Sparse Data Diffusion for Scientific Simulations in Biology and Physics

Phil Sidney Ostheimer[1], Mayank Nagda[1], Andriy Balinskyy[1], Jean Radig[2], Carl Herrmann[2], Stephan Mandt[3], Marius Kloft[1], Sophie Fellenz[1]
[1]RPTU University Kaiserslautern-Landau, [2]Heidelberg University, [3]University of California, Irvine
phil.ostheimer@rptu.de

Abstract

Sparse data is fundamental to scientific simulations in biology and physics, from single-cell gene expression to particle calorimetry, where exact zeros encode physical absence rather than weak signal. However, existing diffusion models lack the physical rigor to faithfully represent this sparsity. This work introduces Sparse Data Diffusion (SDD), a generative method that explicitly models exact zeros via Sparsity Bits, unifying efficient ML generation with physically grounded sparsity handling. Empirical validation in particle physics and single-cell biology demonstrates that SDD achieves higher fidelity than baseline methods in capturing sparse patterns critical for scientific analysis, advancing scalable and physically faithful simulation.

1 Introduction

Sparse data—i.e., data where most values are zero—is fundamental to scientific simulations in biology and physics. Sparsity arises naturally from physical principles: in calorimeter images from high-energy experiments, energy deposits occur only in localized regions corresponding to particle interactions; in scRNA data, each cell expresses only a subset of genes [23, 24, 18].

Formally, we focus on data that is continuous yet exhibits a discrete sparsity pattern:

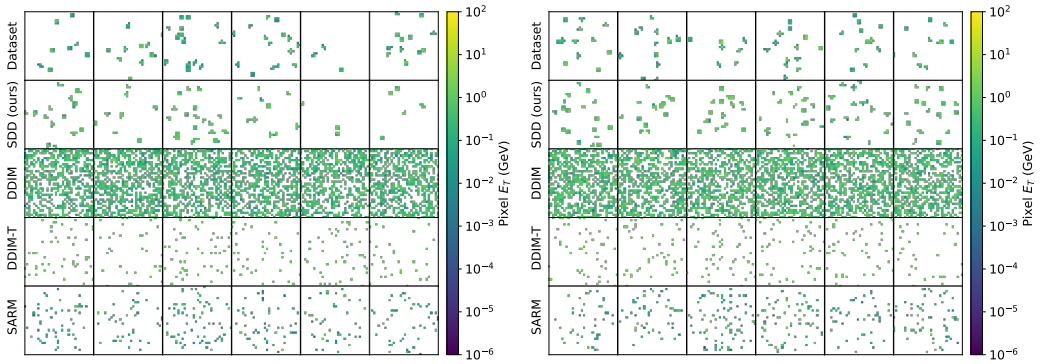
$$\mathbf{x} \in \mathbb{R}^d, \text{ where } |\mathbf{x}|_0 \ll d.$$

That is, most entries of the vector $\mathbf{x}$ with continuous values are exactly zero.

In scientific contexts, exact zeros encode physical meaning—a silent gene represents biological absence, not weak expression; an empty calorimeter cell indicates no particle interaction, not low energy. This physical interpretation makes sparsity a fundamental constraint rather than merely a statistical pattern. As Donoho [4] noted in compressed sensing, most data "can be thrown away", and Tibshirani [43] emphasized with the Lasso that sparsity reflects reality: only a small subset of variables is truly relevant.

While machine learning has accelerated scientific simulation, generative models—including Generative Adversarial Networks (GANs) [6], Variational Autoencoders (VAEs) [17], and Diffusion Models (DMs) [39, 14, 40, 1]—fundamentally mismatch physically sparse data. Adding Gaussian noise to every dimension assumes smooth, recoverable perturbations, but true zeros encode meaningful physical absence and become indistinguishable from weak signal once noised. Their isotropic noise processes, smooth denoising networks, and activations (ReLU [28], Tanh, Sigmoid [20]) bias outputs toward density, compromising physical fidelity. DDIM produces only 49% sparsity on data that is 95% sparse (Fig. 1), and thresholding (DDIM-T) to match dataset sparsity still fails to recover the clustered structures observed in real particle physics data. This underscores the need for generative models that treat sparsity as a physical constraint rather than a post-hoc correction.

European Conference on Neural Information Processing Systems (EurIPS) Workshop: SimBioChem 2025.



(a) Muon Signal

(b) Muon Background

Figure 1: Calorimeter images from the muon isolation study: signal images (left) and background images (right). Rows show samples from (1) real data, (2) SDD (ours), (3) DDIM, (4) DDIM with post-hoc thresholding to match dataset sparsity (DDIM-T), and (5) the domain-specific SARM baseline. Pixel intensity (GeV) visualizes energy deposition per cell (white=zero). SDD uniquely recovers the distinct, sparse, and clustered patterns characteristic of real data, whereas DDIM completely misses the sparsity, and DDIM-T and SARM show unrealistic isolated energy deposits.

To bridge this gap, we introduce Sparse Data Diffusion (SDD), a physically-grounded generative framework that explicitly models sparsity as a fundamental constraint. At its core, SDD uses *Sparsity Bits* (SBs)—discrete latent variables that indicate, for each output dimension, whether it should be active or zero based on the underlying physical structure. Dense values are modeled using a continuous state-space diffusion process, preserving scalability while maintaining physical fidelity. During sampling, SBs enforce exact zeros, ensuring generated data respects the physical constraints of scientific data. We validate SDD primarily on challenging scientific datasets from particle physics and single-cell biology, with additional evaluation on computer vision benchmarks to demonstrate generality. In summary, we make the following contributions:

- We introduce SDD, a physically-grounded DM that bridges scalable ML with physical accuracy by explicitly modeling sparsity as a physical constraint in DMs.
- We propose Sparsity Bits—discrete latent variables that encode physical structure—and integrate them with continuous DMs for dense values, enabling joint modeling.
- We validate SDD on scientific datasets from physics and biology, achieving substantial improvements over standard DMs and domain-specific baselines while maintaining physical fidelity, with additional demonstrations on computer vision tasks.

2 Related work

As no broadly applicable generative model for sparse data exists, our section on related work summarizes existing domain-specific works on sparse data generation, related concepts in diffusion models, and enforcing sparsity in neural model parameters (and not the data itself).

Sparse data generation Past work on sparse, *continuous* data generation focuses on simple architectures and specific applications, such as calorimeter sensor data in physics, which are difficult to compare to general-purpose generative models as they incorporate lots of domain-specific knowledge. They decouple the sparsity information for generating data by introducing decoupled generative models where the sparsity is introduced as a learnable Dirac delta mass at zero [23, 24]. Other works focusing on sparse, *discrete* data generation use Poisson or negative binomial distributions to model sparse count data with bursts [47, 37]. There are also deep variants [5, 9, 38]. However, as already indicated, they do not apply to our setting, which has continuous sparse data that do not exhibit the traits of a Poisson or negative binomial distribution.

Diffusion models Diffusion models can be broadly categorized into continuous state-space models [39, 14, 40, 41, 30] for continuous data and discrete state-space models [1, 8] for discrete data.

While discrete models can represent sparse discrete data exactly, they are not applicable to our sparse continuous setting. Some continuous models, such as Bit Diffusion [2], perform strongly on discrete data, and others like Unigs [32] and DefectSpectrum [46] handle mixed variable types by first mapping discrete variables into continuous embeddings. Recent work has also explored diffusion under unconventional data characteristics, including heavy-tailed distributions [31]. However, these approaches do not jointly diffuse continuous and discrete variables, whereas our method enables their simultaneous diffusion within a unified framework.

Enforcing sparsity Sparsity is not just an inherent trait in datasets that can be, e.g., exploited to store data efficiently. Sparsity can also help in the model architecture. Previous work [12, 44] shows that neural networks are greatly overparameterized and multiple methods [27, 22, 42, 21] inter alia have been introduced to mitigate this overparameterization by sparsifying the underlying neural network weights. Others [15] leverage this property to efficiently train low-rank decomposition matrices, which are added as sparse weight updates to the existing model weights. However, these methods focus on enforcing sparsity in the model weights and are therefore not applicable to our setting, where we enforce sparsity in the model output.

