Do Diffusion Models Learn to Generalize Basic Visual Skills?

Amish Sethi **Boya Zeng** **Wenhao Chai** **Zhuang Liu**
Princeton University

Abstract

Diffusion models generate high-quality images, but how well they generalize is hard to measure. Current evaluations rely on human preference and prompt-following benchmarks for text-to-image models and on distribution-level metrics such as Fréchet Inception Distance (FID) and Inception Score (IS) for class-conditional models, but these cannot tell whether models learn basic visual skills or largely match training patterns. To address this gap, we introduce the Visual Generative Lab (VGL), a controlled experimental framework for studying how diffusion models learn and generalize basic visual skills, including size, position, rotation, count, color, shape, and their compositions. By generating synthetic training data for each skill from explicit rules, VGL allows precise measurement of generalization with task-specific, rule-based metrics. Training models from scratch on these data reveals three patterns in this controlled setting: (1) with our default design, extrapolation is limited: size and position degrade within a few pixels of the training range, count reaches only about 50% accuracy on out-of-range counts, and rotation extrapolation accuracy varies widely across seeds; (2) for compositional generalization, the number of distinct basic visual skill combinations matters more than dataset size; (3) skills mostly fail independently, except position. Code and data-generation scripts: `https://github.com/AmishSethi/visual-generative-lab`.

1 Introduction

The rise of diffusion models [22, 61, 43, 15] has demonstrated strong capabilities in generating high-quality images, from class-conditional synthesis [48, 23] to text-to-image generation [42, 53, 55, 52]. These advances are driven by architectural ideas such as the U-Net [54] and transformer-based designs [48, 3], and by conditioning mechanisms like cross-attention [53] and adaptive normalization [49, 48]. Recent work explores synthetic datasets [29, 69] for controlled empirical analysis [44, 46] and theoretical analysis [20] of diffusion behavior, yet comprehensive controlled studies of how diffusion models learn and compose basic visual skills remain limited.

Existing evaluations of generative models fall short in two ways. First, we lack granular insight into how diffusion models fail. Class-conditional models on ImageNet [13] rely on distribution-level metrics like Fréchet Inception Distance (FID) and Inception Score (IS) [57, 19, 33, 41, 15, 48]. These metrics reveal neither which basic visual skills models struggle with nor how they fail. Large text-to-image systems are judged by human preference [28] and by prompt-following benchmarks like GenEval [17], T2I-CompBench [26], DPG-Bench [24], and OneIG-Bench [9], which decompose evaluation into skills such as counting, spatial relations, and attribute binding [35, 2, 25]. However, these benchmarks evaluate only pre-trained models whose behavior is shaped by hidden training distributions, and their scores cannot separate generalization from retrieval of familiar patterns [70, 1, 59, 8, 27].

Second, we lack control over training distributions. ImageNet and large-scale image-text corpora are fixed collections where multiple visual factors vary simultaneously [13, 14]. This makes it difficult to isolate which *skills* are generalized and to study how conditioning formats, data coverage, or model design affect the learning of basic visual skills. These gaps motivate an evaluation setting that

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

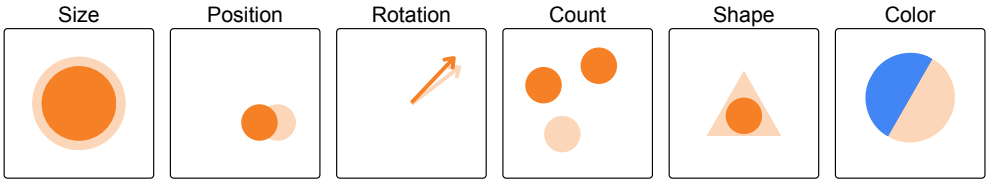


Figure 1: **The Visual Generative Lab (VGL).** VGL trains diffusion models from scratch on synthetic renders labeled with explicit skill values. Each panel varies one skill while holding the others fixed. Models are trained on a subset of skill values or combinations and evaluated on unseen ones within or beyond the training range.

measures whether models generalize specific basic visual skills beyond the support of their training data.

We introduce the Visual Generative Lab (VGL), a framework for training diffusion models from scratch on synthetic geometric data with explicitly controlled training distributions. Conceptually, VGL is inspired by computations in the human ventral visual stream, which build object-centered representations tolerant to variation in position, size, and viewpoint (rotation) [39, 16]. In VGL, we choose exactly which skill values, and which combinations of values, appear in training. Because skill values are provided as explicit numeric or categorical conditions, extrapolation in VGL is a clean distribution-shift test in the conditioning space. Such extrapolation reveals whether a model captures the underlying structure of a skill or falls back to the nearest training value (nearest-support behavior). We then conduct systematic experiments that reveal three patterns about how diffusion models learn and generalize basic visual skills.

Extrapolation is limited. We evaluate single-skill generalization by training models on a restricted set of skill values and testing on held-out values within and beyond the training range. With our default design, size and position degrade within a few pixels of the training range, and count reaches only 52.5% on out-of-range counts. This pattern suggests that, with current diffusion models, most basic visual skills require explicit data coverage (Section 3).

The number of distinct combinations matters more than dataset size. When combining multiple basic visual skills, e.g., generating objects with a specific color and shape, we systematically vary both the percentage of combinations seen during training (*coverage*) and the number of samples per combination (*density*). Models trained on high coverage with few samples per combination outperform models trained on low coverage with many samples per combination (Section 4).

Skills mostly fail independently, except position. We test models conditioned jointly on two to four skills by moving one requested value out of distribution (OOD) while keeping the others in range. We find that the in-range skills mostly keep their accuracy, except when position is OOD, which can also lower in-range size, rotation, and shape (Section 5).

Together, these results suggest that, at least in our controlled setting, diffusion models do not yet generalize basic visual skills far beyond what their training data covers. If this holds more broadly for large-scale visual generative models, data providers and model trainers may need to actively curate data to cover the skill values and combinations users ask for rather than simply relying on more data alone, and evaluations should test models outside that coverage before crediting them with generalization.

2 Visual Generative Lab (VGL)

Motivation. Human visual intelligence builds object representations that are tolerant to changes in position, size, and viewpoint (rotation) via computations in the ventral visual stream [39, 16]. Complex scene understanding and generation build on these skills and their combinations. To test whether diffusion models learn such skills or largely match their training data, we need data that isolate one skill at a time and whose distribution we set exactly. Real-world datasets like ImageNet vary many factors at once, so it is hard to trace model behavior to a specific design choice or property of the data.

Skill	Type	Domain	Extrapolation values
size	continuous	radius $\in [5, 20]$ px	$\{1, 2, 3, 4\} \cup \{21, 22, \ldots, 30\}$ px
position	continuous (2D)	$(x, y) \in [-18, 18]^2$ px	$L_\infty \geq 4$ px outside $[-18, 18]^2$ (320 positions)
rotation	continuous	angle $\in [45°, 315°]$	$\{0°, \ldots, 44°\} \cup \{316°, \ldots, 359°\}$
count	discrete	$\{2, 3, \ldots, 7\}$ objects	$\{0, 1, 8, 9\}$
shape	categorical	4 classes	—
color	categorical	8 classes	—

Table 1: **Training domains and extrapolation values for VGL skills.** size, position, and rotation also have an interpolation split of unseen values inside the domain. shape and color are evaluated on all their classes. Appendix A.1 gives the full value grids and splits.

Defining generalization. We define generalization in VGL relative to the skill values that appear in training. We use size as a running example (Figure 1 shows all six skills). We parameterize object size by radius $r \in [r_{\min}, r_{\max}]$ and render circles on a 64×64 pixel canvas. During training, we control which radius values appear in the training set, e.g., the 16 integer radii between 5 and 20 pixels. The model receives the target skill value as a conditioning signal during training and inference. At test time, we evaluate three forms of generalization, namely (1) interpolation on intermediate values not seen during training, such as radius 12.5; (2) extrapolation on values beyond the training range, such as radius 25; and (3) composition with other basic visual skills (Section 4).

Conditioning format. In VGL, each image is associated with explicit skill values. Continuous skills and count are provided as numeric values, and categorical skills are provided as discrete labels. Our default transformer backbone conditions on these skills by embedding each value into a token and prepending the skill tokens to the image patch tokens, so the denoiser processes conditions and image content jointly through self-attention. Appendix A.3 specifies the exact embedding and tokenization used for each skill.

Evaluation metrics. VGL relies on skill-specific evaluation protocols that measure each skill directly in the generated image with simple computer vision operations, without relying on perceptual metrics [19, 33, 41]. For size, we apply Otsu thresholding [45] and morphological cleaning to obtain a binary mask, estimate the radius from the mask area as $r_{\text{pred}} = \sqrt{\text{area}/\pi}$, and consider a generation correct when the intersection-over-union (IoU) between predicted and ground-truth masks is at least 0.90. We measure the other skills with the mask centroid (position), template matching (rotation), connected components (count), template-based shape classification (shape), and nearest-neighbor color assignment (color). We also report continuous errors such as mean absolute error. Appendix A.4 gives the full metric implementations, and Figure 3 and Appendix D show qualitative examples.

3 Single-Skill Generalization

We use VGL to study the generalization capability of diffusion models on four basic visual skills, namely object size, position, rotation, and count. We also evaluate how common modeling design choices affect generalization on each skill.

3.1 Generalization

Finding 1. Diffusion models extrapolate only slightly beyond the training support.

We first train a baseline diffusion model on each skill and analyze how it generalizes. Table 2 reports its accuracy, and Table 1 lists the training domains and extrapolation values. The *train* columns denote in-support evaluation on newly generated samples whose skill values appear in the training set. Results are means over three random seeds, or ten for the rotation baseline.

Setup. By default, we use the DiT-S/2 [48] architecture and train with the standard DDPM noise prediction objective [22]. Models are trained for 1000 epochs using AdamW [38] with learning rate 1×10^{-4}, except count, which needs 3000 epochs to fit its training values (details in Appendix A.2). At evaluation, we sample with 250-step ancestral DDPM sampling using EMA weights. By default, we do not use classifier-free guidance [21], i.e., guidance scale 1. In Appendix A.5, we show that

Model	size			position			rotation			count		
	Train	Interp.	Extrap.	Train	Interp.	Extrap.	Train	Interp.	Extrap.	Train	Interp.	Extrap.
DiT-S/2 (baseline)	$100.0_{\pm 0.0}$	$96.8_{\pm 0.7}$	$51.5_{\pm 7.5}$	$100.0_{\pm 0.0}$	$97.4_{\pm 3.2}$	$29.7_{\pm 16.8}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$68.0_{\pm 28.9}$	$100.0_{\pm 0.0}$	—	$52.5_{\pm 34.5}$
+ sinusoidal emb.	$100.0_{\pm 0.0}$	$96.9_{\pm 0.4}$	$7.1_{\pm 0.0}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$26.6_{\pm 1.6}$	$98.6_{\pm 1.3}$	—	$20.4_{\pm 11.8}$
+ rotary emb.	$100.0_{\pm 0.0}$	$95.3_{\pm 1.5}$	$9.9_{\pm 3.9}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$38.1_{\pm 6.2}$	$100.0_{\pm 0.0}$	—	$36.3_{\pm 1.3}$
+ linear emb.	$100.0_{\pm 0.0}$	$96.8_{\pm 0.7}$	$51.5_{\pm 7.5}$	$100.0_{\pm 0.0}$	$97.4_{\pm 3.2}$	$29.7_{\pm 16.8}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$68.0_{\pm 28.9}$	$100.0_{\pm 0.0}$	—	$52.5_{\pm 34.5}$
+ AdaLN	$100.0_{\pm 0.0}$	$85.9_{\pm 8.7}$	$31.0_{\pm 4.2}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$61.5_{\pm 14.4}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$58.5_{\pm 11.8}$	$100.0_{\pm 0.0}$	—	$72.5_{\pm 3.3}$
+ concatenation	$100.0_{\pm 0.0}$	$96.8_{\pm 0.7}$	$51.5_{\pm 7.5}$	$100.0_{\pm 0.0}$	$97.4_{\pm 3.2}$	$29.7_{\pm 16.8}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$68.0_{\pm 28.9}$	$100.0_{\pm 0.0}$	—	$52.5_{\pm 34.5}$
+ VAE latent	$99.1_{\pm 0.3}$	$95.9_{\pm 0.7}$	$7.1_{\pm 0.0}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$21.2_{\pm 6.2}$	$93.3_{\pm 0.8}$	—	$48.8_{\pm 15.2}$
+ pixel	$100.0_{\pm 0.0}$	$96.8_{\pm 0.7}$	$51.5_{\pm 7.5}$	$100.0_{\pm 0.0}$	$97.4_{\pm 3.2}$	$29.7_{\pm 16.8}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$68.0_{\pm 28.9}$	$100.0_{\pm 0.0}$	—	$52.5_{\pm 34.5}$
+ flow matching	$100.0_{\pm 0.0}$	$96.7_{\pm 0.3}$	$47.3_{\pm 3.9}$	$100.0_{\pm 0.0}$	$98.9_{\pm 1.9}$	$27.8_{\pm 26.2}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$51.0_{\pm 6.1}$	$100.0_{\pm 0.0}$	—	$24.2_{\pm 1.4}$
+ U-Net (matched #params)	$100.0_{\pm 0.0}$	$67.8_{\pm 1.4}$	$21.2_{\pm 5.8}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$42.7_{\pm 36.4}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$11.2_{\pm 0.1}$	$99.7_{\pm 0.5}$	—	$79.6_{\pm 4.7}$
+ DiT-L (higher #params)	$100.0_{\pm 0.0}$	$85.3_{\pm 4.1}$	$44.5_{\pm 1.1}$	$100.0_{\pm 0.0}$	$98.7_{\pm 1.1}$	$42.5_{\pm 20.4}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$62.0_{\pm 15.6}$	$100.0_{\pm 0.0}$	—	$72.1_{\pm 22.0}$

Table 2: **Single-skill generalization under different design choices.** Entries are mean±std accuracy (%) over three seeds, or ten for the rotation baseline. Success thresholds are IoU ≥ 0.90 for size, 2 px for position, 5° for rotation, and an exact match for count. count has no interpolation split. Each row below the baseline changes one design choice. **Bold** rows are the defaults we adopt, so they repeat the baseline.

adding guidance does not remove the size extrapolation limit. Unless otherwise stated, we generate 20 samples per condition for the accuracy metrics.

