PARACHUTING PARROT: A DATASET FOR EVALUATING VISUAL COMPOSITIONAL GENERALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Proprietary text-to-image models have become remarkably powerful at rendering complex compositional scenes. Did they overcome the longstanding challenge of compositional generalization, and if so, how did they do it? Current datasets that allow to systematically evaluate compositional generalization lack the complexity, diversity and scale required to train image generation models of remotely similar quality to answer this question. Here, we introduce *Parachuting parrot*, a vision dataset and benchmark for evaluating compositional generalization consisting of over 2.5 million images with corresponding structured scene descriptions. *Parachuting parrot* consists of complex scenes defined by a rich scene description language, rendered using large-scale text-to-image models and validated through a scalable vision-language model pipeline. Alongside it, we define systematic splits to evaluate compositional generalization over scenes with varying compositional complexity. We train a text-to-image latent diffusion model from scratch on *Parachuting parrot*, achieving generalization to complex scene compositions with multiple entities, entity-attribute binding and relational binding in a fully controlled experimental setting.

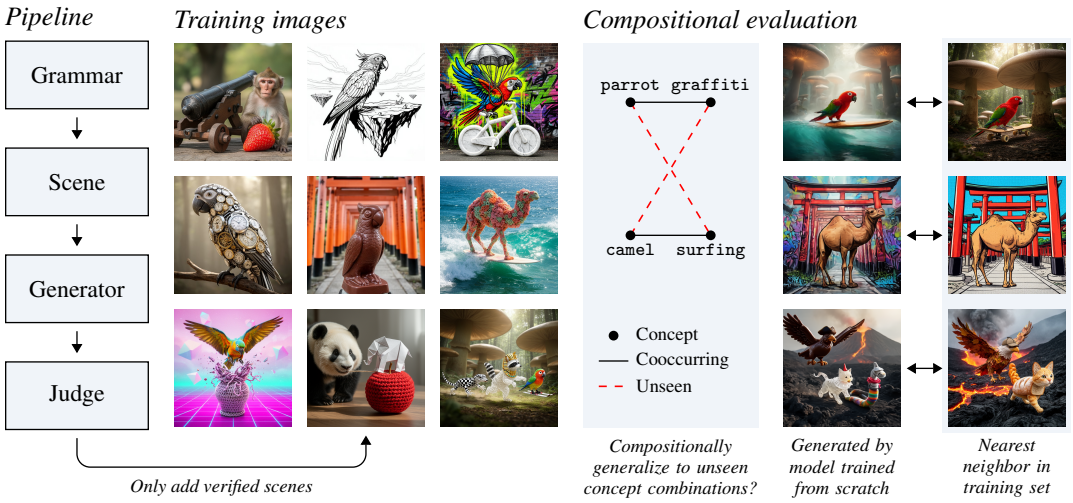


Figure 1: **Overview**. We develop a pipeline to generate the *Parachuting parrot* dataset for evaluating compositional generalization with systematically held-out concept combinations using a text-to-image generation model and vision-language model judge. We then train a text-to-image diffusion model from scratch finding strong visual compositional generalization beyond the training data.

1 INTRODUCTION

Despite possibly never having entertained the thought of *a parachuting parrot*, most humans have the remarkable ability to understand and imagine it. Whether neural networks could similarly obtain

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

the ability to compose familiar concepts into novel combinations has been a longstanding question (Fodor & Pylyshyn, 1988). While models historically struggled to generalize compositionally (Lake et al., 2016; Geirhos et al., 2020), large-scale models like text-to-image generation models can now flexibly compose familiar concepts and for instance generate images of parachuting parrots and more as shown in Figure 1. Have they resolved the historic challenge of compositional generalization?

Unfortunately, we lack control over or knowledge of the training data used to create existing capable, large-scale models, and therefore fundamental questions about the nature of compositional generalization remain unresolved. To what extent are these models genuinely generalizing beyond their training corpus (Mayilvahanan et al., 2025)? And when compositional generalization does occur, what enables it? Is it primarily a consequence of data scale (Redhardt et al., 2025), model architecture (Kamb & Ganguli, 2025), or specific training objectives (Brady et al., 2026)?

To answer these questions, we need a dataset that is both large and diverse enough to train capable models, while offering precise control over the composition of concepts in each image to construct systematic splits for evaluating compositional generalization. Large, real-world image datasets (Russakovsky et al., 2015; Krishna et al., 2017; Schuhmann et al., 2022) do not provide this level of control: the concepts present in each image are difficult to capture through post-hoc annotations and obtaining systematic evaluation splits is hard as a result. Consequently, prior works have largely relied on synthetic images (Johnson et al., 2017; Bordes et al., 2023), in which the underlying concepts are known and controllable by construction. However, these benchmarks lack the visual complexity and scale needed to train capable vision models and test compositional generalization to realistic and complex scene compositions.

Here, we attempt to fill this gap by creating a large and realistic dataset for evaluating compositional generalization consisting of over 2.5 million images with corresponding structured scene descriptions. We use the capable models we seek to understand to systematically construct this dataset.

We summarize our **main contributions** in the following

- We develop a dataset generation pipeline based on a rich scene description language to render and validate complex compositional scenes using a text-to-image generation model and vision-language model judge.
- We use this pipeline to create the *Parachuting parrot* dataset consisting of 2.5 million diverse images with corresponding structured scene descriptions and systematic splits to evaluate compositional generalization
- We train text-to-image latent diffusion models from scratch on *Parachuting parrot*, demonstrating strong compositional generalization in a fully controllable training environment that allows to systematically study the ingredients required for compositionality to emerge.

2 Dataset

In the following section we will introduce the *Parachuting parrot* dataset. We will begin by defining our scene description language in Section 2.1 that allows us to specify complex scene compositions. In Section 2.2, we detail our dataset generation pipeline which uses the scene description language to generate diverse compositional images through text-to-image generation models and validates their faithfulness to the underlying scene description using a vision-language model judge. Finally, in Section 2.3, we specify how we systematically obtain a compositional data split and subdivide the dataset into blocks of increasing complexity. The result is a comprehensive dataset which pairs each rendered image with structured and natural scene descriptions and allows to systematically evaluate visual compositional generalization at varying levels of complexity.

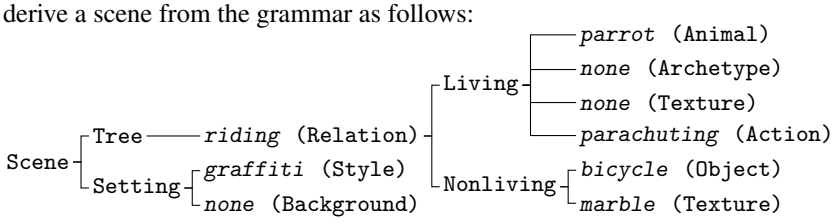
2.1 Scene description language

We use the context-free grammar shown in Figure 2 to programmatically generate structured scene descriptions. Each scene can combine a variable number of entities, bind attributes and relational roles to them, and set global scene properties such as style and background. Structured scene de-

```
      Scene → Tree Setting
    Setting → Style Background
       Tree → Relation | Conjunction | Entity
     Entity → Living | Nonliving
     Living → Animal Archetype Texture Action
  Nonliving → Object Texture
Conjunction → and(Entity, Entity) | and(Entity, Conjunction)
      Chain → Relation | Entity
LivingChain → Relation | Living
   Relation → chasing(Living, LivingChain) | pulling(Living, Chain)
            | riding(Living, Nonliving) | shattering(Living, Nonliving)
            | composed_of_many(Living, Nonliving)
```

Figure 2: **Scene description grammar**. We design a context-free grammar to describe complex compositional scenes. Corresponding terminal production rules are shown in appendix Table 3.

scriptions can be deterministically mapped into a natural language description. For example, we can derive a scene from the grammar as follows:





The grammar is carefully designed to make only well-defined scene descriptions grammatical and reject combinations that are hard to clearly conceive, such as "parachuting pirate apples". Nonliving entities can bind to a texture (like `made of fire`) and each living entity can additionally bind to an archetype (like `pirate`), and an action (like `parachuting`). An arbitrary number of entities can be combined into a scene either through a simple `and` conjunction or through interactions (like `chasing`) which bind a relational role to each entity. In addition, each scene can have a background (like `floating islands`) and a style (like `lineart`). Importantly, object and scene attributes (archetype, texture, action, style, background) are optional and can be omitted from a scene description (in the parse tree above denoted by the keyword `none`).

We construct production rules to be right-branching to ensure an unambiguous, one-to-one-mapping to an English sentence, see Appendix A.3. This allows us to systematically create a large number of compositional scene descriptions of varying complexity. For a detailed description of the scene grammar and the underlying design considerations as well as a full list of all terminal production rules, please see Appendix A.