3 Method

This section presents Sparse Data Diffusion (SDD), a novel framework designed for the generation of sparse data. We begin by introducing the underlying statistical model, followed by an in-depth explanation of the forward and backward diffusion processes, and the overall training and sampling procedure.

3.1 Statistical model

The following statistical model forms the basis for SDD: Let $\mathbf{x_0} \sim p$ with $\mathbb{E}[\mathbf{x_0}] < \infty$ and $\mathbf{x_0} \in \mathbb{R}^d$, where p is unknown and arbitrary. We infer the sparsity for each dimension by applying the indicator function element-wise to $\mathbf{x_0}$:

$$\bar{\mathbf{x}}_\mathbf{0} = 2 * \mathbb{1}_{x \neq 0}(\mathbf{x_0}) - 1 \tag{1}$$

As a result, $\bar{\mathbf{x}}_\mathbf{0} \in \{-1, 1\}^d$ is a binary vector that encodes, for each element in $\mathbf{x_0}$, whether it is zero or represents a dense value. Therefore, we refer to each element in $\bar{\mathbf{x}}_\mathbf{0}$ as a *Sparsity Bit (SB)*. We obtain the extended input $\hat{\mathbf{x}}_0 \in \mathbb{R}^{2d}$, where $\hat{\mathbf{x}}_0 \sim \hat{p}$, by concatenating $\mathbf{x_0} \in \mathbb{R}^d$ and $\bar{\mathbf{x}}$:

$$\hat{\mathbf{x}}_\mathbf{0} = \begin{bmatrix} \mathbf{x_0} \\ \bar{\mathbf{x}}_\mathbf{0} \end{bmatrix} \tag{2}$$

Unlike prior methods, our forward–backward process diffuses continuous and discrete variables jointly. Previous work operated either only on continuous variables [39, 14, 40, 41] or on discrete variables [2].

Forward diffusion The forward diffusion process follows previous work [39, 14, 40, 41]. It consists of a predefined series of transitions from the input space $\hat{\mathbf{x}}_\mathbf{0} \in \mathbb{R}^{2d}$ to pure noise $\boldsymbol{\epsilon} \in \mathbb{R}^{2d}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$. The transition from $\hat{\mathbf{x}}_\mathbf{0}$ to $\hat{\mathbf{x}}_\mathbf{t}$ is defined as

$$\hat{\mathbf{x}}_\mathbf{t} = \sqrt{\alpha(t)}\hat{\mathbf{x}}_\mathbf{0} + \sqrt{1 - \alpha(t)}\boldsymbol{\epsilon}, \tag{3}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$, $t \sim \mathcal{U}(0, T)$ is a continuous time variable, and α is the noise schedule, a monotonically decreasing function from 1 to 0. In the limit, we obtain:

$$\lim_{T \to \infty} \hat{\mathbf{x}}_\mathbf{T} = \boldsymbol{\epsilon} \sim \mathcal{N}(0, I), \tag{4}$$

Backward diffusion The backward diffusion process consists of steps that reverse the forward diffusion process: $\hat{\mathbf{x}}_\mathbf{T} \to \hat{\mathbf{x}}_{\mathbf{T}-\boldsymbol{\Delta}} \to ... \to \hat{\mathbf{x}}_\mathbf{0}$. These steps follow a normal distribution:

$$\hat{\mathbf{x}}_{\mathbf{t}-\mathbf{1}} | \hat{\mathbf{x}}_\mathbf{t} \sim \mathcal{N}(\mu_t(\hat{\mathbf{x}}_\mathbf{t}, \hat{\mathbf{x}}_\mathbf{0}), \sigma_t^2) \tag{5}$$

where $\hat{\mathbf{x}}_\mathbf{0}$ is the denoised input. Since $\hat{\mathbf{x}}_\mathbf{0}$ is not given in the backward diffusion process, we train a denoising neural network f_θ to predict $\hat{\mathbf{x}}_\mathbf{0}$. We describe the training process in the following Section 3.2 and illustrate our model in Figure 2.

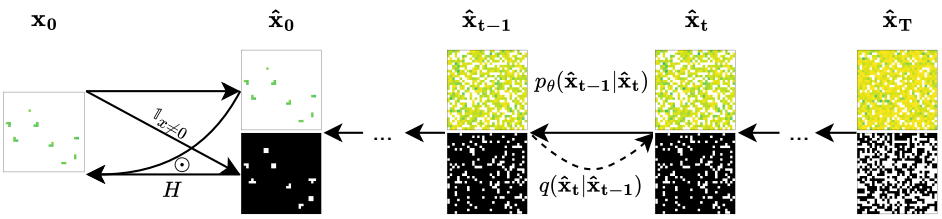


Figure 2: Shown is an illustration of our method SDD. Compared to other diffusion models, we expand the continuous input by discrete Sparsity Bits for forward diffusion and use them in backward diffusion and sampling to enforce sparsity in the data.

3.2 Training

In order to perform the steps $\hat{\mathbf{x}}_\mathbf{T} \rightarrow \hat{\mathbf{x}}_{\mathbf{T-\Delta}} \rightarrow ... \rightarrow \hat{\mathbf{x}}_\mathbf{0}$ of the backward diffusion process, we train a denoising network f_θ to predict $\hat{\mathbf{x}}_\mathbf{0}$, which is part of the distribution $p(\hat{\mathbf{x}}_{\mathbf{t-1}}|\hat{\mathbf{x}}_\mathbf{t})$. The predicted value of $\hat{\mathbf{x}}_\mathbf{0}$ is approximated by $\tilde{\mathbf{x}}_\mathbf{0} = f_\theta(\hat{\mathbf{x}}_\mathbf{t}, t)$ (alternative formulations predict the noise $\boldsymbol{\epsilon}$).

We use self-conditioning [2] to incorporate the previously computed $\tilde{\mathbf{x}}_\mathbf{0}$ (or $\tilde{\boldsymbol{\epsilon}}$) from time step $t+1$ to compute the next $\tilde{\mathbf{x}}_\mathbf{0}$ (or $\tilde{\epsilon}$). The training is realized using an l_2 regression loss on the continuous inputs and a cross entropy loss (CE on labels in $\{-1, 1\}$) on SBs:

$$\begin{aligned}\mathcal{L}(\theta) = \mathbb{E}\Big[&\left\| f_{\theta,0:d-1}(\hat{\mathbf{x}}_\mathbf{t}, t, \tilde{\mathbf{x}}_\mathbf{0}) - \hat{\mathbf{x}}_{\mathbf{0,0:d-1}} \right\|^2 \\ &+ \text{CE}\left(f_{\theta,d:2d-1}(\hat{\mathbf{x}}_\mathbf{t}, t, \tilde{\mathbf{x}}_\mathbf{0}), \hat{\mathbf{x}}_{\mathbf{0,d:2d-1}} \right) \Big],\end{aligned} \tag{6}$$

where $\hat{\mathbf{x}}_\mathbf{0} \sim \hat{p}, t \sim \mathcal{U}(0, T), \boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ and $\mathbf{x_t} = \sqrt{\alpha(t)}\hat{\mathbf{x}}_\mathbf{0} + \sqrt{1-\alpha(t)}\boldsymbol{\epsilon}$. We summarize the SDD training process in Algorithm 1 in Appendix A.

3.3 Sampling

To draw samples, we use the same state transitions as described in the backward diffusion process: $\hat{\mathbf{x}}_\mathbf{T} \rightarrow \hat{\mathbf{x}}_{\mathbf{T-\Delta}} \rightarrow ... \rightarrow \hat{\mathbf{x}}_\mathbf{0}$, following $p(\hat{\mathbf{x}}_{\mathbf{t-1}}|\hat{\mathbf{x}}_\mathbf{t})$ using $f_\theta(\hat{\mathbf{x}}_\mathbf{t}, t, \tilde{\mathbf{x}}_\mathbf{0})$. There are multiple ways to perform these steps. Here, we focus on Denoising Diffusion Probabilistic Models (DDPM) [14] and Denoising Diffusion Implicit Models (DDIM) [40].

