation learning [25, 12], which aims to align individual latents with interpretable factors of variation. While disentanglement work focuses on representation quality, our study emphasizes the downstream generative consequences: we analyze how different training objectives and conditioning strategies enable, or hinder, the synthesis of novel compositions.

## 3 Experimental Setup

Our goal is to identify which architectural and training choices enable or hinder compositional generalization (as seen in Figure 1). To this end, we design controlled comparisons that systematically vary key factors-representation space, training objective, and conditioning information levels-while holding others fixed. This allows us to "interpolate" between the canonical model pairs from different families (such as DiT and MaskGIT), and isolate the contributions of individual design decisions.

**"Interpolation" between the minimal architectural pairs** The main differences between DiT and MaskGIT are in the following design choices: the tokenizer (VAE/VQ-VAE for DiT vs. strictly VQ-VAE for MaskGIT), the training objective (masking-based for MaskGIT, absent in DiT), the loss function (diffusion in DiT vs. categorical negative log-likelihood (NLL) in MaskGIT), and the nature of the latent output distribution (continuous for DiT vs. categorical discrete for MaskGIT). To isolate which choices affect compositional capability, we systematically vary them one at a time.

Recent works have effectively implemented several such interpolations; see Table 1. For example, MAR [37] differs from DiT only in adopting a masking-based prediction while retaining the per-token diffusion objective. Similarly, GIVT [54] replaces MaskGIT's standard categorical NLL with a Gaussian mixture negative log-likelihood (GMM-NLL), where the token distribution is modeled as a mixture of Gaussians [51] (which allows GIVT to operate in a continuous space without employing the denoising diffusion objective). We adopt these controlled variants for our experiments while neutralizing potential confounders. We use *continuous objective* for objectives over continuous outputs, in contrast to categorical prediction over discrete token-ids, which lack a metric structure. Since each objective is only defined over its own prediction (and thus input) space, we treat them as one design axis.

**Testing compositional generalization** We train models on pairs of images conditioned on tuples specifying the underlying factors (e.g., object color). We train only on a subset of the possible factor combinations and evaluate compositional generalization on held-out combinations. We distinguish between *level-1* compositions, which differ from the nearest combination seen during training by one factor, and *level-2* compositions, which differ by two factors. While our primary analysis focuses on simple synthetic images with three binary factors (**Shapes2D**) following Okawa et al. [43], we benchmark the robustness of our findings across diverse domains. This includes multi-valued factors (**Shapes3D**), real-world faces (**CelebA**), and synthetic videos (**CLEVRER-Kubric**). Furthermore, we also show that our findings transfer to complex real-world scenes (**CoVLA**) using the Orbis model, and provide preliminary evidence on a Points24-inspired language reasoning task. Detailed dataset descriptions are in Appendix B.

Table 1: Summary of generative models.

| Model | Masking-based | Training loss | Latent Distribution |
|---|---|---|---|
| MaskGIT | ✓ | Categorical NLL | discrete |
| GIVT | ✓ | GMM-NLL | continuous |
| MAR | ✓ | Diffusion | continuous |
| DiT | ✗ | Diffusion | continuous |

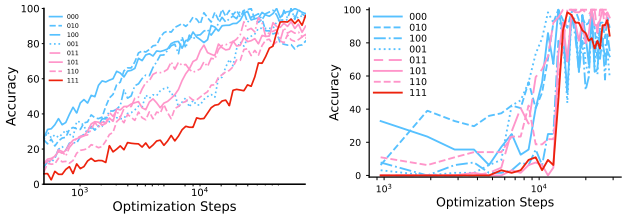
**Evaluation** To evaluate whether generated images faithfully capture the intended factors, we train probes for each factor to classify its presence in the generated outputs. We train the probes on *all* possible combinations, including held-outs from model training. Following Okawa et al. [43], Park et al. [46], a composition is considered correct if all its factors' probe probability is the highest (argmax); we verify that probe decisions are rarely borderline and track visual quality in Appendix F.1. While we additionally report standard metrics (FID, FDD, CRA) on real datasets in Appendix F, probe accuracy directly measures whether the intended factors are present, rather than overall image quality only.

For visualizing training dynamics, we use blue curves for compositions seen in training, pink curves for level-1 compositions, and red curves for level-2 compositions. Performance on level-2 compositions serves as a particularly informative metric for compositional generalization, as they require the model to generate multiple novel factors simultaneously.

## 4 Design choices that drive compositionality

In this section, we leverage the setup in Section 3 to systematically rule out irrelevant factors in our architectural design space by interpolating from DiT to MaskGIT. This process isolates the design elements that are essential for enabling compositional generalization in DiT but not MaskGIT. We focus on results from the Shapes2D dataset in the following. However, our findings also extend to more complex, real-world, and video datasets (quantitative and qualitative results in Appendix G). In addition, Appendix E discusses existing theoretical results on compositional generalization that offer perspectives consistent with our findings.

**Does the choice of tokenizer matter?** The tokenizer is one of the main differences between DiT and MaskGIT (Table 1). To isolate and assess its effects on compositionality (**RQ1**), we evaluate the same second-stage architecture (DiT) with two different tokenizers: a discrete tokenizer (VQ-VAE) and a continuous tokenizer (VAE). Additional implementation details are provided in Appendix C.1.




(a) VAE (b) VQ-VAE

Figure 2: **DiT exhibits compositional generalization regardless of the type of tokenizer used.** While the training dynamics differ, DiT shows compositional generalization at the end of training. The blue, pink, and red curves show linear probe accuracies for the training data, level-1 compositions, or level-2 compositions, respectively. Results are consistent for MAR (Figure 7) and video (Figure 8).

Figure 2 shows that DiT achieves comparable compositional performance with both tokenizer types by the end of training, though training dynamics differ: continuous tokenizers yield gradual progress while discrete tokenizers show more abrupt emergence. Further, we find the discrete tokenizer to be more sensitive to learning rates.[1] Hence, tokenizer regularization does not erode compositionality and is unlikely to explain the DiT–MaskGIT gap.

> **Finding 1:** The choice of tokenizer does not substantially alter the compositional abilities of a generative model. It primarily affects training efficiency and stability.

Since tokenizer choice does not affect compositionality and MaskGIT requires discrete tokens, we fix the tokenizer to VQ-VAE to avoid *VQ* vs. *KL* regularization as a potential confounder. Discrete models use codebook indices; continuous models use pre-quantization latents (details in Appendix C.1).

**Does the masking matter?** Another main difference between DiT and MaskGIT is their objective: MaskGIT uses a masking-based loss[2], whereas DiT relies on the standard denoising approach (Table 1). MAR can be viewed as a hybrid or "interpolation" between these two: it combines a masking-based objective with a diffusion loss per token.

Figure 3c shows that MAR achieves robust compositional generalization by the end of training. This uncouples the masking in the training objective from the compositionality failure modes of MaskGIT and does not critically impact the model's ability to generalize compositionally; neither does MaskGIT's confidence-based unmasking order (Appendix C.4).

**Is it the diffusion loss?** We look at another major difference between DiT and MAR vs MaskGIT: their training losses. GIVT [54] provides a bridge between MAR and MaskGIT on the objective axis by replacing MaskGIT's categorical head with a continuous Gaussian Mixture Model (GMM). This intervention effectively substitutes the diffusion loss of MAR/DiT with maximum likelihood training. This is done by using the continuous parameters of a GMM [51] while preserving the continuous output space. Figure 3a $\rightarrow$ 3b shows that the MaskGIT $\rightarrow$ GIVT intervention (categorical $\rightarrow$ continuous) shows a substantial jump in compositional performance, whereas GIVT $\rightarrow$ MAR (GMM $\rightarrow$ diffusion) brings comparatively smaller gains as seen in Figure 3b $\rightarrow$3c. This identifies a primary driver: the continuous training objective on a continuous output space, rather than the exact form of the objective (denoising vs. parameter prediction), is a key factor for enabling compositional generalization along the axes we varied.

**Output-space continuity emerges as the primary factor** By systematically isolating the architectural axes separating the two main classes of visual generative models, including the tokenizer, masking strategies, and loss functions, we have ruled out these factors as primary drivers of compositional generalization. The remaining distinguishing factor is the nature of the predicted outputs: DiT, MAR, and GIVT predict continuous-valued quantities, while MaskGIT predicts discrete tokens. We conclude that this difference in output representation (and thereby the objective) is a primary

[1] A detailed analysis of these differences is left for future work.

[2] "Masking-based": predicting a subset of tokens from the rest, via continuous or discrete masks.

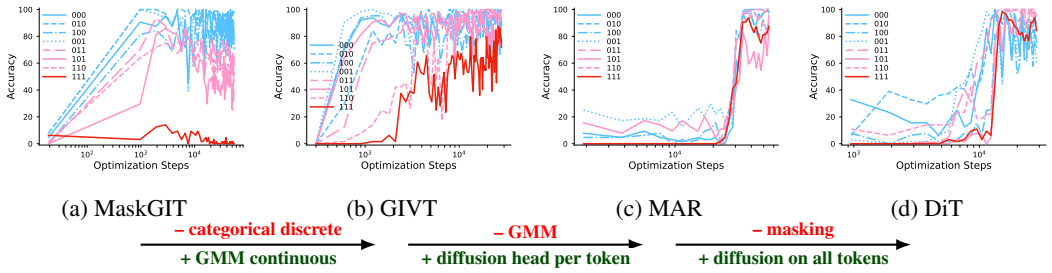
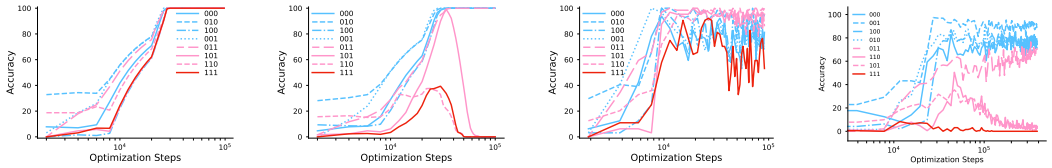



Figure 3: **Compositional generalization performance on Shapes2D across different model architectures.** Models that learn continuous distributions (DiT, MAR, and GIVT) consistently show better level-2 compositions than MaskGIT, with the decisive shift in performance occurring at the categorical-to-continuous intervention. The blue, pink, and red curves denote training, level-1, and level-2 compositions, respectively. Consistent results are observed for CLEVRER-Kubric (Figure 9).




(a) Full-information conditioning

(b) Full-information conditioning + dropout

(c) Quantized conditioning

(d) Quantized conditioning + dropout

Figure 4: **Comparison of conditioning information levels and their impact on compositional generalization in DiT on Shapes2D. (a)** Continuous (full-information) conditioning leads to uniform convergence across all compositions. **(b)** Label dropout conditioning leads to inconsistent generalization; several unseen compositions fail. **(c)** Discrete (quantized) conditioning leads to partial generalization, with some failing samples. **(d)** Discrete (quantized) conditioning with dropout, the most severe loss of information, leads to the most failure. Shaded areas indicate standard deviation across three different seeds. We provide additional results in Appendix C.5. The blue curves show performance on training data, pink curves depict level-1 compositions, and red curve denotes level-2.

factor behind the compositional generalization of continuous models, which discrete models such as MaskGIT lack (also under causal autoregressive decoding; Appendix C.3).

**Finding 2:** Generative models trained to model continuous distributions consistently exhibit stronger compositionality, compared to models trained on discrete, categorical distributions.

## 5 Importance of the conditioning information levels

We also investigate how different forms of conditioning affect compositional generalization (**RQ3**). This question is related to previous findings in literature that scale — or rather diverse data — foster compositional generalization. In our previous experiments, we conditioned models on complete, continuous factor representations (e.g., raw hue or size). In more realistic scenarios, however, factors are often quantized (e.g., the word "red" instead of the exact hue) or provided as lossy description (i.e., some factors are missing in the conditioning signal during training). The latter is particularly problematic, since the factors must then be inferred from the images (unsupervised identification).

For quantized signals, we convert continuous signals into discrete binary signals. For lossy signals, we randomly drop each factor with a 10% probability, so that, on average, one in ten factors is missing. Figure 4c shows that compositional generalization becomes less stable under quantized conditioning. When some factors present in the image are occasionally missing from the conditioning signal, compositional generalization typically fails (Figure 4b). Combining both conditions–a setting common in practice–produces the strongest negative effect (Figure 4d).

Overall, these results show that limited information–whether through quantization or incomplete conditioning– can impair compositional generalization, even when all factors are provided during

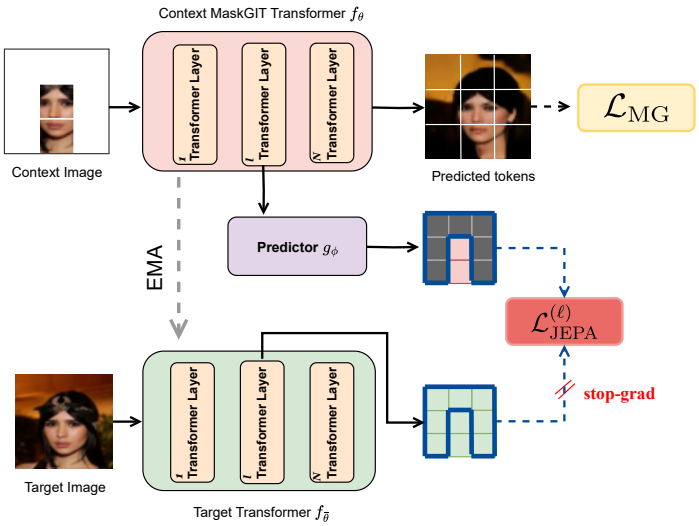



Figure 5: **An overview of MaskGIT combined with the JEPA-based training objective.** We apply the JEPA loss at specific layers ($l$) on an intermediate masked token representation in the transformer ($H_C^{(l)}$) and train a lightweight predictor to reconstruct target states ($H_T^{(l)}$) using MSE as an error metric and a stop-gradient signal to avoid representation collapse.

inference. We observe the same behavior when factors are specified through text prompts (Appendix C.6). This is in line with findings on the importance of scale/diversity for compositional generalization [55, 30, 50], as only diverse data with overlapping conditional distributions allow unsupervised identification of the relevant factors.

**Finding 3:** Full, precise factor labeling during training is important for robust compositional generalization. Models trained with quantized or incomplete signals (concept dropout) lead to weaker and inconsistent recombination of factors.

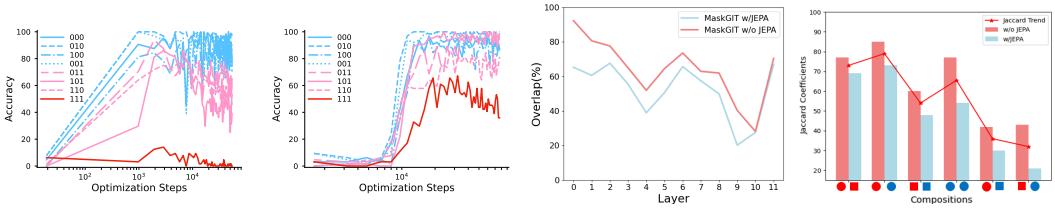
## 6 Enhancing Compositional Learning with Joint Embedding Predictive Architectures

In Section 4, we identified a key factor that can substantially weaken compositional generalization in modern generative models: training objectives that operate over discrete, categorical distributions. Despite this, discrete training objectives–such as the one used in MaskGIT–offer advantages in, e.g., sampling speed compared to alternatives like DiT. This raises the question: can we retain these advantages while also improving compositional generalization?

### 6.1 An Auxiliary Continuous Relaxation Objective for MaskGIT

Building on our previous findings, we extend MaskGIT's training objective with an auxiliary continuous representational loss, similar to Joint Embedding Predictive Architecture (JEPA) [35, 4]. JEPA learns to reconstruct target patch representations by using (other) context patches from the same image. This auxiliary objective resembles MaskGIT's own masking-based formulation but operates directly on continuous latent representations rather than discrete tokens.

