GaussianDiffusion: Learning Image Generation Process in Gaussian Representation Space

JDSE — 10th Junior Conference on Data Sciences and Engineering

Arijit Samal Simon Coessens Akash Malhotra
Nacera Seghouani

CentraleSupélec

September 24, 2025

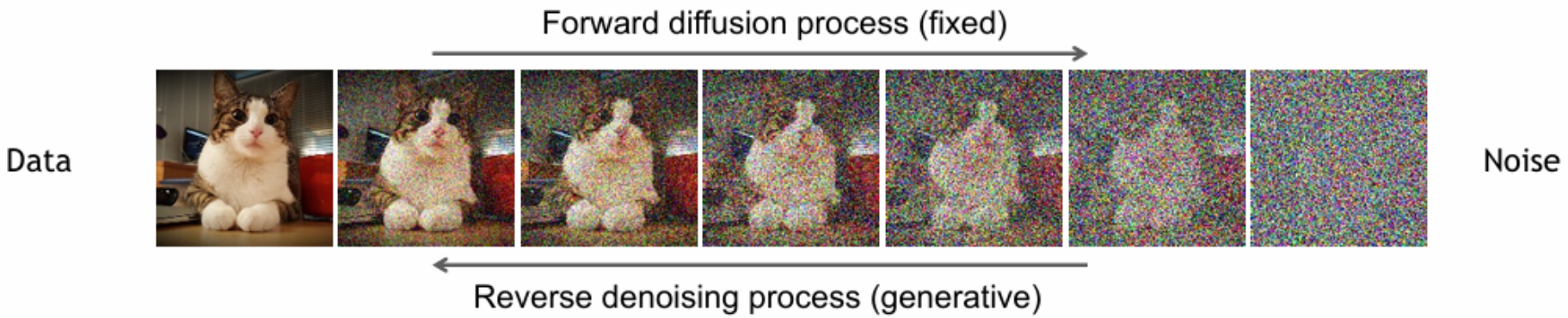
Outline

Introduction

Diffusion Models

Diffusion models are SOTA for image generation, but many still operate in dense pixel space that is computationally intensive and lacks geometric structure.
Diffusion models consist of two processes:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



Introduction

Gaussian Splatting and Gaussian Image Representation (GIR)

Gaussian Splatting has proven highly effective as a representation method in 3D settings.

- **2D Gaussian Representation:** Represents images as sets of points modeled by Gaussian kernels.
- **Key Features and Advantages:**
  - Each Gaussian kernel is parameterized and optimized via gradient descent.
  - Provides a permutation invariant image representation.
  - Provides a more compressed representation compared to pixel. (inherently latent)

Parameter	Gaussian 1	Gaussian 2
Σ_x	0.5404	0.4745
Σ_y	0.5529	0.4569
ρ	-0.7177	0.7665
α	0.5715	0.4888
Colour	0.0000	0.0000
Coordinates	(-1.0322, -1.2237)	(-1.5011, -0.0649)