2.2 Dataset generation pipeline

We use the scene description language detailed above to programmatically generate images of varying complexity through the dataset generation pipeline illustrated in Figure 3. For each image we generate, we sample a structured scene description from our grammar and deterministically map it into a canonical natural language description. We then either directly use the canonical description as the prompt or instruct a large language model to expand the short description into a more detailed, visually descriptive prompt that is fed to a text-to-image generation model, adding non-semantic information such as lighting, camera angle and positioning. Specifically, we use Gemini Flash-Lite 3.1 (Google DeepMind, 2026a) for prompt expansion and the open-weight FLUX.2-dev (Black Forest Labs, 2025) latent diffusion model for text-to-image generation. We verify that each image we generate faithfully represents the structured scene description it is based on by validating it with a vision-language model judge; here we use Gemini 3.1 Flash-Lite for simpler scenes and Gemini 3.7 Flash (Google DeepMind, 2026b) for more complex scenes. The judge is instructed to reconstruct the structured scene description corresponding to the visual scene, based on the scene grammar given as a reference. We normalize both the ground truth structured scene description and the one provided by the judge and verify that they match exactly. We only add an image and

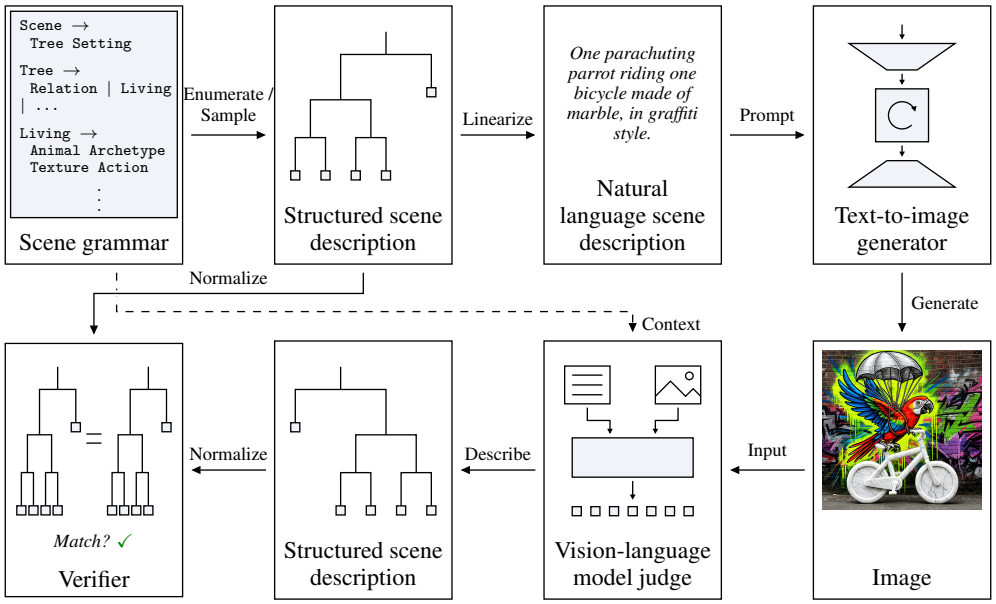


Figure 3: **Dataset generation pipeline**. We use our scene description language to sample scenes and render them via a text-to-image generation model. We then verify whether images faithfully depict their scene descriptions by asking a vision-language model judge to produce the structured scene descriptions from the images using the scene grammar. We only add an image to our dataset if the scene description produced by the judge matches the ground truth after normalization.

its corresponding scene description to our dataset if this cycle consistency verification succeeds. Otherwise, we reject the data point and retry the whole process.

To cross-validate our pipeline, two independent human annotators manually verified 430 scenes of varying complexity, containing 1563 distinct concepts. In total, we find 13 unique images that were incorrectly accepted by the judge. 5 of these errors are due to ambiguity in the `pulling` relation, i.e. the image does not make the patient agent distinction sufficiently clear. The remaining 8 errors are due to partially applied attributes (e.g. the graffiti style not applied throughout the whole image) and crucially do not leak any compositions that we hold out for evaluation.

As both the text-to-image generation model and the vision-language model judge are limited in their abilities to correctly create or respectively describe very complex scenes, the rejection rate eventually limits the complexity of scenes we can practically generate. We will detail the concrete dataset composition in the next section. As model capabilities improve, our pipeline can be used to further increase the complexity of the dataset.

2.3 Dataset

We use our scene description language and image generation pipeline to construct a dataset of over 2.5 million images with corresponding structured scene descriptions. We taxonomize our dataset into blocks of increasing scene complexity as shown in Table 1 and define evaluation splits in each block to distinguish generalization along different types of composition.

2.3.1 Dataset taxonomy

For evaluation, we divide our dataset into blocks, as shown in Table 1, to distinguish different types of scene composition and study the effect of increasing scene complexity. For example, we can compose a complex scene either by combining multiple entities, by binding multiple attributes to an entity or by modifying the background and style of the image. We further distinguish complex scenes in Block F according to the number of concepts they combine as exemplified in Figure 8. For the first blocks (A-E) which contain a manageable number of unique scene descriptions, we exhaustively enumerate all possible scene descriptions to construct the training and compositional evaluation sets and generate multiple image variants per scene description for the training scenes. For Block F, the


Block A


Block B


Block C


Block D


Block E


Block F

Block	Multiple	Attributes	Setting	Relations	Example scene
A	✓				An octopus and a parrot.
B		✓			A parachuting parrot made of fire.
C			✓		A parrot on floating islands, in lineart style.
D	✓			✓	A parrot composed of watches.
E		✓	✓		A parrot made of chocolate, among torii gates.
F	✓	✓	✓	✓	A checkered lizard chasing one pharaoh raccoon made of glass noodles that is chasing one skiing parrot, among giant mushrooms.

Table 1: **Dataset taxonomy**. Overview of how we divide the dataset into blocks according to which aspect of scene complexity it varies. Corresponding to each block we show an example scene.

combinatorial space grows too large to exhaustively enumerate all scene descriptions. We therefore create a subblock, *F-2-fewer-terminals*, with a reduced terminal set that we can fully enumerate. For the remaining complex scenes in Block F, we subsample scenes for the training and evaluation set.

We run our dataset generation pipeline for each block and report the total number of accepted images and first pass acceptance rates in Table 2. Including rejected images, we generated a total 5.85 million images of which we accept 2.5 million images to be included in our dataset. The acceptance rate drops noticeably for complex scenes in Block F. The most complex scenes we can still feasibly generate with FLUX.2-dev as the generator and Gemini 3.7 Flash as the judge have two global scene attributes and contain three entities, each of which bind to two attributes and have relational roles. For example, our dataset contains the scene (Table 2, right) `One wizard otter made of lego bricks pulling one knight pig made of wax that is pulling one mummy monkey made of sand, on a volcano in low-poly style`. For a small fraction of scenes that were never accepted by our pipeline after many repeated attempts, we manually verify the generated scenes (less than 0.1% of released images).

Table 2: **Dataset composition.** *Left* Image counts and first pass acceptance rate using Flux.2-dev as the generator and Gemini 3.7 as the judge. *Right* Example for a scene in Block F-3-complex.

Block	Images	Accept rate
A	544 400	0.951
B	202 200	0.934
C	201 500	0.942
D	220 392	0.724
E	142 500	0.911
F-2-fewer-terminals	659 997	0.805
F-2-all-terminals	349 937	0.811
F-3-simple	138 005	0.640
F-3-complex	89 864	0.452



`One wizard otter made of lego bricks pulling one knight pig made of wax that is pulling one mummy monkey made of sand, on a volcano in low-poly style.`

2.3.2 COMPOSITIONAL SPLIT

One of the central motivations for creating this dataset is to systematically evaluate compositional generalization. To this end, we need to be able to split scenes according to the particular concepts they contain. For example, we would like to understand whether a model that has been trained on images that contain "A parachuting parrot" and "A checkered lizard" would be able to generate an image of "A checkered parrot" despite never having seen it. This requires us to precisely control what enters our training distribution and what is held-out for evaluation.

Concept binding graph We formalize the problem of obtaining a compositional data split by constructing a concept binding graph $G = (V, E)$, where nodes V represent concepts (including entities), and edges E indicate that a scene binds an entity to a concept or contains two co-occurring entities. For example, for the scene `A parachuting parrot`, we connect the `parrot` node to the `parachuting` node as exemplified in Figure 4. For asymmetric relations like `A parrot chasing a lizard`, we create agent and patient nodes so role-specific pairings can be held out independently. We can then obtain a compositional evaluation split by partitioning the edges $E = E_{\text{train}} \cup E_{\text{test}}$ ($E_{\text{train}} \cap E_{\text{test}} = \emptyset$) into a training subset E_{train} and a held-out test subset E_{test}, yielding the training concept binding graph $G_{\text{train}} = (V, E_{\text{train}})$. A scene s is included in the training set if and only if all of its constituent concept pairs are in the training split, i.e., its edge set $E(s) \subseteq E_{\text{train}}$. Intuitively, an edge in E_{test} ensures that there is no training scene that binds the two corresponding concepts.

