Diffusion Domain Expansion: Learning to Coordinate Pre-Trained Diffusion Models

Egor Lifar [*1] Semyon Savkin [*1] Timur Garipov [1] Shangyuan Tong [1] Tommi Jaakkola [1]

Abstract

In this paper, we propose Diffusion Domain Expansion (DDE), a method that efficiently extends pre-trained diffusion models to generate larger objects and handle more complex conditioning beyond their original capabilities. Our method employs a compact trainable network designed to coordinate the denoised outputs of pre-trained diffusion models. We demonstrate that the coordinator can be universally simple while being capable of generalizing to domains larger than those observed during its training time. We evaluate DDE on long audio track generation and conditional image generation, demonstrating its applicability across domains. DDE outperforms other approaches to coordinated generation with diffusion models in qualitative and quantitative evaluations.

1. Introduction

Diffusion models have delivered state-of-the-art results in generating images (Betker et al., 2023), music (Evans et al., 2024), robot control trajectories (Chi et al., 2023), and molecular structures (Abramson et al., 2024). As training at scale incurs significant costs, research on the control and re-use of pre-trained models has gained substantial attention. Post-training tasks include conditional generation (Sohl-Dickstein et al., 2015; Zhang et al., 2023a), in-painting (Choi et al., 2021), editing (Meng et al., 2022), and composition of models (Du et al., 2023; Garipov et al., 2023)[1].

A recent line of work (Bar-Tal et al., 2023; Lee et al., 2023; Zhang et al., 2023b; Wu et al., 2024) focuses on expanding the generative domain of diffusion models to generate data samples with multiple parts (e.g., panoramic images) by coordinating pre-trained diffusion processes. These approaches prescribe fixed algorithmic coordination mechanisms, which do not require any additional training and can be applied off-the-shelf. They are, however, limited to specific tasks they are designed for. Huang et al. (2023) proposed to train a dynamic diffuser module to fuse unimodal diffusion models for multimodal conditioning. This method expands the generative domain by increasing the number of conditioning inputs but requires the modalities to be fixed at training time.

In this paper, we propose Diffusion Domain Expansion (DDE), a method for efficiently training small coordinators that extend the generative domain of pre-trained models along two axes: the size of the generated object (e.g., larger images) and the size of the conditioning object (e.g., multiple labels). We demonstrate the ability of coordinators to generalize to domains larger than those seen during training. We evaluate DDE on music and conditional image generation tasks, including CLEVR scene generation and satellite image generation. We show that our ViT-based coordinator delivers high-quality results by learning long-range dependencies while enforcing local consistency via MultiDiffusion-like updates.

2. Background

Given a training data distribution $p_{\text{data}}(x)$ over $\mathbb{R}^d$, a diffusion process $p(x;t)$ is a time-indexed collection of distributions such that $p(x;t) = \int \mathcal{N}(x; x_0, \sigma^2(t)) p_{\text{data}}(x_0)\, dx_0$ is constructed by adding Gaussian noise with standard deviation $\sigma(t)$ to the "clean" samples from the data distribution ($x(t) = x_0 + \varepsilon; \varepsilon \sim \mathcal{N}(0, \sigma^2(t))$). In the following, we set $\sigma(t) = t$ and consider $t \in [t_{\min}, t_{\max}]$. We refer the reader to (Karras et al., 2022) for the discussion of other possible specifications of diffusion processes.

A diffusion model learns a denoising function $D : \mathbb{R}^d \times [t_{\min}, t_{\max}] \to \mathbb{R}^d$ which estimates the "clean" object given its noisy observation at time t. This function is trained by minimizing the denoising error

$$\mathop{\mathbb{E}}_{t \sim p(t)} \mathop{\mathbb{E}}_{x \sim p_{\text{data}}(x)} \mathop{\mathbb{E}}_{\varepsilon \sim \mathcal{N}(0, t^2)} \left[\lambda(t) \left\| D(x+\varepsilon, t) - x \right\|_2^2 \right], \quad (1)$$

where $p(t)$ is a distribution over $[t_{\min}, t_{\max}]$ and $\lambda(t) > 0$.

[*]Equal contribution [1]MIT CSAIL, Cambridge, USA. Correspondence to: Egor Lifar <egor.lifar@gmail.com>.

Accepted by the Structured Probabilistic Inference & Generative Modeling workshop of ICML 2024, Vienna, Austria. Copyright 2024 by the author(s).

[1]See Appendix A for a detailed review of related work.

The denoising function that minimizes (1) is connected to the score $\nabla_x \log p(x,t)$ of the diffusion process:

$$\nabla_x \log p(x,t) = \big(D(x,t) - x\big) / t^2. \tag{2}$$

It can be shown (Anderson, 1982; Song et al., 2021; Karras et al., 2022) that, given access to the score $\nabla_x \log p(x,t)$, the distributions $p(x,t)$ can be recovered via the probability flow induced by the backward ODE

$$dx = -t\, \nabla_x \log p(x,t)\, dt, \tag{3}$$

which is initialized with $x(t_{\text{max}}) \sim p(x, t_{\text{max}})$ and runs backwards in time from $t = t_{\text{max}}$ to $t = t_{\text{min}}$.

If t_{max} is large enough $p(x, t_{\text{max}})$ can be accurately approximated by a Gaussian prior $p_{\text{prior}}(x) = \mathcal{N}(0, t^2_{\text{max}})$. After $D(\cdot,\cdot)$ is trained with the objective (1), the samples can be generated by sampling $x(t_{\text{max}}) \sim p_{\text{prior}}$ and numerically integrating the backward ODE (3) with the score (2).

3. Method

3.1. General formulation

Consider a pre-trained conditional diffusion model $p(x|y)$ with the denoising network $D(x,y,t)$. We are interested in expanding the generative domain, i.e., solving the conditional generation $p(X_{[L]}|Y_{[L]})$, where $X_{[L]}$ and $Y_{[L]}$ are the expanded generated object (e.g., a larger image) and the expanded conditioning input (e.g., a larger conditioning image). Naturally, we want to re-use the pre-trained model $p(x|y)$ and realize the generation in the expanded domain by employing the knowledge captured by $D(x,y,t)$. We utilize the assumption that the expanded output-input pair $(X_{[L]}, Y_{[L]})$ can be decomposed into a collection of smaller parts $([x_1,\dots,x_L],[y_1,\dots,y_L]) = F(X_{[L]}, Y_{[L]})$, where all $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ are in the respective base domains and L denotes the number of smaller parts. The mapping $F(\cdot,\cdot)$ defines the decomposition process. For instance, in image generation $X_{[L]}$ corresponds to a larger image from which we can extract a set of L smaller same-sized patches $[x_1,\dots,x_L]$ that cover the large image completely. In our framework, the patches are allowed to have overlaps.

Each of the smaller objects x_i can be generated using $p(x_i|y_i)$. However, to generate a coherent large object $X_{[L]}$, the generative processes need to be coordinated. The idea of our approach is to train a coordinator network $C_{[\cdot]}$ that learns to coordinate diffusion processes by operating on the outputs of the pre-trained denoising networks $[D(x_i,y_i,t)]_{i=1}^L$. We use a parameter-efficient architecture for the coordinator network $C_{[\cdot]}$ and we train it on a dataset of larger objects $\mathcal{D}^N_{L_{\text{train}}} = \{(X^{(i)}_{[L_{\text{train}}]}, Y^{(i)}_{[L_{\text{train}}]})\}_{i=1}^N$ of size L_{train}. Crucially, we demonstrate that after being trained on examples of size L_{train}, the coordinated diffusion $p(X_{[\cdot]}|Y_{[\cdot]})$ induced by $C_{[\cdot]}$ can generalize and generate objects of larger sizes $L_{\text{test}} \geq L_{\text{train}}$. We describe the general formulation and the training objective of DDE below, and we describe the coordinator architecture in Section 3.2.

