Compositional Flow Matching with Factored Velocity Fields

Avery HW Ryoo [1 2] **Dane Malenfant** [1 3] **Matthew Perich** [1 2] **Guillaume Lajoie** [1 2]

Abstract

Conditional generative models can have difficulty generating attribute combinations absent from training, even when each individual factor is densely covered, otherwise known as a failure to *compositionally generalize*. We propose a factored conditional flow matching architecture that uses a shared base velocity augmented by per-factor heads, summed at the bottleneck. We show that on the Shapes3D and MPI3D-real datasets, the factored architecture matches or beats a parameter-matched monolithic baseline under three structured zero-shot holdout strengths over a two-attribute lattice, notably lowering held-out FID by $\sim 2.4\times$ on the 50% and 75% patterns on Shapes3D. Next, we conduct a slice-attack ablation using per-factor classifier-free composition but show that the factored architecture remains strictly better on both metrics, confirming the gain is structural rather than a consequence of a weak baseline model. Finally we show that the per-head construction also enables a $K \to K{+}1$ modular extension where a new factor head can be added to a frozen K-factor stack and trained alone, producing a working $(K{+}1)$-factor model without retraining the base model or any existing head.

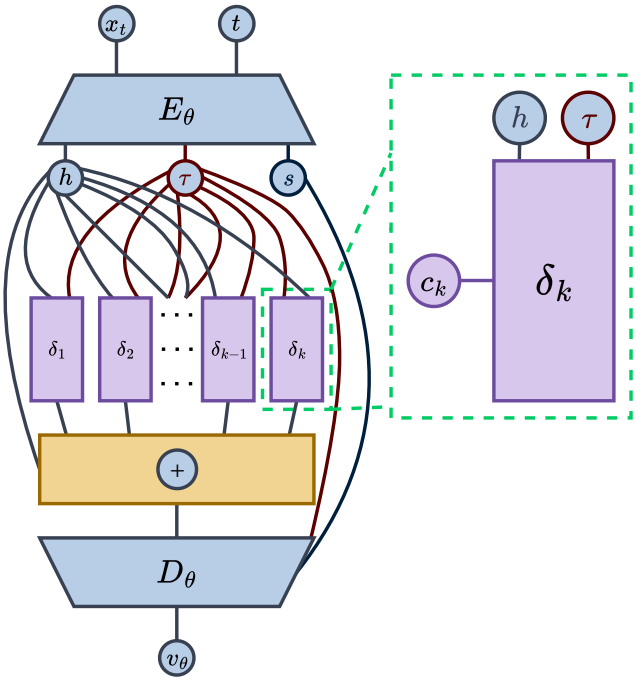


Figure 1. Factored velocity field for zero-shot compositional generalization. A UNet base velocity v_{base} is augmented at the bottleneck by $K{=}6$ per-factor heads $\delta_k(x, t, c_k)$ summed into a single update.

1. Introduction

A generative model with a structured conditioning vector $c = (c_1, \ldots, c_K)$ should support two things: (i) independent control of each factor c_k and (ii) generation of novel factor combinations not seen during training, otherwise known as *compositional generalization*. However, in practice, these models can exhibit failures when asked to generate factor combinations that were absent from the training set (Schott et al., 2022; Montero et al., 2022; Okawa et al., 2023).

A couple existing approaches address this challenge by attaching explicit per-factor structure to the generator. Composable Diffusion (Liu et al., 2022) trains independent diffusion models per concept and composes them at sampling. Concept Sliders (Gandikota et al., 2023) train one LoRA (Hu et al., 2021) per attribute against contrastive data, achieving multi-attribute control via post-hoc weight-space combination of independently trained sliders. ControlNet (Zhang et al., 2023) adds a trainable encoder branch per conditioning signal to a frozen backbone, with multi-condition control obtained by post-hoc composition of independently-trained branches. In all of these, training is independent per factor and multi-factor composition is achieved only at inference, with sampling cost scaling with the number of factors. Furthermore, none are evaluated in the zero-shot combinatorial

[1]Mila - Quebec AI Institute [2]Université de Montréal [3]McGill University. Correspondence to: Avery HW Ryoo <heewoon.ryoo@mila.quebec>.