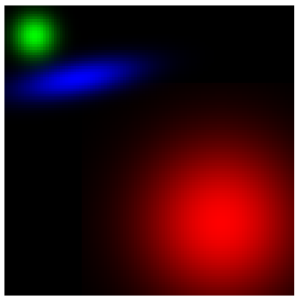


Figure: 2D Gsplat

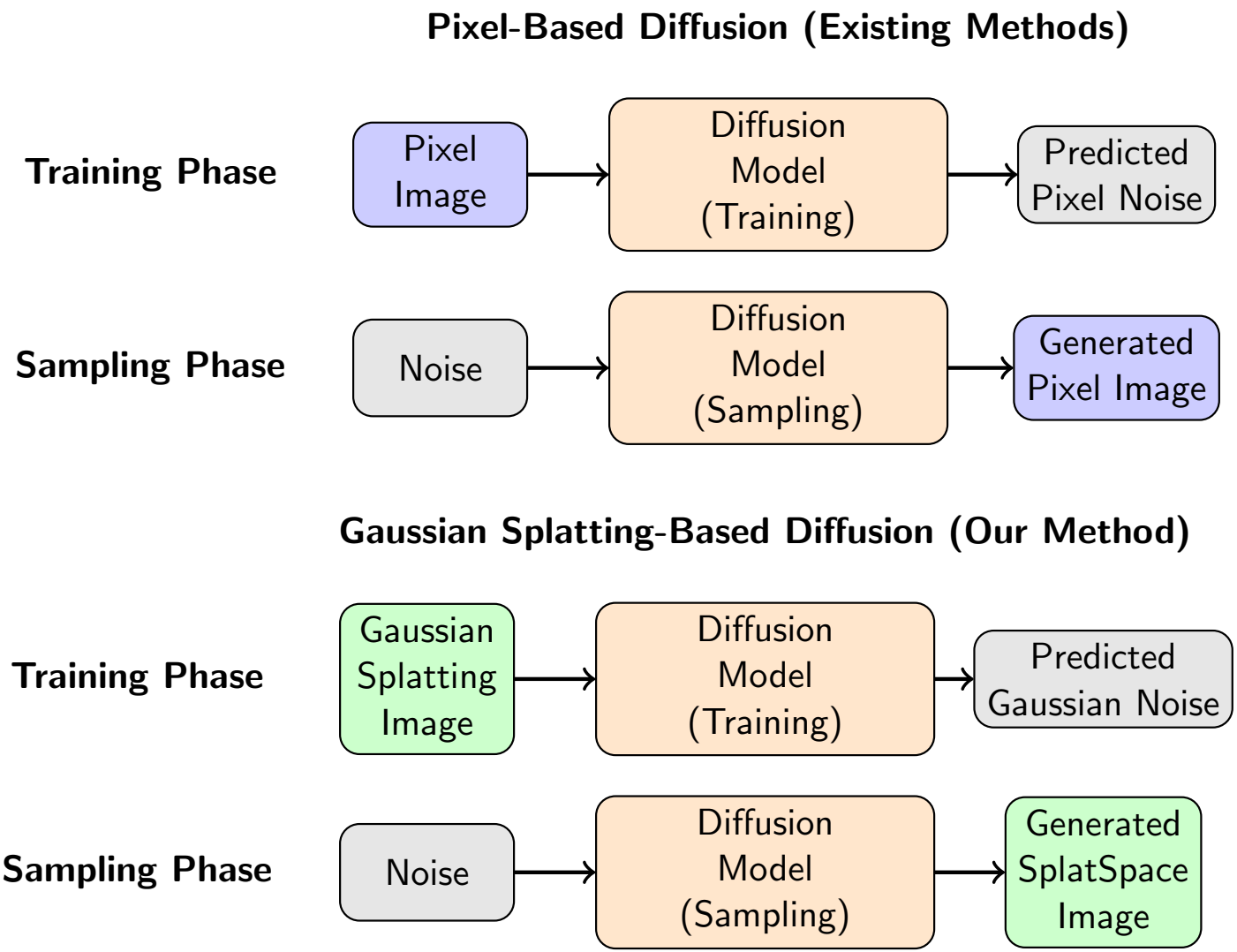
Motivation

- Combining Gaussian Splatting with diffusion models presents a promising opportunity to enhance image generation capabilities while reducing computation overhead
- Gaussian Splatting compresses images into an **inherently latent space**.
- Provides a **permutation-invariant** representation, beneficial for diffusion-based learning.
- Extension to 3D scene generation
- **SplatSpace** offers a more suitable representation for diffusion models.

Objective

- Encode images using Gaussian Splatting to represent input images in **SplatSpace** (the space of Gaussian-based image representations).
- Train a diffusion model directly in **SplatSpace** to explore its generative potential.