The DDE generative process is the reverse of the noising process $X_{[L]}(t) \sim \mathcal{N}(X_{[L]}(0), t^2)$ over expanded data $X_{[L]}$. For this expanded process, we build a composite denoiser $D_{[\cdot]}$ which utilizes the decomposition function F, the pre-trained base domain denoiser network D, and a trainable small-scale coordinator network $C_{[\cdot]}$. Specifically, for an extended domain output-input pair $(X_{[L]}, Y_{[L]})$ of size L, we define the composite denoiser $D_{[L]}$ as

$$\begin{aligned} &D_{[L]}(X_{[L]}(t), Y_{[L]}, t) = \\ &\qquad C_{[L]}\Big([D(x_i(t), y_i, t)]_{i=1}^L, [y_i]_{i=1}^L, t\Big), \end{aligned} \tag{4}$$

where $([x_i(t)]_{i=1}^L, [y_i]_{i=1}^L) = F(X_{[L]}(t), Y_{[L]})$. The coordinator network $C_{[\cdot]}$ takes two sequences as inputs: 1) the outputs $[D(x_i(t), y_i, t)]_{i=1}^L$ of the pre-trained denoiser D evaluated on the smaller objects; 2) the conditioning information $[y_i]_{i=1}^L$, and produces an estimate of $X_{[L]}(0)$.

We train the coordinator by minimizing the denoising error (1) of the composite denoiser (4) on a training dataset $\mathcal{D}^N_{[L_{\text{train}}]}$ of objects of size L_{train}.

$$\begin{aligned} \mathcal{L}_{[L_{\text{train}}]} = &\mathop{\mathbb{E}}_{t\sim p(t)} \mathop{\mathbb{E}}_{(X_{[L_{\text{train}}]}, Y_{[L_{\text{train}}]}) \sim \mathcal{D}^N_{[L_{\text{train}}]}} \mathop{\mathbb{E}}_{\varepsilon \sim \mathcal{N}(0;t^2)} \Big[\\ &\lambda(t) \big\| D_{[L_{\text{train}}]}(X_{[L_{\text{train}}]} + \varepsilon, Y_{[L_{\text{train}}]}, t) - X_{[L_{\text{train}}]} \big\|_2^2 \Big]. \end{aligned} \tag{5}$$

After the coordinator $C_{[\cdot]}$ has been trained, we use it to generate objects of sizes $L_{\text{test}} \geq L_{\text{train}}$ to evaluate the generalization to the generation of larger objects.

3.2. Coordinator Architecture

For each of our experimental domains, we show that a visual transformer is a suitable choice for the architecture of the coordinator C, allowing for better generalization, faster training, and a lower number of parameters compared to the base model. We choose to use the architecture from (Peebles & Xie, 2023), adapting it to the particular domains. For positional encodings of tokens for larger object generation, we used Rotary Position Embedding (Su et al., 2023).

To generate larger objects, we cover them with overlapping patches of a size equal to the generation capability of the base model and pass down the predicted denoised versions of the patches during training into the ViT. Our transformer architecture encodes the positional encoding of the resulting tokens relative to the larger object, and during the forward process, we reconcile the predicted overlapped patches with their mean value for each position, inspired by the sampling method from MultiDiffusion (Bar-Tal et al., 2023). The architecture of the coordinator can be seen in Figure 1.

Table 1: FAD values for long music track generation on Slakh2100 task for different architectures. The base model was trained for 80 epochs and has $405M$ parameters.

Method	Size	FAD for $4l$	FAD for $10l$
Concat	-	4.623	4.596
MultiDiffusion	-	4.732	4.796
RNN	$16M$	4.223	4.081
RNN with overlaps	$50M$	4.447	4.424
DDE (ours)	$66M$	**2.112**	**2.142**

Table 2: Accuracy on the 256 generated samples for various methods and numbers of conditionings on CLEVER conditional image generation.

Model	**Sampler**	**Coordination**				
		1	2	3	4	5
RRR	Euler	98.0	93.8	72.3	48.0	23.0
	Heun	**98.0**	**94.5**	83.2	63.3	33.6
MultiDiffusion	Euler	96.9	94.1	76.2	43.0	25.0
	Heun	97.7	93.4	80.5	58.6	36.3
DDE (ours)	Heun	96.5	94.1	**86.3**	**66.8**	**44.5**

Table 3: Evaluation results for satellite image generation.

Method	FID, $N = 96$	FID, $N = 128$
MultiDiffusion	37.815	35.016
DDE (ours)	**31.753**	**27.373**

4. Experiments

We set up the experiments to highlight the ability of a coordinator model trained on the denoised outputs to perform diverse tasks. In music generation, we pretrain models that can generate short tracks and then generate larger tracks by coordinating these models. In the CLEVR dataset, we perform conditioned sampling with multiple conditionings by coordinating several models that are capable of handling only one conditioning. Finally, in the satellite image generation domain, we combine both the idea of generating a larger object and applying conditionings.

4.1. Music Generation

First, we pre-trained a base model with UNet architecture on Slakh2100 (Manilow et al., 2019) capable of generating music tracks of length l, which translates into 12 seconds of music. In this experiment, we chose $L_{\text{train}} = 5$ and $L_{\text{test}} = 13$. We evaluate the method by computing FAD (Kilgour et al., 2019) between the generated long tracks and the sampled tracks of the same length from the dataset.

Our baselines are concatenation and MultiDiffusion (Bar-Tal et al., 2023). The concatenation method applies a pre-trained model on parts of the large track independently and concatenates the results during inference. The MultiDiffusion method decomposes a large track into a sequence of overlapping patches. In each step of the diffusion process, the pre-trained model is run on each of the patches, and the score at each position is the mean of all scores from the patches that contain this position.

We report the FAD metric based on 128 generated samples for various architectures, and 48 and 120 seconds of music generation (Table 1). We describe all the architectures we trained in detail in Appendix B.2. To train and test RNN with overlaps and DDE, for lengths of $4l$ and $10l$, we select 5 and 13 patches of length l accordingly, so that the overlap of two adjacent patches has a length of $\frac{l}{4}$. We note that, despite not having access to tracks of length $10l$ during training, the coordinator architectures were able to generalize and generate coherent long tracks while performing better than both concatenation and MultiDiffusion. One property we observe is that the coordinator models are much smaller than the base model, and their training converges faster. ViT, which we used in DDE, performs significantly better in terms of the metric on the tracks of both lengths, compared to other methods.

4.2. CLEVR Image Generation

To test the ability of DDE to expand the number of conditioning inputs, we conducted experiments with the conditional generation problem used in (Du et al., 2023) on the CLEVR dataset (Johnson et al., 2016). In this problem, the pre-trained model is a UNet-based conditional diffusion model $p(x|c)$, where x is a 64×64 image and c is conditioning information consisting of a pair of Euclidean coordinates (c^x, c^y). The condition corresponds to the constraint of the image having an object centered at position (c^x, c^y), in addition to possibly having other objects located elsewhere. The goal of the problem is to generate images $x \sim p(x|[c_i]_{i=1}^L)$ conditioned on objects being present at positions $[c_i]_{i=1}^L$. In the notation of Section 3.1, the expanded generative domain in this problem is $Y_{[L]} = [c_1, \dots, c_L]$, $X_{[L]} = [x, \dots, x]$: generating one image x conditioned on multiple positions $[c_i]_{i=1}^L$. We evaluate the generated samples x using a classifier $\text{CLS}(x, c)$, which, given an image x and a position c, outputs a probability of an object being present in the image at position c. Following Du et al. (2023), we trained base diffusion models that realize conditional, unconditional, and classifier-free guided sampling (Ho & Salimans, 2021) with a shared parameterization. We trained a ViT-based coordinator $C_{[\cdot]}$. We provide the details about base model training in Appendix B.2.2, and about the coordinator architecture and training in Appendix B.2.4.