Published as a paper at the 1st FoGen workshop, ICML 2026, Seoul, South Korea, 2026. Copyright 2026 by the author(s).

setting.

To address these shortcomings, we propose a *factored* velocity field that augments a single shared base velocity $v_{\text{base}}(x, t)$ with one per-factor head $\delta_k(x, t, c_k)$ per semantic factor, summed additively, where each head sees only its own corresponding factor c_k (Fig. 1).

Our contributions are summarized as follows:

1. **An end-to-end factored architecture:** We propose a novel conditional flow matching architecture that improves zero-shot combinatorial generalization on Shapes3D and MPI3D-real over a parameter-matched monolithic architecture (Sec. 3.1).
2. **A slice-attack ablation:** We compare our method against a compositional classifier-free guidance (CFG) (Ho & Salimans, 2022) technique inspired by (Liu et al., 2022) that allows a monolithic flow matching model to modulate the conditioning weight per factor (Sec. 3.2).
3. **Modular factor head extension:** We show that a new factor head can be added to a frozen K-factor model and trained alone, producing a working $(K+1)$-factor model without retraining the backbone or any existing head – an unattainable practice with a monolithic model (Sec. 3.3).

2. Framework

2.1. Conditional Flow Matching

We work in the conditional flow matching (CFM) framework (Lipman et al., 2023), where a neural velocity field $v_\theta : \mathbb{R}^d \times [0, 1] \times \mathcal{C} \to \mathbb{R}^d$ defines a probability path from a reference p_0 (typically a tractable probability distribution like an isotropic Gaussian) to a target p_1 conditioned on $c \in \mathcal{C}$, and is trained by regressing against the conditional vector field along a chosen interpolant. To generate a sample, we integrate $\dot{x}_t = v_\theta(x_t, t, c)$ from $t = 0$ to $t = 1$ with $x_0 \sim p_0$. We establish c as an element in a K-factor conditioning space $\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_K$, with $c = (c_1, \ldots, c_K)$.

2.2. Factored Architecture

We parameterize the conditional velocity as

$$v_\theta(x, t, c) \;=\; v_{\text{base}}(x, t) \;+\; \sum_{k=1}^{K} \delta_k(x, t, c_k), \qquad (1)$$

where v_{base} has no factor input and each per-factor head δ_k ingests only its assigned factor c_k. We instantiate each δ_k as a FiLM head (Appendix A) at the backbone bottleneck,

$$\delta_k(x, t, c_k) \;=\; \gamma_k(c_k) \odot h(x, t) + \beta_k(c_k),$$

where γ_k, β_k are linear projections of c_k, and $h(x, t)$ is the backbone bottleneck activation. All of these per-head modulations are then summed at the bottleneck (Fig. 1).

3. Experiments

3.1. Factored vs. monolithic conditioning on held-out combinations

Setup. We evaluate our architecture on two image datasets, each of which has labels across various factors: Shapes3D (Burgess & Kim, 2018) and MPI3D-real (Gondal et al., 2019). Shapes3D features synthetic three-dimensional shapes with variations in floor hue, wall hue, object hue, scale, shape, and orientation. MPI3D-real is composed of real photographs of different objects with variations in object colour, object shape, object size, camera height, background colour, horizontal axis, and vertical axis. Both methods share a 64×64 UNet base with FiLM heads parameter-matched at $\sim$ 7.5M (mono) / $\sim$ 7.6M (factored) and train end-to-end with `factor_dropout=0` (full architecture and training hyperparameters in App. C).

To evaluate the compositional generalization ability of a model, we choose a holdout subset of conditioning combinations $\mathcal{C}_h \subset \mathcal{C}$, train the model on the complement $\mathcal{C} \setminus \mathcal{C}_h$, and then determine if it can accurately generate samples with conditioning $c \in \mathcal{C}_h$. We establish three different holdout levels based on how much of the training dataset they remove: light (25%), medium (50%), and heavy (75%). The holdout is defined over the (object hue, shape) and (object colour, object shape) lattices for Shapes3D and MPI3D-real, respectively. More details regarding the holdout selection procedure are available in App. D.

