CompFlow: Composing Velocity Fields for Multi-Condition Generation

Luca Miglior [1] **Vincenzo Gervasi** [1] **Davide Bacciu** [1]

Abstract

Generating samples that satisfy many conditions simultaneously can be achieved with a surprisingly simple operation: summing conditional velocity fields at inference time. We introduce *CompFlow*, a flow-matching framework that enables fully compositional inference without retraining, architectural changes, or specialized samplers. We show that velocity addition implements a *Product-of-Experts* composition, extending classifier-free guidance to arbitrarily many conditions. On CLEVR, a single-object-conditioned model is composed at inference to *simultaneously* control shape, color, and position for up to five objects, achieving 99.1–86.5% per-object accuracy with 30× fewer network evaluations than prior baselines. On high-resolution images, CompFlow satisfies up to five conditions simultaneously and substantially outperforms state-of-the-art single-prompt composition. We believe compositional generation can become a standard inference-time capability to control complex generation scenarios.

1. Introduction

Generative modeling has advanced rapidly across domains, driven in large part by diffusion (Ho et al., 2020; Sohl-Dickstein et al., 2015) and score-based (Song & Ermon, 2019) approaches. However, maintaining high performance across increasingly diverse and complex generative tasks has come at a growing cost (Kaplan et al., 2020), often requiring large-scale datasets, more complex architectures, and substantial computational resources for training or fine-tuning. Compositional generation offers a natural path toward modular controllable generation: rather than training a separate model for every combination of attributes, one would like to combine conditional experts at inference time. Despite its appeal, compositionality remains a key open problem in current research. Given a model trained to condition on one (simple) attribute at a time, is it possible to generate complex samples satisfying multiple conditions simultaneously, without retraining or expensive sampling strategies? In this work, we answer this question affirmatively. We introduce **CompFlow**, a training-free compositional generative framework that combines conditional experts directly at inference time. Instead of learning or fine-tuning a new model for every conjunction of conditions, CompFlow reuses a model conditioned on single simple conditions and composes its conditional velocity fields during sampling. This yields a deterministic and computationally efficient ODE sampler capable of enforcing multiple heterogeneous constraints simultaneously, without retraining, fine-tuning, or costly sampling strategies. As shown in Figure 1, this enables zero-shot composition of several conditions in a single generation process, while preserving sample quality and controllability.

This problem has received growing attention as recent literature (Huang et al., 2023b; Feng et al., 2023; Chefer et al., 2023) documented systematic failures of standard generative models on compositional tasks, particularly those involving complex relationships and simultaneous attribute matching (Huang et al., 2023b). Early approaches to compositional learning lay their foundations in Mixture-of-Experts (MoE) (Jacobs et al., 1991) architectures or, more recently, in Energy-Based Models (EBMs) (LeCun et al., 2006; Du & Mordatch, 2019; Du et al., 2021), as they naturally implement a Product-of-Experts (PoE) (Hinton, 1999; Mayraz & Hinton, 2000; Hinton, 2002) through the summation of energy functions, as demonstrated in previous work on compositional visual generation (Du et al., 2020; Liu et al., 2021). More recent methods have extended this paradigm to diffusion- and score-based models and introduced specialized samplers that correct the score trajectories of composed distributions at inference time (Liu et al., 2022b; Du et al., 2023; Yu & Gao, 2025). However, these approaches come at a significant cost: training EBMs is notoriously unstable, and often rely on Markov Chain Monte Carlo (MCMC) samplers, such as Unadjusted Langevin Dynamics (ULA) (Du & Mordatch, 2019; Nijkamp et al., 2020) or Hamiltonian Monte Carlo (HMC) samplers (Neal, 1996; Geffner & Domke, 2021; Duane et al., 1987; Neal, 1997) that require careful hyperparameter tuning and thousands of network evaluations per sample.

[1]Department of Computer Science, University of Pisa, Italy. Correspondence to: Luca Miglior <luca.miglior@phd.unipi.it>.

Accepted to the 2nd Workshop on Compositional Learning at ICML 2026, Seoul, South Korea. *Copyright 2026 by the author(s).*

Figure 1. **Zero-shot compositional generation with CompFlow.** Examples of images generated by CompFlow composing four separate prompts (see Section 3 for model details). For example, the first image composes velocity fields for four separate prompts: [a cute sleeping dragon + a pillow-book + a mystical forest + blue-ish dreamlike atmosphere]. Prompts and details are available in Appendix C.1.

Recently, *Flow matching* (Lipman et al., 2023) (FM) emerged as a prominent generative modeling paradigm. In particular, FM learns a time-dependent velocity field $v_\theta(t, \boldsymbol{x})$ that continuously transports a simple base distribution p_0 to a data distribution p_1 via an ODE, offering deterministic sampling at a fraction of the cost of diffusion reverse-SDE integrators.

Contributions.

1. We establish the theoretical basis for compositional Flow Matching by showing that, for Gaussian interpolants, conditional velocity residuals $v_t^c - v_t^0$ are proportional to gradients of noised conditional likelihoods. This connects flow-matching velocities to the score- and energy-based quantities traditionally used for compositional generation.

2. We show that weighted addition of conditional velocity residuals implements a tempered Product-of-Experts density in the noised space, providing an exact interpretation of velocity-field arithmetic and generalizing classifier-free guidance from a single condition to arbitrarily many independently weighted conditions. An equivalent composition was obtained independently by Palma et al. (2025) for latent single-cell flows; we establish that the same principle applies *zero-shot* to large pretrained image backbones, and additionally make the continuity-equation guarantee explicit (Theorem 2.3).

3. We validate CompFlow across controlled and open-domain settings. On synthetic two-dimensional targets, velocity addition recovers the expected PoEs composition. On CLEVR, a single-object-conditioned UNet (Ronneberger et al., 2015) composes simultaneous conditions on shape, color and position, up to five objects. On text-to-image generation, CompFlow composes atomic text prompts with a pretrained flow-matching backbone, improving compositional alignment over single-prompt generation.

2. Additive Flow composition

We now present **CompFlow**, a training-free method for composing conditional flow-matching models at inference time. The goal is the following: given access to a model that can condition on individual attributes, objects, or semantic concepts, we would like to generate from their joint composition without retraining the model or introducing a new sampler. CompFlow achieves this by performing the corresponding compositional arithmetic directly in velocity-field space.

2.1. Background: Conditional Flow Matching preliminaries

Let $p_0 = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ be a simple base distribution and let p_1 denote the data distribution. Flow Matching learns a time-dependent velocity field whose associated ODE

$$\frac{d\boldsymbol{x}}{dt} = v_t(\boldsymbol{x}), \qquad \boldsymbol{x}(0) \sim p_0, \tag{1}$$

transports samples from p_0 to p_1 at time $t = 1$. In practice, the marginal velocity field v_t that realizes a desired probability path p_t is generally intractable, since it requires averaging over all possible data points.

Conditional Flow Matching (Lipman et al., 2023) avoids this intractability by training on sample-level conditional paths. Given $x_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $x_1 \sim p_1$, we use the Gaussian interpolant

$$x_t = t\, x_1 + \sigma_t\, x_0, \qquad \sigma_t := 1 - (1 - \sigma_{\min})t, \tag{2}$$

where $\sigma_{\min} \geq 0$. The corresponding sample-level velocity is

$$u_t(x_0, x_1) := \frac{dx_t}{dt} = x_1 - (1 - \sigma_{\min})x_0. \tag{3}$$

Although the marginal velocity is obtained by averaging this quantity over all training samples compatible with the current state $x_t = x$, the regression objective

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,x_0,x_1}\left[\|v_\theta(t, x_t) - u_t(x_0, x_1)\|^2\right] \tag{4}$$

has the same optimum as the intractable marginal objective. Thus, at convergence, the network approximates the marginal velocity

$$v_t(x) = \mathbb{E}\left[u_t(x_0, x_1) \mid x_t = x\right]. \tag{5}$$

At convergence, the learned velocity field therefore induces the desired probability path and obeys the associated continuity equation (Villani, 2009). For conditional generation, the same construction is applied under a condition c. We denote by $p_t(x \mid c)$ the noised density induced by the interpolant when x_1 is drawn from the conditional data distribution, and by

$$v_t^c(x) = \mathbb{E}\left[u_t(x_0, x_1) \mid x_t = x, c\right] \tag{6}$$

the corresponding exact conditional marginal velocity. During training, we also learn an unconditional velocity v_t^0 by replacing the condition with a null token, as in classifier-free guidance (Ho & Salimans, 2022; Zheng et al., 2023). These conditional and unconditional fields are the basic objects that CompFlow composes.

2.2. Compositional Flow Matching (CompFlow)

Velocity Fields Encode Scores. Whilst composition is straightforward for energy-based and score-based models because energies and scores add naturally (Du et al., 2020), flow-matching models expose velocities rather than energies or scores. The key observation underlying CompFlow is that, for Gaussian probability paths, these quantities are directly related: the marginal velocity is an affine function of the score of the noised density. We use the following velocity–score identity from (Zheng et al., 2023).

Lemma 2.1 (Velocity–score identity for Gaussian paths)**.** *Following Zheng et al., 2023, we specialize the general velocity-score identity for Gaussian probability paths. Let us consider a probability path*

$$p_t(x \mid y) = \mathcal{N}(x \mid \alpha_t y, \sigma_t^2 I)$$

with noise scheduler (α_t, σ_t). Let $p_t(x \mid c)$ be the marginal density obtained by averaging over data points satisfying condition c, and let

$$s_t^c(x) := \nabla_x \log p_t(x \mid c)$$

be its conditional score for a condition c. Then the corresponding marginal velocity field satisfies

$$v_t^c(x) = a_t x + b_t s_t^c(x). \tag{7}$$

For the interpolant in Equation (2), we have $\alpha_t = t$ and $\sigma_t = 1 - (1 - \sigma_{\min})t$. A direct substitution gives $a_t = \frac{1}{t}, b_t = \frac{\sigma_t}{t}$. Therefore, for every condition c (including the null condition) and every $t \in (0, 1]$ we can express the conditional velocity field in terms of the (Gaussian) score function of the conditional probablity:

$$v_t^c(x) = \frac{1}{t}x + \frac{\sigma_t}{t}\nabla_x \log p_t(x \mid c). \tag{8}$$

Interestingly, the term $\frac{1}{t}x$ is shared by all conditions, so subtracting the unconditional velocity cancels the common drift and isolates the condition-specific contribution. We observe that the guided velocity field concretely implements the likelihood gradient of the guidance signal:

$$v_t^c(x) - v_t^0(x) = \frac{\sigma_t}{t}\nabla_x \log p_t(c \mid x). \tag{9}$$

Thus, we have proven that the conditional velocity residual $v_t^c - v_t^0$ is proportional to the gradient of the noised conditional likelihood. This is precisely the quantity that appears in classifier-free guidance, now expressed in velocity-field form. Full derivation can be found in Appendix A.

Velocity Addition Implements a Product of Experts. Equation (9) shows that conditional velocity residuals behave like likelihood-gradient experts. CompFlow composes multiple conditions by adding these residuals, generalizing and extending the CFG guidance for conditional flow matching. Following the well-established Product-of-Experts compositional formulation, we aim to implement a flow-based version using the results obtained in Equation (9). We now state the equality below.