Formally, following MaskGIT, we encode the input image or video $x$ into the discrete tokens $z = \{1 \dots K\}^N$ using a VQ-VAE. MaskGIT is then trained to reconstruct a masked subset of tokens $z_{\mathcal{M}}$ from the unmasked tokens $z_{\bar{\mathcal{M}}}$ ($\mathcal{L}_{\text{MG}}$). For images, masking is applied spatially, whereas for videos, it is applied causally in time. Now, we add the auxiliary representation alignment objective based on JEPA ($\mathcal{L}_{\text{JEPA}}$). Given the context latent representations $H_C^{(l)}$ from selected intermediate layers $l$, the model is trained to reconstruct the target latent representations $H_T^{(l)}$. We use Mean




(a) Standard Objective (b) JEPA-based Objective (c) Polysemanticity Trend (d) Jaccard Trend

Figure 6: Comparison of linear probe accuracy for MaskGIT with (a) Standard and (b) JEPA-based training objectives for Shapes2D. The JEPA-based objective clearly enhances compositional abilities even though it cannot fully compensate the problems introduced by the discrete space. We also see lower polysemanticity between attention heads (c) and a decreasing Jaccard trend over shared circuits (d). The blue curves show performance on training data, pink curves depict level-1 compositions, and red curve denotes level-2 compositions. Consistent videos results shown in Figure 13.

Squared Error (MSE) with a stop-gradient on the target representations from an EMA version of the model to stabilize training. In the final objective, we jointly optimize $\mathcal{L}_{\text{MG}}$ and $\mathcal{L}_{\text{JEPA}}$. More information on the objective and results on video tasks can be found in Appendix Appendix D.

While this auxiliary loss resembles the recent REPresentation Alignment (REPA) framework [71], REPA aligns generative latents with an external pretrained encoder. In contrast, the JEPA-based objective is completely self-supervised, structuring the model's own intermediate representations with a continuous loss to support our findings and, in turn, compositional generalization.

### 6.2 Results and Analysis

Extending MaskGIT with the auxiliary objective improves compositional generalization (Figures 1 and 6b), particularly for level-2 compositions. In contrast, without the auxiliary loss, MaskGIT exhibits substantially weaker compositional generalization (Figure 6a). These results reinforce our earlier finding that having an objective operating on continuous outputs (or, by extension, intermediate latent representations) strongly supports compositional generalization. To better understand the effect of our auxiliary loss, we analyze its effect on the learned representations below. We apply mechanistic interpretability techniques [6], which allow us to probe how individual components encode specific factors during generation.

**Polysemanticity in attention heads** An attention head can easily become *polysemantic*, meaning that it "attends to" multiple distinct and unrelated features simultaneously. This can occur due to phenomena like superposition [17]. However, superposition, resulting in polysemanticity, causes undesired entanglement: if an attention head responsible for processing `color` also strongly activates for `shapes`, it becomes difficult for the model to disentangle these factors.

To quantify polysemanticity, we assess the effect of each attention head on the predefined visual factors/features. We pass pairs of images that differ only in a single factor (e.g., a *red* large square vs. *blue* large square) and compute the feature similarity for the global token. We then systematically deactivate attention heads and re-compute the similarity. A head is considered polysemantic if the feature similarity difference exceeds a threshold, suggesting that it attends to multiple, entangled factors that may hinder compositional control. Figure 6c shows that fewer attention heads are polysemantic in MaskGIT with the auxiliary continuous loss suggesting less entangled factors.

**Mechanistic similarity using circuits** The above analysis shows that attention heads mix up some features, but it only provides a coarse view of representational entanglement. To probe deeper into how concepts are internally represented and transformed across layers, we analyze the model's *internal circuitry* [44]-groups of neurons that jointly implement specific mechanisms.

We aim to identify the most influential components that generate each composition. To do so, we focus on MLP neurons within each transformer block, since prior work has shown that these often encode semantic factors [21, 15]. For each composition, we identify the top-$k$ most influential neurons per layer using activation patching [57, 40]. Specifically, we compute each neuron's *indirect effect* [47], which captures its downstream influence on the model's output rather than just its direct

contribution. This is achieved by comparing the output when running the model on a factual input (the correct composition condition, e.g., "red circle") with the output when running on a counterfactual input (the same condition but with a key intermediate activation corrupted, e.g., replaced with noise), and then patching in the neuron's clean activation. Intuitively, this measures how strongly a neuron shapes the generated image or video.

We then compare the overlap of the top-$k$ neurons across different compositions using the Jaccard index, following Kempf et al. [30]. A high overlap implies that the same neuron is used for multiple factors, suggesting that circuits are more entangled. In contrast, a low overlap suggests separate circuits. (More in Appendix D.5). Figure 6d shows that adding the auxiliary JEPA objective consistently reduces neuron overlap across layers, particularly for composition pairs where both factors differ (e.g., `red circle-blue square`), suggesting more factor-specific circuits.

**Finding 4:** A JEPA-based training objective induces more disentangled and semantically structured representations, and improves compositional generalization.

## 7 Discussion and Limitations

**Do our results extend to world models?** Our previous experiments focused on academic datasets to enable controlled experimentation. To test whether the observed trend extends to a more realistic setting, we curate compositional splits on CoVLA [2], a driving scenes video dataset. The factors are *time of day* (day $\odot$/night $\bigcirc$) and *turn direction* (left $\leftarrow$/straight $\uparrow$/right $\rightarrow$). We hold out $\odot\rightarrow$ and $\bigcirc\leftarrow$. As our generative video model, we adopt state-of-the-art Orbis [41] conditioned only via the previous frames. Since Orbis does not know the explicit factors, it must thus infer them via unsupervised identification. We train the two Orbis variants, Orbis-DiT (continuous) and Orbis-MaskGIT (discrete), with identical training and inference budgets. We measure compositionality as the fraction of nearest neighbors sharing the conditioning factors (Appendix H).

Table 12 shows that DiT outperforms MaskGIT on *novel* compositions, with relative gains of **+161%** on $\odot\rightarrow$ and **+207%** on $\bigcirc\leftarrow$. Per-split nearest-neighbor ratios further suggest that MaskGIT collapses to seen combinations, while DiT assigns more mass to the intended novel target (see Figure 20). Although absolute accuracies remain modest—reflecting the difficulty of real-world video with implicit contextual factors—the trend matches our controlled interventions in Finding 2: *continuous objectives improve compositionality under complete conditioning relative to categorical objectives*.

**Do our results extend to language?** Although our study focuses on *visual* generative models, compositional generalization has been central in language modeling [19, 52]. Does the benefit of continuous objectives transfer to language?

We study compositional generalization for Llama-3.2 [16] on a controlled card-arithmetic task inspired by the Points24 dataset [14]. In this task, the model is given four cards and a target value, and must construct an arithmetic expression that uses each card once to reach the specified target number. Training involves single rules (restricting the target value or doubling the value of red-suit cards), while the (compositional) test split requires composing both rules simultaneously (details in Appendix I). We compare two reasoning mechanisms: standard Chain-of-Thought (CoT) [61] and its continuous variant, COCONUT [24]. Here, reasoning mechanisms play a role analogous to training objectives in our experiments: whether intermediate tokens are treated as discrete symbolic traces (CoT) or continuous thoughts (COCONUT). COCONUT outperforms CoT on the compositional splits (12.33% vs. 4.87%), suggesting that the advantage of continuous objectives may carry over to language.

**Limitations** In comparison to previous works on diffusion models generalization, our work aimed to move beyond toy data while retaining controlled compositional factors. This remains a fundamental limitation for studying web-scale models, where the relevant factors are unknown or entangled.

While our evidence shows that continuous objectives matter for compositional generalization, we can only speculate about the mechanism. We hypothesize that more continuous loss gradients make it easier to establish compositional factors as separate components in the representation, whereas gradients that strongly favor correct categorical decisions can lose them between the output, input

and intermediate representations, even when they are given explicitly. Current mechanistic analysis tools are not yet crisp enough to verify this hypothesis.

Our findings may also have implications for LLMs, which are typically trained with categorical objectives. We provide preliminary evidence of similar effects in language models, but controlled evaluation of compositional generalization in language is non-trivial and requires dedicated study.

## Acknowledgments and Disclosure of Funding

This work was funded by the German Federal Ministry for Economic Affairs and Energy within the project "NXT GEN AI METHODS" (19A23014R). The compute used in this project was also funded by the German Research Foundation (DFG) under grants 417962828 and 539134284. Karim Farid and Rajat Sahay are part of the European Lab for Learning and Intelligent Systems (ELLIS) PhD program. This work was funded in part by the French government under management of Agence Nationale de la Recherche as part of the "France 2030" program, reference ANR-23-IACL-0008(PR[AI]RIE-PSAI project) and the ANR project VideoPredict ANR-21- FAI1-0002- 01. Cordelia Schmid would like to acknowledge the support by the Körber European Science Prize. The authors acknowledge support from the state of Baden-Württemberg through bwHPC. The IPCEI ME/CT project is supported by the Federal Ministry for Economic Affairs and Climate Action on the basis of a decision by the German Parliament, by the Ministry for Economic Affairs, Labor and Tourism of Baden-Württemberg based on a decision of the State Parliament of Baden-Württemberg, the Free State of Saxony on the basis of the budget adopted by the Saxon State Parliament, the Bavarian State Ministry for Economic Affairs, Regional Development and Energy and financed by the European Union – NextGenerationEU. We thank Arian Mousakhan, Silvio Galesso, Sudhanshu Mittal, Johannes Dienert, and Jelena Bratulic for valuable discussions and feedback during the development of this work.

## References

[1] Shengnan An, Zeqi Lin, Qiang Fu, Bei Chen, Nanning Zheng, Jian-Guang Lou, and Dongmei Zhang. How do in-context examples affect compositional generalization? *arXiv preprint arXiv:2305.04835*, 2023.

[2] Hidehisa Arai, Keita Miwa, Kento Sasaki, Kohei Watanabe, Yu Yamaguchi, Shunsuke Aoki, and Issei Yamamoto. Covla: Comprehensive vision-language-action dataset for autonomous driving. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1933–1943. IEEE, 2025.

[3] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[4] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

[5] Vitória Barin-Pacela, Kartik Ahuja, Simon Lacoste-Julien, and Pascal Vincent. On the identifiability of quantized factors. In *Causal Learning and Reasoning*, pages 384–422. PMLR, 2024.

[6] Leonard Bereska and Stratis Gavves. Mechanistic interpretability for AI safety - a review. *Transactions on Machine Learning Research*, 2024.

[7] BFL Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

[8] Arwen Bradley. Local mechanisms of compositional generalization in conditional diffusion. *arXiv preprint arXiv:2509.16447*, 2025.

[9] Jack Brady, Julius von Kügelgen, Sébastien Lachapelle, Simon Buchholz, Thomas Kipf, and Wieland Brendel. Interaction asymmetry: A general principle for learning composable abstractions. *arXiv preprint arXiv:2411.07784*, 2024.

[10] Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning*, pages 4603–4623, 2024.

[11] Chris Burgess and Hyunjik Kim. 3d shapes dataset. https://github.com/deepmind/3dshapes-dataset/, 2018.

[12] Hugo Caselles-Dupré, Michael Garcia-Ortiz, and David Filliat. Symmetry-based disentangled representation learning requires interaction with environments, 2019. URL `https://arxiv.org/abs/1904.00243`.

[13] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11315–11325, 2022.

[14] Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training, 2025. URL `https://arxiv.org/abs/2501.17161`.

[15] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

[16] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.

[17] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[18] Alessandro Favero, Antonio Sclocchi, Francesco Cagnetta, Pascal Frossard, and Matthieu Wyart. How compositional generalization and creativity improve as diffusion models are trained. *arXiv preprint arXiv:2502.12089*, 2025.

[19] Hiroki Furuta, Yutaka Matsuo, Aleksandra Faust, and Izzeddin Gur. Language model agents suffer from compositional generalization in web automation. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.

[20] Sachit Gaudi, Gautam Sreekumar, and Vishnu Boddeti. Coind: Enabling logical compositions in diffusion models, 2025. URL `https://arxiv.org/abs/2503.01145`.

[21] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

[22] Google DeepMind. Veo 3: A high-fidelity video generation model, 2025. URL `https://deepmind.google/technologies/veo/`. Accessed: 2026-01-28.

[23] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3749–3761, 2022.

[24] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2024. URL `https://arxiv.org/abs/2412.06769`.

[25] Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations, 2018. URL `https://arxiv.org/abs/1812.02230`.

[26] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

[27] Drew A Hudson and Larry Zitnick. Generative adversarial transformers. In *International conference on machine learning*, pages 4487–4499. PMLR, 2021.

[28] Mason Kamb and Surya Ganguli. An analytic theory of creativity in convolutional diffusion models. *arXiv preprint arXiv:2412.20292*, 2024.

[29] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2020. URL `https://arxiv.org/abs/1912.04958`.

[30] Elias Kempf, Simon Schrodi, Max Argus, and Thomas Brox. When and how does clip enable domain and compositional generalization? In *PMLR*, 2025.

[31] Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. Measuring compositional generalization: A comprehensive method on realistic data. *arXiv preprint arXiv:1912.09713*, 2019.

[32] Sungwoong Kim, Daejin Jo, Donghoon Lee, and Jongmin Kim. Magvlt: Masked generative vision-and-language transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 23338–23348, 2023.

[33] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

[34] Sébastien Lachapelle, Divyat Mahajan, Ioannis Mitliagkas, and Simon Lacoste-Julien. Additive decoders for latent variables identification and cartesian-product extrapolation. *Advances in Neural Information Processing Systems*, 36:25112–25150, 2023.

[35] Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

[36] Chuanhao Li, Chenchen Jing, Zhen Li, Mingliang Zhai, Yuwei Wu, and Yunde Jia. In-context compositional generalization for large vision-language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17954–17966, 2024.

[37] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37:56424–56445, 2024.

[38] Jinlai Liu, Jian Han, Bin Yan, Fengda Zhu, Xing Wang, Yi Jiang, BINGYUE PENG, Zehuan Yuan, et al. Infinitystar: Unified spacetime autoregressive modeling for visual generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[39] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15(2018):11, 2018.

[40] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2022. URL `https://arxiv.org/abs/2202.05262`.

[41] Arian Mousakhan, Sudhanshu Mittal, Silvio Galesso, Karim Farid, and Thomas Brox. Orbis: Overcoming challenges of long-horizon prediction in driving world models. *arXiv preprint arXiv:2507.13162*, 2025.

[42] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[43] Maya Okawa, Ekdeep Singh Lubana, Robert P. Dick, and Hidenori Tanaka. Compositional abilities emerge multiplicatively: Exploring diffusion models on a synthetic task, 2024. URL `https://arxiv.org/abs/2310.09336`.

[44] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

[45] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL `https://arxiv.org/abs/2304.07193`.

[46] Core Francisco Park, Maya Okawa, Andrew Lee, Ekdeep S Lubana, and Hidenori Tanaka. Emergence of hidden capabilities: Exploring learning dynamics in concept space. *Advances in Neural Information Processing Systems*, 37:84698–84729, 2024.

[47] Judea Pearl. Direct and indirect effects, 2013. URL `https://arxiv.org/abs/1301.2300`.

[48] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[49] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL `https://arxiv.org/abs/2103.00020`.

[50] Florian Redhardt, Yassir Akram, and Simon Schug. Scaling can lead to compositional generalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[51] Douglas A Reynolds et al. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663): 3, 2009.

[52] Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. Revisiting compositional generalization capability of large language models considering instruction following ability, 2025. URL `https://arxiv.org/abs/2506.15629`.

[53] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19 (70):1–57, 2018.

[54] Michael Tschannen, Cian Eastwood, and Fabian Mentzer. Givt: Generative infinite-vocabulary transformers. In *European Conference on Computer Vision*, pages 292–309. Springer, 2024.

[55] Arnas Uselis, Andrea Dittadi, and Seong Joon Oh. Does data scaling lead to visual compositional generalization? In *PMLR*, 2025.

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[57] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401, 2020.

[58] Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual description, 2022. URL `https://arxiv.org/abs/2210.02399`.

[59] Jianfeng Wang, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu, and Lijuan Wang. Git: A generative image-to-text transformer for vision and language. *arXiv preprint arXiv:2205.14100*, 2022.

[60] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *CoRR*, 2024.

[61] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

[62] Thaddäus Wiedemer, Prasanna Mayilvahanan, Matthias Bethge, and Wieland Brendel. Compositional generalization from first principles. *Advances in Neural Information Processing Systems*, 36:6941–6960, 2023.

[63] Thaddäus Wiedemer, Prasanna Mayilvahanan, Matthias Bethge, and Wieland Brendel. Compositional generalization from first principles, 2023. URL https://arxiv.org/abs/2307.05596.

[64] Thaddäus Wiedemer, Jack Brady, Alexander Panfilov, Attila Juhos, Matthias Bethge, and Wieland Brendel. Provable compositional generalization for object-centric learning, 2024. URL https://arxiv.org/abs/2310.05327.