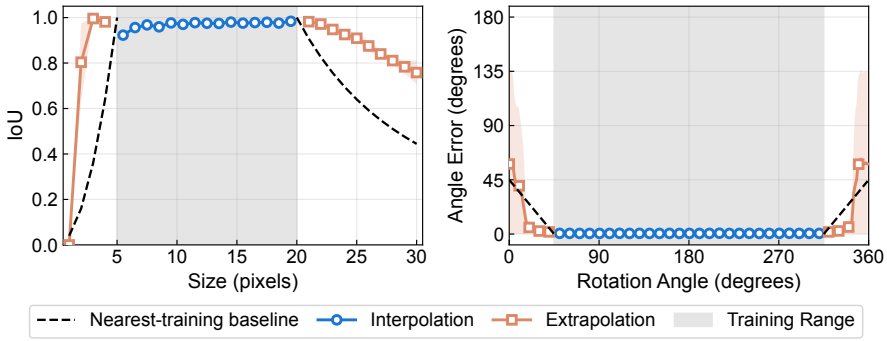


Figure 2: **size interpolates well but extrapolates only slightly, and rotation extrapolation accuracy varies widely across seeds.** Left. IoU per radius. Right. Angular error per angle. Lines are means over the seeds of Table 2 (three for size, ten for rotation), and bands are ±1 standard deviation over seeds. The dashed line is a baseline that outputs the nearest training value (Appendix A.4). The rotation band is wide near 0°, the angle farthest from the training range, because the seeds differ: four keep the error below 5° at every angle, and the others make large errors. Figure 7 shows every seed.

size and position. Prior work has shown that generative models often degrade when asked to generate objects of unseen sizes or positions [12], and we revisit this question in VGL where the training and test distributions are specified directly. For size, we train on circles with 16 integer radii between 5 and 20 pixels and evaluate extrapolation on radii in $\{1, 2, 3, 4\}$ and $\{21, 22, \ldots, 30\}$ pixels. Training radii reach 100% accuracy and interpolation reaches 96.8%, while extrapolation drops to $51.5 \pm 7.5\%$. Above the training range, every seed succeeds (at least half of the samples pass) through radius 24 and fails from 27, while below it, every seed succeeds at radius 3 and fails at radius 1. The extrapolation ability is therefore about 4 px above and 2 px below the training range. The model beats the nearest-training baseline (dashed line in Figure 2) at every out-of-range radius except 1, even where it fails (examples in Figure 3).

For position, we train on circles at 32×32 uniformly spaced positions in $[-18, 18]^2$ pixels and evaluate the far-OOD frame of positions whose L_∞ distance to the training square is at least 4 pixels (320 positions). Training and interpolated positions reach 100% and $97.4 \pm 3.2\%$ accuracy at the 2-pixel tolerance, while far-OOD extrapolation drops to $29.7 \pm 16.8\%$, with accuracy going from 49% at 4.6 px outside training to 11% at 5.8 px out. Figure 3 shows that the model can generate multiple circles or otherwise break the intended single-object structure under out-of-range position queries.

Across both skills, the model produces correct outputs on training and interpolated values but extrapolates only a small amount before performance degrades. To test whether more training data closes the extrapolation gap, we train the baseline on nested sets of 10k to 80k images for each of the four skills at a budget of 78k to 80k optimizer steps. Training and interpolation accuracy stay flat, and extrapolation moves within noise on every skill (Appendix C.3).

rotation. We next ask whether diffusion models can learn precise object rotations. We render asymmetric arrows and train on every degree between $45°$ and $315°$, then evaluate angles outside this interval. Training and interpolation reach $0.30°$ and $0.54°$ mean angular error, and the ten-seed mean extrapolation accuracy is 68.0% at the $5°$ threshold. On the two seeds with the lowest extrapolation MAE (seeds 4 and 6 in Table 7, $0.55°$ and $0.48°$), error stays near zero at every out-of-range angle, far below the nearest-training baseline (Figure 7). rotation is the only skill for which some seeds reach 100% accuracy on the entire out-of-range set. Its input $[\sin\theta, \cos\theta]$ is periodic, so θ and $\theta + 360°$ share one embedding (Appendix A.3) and every out-of-range angle lies within $45°$ of the training range. We do note that relative to the width of each training range, the out-of-range sets of size and count reach farther (Appendix C.1).

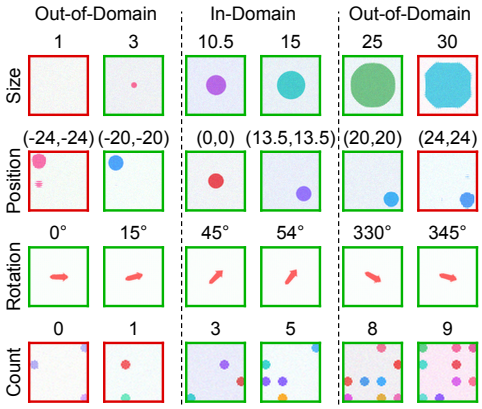


Figure 3: **Generated samples for each skill.** Green/red borders mark success/failure according to the per-skill criteria.

The ten seeds fall into three different extrapolation behaviors. Four seeds extrapolate across the whole out-of-range set with mean angular error below $2°$, three seeds are middling with errors of $8°$ to $15°$, and three seeds collapse near $0°$, the angle farthest from the training range, with errors above $30°$. Appendix B provides per-seed numbers, coverage ablations, and qualitative examples near $0°$.

count. Recent studies show that pre-trained text-to-image diffusion models often fail to generate the correct number of objects [18, 31]. We train on images containing 2 to 7 circles and evaluate extrapolation to counts 0, 1, 8, and 9. The baseline model is perfect on train counts, but reaches only 52.5% on counts just outside the training range. The model shown in Figure 3 draws one to three objects when asked for 0 and usually succeeds on 1, 8, and 9, while one of the three seeds in Table 2 fails on every out-of-range count except 0. Appendix D shows more examples of these failures.

To check whether these trends depend on our default setting, we repeat the single-skill experiments on textured renders (Appendix C.7) and at 128×128 resolution (Appendix C.5). On textured renders, rotation extrapolates best and count worst. At 128×128, size extrapolation is similar and rotation again splits across seeds. We also prompt eight pre-trained text-to-image models for the same four skills (Appendix C.8). None of them follows the requested size, and the best reaches 58% on count.

3.2 Design choices

The baseline extrapolates size, position, and count only slightly, and its rotation extrapolation varies widely across seeds. These trends may depend on modeling choices. We therefore change one design choice at a time and examine how conditioning, data representation, training objective, architecture, and model size affect generalization on the four basic visual skills. The baseline already includes the choices we adopt (the **bold** rows in Table 2).

Conditioning embedding. We compare three ways of embedding condition information: sinusoidal [64], rotary [62], and a simple linear projection. Sinusoidal and rotary embeddings are standard for encoding token positions in transformers, and we adapt them for scalar conditions. The linear embedding instead applies a learnable layer to the scalar value, or to $[\sin\theta, \cos\theta]$ [71] for rotation (Appendix A.3). As shown in Table 2, it outperforms the other two on size, position, and rotation extrapolation and has the highest mean on count. For example, size extrapolation reaches 51.5% versus 7.1% and 9.9%, and rotation extrapolation reaches 68.0% versus 26.6% and 38.1%. This

suggests that embeddings that preserve the structure of the condition (monotone for scalar skills, periodic for rotation) may help extrapolation (Appendix G illustrates this for size).

Conditioning mechanism. We test two methods for injecting condition information: direct concatenation of condition and image tokens, and AdaLN [48], which modulates network activations based on conditions. Both succeed on training values, but neither extrapolates better on every skill. AdaLN improves mean position and count extrapolation (61.5% vs. 29.7% and 72.5% vs. 52.5%), while concatenation is stronger on size (51.5% vs. 31.0%) and rotation (68.0% vs. 58.5%). We use concatenation in the baseline because it has higher interpolation accuracy on average and is stronger on size and rotation extrapolation, but AdaLN still performs competitively.

Pixel vs. latent space. We compare training the model directly in pixel space with training in the latent space of a pre-trained VAE [53]. Pixel-space training improves size extrapolation (51.5% vs. 7.1%), position extrapolation (29.7% vs. 0.0%), and rotation extrapolation (68.0% vs. 21.2%).

Sequence length may account for part of the extrapolation gap. The latent model sees 16 tokens per image, while the pixel model sees 1024. With patch size 1, a latent model with 64 tokens reaches 36.1% on size extrapolation, up from 7.1% with 16 tokens, but only 18.5% on rotation (Appendix C.4). At 128×128, a latent model with 64 tokens is close to the pixel models on size and position extrapolation (Appendix C.5). We keep pixel space as the default, although a latent model with enough tokens may perform similarly on size.

Algorithm and architecture. We also change the generative algorithm to flow matching [37] and the backbone from DiT-S/2 to a U-Net [54, 22]. For controlled comparison, we hold the data and optimization settings fixed. The U-Net matches DiT-S/2 at about 22M parameters (Appendix A.3). Flow matching has lower mean extrapolation accuracy than the baseline on every skill, including rotation (51.0% vs. 68.0%). The U-Net loses most of rotation extrapolation (11.2%, the same as the nearest-training baseline) and is weaker on size extrapolation (21.2%) but is comparatively strong on count extrapolation (79.6%), so an architecture that helps one skill can hurt another.

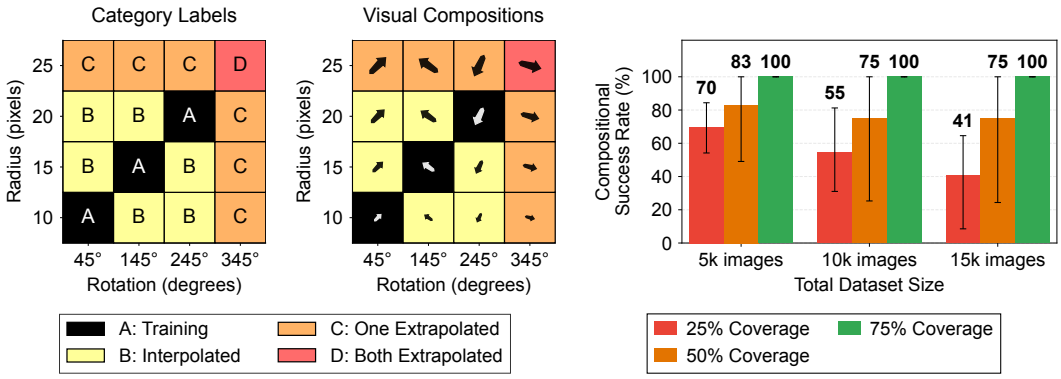
Model size. We further explore whether generalization can be improved by scaling up model size. Scaling the backbone from DiT-S/2 to DiT-L ($\sim$306M parameters) improves mean count and position extrapolation (72.1% vs. 52.5% and 42.5% vs. 29.7%), but does not extend extrapolation on size (44.5% vs. 51.5%) or rotation (62.0% vs. 68.0%). The capacity ablations are discussed in Appendix C.2.

4 Compositional Generalization

Finding 2. For compositional generalization, the number of distinct skill combinations seen in training matters more than dataset size.

Section 3 studies one skill at a time, but controllable generation usually asks for several skills in the same image. Combining skills in this way is closely related to the compositional generalization studied in controlled benchmarks for systematicity [29, 34, 32] and is central to skill-based evaluation of modern text-to-image models [26, 17]. We study compositional generalization by enumerating the full grid of skill values, training on only a fraction of the possible combinations, and evaluating whether models generate correct held-out combinations within range, as well as combinations that move beyond the joint training support. Additional results can be found in Appendix E.

Setup. For each skill pair, we construct a discrete grid of combinations over a small set of values per skill (Appendix A.1). We vary two aspects of the training distribution: (1) coverage, the fraction of unique combinations used in training, and (2) samples per combination (density), the number of images drawn for each training combination. In the default setting (Table 3), every training combination receives the same number of images. In the coverage versus density experiment of Figures 4 and 5, we instead hold the total dataset size fixed. We also note that training combinations are sampled so that every skill value appears in at least one training combination (otherwise a held-out combination would test an unseen value rather than an unseen pairing).



(a) Definition of compositional evaluation categories on a two-skill grid. A: training (both values seen). B: interpolation (each value seen, combination unseen). C: one-skill extrapolation. D: both-skill extrapolation.

(b) Coverage dominates samples per combination. At a fixed total dataset size, more coverage raises accuracy, while more samples per combination do not.

Figure 4: **Covering more skill combinations helps compositional generalization more than adding samples per combination.** In (b), color + shape models are trained on four shapes and eight colors, with 5k, 10k, or 15k images in total. Bars show the mean joint **B** accuracy (both skills correct) over three seeds, and error bars show the range across seeds.

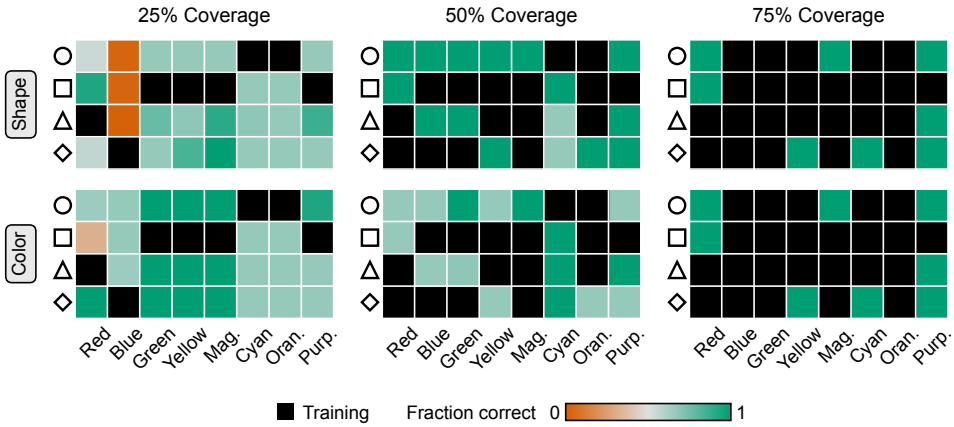


Figure 5: **On color + shape, held-out shape accuracy rises with coverage, and at 75% coverage both skills are correct on every held-out combination.** Results are for the 10k-image models of Figure 4, with one cell per combination. The top row scores shape and the bottom row scores color. Black cells are training combinations, and held-out cells are colored by the fraction of correct samples, averaged over the three seeds.

We evaluate generated images with skill-specific metrics analogous to those used throughout the paper (Appendix A.4) and group test queries into the categories defined in Figure 4. Category **A** denotes the in-distribution baseline averaged across the pair's two skills (training values, training combinations). Category **B** evaluates compositional interpolation, where each skill value lies within its training range but the specific combination is unseen. Category **C** evaluates one-skill extrapolation, where one skill moves outside its training range while the other remains in range (we indicate which skill is out-of-distribution). Category **D** evaluates both-skill extrapolation, where both skills move outside their training ranges.