We perform one final step: $\hat{\mathbf{x}}_\mathbf{0} \rightarrow \mathbf{x_0}$. Since the first d dimensions in $\hat{\mathbf{x}}_\mathbf{0}$ contain the dense and second d dimensions in $\hat{\mathbf{x}}_\mathbf{0}$ contain the SBs, we quantize the SB output dimensions $\hat{\mathbf{x}}_{\mathbf{0;d:(2d-1)}}$ using the heaviside step function H applied element-wise to extract the SBs. Afterward, we apply an element-wise product to get the sparsified output:

$$\mathbf{x_0} = \hat{\mathbf{x}}_{\mathbf{0;0:(d-1)}} \odot H(\hat{\mathbf{x}}_{\mathbf{0;d:(2d-1)}}) \tag{7}$$

We summarize the sampling algorithm in Algorithm 2 in Appendix A.

4 Experiments

To evaluate SDD for sparse data generation, we first assess the overall sparsity of generated data, then report results on challenging scientific applications in physics and biology, and conclude with an analysis of sparse image generation.

4.1 Experimental setup

Baselines We compare SDD to two well-known diffusion models—DDPM [14] and DDIM [40]—which are designed for dense data generation. We provide architectural and training details in Appendix B. To evaluate the performance on sparse outputs, we create thresholded variants DDPM-T and DDIM-T that apply iteratively increasing thresholds (linearly spaced between zero

and maximum output value) to match training data sparsity. This provides a pragmatic baseline for adapting diffusion models to sparse domains without explicit sparsity modeling during training.

For particle physics, we benchmark against SARM D+C [24], a domain-specific model that autoregressively samples calorimeter data using spiral patterns informed by inner and outer regions. For scRNA, we include scDiffusion [25], which uses an autoencoder pretrained on a large cell corpus but lacks explicit sparsity enforcement.

Datasets In physics, we use calorimeter images from a muon isolation study [23], represented as 32×32 pixel grids with 95% sparsity. Pixel intensity represents the transverse momentum (P_T) deposited in each cell. The dataset contains 33,331 signal images of isolated muons and 30,783 background images of muons with jets. For biology, we use two scRNA datasets: Tabula Muris [36] with 57K cells (90% sparsity, 98% when filtered to 1000 highly variable genes) and Human Lung Pulmonary Fibrosis [10] with 114K cells (91% sparsity, 96% filtered). For vision, we use MNIST [19] (60K images, 81% sparsity) and Fashion-MNIST [45] (60K images, 50% sparsity), both 28×28 grayscale. All datasets are linearly scaled to $[-1, 1]$ following Ho et al. [14], with inverse transformation applied after inference.

Evaluation We evaluate sparsity distribution matching and use domain-specific metrics. For physics, we compute the normalized Wasserstein distance W_P between distributions of transverse momentum P_T and invariant mass for the dataset and 50,000 generated images [23, 24]. For scRNA, following [25], we use Spearman Correlation (SCC), Pearson Correlation (PCC), Maximum Mean Discrepancy (MMD) [7], and Local Inverse Simpson's index (LISI) [11], comparing real data to 10,000 generated cells. For images, we measure sparsity after discretizing logit values from $[-1, 1]$ to $[0, 255]$ and compute Fréchet Inception Distance (FID) [13] between 50,000 generated images and training data. Note that FID's validity is limited for sparse data as the Inception network was trained on dense ImageNet images [3].

4.2 Evaluating the recovery of ground truth sparsity patterns

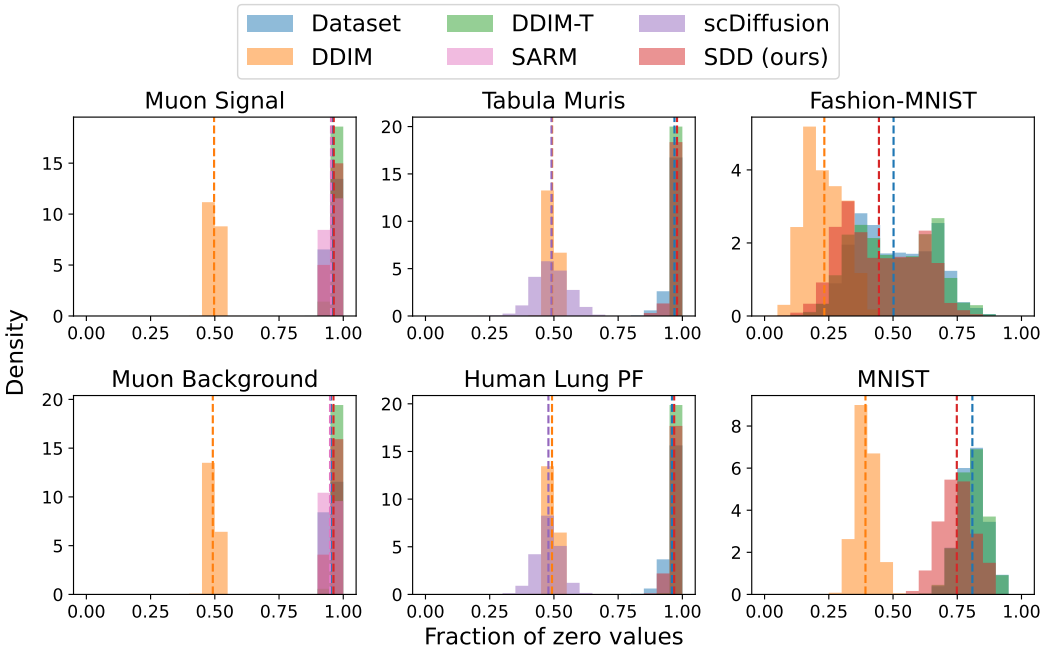


Figure 3: Sparsity distribution of real and generated data. Histograms (20 bins) and average sparsity levels (dashed lines) compare real data to samples from DDIM, DDIM-T, SARM, scDiffusion, and SDD. DDIM and scDiffusion underestimate sparsity; DDIM-T matches the average but lacks diversity and overshoots at high sparsity. SARM also underestimates sparsity, while SDD accurately matches both average value and distribution.

How well do models match ground-truth sparsity? We analyze the sparsity distribution of generated samples in Figure 3 for DDIM, with DDPM results in Appendix D largely consistent with DDIM. Across all datasets, DDIM and the domain-specific scDiffusion tend to underestimate sparsity, producing outputs denser than the real data. While DDIM-T matches average sparsity by design, it often overshoots in the upper range and truncates the lower end of the distribution, especially for sparse image datasets. The particle physics-specific SARM also slightly underestimates sparsity despite domain knowledge. In contrast, SDD stands out by accurately capturing both the average sparsity and the overall sparsity distribution, achieving realistic and faithful sparsity patterns without relying on domain-specific knowledge or post-hoc processing.

In general, diffusion models struggle to generate sparse data. If the standard post-processing step—clipping values to the range $[-1, 1]$ followed by rescaling—is omitted, the resulting samples exhibit near-zero sparsity across all datasets. As this clipping step is standard practice in diffusion-based models, we retain it and focus our subsequent analyses on clipped outputs.

Consequences of sparsity mismatch in scientific and vision tasks In *particle physics*, identifying particle sources from detector signatures is crucial [23, 24]. DDIM and DDIM-T fail to reproduce data sparsity (Figure 1), producing overly dense outputs that inflate particle detections and lead to incorrect interpretations. In *scRNA data*, sparsity reflects biologically meaningful dropout events. DDIM captures only half the true sparsity, misrepresenting dropout patterns critical for clustering and imputation [33]. While DDIM-T matches average sparsity, it underperforms quantitatively (Section 4.4), as does the domain-specific scDiffusion. For *sparse images*, sparsity mismatch trivializes fake image detection, as generated images lacking authentic sparsity patterns become easily identifiable, compromising robustness in security applications.

