Why We Build Toys

Toys are not mini versions of the real world. They are illustrative tools: simplified distributions that help us build intuition about what diffusion models are doing as *distribution learners*. Because diffusion models operate by transforming noise into samples through a sequence of stochastic (or deterministic) steps, many of their behaviors are most naturally understood at the level of distributions—where probability mass moves, which structures are preserved, and what kinds of patterns emerge as the model denoises.

This is why toy examples appear so often in diffusion research. They give us a small, visual, controllable environment where we can observe the dynamics of learning and sampling directly, without the clutter of high-dimensional images or complicated evaluation pipelines. In a well-chosen toy, phenomena that are hard to notice in real datasets become obvious on a plot, making toys a convenient starting point for explaining ideas, stress-testing intuitions, and communicating results.

When Simple Toys Mislead

First, let's consider a simple toy: a two-mixture of Gaussians—Class A centered at (-2, 0) and Class B at (2, 0). We train a toy diffusion model on both classes.

In the visualization below, you can see:

- **Light blue dots**: Class A ground truth samples
- **Dark blue dots**: Class B ground truth samples
- **Orange dots**: Generated samples for Class B using classifier-free guidance

Notice what happens as you move the slider. When w=1.0 (meaning no guidance boost—just the conditional prediction), the orange samples should closely match the dark blue Class B ground truth. This is what we'd expect: the model was trained well, and unguided conditional generation captures the learned distribution.

Yet as guidance increases (w > 1.0), something concerning happens: the orange samples drift away from the dark blue target. The stronger the guidance, the further the generated distribution deviates from ground truth. This is the key phenomenon this toy demonstrates.

Why does this happen?

To understand this phenomenon, let's start with the ideal case. In flow matching, the *optimal* velocity field at position $\mathbf{x}_t$ and time t is defined as:

$$\dot{\mathbf{v}}^*(\mathbf{x}_t, t, \mathbf{c}) = \mathbb{E}_{\mathbf{x}_0 \sim p(\cdot|\mathbf{c}), \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})}[\boldsymbol{\epsilon} - \mathbf{x}_0 \mid \mathbf{x}_t, t]$$

Expanding this conditional expectation:

$$\dot{\mathbf{v}}^*(\mathbf{x}_t, t, \mathbf{c}) = \frac{\sum_{i=1}^{N} (\mathbf{x}_t - \mathbf{x}_0^i) \mathcal{N}(\mathbf{x}_t; (1-t)\mathbf{x}_0^i, t^2\mathbf{I})}{t \sum_{j=1}^{N} \mathcal{N}(\mathbf{x}_t; (1-t)\mathbf{x}_0^j, t^2\mathbf{I})}$$

With this optimal velocity, the sampling process follows the true conditional distribution faithfully, and the model can perfectly reconstruct the data. In our 2D toy setting, this is easily achieved: the model is simple, the distribution is low-dimensional and well-structured, and training converges to near-optimal predictions. When $w = 1.0$ (using no guidance), we sample from the learned conditional distribution, which matches the ground truth well.

However, in high-dimensional settings like natural images or videos, the story is different. Real data lies on complex manifolds, and learning the true optimal velocity becomes fundamentally difficult. The model's predictions inevitably deviate from the optimal field, and these errors accumulate across the many sampling steps. Each denoising step amplifies inaccuracies from previous steps, and the multi-step trajectory diverges from the true path.

Now, when we apply classifier-free guidance with weight w:

```
noise_pred = noise_uncond + w * (noise_cond - noise_uncond)
```

We are amplifying the difference between the conditional and unconditional predictions. At $w = 1.0$, we use the conditional prediction as-is. But as w increases, we exaggerate the class signal—even when the model's conditional prediction already accurately captures the class distinction. In the toy setting where the model is well-trained and this distinction is already clear, the amplification simply *distorts* the distribution, pushing samples away from the learned data manifold. The result: a distribution that is artificially pushed away from the ground truth.