Theorem 2.2 (Product-of-Experts composition for Gaussian Flow Matching)**.** *Consider a Gaussian flow-matching probability path with unconditional noised density $p_t(x)$, conditional noised densities $p_t(x \mid c_i)$, and conditional likelihoods $p_t(c_i \mid x)$. Let $v_t^0(x)$ denote the unconditional marginal velocity field and $v_t^{c_i}(x)$ the marginal velocity field conditioned on c_i. For guidance weights $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$, define the tempered Product-of-Experts density*

$$\tilde{p}_t(x) \propto p_t(x) \prod_{i=1}^{m} p_t(c_i \mid x)^{\lambda_i}. \tag{10}$$

Then the composed velocity field

$$v_t^{\text{comp}}(x) := v_t^0(x) + \sum_{i=1}^{m} \lambda_i \left[v_t^{c_i}(x) - v_t^0(x)\right] \tag{11}$$

is exactly the Gaussian-path velocity field associated with $\tilde{p}_t(x)$. Equivalently, for the Gaussian interpolant:

$$v_t^{\text{comp}}(x) = \frac{1}{t}x + \frac{\sigma_t}{t}\nabla_x \log \tilde{p}_t(x). \tag{12}$$

Moreover, if the conditions are conditionally independent given $x_t = x$, that is,

$$p_t(c_1, \ldots, c_m \mid x) = \prod_{i=1}^{m} p_t(c_i \mid x), \tag{13}$$

Algorithm 1 CompFlow sampling by velocity addition

Require: velocity network v_θ, condition encoder Enc, conditions $\{c_i\}_{i=1}^m$, weights $\{\lambda_i\}_{i=1}^m$, time grid $\{t_k\}_{k=0}^N$
  $x_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $\boldsymbol{e}_i \leftarrow \mathrm{Enc}(c_i)$ **for** $i = 1, \ldots, m$
  $\boldsymbol{e}_\varnothing \leftarrow$ null-condition embedding
  for $k = 0, \ldots, N-1$ **do**
    $\boldsymbol{v}_\varnothing \leftarrow v_\theta(t_k, x_k, \boldsymbol{e}_\varnothing)$
    for $i = 1, \ldots, m$ **do**
      $\boldsymbol{v}_i \leftarrow v_\theta(t_k, x_k, \boldsymbol{e}_i)$
      $\Delta\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i - \boldsymbol{v}_\varnothing$
    end for
    $\boldsymbol{v}_{\mathrm{comp}} \leftarrow \boldsymbol{v}_\varnothing + \sum_{i=1}^m \lambda_i \, \Delta\boldsymbol{v}_i$
    $x_{k+1} \leftarrow \text{ODE-STEP}(x_k, \boldsymbol{v}_{\mathrm{comp}}, t_k, t_{k+1})$
  end for
  return x_N

and $\lambda_i = 1$ for all i, then $\tilde{p}_t(x) \propto p_t(x \mid c_1, \ldots, c_m)$. In this case, $v_t^{\mathrm{comp}}(x)$ coincides with the exact joint conditional velocity field.

Corollary 2.3 (Continuity equation for the composed velocity)**.** *Assume noised conditional independence for all $t \in (0, 1]$, and let $\lambda_i = 1$ for all i. Denote $c_{1:m} = (c_1, \ldots, c_m)$. Then v_t^{comp} generates the conditional path $p_t(\cdot \mid c_{1:m})$ from $p_0 = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ to $p_1(\cdot \mid c_{1:m})$. In particular, it satisfies*

$$\partial_t p_t(x \mid c_{1:m}) + \nabla \cdot \Big(p_t(x \mid c_{1:m}) v_t^{\mathrm{comp}}(x) \Big) = 0. \tag{14}$$

All the proofs are provided in Appendix A.

Remark 2.4 (Relation to prior work). A Product-of-Experts composition equivalent to ours was derived independently by Palma et al. (2025) for latent single-cell flows. Whereas their instantiation requires training a dedicated latent flow per application, we show that the same principle applies *zero-shot* to large off-the-shelf pretrained text-to-image backbones, with no auxiliary flow, turning compositional control into a pure inference-time capability at the scale of modern image generators. Our contribution is thus distinct in both reach and execution: we scale velocity composition to five simultaneous heterogeneous conditions on CLEVR and provide the first quantitative compositional evaluation on T2I-CompBench (Huang et al., 2023b): the high-resolution, multi-attribute, training-free regime, settings in which the composition mechanism is non-trivial to realize and had not previously been demonstrated. We further make the continuity-equation guarantee explicit (Theorem 2.3).

Interpretation. The theorem shows that velocity addition is not merely a heuristic. For Gaussian flow-matching paths, adding conditional velocity residuals is equivalent to adding likelihood scores, and therefore implements the same Product-of-Experts composition used in energy-based modeling. The weights λ_i act as temperatures or guidance strengths for the individual experts. When $m = 1$, Equation (11) reduces to standard classifier-free guidance; for $m > 1$, it extends the same principle to an arbitrary number of independently specified conditions.

This result also clarifies why the unconditional velocity must be included. The conditional field v_t^c contains both the base dynamics shared by all samples and the contribution of condition c. The residual $v_t^c - v_t^0$ removes the shared component, leaving only the expert-specific likelihood gradient. CompFlow therefore composes experts by accumulating residuals around a common unconditional flow.

Inference. At inference time, we replace the exact fields in Equation (11) with the learned network v_θ. Given conditions $c_1, \ldots, c_m$, we encode each condition independently, evaluate all conditional velocities at the same state x_k and time t_k, and integrate the resulting composed field with a standard ODE solver. No additional model is trained, and no correction or rejection step is introduced.

The sample is then advanced by one ODE step using $v_\theta^{\mathrm{comp}}(t_k, x_k)$. Since each step requires one unconditional evaluation and one evaluation per condition, the total cost is $N(m+1)$ network evaluations for N solver steps. The procedure is summarized in Algorithm 1.

3. Experiments

Following prior work (Liu et al., 2022a; Yu & Gao, 2025), we investigate CompFlow generative capabilities across three complementary settings: (i) a synthetic 2D validation of the composition rule; (ii) compositional image generation on the CLEVR Dataset (Johnson et al., 2017) with joint control over shape, color, and position up to five objects; (iii) text-to-image generation capabilities. For the latter, we test how state-of-the-art pre-trained flow-matching models can compose multiple natural-language concepts at inference time. Architectures, training hyperparameters, text-to-image setup, and ablations are deferred to Appendix B and C.

3.1. Synthetic 2D Distributions

We first validate our method on a two-dimensional task where composition of distributions can be visually appreciated. Two independent flow matching models are trained on disjoint distributions: an 8-mode Gaussian ring (p_1) and a vertical bar (p_2). Neither model sees the other's data at training time. At inference, we implement Algorithm 1 with $\lambda_1 = \lambda_2 = 1$. As shown in Figure 2, the composed sampler recovers the intersection modes at $(0, \pm 1)$, according to Theorem 2.2. The composed velocity field (arrows) points consistently towards the two intersection modes, confirming that weighted velocity addition directly implements a PoEs composition in the noised density space.

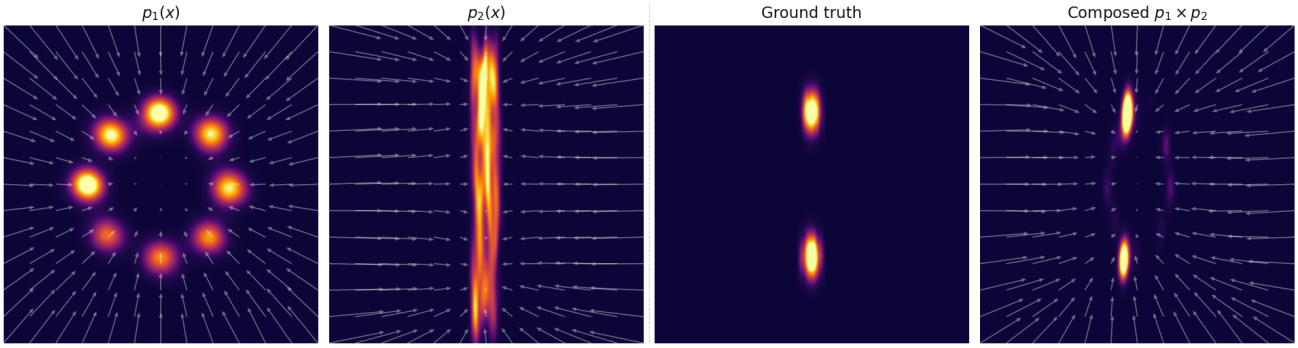

Figure 2. Synthetic 2D validation. Two flow matching models are trained independently on disjoint distributions. Arrows indicate the resulting composed velocity field.

3.2. CLEVR Multi-Object Generation

Setup. We train a single UNet-based flow matching model on CLEVR with *single-object* conditioning: each training example is a rendered scene, conditioned on one of its objects, specified by a vector attribute e encoding shape, color, and position.

This is the only supervision the model ever receives. At inference, we compose up to $m = 5$ single-object conditions simultaneously via Algorithm 1, jointly controlling shape, color, and position for each object. To our knowledge, no prior compositional generative framework has been evaluated under complex simultaneous control of this many heterogeneous attributes: existing methods either condition on position alone (Du et al., 2023; Yu & Gao, 2025), on a single attribute type, or do not scale beyond $m = 3$ conditions without significant quality degradation. Conditions are dropped to a learned null token with probability $p_{\text{drop}} = 0.15$ to enable CFG-style guidance, and we use $N = 20$ Euler ODE steps at inference. The compositional sampler is a standard deterministic ODE integrator, and no additional hyperparameters beyond one scalar guidance weight λ_i per condition are necessary.

Evaluation protocol. In CLEVR, prior work on compositional generation has primarily used segmentation-based evaluation with SAM2 (Ravi et al., 2024) to assess whether objects appear at the specified positions (Du et al., 2023; Yu & Gao, 2025). While sufficient for position-only conditioning, segmentation masks cannot verify whether a detected object matches additional complex attributes. CompFlow conditions jointly on a broader set of attributes, hence a more expressive evaluation protocol is needed. We therefore adopt open-vocabulary detection via GroundingDINO (Liu et al., 2024) as our primary metric: the detector is queried with natural-language attribute descriptions (e.g. "a red cube"), so a detection is only triggered when the generated object matches the full requested attribute set, not merely its spatial location. Detections are then matched to ground-truth coordinates via Hungarian matching, and a requested object is counted as correctly placed if its matched detection falls within $\tau = 0.1 \times width$ of the ground-truth canvas position, jointly verifying shape, color, and position in a single criterion. We additionally report DINOv2 feature similarity to a real-CLEVR reference set as a realism proxy, and mean pixel distance between matched detections and ground-truth coordinates as a spatial precision metric.[1]

Table 1. Compositional generation on CLEVR, position-only task, verified via SAM2. Scene-level accuracy: fraction of samples in which all m requested objects are correctly placed. †Reproduced from (Yu & Gao, 2025). *CompLift applies rejection sampling: candidates failing the lift criterion are discarded and resampled; NFE counts only accepted samples and excludes rejected candidates. At m=5, CompFlow uses $N(m+1) = 120$ forward passes.