Comparison between methods



Datasets Overview

MNIST

- Image size: 28×28 (grayscale)
- 60,000 images

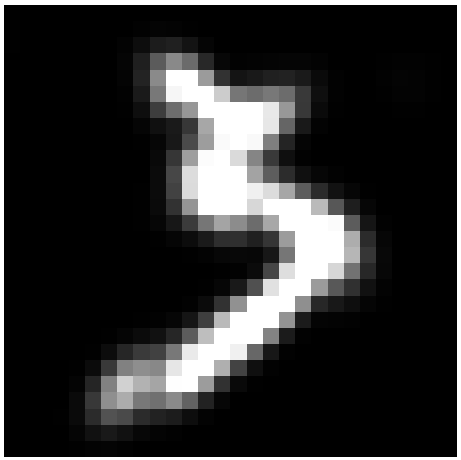


Figure: Sample MNIST image

Sprites

- Image size: 32×32 (RGB)
- 82,000 images

Figure: Sample Sprite image

Gaussian Splatting Demonstration

Figure: Original Image

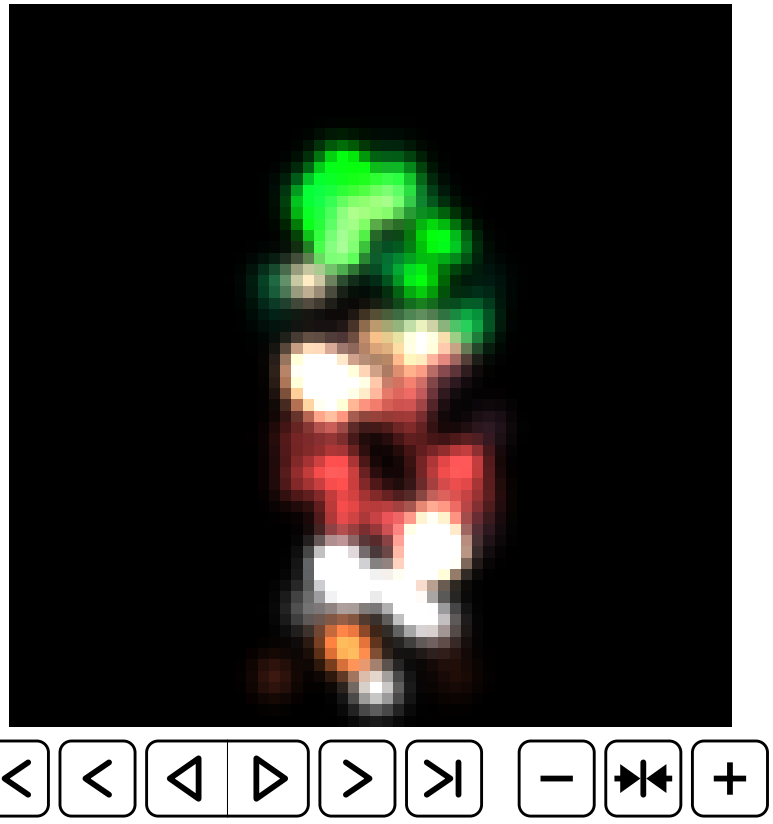


Figure: Reconstructed Image

Final PSNR: 27.45 dB

Forward Diffusion

Noise Injection

- Images are represented via Gaussian parameters: σ_x, σ_y, θ, α, colors, and coordinates.
- A 7-dimensional noise vector (for grayscale images) or 9-dimensional noise vector (for RGB images) is added after normalizing parameters.
- Noise disrupts the structure of the Gaussian blobs.

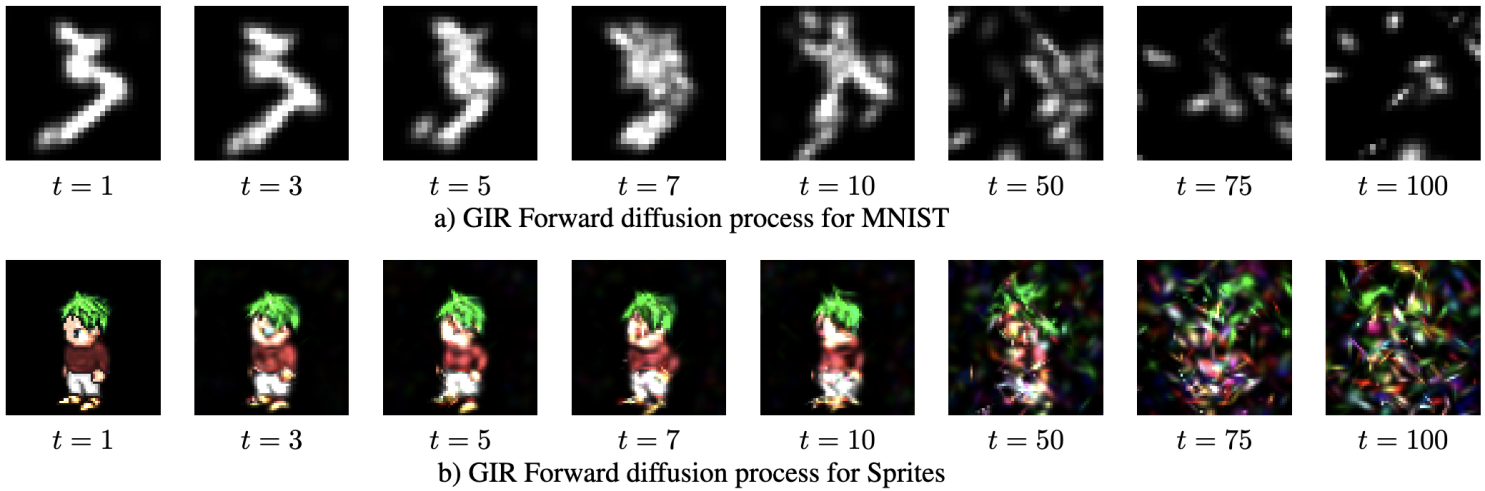


Figure: Forward process in Gaussian space.