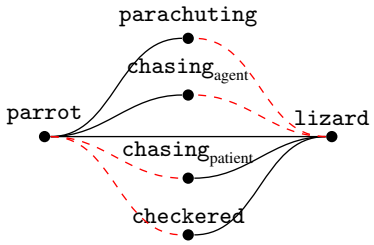


Figure 4: **Concept binding graph**. Simplified example based on the training scenes `A parachuting parrot`, `A checkered lizard`, and `A parrot chasing a lizard`. Solid edges denote concept bindings present in the training scenes, dashed red edges denote compositions that are valid under the grammar but unseen during training (e.g., `A checkered parrot`).

Compositional exposure We would like to evaluate how the number of distinct entity bindings in which a particular concept participates affects the ability to compositionally generalize with that concept. For example, in the scenes "A parachuting parrot" and "A parrot and a pineapple" the concept "parrot" is bound to two distinct other concepts, "parachuting" and "pineapple", whereas both concepts "parachuting" and "pineapple" only appear in one binding each. We therefore formalize *compositional exposure* for a particular concept as the degree of its corresponding node within the training concept binding graph G_{train}. In our training set, we systematically control the compositional exposure for certain concepts in order to evaluate how exposure affects generalization.

The final dataset supports evaluating multiple aspects of visual compositional generalization – entity-attribute binding, multiple entity combination, relational binding and global scene attribute composition – **in a single training run**.

3 EXPERIMENTS

We now train a text-to-image latent diffusion transformer from scratch on our dataset and systematically evaluate its ability to generalize compositionally.

3.1 TEXT-TO-IMAGE DIFFUSION MODEL TRAINING AND EVALUATION SETUP

The specific model we consider consists of three components: a convolutional variational autoencoder (VAE) image tokenizer with 83.7M parameters (Rombach et al., 2022), a transformer-based text encoder with 102M parameters, and a dual-stream diffusion transformer (DiT) with 560M parameters (Labs et al., 2025). We train all components from scratch in order to avoid any type of leakage of existing knowledge from pretraining data into our evaluation of compositional generalization. The VAE is first trained using the ELBO objective, while the DiT and text encoder are

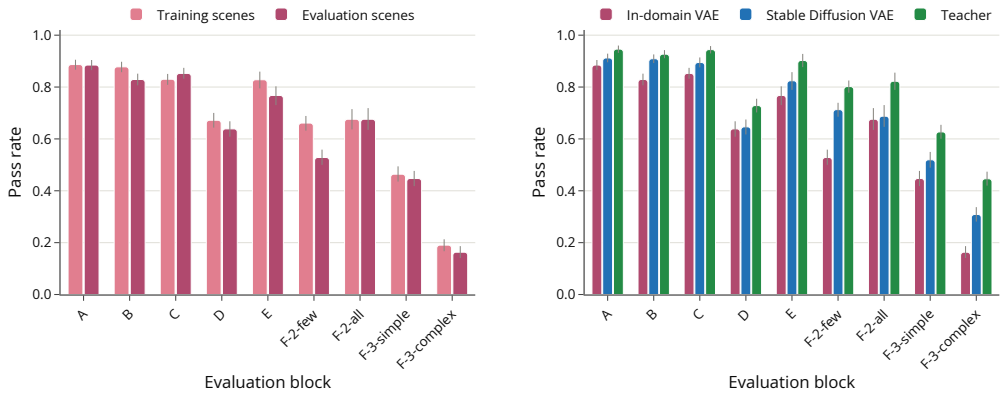


Figure 5: **Models trained from scratch can generalize compositionally.** ***Left.*** Pass rate for images generated by a text-to-image latent diffusion model trained from scratch on *Parachuting parrot* both for training scenes and evaluation scenes held-out from training. ***Right.*** Pass rate on novel scene compositions for models trained from scratch either using a VAE pretrained on *Parachuting parrot* or using the Stable Diffusion VAE (Rombach et al., 2022) in comparison to the pass rate achieved by the teacher model used to generate the dataset (Flux.2-dev; Black Forest Labs, 2025).

trained jointly on frozen VAE latents using flow matching (Lipman et al., 2023). Additional architectural and training details are provided in Appendix E.

During evaluation, we verify that generated images faithfully correspond to the provided scene descriptions using the same vision-language model judge used in our dataset construction pipeline described in Section 2.2.

3.2 Models trained on Parachuting Parrot can generalize compositionally

We begin by evaluating the ability of our trained models to compose scenes of varying complexity according to the evaluation blocks introduced in Section 2.3.1. In the left panel of Figure 5, we compare the pass rate across training and evaluation scenes. We find that our model is able to successfully compose complex scenes and notably generalize to novel scene compositions that contain concept combinations that have never been observed during training. In the right panel of Figure 5, we further compare the from-scratch trained models to Flux.2-dev – the teacher model used to generate the dataset – on the evaluation split with novel scene compositions. We see similar performance between our models trained on *Parachuting parrot* and the teacher across the different evaluation blocks. A large part of the difference between models trained on *Parachuting parrot* and the teacher is alleviated by using a stronger, pretrained VAE (Rombach et al., 2022) although in this case we do not control the pretraining distribution of the VAE.

We show three example evaluation scenes generated by our model trained from scratch on *Parachuting parrot* and contextualize them with the respective nearest neighbor training images using SigLIP 2 (Tschannen et al., 2025) embeddings in Figure 6. This qualitatively shows that the model reuses elements observed during training to compose the novel evaluation scenes.

3.3 Generalization requires sufficient training coverage

Finally, we take a closer look at the circumstances in which our model compositionally generalizes and where it fails. We begin by analyzing how compositional exposure, as defined in Section 2.3.2, affects generalization. During the construction of our training set, we specifically controlled the compositional exposure for a small number of selected concepts. In the left panel of Figure 7, we show that while our model learns to generate the precise scenes encountered during training that contain these concepts regardless of the extent of their compositional exposure, generalization to evaluation scenes noticeably degrades for concepts with a lower compositional exposure. Similarly, we find that our model fails to extrapolate to scenes with more entities than encountered during

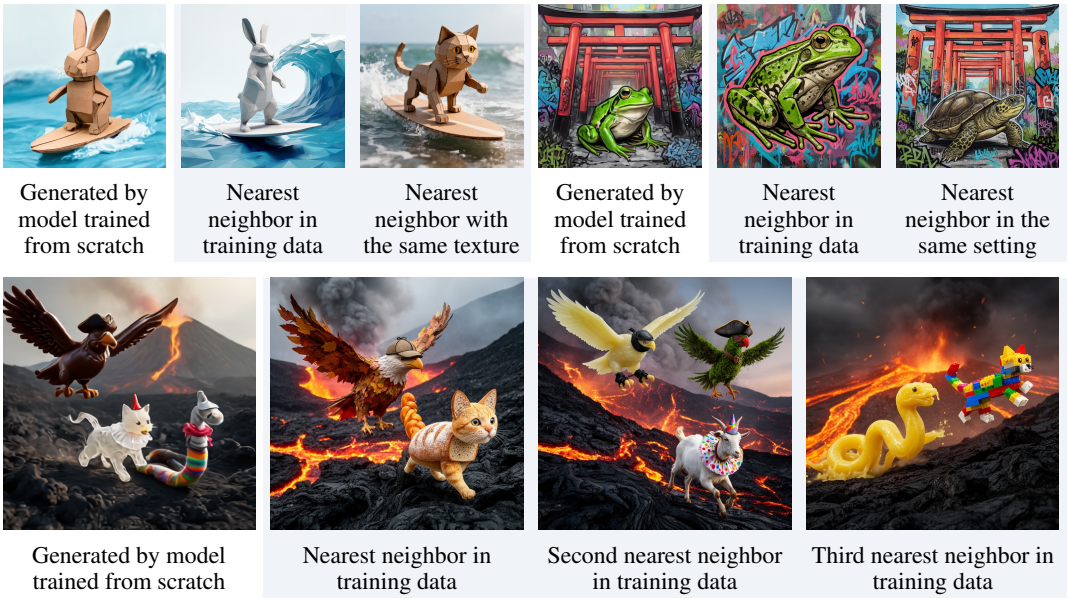


Figure 6: **Example evaluation images** Comparison of held-out scene compositions generated by our model trained from scratch to their closest neighbors in the training set based on SigLip 2 embedding distance.