The lesson: Guidance is not always helpful. In this scenario—where the model is well-trained and the classes are already well-separated—applying guidance *harms* generation. This illustrates why toy examples can mislead: they encode specific scenarios. This toy shows the regime where guidance hurts, but that's not universal. The choice of what scenario to visualize shapes the conclusion we draw.

AutoGuidance's Toy: Richer Structure, Hidden Assumptions

AutoGuidance (AG) takes toy experiments a step further by constructing a richer example with more complex in-class distributions. Instead of simple unimodal Gaussians, each class contains multiple internal modes—giving the toy enough structure to reveal differences in how guidance strategies handle within-class diversity.

Figure from AutoGuidance: the toy experiment comparing CFG and AG on multi-modal in-class distributions.

In this setting, CFG tends to collapse toward the high-density region of the class, sacrificing mode coverage. AG, by contrast, preserves coverage across all within-class modes. This aligns with their main conclusion on ImageNet: AG outperforms CFG in generation quality.

However, other work has reported that CFG remains more robust or even superior to AG in practice. What explains this discrepancy?

The core reason is that AG beats CFG specifically when the model is already well-trained—like EDM2 on ImageNet—where the conditional information is already well captured by the unguided model. In that regime, what remains to be improved is *in-class mode coverage*, which is exactly where AG excels.

And this is precisely what the toy is designed to show. There are only two classes, and for each class a separate diffusion model is trained—making it easy for the model to fit the task perfectly. When the number of classes is small and the model is well-fitted, CFG can only harm generation by pushing samples toward an exaggerated class signal. In this favorable setting, the advantage of AG over CFG becomes especially obvious.

The hidden assumption, then, is not in the method but in the toy: by choosing a scenario where the model is already near-perfect, the experiment naturally favors AG. Whether this advantage holds in harder settings—more classes, weaker models, noisier data—is a different question entirely.

A multi-class toy: Leaves

So far we have moved from a simple two-Gaussian mixture to AutoGuidance's richer multi-modal setup. Each step added realism, but each also locked in specific assumptions—few classes, well-fitted models, hand-picked regimes. To draw more general conclusions about guidance, we need a toy that lets us *independently* control two axes of difficulty that matter in practice:

1. **Class condition complexity** — how many distinct classes the model must separate.
2. **In-class distribution complexity** — how much internal structure each class contains.

Real datasets vary along both axes simultaneously: ImageNet has 1000 classes, each with diverse poses, backgrounds, and sub-types. A useful toy should let us dial these independently so we can isolate their effects on guidance strategies.

This is why we introduce the **Leaves dataset**: a recursive, tree-structured distribution parameterized by `num_classes` and `max_depth`. Each class is a branch system that grows from a shared trunk. Increasing `max_depth` adds recursive bifurcations, making within-class structure richer. Increasing `num_classes` adds more branches along the trunk, making the between-class discrimination harder. The notation `num_classes/max_depth` captures a setting—for example, `4/3` means 4 classes each with depth-3 branching, while `24/1` means 24 classes each with minimal internal structure.

Ground-truth distributions for three Leaves configurations. From left to right: fewer classes with deeper branching (richer in-class structure) to many classes with shallow structure (harder class separation, simpler in-class modes).

The three configurations illustrate the trade-off. In `4/3`, each class is a complex tree with deep recursive branches—the model must capture rich within-class geometry, but only needs to distinguish four classes. In `24/1`, there are many classes but each is a simple elongated shape—the challenge shifts to between-class discrimination. `12/2` sits in between. By sweeping this space, we can ask: *when does CFG help? When does AG win? And when do both fall short?*

Unlike the previous toys, we train a single conditional model on all classes jointly—matching the standard practice in real diffusion training. This means the model must share capacity across classes, and guidance must navigate a genuinely crowded label space rather than operating on a trivially fitted per-class model.

Generated samples (black dots) overlaid on ground-truth distributions across different guidance strategies for each Leaves configuration.