Coverage versus density. At a fixed image budget, seeing a wider variety of combinations helps composition more than seeing the same combinations more often. Figure 4 reports joint **B** accuracy (both color and shape correct) at total budgets of 5k, 10k, and 15k images while varying coverage, so it isolates diversity of combinations from the number of samples per combination. We note these are separate models from those of Table 3, whose 25% color + shape split trains on 11 of the 32 combinations with about 3.4k images in total. Moving toward higher coverage consistently raises **B**

Visual Skills	Cov.	Cat.	A (%)	shape (%)	color (%)	position (%)	size (%)	count (%)	rotation (%)
color + shape	25%	**B**	100.0	$93.2_{\pm 14.9}$	$94.5_{\pm 6.5}$	—	—	—	—
	50%	**B**	100.0	$98.0_{\pm 3.1}$	$95.6_{\pm 7.0}$	—	—	—	—
	75%	**B**	100.0	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	—	—	—	—
color + count	25%	**B**	100.0	—	$79.9_{\pm 22.1}$	—	—	$38.2_{\pm 42.2}$	—
	50%	**B**	100.0	—	$96.1_{\pm 3.7}$	—	—	$37.7_{\pm 3.5}$	—
	75%	**B**	100.0	—	$99.9_{\pm 0.3}$	—	—	$97.4_{\pm 6.5}$	—
shape + count	25%	**B**	100.0	$13.3_{\pm 11.4}$	—	—	—	$69.6_{\pm 17.8}$	—
	50%	**B**	100.0	$13.7_{\pm 10.6}$	—	—	—	$83.7_{\pm 15.4}$	—
	75%	**B**	100.0	$10.0_{\pm 14.2}$	—	—	—	$46.4_{\pm 27.4}$	—
color + position	25%	**B**	100.0	—	$97.4_{\pm 3.8}$	$99.5_{\pm 1.2}$	—	—	—
	50%	**B**	100.0	—	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	—	—	—
	75%	**B**	100.0	—	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	—	—	—
position + size	25%	**B**	99.3	—	—	$99.8_{\pm 0.6}$	$46.3_{\pm 8.8}$	—	—
	50%	**B**	99.9	—	—	$100.0_{\pm 0.0}$	$96.0_{\pm 3.9}$	—	—
	75%	**B**	99.9	—	—	$100.0_{\pm 0.0}$	$99.7_{\pm 0.6}$	—	—
	75%	**C** (size)	99.9	—	—	$100.0_{\pm 0.0}$	$21.8_{\pm 19.1}$	—	—
	75%	**C** (position)	99.9	—	—	$3.7_{\pm 9.0}$	$8.9_{\pm 10.0}$	—	—
	75%	**D**	99.9	—	—	$1.1_{\pm 2.7}$	$2.8_{\pm 6.7}$	—	—
position + shape	25%	**B**	100.0	$12.9_{\pm 8.0}$	—	$97.6_{\pm 2.5}$	—	—	—
	50%	**B**	100.0	$54.6_{\pm 18.4}$	—	$99.6_{\pm 0.9}$	—	—	—
	75%	**B**	100.0	$100.0_{\pm 0.0}$	—	$100.0_{\pm 0.0}$	—	—	—
size + shape	25%	**B**	100.0	$34.4_{\pm 13.6}$	—	—	$26.2_{\pm 19.1}$	—	—
	50%	**B**	100.0	$82.5_{\pm 11.6}$	—	—	$50.5_{\pm 10.5}$	—	—
	75%	**B**	100.0	$93.7_{\pm 9.8}$	—	—	$72.3_{\pm 12.9}$	—	—
position + rotation	25%	**B**	100.0	—	—	$64.9_{\pm 12.9}$	—	—	$5.6_{\pm 6.4}$
	50%	**B**	100.0	—	—	$95.3_{\pm 7.8}$	—	—	$73.8_{\pm 28.8}$
	75%	**B**	100.0	—	—	$100.0_{\pm 0.0}$	—	—	$100.0_{\pm 0.0}$
	75%	**C** (position)	100.0	—	—	$0.8_{\pm 1.9}$	—	—	$54.4_{\pm 38.1}$
	75%	**C** (rotation)	100.0	—	—	$100.0_{\pm 0.0}$	—	—	$0.0_{\pm 0.0}$
	75%	**D**	100.0	—	—	$2.2_{\pm 5.2}$	—	—	$1.5_{\pm 2.1}$

Table 3: **Compositional interpolation improves with coverage for most two-skill pairs.** Entries are per-skill accuracy (%), the mean over six seeds with the standard deviation as a subscript. The **Cat.** column gives the test category from Figure 4: **B** is interpolation, **C** moves the skill in parentheses out of range, and **D** moves both skills out of range. The **A** column is the in-distribution baseline averaged over the pair's two skills, and gray marks accuracy below 90%.

accuracy, even though the number of samples per seen combination must fall to keep the total budget fixed. By contrast, increasing the number of samples per combination at fixed coverage yields no consistent gain. Each additional training combination shows the model another pairing of skill values, while extra samples of the same combination repeat a pairing the model has already seen. Figure 5 shows this at the cell level for the 10k-image models: held-out shape accuracy grows with coverage, and both skills are correct on every held-out combination at 75% coverage.

We also train color + shape models with $S \in \{4, 8\}$ shapes and $C \in \{8, 16\}$ colors at three coverages, using about 15k training images in every setting. Each shape then appears in k training combinations on average, where k is the coverage times C. Accuracy on held-out combinations is near 40% when $k = 2$, between 59% and 100% when $k = 4$, and 100% whenever $k \geq 6$ (Appendix E). At a fixed image budget, more coverage and more colors therefore both raise held-out accuracy.

Coverage across pairs. For most skill pairs, compositional interpolation improves with coverage (Table 3). For example, count accuracy on color + count rises from 38.2% at 25% coverage to 97.4% at 75%, and color from 79.9% to 99.9%. Continuous pairs also improve with coverage, e.g., size accuracy on position + size rises from 46.3% at 25% coverage to 99.7% at 75%. shape + count is the only pair where neither skill improves with coverage: at 75% coverage, it reaches 10.0% shape and 46.4% count accuracy, and most of its samples draw touching objects that merge into one component (Appendix E). We further analyze the failure mode for size + shape in Appendix E. We discuss the out-of-distribution splits of position + size and position + rotation in Section 5.

Split	size	position	shape	color
In range (**B**)	100.0	100.0	100.0	100.0
size OOD (**C**)	54.6	100.0	100.0	100.0
position OOD (**C**)	79.5	0.0	32.0	100.0

Table 4: **Skill interdependence in a four-skill model.** Accuracy (%) of each skill for one model trained on size, position, shape, and color together at 75% coverage, unlike the two-skill models of Table 3. Gray marks accuracy below 90%. The two three-skill runs show the same pattern (Appendix F).

5 One Skill Out of Range

Finding 3. Basic visual skills mostly fail independently, but an out-of-range position also lowers accuracy on some of the skills that stay in range.

Compositional interpolation tests unseen combinations of in-range skill values. We now move one skill outside its training range and test whether the other skills stay correct. A model that learns each visual skill independently would degrade primarily on the out-of-distribution skill while preserving the accuracy of the remaining in-range skills. Diffusion models condition on all skills jointly during denoising, so the resulting representations can become coupled.

Two skills. With two skills, the coupling runs mainly from position to the other skill (Table 3, 75% coverage). When only position is OOD, accuracy on the in-range skill drops from near 100% to 8.9% for size and 54.4% for rotation, although the requested sizes and angles are within their training ranges. An OOD position also hurts size when size is itself OOD, since for the same queried sizes its accuracy falls from 21.8% to 2.8% once position also leaves the training range (**D**). In the opposite direction, position is unaffected, at 100.0% when size is OOD and 100.0% when rotation is OOD. In the other two-skill models, an OOD size or count does not lower shape or color either (Appendix F).

Three and four skills. Models trained on size, position, and shape and on these three plus color show the same pattern (Table 4; Appendix F). An OOD position lowers shape accuracy from 100% on in-range combinations to 20.4% and 59.9% in the two three-skill runs and to 32.0% in the four-skill model, although the requested shapes are in range. An OOD size leaves position and shape at 100%. In the four-skill model, color stays at 100% on every split, so the coupling holds for some skills but not all.

6 Related Work

Evaluating Visual Generative Models. Generative models are commonly evaluated with distribution-level metrics such as IS and FID [57, 19, 56, 33, 41], and text-to-image models with human preference [67, 28] and benchmarks for prompt following, attribute binding, and counting [35, 2, 25, 26, 17, 24, 9, 18, 68]. These evaluations operate on pre-trained models whose training distributions are not controlled, and each score inherits the errors of its detector or VQA model. Studies of compositional generalization instead use synthetic data with known factors, in language and visual reasoning [34, 32, 29] and in representation learning [40, 6, 63]. VGL also trains on carefully controlled data, so every test value is known to lie inside or outside the training range, and scores each skill with task-specific, rule-based metrics.

Memorization and Generalization. Modern models can generalize well while still memorizing their training data [70, 1], and in diffusion models memorization depends on training-set size, data duplication, text conditioning, and classifier-free guidance [8, 59, 60, 30, 27]. Other work trains diffusion models on synthetic data to study how compositional abilities emerge [44, 46] and when a model can generate more objects than it saw in training [7]. VGL complements these efforts with a controlled setting for measuring when diffusion models extrapolate a skill versus when they fall back to nearest-support solutions.

7 Discussion and Conclusion

We introduced VGL to explore whether diffusion models learn basic visual skills or mainly reproduce what their training data contain. In our controlled setting, we find models mostly handle unseen values and combinations within what they were trained on, but rarely carry a skill far beyond it. If larger models behave the same way, data providers and model trainers may need to curate data that cover the skill values and combinations users ask for, rather than rely on more data alone, and evaluations should test outside that coverage. We hope VGL can serve as a starting point for studying these questions in richer settings.

Limitations and future work. VGL relies on controlled synthetic data, and we mainly evaluate a DiT backbone, so how far these findings carry over to natural images remains open. As a first step, we repeat the single-skill experiments on textured renders (Appendix C.7) and at higher resolution (Appendix C.5), and probe eight pre-trained text-to-image models with rule-based extractors (Appendix C.8). Going forward, we aim to scale VGL to richer scenes, broaden architectures beyond DiT, and apply the same skill-centric probes to larger text-to-image models, to see whether the coverage and interdependence patterns we observe hold at larger scales and in more realistic settings.

Acknowledgments and Disclosure of Funding

This material is based upon work supported by the U.S. National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-2630152. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work received no other third-party funding. The authors declare no competing interests.

References

[1] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *ICML*, 2017.

[2] Eslam Mohamed Bakr, Pengzhan Sun, Xiaoqian Shen, Faizan Farooq Khan, Li Erran Li, and Mohamed Elhoseiny. HRS-Bench: Holistic, reliable and scalable benchmark for text-to-image models. In *ICCV*, 2023.

[3] Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A ViT backbone for diffusion models. In *CVPR*, 2023.

[4] Black Forest Labs. FLUX.2 [klein] 4B. https://huggingface.co/black-forest-labs/FLUX.2-klein-4B, 2026.

[5] Black Forest Labs and Krea. FLUX.1 Krea [dev]. https://huggingface.co/black-forest-labs/FLUX.1-Krea-dev, 2025.

[6] Florian Bordes, Shashank Shekhar, Mark Ibrahim, Diane Bouchacourt, Pascal Vincent, and Ari Morcos. PUG: Photorealistic and semantically controllable synthetic data for representation learning. In *NeurIPS Datasets and Benchmarks*, 2023.

[7] Arwen Bradley. Local mechanisms of compositional generalization in conditional diffusion. In *ICML*, 2026.

[8] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *USENIX Security*, 2023.

[9] Jingjing Chang, Yixiao Fang, Peng Xing, Shuhan Wu, Wei Cheng, Rui Wang, Xianfang Zeng, Gang Yu, and Hai-Bao Chen. OneIG-Bench: Omni-dimensional nuanced evaluation for image generation. In *NeurIPS Datasets and Benchmarks*, 2025.

[10] Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. PixArt-Σ: Weak-to-strong training of diffusion transformer for 4K text-to-image generation. In *ECCV*, 2024.

[11] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. PixArt-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *ICLR*, 2024.

[12] Jooyoung Choi, Jungbeom Lee, Yonghyun Jeong, and Sungroh Yoon. Toward spatially unbiased generative models. In *ICCV*, 2021.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[14] Karan Desai, Gaurav Kaul, Zubin Aysola, and Justin Johnson. RedCaps: Web-curated image-text data created by the people, for the people. In *NeurIPS Datasets and Benchmarks*, 2021.

[15] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021.

[16] James J. DiCarlo, Davide Zoccolan, and Nicole C. Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.

[17] Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. GenEval: An object-focused framework for evaluating text-to-image alignment. In *NeurIPS Datasets and Benchmarks*, 2023.

[18] Xuyang Guo, Jiayan Huo, Yingyu Liang, Zhenmei Shi, Zhao Song, Jiahao Zhang, and Zhen Zhuang. Text-to-image diffusion models cannot count, and prompt refinement cannot help. *arXiv preprint arXiv:2503.06884*, 2025.

[19] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017.

[20] Yuji Hirono, Akinori Tanaka, and Kenji Fukushima. Understanding diffusion models by Feynman's path integral. In *ICML*, 2024.

[21] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[22] Jonathan Ho, Ajay N. Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[23] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *ICML*, 2023.

[24] Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. ELLA: Equip diffusion models with LLM for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024.

[25] Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A. Smith. TIFA: Accurate and interpretable text-to-image faithfulness evaluation with question answering. In *ICCV*, 2023.

[26] Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2I-CompBench: A comprehensive benchmark for open-world compositional text-to-image generation. In *NeurIPS Datasets and Benchmarks*, 2023.

[27] Anubhav Jain, Yuya Kobayashi, Takashi Shibuya, Yuhta Takida, Nasir Memon, Julian Togelius, and Yuki Mitsufuji. Classifier-free guidance inside the attraction basin may cause memorization. In *CVPR*, 2025.

[28] Dongfu Jiang, Max Ku, Tianle Li, Yuansheng Ni, Shizhuo Sun, Rongqi Fan, and Wenhu Chen. GenAI Arena: An open evaluation platform for generative models. In *NeurIPS Datasets and Benchmarks*, 2024.

[29] 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 *CVPR*, 2017.

[30] Zahra Kadkhodaie, Florentin Guth, Eero P. Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representations. In *ICLR*, 2024.

[31] Wonjun Kang, Kevin Galim, Hyung Il Koo, and Nam Ik Cho. Counting guidance for high fidelity text-to-image synthesis. In *WACV*, 2025.

[32] Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. Measuring compositional generalization: A comprehensive method on realistic data. In *ICLR*, 2020.

[33] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *NeurIPS*, 2019.

[34] Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *ICML*, 2018.