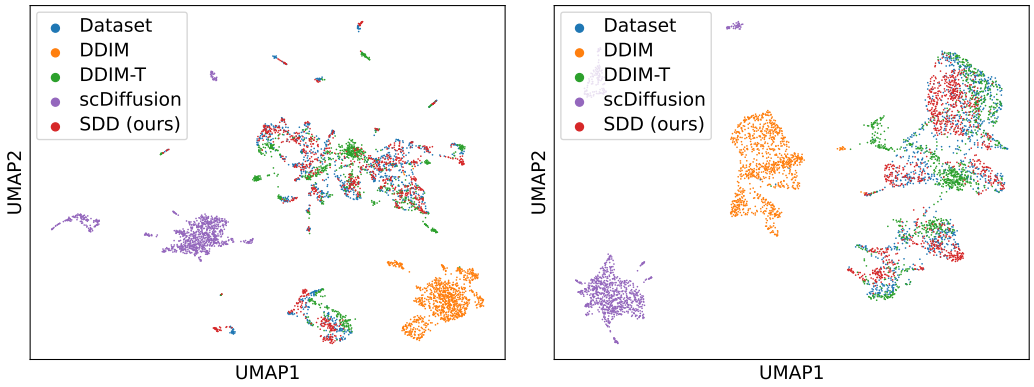
Sharpness of generated SBs As SBs are central to SDD and our model is the first one to simultaneously diffuse the discrete SBs and continuous (dense) inputs, we further examine the distribution of the logits for the SBs in Figure 7 in Appendix D. As can be seen, SBs are sharply distributed towards -1 and 1, indicating the model's confidence in predicting these pixel values as zero or non-zero. This indicates that discrete variables can be reliably diffused alongside continuous variables. The last bin with an upper bound of 1.0 shows almost precisely the pixel sparsity of the respective dataset.

4.3 Physics: calorimeter image generation

Table 1: Shown are the results for sparse calorimeter image generation. SDD performs better in preserving the key structural properties of the data—sum of momenta transverse to the beam P_T and invariant mass—and achieves results on par with the domain-specific SARM baseline, demonstrating its competitive performance without relying on domain-specific design.

	SIGNAL		BACKGROUND	
MODEL	$\downarrow P_T$	$\downarrow$ MASS	$\downarrow P_T$	$\downarrow$ MASS
DDPM	219.30	79.19	227.00	81.43
DDIM	251.79	89.96	259.43	92.20
DDPM-T	23.21	11.30	27.83	12.52
DDIM-T	25.36	12.16	31.28	13.81
SARM D+C	27.96	7.29	**12.50**	**5.39**
SDD DDPM (OURS)	15.00	6.50	13.92	5.44
SDD DDIM (OURS)	**14.67**	**6.43**	13.01	5.41

Quantitative evaluation The results are summarized in Table 1. The Wasserstein Distance W_P between the distributions of the sum of momenta transverse to the beam (P_T) and the invariant mass for both signal and background images is substantially larger for DDPM, DDIM, and their thresholded variants (DDPM-T, DDIM-T) compared to SDD DDPM and SDD DDIM, indicating that sparsity-aware SDD generates outputs more closely aligned with the real data. SDD performs exceptionally well on signal images, where the W_P for P_T is worse for SARM. SARM achieves slightly better



(a) Tabula Muris (b) Human Lung Pulmonary Fibrosis

Figure 5: Shown are two-dimensional UMAPs of Tabula Muris and Human Lung Pulmonary Fibrosis, and the respective DDIM, DDIM-T, and SDD generated samples. DDIM, DDIM-T, and the domain-specific scDiffusion show little to no overlap with the respective dataset while SDD shows significant overlap, demonstrating that SDD more accurately captures the underlying structure and diversity of the real data distributions.

results for background images, but SDD remains competitive overall. These findings suggest that SDD's sparsity-aware modeling effectively captures key domain-specific features without relying on extensive prior knowledge, resulting in improved fidelity and realism of generated calorimeter data.

Qualitative evaluation Figure 1 shows Muon Signal and Background calorimeter images generated by DDIM (similar DDPM results in Appendix E). DDIM fails to capture clustered energy deposits, while thresholded DDIM-T and SARM produce many isolated single-pixel deposits lacking the clustered deposits of real data. In contrast, SDD accurately reproduces these clusters, consistent with physical interactions. Pixel-wise averages in Figure 4 confirm this: DDIM and DDIM-T show almost uniform signals across the image, and SARM underestimates the pixel intensity. For signal images, SDD models the near-uniform energy spread of the calorimeter; for backgrounds, it captures energy near muons from jets. Though DDIM-T and DDPM-T approximate overall sparsity (Section 4.2), they fail to preserve realistic sparsity patterns and spatial coherence. Similarly, despite SARM's domain knowledge, it also shows the circular shape of energy deposits but misses the intensity.

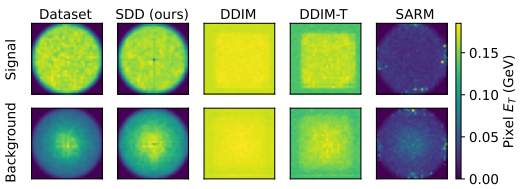


Figure 4: Shown are the average calorimeter images for Muon Signal and Muon Background. DDIM and DDIM-T fail to generate realistic data, while SDD succeeds. Linear scale to reveal the signal and background differences.

4.4 Biology: scRNA data generation

Quantitative evaluation Table 2 shows scRNA generation results. Across both Tabula Muris and Human Lung Pulmonary Fibrosis datasets, SDD outperforms standard DDPM/DDIM and their thresholded variants (DDPM-T/DDIM-T) by orders of magnitude. SCC and PCC scores show SDD better preserves gene expression ordering and linearity. MMD distances indicate SDD's distributions are significantly closer to real data, and LISI scores reflect improved mixing between real and generated samples. Notably, SDD outperforms scDiffusion—despite its pretrained autoencoder on a large cell corpus—demonstrating that explicit sparsity modeling captures biological data distributions more effectively than domain-specific pretraining alone.

Qualitative evaluation In our qualitative evaluation for scRNA data, we provide UMAPs [26] for Tabula Muris and Human Lung Pulmonary Fibrosis in Figure 5 for DDIM, with further analysis

Table 2: The results for sparse data generation on scRNA data are shown. SDD outperforms DDPM and DDIM, the thresholded DDPM-T and DDIM-T, and the task-specific scDiffusion in all considered metrics.

	Tabula Muris				Human Lung PF			
Model	↑ SCC	↑ PCC	↓ MMD	↑ LISI	↑ SCC	↑ PCC	↓ MMD	↑ LISI
DDPM	0.49	0.72	3.58	0.00	0.31	0.85	3.35	0.00
DDIM	0.50	0.74	3.62	0.00	0.30	0.87	3.34	0.00
DDPM-T	0.53	0.68	0.35	0.01	0.31	0.82	0.69	0.00
DDIM-T	0.56	0.73	0.34	0.01	0.28	0.85	0.65	0.00
scDiffusion	0.71	0.71	1.54	0.00	0.77	0.79	1.02	0.00
SDD DDPM (ours)	0.96	0.95	**0.12**	0.21	**0.95**	**0.99**	**0.18**	**0.08**
SDD DDIM (ours)	**0.97**	**0.97**	**0.12**	**0.23**	**0.95**	**0.99**	**0.18**	**0.08**

showing similar results for DDPM in Appendix F. Both subfigures in Figure 5 show almost no overlap between real data and DDIM-generated data. For DDIM-T, the generated cell cluster shifts closer to the real cells but exhibits little diversity. The domain-specific scDiffusion model also demonstrates little to no overlap with real cells, indicating it fails to capture the true biological structure despite its pretraining on a large cell corpus. In contrast, SDD shows significant overlap between real cells and those sampled from SDD, further underlining the high quality and biological fidelity of the generated cells.

