Generating 3D Gaussians: An exploration

Abhimanyu Suthar
Advised by Eugene Vinitsky
NYU Tandon School of Engineering
{abs9477,ev2237}@nyu.edu

Abstract

This paper explores approaches for generating 3D Gaussian primitives, which have emerged as an efficient representation for neural rendering and 3D scene synthesis. While diffusion models show promise in capturing spatial relationships between Gaussians, I find that managing the complex parameter space of 3D Gaussians presents a unique challenge. This paper serves as an educational edification, intended to guide readers through the technical intricacies of 3D Gaussian generation. https://github.com/Abhimanyu-0/Generating_3D_Gaussians

1 Introduction

Recent advances in neural rendering and 3D scene representation have sparked significant interest in generating realistic 3D content. There are several domains that stand to benefit from 3D generation techniques. These include AR/VR applications, immersive games, and virtual environments that can be used to simulate diverse scenarios in order to assist the training of personnel such as astronauts, pilots.
3D Gaussian splatting has emerged as an efficient alternative to traditional neural rendering approaches where scenes are represented as collections of 3D Gaussians with learned parameters (position, covariance, and opacity). While both point clouds and 3D Gaussians serve as primitives for representing 3D scenes, their fundamental characteristics make them distinctly suited for different aspects of scene generation. Point clouds, consisting solely of discrete xyz coordinates in space, require extremely dense sampling to represent continuous surfaces and offer no inherent mechanism for handling uncertainty or shape information. This makes generation particularly challenging, as the model must output precise coordinates to avoid gaps or artifacts in the final rendering. In contrast, 3D Gaussians provide a richer representational primitive through their position (mean), shape (covariance matrix), and opacity parameters. Each Gaussian effectively describes a region of space rather than a single point, allowing for natural handling of uncertainty and smoother surface representation with fewer primitives. This property makes them particularly appealing for generative tasks, as slight imperfections in generation can be absorbed by the Gaussian's spatial extent, while their overlapping nature inherently creates continuous surfaces. The ability of Gaussians to capture local spatial relationships through their covariance matrices also provides a more compact and potentially more learnable representation for neural networks, striking a balance between the simplicity of point clouds and the complexity of full neural fields.

In this work, I investigate two distinct approaches to generating 3D Gaussian parameters: a Transformer-based diffusion model and a Autoencoder-based architecture

- The VAE-based method, while promising in its ability to learn a discrete latent space, produced parameter values that deviated significantly from those needed for realistic scene reconstruction.
- The Transformer-based approach resulted in clustered parameter values, suggesting limitations in capturing the full range of spatial relationships necessary for coherent 3D scenes.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

2 Background

2.1 Diffusion Models

Generative models are trained to sample from a distribution [1]. For example, given a set of 3D Gaussians from a scene, we want to create a model that can generate additional samples that follow the same underlying distribution. Within the landscape of generative modeling, diffusion models [2] have emerged as a powerful approach, demonstrating state-of-the-art capabilities across various domains including image synthesis, text generation, and 3D scene generation. At their core, diffusion models operate by learning to reverse a gradual noising process. The forward process systematically corrupts data by adding Gaussian noise over multiple timesteps until the data becomes pure noise. The model then learns to reverse this process, gradually denoising random Gaussian noise into coherent samples. This approach can be understood through two complementary theoretical frameworks:

2.1.1 Variational Perspective

The first framework interprets diffusion models as a special case of hierarchical variational autoencoders (VAEs). In this view, the forward noising process defines a sequence of latent variables $\mathbf{x}_t{}_{t=1}^{T}$, where each variable represents the data at a different noise level. The model learns a sequence of decoders that progressively remove noise to recover the original data distribution. This interpretation provides a principled way to train diffusion models through likelihood-based objectives.

$$\mathcal{L} = \mathbb{E}\left[-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] \quad + \sum_{t=2}^{T} \mathbb{E}\left[D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)|p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))\right] \tag{1}$$

2.1.2 Score Matching Perspective

Alternatively, diffusion models can be viewed through the lens of score matching, where the model learns to estimate the gradient of the log probability density (score function) of the data at different noise levels:

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) = -\frac{1}{\sigma_t^2}(\mathbf{x} - \mu_\theta(\mathbf{x}, t)) \tag{2}$$

where $\mu_\theta(\mathbf{x}, t)$ is the model's prediction of the mean of the reverse process.