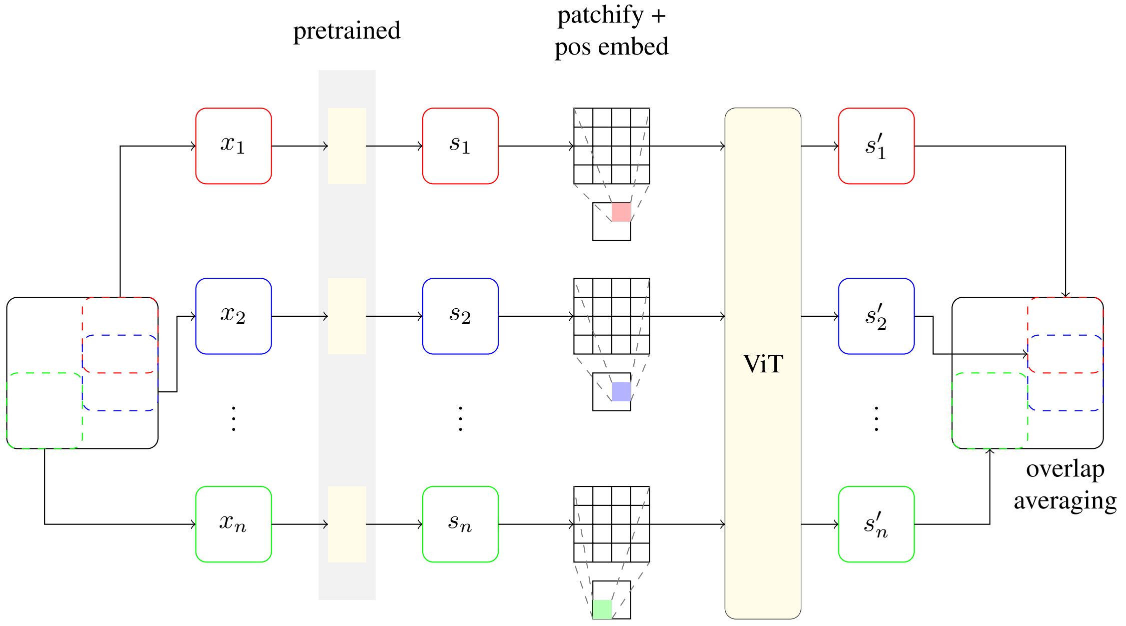

Figure 1: Overview of the architecture. A large image is decomposed into a set of overlapping patches; each patch is processed by a separate pre-trained denoising network. The denoised outputs of pre-trained models are patchified and augmented with global positional encoding. The coordinator processes all patches and produces a new coordinated set of output patches. The final expanded denoised output is constructed from coordinator outputs by averaging the values in the overlaps.

During training, the coordinator observed conditioning objects of size up to $L_{\text{train}} = 2$. For evaluation, we tested conditional generation with conditioning objects of sizes up to $L_{\text{test}} = 5$, and calculated the overall accuracy for each number of conditionings; see Appendix B.2.6 for details.

We considered two baseline methods. For the first baseline, we considered the approach developed in (Du et al., 2023) (denoted "RRR"), which combines conditioned and unconditioned scores of the pre-trained model on several conditionings. We provide the concrete formula in Appendix B.2.6. As another baseline, we evaluated how the method from MultiDiffusion works by simply averaging the outputs of the pre-trained model over all conditionings. Figure 2 shows samples from the RRR model and our Coordinator model.

Table 2 shows the conditional generation accuracy of our method and the baselines with various sampling methods. Our coordinator outperforms the baselines significantly, especially when conditioned on 4 or 5 labels. The results demonstrate that the transformer architecture can generalize more conditioning inputs than observed during training. Our architecture also uses slightly fewer parameters than the base model ($38M$ versus $42M$) and uses significantly fewer epochs of training (20 versus 1050).

4.3. Map Image Generation

Our final experimental domain is conditional image generation. We chose the task of generating satellite images conditioned on schematic maps. We collected a new dataset for this task using the Google Maps API.

We pretrain a model that can sample patches of size $B \times B$ where $B = 64$. Our decomposition function selects $B \times B$-size patches from a larger image with a stride $s = 32$. We train the coordinator on size $N \times N$, where $N = 96$, and perform evaluation for out-of-distribution sampling with $N = 128$. We use FID between the generated conditional samples and the set of all ground truth satellite images as the evaluation metric. The baseline with which we are comparing our method is MultiDiffusion.

The base model is trained for 200 epochs and has 270M parameters. The coordinator has 26M parameters and is trained for 20 epochs. Table 3 displays the value of the FID metric for both the training image size ($N = 96$) and the out-of-distribution sampling size ($N = 128$). Our method outperforms MultiDiffusion by FID.

In Figure 3, we present samples from DDE and multidiffusion. We note that top-right part of the multidiffusion sample that lies within only one patch is visually separable from the rest of the image. In Appendix B.3.8, we show more DDE samples.

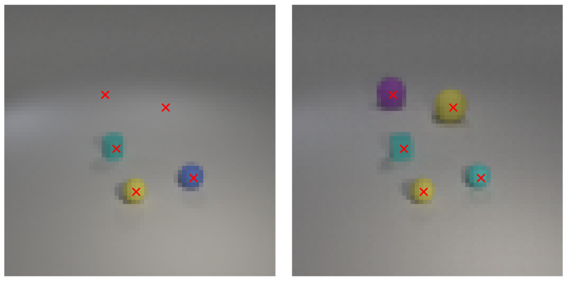
Figure 2: Generated sample from the RRR model (left) and Coordinator model (right). Red crosses correspond to the conditionings.

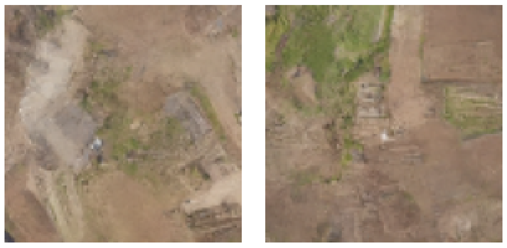
Figure 3: Satellite image generation example: DDE sample on the left and MultiDiffusion sample on the right.

5. Conclusion

In this paper, we proposed Diffusion Domain Expansion (DDE), a method that utilizes a ViT network as a coordinator of outputs from pre-trained diffusion models to expand the generative domain. We demonstrated that our method allows for the generation of larger objects and the combination of multiple conditioning inputs simultaneously. Our experiments indicate that DDE enables the coordinator to generalize to larger instances unseen during training. Lastly, our model requires less training time and fewer parameters to converge to the optimal state compared to the base pre-trained models.

Limitations. The main limitation of our method is the need for additional data and time to train the coordinator model.

Future work. We plan to explore the application of our method in other domains. Potential areas include generating long coherent protein sequences and creating videos from short clips. In addition, we are interested in exploring alternative ways of supervision for the coordinator model.

References

Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A. J., Bambrick, J., et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pp. 1–3, 2024.

Anderson, B. D. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Bar-Tal, O., Yariv, L., Lipman, Y., and Dekel, T. Multidiffusion: Fusing diffusion paths for controlled image generation. In *International Conference on Machine Learning*, pp. 1737–1752. PMLR, 2023.

Ben-Hamu, H., Puny, O., Gat, I., Karrer, B., Singer, U., and Lipman, Y. D-flow: Differentiating through flows for controlled generation. *arXiv preprint arXiv:2402.14017*, 2024.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

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

Black, K., Janner, M., Du, Y., Kostrikov, I., and Levine, S. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=YCWjhGrJFD.

Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Choi, J., Kim, S., Jeong, Y., Gwon, Y., and Yoon, S. Ilvr: Conditioning method for denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14367–14376, October 2021.

Chung, H., Sim, B., Ryu, D., and Ye, J. C. Improving diffusion models for inverse problems using manifold constraints. *Advances in Neural Information Processing Systems*, 35:25683–25696, 2022.

Clark, K., Vicol, P., Swersky, K., and Fleet, D. J. Directly fine-tuning diffusion models on differentiable rewards. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=1vmSEVL19f.

Corso, G., Xu, Y., Bortoli, V. D., Barzilay, R., and Jaakkola, T. S. Particle guidance: non-i.i.d. diverse sampling with diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KqbCvIFBY7.

Couairon, G., Verbeek, J., Schwenk, H., and Cord, M. Diffedit: Diffusion-based semantic image editing with mask guidance. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3lge0p5o-M-.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Du, Y., Li, S., and Mordatch, I. Compositional visual generation with energy based models. *Advances in Neural Information Processing Systems*, 33:6637–6647, 2020.

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 *International Conference on Machine Learning*, pp. 8489–8510. PMLR, 2023.

Evans, Z., Carr, C., Taylor, J., Hawley, S. H., and Pons, J. Fast timing-conditioned latent audio diffusion. *arXiv preprint arXiv:2402.04825*, 2024.

Gal, R., Alaluf, Y., Atzmon, Y., Patashnik, O., Bermano, A. H., Chechik, G., and Cohen-or, D. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NAQvF08TcyG.

Garipov, T., Peuter, S. D., Yang, G., Garg, V., Kaski, S., and Jaakkola, T. S. Compositional sculpting of iterative generative processes. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=w79RtqIyoM.

Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., and Cohen-or, D. Prompt-to-prompt image editing with cross-attention control. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=_CDixzkzeyb.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

Huang, Z., Chan, K. C., Jiang, Y., and Liu, Z. Collaborative diffusion for multi-modal face generation and editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6080–6090, June 2023.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging weights leads to wider optima and better generalization. 2019.