[65] Thaddäus Wiedemer, Yash Sharma, Ameya Prabhu, Matthias Bethge, and Wieland Brendel. Pretraining frequency predicts compositional generalization of clip on real-world tasks, 2025. URL https://arxiv.org/abs/2502.18326.

[66] Mike Wu and Noah Goodman. Multimodal generative models for compositional representation learning, 2019. URL https://arxiv.org/abs/1912.05075.

[67] Keyulu Xu, Mozhi Zhang, Jingling Li, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. *arXiv preprint arXiv:2009.11848*, 2020.

[68] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

[69] Yongyi Yang, Core Francisco Park, Ekdeep Singh Lubana, Maya Okawa, Wei Hu, and Hidenori Tanaka. Dynamics of concept learning and compositional generalization. *arXiv preprint arXiv:2410.08309*, 2024.

[70] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. Clevrer: Collision events for video representation and reasoning, 2020. URL https://arxiv.org/abs/1910.01442.

[71] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think, 2025. URL https://arxiv.org/abs/2410.06940.

[72] Linfeng Zhao, Lingzhi Kong, Robin Walters, and Lawson LS Wong. Toward compositional generalization in object-oriented world modeling. In *International Conference on Machine Learning*, pages 26841–26864. PMLR, 2022.

# A Models

We selected a representative set of generative models to span key design axes relevant to compositional generalization. **Diffusion Transformers (DiTs)** [48] operate in a continuous latent space derived from a VAE and are trained with a standard denoising objective; conditioning is applied via adaptive layer normalization. In contrast, **MaskGIT** [13] works in a discrete latent space obtained from a VQ-VAE and employs a bidirectional transformer to iteratively predict masked tokens, corresponding to a categorical, masking-based objective. **MAR** [37] also uses a continuous latent space but masks subsets of latent tokens and leverages an encoder-decoder to generate intermediate conditions for a diffusion head that denoises each masked token independently, combining aspects of masking and denoising. Together, these models cover continuous versus discrete representations, denoising versus masking objectives, and a variety of conditioning strategies.

## A.1 Diffusion Transformer

To evaluate models in continuous representation spaces, we train **Diffusion Transformers** (DiTs) [48] which adapt the Transformer architecture [56] within the Latent Diffusion Model framework [42]. Initially, a pretrained VAE encodes an input image $x$ into a compressed latent representation $z = E(x)$ which is processed into patch embeddings $s$. A standard forward diffusion process then progressively adds Gaussian noise to this latent representation $s$ over $t$ timesteps. The core of the model is a Transformer network trained to denoise the noisy latent $s_t$ by predicting the added noise $\epsilon_t$ at each timestep $t$, typically minimizing $L = \mathbb{E}_{s_0,\epsilon,t,c}\left[\|\epsilon - \epsilon_\theta(s_t, t, c)\|_2^2\right]$, where $\mathbb{E}$ denotes the expectation over the initial latent representation, $\epsilon_\theta$ is the noise predicted by the Transformer network with parameters $\theta$, and $\| \circ \|_2^2$ denotes the squared L2 norm (Euclidean norm). We implement conditional generation by integrating guidance information $c$ into the Transformer blocks using adaptive layer normalization (adaLN), which modulates network activations based on $c$ and $t$.

## A.2 MAR

**MAR** [37] explores generation within a continuous latent space defined by a VAE while preserving autoregressive principles- iteratively generating a set of tokens conditioned on previously generated tokens. It consists of an encoder, a decoder, and a diffusion head, with both encoder and decoder consisting of a stack of self-attention blocks. Specifically, MAR employs a pretrained VAE to encode an image into a latent representation $\mathcal{Z} \in \mathbb{R}^{h \times w \times d}$ while randomly masking a subset of image tokens within $\mathcal{Z}$. Let $\mathcal{U} = \{z_{i_{\setminus M}}\}_{i=1}^{N_{\setminus M}}$ and $\mathcal{M} = \{z_i\}_{i=1}^{N}$ represent the set of unmasked and masked tokens, respectively, where each $z_{\setminus M}$ and $z$ corresponds to a spatial element in $Z$. Here, $N_{\setminus M}$ denotes the number of unmasked tokens, while $N$ represents the number of masked tokens. The encoder processes $\mathcal{U}$ to extract latent features. The decoder then takes as input both these latent features and a set of learnable tokens of size $N$, each corresponding to a masked token in $\mathcal{M}$. It subsequently generates the features of these learnable tokens, referred to as conditions. Finally, the diffusion head independently performs a denoising process on each masked token $z_i$ using its corresponding condition.

## A.3 MaskGIT

**MaskGIT** or Masked Generative Image Transformer [13] is an autoregressive model operating in a discrete latent space. An input data sample $x$ is first encoded into a sequence of discrete visual tokens $E_{VQ}(x) \in \{1, ..., K\}^N$ where $E_{VQ}$ is the encoder trained with a VQ-VAE objective and $N$ is the sequence length. The standard MaskGIT objective ($\mathcal{L}_{MG}$) focuses on reconstructing masked input tokens. We employ block-masking [4] for images and causal block masking for videos [4] to sample a mask $\mathcal{M} \subset \{1, ..., N\}$. The MaskGIT Transformer $T$ predicts the probability distribution $p(z_i|z_{\setminus M})$ for each masked token $z_i$ ($i \in M$) based on unmasked token $z_{\setminus M}$. The loss is the negative log-likelihood of the true tokens at the masked positions $\mathcal{L}_{MG} = -\mathbb{E}_{x,M}\left[\sum_{i \in M} \log p(z_i|z_{\setminus M})\right]$.

MaskGIT then learns a bidirectional transformer model $p_\theta(z|c)$ trained via the masking procedure, predicting masked tokens based on unmasked one and conditioning $c$. Generation is performed iteratively, starting with all tokens masked and progressively predicting and committing to high-confidence tokens. Conditioning $c$ is embedded to the token sequence using a two-layer MLP and integrated using adaptive layer normalization (adaLN).

# B Datasets

Below, we provide an overview of the general characteristics and qualitative examples of the three datasets used in our paper. Results for generation from the generative models can be found in Appendix G.

## B.1 Shapes2D

Shapes2D is a synthetic dataset introduced by Okawa et al. [43] to systematically study compositional generalization in diffusion models. Each image is composed of a single geometric object characterized by three binary-valued concepts: *shape*, *color*, and *size*, with the composition space defined as `shape={circle, triangle}, color={red, blue}, size={large, small}`. Unless stated otherwise, these concepts are encoded as binary tuples—e.g., the tuple `000` represents a large, red circle. This structured format allows the use of concept tuples as conditioning signals during diffusion model training, mimicking abstract textual prompts. The dataset is constructed by uniformly sampling over the concept combinations, yielding 1500 images per training concept class and a total of 6000 images. Although the concept variables are nominally discrete, the *size* and *color* attributes are realized through a spectrum of sizes within the big and small labels and hues within the red and blue labels, resulting in tightly clustered yet visually diverse instantiations. These clusters are sparsely distributed and exhibit limited continuity in the pixel space, effectively forming "islands" within the data manifold. Such fragmentation introduces discontinuities in the data-generating factors space that further complicate the model's ability to learn smooth conceptual compositional interpolations.

## B.2 Shapes3D

Shapes3D [11] is a synthetic dataset of rendered 3D scenes designed to study disentanglement and compositional generalization. Each image depicts a single object in a colored room, generated by varying six independent factors: floor color (10 values), wall color (10), object color (10), object shape (4), object size (8), and camera azimuth (15). The full dataset contains 480,000 images, corresponding to all possible combinations of these factors. This makes it a significantly more complex data space than Shapes2D and its fully factorial structure makes Shapes3D well-suited for systematic generalization experiments.

For our experiments, we use all 10 object colors, three of the four shapes (cube, cylinder, sphere), and all eight size values (0–7), yielding 240 compositions.

The model is conditioned on the specific colour, shape, and size values. For compositional evaluation, we group the 240 compositions into 8 supergroups to structure the train–test split and to order results by degree of novelty. As in Shapes2D, we define three levels of novelty: first level includes compositions seen during training. second and third levels are used for testing and differ from the training set in one and two factors, respectively, with level three representing the highest degree of novelty.

We partition the factors of color, shape, and size into two supergroups based on predefined thresholds. Specifically, for the color factor, which consists of ten categories, Supergroup 0 includes the first seven colors (red, orange, yellow, lime green, green, cyan, blue), and Supergroup 1 includes the remaining three (indigo, purple, pink). For shape, which contains three categories, Supergroup 0 includes cube and cylinder, and Supergroup 1 includes sphere. Finally, for size, which is defined over ordinal values from 0 to 7, values 0 through 5 belong to Supergroup 0, while values 6 and 7 belong to Supergroup 1.

## B.3 CelebA

To assess the robustness of our findings under real-world settings, we extend our evaluation to the CelebA dataset. Following prior work, we select three visually distinguishable attributes as concept variables: *Gender*, *Smiling*, and *Hair Color*, with the composition space defined as `Gender={Male, Female}, Smiling={Smiling, Not Smiling}, Hair Color={Black Hair, Blonde Hair}`. Conditioning concepts in CelebA are quantized with less information about the degree of smiling or hair color, which results in harder convergence for novel concept generation. These attributes are treated as binary concept tuples similar to the Shapes2D

setting, enabling consistent evaluation across synthetic and real-world domains. We curate 10,000 images per concept class, yielding a total of 40,000 training examples. This setup allows us to examine how well models trained under structured compositional constraints in CelebA can generalize to novel combinations of facial attributes.

### B.4 CLEVRER-Kubric

The CLEVRER-Kubric dataset is a reimplementation of CLEVRER [70] from scratch using Kubric [23]. This dataset is generated by simulating physics-based scenes using Blender. The scenes are configured with parameters similar to CLEVRER, including a resolution of $480 \times 320$ pixels, a frame rate of 12 fps, and a simulation step rate of 240Hz. Each scene includes a static floor with a gray material and a directional light source. A perspective camera is positioned to view the scene. Stationary objects, numbering between 5 and 7, are randomly chosen from cubes or spheres, and are placed within a pre-defined spawn region. The material properties for individual objects are controlled based on the composition being tested (that is, all instances of a specific `color` or `shape` can be held out to ensure that the model does not see a specific combination during training).

The dynamic aspect involves 1 to 3 projectiles, also from the pre-defined shapes, with similar controlled material and size properties. The projectiles are launched from random points around the perimeter with a velocity between 10 and 15 units, slight randomness to the trajectory, and zero angular velocity. There is increased angular damping to maintain consistency with the original CLEVRER videos. Each composition has 250 videos, each with 28 frames, yielding 7000 frames per composition.

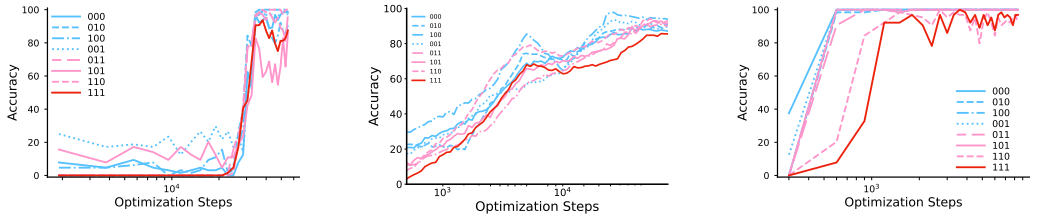
As with Shapes2D, we control for `shape={cube, sphere}, size={large, small}, color={red, green}`. For every composition, we put the desired composition as one (or more) of the static objects to force the model to generate that (and not learn a shortcut by simply generating other parts of the video). We increase the probability of having the dynamic moving object also follow the same properties as the composition to ensure that our methods also learn to model temporal consistency for compositions that are out of the training distribution.

## C Experiments

### C.1 Tokenizer

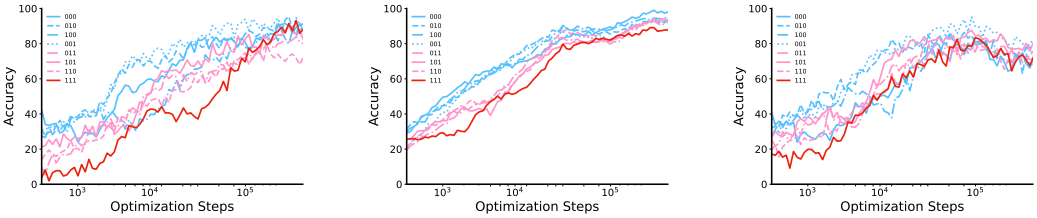
Similar to Section 4.1 of the main text, we also trained MAR with three different types of tokenization schemes to prove that the choice of tokenizer does not alter the compositional capabilities of a model. We show linear probe results in Figure 7. By showing a similar trend, we complement the experiments on DiT shown in the main text, and also show that the choice of downstream generative architecture is not a confounding factor when comparing different tokenizers. To further reinforce the generality of our experiments and demonstrate that the same trend extends to videos, we also present results on CLEVRER-Kubric using a DiT model trained under three different tokenization schemes (Figure 8).

**Mapping discrete tokens to continuous vectors for downstream models** When employing VQ-VAE tokenizer with models like DiT and MAR, the process involves an intermediate step to bridge the discrete nature of VQ-VAE tokens with the continuous vector processing expected by these architectures. *For our experiments in the main text, we use the continuous embedding vectors before the quantization step in the tokenizer*. However, we add a learnable embedding layer in our pipeline for our experiments on CLEVRER-Kubric. The VQ-VAE first encodes an input data sample into a spatial grid of discrete token IDs, where each ID corresponds to a learned codebook vector. For a downstream model like a transformer, these discrete token IDs are then passed through a learnable embedding layer. This layer maps each token ID to a unique, dense, continuous vector representation. It is this grid of continuous vectors (token embeddings) that then serves as the actual input to the downstream generative model. This embedding step effectively translates the discrete symbolic representation from the VQ-VAE into a continuous vector space where the model can perform its computations to achieve generation and composition.




(a) VQ-VAE (b) VAE (c) Pixel (No tokenizer)

Figure 7: MAR exhibits generalization on both level-1 and level-2 compositions in Shapes2D regardless of whether (a) discrete, (b) continuous, or (c) no tokenizer is used. The blue curves show performance on training data, pink curves depict level-1 compositions, and red curve denotes level-2 compositions.




(a) VQ-VAE (b) VAE (c) Pixel (No tokenizer)

Figure 8: DiT exhibits generalization on both level-1 and level-2 compositions in CLEVRER-Kubric regardless of whether (a) discrete, (b) continuous, or (c) no tokenizer is used. The blue curves show performance on training data, pink curves depict level-1 compositions, and red curve denotes level-2 compositions.

### C.2 Training Objective

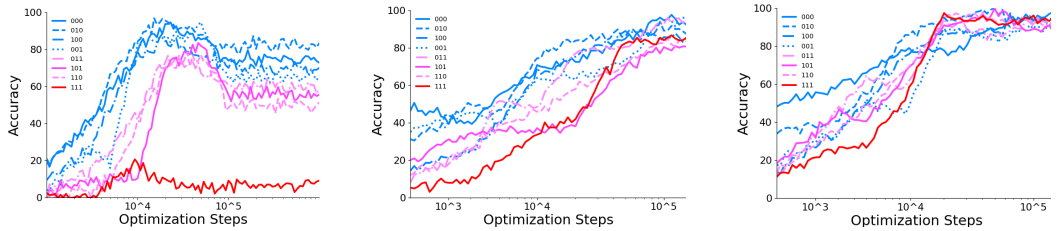
Complementing results from Shapes2D [43] and CelebA [39] in the main text, we show that our findings also hold true for more complex modalities like video. Figure 9 contrasts the performance of a discrete objective (MaskGIT) against a continuous objective (DiT) on the CLEVRER-Kubric dataset. We see the same trend of MaskGIT showing significantly worse performance, especially on level-2 compositions. Following the structure in the main text, we also show results on MAR (Figure 9c) to disentangle the effects of the architectural design from the properties of the latent space and further reinforce our results that objectives modeling underlying continuous factors achieve better compositionality.

**Ablations across training conditions** MaskGIT robustness to training configuration. To rule out under-optimization, capacity, or implicit regularization as confounders for the MaskGIT–DiT gap, we ran an extensive sweep on Shapes2D varying learning rate (5e-5 to 3e-4 and 5e-5 to 1e-5), model depth (12 to 18 layers), batch size (128 and 256), and training length (60k to 200k steps). MaskGIT failed to generalize on level-2 compositions across all configurations, indicating that the gap reflects the categorical objective rather than insufficient tuning. We present our results in Figure 10.