The Leaves dataset provides a controlled environment to systematically vary class complexity and in-class structure, enabling isolated evaluation of guidance methods across different regimes.

We're still limited

Even with a more principled toy like Leaves, fundamental gaps remain between what 2D distributions can show us and what actually matters in high-dimensional generative modeling.

Perceptual Loss: What the Microscope Can't See

A natural question is whether insights from toy experiments extend to training improvements—not just guidance strategies. To test this, we compare three training variants on the Leaves dataset:

- **Vanilla flow matching**: the standard objective.

$$\mathcal{L}_{\text{vanilla}} = \mathbb{E}\left[\sigma^2 \| s_\theta(\mathbf{x}, \sigma, \mathbf{c}) - (\epsilon - \mathbf{x}_0) \|_2^2\right]$$

- **+ Dispersive loss**: adds a regularizer that encourages the model to spread its predictions, penalizing collapsed covariance in the denoised estimates.

$$\mathcal{L}_{\text{dispersive}} = \mathcal{L}_{\text{vanilla}} + \lambda_{\text{disp}} \cdot \frac{1}{\text{Tr}(\text{Cov}(\hat{\mathbf{x}})) + \varepsilon}, \quad \hat{\mathbf{x}} = \mathbf{x} + \sigma^2 s_\theta(\mathbf{x}, \sigma, \mathbf{c})$$

- **+ Deep supervision**: provides auxiliary supervision at intermediate dit blocks using the same stochastic objective used in vanilla training.

$$\mathcal{L}_{\text{deep-supervision}} = \mathcal{L}_{\text{vanilla}} + \lambda_{\text{ds}} \cdot \mathbb{E}\left[\sigma^2 \left\| s_\theta^{\text{aux}}(\mathbf{x}, \sigma, \mathbf{c}) - (\epsilon - \mathbf{x}_0) \right\|_2^2\right]$$

Wasserstein distance during training for three loss variants on the Leaves dataset.

The results show that some regularizers (deep supervision) can help in this low-dimensional setting, while others (dispersive loss) may be neutral or even unstable depending on hyperparameters. But here is the critical point: techniques like dispersive loss and REPA are designed to operate on *representation geometry*—encouraging diverse, well-structured features in high-dimensional latent spaces. In a 2D toy, there is no meaningful representation space to regularize. The model maps directly from coordinates to scores, with no intermediate features where collapse or alignment could occur. The very mechanism these methods exploit simply does not exist in the toy setting.

This is not a flaw in the methods—it is a fundamental limitation of the microscope. Toy distributions can reveal distributional phenomena (mode coverage, class separation, guidance drift), but they are blind to *perceptual* and *representational* phenomena that only emerge in high dimensions. Any conclusion drawn from a toy about representation-level techniques should be treated with caution: absence of effect in 2D does not imply absence of effect in practice.

Unavoidable Preferences in Toy Construction

There is a second, more subtle limitation: the construction of any toy dataset requires extensive design choices and hyperparameter tuning—each of which encodes implicit preferences about what "realistic" means.

For the Leaves dataset alone, we had to decide: How many classes? What branching depth? What variance at each node? What angular spread between branches? How densely to sample? Each of these choices shapes the resulting distribution, and different choices can lead to different conclusions about which guidance method performs best. A toy setting where the class count is large and the model is under-trained will favor CFG; a toy setting where in-class distribution is complex and the model is well-trained will favor AG. The designer's choices—often made for aesthetic or practical reasons rather than principled ones—silently steer the outcome.

This is not unique to our construction. Every toy in the literature carries the same burden: the two-Gaussian mixture assumes symmetric, well-separated classes; AutoGuidance's multi-modal setup assumes a perfectly fitted per-class model. These are not neutral choices. They are *preferences*, baked into the experimental setup before any method is ever evaluated.

The honest conclusion is that no single toy can be authoritative. What toys *can* do is make their assumptions explicit and let readers judge which regime applies to their setting. What they *cannot* do is substitute for evaluation on real data.

Acknowledgements