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, 2016.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35: 26565–26577, 2022.

Kawar, B., Elad, M., Ermon, S., and Song, J. Denoising diffusion restoration models. *Advances in Neural Information Processing Systems*, 35:23593–23606, 2022.

Kawar, B., Zada, S., Lang, O., Tov, O., Chang, H., Dekel, T., Mosseri, I., and Irani, M. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6007–6017, 2023.

Kilgour, K., Zuluaga, M., Roblek, D., and Sharifi, M. Fréchet audio distance: A metric for evaluating music enhancement algorithms, 2019.

Lee, Y., Kim, K., Kim, H., and Sung, M. Syncdiffusion: Coherent montage via synchronized joint diffusions. *Advances in Neural Information Processing Systems*, 36: 50648–50660, 2023.

Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Van Gool, L. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11461–11471, June 2022.

Manilow, E., Wichern, G., Seetharaman, P., and Roux, J. L. Cutting music source separation some slakh: A dataset to study the impact of training data quality and quantity, 2019.

Mariani, G., Tallini, I., Postolache, E., Mancusi, M., Cosmo, L., and Rodolà, E. Multi-source diffusion models for simultaneous music generation and separation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=h922Qhkmx1.

Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.-Y., and Ermon, S. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*,

2022. URL https://openreview.net/forum?id=aBsCjcPu_tE.

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

Peebles, W. and Xie, S. Scalable diffusion models with transformers, 2023.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=HPuSIXJaa9.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831. PMLR, 2021.

Ren, M., Delbracio, M., Talebi, H., Gerig, G., and Milanfar, P. Multiscale structure guided diffusion for image deblurring. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10721–10733, October 2023.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., and Aberman, K. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22500–22510, 2023.

Saharia, C., Chan, W., Chang, H., Lee, C., Ho, J., Salimans, T., Fleet, D., and Norouzi, M. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–10, 2022a.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35: 36479–36494, 2022b.

Sarukkai, V., Li, L., Ma, A., Ré, C., and Fatahalian, K. Collage diffusion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 4208–4217, 2024.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding, 2023.

Trippe, B. L., Yim, J., Tischer, D., Baker, D., Broderick, T., Barzilay, R., and Jaakkola, T. S. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6TxBxqNME1Y.

Wallace, B., Dang, M., Rafailov, R., Zhou, L., Lou, A., Purushwalkam, S., Ermon, S., Xiong, C., Joty, S., and Naik, N. Diffusion model alignment using direct preference optimization. *arXiv preprint arXiv:2311.12908*, 2023.

Whang, J., Delbracio, M., Talebi, H., Saharia, C., Dimakis, A. G., and Milanfar, P. Deblurring via stochastic refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16293–16303, June 2022.

Wu, T., Maruyama, T., Wei, L., Zhang, T., Du, Y., Iaccarino, G., and Leskovec, J. Compositional generative inverse design. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wmX0CqFSd7.

Zhang, L., Rao, A., and Agrawala, M. Adding conditional control to text-to-image diffusion models, 2023a.

Zhang, Q., Song, J., Huang, X., Chen, Y., and Liu, M.-Y. Diffcollage: Parallel generation of large content with diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10188–10198. IEEE, 2023b.

A. Related Work

Controllable generation. Generation control and guidance of diffusion models is an active area of research. One way to formalize controllable generation is to cast it as sampling from a conditional distribution $p(x|y)$. Popular types of conditioning annotations y include class labels (Dhariwal & Nichol, 2021), text prompts (Ramesh et al., 2021; Rombach et al., 2022; Saharia et al., 2022b), and semantic maps (Rombach et al., 2022; Huang et al., 2023). Method-wise, a straightforward way to achieve conditional generation $p(x|y)$ is to train a conditional denoising network $D(x, y, t)$ on pairs (x, y) of generation target x and conditioning information y available at training time. Classifier-free guidance (Ho & Salimans, 2021) enhances image fidelity by combining conditional and unconditional score functions.

Another family of methods enables generation conditioned on information that was not specified at model training time. Classifier guidance (Sohl-Dickstein et al., 2015; Dhariwal & Nichol, 2021; Song et al., 2021) generates samples by combining the learned score function and the gradient of the log-probabilities of a separately trained classifier $p(y|x, t)$. Zhang et al. (2023a) developed ControlNet, a method that employs a supplementary trained network to enable a pre-trained diffusion model to process previously inaccessible conditioning inputs. Collaborative Diffusion (Huang et al., 2023) trains a lightweight module which combines scores of multiple diffusion models, each supporting different conditioning modalities, to enable multi-modality conditioning.

Image inpainting is an extensively explored controllable generation task. The goal of inpainting is to restore or reconstruct missing or corrupted parts of an image, or, formally, sampling $x_{\text{hidden}}|x_{\text{obs}} \sim p(x_{\text{hidden}}|x_{\text{obs}})$, the unobserved part of the image x_{hidden} given the observed part x_{obs}. Song et al. (2021); Lugmayr et al. (2022) proposed inpainting techniques based on the "replacement" of observed pixels with their known values (with noise applied at an appropriate) at each step of the diffusion denoising process. Chung et al. (2022); Ho et al. (2022) utilized "reconstruction"-based approaches which introduce an additional guiding term in the denoising update with the goal of bringing the values in the observed part of the image closer to the known target values. Trippe et al. (2023) developed a method based on particle filtering for inpainting of protein backbones in 3D. Saharia et al. (2022a) proposed a diffusion model trained specifically for image-to-image translation tasks including inpainting.

The task of inpainting is a member of the family of inverse problems, which can be cast as conditional generation $x|y(x) \sim p(x|y(x))$, where $y(x)$ is an observation function that extracts a limited information summary from x (e.g., $y(x)$ might represent a downsampled version of image x). A line of work (Choi et al., 2021; Song et al., 2021; Kawar et al., 2022; Saharia et al., 2022a; Whang et al., 2022; Ren et al., 2023) is focused on diffusion-based solutions for inverse problems in the image generation domain. Recently, Mariani et al. (2024) used diffusion models for the inverse problem of audio source separation. Ben-Hamu et al. (2024) utilized the probability flow interpretation of diffusion models and addressed image and audio inverse problems and conditional molecular generation via differentiation through ODE sampler.

Image editing is another instance of controllable generation problems. In this case, the goal is to modify specific aspects of an existing image according to user-specified goals while preserving image coherence and fidelity. To address semantic image editing, Meng et al. (2022) and Couairon et al. (2023) proposed to first partially noise and then denoise an image to generate an edited version, possibly conditioned on a segmentation mask (Couairon et al., 2023). Collage diffusion (Sarukkai et al., 2024) is a diffusion model extension that processes multiple base images accompanied by text prompts and desired locations and produces a collage by editing the base images and combining layers of edited images. Another line of work (Ruiz et al., 2023; Gal et al., 2023; Hertz et al., 2023; Kawar et al., 2023) introduced techniques for personalization, concept learning, prompt manipulation, and direct editing of real images guided by natural language inputs.

Fine-tuning diffusion models with reward functions or human preference data has emerged as a promising form of controllable generation. Black et al. (2024) interpreted diffusion as a sequential decision process and employed reinforcement learning to optimize objectives such as image compressibility, prompt-image alignment (based on vision-language model feedback), and aesthetic quality (derived from human feedback). Clark et al. (2024) used backpropagation through samplers to optimize differentiable rewards such as scores from human preference models. Wallace et al. (2023) adapted Direct Preference Optimization (Rafailov et al., 2023) to diffusion models, which enables fine-tuning diffusion models on human preference data directly without the need for the auxiliary reward model.

Diffusion composition and coordination. Compositional generation approaches combine multiple diffusion processes to control sampling distributions, re-use pre-trained models, and extend their capabilities. Existing works on compositional generation differ in how they approach the composition of diffusion processes. Notable approaches include exact specification of adjusted distributions (Du et al., 2023; Garipov et al., 2023; Wu et al., 2024), or resolution of individual steps of generation

(Bar-Tal et al., 2023; Lee et al., 2023; Huang et al., 2023; Zhang et al., 2023b; Corso et al., 2024).

Garipov et al. (2023) proposed a set of operations on iterative generative processes: diffusion models and GFlowNets (Bengio et al., 2021). These operations enable one to emphasize or de-emphasize high-likelihood regions of selected base models. The proposed Compositional Sculpting method employs classifier guidance and mixture processes to compose models and realize samples from composite distributions.