4.5 Vision: sparse image generation

Quantitative evaluation For sparse image generation, we summarize our results in Table 4 in Appendix G. We observe similar FID scores for all settings considered. However, the validity of FID for evaluating sparse images is questionable, as the standard Inception network used to compute FID was trained only on dense images from ImageNet [3], which may limit its effectiveness for sparse data. Therefore, we report these values only for completeness.

Qualitative evaluation The quality of generated images on Fashion-MNIST and MNIST (see Figures 11 and 12 in Appendix G) is largely indistinguishable across DDIM, DDPM, their thresholded variants (DDIM-T, DDPM-T), and SDD using DDPM and DDIM. Visually, all models produce plausible samples. However, a closer inspection of the sparsity patterns reveals meaningful differences. In particular, SDD more faithfully reproduces the sparsity structure of the real data. For Fashion-MNIST, thresholded models like DDIM-T and DDPM-T tend to suppress fine-grained noisy regions near the edges. Similarly, in MNIST, the digits generated by DDIM-T and DDPM-T are generally narrower than those in real data. These subtle differences highlight that, while thresholding may yield visually appealing outputs, it often fails to preserve key structural properties of the data, particularly in challenging regions where sparsity transitions are critical. SDD, by contrast, maintains these characteristics more reliably, capturing both visual fidelity and underlying data distribution properties by explicitly modeling the sparsity.

5 Conclusion

We introduce SDD, a physically-grounded method for generating sparse scientific data using DMs. By representing sparsity through Sparsity Bits—discrete variables diffused alongside continuous values—SDD explicitly encodes the physical structure of sparse data. We demonstrate high fidelity across physics and biology applications, with additional validation on computer vision tasks. SDD achieves superior sample quality compared to standard DMs and their thresholded variants, matching the domain-specific SARM in particle physics and surpassing the biologically pre-trained scDiffusion for scRNA data. This demonstrates that explicit sparsity modeling is more effective than approaches relying solely on domain knowledge or pretraining. The underlying concept of Sparsity Bits can be integrated into other generative models, including GANs [6], VAEs [17], and Normalizing Flows [34], opening new research directions for physically-grounded generative modeling in scientific simulation.

Acknowledgments and Disclosure of Funding

This work was conducted within the initiative AI-Care by the Carl-Zeiss Stiftung. Part of this work was conducted within the DFG Research Unit FOR 5359 on Deep Learning on Sparse Chemical Process Data (BU 4042/2-1, KL 2698/6-1, and KL 2698/7-1). MK and SF further acknowledge support by the DFG TRR 375 (ID 511263698), the DFG SPP 2298 (KL 2698/5-2), and the DFG SPP 2331 (FE 2282/1-2, FE 2282/6-1, and KL 2698/11-1). Additional support was provided by the BMBF award 01|S2407A.

References

[1] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 17981–17993, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/958c530554f78bcd8e97125b70e6973d-Abstract.html.

[2] Ting Chen, Ruixiang Zhang, and Geoffrey E. Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=3itjR9QxFw.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[4] David L. Donoho. Compressed sensing. *IEEE Trans. Inf. Theory*, 52(4):1289–1306, 2006. doi: 10.1109/TIT.2006.871582. URL https://doi.org/10.1109/TIT.2006.871582.

[5] Chengyue Gong et al. Deep dynamic Poisson factorization model. *Advances in Neural Information Processing Systems*, 30, 2017.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[7] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[8] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 10686–10696. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01043. URL https://doi.org/10.1109/CVPR52688.2022.01043.

[9] Dandan Guo, Bo Chen, Hao Zhang, and Mingyuan Zhou. Deep poisson gamma dynamical systems. *Advances in Neural Information Processing Systems*, 31, 2018.

[10] Arun C Habermann, Austin J Gutierrez, Linh T Bui, Stephanie L Yahn, Nichelle I Winters, Carla L Calvi, Lance Peter, Mei-I Chung, Chase J Taylor, Christopher Jetter, et al. Single-cell rna sequencing reveals profibrotic roles of distinct epithelial and mesenchymal lineages in pulmonary fibrosis. *Science advances*, 6(28):eaba1972, 2020.

[11] Laleh Haghverdi, Aaron TL Lun, Michael D Morgan, and John C Marioni. Batch effects in single-cell rna-sequencing data are corrected by matching mutual nearest neighbors. *Nature biotechnology*, 36(5):421–427, 2018.

[12] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding. *arXiv: Computer Vision and Pattern Recognition*, 2015. URL https://api.semanticscholar.org/CorpusID:2134321.

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6626–6637, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *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.

[15] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[17] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes, 2014. URL https://arxiv.org/abs/1312.6114.

[18] David Lähnemann, Johannes Köster, Ewa Szczurek, Davis J McCarthy, Stephanie C Hicks, Mark D Robinson, Catalina A Vallejos, Kieran R Campbell, Niko Beerenwinkel, Ahmed Mahfouz, et al. Eleven grand challenges in single-cell data science. *Genome biology*, 21:1–35, 2020.

[19] Yann LeCun. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

[20] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade - Second Edition*, volume 7700 of *Lecture Notes in Computer Science*, pages 9–48. Springer, 2012. doi: 10.1007/978-3-642-35289-8_3. URL https://doi.org/10.1007/978-3-642-35289-8_3.

[21] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. SNIP: Single-shot network pruning based on connection sensitivity. *ArXiv*, abs/1810.02340, 2018. URL https://api.semanticscholar.org/CorpusID:52920837.

[22] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1Y8hhg0b.

[23] Yadong Lu, Julian Collado, Kevin Bauer, Daniel Whiteson, and Pierre Baldi. Sparse image generation with decoupled generative models. In *Neural Information Processing Systems, Machine Learning and the Physical Sciences Workshop*, 2019.

[24] Yadong Lu, Julian Collado, Daniel Whiteson, and Pierre Baldi. Sparse autoregressive models for scalable generation of sparse images in particle physics. *Physical Review D*, 103(3):036012, 2021.

[25] Erpai Luo, Minsheng Hao, Lei Wei, and Xuegong Zhang. scdiffusion: conditional generation of high-quality single-cell data using diffusion model. *Bioinformatics*, 40(9):btae518, 2024.

[26] Leland McInnes and John Healy. UMAP: uniform manifold approximation and projection for dimension reduction. *CoRR*, abs/1802.03426, 2018. URL http://arxiv.org/abs/1802.03426.

[27] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2498–2507. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/molchanov17a.html.

[28] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814. Omnipress, 2010. URL https://icml.cc/Conferences/2010/papers/432.pdf.

[29] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: towards photorealistic image generation and editing with text-guided diffusion models. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 16784–16804. PMLR, 2022. URL https://proceedings.mlr.press/v162/nichol22a.html.

[30] K. Pandey and S. Mandt. A complete recipe for diffusion generative models. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 4261–4272, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/papers/Pandey_A_Complete_Recipe_for_Diffusion_Generative_Models_ICCV_2023_paper.pdf. (Oral presentation).

[31] K. Pandey, J. Pathak, Y. Xu, S. Mandt, M. Pritchard, A. Vahdat, and M. Mardani. Heavy-tailed diffusion models. In *International Conference on Learning Representations (ICLR)*, page (nine pages), 2025. URL https://openreview.net/forum?id=tozlOEN4qp.

[32] Lu Qi, Lehan Yang, Weidong Guo, Yu Xu, Bo Du, Varun Jampani, and Ming-Hsuan Yang. Unigs: Unified representation for image generation and segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 6305–6315. IEEE, 2024. doi: 10.1109/CVPR52733.2024.00603. URL https://doi.org/10.1109/CVPR52733.2024.00603.

[33] Peng Qiu. Embracing the dropouts in single-cell rna-seq analysis. *Nature communications*, 11 (1):1169, 2020.