2.1.3 Forward Process

The forward process is defined as a Markov chain that gradually adds noise to the data:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{3}$$

where $\beta_t{}_{t=1}^{T}$ is a noise schedule that controls the rate of noise addition. After T steps, the data is transformed into pure noise:

$$\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I}) \tag{4}$$

2.1.4 Reverse Process

The reverse process learns to progressively denoise the data:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I}) \tag{5}$$

The model is trained to minimize the reconstruction loss between predicted and actual denoised samples:

$$\mathcal{L}simple = \mathbb{E}\left[t, \mathbf{x}_0, \epsilon\right] |\epsilon - \epsilon\theta(\mathbf{x}_t, t)|^2 \tag{6}$$

For 3D Gaussian generation, this framework is particularly effective as it naturally handles:

- Continuous parameter spaces of Gaussian primitives (position, scale, rotation, opacity)
- Progressive refinement of spatial relationships between primitives
- Preservation of geometric constraints through the denoising process

The step-by-step nature of diffusion models provides several advantages:

1. **Controlled Generation**: The progressive denoising allows for fine-grained control over the generation process.
2. **Quality-Speed Tradeoff**: The number of denoising steps can be adjusted to balance between sample quality and generation speed.
3. **Interpretable Intermediates**: Each intermediate state provides insight into the generation process.

2.2 3D Gaussian Representations

In the context of 3D scene representation [3], each Gaussian primitive is parameterized by a set of attributes:

$$\mathcal{G} = \{\mu, \Sigma, R, \alpha\} \in \mathbb{R}^3 \times \mathbb{R}^3 \times SO(3) \times [0, 1] \tag{7}$$

where $\mu \in \mathbb{R}^3$ represents the spatial position, $\Sigma \in \mathbb{R}^3$ defines the scale along each axis, $R \in SO(3)$ represents the rotation as a quaternion, and $\alpha \in [0, 1]$ denotes the opacity.

2.2.1 Diffusion Process for Gaussian Parameters

When applying diffusion models to 3D Gaussian primitives, special consideration must be given to each parameter type:

Position Diffusion For spatial coordinates μ, the forward process follows standard Gaussian diffusion:

$$q(\mu_t|\mu_{t-1}) = \mathcal{N}(\mu_t; \sqrt{1-\beta_t}\mu_{t-1}, \beta_t\mathbf{I}) \tag{8}$$

Scale Handling The scale parameters Σ require special treatment as they must remain positive. We apply diffusion in log-space:

$$\log(\Sigma_t) = \sqrt{1-\beta_t}\log(\Sigma_{t-1}) + \sqrt{\beta_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}) \tag{9}$$

Rotation Diffusion For rotations represented as quaternions, we must preserve the unit norm constraint. The forward process uses spherical interpolation (slerp):

$$R_t = slerp(R_{t-1}, R_{noise}, \sqrt{\beta_t}) \tag{10}$$

where R_{noise} is a random unit quaternion.

Opacity Diffusion The opacity parameter α must remain in $[0, 1]$. We apply diffusion in logit space:

$$logit(\alpha_t) = \sqrt{1-\beta_t}logit(\alpha_{t-1}) + \sqrt{\beta_t}\epsilon \tag{11}$$

2.2.2 Learning Challenges

The diffusion model must handle several unique challenges when working with 3D Gaussians:

- **Parameter Coupling**: Changes in scale and rotation are interdependent and must maintain physical consistency
- **Manifold Constraints**: Each parameter type lies on a different manifold with specific constraints
- **Multi-scale Structure**: Gaussians at different scales require different noise schedules
- **Scene Coherence**: Generated Gaussians must maintain spatial relationships and avoid interpenetration

2.2.3 Architecture Considerations

To address these challenges, we employ several architectural modifications:

1. **Parameter-Specific Decoders**: Separate decoders for position, scale, rotation, and opacity to respect their constraints

2. **Spatial Conditioning**: Local attention mechanisms to maintain coherent spatial relationships
3. **Scale-Aware Features**: Multi-scale feature hierarchies to handle Gaussians of varying sizes
4. **Constraint Preservation**: Projection layers to ensure generated parameters satisfy manifold constraints