Wu et al. (2024) built on compositions of energy-based diffusion models (Du et al., 2020; 2023) and developed methods for the generation of solutions to inverse design problems: multi-body physical simulations and 2D airfoil design. The compositional generation enables the generation of objects more complex than those seen at training time. In particular, the authors demonstrate the generation of trajectories with 1) longer simulation horizons (via summation of energies of short-horizon models), 2) larger number of interacting bodies in physical simulations (via summation of energy functions for pairwise interactions), and 3) larger spatial simulation domains and the larger number of parts for joint multiple airfoil design.

MultiDiffusion (Bar-Tal et al., 2023) performs controlled image generation via the fusion of diffusion sampling paths. The authors address the generation of panoramic images and scenes with complex structures. The complex scene is divided into several regions, and the desired content of each region is described with a specific text prompt. The generation is performed via the coordination of multiple prompt-conditioned generation paths. The coordination method maintains a shared global image for the whole space and reconciles contradicting updates in the intersections of regions by averaging the updates of individual models whose regions cover the specific intersection.

SyncDiffusion (Lee et al., 2023) is an extension of MultiDiffusion that aims to address the issue of incoherent patches in large panoramic images. To that end, SyncDiffusion introduces an additional step to the MultiDiffusion update to perform a local optimization step on the perceptual similarity across patches.

Zhang et al. (2023b) extended panoramic generation to more general scenarios of large image generation supporting arbitrary graph structure of the overlapping patches comprising the large image (e.g., linear chain, cycle, grid, cubemap). The proposed method, called DiffCollage, translates a given graph structure into a closed-form formula for the total score expressed through marginal scores of individual patches. Zhang et al. (2023b) empirically demonstrated that coordinating multiple diffusion processes run in parallel outperforms inpainting-based panoramic image generation (Lugmayr et al., 2022; Chung et al., 2022) in terms of image quality and generation speed.

Corso et al. (2024) proposed Particle Guidance, a method for improving sample efficiency in diffusion models by running multiple diffusion chains with a time-evolving repulsion force that promotes sample diversity.

Mariani et al. (2024) demonstrated that the combination of multiple single-instrument diffusion models outperforms a joint model in the music source separation problem. The proposed source separation approach is based on the guidance methods for inverse problems proposed in (Song et al., 2021).

B. Experiment Details

All training for both the base and coordinator models was performed using the EDM framework introduced by (Karras et al., 2022) on a single A100 GPU. To enhance the quality of the generated samples, we trained the coordinator using EMA for the model weights. As shown by (Izmailov et al., 2019) and (Karras et al., 2022), this approach improves generalization and made training more stable in terms of metric values per epoch during evaluation.

For sampling from our coordinator models, we primarily used the Heun sampler with 2nd order correction, which we imported from the code provided by (Karras et al., 2022).

B.1. Music Generation Experiments

B.1.1. Dataset description

For the music domain, we used the version of the Slakh2100 dataset released by Mariani et al. (2024) for training both the base model and coordinator models. This dataset contains 1,500 tracks in the training set, each consisting of four different stems: Bass, Drums, Guitar, and Piano.

B.1.2. BASE MODEL DETAILS

For training the base model, we used the same UNet1d architecture and data loader as in (Mariani et al., 2024). The hyperparameters were set as follows: $channels = 256$, $patch\_factor = 16$, $patch\_blocks = 1$, $resnet\_groups = 8$, $kernel\_sizes\_init = [1, 3, 7]$, $multipliers = [1, 2, 4, 4, 4, 4, 4]$, $factors = [4, 4, 4, 2, 2, 2]$, $num\_blocks = [2, 2, 2, 2, 2, 2]$, $attentions = [False, False, False, True, True, True]$, $attention\_heads = 8$, $attention\_features = 128$, and $attention\_multiplier = 2$. This architecture takes as input four different stems of length 2^{18}, corresponding to roughly 12 seconds of music at a sampling rate of 22,050 Hz, and produces output of the same size.

We used the Adam optimizer with a learning rate of 10^{-4}, with $betas = (0.9, 0.99)$, and trained the model for 300 epochs with a batch size of 8. We sampled the noise level σ from a log-normal distribution with a mean of -3 and a standard deviation of 1.0, and we did the same for the training of coordinator models.

B.1.3. SAMPLER DETAILS

To sample the tracks for all the methods and architectures, we used the Heun sampler described in Appendix B with $\sigma_{\max} = 20$, $\sigma_{\min} = 10^{-4}$, and $S_{\text{churn}} = 20.0$. For σ_t, we used the Karras schedule with $\rho = 7$ and 150 timesteps.

B.1.4. EVALUATION

To evaluate the performance of all the models, we calculated the FAD (Fréchet Audio Distance) to the training dataset. To calculate the FAD metric for the sampled tracks of length l, we deterministically cut out the middle parts of the tracks from the training dataset with the same length, as the FAD metric is track-length dependent. In the implementation of the metric calculation we used, a pre-trained VGG-like architecture for music was used. Since the sampling rate was different from the required one, we resampled tracks from 22,050 Hz to 16,000 Hz. We tested how different numbers of sampled tracks affect the FAD score and found that generating 128 tracks differs from generating 1,024 tracks by a small margin, and the relative order of the compared models does not change. To decrease GPU usage, we settled on using 128 tracks in our reported metric values. We reported the saved checkpoint with the smallest FAD value for each of the methods.

B.1.5. RNN AND RNN WITH OVERLAPS DETAILS

We explored different methods of coordinating the denoised tracks via trainable networks. The first method we propose is a combination of RNN and UNet architectures. We choose a constant h—the number of hidden channels—and train a function $f : \mathbb{R}^{h \times L} \times \mathbb{R}^{S \times L} \to \mathbb{R}^{h \times L} \times \mathbb{R}^{S \times L}$ that is parametrized as a UNet with $h + S$ input channels and $h + S$ output channels. Then, let the predicted denoised tracks before reconciliation be $x_0, \ldots, x_{k-1}$. We set $h_0 = 0$, and $(h_{i+1}, x'_i) = f(h_i, x_i)$ for $i = 0, 1, \ldots, k-1$. The output of the RNN denoiser is the concatenation of x'_i, $i = 0, 1, \ldots, k-1$.

What we defined as RNN with overlaps corresponds to training a similar architecture, which takes patches of music tracks with overlaps of 3 seconds and performs a similar reconciliation by averaging the predicted fixed patches, as in MultiDiffusion. For both of the architectures, we used $h = 16$ hidden channels. We noted that a higher number of hidden channels compared to the number of output channels improved the FAD.

We used the Adam optimizer with a learning rate of 10^{-4}, with $betas = (0.9, 0.99)$, and trained the models for 10 epochs with a batch size of 4. For the training dataset, we used sampled tracks of length 48 seconds from the dataset, which correspond to 4 non-overlapping or 5 overlapping patches. Despite RNN with overlaps having a slightly worse metric compared to RNN, using overlaps improved the quality of tracks by removing the slightly noticeable transition between different patches.

B.1.6. VIT DETAILS

Finally, we tested our ViT architecture. For music, we used the ViT architecture with $patch\_size = 128$, $hidden\_size = 768$, $depth = 6$, $num\_heads = 6$, and $mlp\_ratio = 4.0$.

We used the Adam optimizer with a learning rate of $3 \cdot 10^{-5}$, with $betas = (0.9, 0.99)$, and trained the models for 10 epochs with a batch size of 4. In addition, for ViT, we obtained the best results using EMA. We tried using EMA for the RNN-like architectures, but it did not improve the metric values.

We chose these three architectures for comparison with the baseline because they can generalize easily enough for the generation of longer tracks, and we chose RNN to compare with ViT, as it requires fewer parameters for training.

We include generated samples from ViT, MultiDiffusion and Concat for both 48 and 120 seconds music track length in the supplementary materials.

B.2. CLEVR Image Generation Experiments

B.2.1. Dataset description

For the training of the base model, we used the CLEVR dataset released by the authors of (Du et al., 2023). This dataset consists of 30,000 images of size 128 by 128 pixels, which we downsampled to 64 by 64 pixels. For each image in the dataset, we had one conditioning information available, corresponding to the coordinates of the center of one of the objects in the image.