Forward Process Comparison

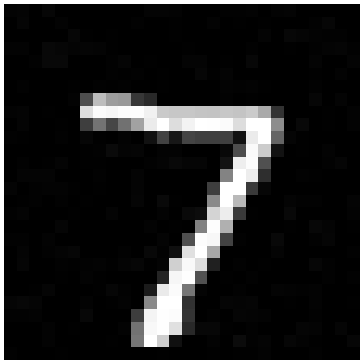


Figure: Pixel MNIST

Figure: Pixel Sprites

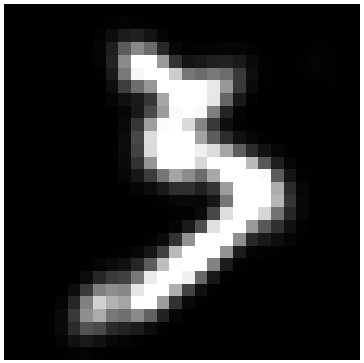


Figure: Gaussian MNIST

Figure: Gaussian Sprites

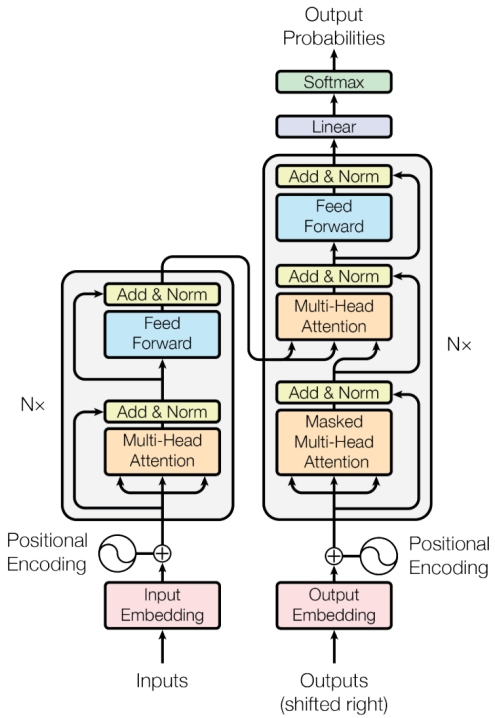
Reverse Diffusion

Transformer Architecture

We use an architecture inspired by the encoder of the transformer.

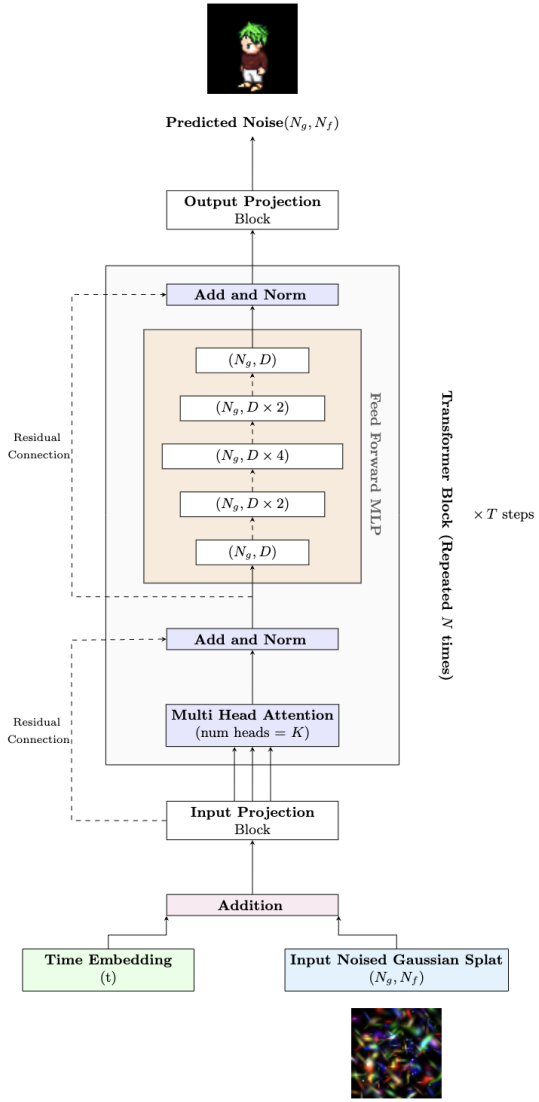
In transformer the input is tokens (one sentence consists of multiple tokens)