[34] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis R. Bach and David M. Blei, editors, *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*, pages 1530–1538. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/rezende15.html.

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28. URL https://doi.org/10.1007/978-3-319-24574-4_28.

[36] Nicholas Schaum, Jim Karkanias, Norma F Neff, Andrew P May, Stephen R Quake, Tony Wyss-Coray, Spyros Darmanis, Joshua Batson, Olga Botvinnik, Michelle B Chen, et al. Single-cell transcriptomics of 20 mouse organs creates a tabula muris: The tabula muris consortium. *Nature*, 562(7727):367, 2018.

[37] Aaron Schein, Hanna Wallach, and Mingyuan Zhou. Poisson-gamma dynamical systems. *Advances in Neural Information Processing Systems*, 29, 2016.

[38] Aaron Schein, Scott W. Linderman, Mingyuan Zhou, David M. Blei, and Hanna M. Wallach. Poisson-randomized gamma dynamical systems. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 781–792, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1543843a4723ed2ab08e18053ae6dc5b-Abstract.html.

[39] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis R. Bach and David M. Blei, editors, *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*, pages 2256–2265. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/sohl-dickstein15.html.

[40] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *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/92c3b916311a5517d9290576e3ea37ad-Abstract.html.

[41] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

[42] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. 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=PxoFut3dWW.

[43] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

[44] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HJGwcKclx.

[45] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[46] Shuai Yang, Zhifei Chen, Pengguang Chen, Xi Fang, Yixun Liang, Shu Liu, and Yingcong Chen. Defect spectrum: A granular look of large-scale defect datasets with rich semantics. In Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part VII*, volume 15065 of *Lecture Notes in Computer Science*, pages 187–203. Springer, 2024. doi: 10.1007/978-3-031-72667-5_11. URL https://doi.org/10.1007/978-3-031-72667-5_11.

[47] Mingyuan Zhou and Lawrence Carin. Augment-and-conquer negative binomial processes. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 2555–2563, 2012. URL https://proceedings.neurips.cc/paper/2012/hash/6a61d423d02a1c56250dc23ae7ff12f3-Abstract.html.

A Training and sampling algorithms

In this section, we present SDD's training procedure in Algorithm 1 and the sampling in Algorithm 2.

Algorithm 1 SDD training algorithm.

```
def train_loss(x):
  # Create & concat sparsity bits
  x_sb = (x != 0).float()
  x_sb = (x_sb * 2 - 1)
  x = (x * 2 - 1)
  x = cat((x, x_sb), dim=1)

  # Forward diffusion steps
  t = uniform(0, 1)
  eps = normal(mean=0, std=1)
  x_t = sqrt(alpha(t)) * x + (1 - sqrt(alpha(t)) * eps

  # Compute self-cond estimate.
  x_0 = zeros_like(x_t)
  if uniform(0, 1) > 0.5:
    x_0 = net(cat([x_t, x_0], -1), t)
    x_0 = stop_gradient(x_0)

  # Predict and compute loss.
  x_0 = net(cat([x_t, x_0], -1), t)
  l2_loss = (x_0[:, :sb] - x[:, :sb]) ** 2
  ce_loss = cross_entropy_loss(x_0[:, sb:] - x[:, sb:])
  return l2_loss.mean() + ce_loss.mean()
```

Algorithm 2 SDD sampling algorithm.

```
def sample(steps, interm=False):
  x_t = randn(mean=0, std=1)
  x_pred = zeros_like(x_t)

  for step in range(steps):
    # Get time for current & next states.
    t_now = 1 - step / steps
    t_next = max(1 - (step + 1)/steps, 0)

    # Predict x_0.
    x_pred = net(cat([x_t, x_pred], -1), t_now)

    # Estimate x at t_next.
    x_t = ddim_or_ddpm_step(x_t, x_pred, t_now, t_next)

  # Return sparsified data point
  x_pred.clamp_(-1,1)
  x_dense = (x_pred[:, :sb] + 1) / 2
  x_sb = (x_pred[:, sb:] > 0).float()
  x_sparse = x_dense * x_sb
  return x_sparse
```

B DM architecture & training

We use a CNN-based U-Net architecture [14, 35, 29] for all DMs on calorimeter and sparse image data, with a base channel size of 256, three stages, two residual blocks per stage, and 37M parameters. For scRNA data, which lacks spatial structure, we adopt a skip-connected MLP following scDiffusion [25] with 5M parameters. Parameter counts and runtime overhead are detailed in Appendix C. We use Adam [16] with a constant learning rate of 0.0002 and batch size 256. Models are trained for 300K steps, using an exponential moving average of parameters (decay 0.9999). Sampling for DDIM and DDPM uses 1,000 steps.

C Compute resources and runtime analysis

Our experiments were executed on a NVIDIA DGX-A100 server with eight A100 (40GB) GPUs, an AMD Epyc 7742 CPU with 64 cores and 2TB of main memory. The server runs NVIDIA DGX Server Version 7.1.0 (GNU/Linux 6.8.0-60-generic x86_64) and has CUDA 12.9 and torch 2.7.0 installed. Due to the large scale and high computational cost of the deployed DMs, we were only able to run each experiment once. Each run requires substantial GPU resources and time making multiple runs or significance testing infeasible. Additionally, sampling is particularly expensive, further limiting repetition. While we acknowledge the importance of variance analysis, these practical constraints necessitated a single-run evaluation.

SDD enforces sparsity in the generated data and does not exploit the data sparsity to make the computations more efficient. Depending on the underlying U-Net architecture, we have more trainable parameters. We use CNNs with a total of 37M parameters for the calorimeter and the sparse images. However, for SDD, we observe an increase of only 0.07% in the number of parameters. For sparse data generation of scRNA data with an MLP with a total of 5M parameters, we observe an increase of 38.37% in the number of parameters. The increase in the number of parameters is much higher, as we have to double the size of the input and output layers. At the same time, we only need to double the number of input and output channels for the CNN-based U-Net for images.

We summarize the training runtime of DDPM and SDD in Table 3. We omit DDIM as the DDPM and DDIM variants have the same training procedure. We measured runtimes per training step on one A100 (40GB) that show minimal overhead for SDD compared to DDPM, as additional computations are efficiently parallelized.

Table 3: Shown are the runtimes of DDPM compared to SDD. The overhead for SDD is minimal.

UNet	Dataset	DDPM	SDD
CNN	MNIST	0.57 s	0.56 s
CNN	Fashion-MNIST	0.50 s	0.51 s
CNN	Muon Signal	0.56 s	0.56 s
CNN	Muon Background	0.59 s	0.58 s
MLP	Tabula Muris	0.02 s	0.02 s
MLP	Human Lung PF	0.02 s	0.02 s

D Further evaluating the recovery of ground truth sparsity patterns

How well do models match ground-truth sparsity? We present the sparsity distribution of samples generated by DDPM in Figure 6, complementing the main text's analysis of DDIM (Figure 3). The trends observed largely mirror those of DDIM. Specifically, DDPM also tends to underestimate sparsity across all datasets. Similar to DDIM-T, DDPM-T produces samples with an exaggerated upper tail in the sparsity distribution and a compressed lower end, particularly noticeable in the sparse image generation datasets. SDD remains the most faithful to the real data's sparsity profile: it mildly overshoots in the physics datasets (Muon Signal and Muon Background) and does so more substantially in the scRNA datasets (Tabula Muris and Human Lung Pulmonary Fibrosis), though still less severely than DDPM-T. On the vision datasets (Fashion-MNIST and MNIST), SDD continues to generate a wider and more accurate sparsity distribution.