B.2.2. Base model details

For the training of the base model with one conditioning, we used a UNet with attention architecture from (Du et al., 2023). We used the following hyperparameters for the UNet: $model\_channels = 128$, $num\_res\_blocks = 2$, $channel\_mult = (1, 2, 2, 2)$, $num\_heads = 4$, and $num\_head\_channels = 64$. We improved the positional encoding of the conditioning by using Gaussian Fourier projection, projecting each of the coordinates into the dimension of $2 \cdot model\_channels$.

We used the Adam optimizer with a learning rate of 10^{-4} and trained for 1,050 epochs with a batch size of 32. We used the original dataset to train the model to satisfy one conditioning and used classifier-free guidance to improve the quality of generated samples. We sampled the noise level σ from a log-normal distribution with a mean of -1 and a standard deviation of 1.6, and we did the same for the training of coordinator models.

The base network $D(x, c, t)$ can optionally take a masked-out conditioning input $D(x, \varnothing, t)$ corresponding to unconditional generation. During coordinator training, we mask out the conditioning information with a probability of 10%. At generation time, we use classifier-free guidance with weight w:

$$(1 + w) \cdot C_{[L]}([D(x(t), c_i, t)]_{i=1}^{L}, [c_i]_{i=1}^{L}, t) - w \cdot C_{[L]}([D(x(t), \varnothing, t)]_{i=1}^{L}, [\varnothing]_{i=1}^{L}, t). \tag{6}$$

B.2.3. Classifier details

To evaluate the models, we trained a UNet classifier on the original dataset, which outputs the probabilities of each pixel being the center of an object. We modified the dataset with both positive and negative examples: for the conditioning from the dataset, the classifier should output a 1 probability on the corresponding pixel, and it should output 0 for a random conditioning.

We used a UNet architecture from `denoising_diffusion_pytorch` (url), with $dim = 64$ and $dim\_mults = (1, 2, 4)$. We added an additional convolution layer and used a sigmoid function on the outputs.

We used the Adam optimizer with a learning rate of 10^{-5}, with $betas = (0.9, 0.99)$ and $eps = 10^{-8}$. We trained it for 50 epochs with a batch size of 32. Figure B.1 shows an example output of the model and that of the classifier.

B.2.4. Coordinator model details

The inputs to the coordinator are the outputs of the pre-trained diffusion $[D(x, c_i, t)]_{i=1}^{L}$ and the positions $[c_i]_{i=1}^{L}$ encoded as one-hot 2D maps. The one-hot 2D maps are appended as additional channels to each diffusion output. The coordinator is trained with the denoising loss (5). During training, we sample an image x from the dataset, then sample a number of conditioning positions $L \sim \text{Uniform}[1, L_{\text{train}}]$, and then extract positions $[c_i]_{i=1}^{L}$ of L objects from the image x. Since the version of the dataset used by Du et al. (2023) had each image annotated with only one object position, we ran the classifier on each sample from the dataset and outputted the centers of the connected components of pixels where the score of the classifier is at least 0.5 as the possible conditioning inputs, as they correspond to the centers of the objects in the image.

To train the coordinator model, we used the ViT architecture with $patch\_size = 4$, $hidden\_size = 384$, $depth = 12$, $num\_heads = 6$, and $mlp\_ratio = 4.0$.

We used the Adam optimizer with a learning rate of 10^{-5}, and trained for 40 epochs with a batch size of 16. During training, we randomly sampled 2 random centers of figures as the conditionings to illustrate the effect of generalization.

B.2.5. Samplers details

To evaluate the methods, we used the Euler sampler, using the ODE equation for the reverse diffusion process with 100 timesteps, and the Heun sampler described in B with $\sigma_{\max} = 80$, $\sigma_{\min} = 10^{-4}$, and $S_{\text{churn}} = 20.0$. For the Heun sampler, we set $s_{\text{churn}} = 0$, and used $\sigma_{\min} = 10^{-4}$, $\sigma_{\max} = 80$. For σ_t, we used the Karras schedule with $\rho = 7$ and 100 timesteps. In each evaluation, we set $w = 20$ for the classifier-free guidance weight. We tested which w works best in terms of accuracy, and report results using the same weight. For sampling using the RRR method, we set $w = 4$, as used in the code released by Du et al. (2023).

B.2.6. Evaluation details

For the RRR baseline, we used formula 7:

$$RRR_{[L]}(x(t), [c_i]_{i=1}^{L}, t) = D(x(t), \varnothing; t) + w \cdot \sum_{i=1}^{L} \left(D(x(t), c_i; t) - D(x(t), \varnothing; t)\right) \tag{7}$$

For each $L \in \{1, \ldots, L_{\text{test}}\}$ we generated 256 random sets of positions $[c_i]_{i=1}^{L}$ and then generated image $x|[c_i]_{i=1}^{L}$ using the pre-trained model and the coordinator. We report the conditional generation accuracy, i.e. the fraction of the generated images satisfying the conditioning constraints. We tested the conditional constraints using the classifier $\text{CLS}(x, c)$. Given positions $[c_i]_{i=1}^{L}$, we consider the generated image x to be valid if $\text{CLS}(x, c_i) \geq 0.5,\ \forall\, 1 \leq i \leq L$. We provide an example of the classifier output on a generated sample in B.1.

For each of the 256 samples generated during evaluation, we generated positions $c_1, \ldots, c_k$ in the conditioning Y_k such that $\forall i : 0.3 \leq c_i^x, c_i^y \leq 0.7$ and $\forall i, j : ||c_i, c_j||^2 \geq 0.15$. We imposed these additional constraints on the generation of conditionings to avoid evaluating samples on very close conditioning inputs, as in such cases, a suitable sample could include objects satisfying multiple conditionings simultaneously. We used the same seed for the generation of all samples for each combination of sampler and method to make the comparison fairer. We provide some additional samples generated from our DDE model in B.2. We can see that even when some of the conditionings are not satisfied, our model still tries to satisfy as many of them as possible.

B.3. Map Image Generation Experiments

B.3.1. Dataset description

We selected 7,300 uniform samples of size 150×150 meters from a 20×20 km square, centered at (35.707° N, 139.600° E). Each sample contains a pair of a map and the corresponding satellite image and has a resolution of 600×600. We cropped a 512×512 sub-square from it and downsampled it to 128×128. Figure B.3 demonstrates a raw dataset example before cropping and downsampling.

B.3.2. Architecture details

We use decomposition functions that take small patches inside the map. Specifically, we have the following parameters: the size of the patch B, the stride s, and the number of patches in each dimension k. Then, the large map will be a square with size $B + s(k-1)$, and there will be k^2 patches, indexed by (i, j), where $0 \leq i, j < k$, such that patch (i, j) spans height coordinates in $[is, is + B)$ and width coordinates in $[js, js + B)$. We denote patches of the satellite image as X_{ij} and patches of the map conditioning as Y_{ij}. Then, we can note that due to the homogeneity of maps, the distribution $p(X_{ij}|Y_{ij})$ does not depend on (i, j), so we can utilize the same pretrained conditional model on all the patches.

Our pretrained model for patches utilized the UNet architecture from denoising_diffusion_pytorch (url). We pass the conditioning as input channels, so the UNet has 6 input channels (current image and conditioning), and 3 output channels. When training the model, we use the reparametrization of the network output from Karras et al. (2022). The UNet has 5 hidden layers with $[128, 256, 512, 1024, 1024]$ hidden units, respectively. We embed time with learned 32-dimensional