### C.3 Autoregressive Decoding

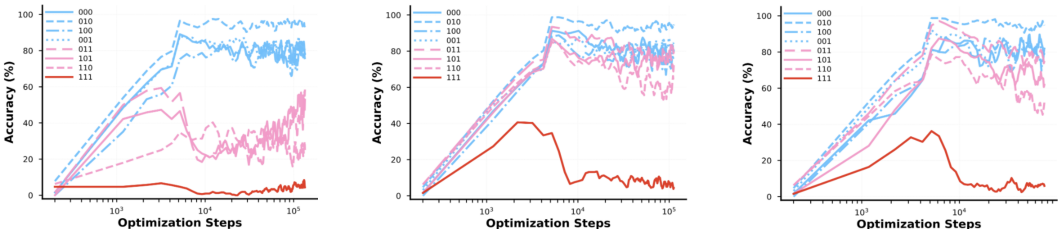
MaskGIT is the only discrete model in our main comparison, and it decodes tokens in parallel. To disentangle the effect of the discrete objective from the decoding scheme, we train an autoregressive (causal) transformer over the same discrete visual tokens that MaskGIT uses, and a continuous counterpart in the GIVT-causal setting [54], which predicts Gaussian mixture parameters over unquantized latents. Both models share the tokenizer/VAE, architecture, and training data of their parallel counterparts on Shapes2D, so the decoding scheme is the only variable.

As shown in Table 2, both causal variants fit the seen compositions comparably (86.7 vs. 90.2), but diverge on unseen ones (38.5/6.2 vs. 64.1/46.9). This reproduces the discrete/continuous gap

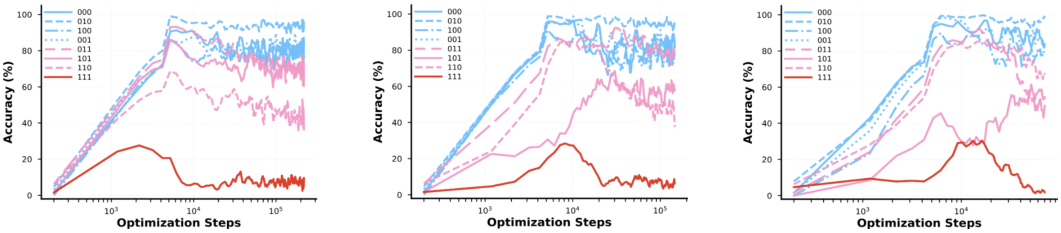



(a) MaskGIT (b) DiT (c) MAR

Figure 9: Comparison of linear probe accuracy for CLEVRER-Kubric across (a) MaskGIT, (b) DiT, and (c) MAR. Models leveraging a continuous latent space (DiT, and MAR) show better level-2 compositions than MaskGIT. The blue curves show performance on training data, pink curves depict level-1 compositions, and red curve denotes level-2 compositions.




(a) High Learning Rate (5e-5 $\rightarrow$ 3e-4) (b) Large Model (12 $\rightarrow$ 18 Layers) – Batch Size 128 (c) Large Model (12 $\rightarrow$ 18 Layers) – Batch Size 128




(d) Long Run (5e-5, 60k $\rightarrow$ 200k steps) (e) Small Learning Rate (5e-5 $\rightarrow$ 1e-5) – Batch Size 128 (f) Small Learning Rate (5e-5 $\rightarrow$ 1e-5) – Batch Size 256

Figure 10: Compositional generalization across different training conditions for MaskGIT on Shapes2D. We see that MaskGIT struggles to compositionally generalize despite varying multiple different training parameters.

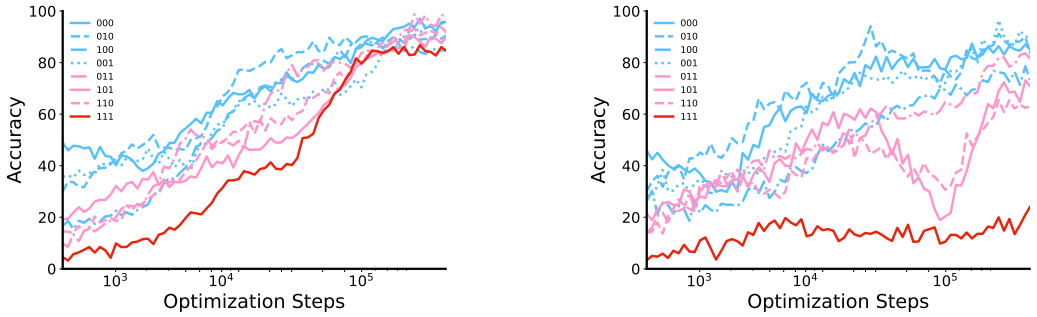
we observe for the parallel models, even though fixed raster-order causal transformers are often suboptimal for vision in contrast to language.

Table 2: Probe accuracy (%) of causal autoregressive models with a discrete (categorical) and a continuous (Gaussian mixture) objective on Shapes2D.

| Model | Seen Mean | Level-1 Novel Mean (011, 110, 101) | Level-2 Novel (111) |
|---|---|---|---|
| AR (discrete) | 86.7 | 38.5 | 6.2 |
| AR-GMM (continuous) | **90.2** | **64.1** | **46.9** |

### C.4 Unmasking Order

MaskGIT decodes iteratively, unmasking the most confident tokens first. One could hypothesize that this heuristic commits early to high-confidence tokens that lock in seen combinations and thereby causes the compositional failures. To test this, we replace confidence-based unmasking with a random




(a) Explicit conditioning

(b) Implicit conditioning

Figure 11: Comparison of linear probe accuracy for DiT trained on CLEVRER-Kubric with (a) Explicit conditioning and (b) Implicit conditioning mechanisms. Training with explicit label information allows for better level-2 compositions. The blue curves show performance on training data, pink curves depict level-1 compositions, and red curve denotes level-2 compositions.

unmasking order while keeping the trained model fixed. Table 3 shows that removing the confidence heuristic does not recover compositional generalization: the novel mean drops by 10.7 points, while seen compositions are unaffected.

Table 3: Probe accuracy (%) of MaskGIT on Shapes2D with confidence-based vs. random unmasking order (256 samples per composition).

| Unmasking | 000 | 001 | 010 | 100 | Seen Mean | 011 | 101 | 110 | 111 | Novel Mean |
|---|---|---|---|---|---|---|---|---|---|---|
| Confidence | 87.8 | 94.3 | 82.9 | 77.3 | 85.6 | 64.9 | 54.0 | 38.4 | 0.7 | **39.5** |
| Random order | 87.8 | 95.0 | 86.1 | 74.3 | 85.8 | 45.7 | 41.2 | 26.3 | 2.0 | 28.8 |

### C.5 Conditioning Information

Section 4.3 examined the effect of different conditioning mechanisms on a DiT trained on Shapes2D. Figure 4d illustrates a memorization failure mode that arises when concept information is only partially provided during training. In such cases, the model tends to generate realistic images that are misaligned with the intended conditioning concept but resemble the closest training instance, e.g., a blue big triangle is rendered as a red big triangle. Figure 11 shows the effect of different conditioning mechanisms for a DiT trained on CLEVRER-Kubric. Given the difference in modalities, we slightly alter our experimental setup to make use of the temporal component in videos that provide a pre-defined context in the form of past frames in which concept information is presented implicitly. For our experiments on CLEVRER-Kubric, we evaluate whether compositional abilities depend on concept information provided explicitly as labels (Figure 11a) or implicitly through context frames (Figure 11b). The results show that explicit signals are crucial for compositional generalization, highlighting the models' limitations in abstracting concept information from implicit observations alone. This finding parallels our Shapes2D results, suggesting that when concept information is partially or not explicitly provided, the model must extract it from raw observations in pixel space, a challenge usually present in representation learning that often limits compositional generalization.

### C.6 Language Conditioning

Our main experiments condition on explicit factor tuples to rule out confounding by a text encoder. Language can be viewed as a projection of the underlying image-generating factors: it quantizes continuous attributes into coarse semantic categories (e.g., many areas become "large") and often omits factors altogether (e.g., "a small circle" instead of "a small red circle"). Language conditioning therefore introduces the same two information-loss mechanisms that we isolate in Appendix C.5.

To test this, we replace factor conditioning with CLIP ViT-B/32 [49] text embeddings on Shapes2D, keeping the experimental setup of DiT and MaskGIT otherwise unchanged. All factors are seen

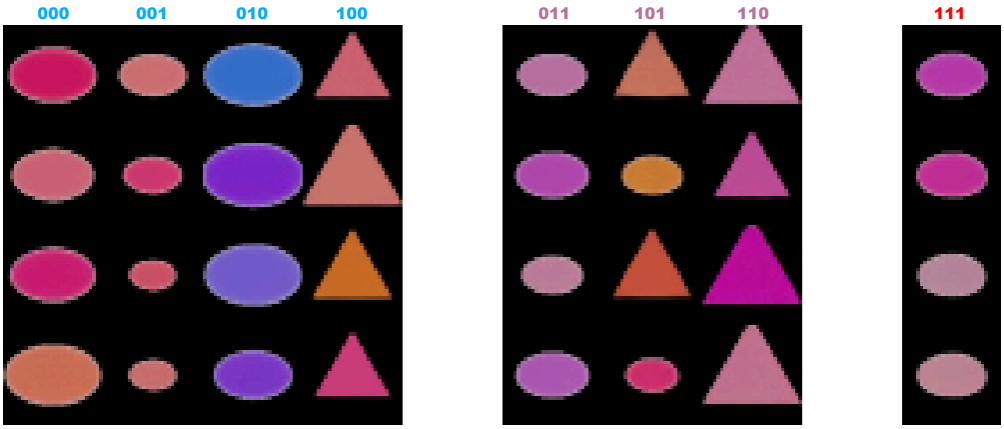



Figure 12: Qualitative Results of the concept dropout with DIT on Shapes2D. The visualization demonstrates that when contextual information about all concepts is incomplete, the model struggles to generalize compositionally. A typical failure mode is the substitution of novel concept combinations with the closest seen ones—for instance, the 110 configuration, which should correspond to a novel blue big triangle, is instead rendered as a red big triangle, likely due to its proximity in the training distribution. Similar behavior is happening to the 101, which should correspond to a novel red small triangle, sometimes result in a red small circle or red big triangle.

during training, but not all of their combinations. We additionally train with partial descriptions, where some factors are omitted from the prompt.

Table 4 shows that, for DiT, language conditioning degrades compositional generalization, and partial descriptions degrade it further, most severely on Level-2 compositions. This matches the degradation predicted by our controlled conditioning study. For MaskGIT, language conditioning does not help the discrete objective, and partial descriptions again hurt. DiT consistently outperforms MaskGIT on novel compositions under both factor and language conditioning.

Table 4: Probe accuracy (%) on Shapes2D with explicit factor conditioning vs. CLIP text conditioning. L1/L2 denote Level-1/Level-2 novel compositions; All Novel averages over all unseen compositions.

| Model | Conditioning | Seen | Novel L1 | Novel L2 | All Novel |
|---|---|---|---|---|---|
| DiT | Factors | **97.3** | **99.0** | **96.9** | **98.4** |
| | Language | 74.2 | 71.4 | 48.4 | 65.6 |
| | Language (partial descriptions) | 76.6 | 62.5 | 1.6 | 47.3 |
| MaskGIT | Factors | 85.6 | 52.4 | 0.7 | 39.5 |
| | Language | 84.1 | 52.0 | 1.6 | 39.4 |
| | Language (partial descriptions) | 85.2 | 19.9 | 0.3 | 15.0 |

### C.7 Per-experiment Compute Budget

Most of our image experiments were carried out on $4\times$NVIDIA L40 GPUs, while our video experiments used $2-4\times$NVIDIA H200 GPUs. We would also be releasing code upon publication to ensure full reproducibility of our results.

## D Enhancing Compositional Learning with Joint Embedding Predictive Architectures

### D.1 Augmenting the MaskGIT Training Objective with JEPA

The core idea of JEPA is to predict the *representation* of masked portions of the input (target) blocks from the *representation* of visible portions (context blocks) in an abstract embedding space

and to learn both the representational embedding *and* the predictor end-to-end. By positioning the target not as a noisy, high-dimensional input, but in a lower-dimensional, cleaned-up embedding, it allows the model to capture relevant factors in the target content. This encourages the encoder to learn representations that capture high-level, predictable semantic information while potentially abstracting away low-level, instance-specific details that may hinder generalization. We formulate our JEPA-based training objective described below.

An input data sample $x$ is first encoded into a sequence of discrete visual tokens $E_{VQ}(x) \in \{1, ..., K\}^N$ where $E_{VQ}$ is the encoder trained with a VQ-VAE objective and $N$ is the sequence length. The standard MaskGIT objective ($\mathcal{L}_{MG}$) focuses on reconstructing masked input tokens. We employ block-masking [4] for images and causal block masking for videos [4] to sample a mask $\mathcal{M} \subset \{1, ..., N\}$. The MaskGIT Transformer $T$ predicts the probability distribution $p(z_i|z_{\setminus M})$ for each masked token $z_i$ ($i \in M$) based on unmasked token $z_{\setminus M}$. The loss is the negative log-likelihood of the true tokens at the masked positions:

$$\mathcal{L}_{MG} = -\mathbb{E}_{x,M}\left[\sum_{i \in M} \log p(z_i|z_{\setminus M})\right] \tag{1}$$

Let $H^{(l)}$ represent the sequence of continuous hidden state vectors output by the $l$-th layer of the MaskGIT Transformer $T$. We introduce a JEPA-style objective applied to the hidden states of specific intermediate layers $l \in L_{JEPA}$. This acts as a representation alignment objective within the network.

For each selected layer $l \in L_{JEPA}$, we apply a separate JEPA masking strategy $M_{JEPA}$. This strategy defines a set of context indices $C \subset \{1, ..., N\}$ and target indices $T \subset \{1, ..., N\}$, typically corresponding to spatial blocks. We extract the hidden states from layer $l : H^{(l)} = \{h_1^l, ..., h_N^l\}$. The extracted *context* hidden states $H_C^{(l)} = \{h_j^{(l)} | j \in C\}$ is used as input to a layer-specific predictor network $P_{JEPA}^{(l)}$. In practice, $P_{JEPA}^{(l)}$ is simply parametrized using a multilayer perceptron (MLP). The predictor network aims to predict the hidden states of the target tokens $\hat{H}_T^{(l)} = P_{JEPA}^{(l)}(H_C^{(l)})$.

The JEPA loss for layer $l$ measures the discrepancy between the predicted target representation (for target index $k \in T$) $\hat{h}_k^{(l)}$ and the actual target representations $h_k^{(l)}$ (computed by the EMA Transformer $T$ itself) using Mean Squared Error (MSE) as a distance metric $d$

$$\mathcal{L}_{JEPA}^{(l)} = \mathbb{E}_{x, M_{JEPA}}\left[\frac{1}{|T_{idx}|}\sum_{k \in T_{idx}} d\left(\hat{h}_k^{(l)}, sg(h_k^{(l)})\right)\right] \tag{2}$$

where $T_{idx}$ denotes the set of target indices and $sg(\circ)$ denotes the stop-gradient operation. This ensures that the encoder is updated to produce context representations $H_C^{(l)}$ that are predictive of the target representations $h_k^{(l)}$, rather than trivially learning to reconstruct $h_k^{(l)}$ through the target pathway.

The final JEPA loss can be the same as Equation (2) if applied only to a single layer, or a summation of losses applied to all layers. In the latter case:

$$\mathcal{L}_{JEPA} = \sum_{l \in L_{JEPA}} \mathcal{L}_{JEPA}^{(l)} \tag{3}$$

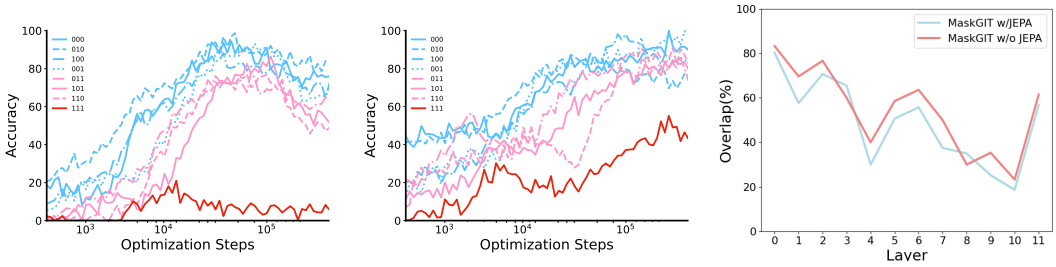
The final training objective is a weighted sum of the standard MaskGIT reconstruction loss (Equation (1) and the JEPA representation alignment loss (Equation (3)):

$$\mathcal{L}_{Total} = \mathcal{L}_{MG} + \lambda \mathcal{L}_{JEPA} \tag{4}$$

where $\lambda$ is a hyperparameter balancing the contribution of the two objectives.