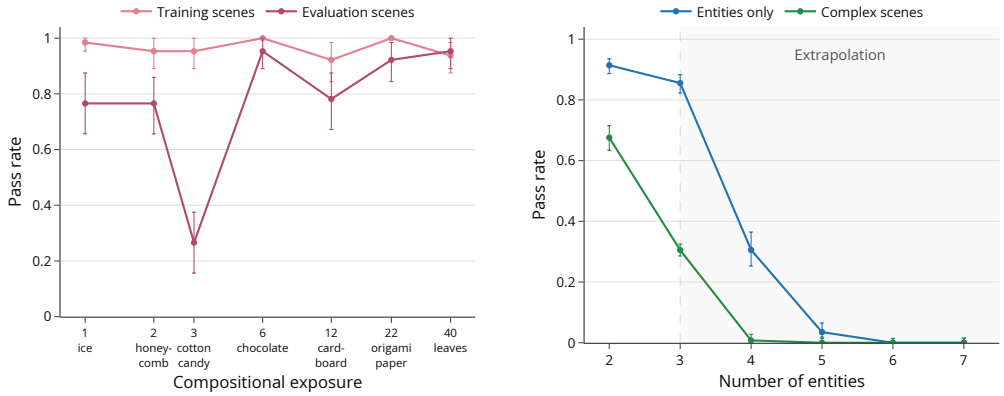


Figure 7: **Generalization requires sufficient training coverage.** ***Left.*** Pass rate on training and evaluation scenes as a function of the compositional exposure during training for selected concepts. ***Right.*** Pass rate on evaluation scenes as a function of the number of entities. During training, our model encounters at most three distinct entities in a scene. The blue line denotes scenes with multiple entities but no attributes, whereas the green line includes multiple attributes and relations.

training, shown in the right panel of Figure 7. Taken together these findings reveal that visual compositional generalization requires coverage both in terms of compositional exposure and scene complexity. It remains an open question under what circumstances models can extrapolate to scenes with more entities than observed during training.

4 Related Work

Evaluating compositional generalization with synthetic data. A large body of theoretical (Wiedemer et al., 2023; Lachapelle et al., 2023; Wiedemer et al., 2024a; Brady et al., 2026) and empirical work (Montero et al., 2021; 2022; Schott et al., 2022; Farid et al., 2025; Uselis et al., 2025;

2026; Wang et al., 2025; Assouel et al., 2026; Camposampiero et al., 2025) leverages synthetic image datasets to evaluate compositional generalization. While popular synthetic datasets like dSprites (Matthey et al., 2017), CLEVR (Johnson et al., 2017), and PUG (Bordes et al., 2023) offer controlled compositional evaluation through precise knowledge of the concepts present in each image, they lack visual complexity and diversity. In particular, visual composition is limited to simple object-attribute binding on exchangeable backgrounds, missing the more complex part-whole hierarchies and relational binding between a variable number of entities offered by our dataset.

Evaluating compositional generalization with real-world data. Prior work that attempts to evaluate compositional generalization with real-world data uses heuristic post-hoc labeling of concept combinations in existing training datasets, either to curate test sets of unseen compositions (Ma et al., 2023; Wiedemer et al., 2024b) or to create compositional training/evaluation splits to train models from scratch (Mayilvahanan et al., 2024; 2025; Kempf et al., 2025). However, because the underlying data-generating process is unknown, these compositional splits are approximate, and experiments in terms of compositional diversity and complexity remain ultimately constrained by the source dataset.

Evaluating compositional abilities in pretrained models. Several works introduce benchmarks for evaluating compositional abilities of pretrained models. Abbasi et al. (2024); Wu et al. (2024); Huang et al. (2025); Kamath et al. (2025) test pretrained text-to-image models on prompts of varying compositional complexity, while analogous vision-language model benchmarks assess understanding of images containing different forms of concept composition (Hsieh et al., 2023; Yuksekgonul et al., 2023; Thrush et al., 2022; Zhao et al., 2022). While these benchmarks provide useful insights into models' abilities to compose concepts, they cannot measure compositional generalization since they are constructed independently of a model's pretraining corpus.

Compositional generalization in diffusion models. Qualitative evidence that diffusion models can generalize compositionally dates back to early studies (Ramesh et al., 2022; Nichol et al., 2021; Rombach et al., 2022), which demonstrated their ability to generate combinations of concepts from curated prompts that were unlikely to have appeared in the training data. Subsequent work has investigated their compositional generalization more rigorously via theoretical analyses (Favero et al., 2025; Kamb & Ganguli, 2025) and controlled, small-scale empirical studies (Okawa et al., 2023; Liang et al., 2024; Farid et al., 2025). However, principled empirical evidence of compositional generalization in diffusion models in large-scale settings has thus far been lacking.

5 Discussion

Despite vision models achieving remarkable capabilities at increasing scale, efforts to understand their ability to generalize compositionally have been limited to simple datasets. With *Parachuting parrot*, we sought to close this gap by combining the data diversity and scale needed to train capable models while maintaining precise knowledge and control over the visual concepts they encounter. We show that training text-to-image latent diffusion transformers from scratch on our dataset can lead to strong compositional generalization and allows us to systematically analyze compositional abilities. We hope *Parachuting parrot* will enable further studies across a broader range of models and ultimately help establish a rigorous understanding of the ingredients and mechanisms underlying visual compositional generalization.

Limitations. Our dataset generation pipeline relies on image generation and vision-language models that are themselves imperfect, limiting the complexity of the scenes that pass the verification and are added to the dataset. As models improve over time, they can be plugged into our pipeline to further extend the dataset. In our experiments, we focus on training text-to-image latent diffusion transformers. An important open question is whether compositional generalization can be similarly observed in other generative models or vision-language models (Radford et al., 2021; Beyer et al., 2024) when trained on our dataset.

AI use statement

In this work, we used generative AI tools to design or provide feedback on research methodology or experiments, implement methods, propose or refine hypotheses, support qualitative and thematic data analysis, interpret results, generate synthetic data sets, and clean and reformat datasets. We have

not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, or provide critical ingredients for proving mathematical claims. Tasks related to assisting in the writing of proofs and assistance with translation are not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures or images, suggest experimental parameters, create or edit software code, summarize or analyse existing literature, discover research topics or identify gaps, brainstorm, source/search for information, edit a research paper to improve readability, identify relevant literature, create artifacts, and format references. We have reviewed all AI-assisted work by manually verifying factual accuracy, manually retrieving primary source and cross-checking claims made about them, and testing and reviewing generated code to ensure scientific rigor and validity. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics Statement

We carefully constructed our dataset generation pipeline to minimize the likelihood of generating explicit or harmful content. In addition, we filtered images using an NSFW image detector, as described in Appendix B.2.

Reproducibility statement

Due to space limitations during submission, we provide a random subset of our full dataset in the supplementary material. To ensure the reproducibility of our work, we will open-source our dataset, the code for our dataset generation pipeline, and all our experiments upon publication. We further expand upon our scene description language in Appendix A, our dataset generation pipeline in Appendix B, our dataset composition in Appendix C, and implementation details for our experiments in Appendix E.

REFERENCES

Reza Abbasi, Mohammad Hossein Rohban, and Mahdieh Soleymani Baghshah. Deciphering the role of representation disentanglement: Investigating compositional generalization in clip models. In *European conference on computer vision*, pp. 35–50. Springer, 2024.

Rim Assouel, Pietro Astolfi, Florian Bordes, Michal Drozdzal, and Adriana Romero-Soriano. Object-centric binding in contrastive language-image pretraining. *Advances in Neural Information Processing Systems*, 38:131418–131458, 2026.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Black Forest Labs. FLUX.2 [dev], 2025. URL https://huggingface.co/black-forest-labs/FLUX.2-dev.

Florian Bordes, Shashank Shekhar, Mark Ibrahim, Diane Bouchacourt, Pascal Vincent, and Ari Morcos. Pug: Photorealistic and semantically controllable synthetic data for representation learning. *Advances in Neural Information Processing Systems*, 36:45020–45054, 2023.

Jack Brady, Bernhard Schölkopf, Thomas Kipf, Simon Buchholz, and Wieland Brendel. Is generation required for data-efficient perception? In *Forty-third International Conference on Machine Learning*, 2026.

Giacomo Camposampiero, Pietro Barbiero, Michael Hersche, Roger Wattenhofer, and Abbas Rahimi. Scalable evaluation and neural models for compositional generalization. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 7341–7401. Curran Associates, Inc., 2025. doi: 10.52202/085713-0252.

Karim Farid, Rajat Sahay, Yumna Ali Alnaggar, Simon Schrodi, Volker Fischer, Cordelia Schmid, and Thomas Brox. What drives compositional generalization? the importance of continuous training objectives in visual generative models. *arXiv preprint arXiv:2510.03075*, 2025.