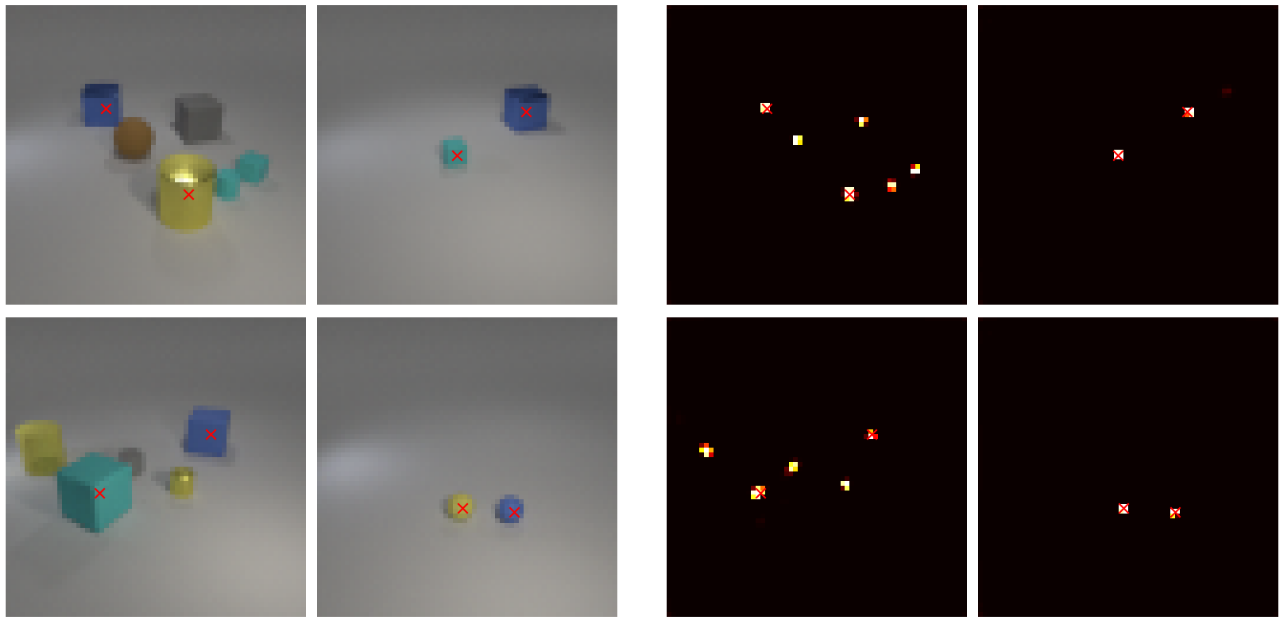

Figure B.1: A batch of generated samples from the model based on 2 conditionings and the heatmaps of the classifier output for these samples. Red crosses correspond to the conditionings.

sinusoidal embeddings. We use attention on the bottleneck layer, and on the upsampling and downsampling layers that are closest to it, with 8 attention heads and dimension 64 per head. In total, the base model has 270M parameters.

The coordinator model operates on the outputs of the base model UNets (i.e., we do not apply the reparametrization). We utilize the ViT architecture for the map coordinator. We split each patch into smaller patches used in ViT with size 2×2, and apply positional encodings that encode the position in the large image, and not the relative position in the patch. Since the $B \times B$-size patches may overlap, there could be ViT patches with the same positional encodings.

The ViT has depth 6, 6 attention heads, hidden size 384, and MLP ratio 4.0. It has 26M parameters. After all the transformer blocks, we unpatchify the tokens and extract 3 channels. Since we still have overlapping patches, we reconcile them by averaging the overlaps. Then, we apply the same reparametrization as the one that would have been applied to the outputs of the base model.

B.3.3. Conditioning

Here, we describe how ViT handles the conditional information (schematic map). We split a large $N \times N$ schematic map into 4×4 patches and pass them to ViT together with the patches of the model outputs. The ViT performs self-attention on all of the tokens; however, when performing MLP, we use different weights for tokens derived from model outputs and tokens derived from the conditioning.

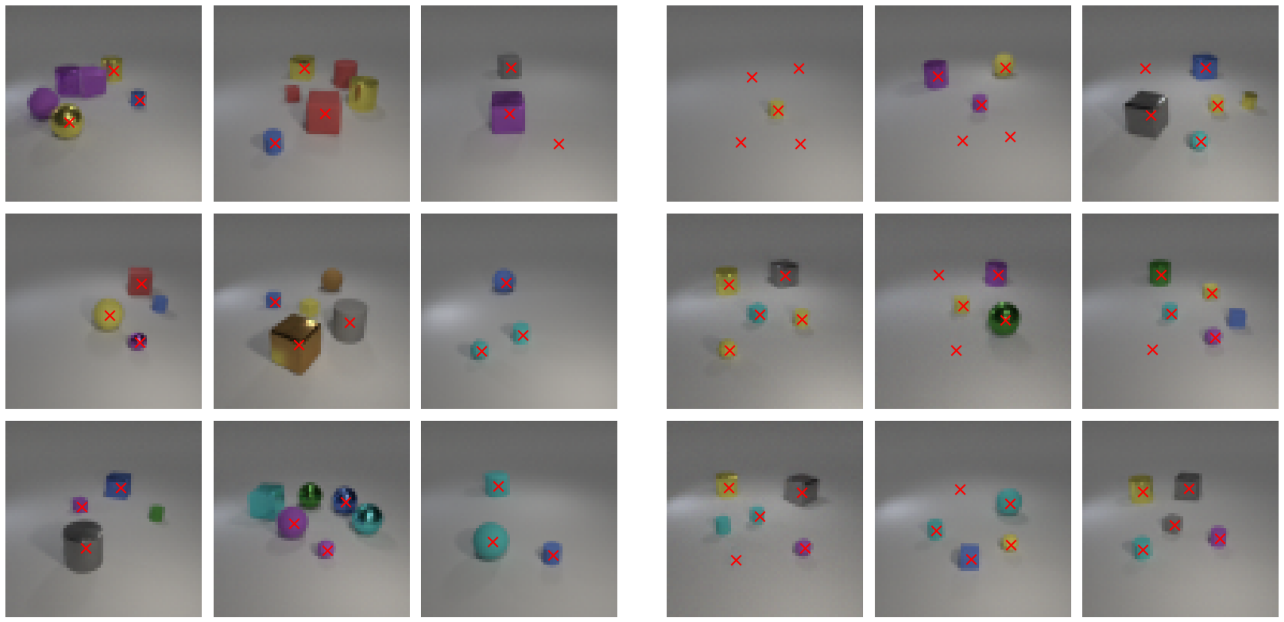

Figure B.2: Generated samples from the coordinator model for 3 and 5 different conditionings. Each red cross on the image corresponds to a conditioning.

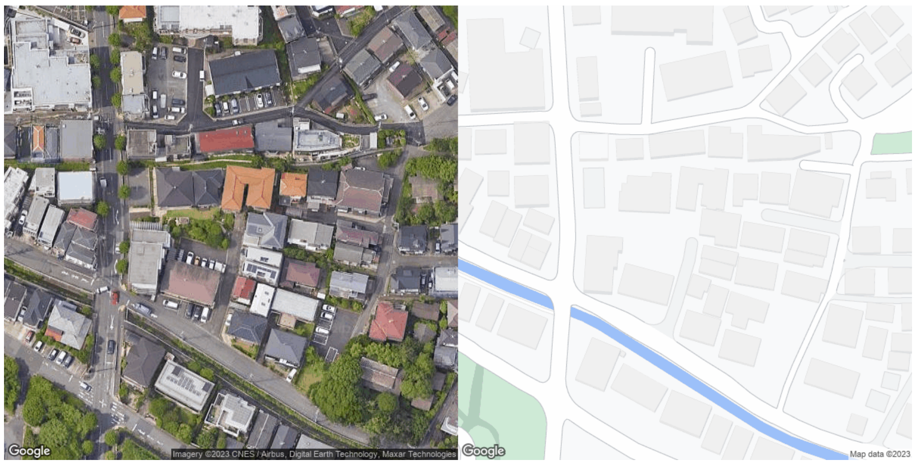
Google
Imagery ©2023 CNES / Airbus, Digital Earth Technology, Maxar Technologies
Google
Map data ©2023

Figure B.3: Example from the map dataset. Image attribution: Google Maps

B.3.4. 2D RoPE

To improve the generalization capabilities of the ViT coordinator, we use rotary positional encodings from Su et al. (2023) (RoPE). RoPE are defined for the case when tokens have 1-dimensional positions. We do not add positional encodings explicitly at the beginning of the forward pass. Instead, in each layer of the transformer when we calculate keys or queries, we multiply them by a rotation matrix $R^d_{\Theta,m}$. Let x_m be the m-th token, d be its dimension, Θ be a parameter vector, $W_{\{q,\,k\}}$ be a learnable query/key matrix, respectively, and f the function that computes the query/key of a token. Then, according to equation 14 from Su et al. (2023):

$$f_{\{q,\,k\}}(x_m, m) = R^d_{\Theta,m} W_{\{q,\,k\}} x_m$$

We generalize this method to the case when positions of tokens are two-dimensional. We do this by splitting our embeddings into two halves and applying the transform associated with the vertical coordinate on the first half, and the transform associated with the horizontal coordinate on the second half. Let x_{nm} be a $2d$-dimensional token at position (n, m). Then:

$$f_{\{q,\,k\}}(x_{nm}, n, m) = \begin{pmatrix} R^d_{\Theta,n} & 0 \\ 0 & R^d_{\Theta,m} \end{pmatrix} W_{\{q,\,k\}} x_{nm}$$

For 1D positional encodings, we have the property that the attention value between two tokens (i.e., the dot product between key and value) depends only on their coordinates and the relative position:

$$(f_k(x_n, n), f_q(x_m, m)) = g(x_n, x_m, n - m)$$

By decomposing the vectors into two halves, it is easy to show that a similar statement holds for 2D RoPE:

$$(f_k(x_{ab}, a, b), f_q(x_{cd}, c, d)) = g(x_{ab}, x_{cd}, a - c, b - d)$$

B.3.5. Training procedure

We train the coordinator for 100 epochs, with a batch size of 16. We sample the noise level σ from a log-normal distribution with a mean of -1 and a standard deviation of 1.6. We use the Adam optimizer with a learning rate of $3 \cdot 10^{-5}$.