[35] Tony Lee, Michihiro Yasunaga, Chenlin Meng, Yifan Mai, Joon Sung Park, Agrim Gupta, Yunzhi Zhang, Deepak Narayanan, Hannah Teufel, Marco Bellagente, Minguk Kang, Taesung Park, Jure Leskovec, Jun-Yan Zhu, Fei-Fei Li, Jiajun Wu, Stefano Ermon, and Percy Liang. Holistic evaluation of text-to-image models. In *NeurIPS Datasets and Benchmarks*, 2023.

[36] Zhimin Li, Jianwei Zhang, Qin Lin, Jiangfeng Xiong, Yanxin Long, Xinchi Deng, et al. Hunyuan-DiT: A powerful multi-resolution diffusion transformer with fine-grained Chinese understanding. *arXiv preprint arXiv:2405.08748*, 2024.

[37] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

[38] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[39] David Marr. *Vision: A computational investigation into the human representation and processing of visual information*. W. H. Freeman, 1982.

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

[41] Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *ICML*, 2020.

[42] 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. In *ICML*, 2022.

[43] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021.

[44] Maya Okawa, Ekdeep S Lubana, Robert Dick, and Hidenori Tanaka. Compositional abilities emerge multiplicatively: Exploring diffusion models on a synthetic task. In *NeurIPS*, 2023.

[45] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[46] Core Francisco Park, Maya Okawa, Andrew Lee, Hidenori Tanaka, and Ekdeep Singh Lubana. Emergence of hidden capabilities: Exploring learning dynamics in concept space. In *NeurIPS*, 2024.

[47] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[48] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

[49] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

[50] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *ICLR*, 2024.

[51] 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. In *ICML*, 2021.

[52] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 2022.

[53] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[54] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[55] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Raphael Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022.

[56] Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In *NeurIPS*, 2018.

[57] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *NeurIPS*, 2016.

[58] ShuttleAI. Shuttle 3 Diffusion. https://huggingface.co/shuttleai/shuttle-3-diffusion, 2024.

[59] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? Investigating data replication in diffusion models. In *CVPR*, 2023.

[60] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Understanding and mitigating copying in diffusion models. In *NeurIPS*, 2023.

[61] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

[62] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024.

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

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[65] Luc Vincent and Pierre Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1991.

[66] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 1927.

[67] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.

[68] Xindi Wu, Dingli Yu, Yangsibo Huang, Olga Russakovsky, and Sanjeev Arora. ConceptMix: A compositional image generation benchmark with controllable difficulty. In *NeurIPS Datasets and Benchmarks*, 2024.

[69] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. CLEVRER: Collision events for video representation and reasoning. In *ICLR*, 2020.

[70] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

[71] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, 2019.

Appendix

A Experimental Details **16**

A.1 Dataset specifications 16

A.2 Count object layout and training schedule 17

A.3 Model architecture and training 17

A.4 Evaluation protocol 18

A.5 Classifier-free guidance ablation (size) 19

B Rotation Ablations **21**

C Additional Single-Skill Experiments **24**

C.1 Extrapolation at matched distance 24

C.2 Model size 24

C.3 Dataset size 24

C.4 Sequence length and latent space 24

C.5 Resolution 25

C.6 Text-encoder conditioning 25

C.7 Textured renders 25

C.8 Pre-trained text-to-image models 26

D Qualitative Analysis of Generalization **28**

E Compositional Results **30**

F Three- and Four-Skill Models **31**

G Embedding Manifold Analysis **32**

H Predicting Extrapolation Accuracy from Per-Skill Features **33**

A Experimental Details

A.1 Dataset specifications

We define six visual skills and their value ranges. Table 5 summarizes the complete domain specification for each skill.

Visual Skill	Type	Range / Values	Value grid
size	Continuous	$[5.0, 20.0]$ pixels	16 values (step 1.0)
position	Continuous (2D)	$[-18, 18] \times [-18, 18]$	32×32 grid (1024 positions)
rotation	Continuous	$[45°, 315°]$	271 values (step $1°$)
count	Discrete	$\{2, 3, \ldots, 7\}$ objects	6 values
shape	Categorical	circle, square, triangle, diamond	4 classes
color	Categorical	red, blue, green, yellow, magenta, cyan, orange, purple	8 classes

Table 5: **Visual skill domains and value grids.** The value grid column gives the number of values in each domain, and positions form a uniform grid over the listed range.

Rendering pipeline. All images are rendered programmatically using Python with PIL (Pillow) and NumPy at 64×64 pixels, except the 128×128 runs of Appendix C.5. Each image is initially rendered at 256×256 pixels and downsampled to 64×64 using Lanczos resampling; the single-skill count images are instead drawn directly at 64×64. The eight color classes use standard RGB values. These are red (255,0,0), blue (0,0,255), green (0,255,0), yellow (255,255,0), magenta (255,0,255), cyan (0,255,255), orange (255,128,0), and purple (128,0,255). Background colors are sampled from whites, light grays, and pastels with RGB values typically above 200 per channel.

Shapes are rendered as follows. Circles use PIL's ellipse primitive with diameter $= 2\times$ size. Squares are axis-aligned with side length $= 2\times$ size. Triangles are equilateral with side length $= 2\times$ size. Diamonds are squares rotated by $45°$ with their vertices at distance size from the center. All shapes use solid fills with no borders. The only object drawn at varying angles is the arrow used in the single-skill rotation experiments, which the position + rotation pair also uses.

In the single-skill datasets of Table 2, each object takes a random color from a separate muted eight-color palette, perturbed by Gaussian noise with $\sigma = 10$ per RGB channel (clipped to $[0, 255]$). Gaussian noise with $\sigma = 5$ is then added to the entire image. The rotation dataset is the exception, with the arrow drawn in (255,100,100) on white and no perturbation.

The compositional datasets of Table 3 and Appendix F draw objects in red unless color is one of the skills, add uniform jitter of up to ± 15 to each object color channel, and add Gaussian image noise with $\sigma \approx 5$. The color + shape datasets of Figure 4 instead shift each color channel of the whole image by a uniform offset in $[-8, 8]$ and add uniform pixel noise in $[-3, 3]$. All color and noise perturbations are applied during dataset generation only; the single-skill size and count models also flip training images horizontally at random.

Train and test splits. For single-skill experiments, we discretize the skill's continuous range and treat each discrete value as a class. Training sets use every value of the grid in Table 5; the rotation ablations of Appendix B instead vary the range and step. Interpolation samples are midpoints between consecutive training values. Extrapolation out-of-distribution samples are values outside the training range but within the valid physical range. For size, the extrapolation range stops at radius 30, where the circle comes within 2 px of each edge of the 64×64 canvas. For position, Table 2 continues the training lattice outward in its 1.16 px steps and scores the 320 points that lie at least 4 px outside the training square in L_∞ distance (4.6 or 5.8 px).

For compositional experiments involving multiple skills, coverage is defined over the Cartesian product of skill values. In the two-skill grids of Table 3, size uses radii $\{6, 8, \ldots, 18\}$ px, trained on $\{6, 8, \ldots, 14\}$ with $\{16, 18\}$ held out. position uses $\{-14, -7, 0, 7, 14\}^2$ px, trained on $x \leq 7$ with the $x = 14$ column held out. rotation is trained on $\{45°, 135°, 225°, 315°\}$ with $\{0°, 15°, 30°, 330°, 345°\}$ held out, and count uses $\{1, 2, 3, 4\}$, trained on $\{1, 2, 3\}$ with 4 held out. shape and color use all their classes in training. A skill that a pair does not vary is fixed at radius

10 px, the image center, a circle (the arrow for position + rotation), red, one object, and $0°$. In color + count and shape + count, the objects have radius 5 px and sit in one horizontal row through the image center, with centers 12.5 px apart.

Using seed 42, we randomly sample a fraction $p\%$ of all combinations of training values without replacement. We then add one combination for each skill value that is still missing, so that every value appears in training. Each added combination pairs the missing value with the first training value of the other skill (e.g., radius 6, one object, $45°$, a circle, or red). In the datasets of Table 3, this raises the realized coverage above its nominal value, most for the smallest grid, shape + count, which reaches 50% at a nominal 25% and 58% at a nominal 50%. At a nominal 25%, color + shape trains on 11 of its 32 combinations. The test splits follow Categories **A**–**D** of Section 4.

For the single-skill experiments of Table 2, 10,000 training images, generated with seed 42, are split evenly across the training values (625 per size value, about 10 per position grid point, about 37 per rotation angle, and about 1,667 per count value). For the two-skill results in Table 3, each training combination receives the same number of samples, from 50 to 357 depending on the pair (312 for color + shape). The total therefore grows with coverage, and so does the number of optimizer steps, since training runs for a fixed 1000 epochs.

The fixed-budget comparisons in Figure 4 and Table 15 instead fix the total training size (for example 15,000 images) and distribute samples uniformly across selected combinations. In Figure 4(b), the three coverages train on 8, 16, and 24 of the 32 color + shape combinations, with 208 to 1,875 images per training combination.

A.2 Count object layout and training schedule

The count dataset of Table 2 places radius-4 circles, drawn 9 px wide, at distinct sites of a 16-px lattice, which leaves 7 px of clearance between neighbors. count is the only skill still improving at 1000 epochs: on this layout the DiT-S/2 baseline reaches $69.7_{\pm 11.9}\%$ training accuracy at 1000 epochs (78k steps) and $100.0_{\pm 0.0}\%$ at 3000 (234k steps), with three seeds each. For the model shown in Figure 3, training accuracy is 14% at 30k steps, 95% at 120k, and 100% at about 200k steps, after which it is flat, while the training loss stays at ≈ 0.017 over the same interval. We use accuracy rather than loss as the convergence criterion for this skill and train it for 3000 epochs.

A.3 Model architecture and training

We use DiT-S/2 (Diffusion Transformer, Small variant) with 12 layers, hidden dimension 384, 6 attention heads, and patch size 2, yielding 32×32 patches for 64×64 images. The model has approximately 22M parameters without the VAE. The U-Net variants in Table 2 have 22.35M parameters for size and count, 22.39M for position, and 22.96M for rotation, against 22.20M for DiT-S/2 (22.44M for position). The AdaLN variants have 33.12M parameters (33.36M for position), because each block adds a modulation layer.

Each visual skill is embedded into the model's hidden dimension (384). Continuous skills use a linear embedding $\mathbf{e} = \mathrm{MLP}(\mathbf{W}v + \mathbf{b})$, where a two-layer MLP maps the 256-dimensional projection to the hidden size. For size and count, v is the scalar value and $\mathbf{W} \in \mathbb{R}^{256\times 1}$; position embeds x and y separately through two such projections and prepends both tokens. rotation uses $\mathbf{W} \in \mathbb{R}^{256\times 2}$ on $[\sin\theta, \cos\theta]$, so its input is periodic by construction. The compositional models instead embed (x, y) jointly as one token with $\mathbf{W} \in \mathbb{R}^{256\times 2}$ and replace $\mathbf{W}v$ for rotation with a fixed 256-dimensional sinusoidal embedding of θ in radians, which is not periodic. We use no periodic construction in these models so that the representation stays consistent across all skills.

The sinusoidal variant of Table 2 replaces $\mathbf{W}v$ with $[\cos(\omega_k v), \sin(\omega_k v)]_{k<128}$, and the rotary variant replaces it with a learned unit vector whose coordinate pairs are rotated by the angles $\omega_k v$, where $\omega_k = 10000^{-k/128}$. For rotation, v is $\theta/360°$ in the sinusoidal and θ in radians in the rotary embedding, with θ taken modulo $360°$. Shape and color use learned embedding tables where each category c has a learnable vector $\mathbf{e}_c \in \mathbb{R}^{384}$.

We use concatenation conditioning (in-context conditioning [48, 3]), where skill embeddings are prepended as separate tokens to the image patch sequence. The input sequence is $[\mathbf{e}_{\text{time}}, \mathbf{e}_{\text{skill}_1}, \mathbf{e}_{\text{skill}_2}, \ldots, \mathbf{x}_{\text{patch}_1}, \mathbf{x}_{\text{patch}_2}, \ldots]$. The timestep embedding $\mathbf{e}_{\text{time}}$ (sinusoidal, 256-dimensional, projected to 384 dimensions) is always included. Each active skill adds one token

(position adds separate x and y tokens in the single-skill models). Standard transformer self-attention processes all tokens jointly. After the transformer, we discard conditioning tokens and apply a final linear projection to predict noise. The U-Net has no token sequence, so it instead adds a learned projection of the skill embedding to the features of every block.

Training objective. We use standard DDPM training with a linear noise schedule over $T = 1000$ diffusion steps with β_t from $\beta_1 = 0.0001$ to $\beta_T = 0.02$. The model is trained to predict the noise ϵ with a mean squared error loss, $\mathcal{L} = \mathbb{E}_{\mathbf{x}_0,\epsilon,t}[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t, \text{cond})\|^2]$. It also predicts the variance (a 6-channel output for 3-channel RGB images, split into noise and log-variance predictions). The loss adds the variational lower bound term with weight 0.001 [43]. The flow-matching models instead regress the velocity $\mathbf{x}_0 - \epsilon$ along $\mathbf{x}_t = (1-t)\,\epsilon + t\,\mathbf{x}_0$ with $t \sim \mathcal{U}[0,1]$, which enters the DDPM timestep embedding without rescaling.

Optimization. We use the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, constant learning rate 1×10^{-4}, and weight decay 0.0. The single-skill rotation models and the count models of Table 2 also clip the global gradient norm at 1.0. Global batch size is 128. The two size models of Appendix G, the single runs of Table 6, and the compositional models other than the vocabulary sweep of Table 15 use 64. We train for 1000 epochs for all experiments except the count models of Table 2, which use 3000 (Appendix A.2), the data-scaling runs, which use 78k to 80k steps (Appendix C.3), and the vocabulary sweep of Table 15, which uses 500.

The DiT-S/2 models of Tables 2 and 3 train on one NVIDIA L40 GPU, except the 3000-epoch count models, which train on four L40 or RTX PRO 6000 GPUs; the DiT-L and U-Net models train on four to eight. Runs with a smaller per-GPU micro-batch accumulate gradients so that the global batch size is unchanged. We maintain an EMA copy of model weights with decay rate 0.9999 updated every training step. Evaluation uses the EMA model, except for the two-skill models of Table 3, which are evaluated with the online (non-EMA) weights. We do this because the EMA starts from the random initialization, which keeps a weight of 0.9999^n after n steps (0.12 at 21k), and many of these runs train for only 21k to 45k steps. Thus, on the shortest runs, the EMA weights lower accuracy on the training combinations from 100% to below 16%.