In our case it's gaussians (one image is made using N gaussians)

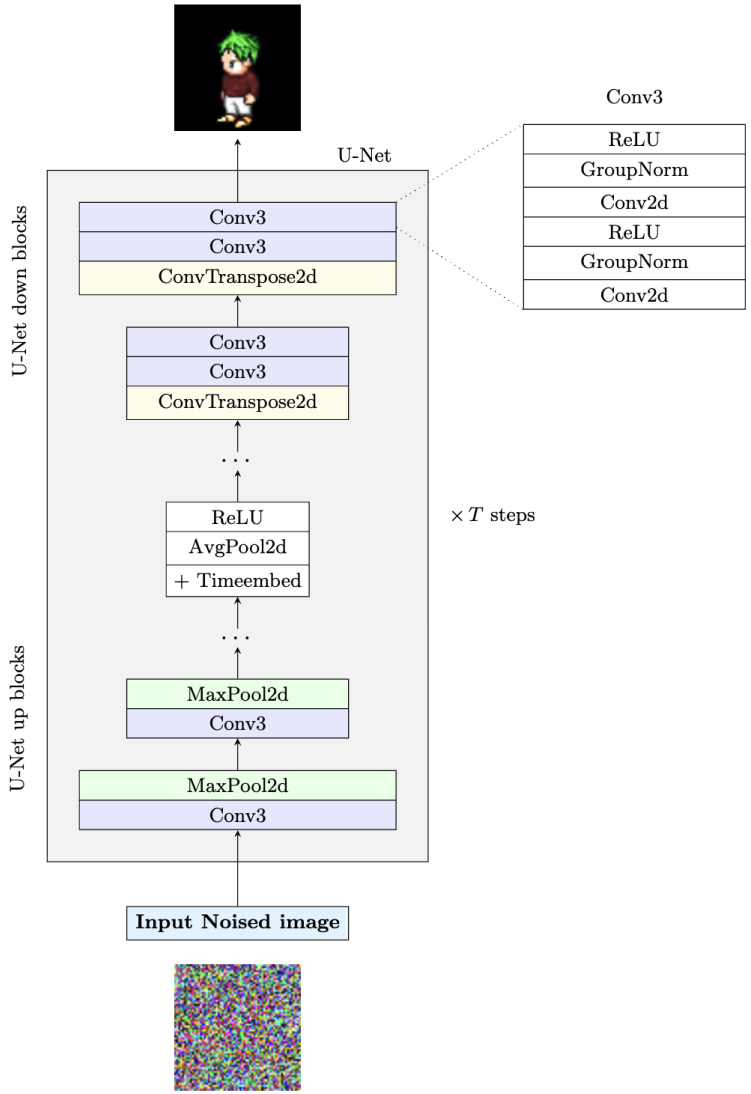


Reverse Diffusion

GaussianTransformer Architecture



U-NET architecture



Results & Evaluation

Overview

- Benchmarks: MNIST (grayscale) and Sprites (RGB), both downsampled to 32×32 for resolution-controlled comparison.
- Consistent GIR latent across datasets: fixed $N = 150$ splats, $F = 8$ features $\Rightarrow N \times F = 1200$-D. (intensity alpha not considered)
- Equal training budget: 24 GPU-hours on a single NVIDIA A100-40GB.