### D.2 Results on CLEVRER-Kubric

To reinforce our findings from the main text, we also extend our JEPA-based objective for video generation using the CLEVRER-Kubric dataset. We see that our results follow a similar trend as reported for Shapes2D in the main text. Figure 13a shows that the standard MaskGIT objective is insufficient for reliable compositional generalization. However, augmenting it with our JEPA-based objective yield improvement in compositional abilities (Figure 13b). We also conduct a




(a) Standard Objective (b) JEPA-based Objective (c) Polysemanticity Analysis

Figure 13: Comparison of linear probe accuracy for MaskGIT with Standard (a) and JEPA-based (b) training objectives for CLEVRER-Kubric. The JEPA-based training objective clearly enhances compositional abilities, and lowers polysemanticity (c) even though it cannot fully compensate the problems introduced by the discrete space. The blue curves show performance on training data, pink curves depict level-1 compositions, and red curve denotes level-2 compositions.

polysemanticity analysis (Figure 13c), observing the same trend as Shapes2D, with the JEPA-augmented objective exhibiting lower polysemanticity in attention heads- which points to the model learning factor-specific representations.

### D.3 Ablation Studies

**Analysis on JEPA-loss on different layers** As seen in the previous sections, we apply the JEPA loss to different layers $l$ within the transformer block. We show an empirical ablation below contrasting the performance of applying the JEPA loss to different layers for a MaskGIT model trained on CLEVRER-Kubric Table 5 shows the maximum probe accuracy of the `111` composition with the JEPA losses at different layers or combinations of layers. Following previous works [71], we also applied predictive coding in the representations of mid-level layers. We see slight differences after varying combinations but we phrase the problem as parameter tuning and empirically select the best combination for optimal trade-off between accuracy and performance.

Table 5: Maximum linear probe accuracy after applying the JEPA loss on different layers and combinations of layers for a MaskGIT model trained on CLEVRER-Kubric

| Layers | Accuracy |
|---|---|
| {6} | 27.61 |
| {8} | 26.35 |
| {6,8} | 38.16 |
| {7,9} | 36.62 |
| {6,8,10} | 53.52 |
| {7,9,11} | **56.27** |
| {7,8,9,11} | 47.25 |

**Effect of $\lambda$** We also provide an empirical ablation on varying the weighting factor $\lambda$ as seen in Equation (3). Based on the graph, we select 0.6 as the weighting factor for our experiments.

### D.4 Polysemanticity in Attention Heads

We give a deeper explanation into how we perform the polysemanticity analysis mentioned in the main text. We quantify polysemanticity in attention heads by assessing their impact on the model's ability to distinguish between different visual features like `color` or `shape`. An attention head is identified as polysemantic if it influences the processing of multiple distinct features. The core of this approach involves an intervention- specifically, ablating individual attention heads and measuring the resultant change in the model's representation for contrasting pairs of data samples.

First, we establish a baseline for how the original, unablated model represents and distinguishes the features of interest. We do so by passing curated pairs of images or videos, with the samples in each pair differing along a single, targeted visual feature dimension. For instance, to assess color processing, we use pairs of `red square` and `blue square`. For each sample $I$, its global representation is extracted from the model, typically by using an additional token that aggregates all features of the sample (similar to the `[CLS]` token). These embeddings are L2 normalized before further use.

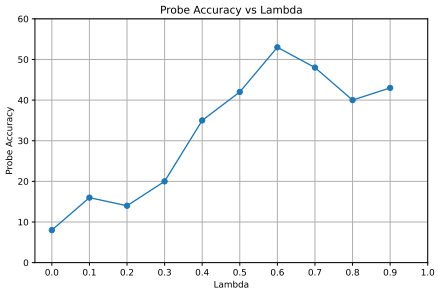



Figure 14: Effect of weighting factor on probe accuracy

$$[CLS]_{norm}(I) = \frac{[CLS](I)}{||[CLS](I)||_2} \tag{5}$$

The model's ability to distinguish a feature $f$ is quantified by the cosine similarity between these embeddings for pair $(I_a, I_b)$ contrasting in that specific feature. Given that the tokens are already L2-normalized, their cosine similarity is simply their dot product

$$sim(I_a, I_b) = [CLS]_{norm}(I_a) \cdot [CLS]_{norm}(I_b) \tag{6}$$

This gives us a baseline similarity score $sim_{baseline,f}$ for each feature under consideration.

Next, we perform targeted ablation of a specific attention head $h$ within a given layer $l$. This intervention involves creating a copy of the model and modifying the forward pass of its multi-head self-attention (MHSA) module. Specifically, the $QKV$ vectors corresponding to $h$ are set to 0 immediately before the attention scores are computed. This allows us to observe the model's behavior in the absence of the specific attention head $h$.

Using the ablated model, $M_{ablated(l,h)}$, we recalculate the L2-normalized `[CLS]` token embeddings and subsequently the feature similarity score $sim_{ablated,f}$ for the same pairs. The impact $E_f(l,h)$ of ablating head $(l,h)$ on the processing of feature $f$ is defined as the change in this similarity score:

$$E_f(l,h) = sim_{ablated,f} - sim_{baseline,f} \tag{7}$$

An attention head $(l,h)$ is considered to significantly impact feature $f$ if the absolute magnitude of its effect $|E_f(l,h)|$, exceeds a predefined threshold $\tau$. For our experiments, we consider $\tau = 0.005$. The head is then defined polysemantic if it meets this significance criterion for multiple distinct features.

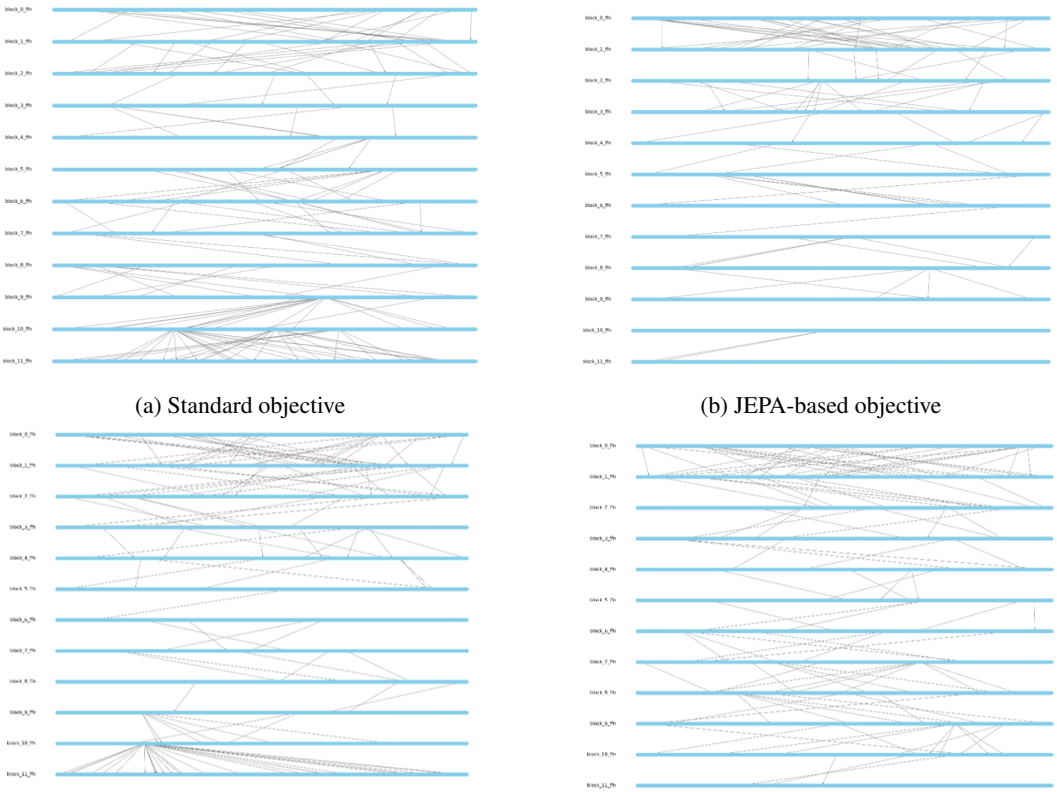
### D.5 Mechanistic Similarity

We provide a more comprehensive explanation of our method to compare the circuit overlap for different concepts across different models. Similar to Kempf et al. [30], we first identify important neurons within the MLP blocks by perturbing their activations. Specifically, we zero out the output of the fully connected layer for a neuron and measure the impact on the model's classification score for a target class. The impact is defined as

$$\text{Impact}(a_{l,j}) = |S_C(I) - S'_C(I|a_{l,j} = 0)| \tag{8}$$

where $S_C(I)$ is the baseline score for class $C$ given data sample $I$ and $S'_C(I|a_{l,j} = 0)$ is the score when neuron $j$'s activation in block $l$ is zeroed out $(a_{l,j} = 0)$. The top $10\%$ neurons causing the largest score change are selected.

Next, we identify important connections between these MLP neurons across consecutive transformer blocks (from block $k$ to $k+1$). We attribute the activation of a target important neuron in the fully connected layer of block $k+1$ to the activations of neurons in block $k$. A connection is deemed important if the source neuron has high attribution and was also identified as an important neuron.



(a) Standard objective

(b) JEPA-based objective

(c) Standard objective

(d) JEPA-based objective

Figure 15: Visualization of MaskGIT circuits for a `large red cube` and `small blue sphere` trained with the standard training objective versus the JEPA-based training objective.

Finally, these important neurons and connections form a class specific circuit-graph (visualized in Figure 15). We then compare circuits from different classes using Jaccard similarity for measuring neuron overlap.

**Do factor-specific circuits support compositional generalization?** To provide evidence for the role of factor-specific circuits, we conduct a circuit ablation experiment. For each factor, we identify relevant neurons by comparing circuits of composition pairs differing only in that factor (e.g., large red sphere vs. large blue sphere yields color-relevant neurons), zero them out at inference, and measure the resulting drop in level-2 probe accuracy. Table 6 shows our results.

Table 6: Level-2 probe accuracy drops after ablating factor-specific circuits. Bold entries mark factor-aligned ablations. MaskGIT+JEPA shows selective degradation; standard MaskGIT shows uniform drops, consistent with entangled representations.

| | Color-ablated | Shape-ablated |
|---|---|---|
| *MaskGIT+JEPA* | | |
| Color-varying | ↓ **38pp** | ↓ 6pp |
| Shape-varying | ↓ 5pp | ↓ **41pp** |
| *Standard MaskGIT* | | |
| Color-varying | ↓ 18pp | ↓ 14pp |
| Shape-varying | ↓ 15pp | ↓ 17pp |

In MaskGIT+JEPA, ablating a factor's neurons selectively degrades the compositions influenced by that factor, whereas standard MaskGIT shows uniform drops across factors, consistent with entangled

representations. This provides evidence that the factor-specific circuits induced by the JEPA objective support compositional generalization.

# E Theoretical Context for Our Empirical Findings

In this section, we connect our empirical findings to existing theoretical results on compositional generalization. Several independently developed frameworks provide perspectives that are consistent with our observations, although their assumptions do not exactly match our experimental setting. For each finding, we identify the relevant theoretical result and discuss how its assumptions relate to our setup.

## E.1 Finding 2: Theoretical Perspectives on Continuous Objectives and Compositional Generalization

Our central empirical result is that models trained with continuous objectives (DiT, MAR, GIVT) generalize compositionally to unseen factor combinations, while models trained with discrete categorical objectives (MaskGIT) do not. Three independent theoretical frameworks provide complementary perspectives consistent with this pattern.

**Smooth-decoder results on Cartesian-product extrapolation.** Lachapelle et al. [34] (Theorems 1 & 2) prove that an additive decoder $\mathbf{f}(\mathbf{z}) = \sum_k \mathbf{f}^k(\mathbf{z}_{B_k})$ achieves *Cartesian-product extrapolation*: if the model fits the ground-truth generator on a training support $\mathcal{Z}_{\text{supp}}$, it also agrees on the full Cartesian product $\mathcal{Z}_1 \times \cdots \times \mathcal{Z}_K$. This result requires each component function $\mathbf{f}^k$ to satisfy a *sufficient nonlinearity* condition stated in terms of the linear independence of first- and second-order partial derivatives of $\mathbf{f}$. Brady et al. [9] generalize this in Theorem 4.4: compositional generalization holds when both the ground-truth generator and the learned model are $C^{n+1}$ diffeomorphisms with at most $n$th-order interactions across concept slots. Wiedemer et al. [62] (Theorem 1) prove a complementary result: a compositional model generalizes from training support to unseen compositions when the support satisfies a *sufficient support* condition (Definition 3) and the model shares the compositional function class with the ground truth.

All three results require the decoder to be a smooth, continuously differentiable function of its latent inputs. Models trained with the continuous objectives considered here (denoising diffusion, Gaussian mixture likelihood) operate on continuous-valued representations and are parameterized by smooth neural networks, making them more closely aligned with these assumptions. In contrast, MaskGIT's categorical cross-entropy loss predicts a distribution over discrete codebook indices, and the derivative-based arguments above do not directly apply to categorical token prediction. This does not imply that categorical models cannot generalize compositionally; rather, these particular theoretical guarantees do not directly characterize categorical token prediction. Specifically, the key mechanism in Brady et al.'s compositional generalization proof—that the partial derivative $\partial f_l / \partial z_1(z_1^*, z_2)$ is a bounded-degree polynomial in $z_2$ and therefore globally determined by its local behavior—is formulated for continuous latent variables and differentiable decoders.

**Score-based objectives and compositional inductive biases.** Beyond the smoothness requirement, recent theoretical work identifies specific inductive biases of score-matching objectives that favor compositional generation. Kamb & Ganguli [28] prove that convolutional diffusion models, through their locality and equivariance biases, learn *local score functions* that produce novel images by combinatorially composing local patches from training examples—a mechanism they term "combinatorial creativity." Bradley et al. [8] extend this to conditional diffusion, proving an exact equivalence between *conditional projective composition* (a formal definition of correct compositional generation) and *local conditional scores* – score functions with sparse dependencies on both pixels and conditioning variables. These results provide concrete examples in which score-based objectives admit structural biases favorable to compositional generation, although both analyses rely on locality or architectural assumptions that differ from the transformer models we train. These guarantees do not directly extend to categorical token prediction, where a score with respect to a discrete token variable is not defined in the same sense.

**Regression versus classification extrapolation biases.** From an optimization perspective, results on ReLU networks trained with MSE show a tendency toward linear extrapolation outside the training

support under specific assumptions [67]. This provides another perspective consistent with the idea that regression objectives can favor smooth extensions beyond observed samples. In contrast, classical results for cross-entropy in separable classification characterize a maximum-margin bias [53]. Although neither result directly characterizes the transformer settings studied here, they illustrate qualitatively different extrapolation biases induced by regression and classification objectives.

### E.2 Finding 1: Tokenizer Choice Does Not Substantially Alter Compositionality

Our experiments show that using a VQ-VAE (discrete) versus a VAE (continuous) tokenizer does not alter compositional generalization, as long as the downstream generative objective is continuous. This is consistent with the theoretical framework above: the compositional generalization theorems constrain the *generative model's decoder* (the function from latent space to observations), not the *tokenizer's* encoding scheme. A VQ-VAE tokenizer maps images to discrete codebook indices, but the codebook entries themselves are continuous vectors. When a continuous generative model (e.g., DiT) operates on continuous-valued inputs (like learned token embeddings for CLEVRER-Kubric, Section C.1) rather than on the discrete indices, the downstream mapping remains continuous, retaining the type of smooth structure assumed by these results. Barin-Pacela et al. [5] formalize a related point: latent factors can be identified even when quantized, provided the underlying mixing function remains a smooth diffeomorphism. Their analysis concerns identifiability rather than generation, so we treat it as supporting context.