Results. The factored architecture matches or beats monolithic on both heldout accuracy and FID at every holdout strength on both datasets (Tab. 1). In the light holdout regime, both methods saturate combo accuracy – the joint distribution is dense enough on the seen split that monolithic conditioning generalizes without architectural help – but factored already shows a modest heldout-FID advantage. In the medium regime, the factored architecture achieves its largest gap with the monolithic architecture in terms of heldout accuracy. In the heavy regime, both methods take a heavy hit in performance, but the factored architecture maintains its advantage, particularly in heldout FID, indicating a more graceful degradation in generation quality. This is reflected qualitatively in Fig. 2: on Shapes3D, monolithic samples often have incorrect object colours, while on MPI3D they also show shape errors; the factored architecture produces more faithful heldout combinations overall, though minor artefacts remain, such as the lime capsule artefact in Shapes3D and imperfect geometry for the white cylinder, brown cube, and green pyramid in MPI3D. The

Table 1. **Zero-shot combinatorial holdout, end-to-end training.** Heldout factor accuracy: classification accuracy on 100 generated samples per heldout combination at 100 ODE steps; seen/heldout FID computed with cleanfid on 10K samples per side. **Shapes3D** (K=6): factored matches or beats monolithic on every metric at every holdout strength (seed-std order reverses on *heavy* ZS combo, discussed in §4). **MPI3D-real** (K=7): factored continues to beat monolithc on every metric.

Dataset	Pattern	Method	Heldout Factor Accuracy	Seen FID	Heldout FID
Shapes3D (K=6)	light (25%)	mono	1.000	3.84 ± 0.91	3.94 ± 0.89
		factored	1.000	$\mathbf{3.36 \pm 0.33}$	$\mathbf{3.53 \pm 0.51}$
	medium (50%)	mono	0.754 ± 0.130	4.39 ± 0.32	12.49 ± 1.83
		factored	$\mathbf{0.987 \pm 0.018}$	$\mathbf{4.03 \pm 0.26}$	$\mathbf{5.31 \pm 0.45}$
	heavy (75%)	mono	0.221 ± 0.047	4.73 ± 0.53	22.76 ± 1.00
		factored	$\mathbf{0.282 \pm 0.130}$	$\mathbf{4.08 \pm 0.22}$	$\mathbf{9.55 \pm 0.78}$
MPI3D-real (K=7)	light (25%)	mono	0.9111 ± 0.077	3.04 ± 0.18	5.71 ± 0.17
		factored	$\mathbf{0.9186 \pm 0.0033}$	$\mathbf{2.86 \pm 0.09}$	$\mathbf{5.53 \pm 0.07}$
	medium (50%)	mono	0.4709 ± 0.0489	3.46 ± 0.14	5.77 ± 0.39
		factored	$\mathbf{0.7834 \pm 0.019}$	$\mathbf{3.33 \pm 0.09}$	$\mathbf{3.9 \pm 0.17}$
	heavy (75%)	mono	0.1399 ± 0.0489	5.48 ± 0.24	7.93 ± 0.18
		factored	$\mathbf{0.2841 \pm 0.0881}$	$\mathbf{5.3 \pm 0.11}$	$\mathbf{6.61 \pm 0.49}$