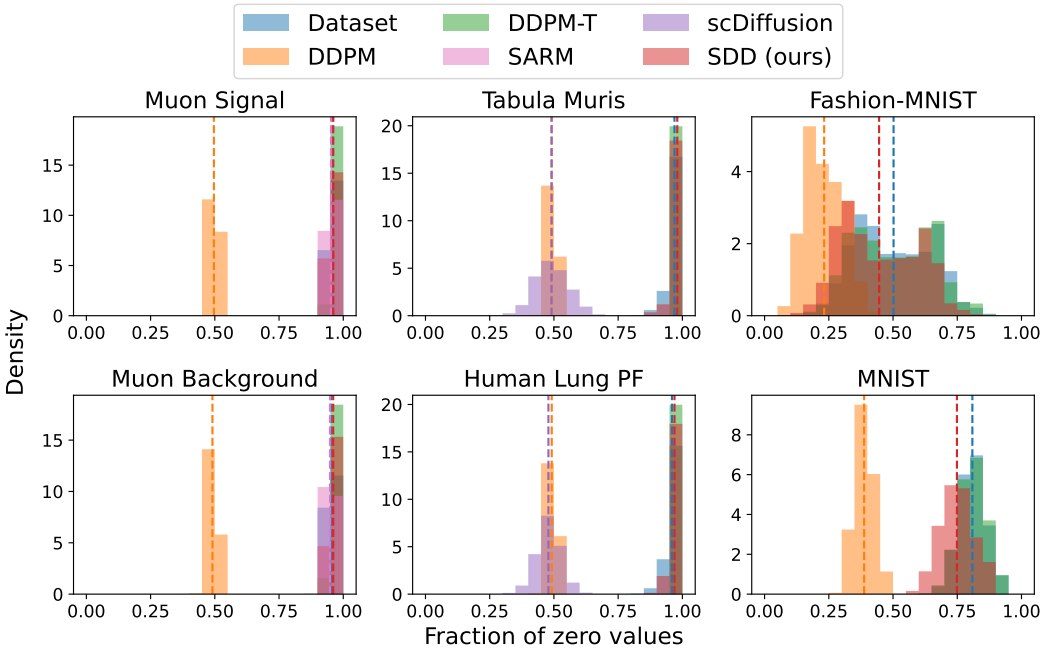


Figure 6: Sparsity distribution of real and generated data. Histograms (20 bins) and average sparsity levels (dashed lines) compare real data to samples from DDPM, DDPM-T, SARM, scDiffusion, and SDD. DDPM and scDiffusion underestimate sparsity; DDPM-T matches the average but lacks diversity and overshoots at high sparsity. SARM also underestimates sparsity, while SDD accurately matches both average value and distribution.

Sharpness of generated SBs Figure 7 illustrates the distribution of the sparsity bit (SB) logits across datasets. The histograms show that the logits are sharply concentrated near -1 and 1, indicating high model confidence when classifying each dimension as corresponding to a zero or non-zero pixel. This supports the feasibility of jointly diffusing discrete and continuous components. Moreover, the final bin (with upper bound 1.0) closely aligns with the true sparsity levels of each dataset: 95.2% for Muon Signal, 97.9% for Tabula Muris, 50.2% for Fashion-MNIST, 95.2% for Muon Background, 95.9% for Human Lung Pulmonary Fibrosis, and 80.9% for MNIST.

E Further results for particle physics: calorimeter image generation

Figure 8 displays representative calorimeter images generated by DDPM for two distinct cases: isolated muons (signal) and muons produced in association with jets (background). These qualitative results reveal behaviors similar to those observed for DDIM in Figure 1, illustrating fundamental limitations inherent in standard diffusion models when applied to sparse data. Specifically, DDPM-generated images fail to faithfully reproduce the localized and clustered energy depositions that characterize true muon interactions in the calorimeter. Instead, the energy distribution appears more diffuse and lacks the sharp, spatially coherent patterns critical for meaningful physics interpretation.

The thresholded variant, DDPM-T, which applies a post-hoc procedure to enforce dataset-level sparsity by zeroing out low-intensity pixels, mitigates some of these issues by producing more spatially isolated activations. However, these tend to manifest largely as single-pixel hits scattered across the calorimeter grid rather than physically plausible clusters, indicating that simple thresholding does not recover the underlying complex sparsity structure or spatial correlations.

In stark contrast, SDD-DDPM generates highly realistic calorimeter images that significantly improve these shortcomings. By explicitly modeling sparsity through Sparsity Bits and jointly diffusing continuous and discrete latent representations, SDD-DDPM reproduces the expected clustered energy patterns with remarkable fidelity. This confirms that incorporating sparsity awareness directly into the

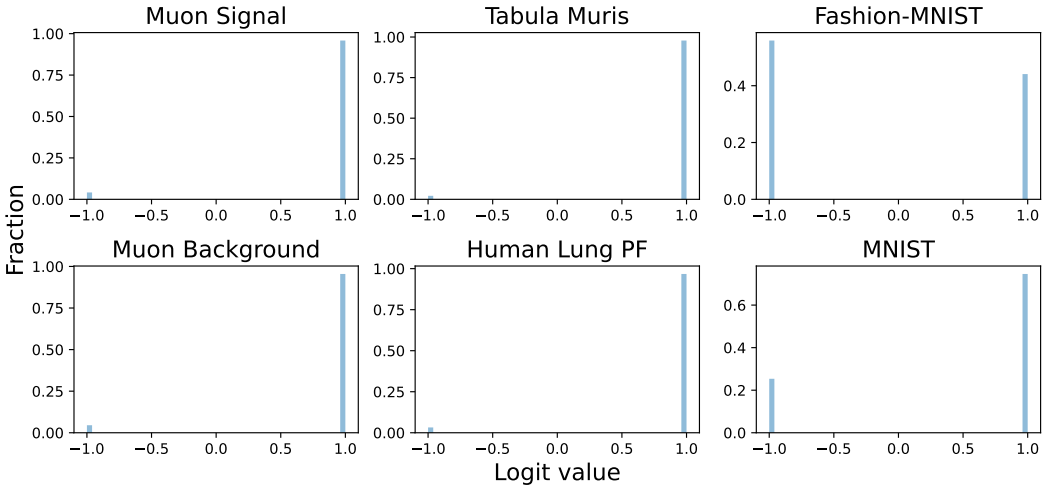
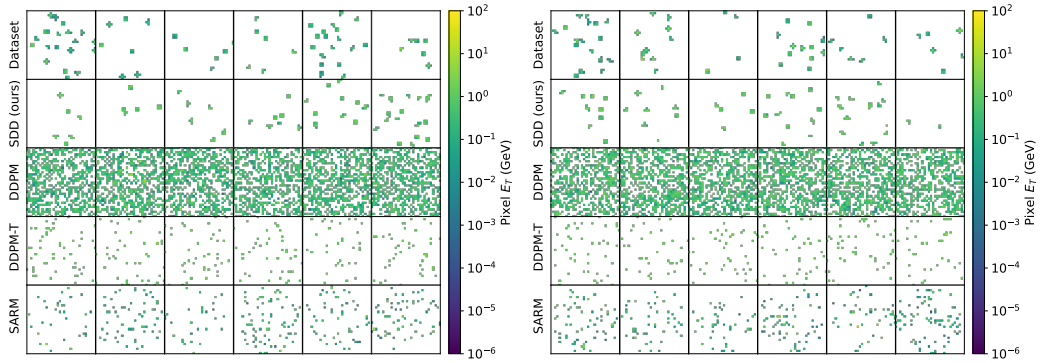


Figure 7: The histogram distribution shows 50 bins of the logits for sparsity bits for datapoints using SDD. The sparsity bits are extremely concentrated towards -1 and 1, which shows that discrete variables can be reliably diffused alongside continuous variables.



(a) Muon Signal

(b) Muon Background

Figure 8: Calorimeter images from the muon isolation study: signal images (top) and background images (bottom). Rows show samples from (1) real data, (2) SDD (ours), (3) DDPM, (4) DDPM with post-hoc thresholding to match dataset sparsity (DDPM-T), and (5) the domain-specific SARM baseline. Pixel intensity (GeV) visualizes energy deposition per cell (white=zero). SDD uniquely recovers the distinct, sparse, and clustered patterns characteristic of real data, whereas DDPM completely misses the sparsity, and DDPM-T and SARM show unrealistic isolated energy deposits.

generative process enables the model to capture the intricate spatial dependencies and zero patterns intrinsic to the data.