Alessandro Favero, Antonio Sclocchi, Francesco Cagnetta, Pascal Frossard, and Matthieu Wyart. How compositional generalization and creativity improve as diffusion models are trained. In *Forty-second International Conference on Machine Learning*, 2025.

Jerry A. Fodor and Zenon W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1):3–71, 1988. ISSN 0010-0277. doi: https://doi.org/10.1016/0010-0277(88)90031-5.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Google DeepMind. Gemini 3.1 Flash-Lite: Model card, 2026a. URL https://deepmind.google/models/model-cards/gemini-3-1-flash-lite/.

Google DeepMind. Gemini 3.7 Flash: Model card, 2026b. URL https://deepmind.google/models/model-cards/gemini-3-7-flash/.

Cheng-Yu Hsieh, Jieyu Zhang, Zixian Ma, Aniruddha Kembhavi, and Ranjay Krishna. Sugarcrepe: Fixing hackable benchmarks for vision-language compositionality. *Advances in neural information processing systems*, 36:31096–31116, 2023.

Kaiyi Huang, Chengqi Duan, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench++: An enhanced and comprehensive benchmark for compositional text-to-image generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(5):3563–3579, 2025.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2901–2910, 2017.

Amita Kamath, Kai-Wei Chang, Ranjay Krishna, Luke Zettlemoyer, Yushi Hu, and Marjan Ghazvininejad. Geneval 2: Addressing benchmark drift in text-to-image evaluation. *arXiv preprint arXiv:2512.16853*, 2025.

Mason Kamb and Surya Ganguli. An analytic theory of creativity in convolutional diffusion models. In *Forty-second International Conference on Machine Learning*, 2025.

Elias Kempf, Simon Schrodi, Max Argus, and Thomas Brox. When and how does CLIP enable domain and compositional generalization? In *Forty-second International Conference on Machine Learning*, 2025.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

Black Forest Labs, Stephen Batifol, A. Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Muller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space. *ArXiv*, abs/2506.15742, 2025.

Sebastien Lachapelle, Divyat Mahajan, Ioannis Mitliagkas, and Simon Lacoste-Julien. Additive decoders for latent variables identification and cartesian-product extrapolation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *arXiv preprint arXiv:1604.00289*, 2016.

Qiyao Liang, Ziming Liu, Mitchell Ostrow, and Ila Fiete. How diffusion models learn to factorize and compose. *Advances in Neural Information Processing Systems*, 37:15121–15148, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Zixian Ma, Jerry Hong, Mustafa Omer Gul, Mona Gandhi, Irena Gao, and Ranjay Krishna. Crepe: Can vision-language foundation models reason compositionally? In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10910–10921. IEEE, 2023.

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

Prasanna Mayilvahanan, Thaddäus Wiedemer, Evgenia Rusak, Matthias Bethge, and Wieland Brendel. Does CLIP's generalization performance mainly stem from high train-test similarity? In *The Twelfth International Conference on Learning Representations*, 2024.

Prasanna Mayilvahanan, Roland S. Zimmermann, Thaddäus Wiedemer, Evgenia Rusak, Attila Juhos, Matthias Bethge, and Wieland Brendel. In search of forgotten domain generalization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Milton L Montero, Jeffrey S Bowers, Rui Ponte Costa, Casimir JH Ludwig, and Gaurav Malhotra. Lost in latent space: Disentangled models and the challenge of combinatorial generalisation. *arXiv preprint arXiv:2204.02283*, 2022.

Milton Llera Montero, Casimir JH Ludwig, Rui Ponte Costa, Gaurav Malhotra, and Jeffrey Bowers. The role of disentanglement in generalisation. In *International Conference on Learning Representations*, 2021.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Maya Okawa, Ekdeep S Lubana, Robert Dick, and Hidenori Tanaka. Compositional abilities emerge multiplicatively: Exploring diffusion models on a synthetic task. *Advances in Neural Information Processing Systems*, 36:50173–50195, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

Florian Redhardt, Yassir Akram, and Simon Schug. Scaling can lead to compositional generalization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Lukas Schott, Julius Von Kügelgen, Frederik Träuble, Peter Vincent Gehler, Chris Russell, Matthias Bethge, Bernhard Schölkopf, Francesco Locatello, and Wieland Brendel. Visual representation learning does not generalize strongly within the same domain. In *International Conference on Learning Representations*, 2022.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022.

Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. Winoground: Probing vision and language models for visio-linguistic compositionality. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5228–5238. IEEE, 2022.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim M. Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier H'enaff, Jeremiah Harmsen, Andreas Steiner, and Xiao-Qi Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *ArXiv*, abs/2502.14786, 2025.

Arnas Uselis, Andrea Dittadi, and Seong Joon Oh. Does data scaling lead to visual compositional generalization? In *Forty-second International Conference on Machine Learning*, 2025.

Arnas Uselis, Andrea Dittadi, and Seong Joon Oh. Compositional generalization requires linear, orthogonal representations in vision embedding models. In *Forty-third International Conference on Machine Learning*, 2026.

Yanbo Wang, Justin Dauwels, and Yilun Du. Compositional scene understanding through inverse generative modeling. In *Forty-second International Conference on Machine Learning*, 2025.

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

Thaddäus Wiedemer, Jack Brady, Alexander Panfilov, Attila Juhos, Matthias Bethge, and Wieland Brendel. Provable compositional generalization for object-centric learning. In *International Conference on Learning Representations*, volume 2024, pp. 20137–20164, 2024a.

Thaddäus Wiedemer, Yash Sharma, Ameya Prabhu, Matthias Bethge, and Wieland Brendel. Pretraining frequency predicts compositional generalization of CLIP on real-world tasks. In *NeurIPS 2024 Workshop on Compositional Learning: Perspectives, Methods, and Paths Forward*, 2024b.

Xindi Wu, Dingli Yu, Yangsibo Huang, Olga Russakovsky, and Sanjeev Arora. Conceptmix: A compositional image generation benchmark with controllable difficulty. *Advances in Neural Information Processing Systems*, 37:86004–86047, 2024.

Mert Yuksekgonul, Federico Bianchi, Pratyusha Kalluri, Dan Jurafsky, and James Zou. When and why vision-language models behave like bags-of-words, and what to do about it? In *The Eleventh International Conference on Learning Representations*, 2023.

Tiancheng Zhao, Tianqi Zhang, Mingwei Zhu, Haozhan Shen, Kyusong Lee, Xiaopeng Lu, and Jianwei Yin. Vl-checklist: Evaluating pre-trained vision-language models with objects, attributes and relations. *arXiv preprint arXiv:2207.00221*, 2022.

A SCENE DESCRIPTION LANGUAGE

A.1 FULL GRAMMAR

In total we have 50 animals, 50 objects, 10 archetypes, 18 textures, 4 actions, 12 styles, and 7 backgrounds. The individual terminal production rules are listed in Table 3.

Table 3: Grammar terminal symbols. Attributes (Archetype, Texture, Action, Style, Background) are optional and can evaluate to `none`.

Nonterminal	Terminals	Count
Animal	man, woman, cat, dog, monkey, horse, bear, elephant, eagle, lizard, cow, lion, fish, snake, shark, chicken, duck, tiger, zebra, giraffe, deer, wolf, penguin, pig, sheep, squirrel, spider, frog, fox, dolphin, whale, turtle, crocodile, owl, rabbit, parrot, flamingo, octopus, crab, panda, raccoon, kangaroo, hippo, hedgehog, swan, goat, camel, otter, moose, sloth	50
Object	watch, scissors, chair, ball, anvil, umbrella, cannon, vase, bottle, car, truck, bus, bicycle, airplane, windmill, lighthouse, church, castle, hospital, anchor, bed, computer, phone, tv, couch, shovel, shelf, suitcase, teapot, apple, banana, pizza, bread, egg, watermelon, pineapple, strawberry, carrot, tomato, bell, bag, cup, globe, dumbbell, drum, train, boat, helicopter, motorcycle, barrel	50
Archetype	wizard, knight, pirate, cowboy, pharaoh, scuba diver, ninja, clown, detective, mummy, `none`	10
Texture	marble, fire, sand, knitted yarn, cotton candy, gelatin, ice, coral, chocolate, lego bricks, wax, leaves, origami paper, honeycomb, cardboard, glass noodles, mossy, checkered, `none`	18
Action	skiing, surfing, parachuting, skateboarding, `none`	4
Style	graffiti, cyberpunk, watercolor, pixel-art, pop-art, low-poly, oil-painting, lineart, cartoon, mosaic, ukiyo-e, vaporwave, `none`	12
Background	space, floating islands, chess board, labyrinth, volcano, giant mushrooms, torii gates, `none`	7

A.2 ON RIGHT BRANCHING

Every description has to be matched to exactly one scene; otherwise, we implicitly contaminate other relations by giving a scene a double meaning. We choose right-branching trees as the most complex structure that still allows unambiguous scenes. There are multiple ways to describe a scene through its constraints. One can use a set of constraints, e.g., for a raccoon made of glass noodles chasing a parrot, we would get `chasing(raccoon, parrot)`; parrot in scene; raccoon in scene; raccoon made of glass noodles. This is often how compositional benchmarks grade images (Wu et al., 2024), making it easy to check individual constraints. However, this becomes an issue once we chain relations. Consider a lizard chasing one raccoon that is chasing one parrot. An image could pass the relational constraints by showing `chasing(lizard, raccoon`$_1$`)`; `chasing(raccoon`$_2$`, parrot)` instead of a chained relation. To avoid this issue, we represent the scene as a tree, `chasing(lizard, chasing(raccoon, parrot))`, which clearly defines that it is the same raccoon.