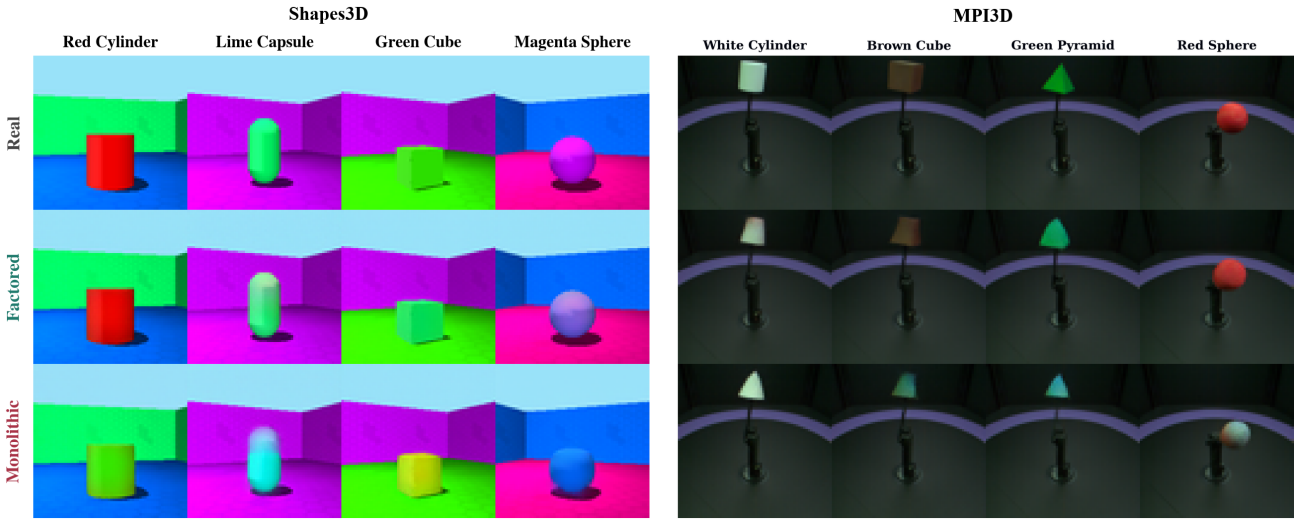


Figure 2. Heavy heldout combination samples for zero-shot compositional generalization. Samples from the factored and monolithic architectures are compared against real images on heldout factor combinations for (a) Shapes3D and (b) MPI3D-real.

same pattern carries from Shapes3D to MPI3D despite the synthetic-to-real shift and the K=6 $\rightarrow$ K=7 change in factor count.

3.2. Slice-attack ablation

Tab. 1 pits the factored architecture against *vanilla* monolithic conditioning, which uses a single global CFG weight w – a fair and direct comparison since neither model was trained for inference-time CFG amplification. To test the importance of the factored architecture, we aim to design a stronger monolithic baseline inspired by the Composable Diffusion framework (Liu et al., 2022) which trains a monolithic flow matching model with per-factor dropout and allows for per-factor CFG weighting at inference. Essentially, this technique (which we refer to as slice-CFG) trains a single monolithic head with factor dropout (each factor randomly masked at training time) so the model can be queried with arbitrary subsets of c. At inference, per-factor velocities are composed as

$$\hat{v}(x,t,c) \;=\; v(x,t,\emptyset) + \sum_{k=1}^{K} w_k \big[v(x,t,c_k) - v(x,t,\emptyset)\big], \tag{2}$$

where $v(x,t,c_k)$ is the velocity with only factor k active, $v(x,t,\emptyset)$ is the unconditional (all factors masked), and $w_k \geq 1$ controls factor k's amplification (w_k=1 recovers the conditional contribution and w_k>1 strengthens it).

We retrain the monolithic baseline in the medium and heavy holdout regimes for Shapes3D (same hyperparameters as in App. C) and at inference evaluate this

composition on the heldout combinations with weights $(w_{\texttt{shape}}, w_{\texttt{object_hue}}, w_{\text{others}}) = (3, 3, 1)$. By doing this, we amplify the holdout-active factors for the monolithic model. It is important to note that the factored architecture remains unchanged and continues to be free from inference-time augmentations.

Results. The slice-CFG monolithic approach is consistently outperformed by the factored architecture and is in fact worse than the vanilla monolithic model on heldout accuracy in both the medium and heavy regimes (Tab. 2). It seems the factor dropout regularization significantly hurts the generation quality to a degree which the per-factor CFG composition at w=3 cannot recover from. Interestingly, at the heavy holdout, the slice-CFG inference cuts the monolithic architecture's heldout FID in half, but is still worse than the factored architecture. This indicates that the performance gap induced by the factored architecture may not be easily surmountable through inference-time techniques.