	Scene-level accuracy (%) ↑					
Method	m=1	m=2	m=3	m=4	m=5	NFE ↓
EBM (ULA)†	86.0	74.0	48.0	27.0	19.0	∼4000
EBM (U-HMC)†	82.0	66.0	34.0	24.0	11.0	∼4000
CompLift†*	100.0	100.0	99.9	97.5	90.3	rejection*
Cached CompLift†*	100.0	100.0	99.7	91.6	84.3	rejection*
CompFlow (ours)	**96.0 ± 1.0**	**88.0 ± 0.7**	**82.0 ± 1.1**	**82.0 ± 1.0**	**66.0**	**≤ 120**

Main results. Table 2 reports per-object accuracy, mean pixel distance, and DINOv2 similarity for CompFlow on the full shape+color+position task. Per-object accuracy remains above 94% up to $m = 4$ and degrades gracefully to 86.5% at $m = 5$, while DINOv2 similarity stays essentially flat across all m (0.920 to 0.904), indicating that sample quality remains stable as compositional load increases. The Number of Function Evaluation (NFE) cost scales linearly as $N(m + 1)$, reaching 120 forward passes at $m = 5$.

Table 1 compares CompFlow with prior methods on the position-only CLEVR setting evaluated with SAM2 (Ravi et al., 2024). CompFlow outperforms all EBM-based base-

[1]To enable direct comparison with prior work, we additionally report accuracy under the same SAM2-based protocol, checking only whether the targeted canvas positions are occupied regardless of object identity.

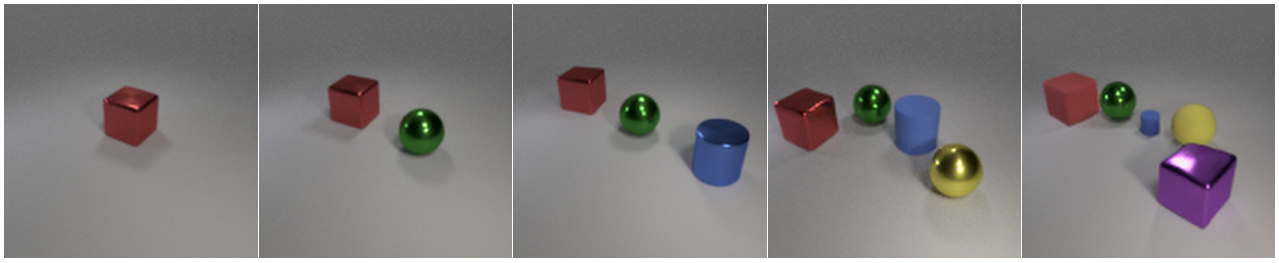

Figure 3. Progressive CLEVR composition with CompFlow. From left to right, we incrementally add object-level conditions to the same scene, composing velocity residuals for one to five requested objects. The model preserves the previously generated objects while incorporating each new shape–color–position constraint, despite being trained only with single-object conditioning.

Table 2. GroundingDINO evalutation of CompFlow on the full CLEVR compositional task.

	m (number of composed conditions)				
Metric	m=1	m=2	m=3	m=4	m=5
Position acc. (%) ↑	99.1	97.5	94.7	94.5	86.5
Mean dist. (px) ↓	4.80	5.32	6.57	6.18	9.07
DINOv2 sim. ↑	0.920	0.909	0.918	0.912	0.904
NFE	40	60	80	100	120

lines across every value of m, surpassing EBM (ULA) and EBM (U-HMC) by a substantial margin despite using at most 120 forward passes compared to their ∼4000. The only methods that exceed CompFlow's accuracy are CompLift and Cached CompLift, both of which operate as rejection samplers, discarding samples that fail a lift-score criterion. CompFlow instead enforces compositional constraints through its single deterministic ODE trajectory. Where sampling budget is limited or latency matters, our single-pass, high accuracy method is preferable to rejection-based approaches. A qualitative sample is provided in Figure 3, where CompFlow composes five objects conditions simultaneously. We direct the reader to Appendix B for additional visualization and benchmarks and to Appendix B.2 for a insightful discussion about composition failure models.

3.3. Text-to-Image Compositional Generation

Having validated CompFlow on controlled synthetic and rendered benchmarks, we now move to open-domain text-to-image generation. Prior work on compositional diffusion models (Liu et al., 2022b; Du et al., 2023) has reported only qualitative results in this setting, leaving the quantitative picture largely uncharted. We close this gap by evaluating CompFlow on top of `FLUX.1[dev]` (Labs et al., 2025; Labs, 2024), a state-of-the-art flow-matching model, and provide quantitative scores on the T2I-CompBench benchmark (Huang et al., 2023b). The prompts for the image generation task are taken from the non-spatial subset of the benchmark and underwent a pre-processing step before being used to generate images. In the following paragraphs we provide quantitative and qualitative comparisons that illustrate the compositional failure modes of single-prompt generation. A comprehensive overview of the experimental setup for T2I generation is detailed in Appendix C.1.

Setup. We use `FLUX.1[dev]` as the backbone velocity network v_θ. Given a prompt from the test-set of T2I-CompBench $c = c_1 \wedge c_2 \wedge \cdots \wedge c_m$ (e.g. *"a woodcarver and a bird sculpture from a wood block"*), we decompose it into m atomic concept strings $\{c_i\}_{i=1}^m$ using a large language model (LLM). Concretely, we prompt the LLM to return the minimal set of noun-phrase conditions whose conjunction recovers the full semantics of y, without rephrasing or adding information not present in the original prompt. Each condition c_i is then encoded independently by the `FLUX.1[dev]` text encoder, yielding m conditioning embeddings $\{e_i\}_{i=1}^m$ alongside the null embedding $e_\varnothing$ used for unconditional generation. We refer the reader to Appendix C.2 for the full prompt decomposition pipeline used to generate the single-concept prompts.

At each step k of the ODE integrator we compute the composed velocity field described in Equation (11) where $\lambda_i = 1$ for all i unless stated otherwise, following Algorithm 1. This requires $m{+}1$ forward passes per ODE step (one unconditional plus one per condition), for a total of $N(m{+}1)$ network evaluations with $N = 28$ ODE steps. The baseline (`FLUX.1[dev]` *single-prompt*) generates images by conditioning on the full composed string y directly, using the same number of steps N and guidance scale. No additional training, fine-tuning, or rejection sampling is performed in either setting.

T2I-CompBench evaluation. We evaluate on the non-spatial subset of T2I-CompBench (Huang et al., 2023b) and include results from T2I-CompBench++ (Huang et al., 2023a) for comparison with recent text-to-image models.

Following the recent T2I-CompBench++, we use ShareGPT4V (Chen et al., 2023) (Share-CoT) image–text

"a dog playing with its owner in a park" [a dog playing + its owner + a park]

"The woodcarver is creating a sculpture of a bird from a block of wood" [The woodcarver + the sculpture of a bird from a block of wood]

Figure 4. Qualitative comparison between vanilla `FLUX.1[dev]` and CompFlow on non-spatial compositional generation. In each pair, the left image is generated by conditioning `FLUX.1[dev]` on the full prompt as a single text string, while the right image is generated by decomposing the same prompt into atomic concept strings and composing the corresponding conditional velocity fields with CompFlow.

Table 3. Compositional text-to-image evaluation using Share-CoT and CLIP metrics.

Model	Share-CoT	CLIP
Stable v1-4 (Rombach et al., 2021)	0.7487	0.3079
Stable v2 (Rombach et al., 2021)	0.7567	0.3127
Composable v2 (Liu et al., 2022b)	0.6927	0.2980
Structured v2 (Feng et al., 2023)	0.7560	0.3111
Attn-Exct v2 (Chefer et al., 2023)	0.7593	0.3109
GORS-unbiased (Huang et al., 2023b)	0.7650	0.3158
GORS (Huang et al., 2023b)	0.7637	0.3193
Stable XL (Podell et al., 2024)	0.7673	0.3119
PixArt-α-ft (Chen et al., 2024)	0.7747	0.3197
DALL·E 3 (Betker et al., 2023)	0.7853	0.3003
Stable v3 (Esser et al., 2024)	0.7782	0.3140
FLUX (Labs et al., 2025)	0.7809	0.3127
CompFlow (ours)	$\mathbf{0.8021 \pm 0.011}$	$\mathbf{0.3201 \pm 0.002}$

similarity as the primary automatic metric for non-spatial relationships. This choice is motivated by the finding that MLLM-based metrics are better suited to evaluate compositionality, where fine-grained attribute binding is fundamental. For completeness and compatibility with earlier benchmarks, we also report CLIP (Radford et al., 2021; Hessel et al., 2021) and BLIP-CLIP (B-CLIP) (Chefer et al., 2023); however, we treat them as secondary diagnostic metrics, since they mainly capture coarse image–text alignment and are less sensitive to fine-grained correspondences. For each benchmark prompt, we apply the LLM decomposition described above to obtain atomic concept conditions, run CompFlow with Eq. (11), and score the resulting image against the original *full* prompt rather than the decomposed strings. This setup ensures a fair comparison and guarantees that CompFlow and all baselines are evaluated against the same semantic target. Under the Share-CoT metric, CompFlow obtains the strongest score among all compared methods, improving over the `FLUX.1[dev]` single-prompt baseline from 0.7809 to 0.8021. This gain suggests that decomposing a prompt into atomic concepts and composing their velocity residuals improves the model's ability to satisfy multiple semantic constraints simultaneously, rather than merely increasing global prompt affinity. CLIP provides a secondary consistency check. CompFlow reaches a CLIP score of 0.3201, remaining competitive with or above the strongest reported compositional baselines, including Composable Diffusion (Liu et al., 2022b), The fact that the largest improvement appears on the MLLM metric rather than only on CLIP, indicates that the benefit is not simply an artifact of CLIP-based scoring, but is also visible under a stronger semantic evaluator. Importantly, these gains are obtained purely at inference time, maintaining the high efficiency of flow-based generative models. We refer the reader to Appendix C.2 for additional details on sampling times and setup. Interestingly, we also investigated dynamics-induced style mixing and we detail results of these experiments in Appendix 3.4.

Qualitative analysis. Figure 4 compares vanilla `FLUX.1[dev]` and CompFlow on two representative compositional prompts. Within each pair, the left image is produced by the `FLUX.1[dev]` single-prompt baseline, which receives the entire prompt as one text input, while the right image is produced by CompFlow, which decomposes the same prompt into atomic concept strings and composes their conditional velocity fields during sampling. In the dog example, the full prompt given to `FLUX.1[dev]` is *"A dog is wagging its tail and playing with its owner"*. CompFlow instead receives three concept-level conditions: *"a dog playing"*, *"a dog and its owner"*, and *"a dog and its owner in a park"*. The single-prompt baseline produces a plausible dog scene, but the different semantic requirements are represented only implicitly through one global text embedding. CompFlow makes the individual constraints explicit: the dog, the owner and the park context are each introduced as separate experts and combined in the

a horse | + grass field | + pink sky | + anime style | + the big full moon

Figure 5. Progressive concept composition with CompFlow. Starting from *"a horse"*, we incrementally add styling and objects prompts shown below each image. The scene evolves smoothly, incorporating each new semantic or stylistic constraint while preserving the previously generated content.

velocity field. In the woodcarver example, the full prompt given to `FLUX.1[dev]` is *"The woodcarver is creating a sculpture of a bird from a block of wood"*. CompFlow decomposes this into *"a woodcarver"* and *"the sculpture of a bird from a block of wood"*. This example probes a different kind of non-spatial composition: the model must jointly represent an agent and an object undergoing a process, rather than merely placing two nouns in the same scene. The CompFlow sample more explicitly preserves both semantic factors by maintaining the artisan and the bird sculpture as distinct, jointly satisfied concepts.