Sampling Results Pixel vs Gaussian MNIST

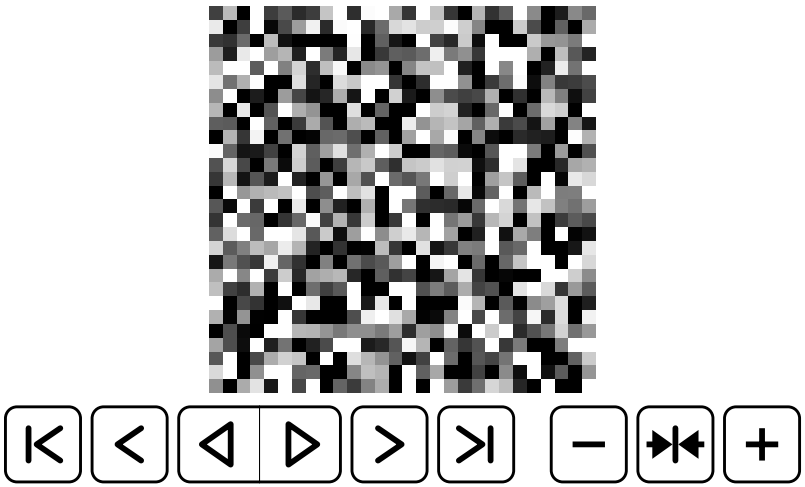


Figure: Sampling MNIST Pixel

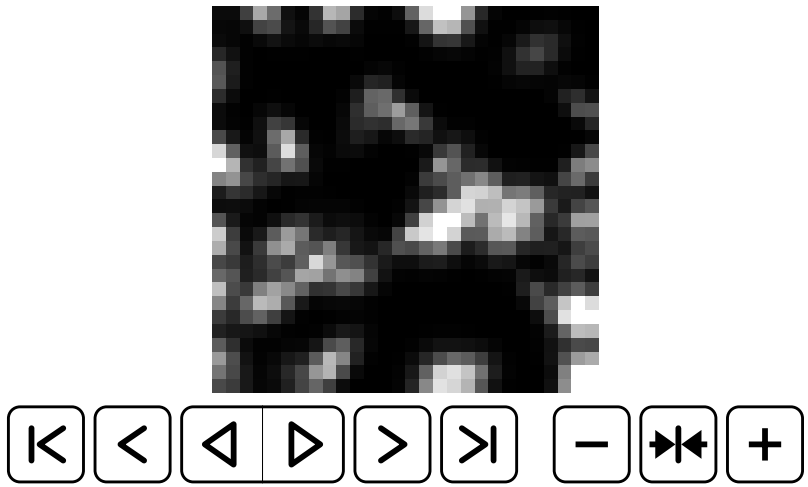


Figure: Sampling MNIST Gaussian

Sampling Results Pixel vs Gaussian

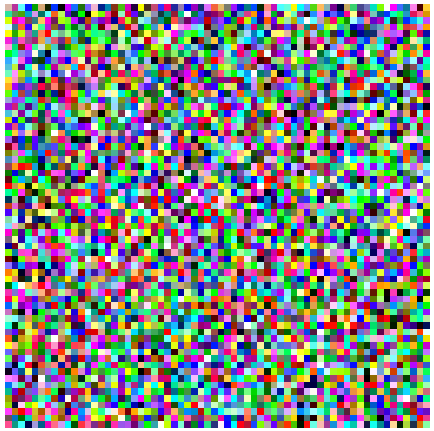


Figure: Sampling Sprites Pixel

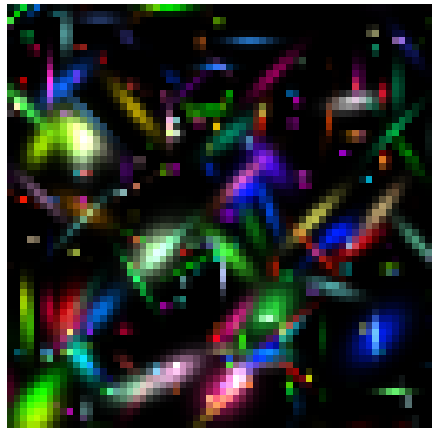


Figure: Sampling Sprites Gaussian

GIR Statistics

Dataset	Image Size	Gauss./image	compr. ratio	PSNR (dB)
MNIST (grayscale)	32×32	150	2.0	30.1
Sprites (RGB)	32×32	150	2.7	36.4

Table: GIR is fixed across datasets: compact 1200D latent per image.

Runtime & Sampling Efficiency

Model	**Params** (M)	**FLOPs**/step (G)	**Wall**/step (ms)‡	**Steps** (T=gen)	**Total FLOPs** (T)	**Total time** (T)
GaussianTransformer	≈40	4.0	1.2	200	0.80 T	0.24 s
U-Net (pixel)	≈38	11.4	5.4	1000	11.4 T	5.40 s

‡FP16/TF32 mixed precision, batch 32, single NVIDIA A100-40GB. Backward pass cost is ≈ 2× forward.