B.3.6. Sampler details

We use the Heun sampler described in B with $\sigma_{\max} = 20$, $\sigma_{\min} = 10^{-4}$, and $S_{\text{churn}} = 20.0$. For sigmas, we use the Karras schedule with $\rho = 7$ and 150 timesteps.

B.3.7. Evaluation details

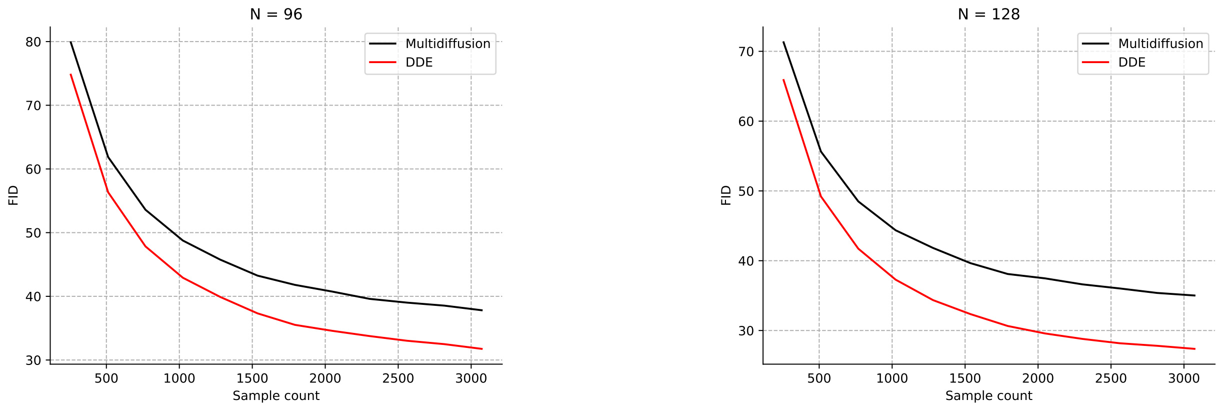

Figure B.4: Change of FID depending on the number of samples

We use the CleanFID implementation of the FID metric (Parmar et al., 2022). When reporting the FID metric for a given model, we compare the whole satellite dataset, containing 7,300 images, with 3,072 generated samples. In Figure B.4, we illustrate the dependency between the number of samples and the value of FID, showing that the comparison is consistent across the sample counts.

B.3.8. Samples

In each of the figures below, the columns indicate (from left to right) map conditioning, ground truth satellite image, and generated satellite image by DDE.

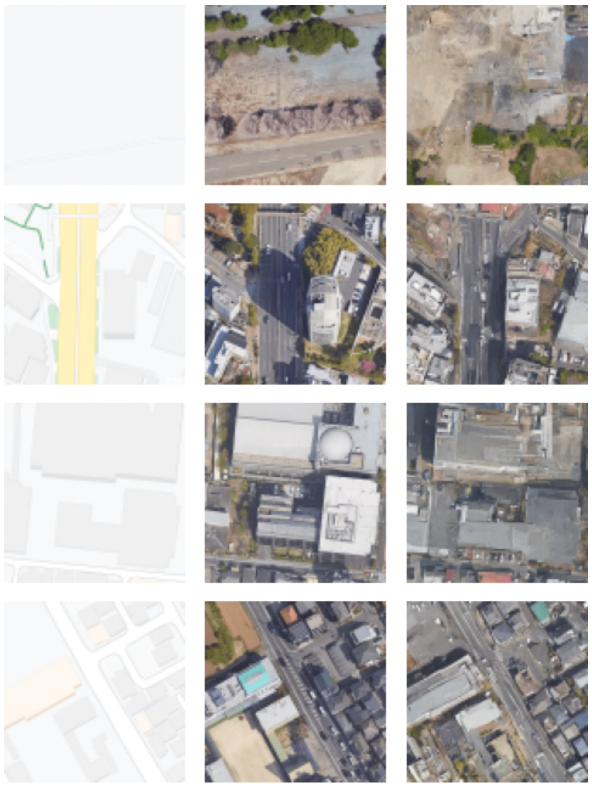
Figure B.5: DDE samples of size 96×96

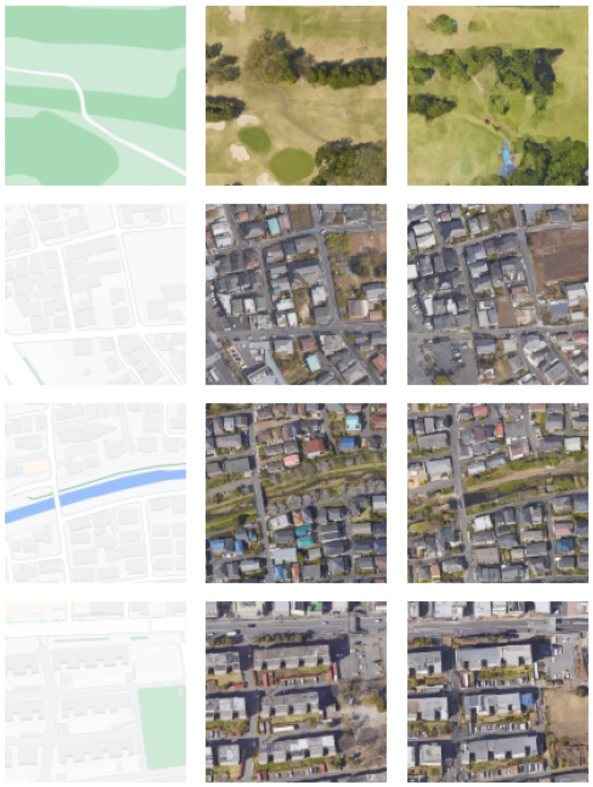
Figure B.6: DDE samples of size 128×128

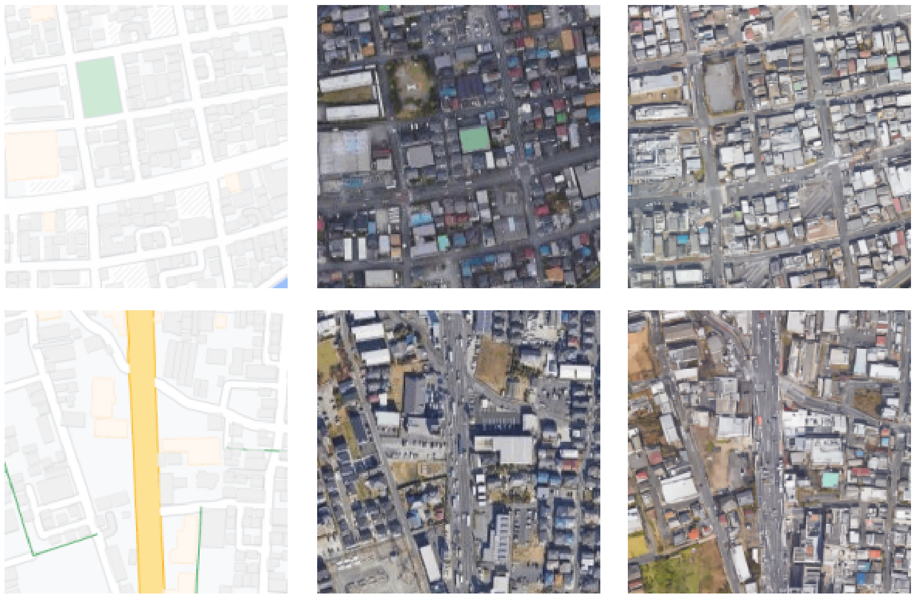
Figure B.7: DDE samples of size 256×256