A.4 Evaluation protocol

Metrics per skill. For size, we threshold the grayscale image with Otsu's method and take the class on the frame border as background. When Otsu's two classes are less than 20 gray levels apart, we instead threshold halfway between the median of the frame border and the gray level farthest from it. This happens when Otsu's method splits the background noise around a very small disk. We apply a 3×3 morphological closing and, to masks of at least 20 pixels, a 3×3 opening. We then estimate size as $r_{\text{gen}} = \sqrt{A/\pi}$ from mask area A, render a ground-truth circle with expected size r_{true}, and compute IoU between generated and ground-truth masks (success criterion IoU at least 0.90, which for concentric disks allows a radius error of about 5%). We report accuracy and mean IoU.

For position, we compute the binary mask centroid (c_x, c_y) as center of mass, convert to image relative coordinates (center is (0,0), range $[-32, 32]$), and compute Euclidean distance d in pixels (success criterion $d \le 2.0$ pixels). We report accuracy and mean absolute error (MAE) in pixels. For the DiT-L and U-Net rows of Table 2, position training and interpolation accuracy is measured on one fixed random sample of 100 conditions per split, and extrapolation on all 320 conditions.

For rotation, we use template matching against arrow renders at whole-degree angles, searched coarse to fine in 10°, 3°, and 1° steps, and compute circular angular error. Because the templates are at whole degrees, interpolation midpoints are read 0.5° off, so interpolation MAE cannot fall below 0.5°. The success threshold is $\Delta\theta \le 5^\circ$. We report accuracy and MAE in degrees.

For count, we blur the grayscale image and split it with Otsu's method; if the two classes are less than 20 gray levels apart, the frame is treated as blank and the count is 0. Otherwise the class on the frame border is background, and the object mask is thresholded halfway between Otsu's threshold and the background mean, which keeps light objects whole. We separate touching objects by watershed segmentation [65] of the distance transform, with markers where the distance exceeds 0.6 of the largest distance in the same blob, count regions with area larger than 30 pixels, and require an exact match with no partial credit. The area threshold is below the area of one object (61 px^2 as drawn at radius 4), so it removes speckle and keeps every object. We report accuracy.

For shape, we classify the object by template matching against renders of each shape. For color, we compute mean RGB within the object mask and assign the closest color from the eight-color palette of Table 5 using L2 distance. The vocabulary sweep of Table 15 averages over the mask eroded by one pixel and assigns the closest of the cell's C colors. We report accuracy for both shape and color.

Multi-skill metrics. For the two-, three-, and four-skill models (Tables 3 and 16, Figures 4(b) and 5), every generated sample is scored on its own against the values it was conditioned on, and accuracy is the fraction of samples that pass. The object mask is the Otsu threshold of each pixel's RGB distance to the background color. A sample in which at most 10 pixels differ from the background by more than 60 contains no object and fails every skill.

Without count, the object is the largest connected component. With count, every connected component of at least 30 pixels is an object, and shape passes only when each has the requested shape, so touching objects that merge into one component fail shape. In the count pairs, color is the mean over the whole object mask. size passes when the object's area is between 0.9 and $1/0.9$ times the area of a render of the requested shape and size, which allows a size error of about 5% (two concentric renders with this area ratio have an IoU of 0.90).

position passes when the object's centroid is within 2 pixels of the centroid of a render at the requested position, and rotation passes when template matching against a 1° bank of arrow renders is within 5°. count counts connected components of at least 30 pixels and splits touching objects where the distance transform narrows to at most 0.75 of the smaller object's peak. shape takes the best fit among circle, square, and triangle templates searched over orientation. A square within 15° of axis-aligned is a square and one within 15° of 45° is a diamond, a triangle must be within 15° of upright, and elongated or poorly fitting objects match no shape.

Ground-truth check. On ground-truth renders, every metric above scores 100% on each split of Table 2, of the multi-skill results, and of the vocabulary sweep of Table 15. On the other size and count data of Appendix C, including the textured renders, the pass rate is at least 99% per split.

Nearest-training baseline. For each out-of-range size or rotation query, the baseline returns the nearest training value, which is the closer end of the training range. We score it in closed form, by the IoU of concentric circles with the queried and returned radii for size and by the circular angular error for rotation; Figure 2 shows this baseline as the dashed line. It scores 7.1% on size and 11.2% on rotation extrapolation. Applied to position and count, the same rule scores 0% on extrapolation, since no out-of-range query is within the success threshold of a training value. The DiT-S/2 (baseline) row of Table 2 is above it on extrapolation for all four skills. On interpolation, the rule scores about 70% for size and 100% for position and rotation, whose midpoints all lie within the success threshold of a training value.

Sampling procedure. At evaluation, we use DDPM ancestral sampling with 250 steps respaced from the 1000 training steps [43]; the flow-matching models instead integrate the learned velocity field with 250 Euler steps that evaluate the velocity at each step's midpoint time. We initialize $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ and iteratively denoise. For each test condition, we generate $N = 20$ samples and report metrics averaged over these samples. For the single-skill DDPM models, the initial noise $\mathbf{x}_T$ uses a fixed seed per sample index.

A.5 Classifier-free guidance ablation (size)

For the size experiment, we also train the baseline of Table 2 for classifier-free guidance (CFG), with conditioning dropout 0.1 on one seed, and evaluate how guidance strength affects accuracy on the same interpolation and extrapolation radii, with 20 samples per condition. At inference time, we vary the guidance scale from 1 to 100 (Figure 6). Scales up to 4 improve both splits but extend size extrapolation by only two radii, and from guidance scale 10 on, extrapolation accuracy stays below 10%; thus, guidance does not remove the extrapolation limit. At every scale, this model extrapolates less accurately than the baseline trained without dropout (51.5%).

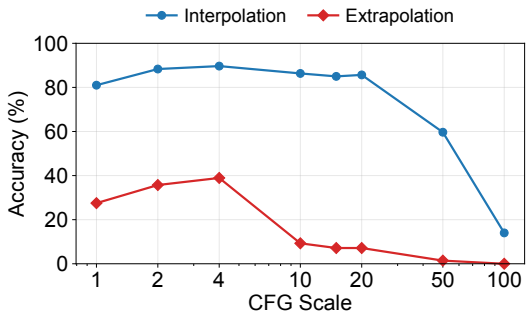


Figure 6: **Classifier-free guidance does not remove the size extrapolation limit.** Accuracy of the size baseline trained with conditioning dropout 0.1 (one seed), across inference guidance scales. Scales up to 4 raise accuracy on both splits but move the upper extrapolation limit only from radius 22 to 24, and larger scales lower both.

B Rotation Ablations

Training-coverage ablations. In these ablations, accurate rotation extrapolation needs a dense grid of training angles. We evaluate six variants of DiT-S/2 trained on other subsets of the rotation skill space (Table 6). Reducing angular coverage from 270° to 180° (90° to 269° at a 1° step) keeps the training angles accurate (MAE 0.17°), with 45.9% extrapolation accuracy and 44.5° MAE. Its extrapolation angles reach 90° beyond the training range, while those of the main configuration reach 45°. Within 45° of the training range, its extrapolation accuracy is 73.3%, inside the spread of the ten seeds of the main configuration (32.3% to 100.0%, mean 68.0%). At a 1° step, the run on 30° to 329° reaches 100.0% extrapolation accuracy on angles up to 30° from its training range; at the same distances, the ten seeds of the main configuration average 79.7% (47.9% to 100.0%).

Value-grid spacing impacts generalization. Models trained with 1° increments substantially outperform those with 30° increments, which fit their training angles but extrapolate poorly. For the range starting at 45°, increasing the increment from 1° to 30° lowers extrapolation accuracy from 68.0% (mean of the ten seeds, 32.3% to 100.0%) to 14.7%. The 30° grid ends at 285°, but even within 10° of its training range its accuracy is 70.5%, against 99.2% on average for the ten seeds of the 1° grid (94.3% to 100.0%). We use the 10-seed analysis below to characterize seed sensitivity for our main configuration.

			MAE (°)			Acc. (%)
Training angles	**Step**	***N***	**Train**	**Interp.**	**Extrap.**	**Extrap.**
30° to 329°	1°	300	0.30	0.53	0.87	100.0
30° to 300°	30°	10	0.10	10.6	26.4	11.2
45° to 285°	30°	9	0.00	2.6	47.9	14.7
90° to 269°	1°	180	0.17	0.51	44.5	45.9
90° to 255°	15°	12	0.08	2.4	98.4	22.8
90° to 240°	30°	6	0.17	3.1	82.2	8.2
45° to 315° (main, 10 seeds)	1°	271	$0.30_{\pm 0.03}$	$0.54_{\pm 0.01}$	$19.2_{\pm 22.5}$	$68.0_{\pm 28.9}$

Table 6: **Rotation training set ablations.** Each row is a DiT-S/2 model (linear embedding, concatenation conditioning, no conditioning dropout) trained on N angles over the listed range and step. Columns give mean absolute error (MAE) on the training angles, their midpoints (Interp.), and every whole degree outside the training range (Extrap.), plus extrapolation accuracy at the 5° threshold. Training accuracy is 100.0% for every model. Rows above the line are one seed each, and the two at a 1° step score Train and Interp. on 50 evenly spaced angles. The last row is the mean±std of the ten seeds of Table 2. For the same starting angle, a 1° step gives lower interpolation and extrapolation errors than a coarser one.

Per-seed behavior of the main configuration. For the main configuration (45° to 315°, 1° step), we train ten seeds with identical data and schedule and report per-seed mean angular error and accuracy at the 5° threshold in Table 7. Inside the training range, all ten seeds produce essentially identical numbers (training MAE $0.30 \pm 0.03°$, interpolation MAE $0.54 \pm 0.01°$).

By extrapolation MAE, the ten seeds fall into three different behaviors. Four seeds (2, 3, 4, 6) generalize well across the full extrapolation grid (MAE $< 2°$, accuracy 99.9% to 100%); three seeds (0, 5, 7) are middling (8° to 15° MAE); and three seeds (1, 8, 9) collapse near 0° (43° to 59° MAE). The middling and collapsed seeds average 45.7% and 47.7% extrapolation accuracy. The collapse is concentrated within $\sim 15°$ of 0°, the angle farthest from the training range; outside that region, accuracy is 99.8% to 100% for the four healthy seeds but at most 81.7% for the other six. The collapse rate is 3 of 10 (30%, Wilson [66] 95% CI: $[11\%, 60\%]$). Figure 7 plots the angular error per angle for every seed. Figure 8 shows generations from one seed per behavior.

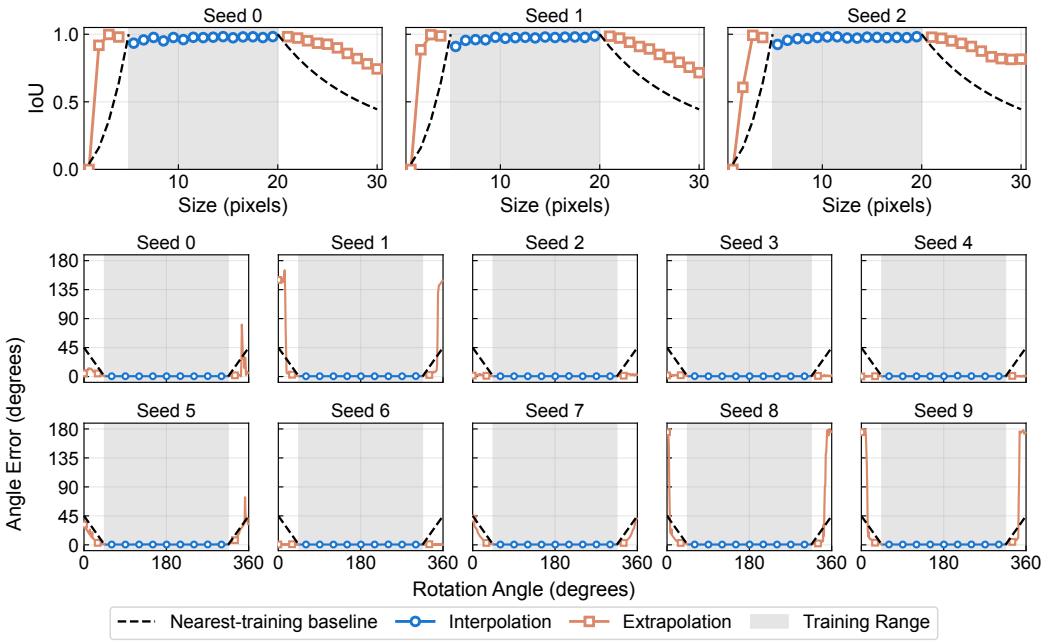


Figure 7: **Per-seed size and rotation curves.** Top. IoU per radius for each of the three size seeds of Figure 2. Bottom. Angular error per angle for each of the ten rotation seeds. The dashed line and the shading are as in Figure 2.

	Train		Interp.		Extrap.	
Seed	MAE (°)	Acc. @ 5° (%)	MAE (°)	Acc. @ 5° (%)	MAE (°)	Acc. @ 5° (%)
0	0.31	100.0	0.54	100.0	8.41	63.2
1	0.30	100.0	0.53	100.0	48.94	53.2
2	0.29	100.0	0.53	100.0	1.90	99.9
3	0.27	100.0	0.53	100.0	1.49	100.0
4	0.32	100.0	0.54	100.0	0.55	100.0
5	0.37	100.0	0.57	100.0	15.33	32.3
6	0.24	100.0	0.52	100.0	0.48	100.0
7	0.30	100.0	0.53	100.0	12.41	41.6
8	0.33	100.0	0.55	100.0	42.88	53.3
9	0.28	100.0	0.53	100.0	59.28	36.6
mean±std	$0.30_{\pm 0.03}$	$100.0_{\pm 0.0}$	$0.54_{\pm 0.01}$	$100.0_{\pm 0.0}$	$19.17_{\pm 22.46}$	$68.0_{\pm 28.9}$

Table 7: **Per-seed rotation results for the main configuration.** The ten seeds share one setup (DiT-S/2, linear embedding, concatenation conditioning, no conditioning dropout) and the same training data ($[45°, 315°]$ at a $1°$ step). They match inside the training range but split in extrapolation: seeds 2, 3, 4, and 6 are healthy, seeds 0, 5, and 7 are middling, and seeds 1, 8, and 9 collapse near $0°$.