3.3. Modular head extension: adding a factor without retraining

For a factored architecture, we demonstrate that a new factor head can be added to a frozen K-factor model and trained alone in order to produce a working $(K{+}1)$-factor model at the mere cost of a single small head's training run. Conversely, a monolithic architecture must retrain its entire set of parameters in order to add a new factor due to the entangled nature of its conditioning. Furthermore, unlike post-hoc composition methods such as Composable Diffusion, the extended model shares a single base velocity across all heads and generates samples with a single model's forward pass. We demonstrate this on Shapes3D (K=5 $\rightarrow$ K=6, adding the floor hue factor) and MPI3D-real (K=6 $\rightarrow$ K=7, adding the background hue factor).

Setup. For each dataset we train a K-factor factored base with one factor excluded, matching Sec. 3.1 but with no zero-shot holdout. We then freeze the backbone and the K existing heads, initialize a new factor head δ_{K+1} for the excluded factor, and train it alone for 30 epochs (full hyperparameters in App. C).

Table 2. Slice-attack ablation on Shapes3D over 3 seeds. *Vanilla* mono is the standard monolithic baseline from Tab. 1; *slice-CFG* mono uses a factor dropout probability of 0.1 and inference weights $(w_{\texttt{shape}}, w_{\texttt{object_hue}}, w_{\text{others}}) = (3, 3, 1)$.

Pattern	Method	Heldout factor acc.	FID
50%	mono, vanilla	0.754 ± 0.130	12.49 ± 1.83
	mono, slice-CFG	0.239 ± 0.118	13.82 ± 1.34
	factored	$\mathbf{0.987 \pm 0.018}$	$\mathbf{5.31 \pm 0.45}$
25%	mono, vanilla	0.221 ± 0.047	22.76 ± 1.00
	mono, slice-CFG	0.115 ± 0.024	11.26 ± 0.42
	factored	$\mathbf{0.282 \pm 0.130}$	$\mathbf{9.55 \pm 0.78}$

Results. The extension produces a $(K{+}1)$-factor controller with high combo accuracy and generation fidelity across the full $(K{+}1)$ lattice, achieving a (factor accuracy/FID) of (1.000, 3.75) and (0.954, 2.83) for Shapes3D and MPI3D-real, respectively. This shows that factor heads can be efficiently trained on top of frozen models with little-to-no drop in performance. Examples showing that the controlling over the new factor does not affect the original factors can be seen in App. G

4. Limitations

Currently the empirical scope of this investigation is image-domain conditional flow matching on two pixel-domain benchmarks. We hope to extend this framework to other modalities, such as molecules. The framework also currently assumes c comes as an explicit K-tuple $(c_1, \ldots, c_K)$ with a known semantic assignment of each component to a head, which is standard on the disentanglement benchmarks we evaluate on, but sidestepping the factor-discovery problem that motivates much of the slot-attention (Locatello et al., 2020) and routed-MoE literature (Fedus et al., 2022). This current formulation constrains the datasets for which this method would be applicable.

References

Burgess, C. and Kim, H. 3d shapes dataset. https://github.com/deepmind/3d-shapes, 2018.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 2022.

Gandikota, R., Materzyńska, J., Zhou, T., Torralba, A., and Bau, D. Concept sliders: Lora adaptors for precise control in diffusion models, 2023. arXiv:2311.12092.

Gondal, M. W., Wuthrich, M., Miladinovic, D., Locatello, F., Breidt, M., Volchkov, V., Akpo, J., Bachem, O., Schölkopf, B., and Bauer, S. On the transfer of inductive bias from simulation to the real world: A new disentanglement dataset. In *Advances in Neural Information Processing Systems*, 2019.