### E.3 Finding 3: The Role of Conditioning Information

We find that quantized or incomplete conditioning on generative factors during training degrades compositional generalization, even for continuous models. Wiedemer et al. [62] (Theorem 1) provide one useful theoretical lens: their result requires the training distribution to have *sufficient support* (Definition 3)—the support must lie in an open set and, for each component function, the subset of training points sharing a fixed value of one factor must contain enough variation in the remaining factors for the Jacobian to be identifiable. Viewed through this lens, quantized or incomplete conditioning can make the effective factor information available to the model coarser or ambiguous, potentially moving the learning problem away from the support and identifiability conditions required by these guarantees. The theorem does not directly model conditioning corruption of the form used in our experiments, so we view this connection as suggestive rather than a direct implication.

Brady et al. [9] offer a related lens through a different formalism: their disentanglement result (Theorem 4.3) requires $\mathcal{Z}_{\text{supp}}$ to be *regular closed* (closure of its interior) and *aligned-connected* (for any slot value, the set of training points sharing that value must be path-connected). Under this view, quantized conditioning could make the effective support more fragmented and incomplete conditioning could reduce its effective dimensionality, both of which would move the problem away from aligned-connectedness and regular closedness. As with Wiedemer et al. [62], these conditions are stated for the latent support rather than for the conditioning signal, so we view this connection as suggestive.

### E.4 Finding 4: JEPA Auxiliary Loss Partially Recovers Compositionality

Adding a continuous JEPA-based auxiliary loss to MaskGIT partially recovers compositional generalization and yields more disentangled internal representations. The JEPA objective introduces a continuous regression signal on intermediate representations. This is qualitatively aligned with the smooth-regression perspective above and provides one possible interpretation of why the auxiliary objective helps. However, the theoretical results discussed above do not directly apply to these intermediate representations, so we do not interpret them as a proof of the mechanism. The empirical improvement from JEPA, which narrows but does not close the gap to models whose generative loss is itself continuous, is consistent with this interpretation.

# F Evaluation

## F.1 Validity of Probe-Based Evaluation

A composition is considered correct if, for every factor, the intended class has the highest probe probability among its $C$ classes (i.e., more than $0.5$ for binary factors). We verify that this criterion does not reward distorted or ambiguous generations on Shapes2D with three analyses.

**Consistency with qualitative results.** In randomly sampled generations (Figure 16), MaskGIT produces visibly worse compositions than DiT, consistent with the probe results.

**Probe decisions are rarely borderline.** Across 512 generated images (three factor predictions each), only 11.6% of MaskGIT's correct predictions fall in the borderline range and 75.5% exceed 0.9. For DiT, these numbers are 3.6% and 90.7%.

**Probes are sensitive to visual quality.** On 1600 real test images, probe accuracy degrades monotonically under increasing corruption strength for blur, Gaussian noise, patch shuffling, occlusion, and downsampling (Table 7). Factors that depend on spatial structure (shape, size) are affected first, while color remains largely intact, as expected.

Table 7: Probe accuracy (%) on real Shapes2D test images under increasing corruptions. Joint denotes all three factors being classified correctly.

| Corruption | Strength | Shape | Color | Size | Joint |
|---|---|---|---|---|---|
| Blur ($\sigma$) | 1 | 98.9 | 100.0 | 89.5 | 88.4 |
| | 2 | 99.2 | 100.0 | 91.7 | 90.9 |
| | 4 | 73.1 | 100.0 | 89.3 | 64.7 |
| | 8 | 0.0 | 100.0 | 50.0 | 0.0 |
| Noise ($\sigma$) | 0.1 | 97.4 | 100.0 | 93.9 | 91.2 |
| | 0.25 | 91.1 | 100.0 | 78.2 | 75.5 |
| | 0.5 | 65.1 | 99.8 | 50.0 | 45.4 |
| | 1 | 4.5 | 94.4 | 50.0 | 4.0 |
| Patch shuffle ($k \times k$) | 2 | 42.6 | 99.6 | 57.6 | 18.5 |
| | 4 | 22.6 | 99.9 | 53.1 | 11.7 |
| | 7 | 21.9 | 100.0 | 50.9 | 9.1 |
| | 14 | 12.9 | 100.0 | 50.0 | 3.6 |
| Occlusion (fraction) | 0.25 | 99.1 | 98.5 | 92.1 | 91.5 |
| | 0.5 | 88.4 | 69.7 | 55.6 | 51.2 |
| Downsample (factor) | 4 | 92.2 | 100.0 | 89.0 | 81.2 |
| | 7 | 44.5 | 100.0 | 87.4 | 39.8 |
| | 14 | 0.0 | 100.0 | 50.0 | 0.0 |

## F.2 Shapes3D

| **Model** | (0,0,0) | (0,0,1) | (0,1,0) | (1,0,0) | (1,0,1) | (1,1,0) | (0,1,1) | (1,1,1) |
|---|---|---|---|---|---|---|---|---|
| DiT | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% |
| MaskGIT | 94.7% | 97% | 94.7% | 94% | 79.49% | 62.6% | 58.85% | 30% |

Table 8: Performance comparison of DiT and MG across different compositions.

We follow the Shapes2D evaluation protocol and measure compositional generalization using probe accuracy. The probe is a lightweight ResNet-style classifier trained to predict object properties from generated images. It consists of three convolutional blocks with residual connections, followed by global average pooling. The output is passed to three separate linear heads that predict the object's hue, shape, and scale.

As shown in Table 8, our key finding still holds: DiT (continuous) generalizes effectively to novel compositions, while MaskGIT (discrete) struggles. Notably, MaskGIT exhibits failure cases such as non-monotonic color attribution to objects and regression to nearest-neighbour training examples.

### F.3 CelebA

Table 9: Compositional Retrieval Accuracy (CRA, %) 100 generated samples.

| Model | 1-NN | 5-NN |
|---|---|---|
| DiT | 27 | **36** |
| MaskGIT | 21 | 16 |
| MaskGIT + JEPA | **31** | 27 |

Since there are no established metrics that effectively measure compositional generalization in visual generative models, we introduce a retrieval-based evaluation metric that assesses the compositionality. The core idea is to test whether generated samples from an unseen composition (e.g., `111`, two factors away from the trainset) are closest in feature space to real images of the same composition. We compute DINOv2 [45] distances between each generated image and all real images for the eight compositions. Using Nearest Neighbor retrieval accuracy, we measure how often a generated image is closest to real images from its target composition. We evaluate DiT, MaskGIT, and JEPA-enhanced MaskGIT on samples from the unseen composition `111`. Models leveraging continuous distributions show a clear advantage in compositional generalization (Table 9).

We report FID and FDD on seen and unseen splits as complementary distributional metrics (Table 10). These Fréchet distances capture split-wise alignment between generated and real samples, but they are not compositionality measures. MaskGIT degrades substantially on the unseen split, while JEPA-enhanced MaskGIT reduces this gap, particularly in unseen FDD, indicating improved alignment to the unseen distribution.

Table 10: FID and FDD scores on seen and unseen splits on CelebA.

| Method | Unseen FID ↓ | Unseen FDD ↓ | Seen FID ↓ | Seen FDD ↓ |
|---|---|---|---|---|
| DiT | **118.60** | 993.92 | **109.96** | 681.78 |
| MG | 162.38 | 1470.78 | 113.94 | 719.95 |
| MG+JEPA | 120.76 | **980.92** | 117.35 | **621.43** |

## G Results

### G.1 Shapes2D

Figure 16 gives a comprehensive overview of qualitative results on Shapes2D.

### G.2 Shapes3D

Figures 17 and 18 gives a comprehensive overview of qualitative results on Shapes3D. For easier indication of compositional novelty, we use padding colors based on the sum of supergroup assignments across the three factors (color, shape, size), where each factor contributes 0 or 1 depending on its supergroup. Dark blue indicates a sum of 0 (all factors from Supergroup 0), light blue indicates a sum of 1, pale orange corresponds to a sum of 2, and dark red represents a sum of 3 (all factors from Supergroup 1, i.e., the most novel compositions).

### G.3 CelebA

Figure 19 gives a comprehensive overview of qualitative results on CelebA.

### G.4 CLEVRER-Kubric

Qualitative results on CLEVRER-Kubric are presented as a separate website in the included HTML file. We show results for `111 - {large red cube}`, and can see that MaskGIT trained with the standard objective generates colors well but struggles to assign them to specific shapes. In contrast, MaskGIT trained with a JEPA-based objective and models like DiT which leverage a continuous latent space show better performance, as reflected by the probe accuracies (Figure 9).

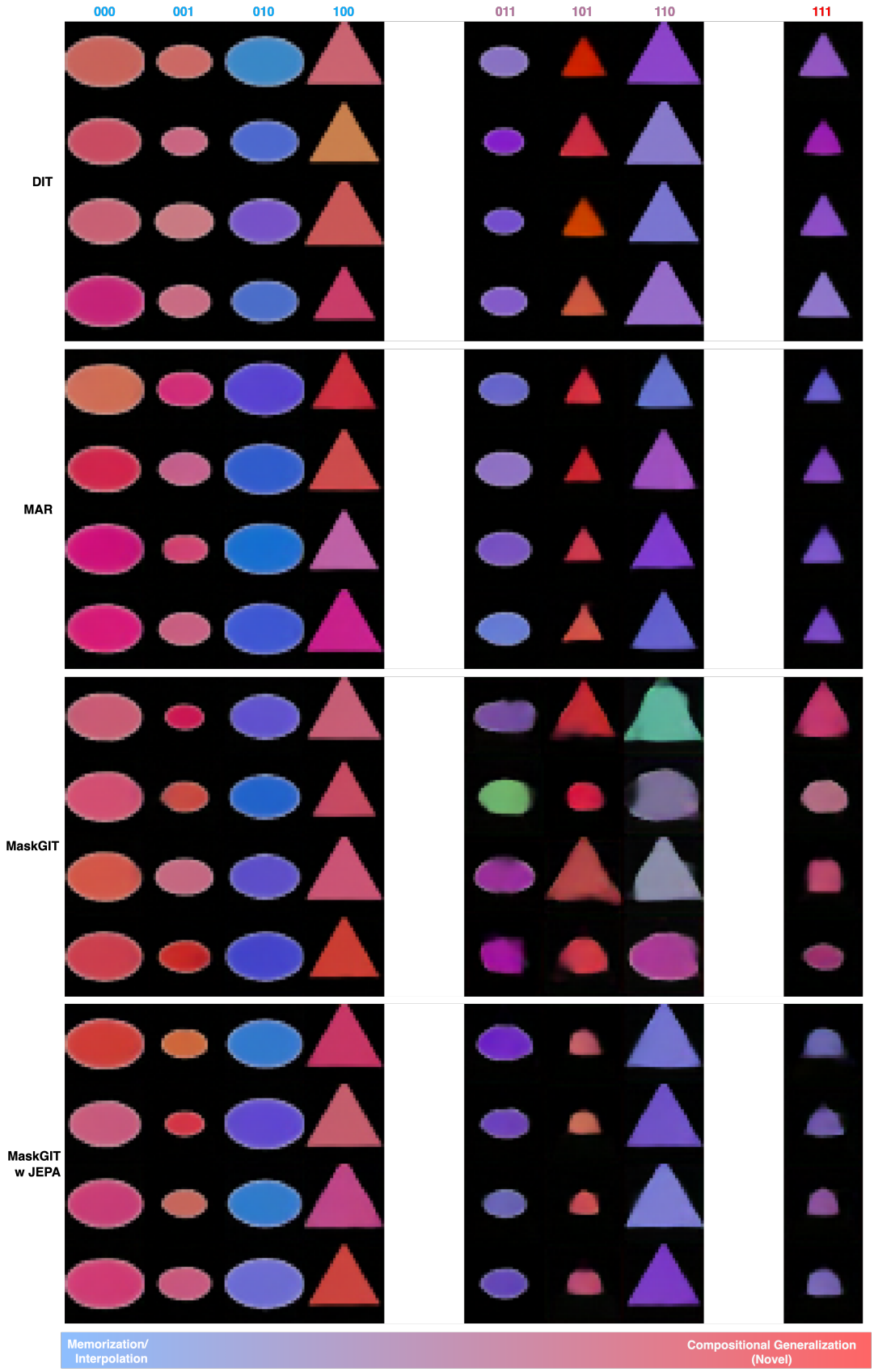



Figure 16: Qualitative Results on Shapes2D

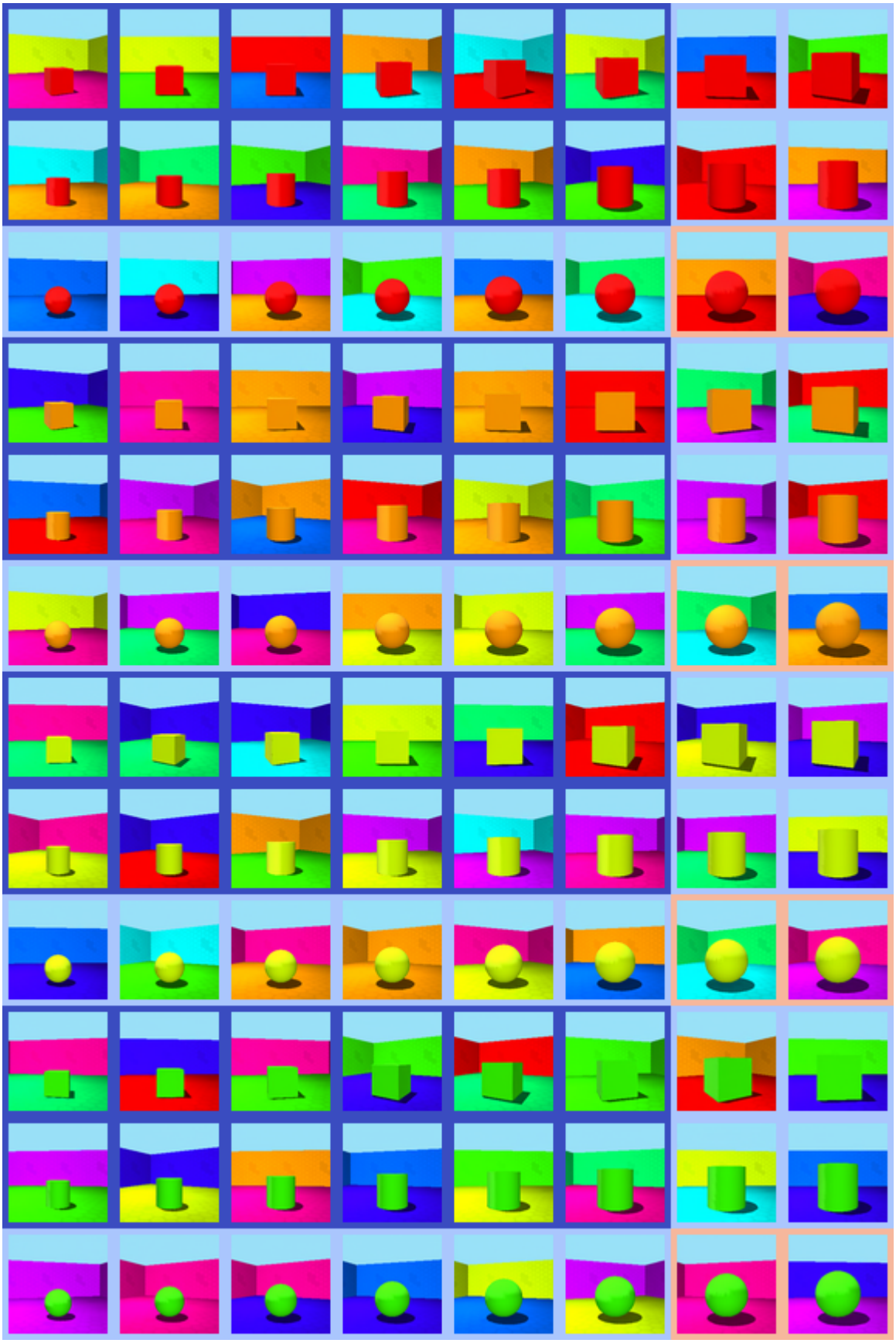


Figure 17: Shapes3D results of DiT. Dark blue: all factors from Supergroup 0; light blue: one factor from Supergroup 1; pale orange: level-1 compositions; and dark red: level-2 novel compositions.