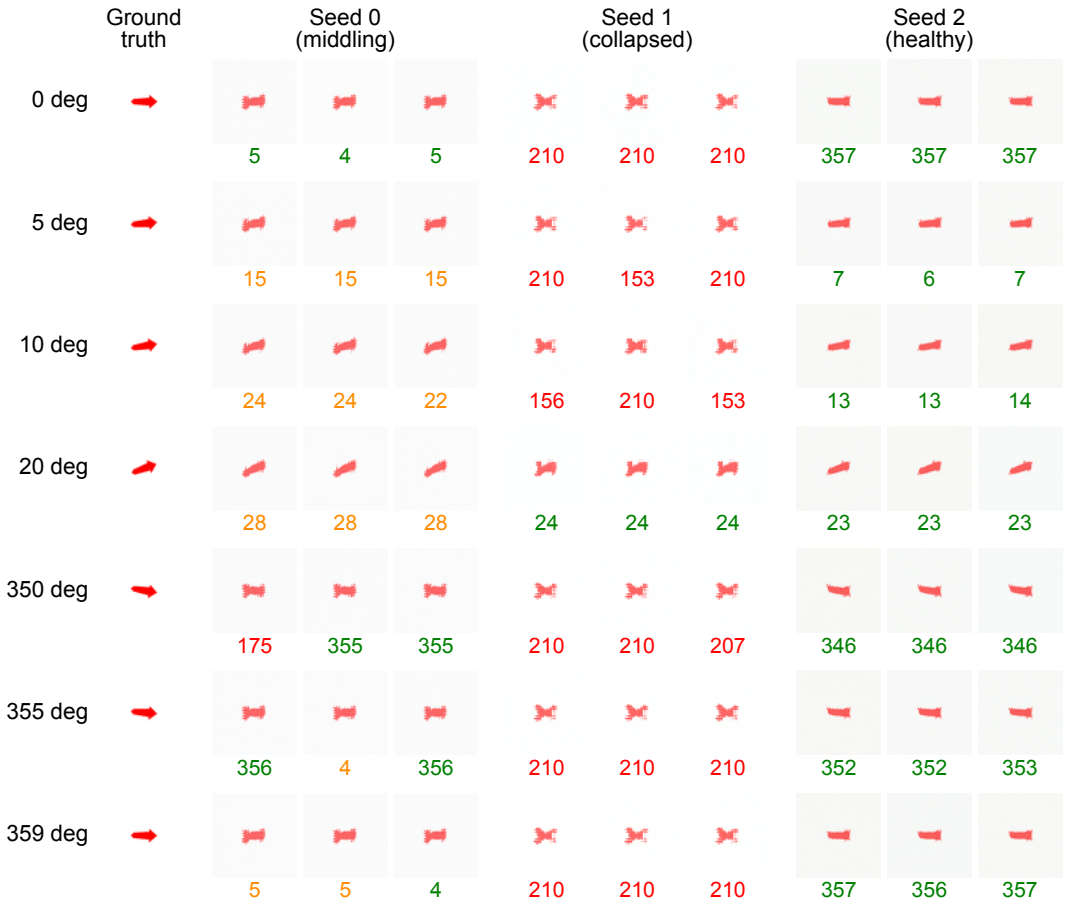


Figure 8: **Rotation generations near $0°$ from a middling, a collapsed, and a healthy seed.** Each row is a target angle outside the training range, with three samples per seed. The number below each sample is its detected angle, colored by angular error (green $\leq 5°$, orange $\leq 30°$, red larger). Within $\sim 15°$ of $0°$, the collapsed seed draws X-shaped blobs instead of arrows.

C Additional Single-Skill Experiments

Apart from Appendix C.1 and Appendix C.2, which use the baseline and DiT-L models of Table 2, the count models in this appendix train on radius-8 circles (radius 16 at 128×128) at random positions and on the same schedule as the other skills, rather than on the lattice and 3000 epochs of Appendix A.2. In this setup, the 64×64 baseline fits 54.7% of its training counts.

C.1 Extrapolation at matched distance

The out-of-range sets of Table 2 reach different distances from the training range. To compare skills at the same distance, we divide the distance outside the training range by the width of that range, $\delta = d_{\text{out}}/\text{width}$, and bin the baseline extrapolation accuracy by δ (Table 8). For position, d_{out} is the L_∞ distance to the training square, and the out-of-range positions form two rings at 4.6 and 5.8 px. Every out-of-range angle and position has $\delta \le 1/6$, while size and count are also tested farther out. In this range, rotation is more accurate than position on average, and four of its ten seeds stay at or above 90% on every value. size is at 100% on every seed in the two bins it shares with rotation, and count has no value with $\delta \le 1/6$.

	size	position	rotation	count
range width	15 px	36 px	270°	5
$\delta \in [0, 0.07)$	$100.0_{\pm 0.0}$	—	$95.6_{\pm 7.6}$	—
$\delta \in [0.07, 0.14)$	$100.0_{\pm 0.0}$	$48.8_{\pm 25.8}$	$51.5_{\pm 43.9}$	—
$\delta \in [0.14, 1/6]$	—	$11.5_{\pm 15.7}$	$43.7_{\pm 49.8}$	—
$\delta \in (1/6, 0.25)$	$72.5_{\pm 15.0}$	—	—	$55.0_{\pm 49.9}$
$\delta \in [0.25, 0.45)$	$44.2_{\pm 23.1}$	—	—	$50.0_{\pm 35.0}$
$\delta \in [0.45, 2/3]$	$0.0_{\pm 0.0}$	—	—	—
$\delta \le 1/6$, mean	$100.0_{\pm 0.0}$	$29.7_{\pm 16.8}$	$68.0_{\pm 28.9}$	—
$\delta \le 1/6$, seeds $\ge 90\%$	3/3	0/3	4/10	—

Table 8: **Extrapolation accuracy (%) at matched distance from the training range.** δ is the distance outside the training range divided by the width of that range. Cells are mean±std over the seeds of Table 2 (three, or ten for rotation), with the values in a bin weighted equally, and — marks bins with no tested value. The last two rows pool $\delta \le 1/6$, which covers every out-of-range angle: the mean, and the number of seeds at or above 90% on every value.

C.2 Model size

The DiT-L row of Table 2 scales the backbone from 22.20M to ∼306M parameters and keeps the data, conditioning mechanism, optimizer, schedule, and sampler of DiT-S/2. At the larger scale, training accuracy stays at 100%, and mean extrapolation accuracy on size and position is below 50%. Scaling to DiT-L improves count and position on average ($+19.6$ pp and $+12.8$ pp on extrapolation, with a large spread across seeds), but does *not* improve size (-7.0 pp) or rotation (-6.0 pp). On rotation, the three DiT-L seeds reach 52% to 80% extrapolation accuracy, and one of them collapses near $0°$, as three of the ten DiT-S/2 seeds do (Appendix B).

C.3 Dataset size

Table 9 reports the baseline trained on nested subsets of 10k, 20k, 40k, and 80k images for each skill, three seeds each, at a budget of 78k to 80k optimizer steps (1000 epochs at 10k images, 125 at 80k). Extrapolation here is scored on a narrower out-of-range grid than Table 2 (radii up to 25 rather than 30; positions from the training boundary rather than 4 px beyond it), so absolute size and position values are higher than in the main table; compare within a row. No skill improves with dataset size beyond seed noise.

C.4 Sequence length and latent space

The VAE row of Table 2 compresses a 64×64 image to an 8×8 latent, which DiT-S/2 turns into 16 tokens, while the pixel baseline uses 1024. Table 10 holds parameters at ∼22M and varies the token

skill	10k	20k	40k	80k
size	$77.6_{\pm 6.4}$	$76.3_{\pm 11.7}$	$75.4_{\pm 10.0}$	$80.4_{\pm 2.5}$
position	$67.7_{\pm 20.1}$	$69.1_{\pm 13.9}$	$70.1_{\pm 9.5}$	$73.5_{\pm 7.6}$
rotation	$81.2_{\pm 32.5}$	$79.3_{\pm 34.4}$	$87.8_{\pm 21.2}$	$64.2_{\pm 31.1}$
count	$14.6_{\pm 2.9}$	$17.5_{\pm 6.5}$	$16.3_{\pm 9.4}$	$13.3_{\pm 3.8}$

Table 9: **Extrapolation accuracy (%) against training-set size.** Mean±std over three seeds.

count by changing the patch size. A latent model with 64 tokens (patch size 1) reaches 36.1% on size extrapolation, far above the 16-token latent model, but only 18.5% on rotation, no better than the 16-token latent model (21.2%); a pixel model with 16 tokens (patch size 16) reaches 2.7% on rotation and fits its training data poorly, with each token encoding a 16×16 patch (768 values).

space	model	tokens	size	rotation
latent	DiT-S/2	16	7.1	21.2
pixel	DiT-S/16	16	—	2.7
latent	DiT-S/1	64	36.1	18.5
pixel	DiT-S/2	1024	51.5	68.0

Table 10: **Extrapolation accuracy (%) as the token count varies at about 22M parameters.** The DiT-S/2 rows repeat the means from Table 2. The other rows are one seed each, scored on the same out-of-range values as that table.

C.5 Resolution

Table 11 reports the baseline at 128×128 in pixel space (DiT-S/4, three seeds) and in a VAE latent space (one seed), with length thresholds scaled by two and the count blob threshold by four. Each split is capped at 100 evenly spaced conditions, size extrapolation is scored on radii 2 to 9 and 41 to 60, and position extrapolation starts at the training boundary. size extrapolation is similar to 64×64 (46.8% vs. 51.5%, within the seed spread); position extrapolates further in absolute pixels but about as far relative to image size; rotation seeds again fall into different behaviors as at 64×64. In pixel space, count fits only 36.4% of its training counts, against 54.7% at 64×64. The latent model (64 tokens) extrapolates size and position about as well as the pixel models, and its rotation accuracy (13.7%) is close to that of the weakest pixel seed (14.3%).

	pixel			VAE latent		
skill	train	interp	extra	train	interp	extra
size	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$46.8_{\pm 6.6}$	100.0	90.3	47.5
position	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$73.3_{\pm 11.8}$	100.0	100.0	72.2
rotation	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$59.6_{\pm 43.1}$	100.0	100.0	13.7
count	$36.4_{\pm 4.9}$	—	$13.8_{\pm 3.3}$	89.2	—	55.0

Table 11: **Accuracy (%) at** 128×128**.** Pixel: mean±std over three seeds. Latent: one seed.

C.6 Text-encoder conditioning

Table 12 replaces the numeric embedding with a frozen CLIP ViT-B/32 text encoder [51]: each skill value selects the embedding of a sentence that states it, and a learned linear layer takes the place of $\mathbf{W}v + \mathbf{b}$ (Appendix A.3), with the rest of the model trained as usual. Training accuracy matches numeric conditioning on all three skills, but interpolation falls from 96.8% to 13.3% for size and from 100.0% to 60.9% for rotation, and extrapolation collapses, with rotation dropping from 68.0% to 8.4%. This suggests that the ordered numeric input matters for extrapolation.

C.7 Textured renders

Table 13 reports the baseline trained on renders with textured backgrounds, radial shading, and varied object color. Otsu thresholding fails on textured backgrounds, so these runs are scored with a

skill	train	interp	extra
size	100.0	13.3	0.0
rotation	100.0	60.9	8.4
count	54.2	—	1.3

Table 12: **Accuracy (%) with CLIP text conditioning in place of the numeric embedding.** One seed.

saturation-mask extractor. On the textured renders, size extrapolation is about 20 points lower and rotation about 20 points higher than on the simple renders, although the rotation gain is within its seed spread. rotation extrapolates best on these renders and count worst.

skill	simple	textured	Δ
size	77.3	56.9	−20.4
position	66.5	67.7	+1.2
rotation	66.0	86.3	+20.3
count	12.9	15.4	+2.5

Table 13: **Extrapolation accuracy (%) on simple and textured renders.** Both columns are scored on the narrower out-of-range grid of Appendix C.3.

C.8 Pre-trained text-to-image models

We prompt eight pre-trained text-to-image models for the four numeric skills and score the outputs with rule-based extractors adapted to these images: a redness mask, watershed counting, and the grid cell, equivalent-circle diameter, and arrow direction of the largest red blob that neither touches the image border nor encloses a hole, or of the largest blob if none qualifies (Table 14). Prompts request red circles on a white background with a stated count (1 to 10), position (nine grid cells), size (six diameters from 10% to 90% of the frame width), or rotation (eight compass directions for an arrow). A size passes within 25% of the requested diameter and an angle within 22.5°, while count and position must match exactly. Accuracy is computed over the 97.9% of samples that contain a measurable red subject, and bold marks accuracy above chance level, which we take as the 95th percentile of accuracy when the requests are randomly shuffled across images.

Figure 9 shows the median-error sample for one prompt per skill from four of the eight models. No model is above chance level on size; count tops out at 58%; position improves sharply only in Shuttle 3 and FLUX.2 klein, and rotation most in FLUX.2 klein. For size, regressing the measured diameter on the requested one gives slopes from 0.013 to 0.076 across the eight models, against 1 for a model that follows the request. These numbers compare models under one protocol and are not comparable to Table 2. The prompted value ranges and the tolerances differ from VGL's, and the models were not trained on this distribution.

model	count	position	size	rotation
Stable Diffusion 1.5 [53]	**8.6**	11.5	24.6	16.4
SDXL [50]	1.0	**15.7**	11.4	13.1
PixArt-α [11]	**32.0**	11.1	31.7	13.8
PixArt-Σ [10]	**32.5**	11.1	26.7	13.8
HunyuanDiT [36]	**30.0**	11.1	22.8	**18.1**
Shuttle 3 [58]	**49.2**	**61.2**	16.7	**31.7**
FLUX.1 Krea [5]	**41.0**	**32.2**	20.0	**35.6**
FLUX.2 klein [4]	**58.0**	**89.4**	16.7	**74.4**

Table 14: **Skill accuracy (%) of pre-trained text-to-image models under one prompting protocol. Bold** marks accuracy above chance level.