Ho, J. and Salimans, T. Classifier-free diffusion guidance, 2022. arXiv:2207.12598.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Liu, N., Li, S., Du, Y., Torralba, A., and Tenenbaum, J. B. Compositional visual generation with composable diffusion models. In *European Conference on Computer Vision*, 2022.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems*, 2020.

Montero, M. L., Bowers, J. S., Costa, R. P., Ludwig, C. J. H., and Malhotra, G. Lost in latent space: Examining failures of disentangled models at combinatorial generalisation. In *Advances in Neural Information Processing Systems*, 2022.

Okawa, M., Lubana, E. S., Dick, R. P., and Tanaka, H. Compositional abilities emerge multiplicatively: Exploring diffusion models on a synthetic task. In *Advances in Neural Information Processing Systems*, 2023.

Parmar, G., Zhang, R., and Zhu, J.-Y. On aliased resizing and surprising subtleties in gan evaluation. In *CVPR*, 2022.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

Schott, L., von Kügelgen, J., Träuble, F., Gehler, P., Russell, C., Bethge, M., Schölkopf, B., Locatello, F., and Brendel, W. Visual representation learning does not generalize strongly within the same domain. In *International Conference on Learning Representations*, 2022.

Zhang, L., Rao, A., and Agrawala, M. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

A. FiLM Conditioning

Feature-wise Linear Modulation (FiLM) (Perez et al., 2018) injects a conditioning signal into an intermediate feature map of a network via a per-channel affine transform: given a feature tensor h with C channels and a conditioning vector c,

$$\mathrm{FiLM}(h \mid c) \;=\; \gamma(c) \odot h + \beta(c),$$

where $\gamma(c), \beta(c) \in \mathbb{R}^C$ are learned (small MLP) projections of c and $\odot$ broadcasts over the spatial dimensions of h. FiLM is used for both the monolithic baseline and the per-factor heads of Sec. 2.2.

B. Architecture

The base velocity v_{base} is a 64×64 pixel-domain UNet with the structure summarized in Tab. 3. The downsampling path has three stages with channel multipliers $\{1, 2, 4\}$ on top of the base channel width `base_ch=64`, one residual block per stage, and a single mid-block at the bottleneck. The upsampling path mirrors this structure and consumes skip connections from the downsampling path. Time t is encoded with a sinusoidal embedding into a 256-dim feature followed by a two-layer MLP, then injected into every residual block via FiLM modulation.

Monolithic head. A single FiLM head with hidden width 1536 ingests the full one-hot factor vector (57-dim on Shapes3D, $K{=}6$; 40-dim on MPI3D-real, $K{=}7$) through a two-layer MLP and outputs scale and shift parameters that modulate the bottleneck residual block's output. Total head parameters: ~ 7.5M.

Factored heads. One FiLM head per factor with hidden width 256. Each head ingests only its own centered one-hot $\tilde{c}_k = c_k - \bar{c}_k$ and outputs scale/shift parameters that are summed into the bottleneck residual block's output. Total head parameters: ~ 7.6M (parameter-matched to monolithic within 0.1M). The conditioning linear projections inside each head retain their default biases, in keeping with the standard FiLM implementation.

Table 3. Backbone and head sizing summary. Backbone: UNet, `base_ch=64`, channel multipliers $\{1, 2, 4\}$, 1 residual block per stage; shared between mono and factored. Head parameter counts are matched within 0.1M.

Component	Mono	Factored
Input dim (Shapes3D, $K{=}6$)	57	$c_{k,\text{dim}}$ per head
Input dim (MPI3D-real, $K{=}7$)	40	$c_{k,\text{dim}}$ per head
Hidden width	1536	256 per head
Number of heads	1	K
Bias on cond. linears	yes	yes (default)
Total head params (Shapes3D)	~ 7.5M	~ 7.6M

C. Training

Both methods are trained end-to-end (backbone + head(s)) on the seen split of each holdout pattern, with identical optimizer settings unless otherwise noted. The modular extension (Sec. 3.3) trains δ_{K+1} alone for 30 epochs against the frozen heads, using the same optimizer settings.