VQVAE

The Vector Quantized Variational Autoencoder (VQ-VAE)[4], introduced by van den Oord et al. (2017), represents a significant advancement in discrete latent variable models. Unlike traditional VAEs that use continuous latent variables, VQ-VAE combines vector quantization with the VAE framework to learn discrete latent representations while maintaining high reconstruction quality. The key innovation of VQ-VAE lies in its approach to discretization. The model maps encoder outputs to the nearest vectors in a learned codebook of embeddings, effectively creating a discrete bottleneck in the latent space. To handle the non-differentiable nature of this quantization operation, VQ-VAE employs a straight-through gradient estimator that copies gradients from the decoder input directly to the encoder output. The model's loss function consists of three terms: a reconstruction loss, a codebook learning term that moves embedding vectors toward encoder outputs, and a commitment loss that prevents the encoder outputs from growing arbitrarily large.

3 Dataset

ShapeSplatsV1 is a new dataset with over 52,000 3D objects, each described by Gaussian splats. These splats are stored in PLY files, where each point has attributes like position, opacity, its size, and how it's rotated. This setup helps in creating detailed 3D models for research in computer graphics.

The dataset also includes quality checks like PSNR and SSIM, plus it measures how different these splats are from traditional point clouds. It's designed for people to train AI models on 3D shapes without needing lots of labeled data, thanks to the Gaussian-MAE method.

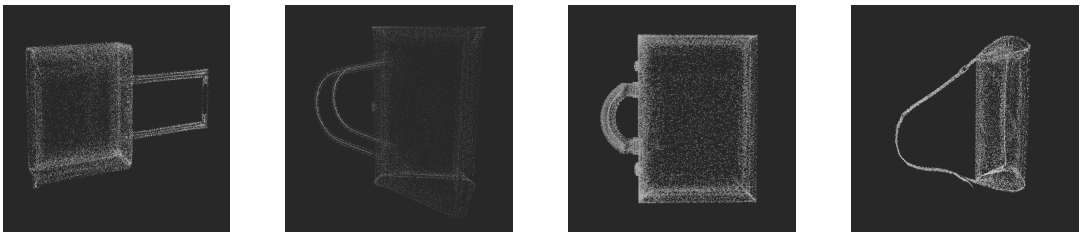

Figure 1: Example Bag PLY files in the ShapeSplatsV1 dataset

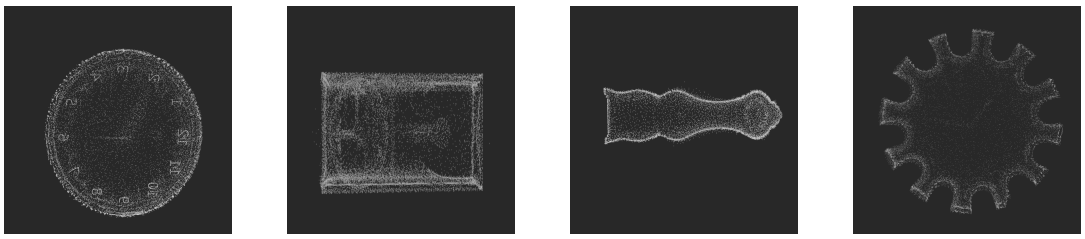

Figure 2: Example Clock PLY files in the ShapeSplatsV1 dataset

4 Implemented Architectures

4.1 Vector Quantized Variational Autoencoder

The implemented (VQ-VAE) architecture is designed specifically for processing 3D Gaussian parameters[5], employing a discrete latent space that effectively captures the essential characteristics

of 3D shapes. The encoder pathway consists of 3D convolutions that progressively transform the 11-channel input (comprising position, scale, rotation, and opacity parameters) through a series of transformations, first to 32 channels and then to 64 channels. Each convolution is paired with Group Normalization and GELU activations, ensuring stable training and effective feature extraction. The encoded features are then passed through a vector quantization layer, which maps them to the nearest vectors in a discrete codebook of 512 learned embeddings. The decoder mirrors this structure in reverse, starting from the quantized embeddings and reconstructing the full set of Gaussian parameters. It employs transposed 3D convolutions to gradually transform the 64-channel quantized representation back to the original 11 parameters, maintaining spatial coherence throughout the process. Both the encoder and decoder utilize residual connections to facilitate better gradient flow and preserve fine details. The architecture's uniqueness lies in its vector quantization layer, which creates a discrete bottleneck that helps capture meaningful, reusable patterns in the Gaussian parameter space while maintaining the ability to generate diverse and coherent 3D shapes. This discrete representation proves particularly effective for maintaining the structural consistency.