These examples illustrate the same phenomenon measured quantitatively in Table 3: composing per-concept velocity residuals improves the model's ability to satisfy multiple subject-level constraints simultaneously. Global image–text metrics such as CLIP can capture part of this improvement, while Share-CoT provides a more semantic evaluation signal that better reflects whether the generated image satisfies the intended non-spatial relation. Figure 5 further stresses the stability of the composition mechanism by incrementally adding conditions to the same sample trajectory; the progressive visualization provides a qualitative counterpart to the Share-CoT evaluation in Table 3. Since Share-CoT is designed to judge whether the image satisfies the semantic content of the prompt rather than only its global CLIP similarity, it is especially relevant for examples where multiple concepts must remain simultaneously active. The progressive sequence shows that CompFlow can accumulate several semantic and stylistic constraints without one expert dominating or destroying the others, supporting the Product-of-Experts interpretation of Eq. (11).

3.4. Dynamics-induced style mixing

The velocity-composition operator in Eq. (11) acts pointwise along a single latent trajectory and is symmetric under permutation of the (c_i, λ_i) pairs. This structure naturally encourages *style mixing*, reflecting the attribute-blending that we study in Appendix B, rows 1–2, but in a more coherent and visually graceful form. When several conditions provide compatible but distinct semantic, geometric, or stylistic directions, the composed trajectory does not necessarily instantiate each condition as a separate object. Instead, it may follow an averaged dynamical path that blends their visual attributes into a single coherent sample. We illustrate this behavior in Figure 6.

Style and attribute fusion. When two or more subprompts describe visually compatible concepts of comparable salience, the composed velocity field can interpolate between their associated directions instead of selecting one condition or creating multiple separate entities. For example, composing "a horse" and "a sheep" produces an animal with attributes of both species; composing "a sphere" and "a cube" yields a rounded cubic object; and composing "a glossy chair" with "a lamp" produces an object that mixes shape, material, illumination, and function. These samples are not degenerate: they are coherent images produced by following a single composed dynamical system. The behavior reflects the fact that CompFlow performs composition in velocity space, so compatible experts can jointly deform the trajectory toward an intermediate visual mode.

Implications for compositional control. This behavior is useful when the intended prompt involves global style, material, texture, color, or aesthetic modifiers, since the velocity addition can smoothly combine such factors during sampling. However, it also reveals a limitation for prompts requiring hard binding, counting, or explicit separation of instances. If the target semantics require "a horse and a sheep" as two distinct entities, the pointwise additive dynamics alone does not specify that the two concepts should occupy different spatial regions or correspond to different object slots. In such cases, the method may instead produce a single blended concept.

This suggests that velocity composition is especially well suited to style-level and attribute-level composition, while precise multi-object binding may require an additional mechanism, such as spatial conditioning, object slots, masks, or

horse + sheep | sphere + cube | glossy chair + lamp | green apple + bear

Figure 6. Dynamics-induced style mixing. When conditions are composed along the same latent trajectory, CompFlow often produces coherent visual blends rather than discrete object conjunctions. The resulting images combine semantic, geometric, material, and stylistic attributes from multiple experts into a single sample.

region-specific experts. These limitations are inherent to the velocity-additive form rather than artifacts of the FLUX backbone: the same averaging behavior would arise for any rectified-flow model under Eq. 11. Lifting them likely requires either a non-symmetric composition strategy or an explicit spatial prior on top of the per-condition velocities; we leave both directions to future work.

Spatial-relational failures. The same averaging dynamics that enables the style mixing above breaks down when the composed conditions encode *spatial-relational* constraints. Figure 7 reports a representative case obtained while attempting to reproduce a CLEVR-style scene with CompFlow: each sub-prompt anchors a single primitive to a specific half of the canvas, and the FLUX backbone handles the side assignment in isolation. Under additive composition, however, the dynamics discards the spatial binding, while the description of two simple primitives drives the same blending behavior described above; the result is a chimeric solid in which sphere-shaped patches appear painted onto the faces of a cube, rather than two distinct objects in their requested halves. We were unable to recover correct binding by varying λ, the seed, the resolution, or the number of ODE steps. The same mechanism is responsible for analogous failures on counting prompts (e.g. "a horse" composed with "a horse" or with "two horses" as a third condition does not yield two instances) and, more broadly, for any prompt whose intended semantics rely on a structured assignment of conditions to image regions or object slots. These failure regimes mark the boundary of what an additive operation can encode: spatial-relational composition appears to require an explicit spatial prior on top of the single-condition velocities, both of which fall outside the scope of this work, and are left as future development.

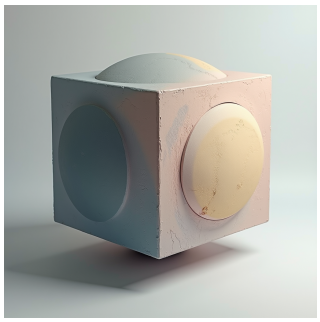

Figure 7. CompFlow output for the prompts *"a cube on the left"*The left/right assignment is destroyed and the generation collapse into a single hybrid solid.

4. Conclusion

We introduced **CompFlow**, a zero-shot compositional generation method based on Flow Matching. By showing that conditional velocity fields are affine in the noised score under Gaussian paths, we established that PoEs composition in the noised density space is theoretically grounded and practically effective: it transfers *zero-shot* to large pretrained text-to-image backbones, enforces many simultaneous heterogeneous conditions at high resolution without retraining, fine-tuning, or specialized samplers, and reduces network evaluations by up to 30$\times$ compared to prior baselines. CompFlow supports complex compositional scenarios surpassing existing benchmarks in both generation quality and controllability. Its interpretation is exact for Gaussian paths and independent experts; real prompts involving counting, spatial binding, or object relations may violate these assumptions, and visually compatible conditions may be blended rather than instantiated as separate entities. Future work includes studying more general interpolants (e.g., Optimal Transport) and adaptive corrections when expert conditions conflict, as well as extending the velocity–score connection beyond Gaussian paths. More broadly, stronger compositional control may amplify misuse risks already present in pretrained text-to-image models; responsible use should therefore follow the safety mechanisms of the underlying generative model. We believe that CompFlow can help make compositional generation a practical and efficient standard inference-time capability in generative modeling.

References

Betker, J., Goh, G., Jing, L., Brooks, T., Wang, J., Li, L., Ouyang, L., Zhuang, J., Lee, J., and et al., Y. G. Improving image generation with better captions. Technical report, OpenAI, 2023. URL https://cdn.openai.com/papers/dall-e-3.pdf.

Chefer, H., Alaluf, Y., Vinker, Y., Wolf, L., and Cohen-Or, D. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models, 2023. URL https://arxiv.org/abs/2301.13826.

Chen, J., Yu, J., Ge, C., Yao, L., Xie, E., Wang, Z., Kwok, J. T., Luo, P., Lu, H., and Li, Z. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=eAKmQPe3m1.

Chen, L., Li, J., Dong, X., Zhang, P., He, C., Wang, J., Zhao, F., and Lin, D. Sharegpt4v: Improving large multimodal models with better captions, 2023. URL https://arxiv.org/abs/2311.12793.

Du, Y. and Mordatch, I. Implicit generation and modeling with energy based models. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 3603–3613, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/378a063b8fdb1db941e34f4bde584c7d-Abstract.html.

Du, Y., Li, S., and Mordatch, I. Compositional visual generation with energy based models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/49856ed476ad01fcff881d57e161d73f-Abstract.html.

Du, Y., Li, S., Tenenbaum, J., and Mordatch, I. Improved contrastive divergence training of energy based models, 2021. URL https://arxiv.org/abs/2012.01316.

Du, Y., Durkan, C., Strudel, R., Tenenbaum, J. B., Dieleman, S., Fergus, R., Sohl-Dickstein, J., Doucet, A., and Grathwohl, W. S. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and MCMC. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8489–8510. PMLR, 2023. URL https://proceedings.mlr.press/v202/du23a.html.

Duane, S., Kennedy, A., Pendleton, B. J., and Roweth, D. Hybrid monte carlo. *Physics Letters B*, 195(2):216–222, 1987. ISSN 0370-2693. doi: https://doi.org/10.1016/0370-2693(87)91197-X. URL https://www.sciencedirect.com/science/article/pii/037026938791197X.

Elfwing, S., Uchibe, E., and Doya, K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, 2017. URL https://arxiv.org/abs/1702.03118.

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., Podell, D., Dockhorn, T., English, Z., and Rombach, R. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=FPnUhsQJ5B.

Feng, W., He, X., Fu, T., Jampani, V., Akula, A. R., Narayana, P., Basu, S., Wang, X. E., and Wang, W. Y. Training-free structured diffusion guidance for compositional text-to-image synthesis. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=PUIqjT4rzq7.

Geffner, T. and Domke, J. MCMC variational inference via uncorrected hamiltonian annealing. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 639–651, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/05f971b5ec196b8c65b75d2ef8267331-Abstract.html.

Hessel, J., Holtzman, A., Forbes, M., Le Bras, R., and Choi, Y. CLIPScore: A reference-free evaluation metric for image captioning. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Proceedings of the 2021*

Conference on Empirical Methods in Natural Language Processing, pp. 7514–7528, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.595. URL https://aclanthology.org/2021.emnlp-main.595.

Hinton, G. E. Products of experts, 1999. URL https://api.semanticscholar.org/CorpusID:15059668.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8): 1771–1800, 2002. ISSN 0899-7667. doi: 10.1162/089976602760128018. URL https://doi.org/10.1162/089976602760128018.

Ho, J. and Salimans, T. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Huang, K., Duan, C., Sun, K., Xie, E., Li, Z., and Liu, X. T2i-compbench++: An enhanced and comprehensive benchmark for compositional text-to-image generation, 2023a. URL https://arxiv.org/abs/2307.06350.

Huang, K., Sun, K., Xie, E., Li, Z., and Liu, X. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023b. URL http://papers.nips.cc/paper_files/paper/2023/hash/f8ad010cdd9143dbb0e9308c093aff24-Abstract-Datasets_and_Benchmarks.html.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991. URL https://api.semanticscholar.org/CorpusID:572361.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1988–1997, 2017. doi: 10.1109/CVPR.2017.215.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Labs, B. F. Flux. https://github.com/black-forest-labs/flux, 2024.

Labs, B. F., Batifol, S., Blattmann, A., Boesel, F., Consul, S., Diagne, C., Dockhorn, T., English, J., English, Z., Esser, P., Kulal, S., Lacey, K., Levi, Y., Li, C., Lorenz, D., Müller, J., Podell, D., Rombach, R., Saini, H., Sauer, A., and Smith, L. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. URL https://arxiv.org/abs/2506.15742.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. J. A tutorial on energy-based learning. In Bakir, G., Hofmann, T., Schölkopf, B., Smola, A., and Taskar, B. (eds.), *Predicting Structured Data*. MIT Press, Cambridge, MA, 2006. URL http://yann.lecun.com. v1.0, August 19, 2006.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=PqvMRDCJT9t.

Liu, N., Li, S., Du, Y., Tenenbaum, J., and Torralba, A. Learning to compose visual relations. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 23166–23178, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/c3008b2c6f5370b744850a98a95b73ad-Abstract.html.

Liu, N., Li, S., Du, Y., Torralba, A., and Tenenbaum, J. B. Compositional visual generation with composable diffusion models. In Avidan, S., Brostow, G., Cissé, M., Farinella, G. M., and Hassner, T. (eds.), *Computer Vision – ECCV 2022*, pp. 423–439, Cham, 2022a. Springer Nature Switzerland. ISBN 978-3-031-19790-1.