Complementary evidence is provided by the pixel-wise average calorimeter images illustrated in Figure 9. These averages further underscore the qualitative differences: both DDPM and DDPM-T exhibit relatively uniform, blurred activations across the image, indicative of non-specific and diffuse energy distributions that do not differentiate between zero and non-zero values effectively. By contrast, the averages constructed from SDD-DDPM samples reveal well-defined and physically meaningful spatial structures. For signal images, SDD-DDPM captures the characteristic uniform cylindrical energy distribution observed in isolated muon events. In contrast, in the background case, the energy is correctly concentrated near the muon's expected location rather than dispersed randomly. These distinct and interpretable patterns closely mirror the true physical phenomena measured by the calorimeter, substantiating the advantage of SDD in sparse data generation.

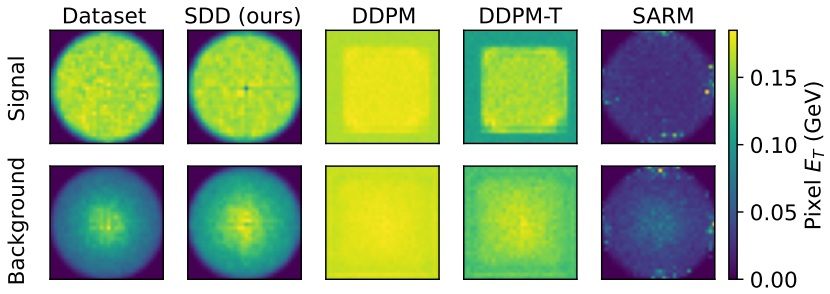
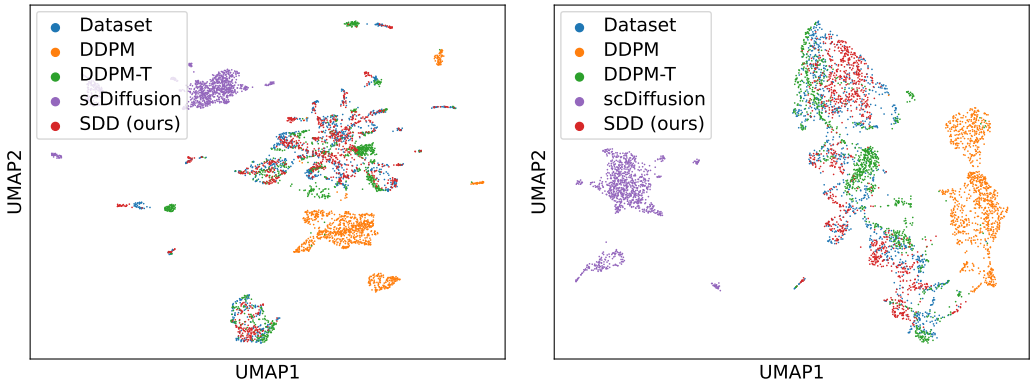


Figure 9: Averaged calorimeter images for Muon Signal and Background. SDD-generated averages closely match the actual data's spatial structure and sparsity. DDPM and DDPM-T produce almost uniform images, while the domain-specific SARM baseline fails to capture the actual intensity.



(a) Tabula Muris

(b) Human Lung Pulmonary Fibrosis

Figure 10: Shown are two-dimensional UMAPs of Tabula Muris and Human Lung Pulmonary Fibrosis, and the respective DDPM, DDPM-T, and SDD generated samples. DDPM, DDPM-T, and the domain-specific scDiffusion show little to no overlap with the respective dataset while SDD shows significant overlap, demonstrating that SDD more accurately captures the underlying structure and diversity of the real data distributions.

F Further results for biology: scRNA data generation

Figure 10 shows UMAP visualizations for the Tabula Muris and Human Lung Pulmonary Fibrosis datasets generated using DDPM. The trends mirror those observed with DDIM (Figure 5). Specifically, DDPM-generated samples show minimal overlap with the real cell populations, indicating limited biological realism. The thresholded variant, DDPM-T, moves the generated cluster closer to the real data manifold but at the cost of reduced diversity. In contrast, SDD-DDPM samples exhibit substantial overlap with the true cell populations, capturing both structure and diversity more faithfully. These results confirm that the benefits of sparsity-aware diffusion extend to DDPM-based models in the biological domain.

G Further Results for Sparse Image Generation

As discussed in Section 4.5, the visual quality of generated images for Fashion-MNIST and MNIST is largely comparable across all methods—DDIM, DDPM, their thresholded counterparts (DDIM-T and DDPM-T), and SDD. To complement these observations, Figures 11 and 12 provide detailed visualizations of the corresponding sparsity patterns. While DDIM, DDPM, and their thresholded versions produce images that appear visually plausible, closer examination reveals discrepancies in how well they capture the underlying sparsity structure of the datasets.

Table 4: Shown are the FID scores for sparse image generation. The results are similar for all considered settings.

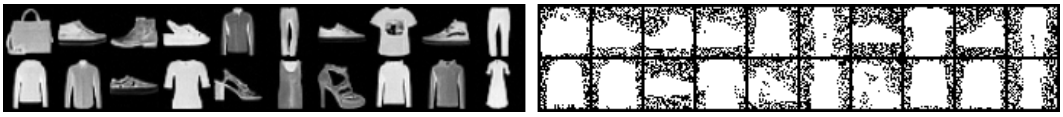
	FASHION-MNIST	MNIST
DDPM	25.62	23.35
DDIM	24.11	23.68
DDPM-T	25.83	23.82
DDIM-T	25.72	24.04
SDD DDPM (OURS)	27.81	25.37
SDD DDIM (OURS)	26.37	25.69

In particular, thresholded models such as DDIM-T and DDPM-T often fail to reproduce the fine-grained noise and soft sparsity transitions present in real data. For Fashion-MNIST, these models tend to eliminate subtle pixel activations near the edges, leading to overly clean object contours. In MNIST, the digits generated by DDIM-T and DDPM-T are noticeably narrower, lacking some of the variability found in true samples. These limitations underscore the fact that simple thresholding, while improving sparsity alignment on average, can suppress important structural variation in the data.

By contrast, SDD not only maintains competitive visual quality but also more accurately reproduces the sparsity distributions and edge-region behavior of real images. This highlights the advantage of explicitly modeling sparsity rather than relying on post-hoc thresholding to enforce it.

(a) Dataset

(b) DDPM



(c) DDIM

(d) DDPM-T

(e) DDIM-T

(f) SDD DDPM (ours)

(g) SDD DDIM (ours)

Figure 11: Shown are, from top to bottom: Fashion-MNIST images sampled from the dataset, DDPM and DDIM sampled images, thresholded DDIM and DDPM sampled images (DDPM-T, DDIM-T), and SDD (DDPM, DDIM) sampled images. The first column contains the samples, and the second contains the respective sparsity information. Despite highly visually similar images, DDIM and DDPM fail to reflect the sparsity, while the thresholded variants DDIM-T and DDPM-T miss fine-grained details on the edges. The proposed SDD more faithfully reflects these fine-grained details.

(a) Dataset

(b) DDPM

(c) DDIM

(d) DDPM-T

(e) DDIM-T

(f) SDD DDPM (ours)

(g) SDD DDIM (ours)

Figure 12: Shown are, from top to bottom: MNIST images sampled from the dataset, DDPM and DDIM sampled images, thresholded DDIM and DDPM sampled images (DDPM-T, DDIM-T), and SDD (DDPM, DDIM) sampled images. The first column contains the samples, and the second contains the respective sparsity information. Despite highly visually similar images, DDIM and DDPM fail to reflect the sparsity, while the thresholded variants DDIM-T and DDPM-T miss fine-grained details on the edges, resulting in narrower digits (and the respective sparsity information). The proposed SDD more faithfully reflects these fine-grained details.