Table 4. Training hyperparameters (shared by mono and factored unless noted).

Setting	Value
Optimizer	AdamW
Learning rate (peak)	2×10^{-4}
LR schedule	cosine to 1×10^{-5}
Warmup steps	5,000
Total steps	100,000
Global batch size	256
Gradient clip	1.0
EMA decay	0.999
Time sampling	logit-normal (σ=1)
Factor dropout (headline)	0
Factor dropout (slice-attack)	0.1
Seeds (Shapes3D headline)	$\{0, 42, 123\}$

D. Holdout Patterns

For each dataset the holdout is parameterized by indexing the 2-tuple lattice and partitioning it via a modular formula. On Shapes3D we index $(s, h) \in \{0, 1, 2, 3\} \times \{0, \ldots, 9\}$ over the (shape, object hue) lattice (40 cells total). A cell (s, h) is held out under *light* if $(s + h) \bmod 4 = 0$ (10 cells, 25%); under *medium* if $(s + h) \bmod 2 = 1$ (20 cells, 50%); and under *heavy* if $(s + h) \bmod 4 \neq 0$ (30 cells, 75%). The *medium* pattern is the only one that preserves both per-factor marginals exactly: every shape value is paired with exactly 5 of 10 object hue values in the seen split, and every object hue value with exactly 2 of 4 shape values. *Light* and *heavy* are complements, with both leaving each per-factor marginal mostly covered but not exactly balanced. For MPI3D-real, the analogous formulas on the 6×6 lattice over (object color, object shape) yield 9, 18, and 27 heldout cells out of 36 for *light*, *medium*, and *heavy*, respectively.