4.2 Variational Autoencoder

The Variational Autoencoder (VAE) architecture implemented here is specifically designed to handle the unique challenges of 3D Gaussian parameters, processing a 11-dimensional feature space that encompasses position, scale, rotation, and opacity values. The encoder pathway employs a series of 3D convolutions that gradually increase the channel depth while preserving crucial spatial relationships, starting from 11 channels and expanding to 32, then 64 channels. Each convolution layer is followed by Group Normalization with 8 groups and GELU activation functions, helping maintain stable training dynamics and effective feature extraction. The encoder culminates in producing the parameters for the latent distribution, capturing the essential characteristics of the 3D Gaussian representations.

The decoder mirrors this structure in reverse, beginning with a latent sample and progressively reconstructing the full parameter space. It utilizes transposed convolutions to upsample the spatial dimensions while reducing the channel depth back to the original 11 parameters. A key feature of our architecture is the inclusion of residual connections in both the encoder and decoder, which facilitate better gradient flow and enable the network to preserve fine details of the Gaussian parameters. The decoder's final output undergoes parameter-specific normalization to ensure physical validity: quaternion normalization for rotations, positive value constraints for scales, and sigmoid activation for opacity values. This careful handling of parameters ensures that the generated Gaussians maintain realistic physical properties while allowing for creative variation in the generated shapes.

4.3 Diffusion Transformer model

The architecture centers around a Transformer model specifically designed to process and generate 3D Gaussian parameters through a diffusion process. The model begins with an 11-dimensional input representing each Gaussian primitive (3D position, 3D scale, 4D rotation quaternion, and opacity), which is first normalized through a batch normalization layer. These parameters are then embedded into a higher-dimensional space through a linear projection, followed by another normalization step. A crucial component is the time embedding module, which converts the diffusion timestep into a positional encoding using sinusoidal functions, allowing the model to understand where it is in the diffusion process. This temporal information is added to the Gaussian parameter embeddings before being processed by the Transformer. The core of the model consists of a Transformer encoder with six layers and four attention heads, processing the embedded Gaussian parameters while maintaining their relationships. Each Transformer layer includes self-attention mechanisms and feed-forward networks with a dimensionality four times the hidden size, using dropout for regularization. The output features from the Transformer are then processed by separate parameter prediction networks - one each for position (mean), scale, rotation, and opacity. These networks ensure the outputs maintain their physical constraints: scales are kept positive through softplus activation, rotations are normalized to valid quaternions, and opacity is bounded appropriately. This specialized architecture allows the model to learn the complex distributions of 3D Gaussian parameters while respecting their geometric and physical constraints.

5 Results

5.1 VAE

The VAE architecture's reconstruction capabilities was tested across three distinct datasets, demonstrating its ability to capture complex geometric structures. The training experiments encompassed diverse object categories, each presenting unique geometric challenges and structural patterns: The bag reconstruction model was trained on a dataset of 484 MB, comprising 84 PLY files. For the clock reconstruction, we utilized a more extensive dataset of 3.5 GB containing 652 PLY files, enabling the model to learn finer temporal device details. The bottle reconstruction leveraged an intermediate-sized dataset of 1.8 GB, consisting of 499 PLY files, which provided sufficient variation in cylindrical geometries and surface features. Across all three object categories, the VAE did not optimally learn the inherent structural patterns and geometric constraints present in the training data. However, they maintain the same spatial characteristics and these results validate approach to learning the complex distributions of 3D Gaussian parameters while preserving essential geometric and physical constraints.