Liu, N., Li, S., Du, Y., Torralba, A., and Tenenbaum, J. B. Compositional visual generation with composable diffusion models, 2022b. URL https://arxiv.org/abs/2206.01714.

Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., and Zhang, L. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, 2024.

Mayraz, G. and Hinton, G. E. Recognizing hand-written digits using hierarchical products of experts. In Leen, T. K., Dietterich, T. G., and Tresp, V. (eds.), *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pp. 953–959. MIT Press, 2000. URL https://proceedings.neurips.cc/paper/2000/hash/1f1baa5b8edac74eb4eaa329f14a0361-Abstract.html.

Neal, R. M. *Monte Carlo Implementation*, pp. 55–98. Springer New York, New York, NY, 1996. ISBN 978-1-4612-0745-0. doi: 10.1007/978-1-4612-0745-0_3. URL https://doi.org/10.1007/978-1-4612-0745-0_3.

Neal, R. M. Monte carlo implementation of gaussian process models for bayesian regression and classification, 1997. URL https://arxiv.org/abs/physics/9701026.

Nijkamp, E., Hill, M., Han, T., Zhu, S., and Wu, Y. N. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 5272–5280. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/5973.

Palma, A., Richter, T., Zhang, H., Lubetzki, M., Tong, A., Dittadi, A., and Theis, F. Multi-modal and multi-attribute generation of single cells with cfgen, 2025. URL https://arxiv.org/abs/2407.11734.

Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., and Rombach, R. SDXL: improving latent diffusion models for high-resolution image synthesis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=di52zR8xgf.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021. URL http://proceedings.mlr.press/v139/radford21a.html.

Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., Mintun, E., Pan, J., Alwala, K. V., Carion, N., Wu, C.-Y., Girshick, R., Dollár, P., and Feichtenhofer, C. Sam 2: Segment anything in images and videos, 2024. URL https://arxiv.org/abs/2408.00714.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models, 2021. URL https://arxiv.org/abs/2112.10752.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 2256–2265. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/sohl-dickstein15.html.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 11895–11907, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/3001ef257407d5a371a96dcd947c7d93-Abstract.html.

Villani, C. *Optimal Transport: Old and New*, volume 338 of *Grundlehren der mathematischen Wissenschaften*. Springer, 2009.

Yu, C. and Gao, S. Improving compositional generation with diffusion models using lift scores, 2025. URL https://arxiv.org/abs/2505.13740.

Zheng, Q., Le, M., Shaul, N., Lipman, Y., Grover, A., and Chen, R. T. Q. Guided flows for generative modeling and decision making, 2023. URL https://arxiv.org/abs/2311.13443.

A. Proofs and derivations

The velocity–score identity for Gaussian probability paths was established in full generality by (Zheng et al., 2023) and is restated in Lemma 2.1. We do not reprove that result here. Instead, we specialize it to the Gaussian interpolant used in this work, derive the conditional residual identity, and use these two ingredients to prove Theorem 2.2: that linear composition of conditional velocities implements a tempered Product-of-Experts. All equalities below hold for $t \in (0, 1]$.

A.1. Specialization to the Gaussian interpolant

For the interpolant in Equation (2), the schedule is

$$\alpha_t = t, \qquad \sigma_t = 1 - (1 - \sigma_{\min})\, t, \qquad \dot{\alpha}_t = 1, \qquad \dot{\sigma}_t = -(1 - \sigma_{\min}).$$

Step 1: compute a_t. Directly from the definition,

$$a_t \;=\; \frac{\dot{\alpha}_t}{\alpha_t} \;=\; \frac{1}{t}.$$

Step 2: compute b_t. First note that $t(1 - \sigma_{\min}) = 1 - \sigma_t$ by the definition of σ_t. Then

$$\begin{aligned} b_t &= \frac{\big(\dot{\alpha}_t \sigma_t - \alpha_t \dot{\sigma}_t\big)\sigma_t}{\alpha_t} \\ &= \frac{\big(\sigma_t + t(1 - \sigma_{\min})\big)\sigma_t}{t} \\ &= \frac{\big(\sigma_t + 1 - \sigma_t\big)\sigma_t}{t} \;=\; \frac{\sigma_t}{t}. \end{aligned}$$

Step 3: assemble the velocity–score identity. Substituting $a_t = 1/t$ and $b_t = \sigma_t/t$ into $v_t^c(x) = a_t x + b_t\, s_t^c(x)$ yields

$$v_t^c(x) \;=\; \underbrace{\tfrac{1}{t}\, x}_{\text{drift}} \;+\; \tfrac{\sigma_t}{t}\, \underbrace{\nabla_x \log p_t(x \mid c)}_{\text{conditional score}}. \tag{15}$$

Sampling is initialized from $x_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, so the $1/t$ singularity at $t = 0$ is never evaluated.

A.2. Conditional velocity residuals are likelihood gradients

We compute the residual between the conditional and unconditional velocities. Recall

$$s_t^c(x) = \nabla_x \log p_t(x \mid c), \qquad s_t^0(x) = \nabla_x \log p_t(x).$$

Step 1: drift cancels in the residual. Both v_t^c and v_t^0 share the drift $\frac{1}{t}x$, which vanishes upon subtraction:

$$v_t^c(x) - v_t^0(x) \;=\; \tfrac{\sigma_t}{t}\big(\nabla_x \log p_t(x \mid c) - \nabla_x \log p_t(x)\big) \;=\; \tfrac{\sigma_t}{t}\, \nabla_x \log \frac{p_t(x \mid c)}{p_t(x)}.$$

Step 2: apply Bayes' rule. Substituting $p_t(x \mid c) = p_t(c \mid x)\, p_t(x) \,/\, p_t(c)$,

$$\nabla_x \log \frac{p_t(x \mid c)}{p_t(x)} \;=\; \nabla_x \log \frac{p_t(c \mid x)}{p_t(c)} \;=\; \nabla_x \log p_t(c \mid x),$$

where the last equality uses $\nabla_x \log p_t(c) = 0$.

Conclusion.

$$\boxed{v_t^c(x) - v_t^0(x) \;=\; \tfrac{\sigma_t}{t}\, \nabla_x \log p_t(c \mid x)} \tag{16}$$

The conditional velocity residual is exactly the (rescaled) gradient of the noised log-likelihood of the condition. □

A.3. Proof of Theorem 2.2: velocity addition gives a Product of Experts

Let $c_1, \ldots, c_m$ be conditions to compose, with weights $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$. The composed velocity field is defined as the unconditional velocity plus a weighted sum of conditional residuals,

$$v_t^{\text{comp}}(x) \;:=\; v_t^0(x) \;+\; \sum_{i=1}^{m} \lambda_i \big[v_t^{c_i}(x) - v_t^0(x)\big], \tag{17}$$

and we define the tempered Product-of-Experts density

$$\tilde{p}_t(x) \;\propto\; p_t(x) \prod_{i=1}^{m} p_t(c_i \mid x)^{\lambda_i}. \tag{18}$$

Step 1: rewrite each residual via the score identity. Substituting Equation (16),

$$v_t^{\text{comp}}(x) \;=\; v_t^0(x) \;+\; \frac{\sigma_t}{t} \sum_{i=1}^{m} \lambda_i \, \nabla_x \log p_t(c_i \mid x).$$

Step 2: expand the unconditional velocity. Using Equation (15) with no conditioning, $v_t^0(x) = \frac{1}{t}x + \frac{\sigma_t}{t}\nabla_x \log p_t(x)$, so

$$v_t^{\text{comp}}(x) \;=\; \tfrac{1}{t}\, x \;+\; \tfrac{\sigma_t}{t}\left[\nabla_x \log p_t(x) \;+\; \sum_{i=1}^{m} \lambda_i \nabla_x \log p_t(c_i \mid x)\right].$$

Step 3: collect into a single log-gradient. The bracketed expression is the gradient of a sum of logs, hence the log of a product:

$$\nabla_x \log p_t(x) + \sum_{i=1}^{m} \lambda_i \nabla_x \log p_t(c_i \mid x) \;=\; \nabla_x \log\left[p_t(x) \prod_{i=1}^{m} p_t(c_i \mid x)^{\lambda_i}\right] \;=\; \nabla_x \log \tilde{p}_t(x),$$

where the last equality uses Equation (18) and the fact that the proportionality constant in $\tilde{p}_t$ is independent of x.

Conclusion.

$$\boxed{v_t^{\text{comp}}(x) \;=\; \tfrac{1}{t}\, x \;+\; \tfrac{\sigma_t}{t}\, \nabla_x \log \tilde{p}_t(x)} \tag{19}$$

The linearly composed velocity field is exactly the velocity associated with the tempered Product-of-Experts density $\tilde{p}_t$. □

A.4. Exactness and continuity equation under conditional independence

We now show that under a conditional-independence assumption, v_t^{comp} generates the exact joint conditional probability path.

Setup. Assume that for all $t \in (0, 1]$ the noised conditions are conditionally independent given x,

$$p_t(c_1, \ldots, c_m \mid x) \;=\; \prod_{i=1}^{m} p_t(c_i \mid x), \tag{20}$$

and take $\lambda_i = 1$ for all i.

Step 1: the PoE density is the joint conditional. Under Equation (20),

$$\begin{aligned}\tilde{p}_t(x) \propto\; p_t(x) \prod_{i=1}^{m} p_t(c_i \mid x) \;&=\; p_t(x)\, p_t(c_1, \ldots, c_m \mid x) \\ = p_t(x, c_1, \ldots, c_m) \;&\propto\; p_t(x \mid c_1, \ldots, c_m),\end{aligned}$$

where the last proportionality holds because $p_t(c_1, \ldots, c_m)$ does not depend on x. Since $\tilde{p}_t$ and $p_t(\cdot \mid c_1, \ldots, c_m)$ differ only by an x-independent constant, their scores coincide:

$$\nabla_x \log \tilde{p}_t(x) \;=\; \nabla_x \log p_t(x \mid c_1, \ldots, c_m). \tag{21}$$

Step 2: the conditional path is a Gaussian probability path. The conditional marginal $p_t(\cdot \mid c_1, \ldots, c_m)$ is itself a Gaussian probability path on the same schedule (α_t, σ_t), obtained by integrating the same forward kernel $p_t(x \mid x_1) = \mathcal{N}(x; \alpha_t x_1, \sigma_t^2 I)$ against the conditional base $p_{\text{data}}(x_1 \mid c_1, \ldots, c_m)$. The velocity–score identity (Lemma 2.1, specialized as in Equation (15)) therefore applies, giving the velocity generating this path as

$$v_t^{c_1,\ldots,c_m}(x) \;=\; \tfrac{1}{t}\, x \;+\; \tfrac{\sigma_t}{t}\, \nabla_x \log p_t(x \mid c_1, \ldots, c_m). \tag{22}$$

Step 3: the composed and joint conditional velocities coincide. Combining Equation (19), Equation (21), and Equation (22),

$$v_t^{\text{comp}}(x) \;=\; \tfrac{1}{t}\, x \;+\; \tfrac{\sigma_t}{t}\, \nabla_x \log p_t(x \mid c_1, \ldots, c_m) \;=\; v_t^{c_1,\ldots,c_m}(x).$$

Continuity equation on the conditional path. (Proof of Corollary 2.3) Since $v_t^{c_1,\ldots,c_m}$ is by construction the velocity generating $p_t(\cdot \mid c_1, \ldots, c_m)$, and v_t^{comp} equals it, the pair $(p_t(\cdot \mid c_1, \ldots, c_m),\ v_t^{\text{comp}})$ satisfies the continuity equation:

$$\boxed{\partial_t\, p_t(x \mid c_1, \ldots, c_m) \;+\; \nabla \cdot \big(p_t(x \mid c_1, \ldots, c_m)\, v_t^{\text{comp}}(x)\big) \;=\; 0.} \tag{23}$$

Integrating $\dot{x}_t = v_t^{\text{comp}}(x_t)$ from $x_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ therefore produces samples from $p_1(\cdot \mid c_1, \ldots, c_m)$. □

Remark on the conditional-independence assumption. Equation (20) is an assumption on the *noised* conditionals at every $t \in (0, 1]$, not on the clean data conditionals at $t = 1$. Even when the clean conditions are conditionally independent given x_1, the noised conditionals generally are not, because marginalization over x_1 couples them through the posterior $p(x_1 \mid x_t)$. The assumption is the intuition behind what allows our deterministic ODE to sample the exact joint conditional without the MCMC corrections used by prior score-based composition methods (Du et al., 2023); we treat it as a modeling assumption and validate it empirically in Section 3.