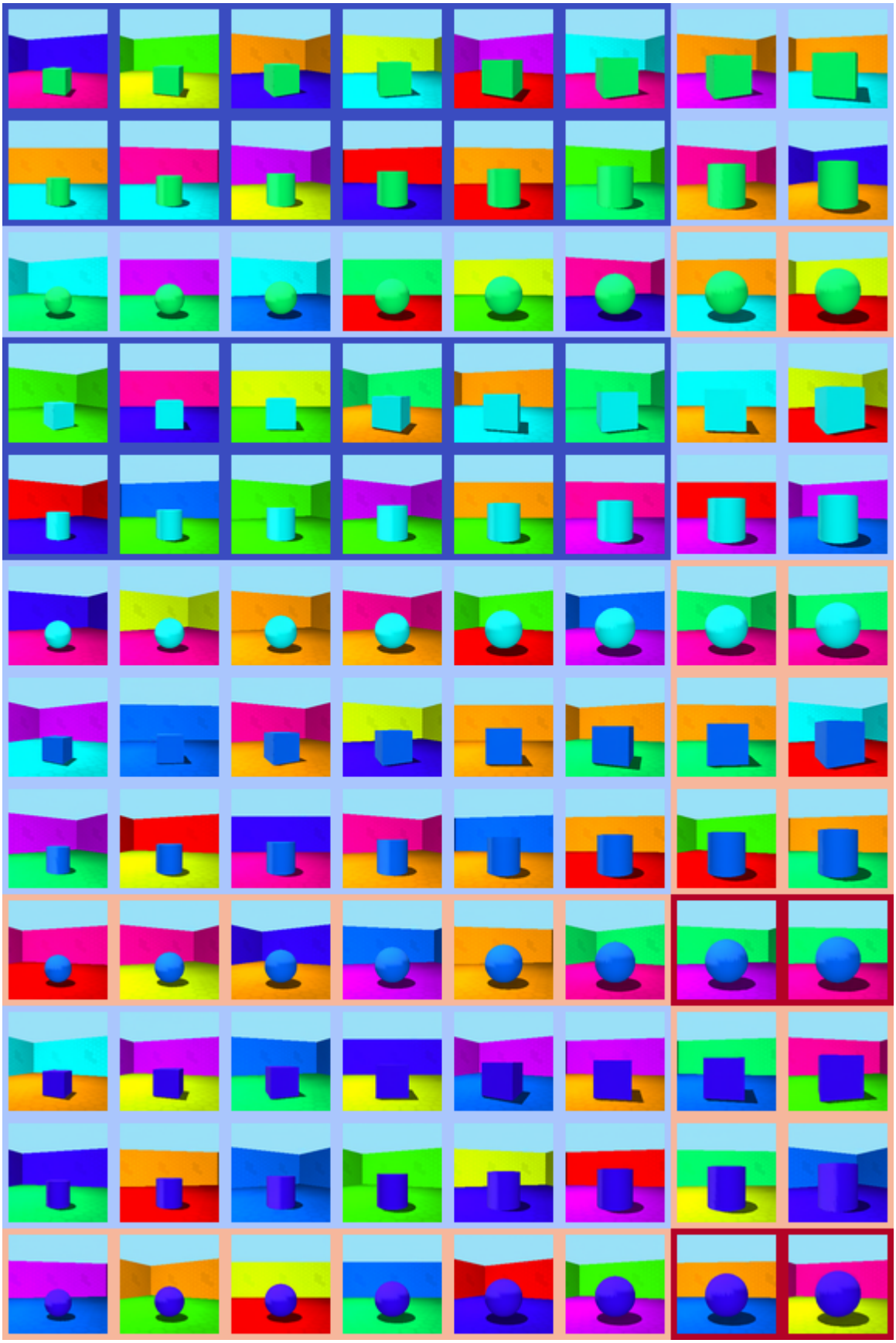


Figure 17: Shapes3D results of DiT. Dark blue: all factors from Supergroup 0; light blue: one factor from Supergroup 1; pale orange: level-1 compositions; and dark red: level-2 novel compositions.

Table 11: Factorization of the data into *Super Groups*. This partitioning is used to evaluate compositional generalization across disjoint super groups.

| Factor | Possible Values | Super Group 0 | Super Group 1 |
|---|---|---|---|
| Color | red, orange, yellow, lime green, green, cyan, blue, indigo, purple, pink | red, orange, yellow, lime green, green, cyan, blue | indigo, purple, pink |
| Shape | cube, cylinder, sphere | cube, cylinder | sphere |
| Size | 0–7 (ordinal) | 0, 1, 2, 3, 4, 5 | 6, 7 |

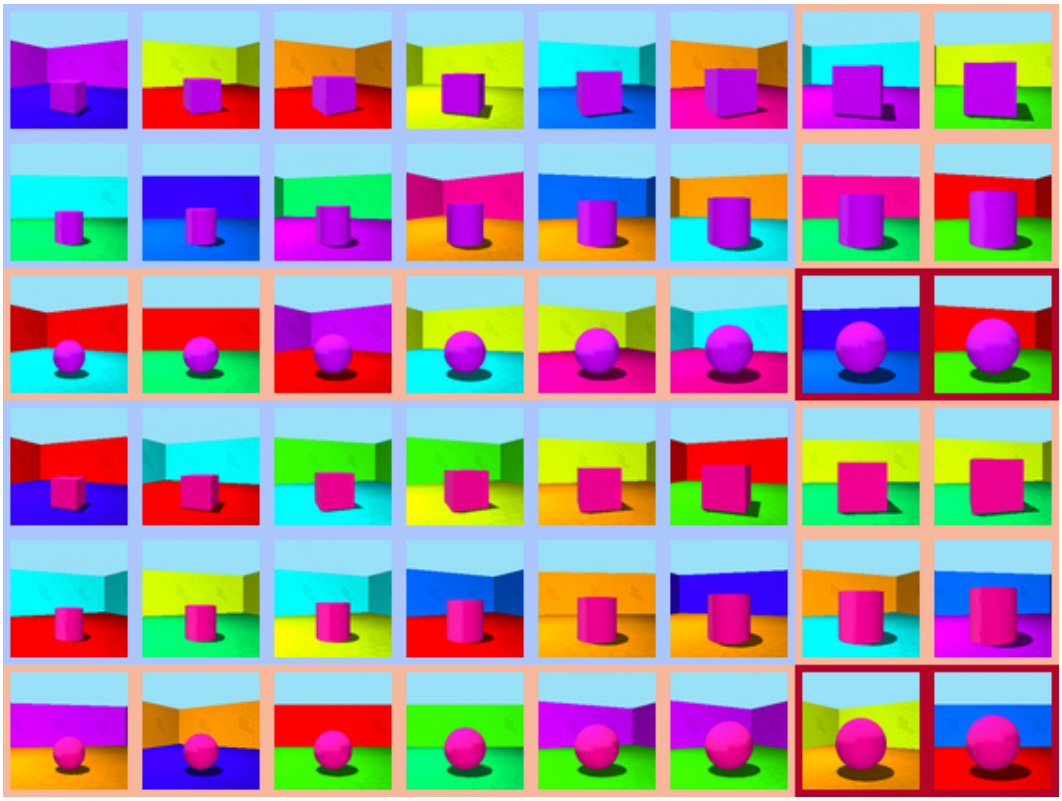


Figure 17: Shapes3D results of DiT. Dark blue: all factors from Supergroup 0; light blue: one factor from Supergroup 1; pale orange: level-1 compositions; and dark red: level-2 novel compositions.

## H Worlds Models: CoVLA

We use CoVLA [2], a real-world driving video dataset collected in Tokyo with synchronized front-facing camera and CAN/GNSS/IMU signals. We do the compositional splits on two factors: time of day (day $\odot$, night $\bigcirc$) and turn direction (left $\leftarrow$, straight $\uparrow$, right $\rightarrow$). The held-out novel compositions are $\odot\rightarrow$ and $\bigcirc\leftarrow$; all remaining combinations constitute the *real-train* set.

**Labeling and segmentation.** We assign the night label using `in_tunnel` flag or explicit night time at the exact timestamp for the frame, label *turns* when speed $\geq 1.8\,\mathrm{km/h}$ and steering angle $\geq 10^\circ$ (left/right), and segment videos so each clip contains exactly one scenario (no factor changes within a clip).

**Models and training.** We instantiate ORBIS [41] in two matched variants: Orbis-MaskGIT (categorical objective) and Orbis-DiT (continuous objective). Both use a spatiotemporal Transformer (24 layers, hidden size 768, input resolution $320 \times 640$), trained for 18 epochs with identical step

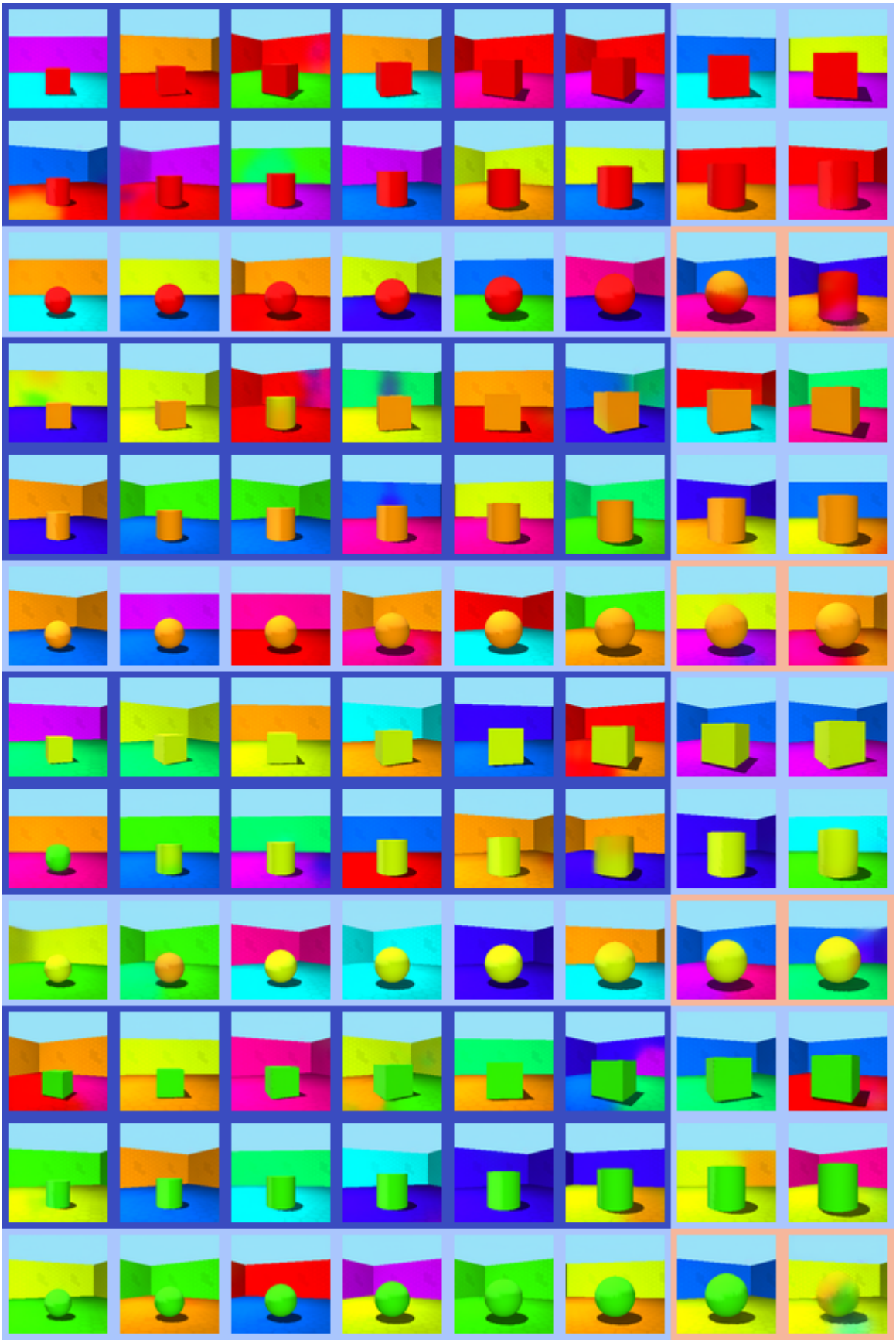


Figure 18: Shapes3D results of MaskGIT. Dark blue: all factors from Supergroup 0; light blue: one factor from Supergroup 1; pale orange: level-1 compositions; and dark red: level-2 novel compositions.

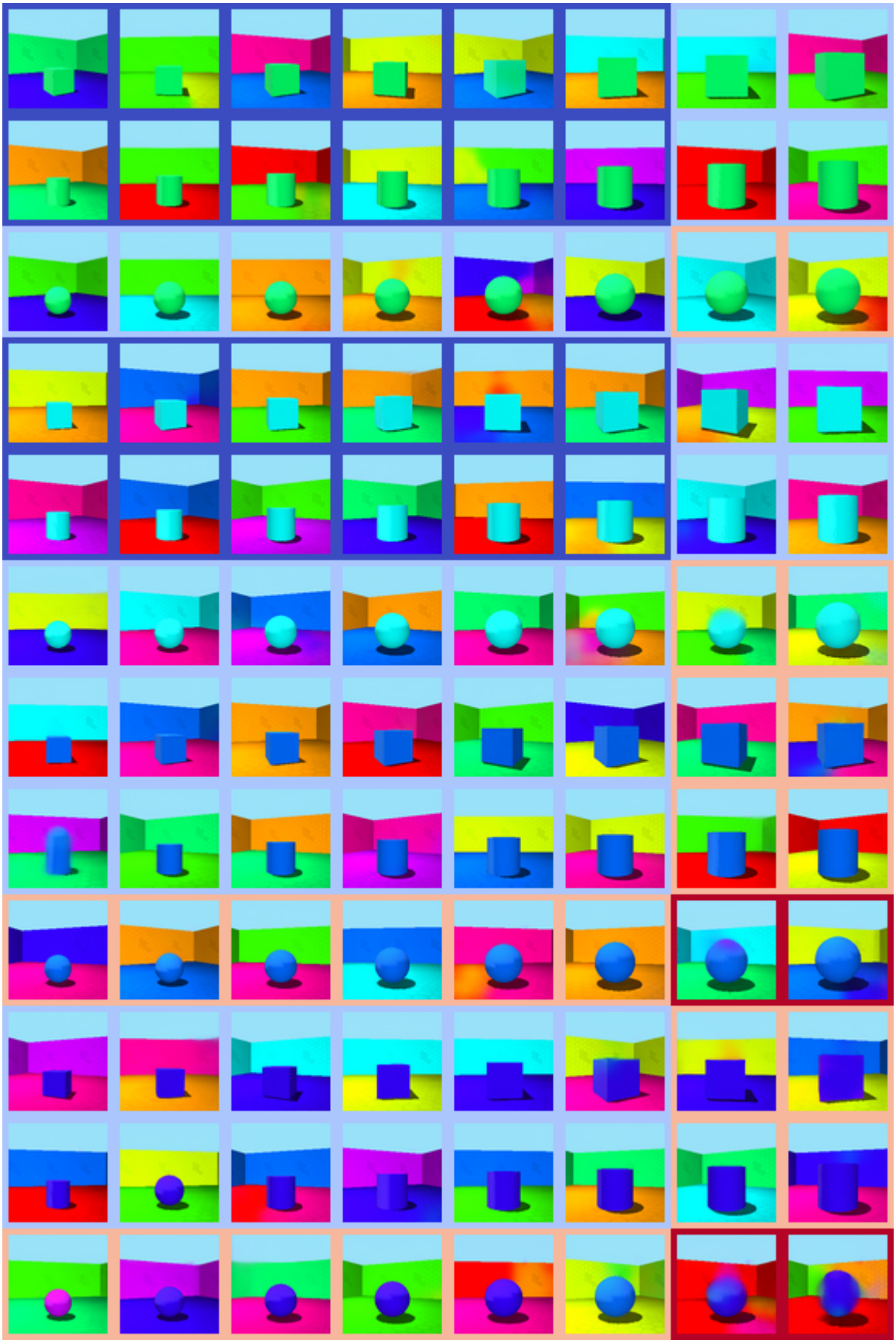


Figure 18: Shapes3D results of MaskGIT. Dark blue: all factors from Supergroup 0; light blue: one factor from Supergroup 1; pale orange: level-1 compositions; and dark red: level-2 novel compositions.