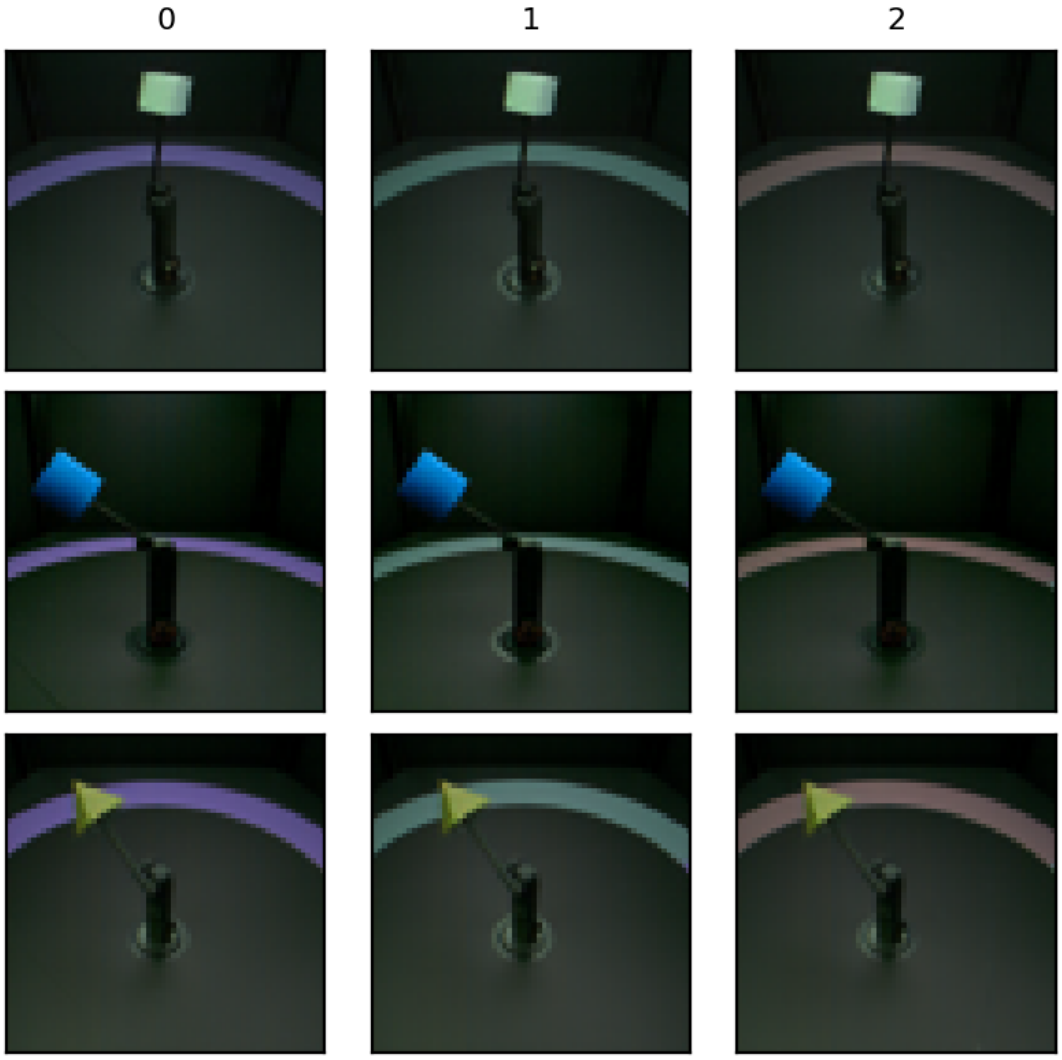
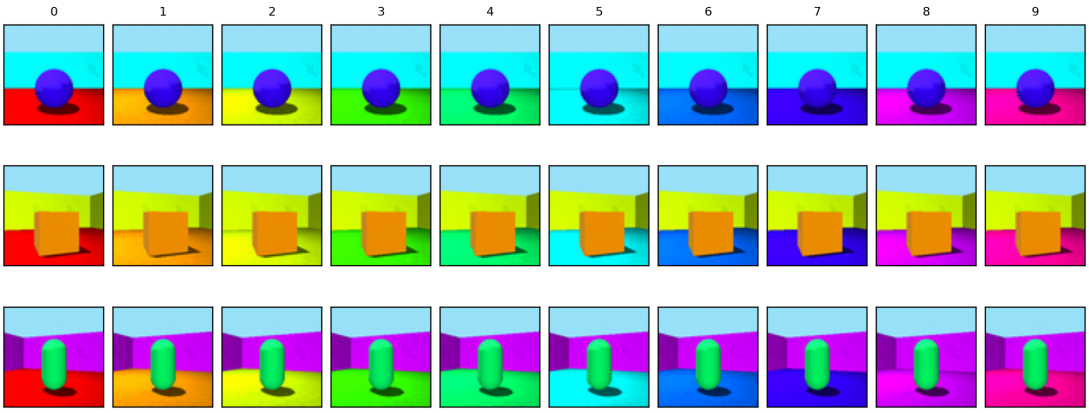
E. Sampling and Evaluation

Sampling. We integrate $\dot{x}_t = v(x_t, t, c)$ from t=0 to t=1 via Euler integration with 100 steps, using EMA-averaged backbone+head(s) weights at inference (EMA decay 0.999). Paired comparisons across mono and factored use matched initial noise $x_0 \sim \mathcal{N}(0, I)$ between rows.

Zero-shot heldout accuracy. For each heldout factor combination $\bar{c}$ we sample 100 images under that combination (random nuisance settings on factors orthogonal to the holdout pair) and classify each generated image with a CNN classifier trained on the seen split of the dataset (one head per factor). Zero-shot heldout accuracy is the fraction of generated images for which the classifier returns the requested values on both holdout-pair factors jointly.

FID. Seen and heldout FID are computed with the cleanfid implementation (Parmar et al., 2022) on 10,000 generated samples versus 10,000 ground-truth samples drawn from the corresponding split (seen or heldout).

F. Compute Infrastructure

Each (dataset $\times$ method $\times$ pattern $\times$ seed) training run uses a single GPU (A100, L40S, or H100) and takes $3 - 4$ hours. Slice-attack mono retraining and modular-extension head training are likewise single-GPU but shorter.

G. Example Generations from the (K+1)–Factor Models