General scenes, however, are even more complex than right-branching trees; they are graphs. Right-branching trees cannot express shared subjects (a dog riding a cat while chasing a parrot), shared objects (two dogs chasing one cat), or cycles (a dog chasing a cat that is chasing the same dog). In scenes like the first, it becomes unclear how the relations apply. Is the cat also chasing the parrot? Probably yes; one can imagine the combined entity chasing after the parrot with both of them trying to reach for it. On the other hand, it could be that only the dog on top is reaching for the parrot while the cat is looking in a completely different direction. These scenes become increasingly difficult to judge, and one would need to describe every edge in the scene graph to properly disambiguate them.

We also exclude conjunction-relation mixtures, where a tree contains both relations and conjunctions, e.g., `chasing(dog, and(cat, mouse))`. The same sentence, a dog chasing a cat and a mouse, could also be formulated as `and(chasing(dog, cat), chasing(dog, mouse))`, so the

language description alone does not determine a unique tree. Conjunctions over relations do not resolve this in general either: a dog chasing a cat and a cat chasing a mouse leaves open whether it is the same cat, so we exclude the whole family. Lastly, we exclude left-branching trees, which again do not yield a unique tree per scene: `chasing(chasing(dog, cat), mouse)` can produce the same image as `chasing(dog, chasing(cat, mouse))`. Given these issues, we restrict ourselves to right-branching trees with the convention that the predicate attaches the action to the first entity in the right subtree, e.g., `chasing(dog, pulling(cat, mouse))` is a dog chasing a cat, and that cat is pulling a mouse. This structure gives every tree a unique sentence, with nesting rendered as a relative clause, and lets the judge describe an image as a flat sequence of entities and relations that parses back into the same tree.

A.3 Linearization

Given a tree, we construct English sentences as follows. To construct entities, actions and archetypes attach before the living or non-living entity, while textures attach after it as "made of", e.g., *one parachuting knight dog made of fire*. The two adjective textures, mossy and checkered, attach at the very beginning of the entity, e.g., *one checkered parachuting lizard*. Every entity is introduced with "one". For conjunction trees, we first sort the entities, those with more attributes first and ties in alphabetical order, and join them with commas, e.g., `and(lizard, and(raccoon, parrot))` becomes *one lizard, one parrot, and one raccoon*. We sort to obtain a single linearization, since all *and*-trees over the same entities describe the same scene, and we put attributed entities first because otherwise it becomes unclear where attributes attach: in *one cat and one dog made of chocolate*, is the cat made of chocolate as well? Relations are joined through relative clauses, e.g., the tree `chasing(lizard, chasing(raccoon, parrot))` becomes *one lizard chasing one raccoon that is chasing one parrot*. Shattering is rendered as "shattering X into pieces", and composed of takes a plural object, e.g., *one parrot composed of watches*. Lastly, we attach scene attributes, background then style, at the end of the sentence, e.g., *one dog chasing one cat, on a volcano in watercolor style*, and prefix every caption with "An image with". We similarly normalize our trees, flattening and sorting conjunctions so that each scene corresponds to exactly one tree. The same normalization is applied to the tree recovered from the judge's description, so grading compares two canonical trees.

A.4 Sampling

We sample as follows. First, we fix the desired held-out fraction and assign each sample its label. This is necessary because as trees grow, a larger fraction of the underlying distribution is held out, since every additional edge can make a tree held out: a tree with 10 edges is held out as soon as any edge within it is held out. Next, for each sample and independently of its label, we draw a skeleton: we sample the number of entities and scene attributes uniformly, then, starting at the root, decide with equal probability whether the tree is a conjunction or a relation, and lastly, for each entity, sample whether it is living or non-living, the number of its attributes, and which attribute categories it carries. The skeleton must pass the complexity rejection filter. For example, if a block contains scenes with two entities and should additionally contain attributes, our sampler draws the number of attributes per entity individually; if neither entity receives an attribute, we reject the skeleton and draw a new one. Similarly, no textures are allowed inside composed-of, as it is unclear where the texture applies: for a parrot composed of shelves made of chocolate, the image might still show the outline of a parrot made of chocolate, potentially contaminating the pairing parrot–chocolate. As we cannot reliably tell, we exclude such scenes. Once we have a passing skeleton, we sample terminals for it until the tree carries its assigned label; the skeleton itself is never redrawn. We need to fix the skeleton first because sampling full trees and rejecting those with the wrong label would bias held-out samples to be more complex than in-distribution samples, as a tree with fewer edges has a higher chance of being in-distribution.

B Dataset generation pipeline

B.1 Generation

Starting from the linearized sentence, we upsample some of the prompts, i.e., we expand them into more detailed descriptions using an LLM. Upsampling has two main benefits. A) It leads to more

expressive scenes, enriching the complexity of the dataset towards real-world data while preserving the requested concepts such as attributes and entities. We additionally run word-matching checks to detect whether the LLM introduces concepts from our grammar that were not requested in the scene. This serves as an additional layer to reduce contamination. B) The FLUX.2 prompting guide recommends upsampling for better generation results. During testing, we observed that the diffusion model generally has an easier time rendering a complex scene from an upsampled prompt, potentially because the prompt makes it more explicit how the concepts should look together. Furthermore, some grammar combinations have natural meanings, like pharaoh dog, where the model might render the dog breed, or ninja horse, where the model could render a ninja on a horse rather than a horse in ninja clothing. Here, upsampling helps, as we can provide guidelines for how such prompts should be upsampled, e.g., a dog wearing a pharaoh mask. The upsampled prompts are used solely as a generation aid and not during training.

Not upsampling has a potential advantage as well: the resulting images stay closer to the canonical caption the model is trained on. Initially, we were not sure whether a model could learn a distribution of this complexity from a dataset of ImageNet scale; reference models that showed generalization (Nichol et al., 2021) were trained on significantly more images. Hence, we expected that keeping some images without upsampling might allow the model to learn the basic concepts more easily. The dataset therefore contains a mixture, with at most 50% of images per block not upsampled and a decreasing share in the more complex blocks.

For upsampling, we provide the LLM with the linearized sentence, a JSON version of the scene description that maps each concept to its entity, e.g., `horse:{archetype:ninja}`, and guidelines on how to upsample the prompt. We then provide the prompt to FLUX.2 together with a JSON description of the scene, following the FLUX.2 guidelines for increased prompt adherence. When no style is specified in the scene description, we additionally request photorealistic generation. During initial testing, the JSON description helped FLUX.2 bind attributes to the right entities more consistently, and requesting photorealism prevented FLUX.2 from introducing extra styles, reducing the rejection rate of the judge. Finally, we use FLUX.2-dev in NVFP4 quantization with 28 inference steps and a guidance scale of 4 to generate images at 512×512 pixels.

B.2 Judge

The judge receives each image together with a guide on how to describe it and how to construct right-branching trees from the terminals. Importantly, it does not receive the generation prompt or the intended bindings. It then outputs a JSON description, which we parse into a canonical tree. If the judge finds all requested concepts and their bindings without finding additional ones, we admit the image into the dataset; otherwise, we retry. There is one difference between the judge used to admit images into the dataset and the judge used for benchmark grading. For the dataset, we reject images containing unrequested relations, e.g., a dog chasing a cat for the scene *one dog and one cat*, as we do not want this relation to be implicitly present during training if that combination is held out. During benchmark grading, however, the request for a dog and a cat is fulfilled when a dog chases a cat in the image. Besides judging whether the image is semantically correct, we run the NSFW filter `Falconsai/nsfw_image_detection`, rejecting images containing humans with a score above 0.85 and regenerating replacements. To check for systematically recognizable intellectual property (IP) within our dataset, we additionally ran an IP checker on a sample of 2,125 images and manually reviewed the flags, finding occasional recognizable elements, e.g., a car brand symbol. We did not run the IP checker over the entire dataset, so residual recognizable likenesses may remain.