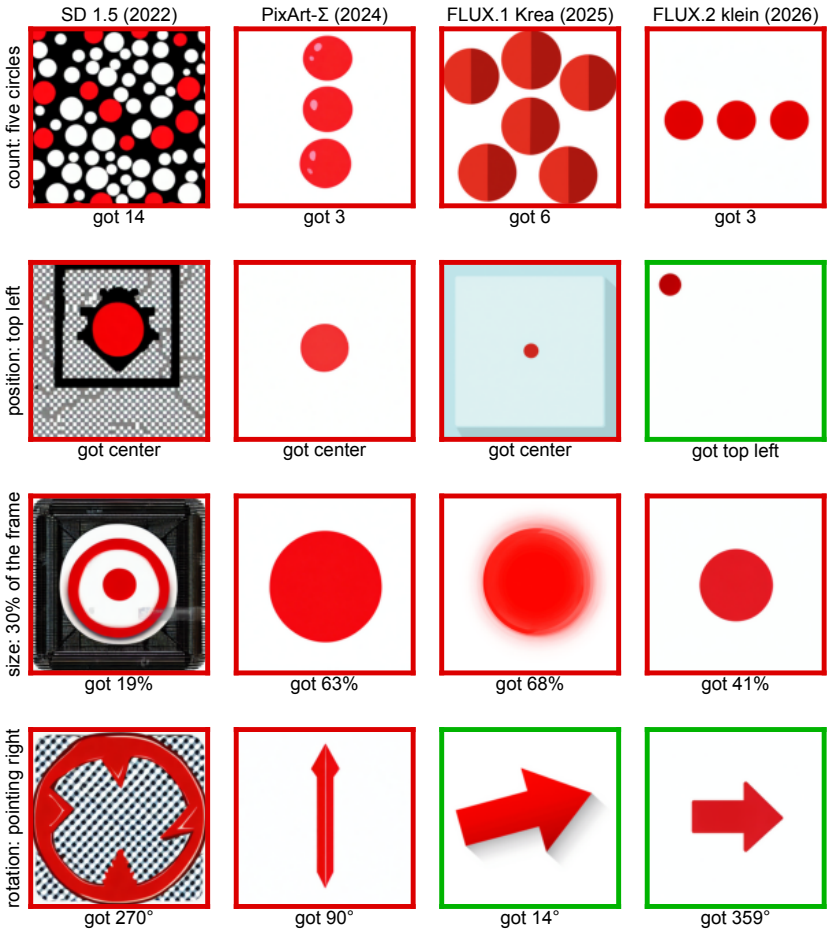


Figure 9: **Median-error samples from pre-trained text-to-image models for one prompt per skill.** Each tile is the sample with the median error among 20 generations. The value measured by the Table 14 scorer is printed below it, and a green border marks a sample that counts as correct in that table. In these samples, only FLUX.2 klein places the object, the two FLUX models orient it, and no model gets the count or size right.

D Qualitative Analysis of Generalization

We provide qualitative analysis of model behavior across all four skills. Figure 3 summarizes per-skill generalization, with borders marking success or failure. Figure 10 summarizes generalization behavior for position and count. The size and rotation curves are in Figure 2, with size extrapolation attempts in Figure 11a and per-seed rotation results in Appendix B. Figure 12 provides additional qualitative failure modes for count beyond the curve view.

Size extrapolation limits. The size model achieves strong interpolation but fails at extreme extrapolation (Figure 11a). Across the three seeds of Table 2, radius 24 always succeeds (at least half of the samples pass), radius 25 succeeds for two seeds, radius 26 for one, and radii 27 to 30 fail for all. Radius 2 succeeds for one seed (45.0% of samples over the three seeds), and for radius 1 pixel the model fails to generate anything. For larger sizes beyond training, circles lose their shape and appear more square-like. Single runs trained on radii 9 to 16, 5 to 20, and 11 to 26 succeed from radius 5 to 18, 3 to 24, and 7 to at least 32, respectively, so the extrapolation limits move with the training range rather than staying at fixed sizes.

Position extrapolation. The position model demonstrates good interpolation and attempts extrapolation to far corners (Figure 11b). However, at extreme positions, the model generates additional circle fragments rather than a single displaced circle.

Count challenges. The count model trained on $\{2, 3, 4, 5, 6, 7\}$ circles reproduces every trained count exactly but extrapolates poorly (Figures 10 and 12). On out-of-range queries, the model shown in Figures 3 and 12 produces the correct count for 1, 8, and 9 most of the time and almost never for 0, where it draws one to three objects. One of the three seeds in Table 2 fails on every out-of-range count except 0, where it leaves the frame empty, so the three-seed extrapolation accuracy is 52.5%. Because the model fits every training count, these failures are not due to underfitting.

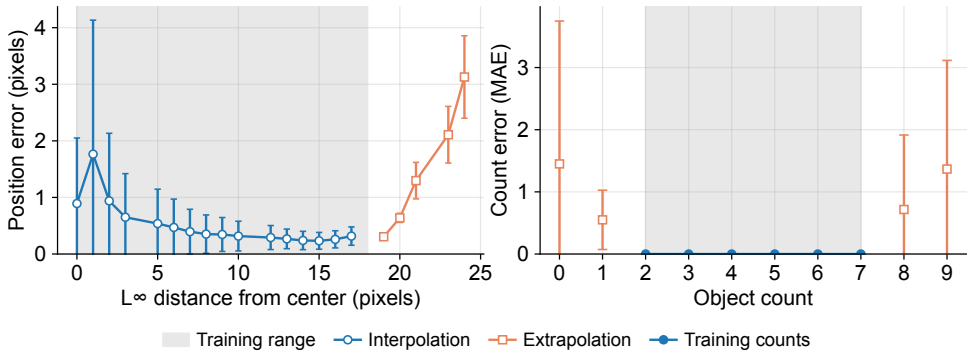
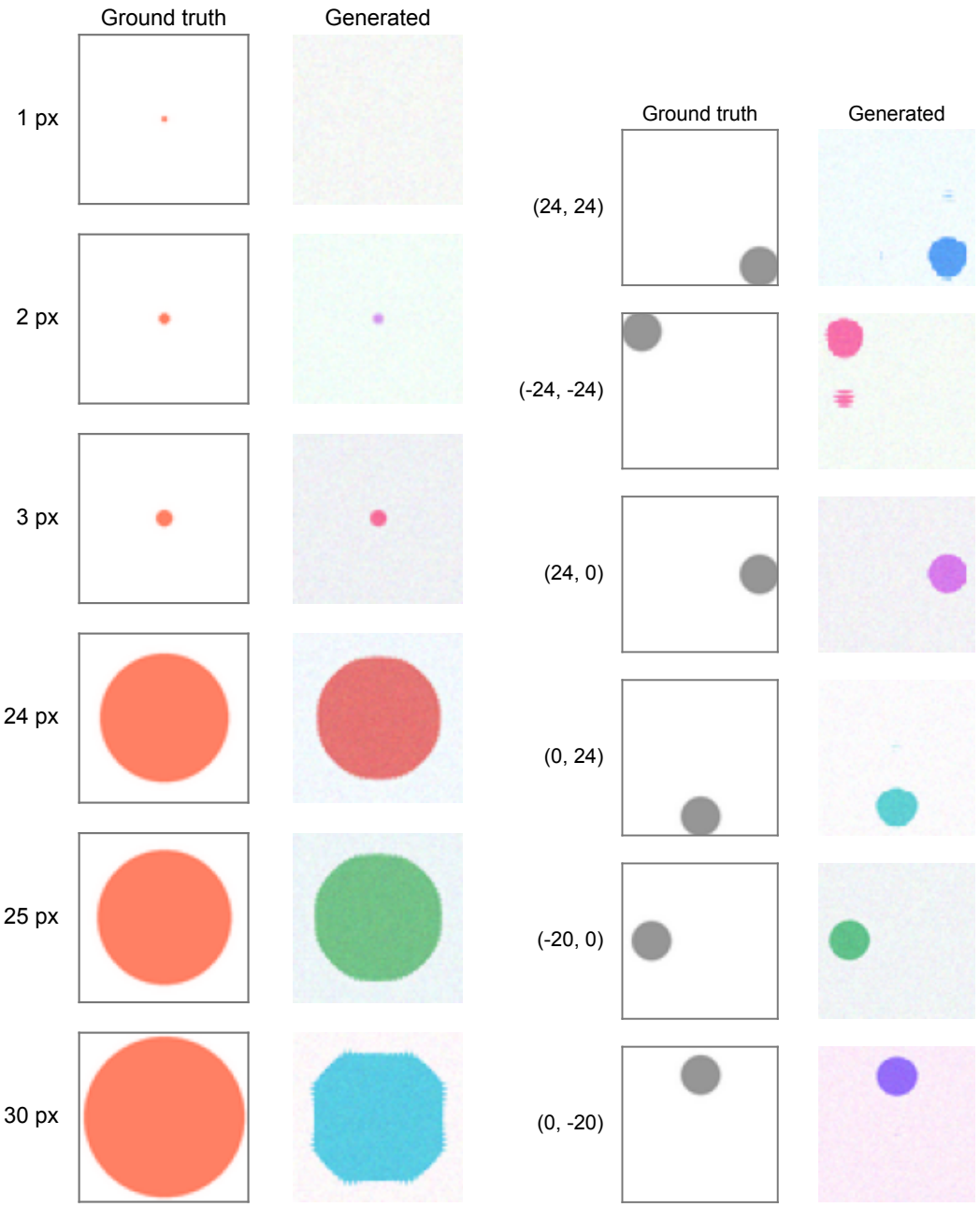


Figure 10: **Position and count errors stay low inside the training range and grow outside it.** Left. Mean position error of the position baseline against L_∞ distance from the image center. Right. Mean absolute count error of the count baseline. Error bars show one standard deviation over three seeds. The wide bar at count 0 reflects one seed that draws about four objects while the other two mostly leave the frame empty.



(a) size extrapolation attempts. (b) position extrapolation attempts.

Figure 11: **Beyond the training range, large circles turn square-like and distant circles are pulled inward.** Rows are labeled by radius or by (x, y) offset from the image center, and each pairs the ground truth (canvas outlined) with a generated sample. At the farthest offsets, the generated circle usually lands just outside the training range, sometimes with a small extra fragment.

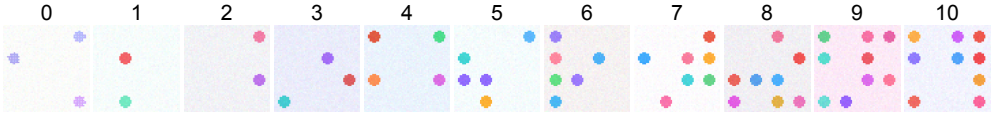


Figure 12: **Count extrapolation failure modes.** Each panel shows one generated sample for the requested count, from a model trained on 2 to 7 objects. Counts 1, 8, and 9 are usually correct, although the sample shown for 1 has two objects.

E Compositional Results

Analysis of size + shape failure. The size and shape composition difficulties may stem from the context-dependent nature of the radius parameter. A circle with radius 10 differs visually from a square or triangle with the same radius parameter. This mismatch means the same numeric size value corresponds to different pixel footprints across shapes, which can make held-out pairings harder to learn (training combinations score 100%). size is the weaker skill of this pair. At 75% coverage, size reaches 72.3% against 93.7% for shape, and most size failures draw the requested shape at the wrong size (22% of samples, against 6% that get both wrong). Two thirds of the size misses are radius-12 triangles drawn about 8% too large.

Analysis of shape + count failure. shape + count is the one pair in Table 3 whose interpolation fails for both skills at 75% coverage (10.0% shape accuracy, 46.4% count accuracy). In 79% of its samples, touching objects merge into one component that matches no shape and must be split before counting.

Combinations per attribute value. Table 15 lists twelve color + shape cells from a vocabulary-size sweep ($S \in \{4, 8\}$ shapes, $C \in \{8, 16\}$ colors, coverage 25–75%, about 15,000 training images in every cell) with $k = K/S$, where K is the number of training combinations. The eight-shape vocabulary adds pentagon, hexagon, star, and cross to the shapes of Table 5, and the 16-color vocabulary adds lime, teal, pink, brown, navy, maroon, olive, and azure to its colors. Each image shows one centered object of radius 12 px, and a sample counts as correct only when both its shape and its color are right under the metrics of Appendix A.4. Across the twelve cells, $\log_2 k$ alone gives $R^2 = 0.62$ on held-out-combination accuracy, against 0.60 for $\log_2 K$ and 0.49 for coverage; with $n = 12$ the first two are not separable, but every cell with $k \geq 6$ reaches 100%. Five random draws of the training combinations at $S = 4$, $C = 8$, and 50% coverage, each trained with one seed, reach 100.0, 100.0, 100.0, 100.0, and 87.5% on held-out combinations, within the range of the three training seeds of that cell (68.8–100%). For the same S and C, coverage moves held-out accuracy from 38.4% at 25% to 100% at 75%, so which combinations are held out matters less than how many.

S	C	coverage	K	k	seeds	held-out (%)
4	8	25%	8	2	3	$38.4_{\pm 38.4}$
8	8	25%	16	2	3	$42.0_{\pm 25.2}$
4	16	25%	16	4	3	$59.1_{\pm 1.9}$
8	16	25%	32	4	1	100.0
4	8	50%	16	4	3	$89.6_{\pm 18.0}$
8	8	50%	32	4	3	$100.0_{\pm 0.0}$
4	8	75%	24	6	3	$100.0_{\pm 0.0}$
8	8	75%	48	6	3	$100.0_{\pm 0.0}$
4	16	50%	32	8	1	100.0
8	16	50%	64	8	1	100.0
4	16	75%	48	12	1	100.0
8	16	75%	96	12	1	100.0

Table 15: **Held-out color + shape accuracy saturates once k, the average number of training combinations per shape, reaches about six.** S, C, and K are the numbers of shapes, colors, and training combinations, so $k = K/S$. Every cell trains on about 15,000 images, and a held-out sample counts only when both skills are correct. Three-seed rows give the mean with the standard deviation as a subscript.

Out-of-range splits at lower coverage. Section 5 uses the 75% models of Table 3, which reach near 100% interpolation accuracy. The coupling of Section 5 also holds at 50% coverage: an out-of-range position lowers size accuracy from 96.0% to 20.8% and rotation accuracy from 73.8% to 57.9%. At 25%, an out-of-range position does not lower size (46.3% in range, 49.5% out of range), and rotation interpolation is only 5.6%, so the 39.6% on its out-of-range split says little about interdependence.

Out-of-range rotation. On the **C** (rotation) split of Table 3, the position + rotation models draw nearly every arrow at the nearest training angle. Unlike the single-skill rotation models, they are trained on four angles 90° apart and embed the angle without periodicity (Appendix A.3).

F Three- and Four-Skill Models

We evaluate trained three-skill and four-skill compositional models (DiT-S/2, 75% coverage, two training runs for the three-skill setting and one for the four-skill setting) on splits built from their training grid. The grid has radii 6, 8, ..., 14, four shapes, and, for the four-skill model, eight colors. Its positions lie on a 7-pixel grid from -14 to 14 in x and y, of which the 20 with $x \leq 7$ are used in training. The splits are training combinations (**A**), held-out in-range combinations (**B**), radius 16 or 18 (**C**, size OOD), $x = 14$ (**C**, position OOD), and both (**D**). Table 16 reports the per-sample accuracy of every skill, with 20 samples per combination and the multi-skill metrics of Appendix A.4.

Section 5 summarizes the four-skill model; Table 16 adds the two three-skill runs and the splits where both size and position are out of range. In both three-skill runs, an out-of-range position also lowers shape (20.4% and 59.9%) and size (30.9% and 84.7%), while an out-of-range size leaves shape and position at 100%. In the first three-skill run and the four-skill model, most samples with only position out of range contain a malformed object that matches no shape (55% and 65%, against 13% in the second three-skill run). The two-skill out-of-range splits not listed in Table 3 show the same pattern at 75% coverage. With position out of range, shape falls from 100.0% to 33.0% in position + shape, while color stays at 99.9% in color + position. With size or count out of range, the other skill does not fall: shape is at 99.4% in size + shape (93.7% in range), color at 100.0% in color + count, and shape at 51.5% in shape + count (10.0% in range).