B. Additional details on CLEVR experiments

UNet backbone. The velocity network v_θ is a UNet operating at 128×128 RGB resolution with base channel width $C = 128$ and channel multipliers $(1, 2, 2, 3)$, giving feature widths $(128, 256, 256, 384)$ across the four spatial levels. Each level uses two residual blocks with GroupNorm (32 groups) and SiLU activations (Elfwing et al., 2017); downsampling halves the spatial resolution with a strided 3×3 convolution. Multi-head self-attention with four heads is applied only at the lowest resolution (16×16, downsampling factor 8), which we found sufficient for CLEVR and keeps per-step cost tractable under the $N(m{+}1)$ scaling of compositional sampling. Skip connections concatenate matching-resolution features across the encoder and decoder. Timestep $t \in (0, 1]$ is embedded by sinusoidal features of dimension C followed by a two-layer MLP to dimension $4C = 512$ and injected into every residual block via scale–shift modulation. The output head applies a zero-init 3×3 convolution, so the learned velocity field starts at zero and is updated only by the training signal. The backbone contains 54.7M parameters.

Condition encoder. Each object is described by an attribute vector $c \in \mathbb{R}^{d_c}$ stacking normalized spatial coordinates and the integer-encoded shape and color classes ($d_c = 4$ in our runs). A two-layer MLP with SiLU activation projects c to a $4C$-dimensional embedding. This embedding is added to the timestep embedding before being fed to each residual block, sharing the same scale–shift modulation path; no cross-attention is used. A single learned null token $e_\varnothing \in \mathbb{R}^{4C}$, initialized from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$, replaces the embedding with probability $p_{\text{drop}} = 0.15$ during training and provides the unconditional velocity v_t^0 required by Algorithm 1 at inference.

Training dataset. We train on CLEVR (Johnson et al., 2017) rendered scenes containing between one and five objects. We generated the images for training according to the original CLEVR reported on the CLEVR webpage and GitHub repository. The training corpus aggregates all available relation shards and totals $185{,}803$ scene images, each paired with the scene JSON that enumerates every object together with its world-frame coordinates, shape, color, and projected pixel coordinates. At training time we draw a single random object per scene as the conditioning target, yielding single-object supervision exclusively. Images are resized to 128×128 and normalized to $[-1, 1]$. We split the dataset 90:5:5 (train/val/test) with a fixed seed.

Table 5. Compute configuration used for training and evaluation.

Component	Specification
Operating system	Ubuntu 24.04.3 LTS
Kernel	Linux 6.8.0-83-generic
CPU	2× AMD EPYC 9274F 24-Core Processor
CPU cores / threads	48 physical cores / 96 hardware threads
System memory	1.5 TiB RAM
GPU	4× NVIDIA L40S
GPU memory	48 GB per GPU, 192 GB total
Training precision	TF32 matmul precision

Optimizer and schedule. We optimize with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$), weight decay 10^{-5}, and base learning rate 10^{-4}. The schedule is a linear warmup over the first 10,000 steps followed by a linear decay to 10^{-8} over training. Gradients are clipped at a global norm of 1.0. The per-device batch size is 64 with gradient accumulation to an effective batch of 256. Training uses TF32 matmul precision. A dropout of 0.15 is applied inside residual blocks. Early stopping monitors the validation CFM loss with patience 30 epochs; the selected checkpoint reaches validation loss 0.0079 after 103 epochs. Each training epoch required $\approx$ 15 minutes of training on our machines. This corresponds to approximately 25.75 wall-clock hours, or 103 GPU-hours, for the selected training run. Our compute configuration is detailed in Table 5.

Table 4. Wall-clock sampling time vs. number of composed conditions m on CLEVR (128×128, Euler solver, 20 steps, single NVIDIA GPU). Mean ± std over 5 runs after 1 warmup pass; m=1 uses single-condition generation, $m\geq2$ uses velocity composition. Cost scales roughly linearly in m, with one model forward per condition plus one null-token pass per step.

m	Time (s)	Min (s)	Max (s)	ms/step
1	0.5402 ± 0.0276	0.4860	0.5639	24.01
2	0.9084 ± 0.0026	0.9040	0.9119	45.42
3	1.1540 ± 0.0114	1.1325	1.1636	57.70
4	1.4081 ± 0.0143	1.3798	1.4173	70.40
5	1.6568 ± 0.0097	1.6440	1.6724	82.84

ODE solver and sampling. At inference we integrate the composed velocity field from Equation (11) with a fixed-step Euler solver and $N = 20$ steps, which empirically matches adaptive higher-order solvers (Heun, Dopri5, Tsit5 as implemented by `torchode`) on sample quality at a fraction of the per-step cost. The initial condition is $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and composition uses per-condition weights $\lambda_i = w$ with w set by the guidance-weight ablation below. Under this configuration the NFE per sample is $N(m + 1)$, i.e. 40, 60, 80, 100, and 120 for $m = 1, \ldots, 5$ respectively. We additionaly report sampling times in Table 4

B.1. Guidance-Weight Ablation For CLEVR experiments

Protocol.

SAM2 cross-check. We sweep a single guidance weight w applied uniformly to every conditioning slot ($\lambda_i = w$ for all i), jointly over $m \in \{1, 2, 3, 4, 5\}$ and $w \in \{1, 2, 3, 4, 5, 10\}$. For every (m, w) we generate 200 scenes whose per-object conditions are sampled from real CLEVR ground truth (matching the m-object budget) and evaluate them under the GroundingDINO protocol of Section 3.2: per-object accuracy at threshold τ, mean matched-detection pixel distance, and DINOv2 top-1 similarity to a real-CLEVR reference pool. All entries are produced by the same checkpoint, differing only in w. We provide additional qualitative samples in Figure 11

Findings. Table 6 reports the full sweep. Accuracy peaks at low w for $m = 1$ (near-perfect already at $w = 1$), and shifts to $w \in \{2, 3, 4\}$ as m grows, in line with the standard tradeoff in classifier-free guidance between condition adherence and sample realism. For

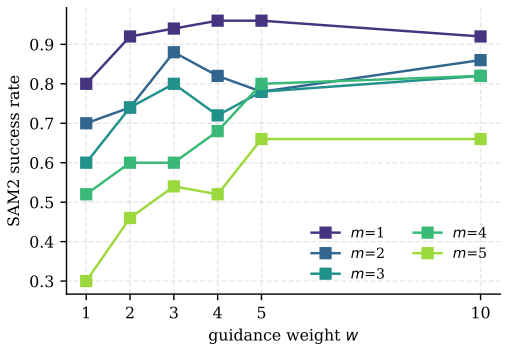

Figure 8. SAM2 position-only success rate as a function of guidance weight w, one curve per m. Same sweep grid as Table 6.

Table 6. Guidance-weight sweep on CLEVR (GroundingDINO protocol, 200 scenes per cell). For each (m, w) we report per-object accuracy at threshold τ (higher is better) and mean matched-detection pixel distance (lower is better).

m	metric	w=1	w=2	w=3	w=4	w=5	w=10
1	acc @ τ	**0.991**	0.963	0.943	0.942	0.942	0.923
	dist (px)	6.71	**4.80**	5.51	5.19	4.94	5.87
2	acc @ τ	0.921	**0.975**	0.944	0.947	0.961	0.931
	dist (px)	7.39	**5.32**	6.09	6.05	6.15	6.66
3	acc @ τ	0.887	0.945	0.907	**0.947**	0.927	0.945
	dist (px)	7.89	6.86	6.77	**6.57**	6.78	6.67
4	acc @ τ	0.875	0.905	0.898	**0.945**	0.945	0.870
	dist (px)	9.00	8.11	7.62	6.81	**6.18**	8.57
5	acc @ τ	0.804	0.860	**0.865**	0.822	0.832	0.737
	dist (px)	11.66	9.49	**9.07**	9.42	10.59	14.21

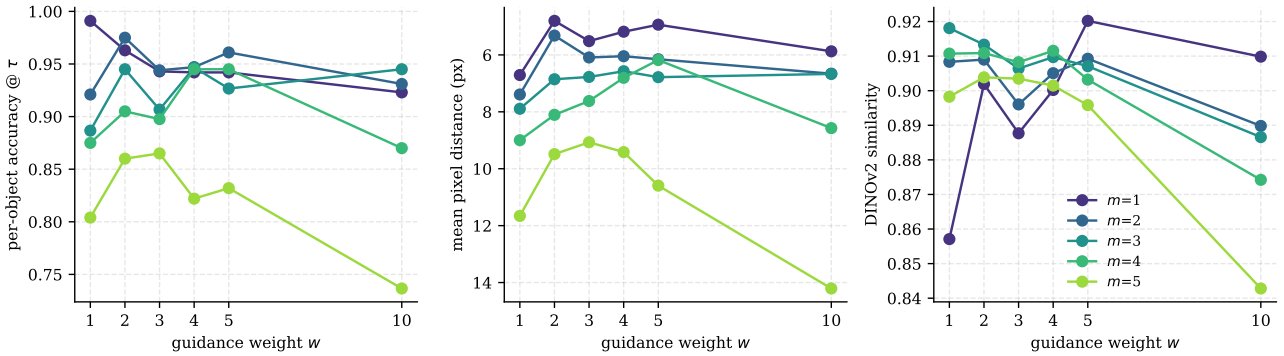

Figure 9. Guidance-weight ablation under the GroundingDINO protocol. One curve per m. *Left:* per-object accuracy at threshold τ. *Middle:* mean matched-detection pixel distance (the y-axis is inverted so that upward is better). *Right:* DINOv2 top-1 similarity to the real-CLEVR reference pool. The best operating point for each m shifts from low w at m=1 (saturation) to $w \in \{2, 3, 4\}$ as compositional load grows; very large w hurts consistently by $m \geq 4$. Data from Table 6;

$m \geq 2$ the degradation at $w = 10$ is consistently sharp: the model starts to sacrifice realism for excessive condition sharpness. The choice $w = 4$ used in the main-text results sits within one percentage point of the per-m best for every $m \leq 4$ and remains competitive at $m = 5$. Figure 9 plots the same data as curves in w; Figure 10 summaries the best-per-m operating point.