C Dataset

C.1 Blocks

The block categories can be split into smaller blocks along the categories axis, with the number after the block letter signifying the concept count on the compositional axis of interest. The entity block A is split into A1 (one entity), A2 (two entities), and A3 (three entities), introducing multiple entities together at increasing complexity. The attribute block B is split into B1 (entity + one attribute) and B2 (entity + two attributes), differentiating whether the model can use multiple attributes on the same entity. The scene block C is split into C1 (one scene attribute) and C2 (two scene attributes),

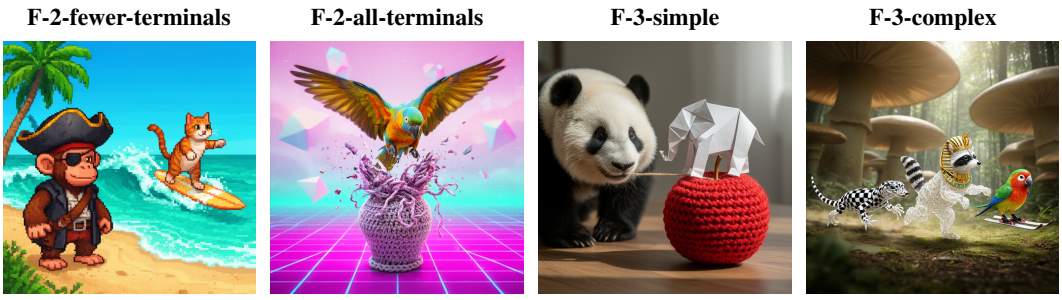


Figure 8: **Complex scenes.** We further subdivide Block F consisting of complex scenes with multiple entities, attributes, settings and relations. The number (F-2 or F-3) indicates the number of entities in a scene.

testing again whether we can combine them and how much harder it is to apply multiple at the same time. The relational block D is split into D2 (two-entity relation) and D3 (three-entity relation chain, which is sampled). In block E, we then evaluate what happens if you combine testing axes on one entity with E1 (entity attribute + scene attribute). Block F then combines the attributes and scene attributes from E and the multiple entities from A and D. F-2-fewer-terminals-and tests this without introducing interaction between the entities on a reduced terminal set to still allow for enumeration. F-2-fewer-terminals-relation then additionally adds a relation between the two entities. F-2-all-terminals uses the full terminal set. F-3 escalates this to three entities; we differentiate blocks with and vs. relation as they showed large differences in difficulty, as well as "simple" and "complex" blocks where complex blocks have more concepts, giving F-3-and-simple, F-3-and-complex, F-3-rel-simple, and F-3-rel-complex.

Compositional novelty In order to quantify how different a held-out scene is from those encountered during training, we define compositional novelty of a scene s compared to the training corpus as the number of unique edges it contains that are not in E_{train}, i.e., $|E(s) \setminus E_{\text{train}}|$. Among the set of held-out scene compositions, we ensure a variety of novelty to evaluate to what extent compositional novelty is associated with successful generalization.

Scene complexity Formally, we describe a scene's complexity by k, the number of entities in the scene; j, the maximal number of attributes per entity (texture, archetype, action); $J = \sum_i j_i$, the total number of attributes in the scene; and s, the number of scene attributes (background, style). We define the binding load as $L = J + s$. The detailed breakdown of the dataset is shown in Table 4.

Holdout We hold out concepts in three ways, balancing a sufficiently large held-out set against maximizing the number of training images in the enumerated blocks. For attributes and scene attributes, we hold out 10% of each attribute's entity pairings. For entity pairs, we hold out 5% of all pairings. We hold out fewer entity pairs than attribute pairings because a held-out entity pair restricts the entire scene: holding out *parachuting parrot* results in one constraint on a parrot in the scene, while holding out the pair parrot–rabbit constrains all other entities in the scene not to be a rabbit once a single parrot is part of the scene. Lastly, for relations, we independently hold out a fraction f of the entity pairings for each role, agent and patient. A triplet (e_1, r, e_2) is in-distribution only if both role-specific pairings are retained. The held-out fraction of triplets is therefore $F = 1-(1-f)^2$. Solving for a desired held-out fraction F gives $f = 1 - \sqrt{1-F}$. We adopt $F = 0.1$ to match the attribute holdout, i.e., $f \approx 0.05$. Note that the held-out fraction of scenes grows significantly faster for relations than for attributes: n attributes on an entity impose n constraints, while n relations impose $2n$ constraints. For example, a lizard chasing a raccoon that is chasing a parrot requires the pairings lizard–chasing (agent), raccoon–chasing (patient), raccoon–chasing (agent) and parrot–

BASIC SCENE FAMILIES



A1–A3, 544k images
One cannon, one monkey, and one strawberry.



B1, 100k images
One parachuting parrot.



B2, 102k images
One pirate horse made of gelatin.



C1–C2, 202k images
One cat, on a volcano in mosaic style.

COMPLEX COMPOSITIONS



D2–D3, 220k images
One man chasing one eagle that is riding one pizza.



E1, 143k images
One skateboarding bear, in pop-art style.



F-2-fewer-terminals, 660k images
One skiing woman and one wizard dog.



F-2-all-terminals, 350k images
One fox chasing one wizard penguin, in watercolor style.

MULTI-ENTITY SCENES



F-3-and-simple, 102k images
One ninja horse made of origami paper, one watch made of origami paper, and one pizza, in ukiyo-e style.



F-3-and-complex, 74k images
One skiing parrot made of sand, one wizard lion made of gelatin, and one parrot, on a volcano in watercolor style.



F-3-rel-simple, 36k images
One panda pulling one elephant made of origami paper that is riding one apple made of knitted yarn.



F-3-rel-complex, 16k images
One detective sloth chasing one skiing scuba diver turtle that is composed of pineapples, in a labyrinth in pop-art style.

Figure 9: **Additional examples per dataset block.** One example scene per main block which can be further broken down as shown for B1 for a single attribute and B2 for multiple, complementing the parrot examples in Table 1. For a full breakdown of the block concepts see Table 4.

Table 4: **Detailed dataset taxonomy and composition**. Train is the number of images the model trains on. $|\mathcal{T}|$ is the theoretical number of distinct scenes the block's grammar can produce. $|\mathcal{D}|$ is the number of images in the dataset: the smallest integer multiple of $|\mathcal{T}|$ that reaches 100,000 samples where the grammar is enumerable, structure-first sampled otherwise; D3, F-3 blocks are best effort. The extrapolation scenes are prompt-only test sets (no images) of 15,000 prompts per label. $L = J + s$ is the binding load.

Block	Description	k	j	s	Constraint	Train	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{D}\rvert$
A – entities only								
A1	Single entity (training only)	1	0	0	—	100,000	100	100,000
A2	Two entities	2	0	0	—	91,238	5,050	101,000
A3	Three entities	3	0	0	—	281,352	171,700	343,400
B – attributes only								
B1	Ent. + 1 attr.	1	1	0	—	58,558	2,500	100,000
B2	Ent. + 2 attrs.	1	2	0	—	49,929	14,600	102,200
C – scene attributes only								
C1	Ent. + 1 scene attr.	1	0	1	—	86,099	1,900	100,700
C2	Ent. + 2 scene attrs.	1	0	2	—	77,555	8,400	100,800
D – relations								
D2	Two ent. + rel.	2	0	0	—	83,618	15,000	105,000
D3	Three ent. chain	3	0	0	—	98,768	1.5×10^{6}	115,392
E – attribute + scene attributes								
E1	Ent. + attr. + scene	1	1	1	$j\geq1 \wedge s\geq1$	75,189	47,500	142,500
F – full								
F-2-fewer-terminals-and	Reduced terminals, AND	2	≤1	≤1	$j\geq1 \vee s\geq1$	150,617	35,585	320,265
F-2-fewer-terminals-relation	Reduced terminals, rel.	2	≤1	≤1	$j\geq1 \vee s\geq1$	146,826	113,700	339,732
F-2-all-terminals	Full vocabulary	2	≤1	≤1	$j\geq1 \vee s\geq1$	259,735	2.7×10^{8}	349,937
F-3-and-simple	Three ent., AND, low load	3	≤2	≤2	$1\leq L\leq4$	67,640	2.2×10^{11}	102,174
F-3-and-complex	Three ent., AND, high load	3	≤2	≤2	$L\geq5$	39,098	8.8×10^{13}	73,684
F-3-rel-simple	Three ent., rel., low load	3	≤2	≤2	$1\leq L\leq4$	25,094	3.0×10^{12}	35,831
F-3-rel-complex	Three ent., rel., high load	3	≤2	≤2	$L\geq5$	9,044	1.9×10^{15}	16,180
Total scene corpus						1,700,360		2,548,795
F – extrapolation (prompt-only)								
F-4-and-none	Bare entity list	4–5	0	0	$L=0$	0	9.6×10^{7}	30,000
F-4-and-simple	AND, low load	4–5	≤2	≤2	$1\leq L\leq6$	0	2.3×10^{17}	45,000
F-4	Full grammar	4–5	≤3	≤2	—	0	6.7×10^{26}	45,000
F-5-and-none	Bare entity list	6–7	0	0	$L=0$	0	2.6×10^{10}	30,000
F-5-and-simple	AND, low load	6–7	≤2	≤2	$1\leq L\leq8$	0	1.2×10^{23}	45,000
F-5	Full grammar	6–7	≤3	≤2	—	0	7.3×10^{36}	45,000

chasing (patient) to be retained. In practice, we hold out 10% of two-entity relational scenes and subsample relational scenes with three entities, as the combination space they span is sufficiently large.