Model	Split	size (%)	position (%)	shape (%)	color (%)
3-skill, run 1	**A** (training)	100.0	100.0	100.0	—
	B (in-range)	99.6	100.0	100.0	—
	C (size OOD)	37.6	100.0	100.0	—
	C (position OOD)	30.9	0.0	20.4	—
	D (both OOD)	6.6	0.0	25.5	—
3-skill, run 2	**A** (training)	100.0	100.0	100.0	—
	B (in-range)	97.9	100.0	100.0	—
	C (size OOD)	37.2	100.0	100.0	—
	C (position OOD)	84.7	0.0	59.9	—
	D (both OOD)	44.6	0.0	56.5	—
4-skill	**A** (training)	100.0	100.0	100.0	100.0
	B (in-range)	100.0	100.0	100.0	100.0
	C (size OOD)	54.6	100.0	100.0	100.0
	C (position OOD)	79.5	0.0	32.0	100.0
	D (both OOD)	50.5	0.0	19.2	100.0

Table 16: **An out-of-range position degrades size and shape in every model, while an out-of-range size leaves the other skills intact.** The three-skill models condition on size, position, and shape, and the four-skill model adds color. All are trained at 75% coverage. Entries are per-skill accuracy (%). Gray marks accuracy below 90%.

G Embedding Manifold Analysis

To compare how different conditioning embedding schemes organize skill information, we visualize learned embedding manifolds for size conditioning. Figure 13 shows 3D PCA projections of the conditioning token vector at radii from 1 to 24 pixels in steps of 0.2 (the embedding fed into the transformer after the learnable projection to hidden dimension 384). The linear embedding traces a smooth, near-monotone trajectory in PCA space, while the sinusoidal embedding follows a longer, winding path, consistent with its multi-frequency construction. The figure is a qualitative illustration of why the linear embedding's monotone structure may be easier to extrapolate from.

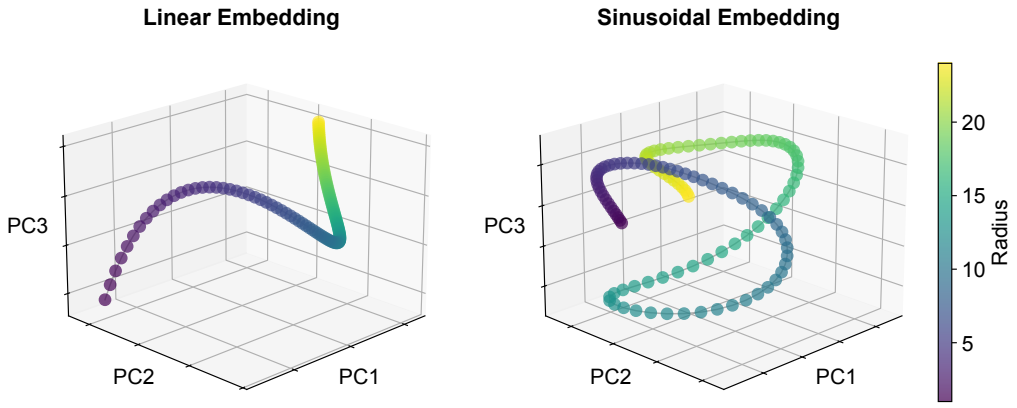


Figure 13: **The linear size embedding follows a near-monotone path, while the sinusoidal one follows a longer, winding path.** Each point is the 3D PCA projection of the learned conditioning embedding for one radius from 1 to 24 pixels, colored by radius. Both models are DiT-S/2 trained on radii 5 to 20.

H Predicting Extrapolation Accuracy from Per-Skill Features

Setup. We summarize the three-seed ablation sweep from Table 2 as a small tabular dataset with one row per skill, variant, and seed. This gives n=103 rows from four skills, eight variants, and three seeds (ten for the rotation baseline). The response is extrapolation accuracy `extra_acc`. The predictors are the skill, embedding type, conditioning mechanism, pixel or VAE latent space, DDPM or flow objective, and architecture (DiT, including DiT-L, or U-Net). We fit ordinary least-squares models in scikit-learn [47] with one-hot encoded categorical features. The dropped reference designs are linear embedding, concatenation conditioning, pixel space, DDPM, DiT, and rotation, so each coefficient is a percentage point change relative to that reference design.

Regression results. The across-skill categorical model attains $R^2 = 0.540$. Adding interpolation accuracy as a predictor leaves the fit essentially unchanged, with $R^2 = 0.553$ and a coefficient of 0.712 (standard error 0.43) on `interp_acc`. A model using only skill, training accuracy, and interpolation accuracy as predictors explains substantially less variance ($R^2 = 0.183$). Figure 14 breaks extrapolation accuracy down by skill and variant, and Figure 15a shows that the categorical model captures broad trends but leaves substantial residuals, especially for rotation, where the failure near 0° is seed driven. The largest across-skill coefficients are sinusoidal embedding (-41.2), VAE latent space (-35.4), and rotary embedding (-33.6). Fitting each skill separately, with one indicator per variant and baseline as the reference, gives $R^2 = 0.96$ for size, 0.72 for count, 0.66 for position, and 0.60 for rotation. Each value is the share of that skill's variance that lies between variants.

Interpretation and caveats. First, in-range performance is a weak proxy for extrapolation. Within-skill Spearman correlations between interpolation and extrapolation accuracy are weak for size and position ($|\rho| \leq 0.30$). They are undefined for rotation, whose interpolation accuracy is 100% for every variant, and for count, which has no interpolation split. Second, design effects are skill dependent. For example, AdaLN improves position but hurts size, while U-Net improves count but hurts size and rotation. Thus the regression supports the paper's qualitative claim that no single architecture or conditioning choice gives uniformly reliable extrapolation across these basic skills. We note that the analysis uses only 103 rows, each variant changes one design choice from the baseline (so interactions between choices are not observed), and the held-in R^2 values should not be read as performance estimates for unseen designs.

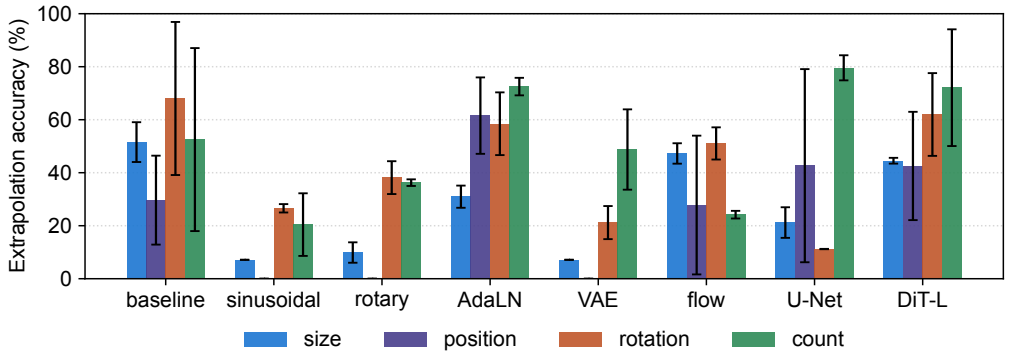
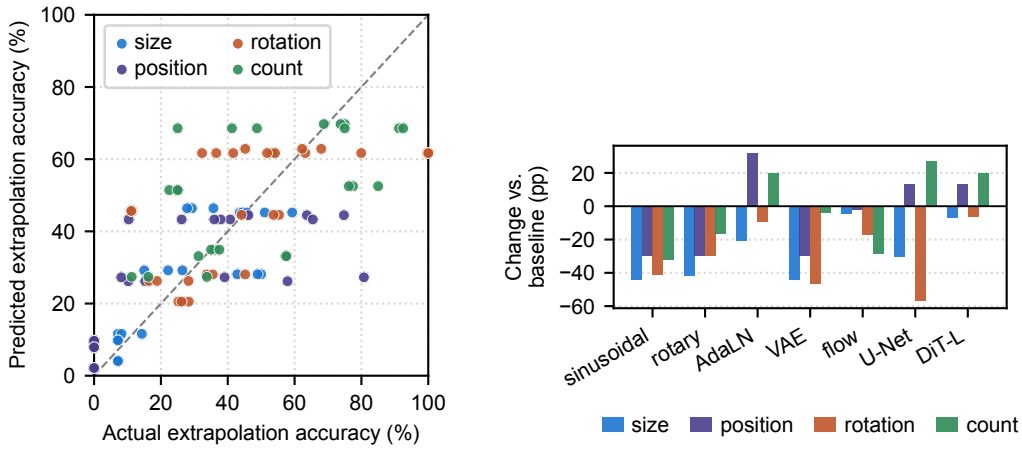


Figure 14: **The same design choice can help one skill extrapolate and hurt another.** Bars give extrapolation accuracy for each skill and variant, averaged over three seeds (ten for the rotation baseline), and error bars show one standard deviation. AdaLN, for example, helps position but hurts size. Figure 15b summarizes these effects as per-skill regression coefficients.



(a) Predicted vs. actual extrapolation accuracy.

(b) Per-skill variant effects.

Figure 15: **One regression across all skills leaves large residuals, partly because a design choice can push different skills in opposite directions.** Panel (a) plots that fit's predictions from skill and design choices against measured extrapolation accuracy, with one point per skill, variant, and seed. Panel (b) shows the coefficients of separate per-skill fits, as percentage-point changes from the baseline.

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 and Section 1 enumerate the three findings (limited extrapolation, the number of distinct combinations matters more than dataset size, skills do not always fail independently), each of which is supported by Sections 3–5 with quantitative results in Tables 2, 3, and 4 and Figure 4b.

   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: The Limitations paragraph in Section 7 states that VGL limits the study to synthetic data, that we primarily evaluate a DiT backbone, and that transfer to natural images remains open; Appendix B reports the seed sensitivity of the rotation results, and Appendix C.6 suggests that the ordered numeric input matters for extrapolation.

   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: [N/A]

   Justification: The paper is purely empirical and contains no theorems, lemmas, or formal proofs.

   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: Sections 2 and 3 specify the architecture (DiT-S/2, 64×64 pixel space, DDPM), training recipe (1000 epochs, 3000 for count, AdamW, lr 10^{-4}), and training domains (Table 1); Appendix A provides hyperparameters, embeddings, and dataset construction details, and the datasets are generated programmatically from rules in the paper.

   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: The training, evaluation, and dataset-generation code is released at https://github.com/AmishSethi/visual-generative-lab.

   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: Section 3 describes the optimizer (AdamW, lr 10^{-4}, 1000 epochs, 3000 for count), sampler (250-step DDPM, guidance scale 1), and 20-samples-per-condition evaluation protocol; Table 1 lists the training domains and extrapolation values per skill, and Appendix A provides hyperparameter and embedding details.

   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: Table 2 reports mean ± sample standard deviation across three random seeds (ten for the rotation baseline), and Table 3 reports mean and standard deviation over six training seeds; the variability factor is the random seed under fixed data and schedule, except that in 14 of the 24 pair and coverage settings of Table 3, seeds 3–5 were trained on a second rendering of the same training combinations. The split of rotation seeds into three extrapolation behaviors is discussed explicitly, with per-seed numbers in Table 7, and Figure 4b shows the range across three seeds.

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: A 1000-epoch DiT-S/2 run takes about 25 GPU-hours on one NVIDIA L40; DiT-L runs take about 210 to 220 L40 GPU-hours on four GPUs, U-Net runs about 30 to 110 on four GPUs, two-skill compositional runs about 4 to 20 on one GPU, and the 3000-epoch count runs 2 to 59 hours of wall-clock time on four NVIDIA RTX PRO 6000 GPUs (four L40 GPUs for the flow-matching row and four to eight for the U-Net row). We estimate from our Slurm accounting that the experiments reported in the paper used roughly 19,000 L40 GPU-hours and 1,800 RTX PRO 6000 GPU-hours; the full project, including preliminary and discarded runs and evaluation, used about 31,500 L40 GPU-hours. Appendix A.3 gives the architecture and training schedule.

   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 train only on synthetic, programmatically generated geometric renders (64×64, and 128×128 in Appendix C.5) with no human subjects, no scraped data, and no personally identifiable information. The latent-space models use the pretrained Stable Diffusion VAE and Appendix C.6 uses a frozen CLIP text encoder, both trained by others on web data, and Appendix C.8 only prompts publicly released text-to-image models for simple shapes. The work conforms to the NeurIPS Code of Ethics in all respects.

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: [N/A]

Justification: This is a foundational diagnostic study of small diffusion models trained from scratch on synthetic geometric renders. No model of natural imagery or deployable system is released, so we see no direct path to societal impact or misuse.

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: The released artifacts are code that generates synthetic 64×64 renders of geometric shapes, and small from-scratch diffusion checkpoints trained on such renders, neither of which poses a meaningful misuse risk.

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: The DiT backbone [48], U-Net [54], DDPM [22], flow matching [37], and Stable Diffusion VAE [53] are cited in Section 3, CLIP [51] in Appendix C.6, the eight text-to-image models in Appendix C.8, and scikit-learn [47] in Appendix H; the released code is built on the DiT codebase, whose original license terms are respected. The licenses are CC BY-NC 4.0 (DiT code), MIT (CLIP, Stable Diffusion VAE, and the ADM diffusion code [15] included in the DiT codebase), CreativeML OpenRAIL-M (Stable Diffusion 1.5), CreativeML Open RAIL++-M (SDXL, PixArt-α, PixArt-Σ), the Tencent Hunyuan Community License (HunyuanDiT), Apache 2.0 (Shuttle 3, FLUX.2 klein), BSD 3-Clause (scikit-learn), and the FLUX.1 [dev] Non-Commercial License (FLUX.1 Krea [dev]). We use all of them for non-commercial research only.

    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: The synthetic VGL datasets and trained-from-scratch diffusion checkpoints are documented in Section 2 and Appendix A (skill domains, splits, generation rules, training and evaluation protocols), and the released repository includes the dataset-generation scripts, training and evaluation code, and a README that links the checkpoints on the Hugging Face Hub. Our code and the checkpoints are released under CC BY-NC 4.0.

    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: The paper does not involve any crowdsourcing or human subjects; all data are synthetic and all evaluation is automatic via deterministic computer-vision metrics described in Section 2 and Appendix A.4.

    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: The paper does not involve any human subjects, so IRB approval is not applicable.

    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: [N/A]

    Justification: LLMs are not part of the core methodology; the models we train are DiT and U-Net diffusion models trained from scratch on synthetic geometric renders and conditioned on skill values, or on frozen CLIP text embeddings in Appendix C.6.

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.