Running the same sweep under the SAM2 position-only verifier yields the same qualitative shape: the optimal w rises from $w = 4$ at $m = 1$ (96% success) to $w \in \{3, 10\}$ for larger m, and the degradation at $w = 10$ flattens once identity is not being checked. Figure 8 shows the curves. The two protocols therefore agree on the coarse pattern but disagree on the best w for larger m: GroundingDINO penalizes the realism loss that SAM2 ignores, so it rewards more moderate guidance.

B.2. Discussion of failure modes in CLEVR.

Figure 12 characterizes the regimes in which CompFlow's composition mechanism breaks down, complementing the in-distribution results of Section 3.2. Interestingly, three of the five regimes are informative about the method itself.

Co-located (row 1) is the most surprising result: when two objects are requested at the same (x, y), the model resolves the conflict geometrically by stacking or producing physically plausible configurations despite the absence of stacking in the single-object training distribution. Moreover, it is possible to observe collapsed modalities towards style merging. We attribute this to the composed velocity field locally satisfying both position constraints in the only way the rendering geometry permits and reveals interesting Out-of-Distribution (OOD) results.

Conflicting attributes (row 2) shows the predicted behavior of weighted velocity addition when two experts disagree on a single attribute at a single location: the composed score becomes the average of two incompatible velocity fields, and the resulting samples exhibit color-bleed artifacts rather than a clean choice between ex-

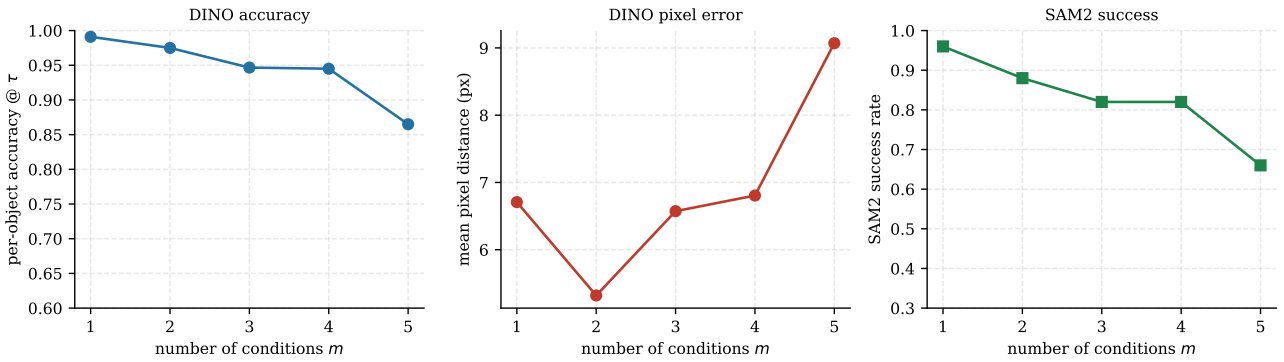

Figure 10. Best-per-m operating points: accuracy and pixel error under GroundingDINO, and success rate under SAM2, as a function of m. The w selected for each point is the per-m argmax of the corresponding metric from Table 6 and the SAM2 sweep.

perts. In other words, asking the model to generate at the same time "a blue sphere" and "a red sphere" in the same position results in dynamics induced style mixing, producing a purple object. This is consistent with the Product-of-Experts interpretation of Theorem 2.2. *OOD position* (row 3) probes coordinates far outside the training range, where the model falls back to in-distribution placements, often hallucinating extra objects.

Dense cluster (row 4) packs four objects into a region smaller than the typical inter-object spacing in CLEVR; here attribute identity degrades into an overlapping mass, though object count is approximately preserved. Both failures would likely be mitigated by training on a wider positional range and denser scenes, and neither contradicts the PoE analysis. Overall, the behavior of CompFlow under stress is consistent with its theoretical interpretation: it composes constraints faithfully where the backbone supports them, and degrades predictably where it does not.

Duplicated condition (row 5) provides a useful sanity check: repeating the same condition three times does *not* produce three objects but a single one, since the composed density $\tilde{p}_t(x) \propto p_t(x) \prod_i p_t(c|x)^{\lambda_i}$ increases around the same mode rather than instantiating multiple objects. This confirms that CompFlow composes *constraints on a sample*, not *independent objects in a scene*, and clarifies the semantics of multi-object generation: scene multiplicity must be encoded through distinct conditions, not through repetition.

Table 7. Failure-mode scene specifications for CLEVR.

Scenario	Objects (shape, color, position)
Co-located	red cube @ $(0.0, 0.0)$ blue sphere @ $(0.0, 0.0)$
Conflicting attrs	red cube @ $(0.0, 0.0)$ green cube @ $(0.0, 0.0)$
OOD position	yellow sphere @ $(-5.0, -3.0)$
Dense cluster	red cube @ $(-0.3, -0.3)$ green sphere @ $(0.3, -0.3)$ blue cylinder @ $(0.3, 0.3)$ yellow cube @ $(-0.3, 0.3)$
Duplicated cond	red sphere @ $(0.0, 0.0)$ red sphere @ $(0.0, 0.0)$ red sphere @ $(0.0, 0.0)$

C. Details on Text2Image generation task

C.1. Experimental setup

Backbone and pipeline. All text-to-image experiments use the publicly released `black-forest-labs/FLUX.1[dev]`. We rely on the `diffusers FluxPipeline` ($\geq$0.30) from h and reuse all native components (text encoders, VAE, and the `FlowMatchEulerDiscreteScheduler`) without modification: there is no finetuning, no LoRA adaptation, and no change to either the transformer architecture or the scheduler. CompFlow is implemented purely as a sampling-time wrapper around the stock pipeline. Unless stated otherwise, all FLUX experiments use the configuration summarized in Table 8.

Velocity composition. At every ODE step the transformer is queried once per conditional sub-prompt c_i plus once on the unconditional prompt $c_0 = \varnothing$. The predicted velocities are combined according to Equation (11) which exactly recovers

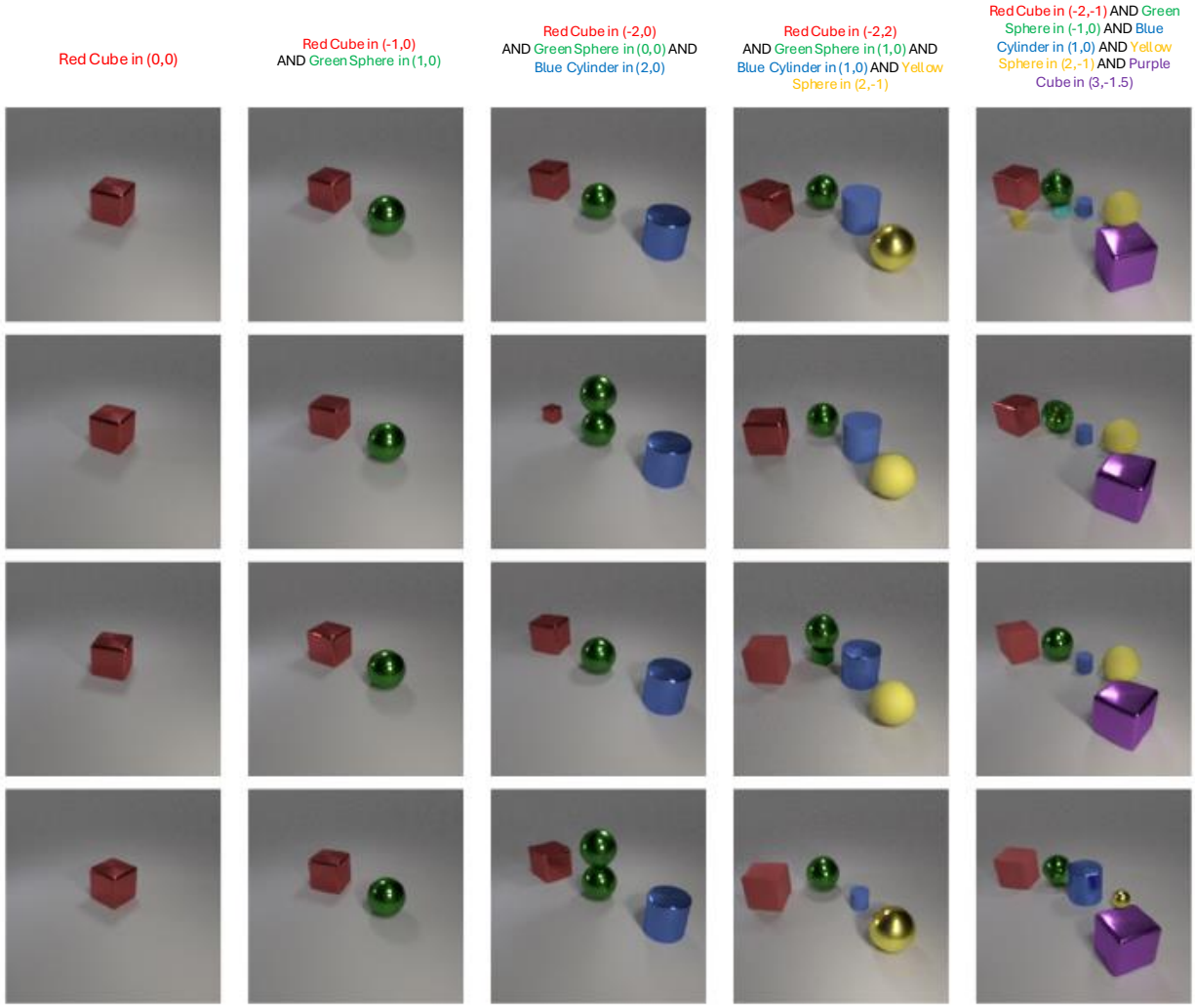

Figure 11. **Additional CLEVR samples** generated with CompFlow under increasing compositional load. Each column corresponds to a different set of composed object-level conditions, from a single red cube to five simultaneous shape–color–position constraints. Rows show independent samples generated from different random seeds for the same column conditions. Across seeds, the model consistently preserves the requested object attributes and approximate spatial layout, despite being trained only with single-object conditioning.