Multiple variants per scene Some of the early blocks contain only few unique scenes. For these, we generate multiple images of the same canonical scene by varying the seed of the diffusion model, the upsampled prompt, or both. This ensures that there is enough data to learn the basic concepts.

C.2 Compositional splits

We partition test samples into bins to evaluate compositional generalization across blocks, compositional exposure, compositional novelty, concept categories and complexity beyond the training envelope, all within one training run. Within each evaluation axis, we define main bins. For the main generalization analysis, these are our blocks. For compositional exposure, these are the textures with different numbers of unique entity pairings. For complexity beyond the training envelope, we construct additional blocks, separating scenes that only require additional complexity from scenes that require both additional complexity and novel combinations. For compositional novelty, we bin scenes by how many held-out concept combinations they contain. Similarly, we bin by category, e.g., texture or background, to compare how models perform across categories. Each main bin is allocated a fixed number of prompts, 512 in our case. This number can be increased to reduce the uncertainty of the metrics or decreased due to compute constraints.

Inside the main bins, the scores can often be split further by evaluating on the individual blocks of our benchmark, e.g., B1 or C1. For compositional exposure, it is interesting to compare, for example, only single entity-attribute combinations (see Figure 7), because additional concepts the model has to get right could confound the analysis or add noise. Hence, for compositional exposure, we additionally bin by block, giving a block $\times$ texture matrix. For compositional novelty, it is important not to confound novelty with the difficulty of the block a scene comes from. Early blocks like B1 cannot reach high novelty because they contain only a single pair that can be held out. Hence, we additionally bin novelty by block. For our analysis (Table 7), we use F-3, as it contains sufficiently many concepts to compare multiple levels of novelty at equal complexity. Similarly, how well a category like texture performs compared to a category such as relations depends on the blocks in which it can appear; hence, we again bin by category inside a block. For our evaluation (Table 8), we use F-2, because it is the first block containing all categories. All sub-bins that differentiate by block contain 64 samples. This number can likewise be increased to reduce uncertainty at the cost of additional compute.

To minimize the number of samples needed, we fill each sub-bin with all available unique prompts first, and then duplicate prompts evenly where not enough unique ones exist. For example, the in-distribution exposure bin for B1 contains exactly one unique prompt, as B1 only contains single entity-attribute pairs and the model sees only one unique pair during training. Duplicate prompts are generated with different seeds.

D Additional results

D.1 Compositional generalization and exposure

Complementary to Figure 5, Table 5 reports the pass rate per block. A small gap between in-distribution and held-out scenes persists even when using the Stable Diffusion VAE (0.74 vs. 0.71 on average, compared to 0.67 vs. 0.63 for the in-domain VAE), suggesting that the diffusion model itself is at least partially responsible for the remaining failures to generalize. The teacher, which was not trained on our split, shows no gap (0.79 for both). Similarly, the compositional exposure results in Table 6 confirm the trend from Figure 7: increasing compositional exposure helps the model generalize to held-out combinations, also when using the Stable Diffusion VAE. This suggests that it is the diffusion model that requires sufficient exposure. With the Stable Diffusion VAE, the dip at cotton candy disappears, supporting our hypothesis that it stems from the in-domain VAE failing to render this texture sharply enough.

Table 5: **Pass rate per block.** Pass rate on in-distribution and held-out scenes for models trained from scratch with the in-domain VAE or the Stable Diffusion VAE, and for the teacher model (FLUX.2-dev).

	A2	A3	B1	B2	C1	C2	D2	D3	E1	F-2-fewer-terminals-and	F-2-fewer-terminals-relation	F-2-all-terminals	F-3-and-simple	F-3-and-complex	F-3-rel-simple	F-3-rel-complex	Mean
In-domain VAE																	
ID	0.91	0.86	0.93	0.82	0.94	0.72	0.71	0.63	0.83	0.79	0.53	0.68	0.60	0.26	0.33	0.12	0.67
Held-out	0.91	0.86	0.89	0.77	0.94	0.77	0.67	0.61	0.77	0.64	0.41	0.68	0.59	0.23	0.31	0.10	0.63
Stable Diffusion VAE																	
ID	0.94	0.91	0.97	0.85	0.95	0.87	0.71	0.66	0.87	0.88	0.68	0.72	0.68	0.42	0.45	0.24	0.74
Held-out	0.95	0.88	0.94	0.88	0.95	0.84	0.66	0.63	0.82	0.81	0.61	0.69	0.65	0.39	0.39	0.22	0.71
Teacher (FLUX.2-dev)																	
ID	0.99	0.92	0.96	0.93	0.96	0.92	0.83	0.61	0.92	0.93	0.69	0.80	0.81	0.63	0.50	0.28	0.79
Held-out	0.96	0.93	0.94	0.91	0.96	0.93	0.85	0.61	0.90	0.91	0.70	0.82	0.81	0.65	0.44	0.25	0.79

Table 6: **Pass rate by compositional exposure.** Pass rate on in-distribution and held-out scenes for textures with increasing compositional exposure in Block B1.

		In-domain VAE		Stable Diffusion VAE	
Texture	Exposure	ID	Held-out	ID	Held-out
Ice	1	0.98	0.77	0.98	0.67
Honeycomb	2	0.95	0.77	1.00	0.56
Cotton candy	3	0.95	0.27	1.00	0.72
Chocolate	6	1.00	0.95	1.00	0.88
Cardboard	12	0.92	0.78	1.00	0.92
Origami paper	22	1.00	0.92	1.00	0.91
Leaves	40	0.94	0.95	0.94	0.91

Table 7: Pass rate on evaluation scenes by compositional novelty on all F-3 blocks.

Scene novelty	**Pass rate**
0	0.31
1	0.29
2	0.30
3	0.25
4	0.20

Table 8: Per-edge accuracy of concept categories on F-2-all-terminals for seen and held-out edges.

Category	**ID**	**Held-out**
Texture	0.65	0.58
Action	0.89	0.86
Archetype	0.86	0.83
Background	0.94	0.94
Style	0.72	0.73
Relation	0.60	0.55

D.2 Compositional novelty

We analyze compositional novelty to understand whether it is harder to generalize to scenes with multiple held-out concept combinations. We fix one block and compare different levels of novelty within that block at equal total concept count, thereby avoiding confounding the pass rate with scene complexity. Table 7 shows that as compositional novelty increases, the pass rate drops, indicating that the model has a harder time generalizing when a scene contains more held-out concept combinations.

D.3 Concept categories

An interesting question is which concept categories are hard for the model to generalize. To this end, we compare the per-edge accuracy of seen and held-out edges across categories within a block that contains all categories. Within F-2-all-terminals, entity attributes and relations are harder to generalize than scene attributes, potentially because scene attributes interact less with an individual entity and can be identified from the entire scene.

E Implementation Details

For our diffusion model, we roughly follow the FLUX architecture with 8 double-stream and 16 single-stream blocks and 16 attention heads of dimension 64, giving a total width of 1024. For the VAE, we omit the LPIPS loss, which requires another pretrained model, as well as the GAN loss. We train the VAE for 300k steps at batch size 64 using AdamW with a learning rate of 10^{-4} and a weight decay of 0.01. We train the diffusion model for 300k steps at 256×256 resolution and batch size 128 using flow matching with logit-normal$(0, 1)$ timestep sampling, a timestep shift of 1 and a learning rate of 10^{-4}. We then train for another 40k steps at 512×512 with a timestep shift of 2 and a learning rate of $3 \cdot 10^{-5}$. Throughout, we keep an exponential moving average (EMA) of the weights with a decay of 0.9999. Training the VAE takes 30 hours on a single B200, and training the diffusion model takes 18 hours at 256×256 and 7 hours at 512×512.