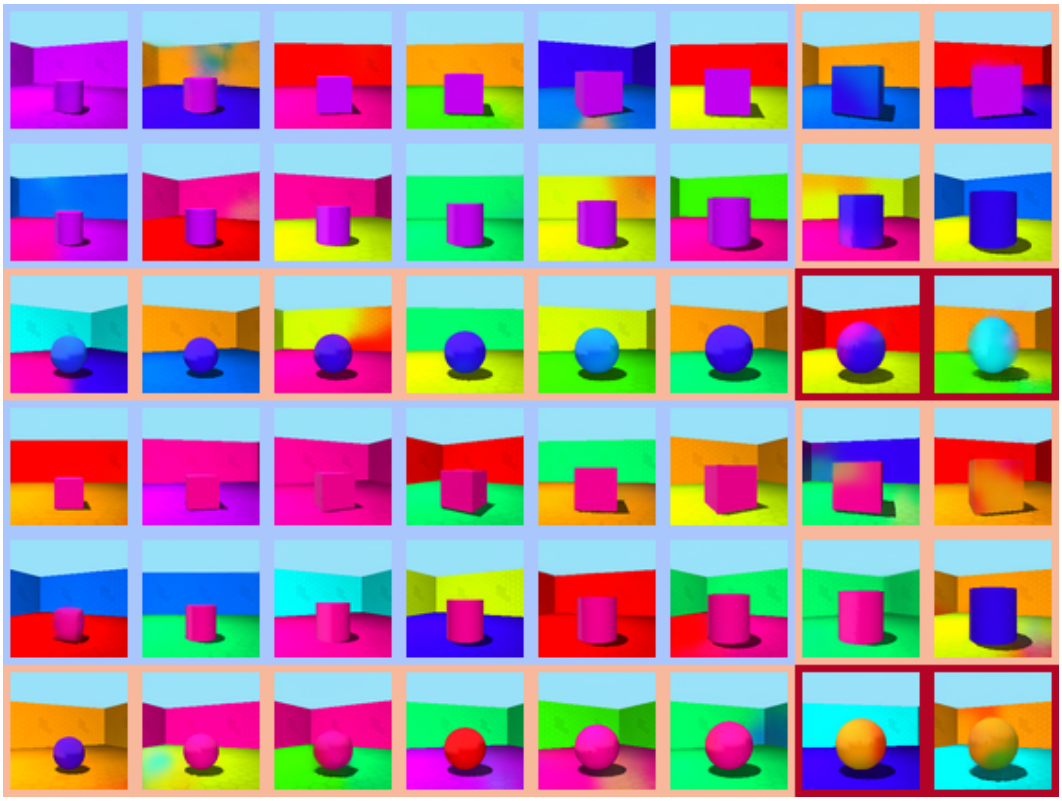


Figure 18: Shapes3D results of MaskGIT. Dark blue: all factors from Supergroup 0; light blue: one factor from Supergroup 1; pale orange: level-1 compositions; and dark red: level-2 novel compositions.

counts and batch sizes; optimizer and schedule follow Mousakhan et al. [41]. Each model observes a 7-frame context (implicitly revealing the factors discussed) and autoregressively generates 30 frames one frame at a time. No clips from $\odot\rightarrow$ or $\bigcirc\leftarrow$ appear in training.

**Evaluation protocol.** We assess compositionality via nearest-neighbor retrieval against real videos using V-JEPA2 features [3]. For each generated clip, we: (i) encode frames; (ii) spatially pool to a $4 \times 6$ grid (feature dim 1024); (iii) take the last 10 generated frames and uniformly subsample 5; and (iv) flatten to a descriptor of size $4 \times 6 \times 5 \times 1024$. The real gallery contains 100 clips per composition (600 total). Distances are based on cosine similarity; we take $k = 1$ nearest neighbors and report

$$\mathrm{CRA}_k(c^\star) = \frac{1}{k} \sum_{i=1}^{k} \mathbf{1}\{\, c(\mathrm{NN}_i) = c^\star \,\},$$

averaged over generated clips targeting composition $c^\star$. With a balanced 6-class gallery, the chance is $1/6 \approx 0.167$.

We additionally report Fréchet Video Distance (FVD) between generated and real clips, computed separately on the seen and unseen splits (Table 13).

# I Language

To test whether our findings on continuous representations playing a role in compositional performance extend beyond the visual domain, we constructed a controlled card arithmetic dataset taking inspiration from the Points24 dataset [14]. Each instance requires forming an arithmetic expression from four playing cards from a standard deck of 52 cards, such that all cards are used exactly once to reach a specified target value. To map cards to numerical values, the Ace was mapped to 1, the




Figure 19: Qualitative Results on CelebA

Table 12: Nearest-neighbor compositional retrieval metric (CRA) comparing generated videos to real videos. Factors are time of day: Day $\odot$, Night $\bigcirc$; directions: Left $\leftarrow$, Straight $\uparrow$, Right $\rightarrow$. Rows show a generated novel split evaluated against *real train* and *real novel* classes. The Hit@1 (correct target novel class) is bolded; **black bold** indicates better results.

| | | Real train | | | | Real novel | |
|---|---|---|---|---|---|---|---|
| Gen. novel | Model | $\odot\leftarrow$ | $\odot\uparrow$ | $\bigcirc\rightarrow$ | $\bigcirc\uparrow$ | $\odot\rightarrow$ | $\bigcirc\leftarrow$ |
| $\odot\rightarrow$ | DiT | 0.190 | 0.190 | 0.060 | 0.060 | **0.470** | 0.030 |
| | MaskGIT | 0.380 | 0.260 | 0.110 | 0.050 | **0.180** | 0.020 |
| $\bigcirc\leftarrow$ | DiT | 0.130 | 0.020 | 0.340 | 0.040 | 0.040 | **0.430** |
| | MaskGIT | 0.150 | 0.030 | 0.560 | 0.100 | 0.020 | **0.140** |

Table 13: FVD on seen and unseen splits on CoVLA.

| Method | FVD (Seen) ↓ | FVD (Unseen) ↓ |
|---|---|---|
| DiT | **730.30** | **2257.51** |
| MG | 2320.95 | 3197.74 |

number cards (2-10) retained their face value, and face cards were mapped to the next integers (Jack = 11, Queen = 12, King = 13).

The data was split into compositional sets based on rules that the model had to follow while performing the arithmetic task: (i) The model could be asked to formulate an equation from the cards that satisfies a target value of either 12 or 24, (ii) cards belonging to red suits (Hearts, Diamonds) can either retain their original value (same as the cards belonging to the black suit (Clubs, Spades)), or have their value doubled. Training examples to the model contained only single-rule instances, while the test set required applying both rules simultaneously.

We use Llama-3.2 [16] and treat its reasoning mechanism as an analogue of the objective in our visual experiments. We compare the same base model under two reasoning strategies: (i) normal chain-of-thought (CoT) [61] which is the standard textual step-by-step reasoning, and (ii) COntinuous-Chain-Of-Thought (COCONUT) [24], where intermediate reasoning states are represented in a continuous latent space as "thoughts" rather than purely symbolic text.

Both models were trained on identical data and evaluated on the same compositional test split. We report the success rate, defined as the proportion of trials in which a programmatic verifier confirms that the output applies the correct rules, uses each card exactly once, and evaluates to the target. Averaged over three seeds on 500 compositional test instances, the success rate with continuous reasoning was $12.33 \pm 1.3\%$, higher than the standard CoT baseline at $4.87 \pm 0.9\%$. Although the absolute numbers are low due to the difficulty of the task, this preliminary result suggests that continuous reasoning states may benefit compositional generalization on this task.

This small-scale experiment serves two purposes: First, it provides preliminary evidence that the advantage of continuous representations observed in visual generative models may extend to a language reasoning task. Second, it highlights an early but promising signal that *continuous reasoning for language* is a fruitful research direction. While the present setup is limited and minimal, it lays the groundwork for future studies that explore richer tasks, larger models, and more principled strategies for continuous reasoning.

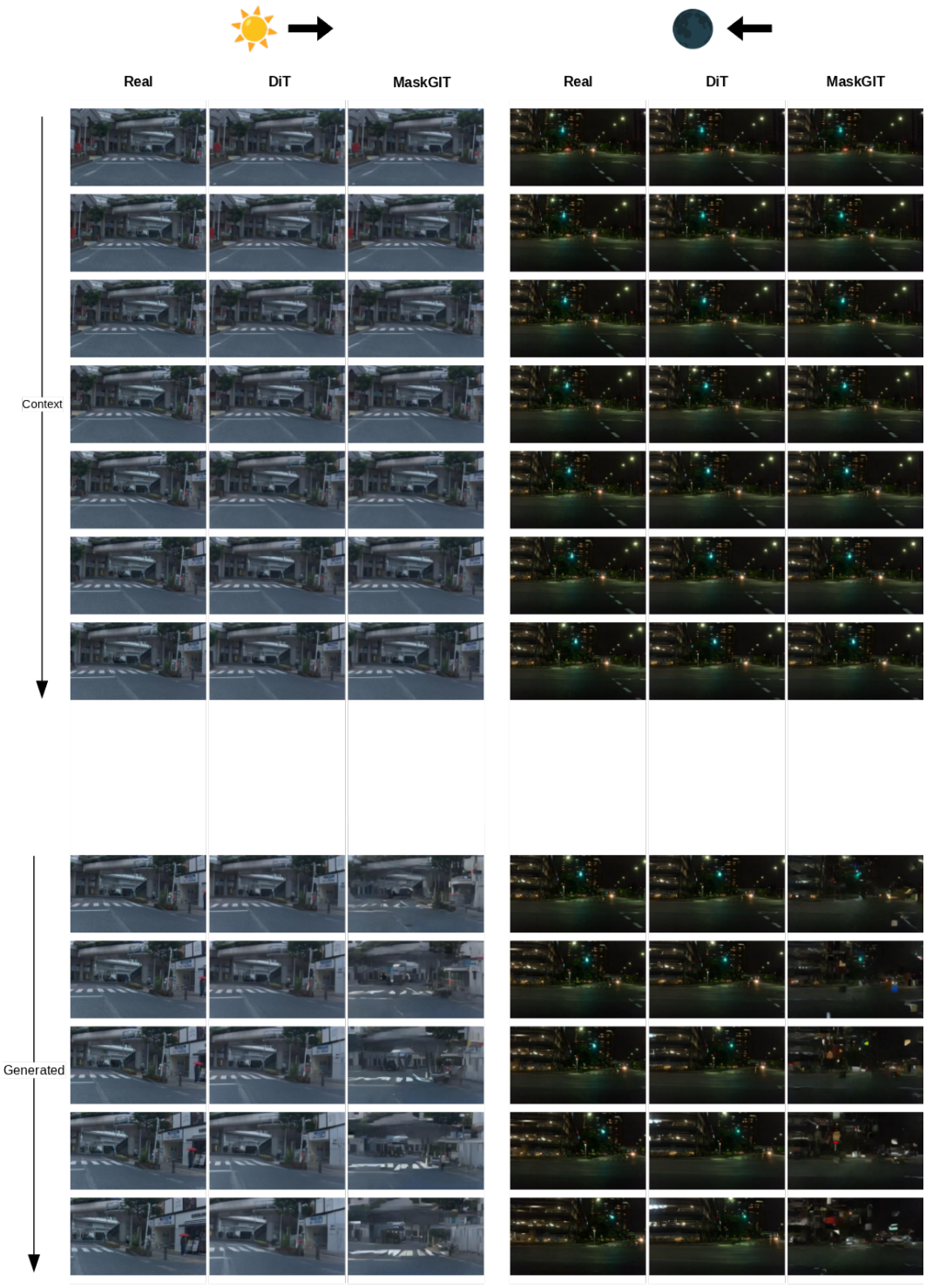



Figure 20: Qualitative results with Orbis on CoVLA. Two novel compositions are shown, $\odot\rightarrow$ and $\bigcirc\leftarrow$, each conditioned on a 7-frame initial context and predicting 5 frames ahead. Orbis-DiT follows the target compositions, whereas Orbis-MaskGIT often struggles.

**System Prompt** ($v_0^{in}$)

**[Task Description]**
You are an expert in the card arithmetic game. You are observing four cards: each card must be used exactly once to form an arithmetic expression. Number cards take their face value (Ace = 1, 2–10 as given). The suits are abbreviated as Hearts = H, Diamonds = D, Clubs = C, Spades = S. Face cards are valued as J = 11, Q = 12, K = 13. In addition, if a card belongs to a red suit (Hearts H or Diamonds D), its value is doubled. Your goal is to output a formula that evaluates to the target number, which is either 12 or 24. The allowed operators are +, -, *, parentheses, and =.

**[Input]**
Cards: [2H, 3S, QC, 4D]
Target: 24

**[Output]**
Your response should be a valid JSON file in the following format:

```
{
  "cards": [x, y, z, w],
  "values_after_rules": [a, b, c, d],
  "target": T,
  "formula": "an equation that equals T"
}
```

---

**Model output** ($v_t^{out}$)

```
{
  "cards": ["2H", "3S", "QC", "4D"],
  "values_after_rules": [4, 3, 12, 8],
  "target": 24,
  "formula": "(12+8)+(4*3)"
}
```

---

**Verifier Output** ($v_t^{ver}$)
This trial is unsuccessful because the proposed formula does not evaluate to 24.

Figure 21: Example system prompt, unsuccessful model output, and verifier feedback for our language task with compositional rules: target restriction and red-suit value doubling.

**System Prompt** ($v_0^{in}$)

**[Task Description]**
You are an expert in the card arithmetic game. You are observing four cards: each card must be used exactly once to form an arithmetic expression. Number cards take their face value (Ace = 1, 2–10 as given). The suits are abbreviated as Hearts = H, Diamonds = D, Clubs = C, Spades = S. Face cards are valued as J = 11, Q = 12, K = 13. Your goal is to output a formula that evaluates to the target number, which is either 12 or 24. The allowed operators are +, -, *, parentheses, and =.

**[Input]**
Cards: [AD, JH, 5D, 9S]
Target: 24

**[Output]**
Your response should be a valid JSON file in the following format:

```
{
  "cards": [x, y, z, w],
  "values_after_rules": [a, b, c, d],
  "target": T,
  "formula": "an equation that equals T"
}
```

---

**Model output** ($v_t^{out}$)

```
{
  "cards": ["AD", "JH", "5D", "9S"],
  "values_after_rules": [1, 11, 5, 9],
  "target": 24,
  "formula": "(11+9)+(5-1)"
}
```

---

**Verifier Output** ($v_t^{ver}$)
This trial is successful because the proposed formula evaluates to 24.

Figure 22: Example system prompt, successful model output, and verifier feedback for a single-rule training instance of our language task: target restriction without red-suit value doubling.

## 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: The abstract summarizes our work for the reader to get a gist of the paper. The Introduction explains the motivation of the paper and lists down the main contributions and findings.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: We have a separate paragraph in the Discussion to put forward the limitaions of our work.

   Guidelines:

   - The answer [N/A] 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: Any formula used in the paper is referenced and performance is numerically evaluated either in the main text or appendix. In addition, we have a separate section that provides theoretical validation for our empirical results.

Guidelines:

- The answer [N/A] 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: We give an overview of experimental setups in the main text. In the appendix, we provide more detail and information required to accurately reproduce our results.

Guidelines:

- The answer [N/A] 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 provide all information required to reproduce our experimental results. Additionally, we plan to release code and models for experiments in this paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all information required to reproduce our experimental setup and results in the main text and appendix.

   Guidelines:

   - The answer [N/A] 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: We show statistical significance for some experiments. For others, we prove generality by testing across multiple different architectures and training setups. This ensures that our findings transfer to other scenarios within observed variances.

   Guidelines:

   - The answer [N/A] 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: We provide per-experiment compute budget in the appendix.

   Guidelines:

   - The answer [N/A] 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: We do not violate any code of ethics.

   Guidelines:

   - The answer [N/A] 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: We provide a brief discussion on broader impact in the last section.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper is an academic analysis for what matters in compositional learning. All experiments were conducted in controlled settings with example real-world validations. The findings of this paper pose no risk to the community and only help in progressing the field further.

    Guidelines:

    - The answer [N/A] 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: We clearly cite any work including code repositories used in our paper.

    Guidelines:

    - The answer [N/A] 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: We plan to release code upon acceptance and it will be extensively documented so that readers can reproduce results. In addition, we already provide relevant information in our main text and appendix.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper does not perform any experiments with crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper performs no experiments involving crowdsourcing or human subjects and thus does not need any IRB approval.

    Guidelines:

- The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We describe usage of LLMs for our LLM experiment in the main text and appendix.

    Guidelines:

    - The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.