Table: Compute profile at 32×32. GaussianDiffusion uses **2.8**× fewer FLOPs/step and runs **4.5**× faster per step; with **5**× fewer steps, total time drops from **5.4 s** to **0.24 s** (**22**× speed-up).

- **Smaller latent**: 3072 pixels → 1200 GIR parameters.
- **Stronger prior**: ∼5× fewer diffusion steps.
- **GPU-friendly**: attention GEMMs vs. many 3×3 convs.
- **training stability:** Pixel diffusion unstable/noisy early samples but GaussianDiffusion coherent samples from epoch 10 onward faster, more stable convergence.

Conclusion

- Replacing the pixel grid with a structured set of **2D Gaussians (GIR)** is a viable alternative for image generation.
- The diffusion process runs entirely in splat space with a **GaussianTransformer**.
- massive drops in FLOPs, steps and total time
- coherent digits/sprites emerge by **epoch $\sim$10**; pixel baseline is noisier early.
- Visual samples show **comparable perceptual quality**
- Offers a **clearer, more interpretable** route to structured diffusion.

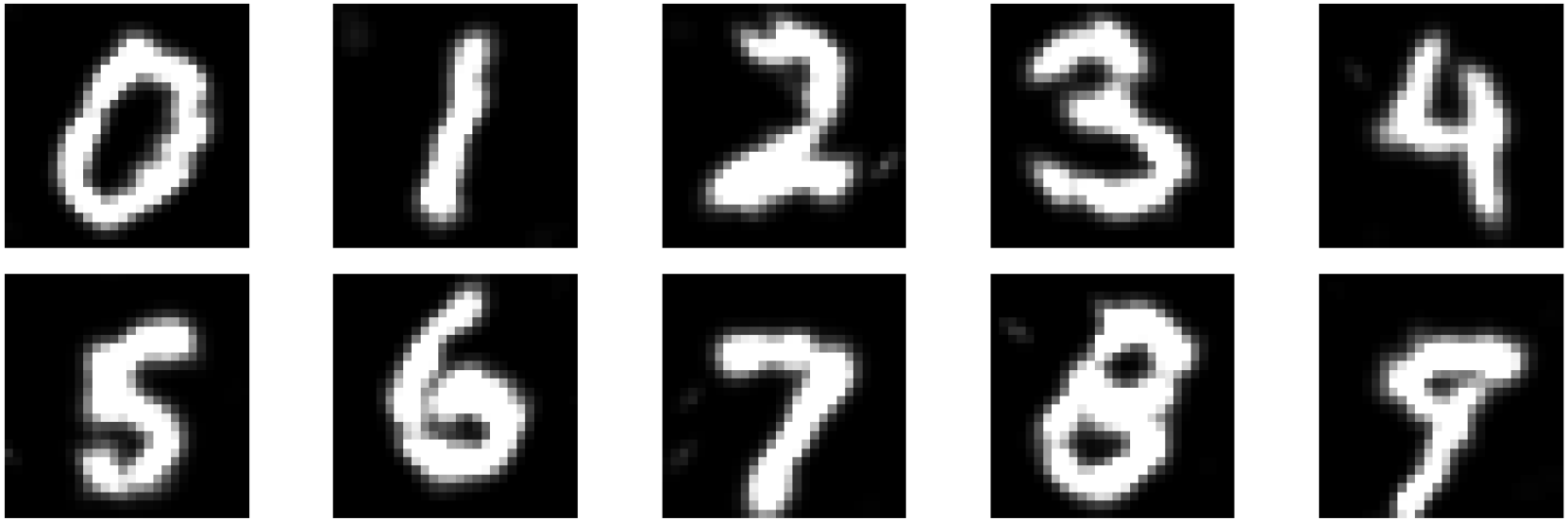
Future Work

- **Geometric attention:** Augment the core transformer with better modules for better quality results
- **Quantitative metrics:** add FID/KID; conduct ablations over N (splats), F (features), step counts etc.
- **Scale-up studies:** higher resolutions, diverse datasets; analyze GIR vs. pixel-space scaling laws.
- **Systems optimization:** deeper mixed-precision tuning, fused kernels, and memory-aware batching for attention GEMMs.

Thank you!

Questions?

MNIST sampling grid



Sprites sampling grid