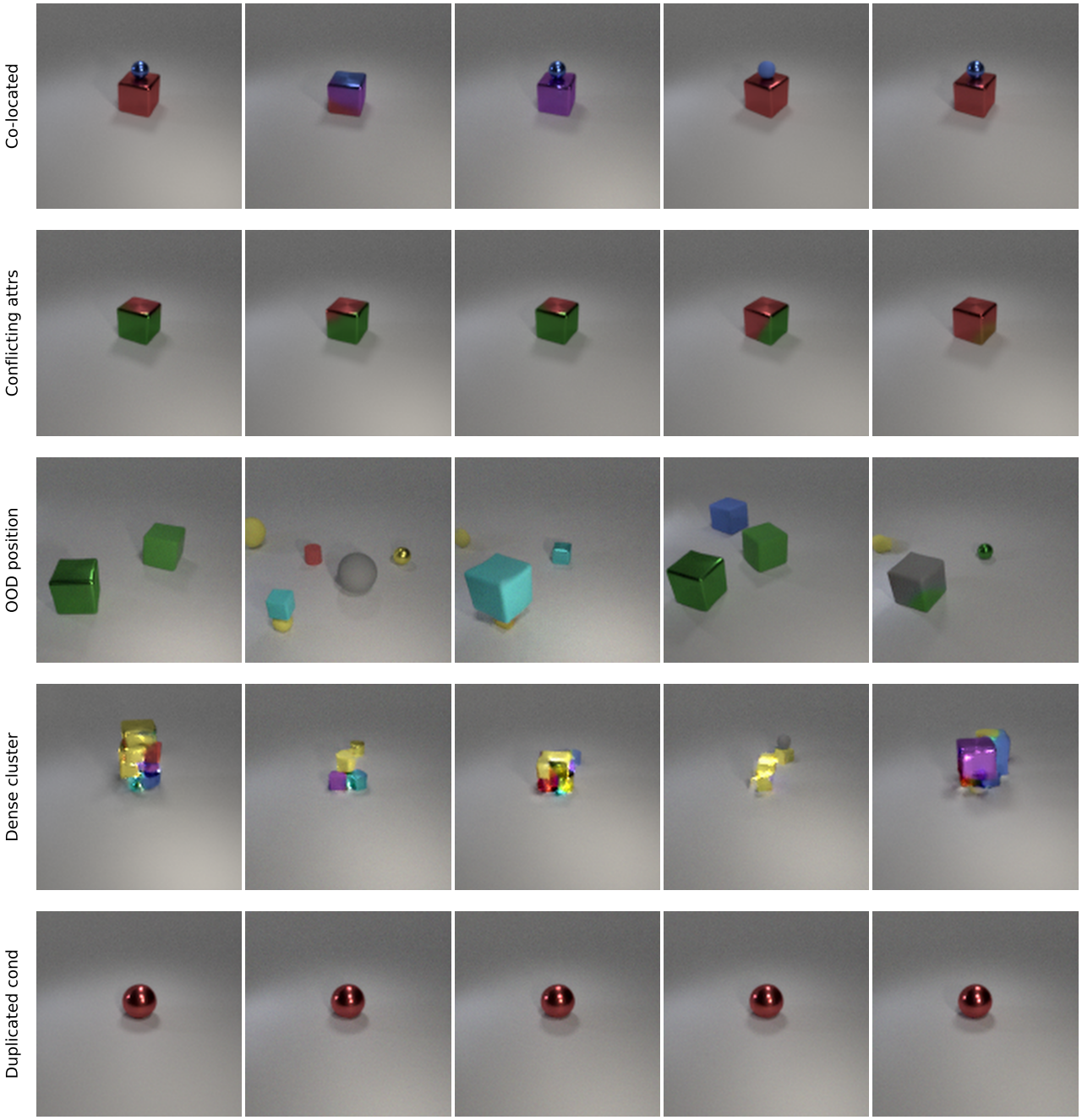

Figure 12. **CompFlow failure modes** Five rows probe regimes the training distribution does not cover. Each row shows five independent samples at w=4, N=20, requested contidions are detailed in Table 7

Table 8. Default hyperparameters for CompFlow on `FLUX.1[dev]`. The "Source" column distinguishes parameters inherited from the stock `FLUX.1[dev]` pipeline from those introduced by CompFlow.

Component	Value	Source
Backbone	`black-forest-labs/FLUX.1[dev]`	FLUX
Scheduler	`FlowMatchEulerDiscreteScheduler`	FLUX
Resolution	1024×1024 (mult. of 16)	FLUX
ODE steps N	28	ours
FLUX-distilled guidance	1	ours
Numerical precision	`bfloat16`	ours
Unconditional prompt	empty string `""`	ours
$\lambda_\varnothing$ (uncond. weight)	1.0	ours
Per-condition λ_i	uniform, sweep $\{1, 2, 4, 8\}$	ours

Algorithm 1 of the main paper for $\lambda_\varnothing = 1$. The composed velocity v_{comp} is then passed to the native scheduler step. We keep the FLUX guidance scalar fixed across all $m + 1$ forward passes within a step (see below), so that velocity composition acts strictly on top of the backbone's native prediction rather than re-entangling with classifier-free guidance.

C.2. Prompt decomposition and why it is non-trivial

CompFlow generates images by composing a set of conditional velocity fields from the `FLUX.1[dev]` backbone, one per condition c_i, where each condition is a natural-language sub-prompt describing a scene constraint. For this reason, evaluating CompFlow on the non-spatial, test-set category of T2I-CompBench requires decomposing each benchmark prompt, i.e. a single sentence describing a complex scene, into a list of semantically *disjoint* sub-prompts that will as independent conditions. This decomposition step is necessary and non-trivial. In fact, the benchmark prompts were designed to be natural and descriptive rather than structurally separable, and they exhibit patterns that poorly fits with naive splitting. The non-spatial test-set split of T2I-CompBench contains 300 prompts of the form:

A person is wearing a hat and sunglasses while fishing.
A woman is holding a camera and taking photos of a beautiful landscape.
A group of activists are marching in protest, chanting slogans and waving signs.

We concluded that a simple string-level decomposition is insufficient for these prompts. In particular, splitting on conjunctions such as *"and"* may produces fragments that are either incomplete, redundant, or semantically dependent on missing context. For example, in *"a person is wearing a hat and sunglasses while fishing"*, the phrase *"sunglasses while fishing"* is not a standalone condition unless the subject *"person"* is recovered from the first conjunct. Similarly, in *"holding a camera and taking photos of a beautiful landscape"*, the two clauses describe a coupled action involving the same agent and object, rather than two independent visual entities. Other prompts contain coordinated verb phrases, nested noun phrases, or conjunctions inside a single concept, such as *"chanting slogans and waving signs"*. A regex-based splitter would therefore either *over-segment* a single coherent event or under-specify the resulting conditions, producing condition sub-prompts that do not preserve the meaning of the original benchmark prompt. This makes prompt decomposition a semantic parsing problem rather than a purely syntactic preprocessing step.

Decomposition strategy. The design principle behind our chosen strategy is that **sub-prompts must not share actions or visual objects**, guaranteeing fairness (i.e.there is no a single prompt describing the whole scene that has to be generated). Each sub-prompt is a grammatically complete and standalone sentence describing a single action or attribute of the subject, containing the minimum information needed to ground that fragment in a Text-to-image generative setting. The decomposition preserves the subject across sub-prompts while assigning each sub-prompt its own *non-overlapping* predicate: the visual content described by any one sub-prompt does not appear in any other. Table 9 illustrates this strategy on randomly sampled benchmark prompts.

LLM-based automatic decomposition. Given the diversity of syntactic patterns, we automated the decomposition using an LLM. Each of the 300 non-spatial prompts was sent as a separate user message; the model was instructed to return only a JSON array with no additional text. The system prompt is reported below.

Table 9. Prompt decomposition examples from the non-spatial T2I-CompBench split. Each compound benchmark prompt is split into sub-prompts with non-overlapping predicates, used as independent conditions in CompFlow.

Benchmark prompt	**Decomposition (ours)**
A person is wearing a hat and sunglasses while fishing.	`["A person is wearing a hat", "A person is wearing sunglasses", "A person is fishing with a rod"]`
A man is sitting on a park bench feeding pigeons.	`["A man is sitting on a park bench", "A man is feeding pigeons"]`
A dog is wagging its tail and playing fetch with its owner.	`["A dog is wagging its tail", "A dog is playing fetch with its owner"]`
A person is standing on a ladder and fixing a leaky roof.	`["A person is standing on a ladder", "A person is fixing a leaky roof"]`
A man is holding a basketball and practicing his jump shot.	`["A man is holding a basketball", "A man is practicing his jump shot"]`

System prompt for LLM-based decomposition

```
Your task is to decompose a natural-language image prompt into a list of
non-overlapping sub-prompts, one per independent visual concept, suitable as separate
conditioning signals for a compositional generative model.

Rules:

 1. Each sub-prompt must be a complete, self-contained sentence that could stand alone
    as an image-generation prompt.

 2. Preserve the subject across sub-prompts.  Assign each sub-prompt a single,
    distinct action or attribute -- no action or object should appear in more than
    one sub-prompt.

 3. If the prompt describes a single atomic scene with one action, return a one-element
    list containing the original prompt unchanged.

 4. Return only a JSON array of strings.  No explanation, no preamble, no markdown
    fences.

Examples:
Input:  "A person is wearing a hat and sunglasses while fishing."
Output:  ["A person is wearing a hat", "A person is wearing sunglasses",          "A
person is fishing with a rod"]
Input:  "A man is sitting on a park bench feeding pigeons."
Output:  ["A man is sitting on a park bench", "A man is feeding pigeons"]
```

The LLM output was then post-processed by parsing and validating the JSON array and fed to the compositional sampler.

Prompts used for generating images in Figure 1

- `["a cute sleeping dragon" + "a pillow-book" + "a mystical forest" + "blue-ish dreamlike atmosphere"]`
- `["A red fox in a snowy forest with trees", "film photo style, soft lighting, detailed, intricate, sharp focus", "amber atmosphere, cinematic, volumetric lighting, 85mm lens"]`
- `["a lone traveler wearing a wool cloak" + "walking through the Scottish Highlands" + "epic cinematic landscape photography" + "muted colors"]`
- `["a robot" + "standing in a mossy forest" + "fireflies floating around" + "storybook illustration style"]`

T2I-CompBench evaluation. For the non-spatial test-set split, each benchmark prompt is first decomposed into sub-prompts as described in Appendix C.2; the resulting list is then fed to the sampler with uniform weights $\lambda_i = \lambda$ and $\lambda_\varnothing = 1$. All metrics are computed against the original, un-decomposed benchmark prompt, so that CompFlow is judged on its ability to reconstruct the full benchmark scene from disjoint sub-prompts rather than against the easier per-fragment task.

For each of the 300 prompts in the T2I-CompBench non-spatial split, we generate five images using independent random seeds, yielding $5 \times 300 = 1500$ images per hyperparameter configuration. We sweep the guidance weight over

$$\lambda \in \{0.5, 1, 2, 3\}$$

and the number of sampling steps over

$$N \in \{20, \ldots, 40\}.$$

We report the results obtained by the best aggregate configuration in this sweep. The best-performing setting was $\lambda = 1$; unless stated otherwise with N=28 sampling steps.

Computational cost. Figure 13 illustrates the computational cost above with a direct measurement of the composition sampler's wall-clock time as a function of the number of conditions $m \in \{1, \ldots, 6\}$.

The benchmark was run on a single NVIDIA L40S (48 GB, `bfloat16`) at 512×512 resolution with $N = 20$ ODE steps. The measured curve is essentially linear in m, in close agreement with the predicted $N \cdot (m + 1)$ forward-pass count: the slope corresponds to the cost of one additional conditional forward pass per ODE step, while the intercept absorbs the unconditional pass and the constant per-step scheduler overhead. Trial-to-trial variance stays within a few percent and does not grow with m, indicating that at the resolutions we use no allocator pressure or memory thrashing kicks in, and that deployment-time inference cost is fully predictable from m alone. All the experiments were run on a single GPU under the configuration detailed in Table 5

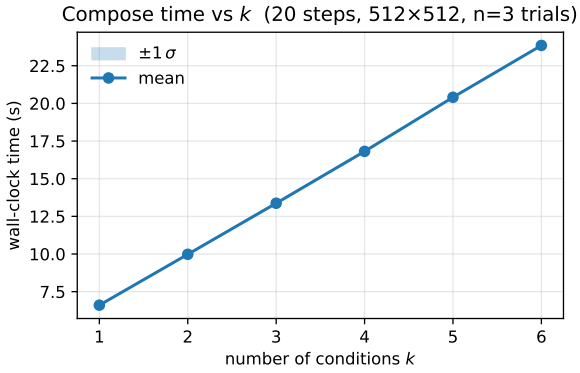

Figure 13. Wall-clock time of velocity composition as a function of the number of conditions m, measured on a single NVIDIA L40S.