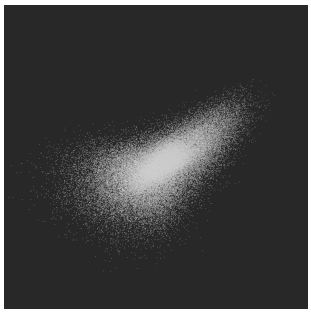

Figure 3: VAE reconstructed bag

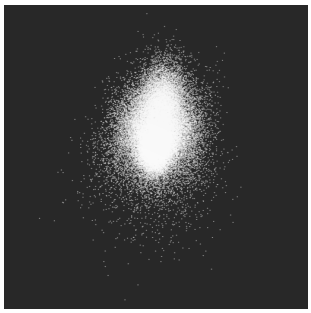

Figure 4: VAE reconstructed clock

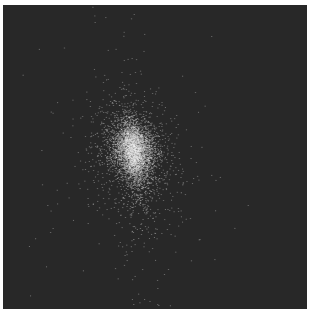

Figure 5: VAE reconstructed bottle

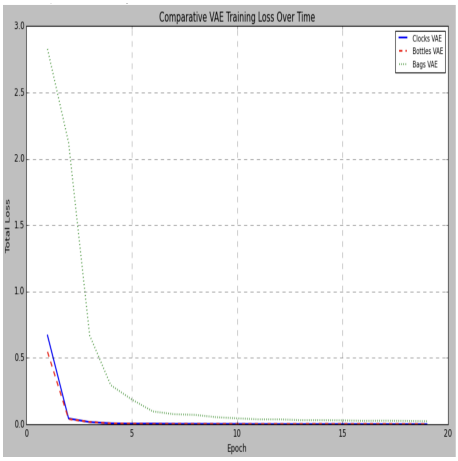

Figure 6: VAE losses for 3 datasets

5.2 Diffusion Transformer

5.3 VQVAE

Key Comparative Analysis:

1. **Position Coverage**: - VQVAE shows the widest position range (-0.341 to 0.278) - All models still exhibit significant compression from original range - VQVAE's broader range suggests better spatial distribution

Table 1: Comparison of Generated Parameters Across All Models for Bag Files

	Transformer		VAE		VQVAE		Original	
Parameter	Min	Max	Min	Max	Min	Max	Min	Max
Position	-0.274	0.188	-0.284	0.207	-0.341	0.278	-0.750	0.678
Scale	0.001	0.001	0.001	0.100	0.038	0.100	0.001	0.100
Opacity	0.100	0.100	0.000	1.000	0.050	0.166	0.000	0.677
Gaussians	32,000				1,488		N/A	

2. **Scale Parameters**: - VQVAE maintains meaningful scale variation (0.038 to 0.100) - Better than Transformer's collapsed scales (0.001) - More concentrated than VAE's full range (0.001 to 0.100) - Minimum scale of 0.038 suggests better minimum size constraint

3. **Opacity Distribution**: - VQVAE shows controlled opacity range (0.050 to 0.166) - More realistic than VAE's extreme range (0.000 to 1.000) - More varied than Transformer's fixed value (0.100) - Better minimum opacity threshold than both alternatives

These statistics suggest that the VQVAE achieves a better balance between parameter preservation and model efficiency, maintaining meaningful variation in all parameters while using far fewer Gaussians.

The results with transformer diffusion model were sub-optimal. I suspect this is to do with how the transformer learns about the spatial variance of the input data. I took inspiration from [6] the Point-E architecture as it seemed to be an elegant solution. However based on testing, there are more nuances that need to be dealt when the input to the network is a pair of 3D Gaussians instead of only point cloud coordinates .

6 Conclusion

The VAE model demonstrates mixed results in learning 3D Gaussian primitive representations. It successfully captures the primary structural components of input shapes and maintains broad parameter coverage (positions: -0.284 to 0.207, scales: 0.001 to 0.100, opacities: 0 to 1.000). This implementation, using a 64-dimensional latent space, achieves parameter range preservation but at a significant computational cost. The VAE maintains the full theoretical range of scale parameters matching the original data distribution.

Comparative analysis reveals distinct trade-offs between approaches. The Transformer demonstrates parameter collapse (fixed scale at 0.001 and opacity at 0.100), representing one extreme of over-compression. In contrast, the VAE shows broad but inefficient parameter distribution, while the VQVAE achieves a better balance, maintaining meaningful parameter ranges (scales: 0.038 to 0.100, opacities: 0.050 to 0.166).
Limitations As this field is still developing, it was a struggle to find good resources, I have released a polished version of the code on Github that can serve as a tutorial.

Learning

The idea behind proposing this project was to get practice training generative models which was a skill bottleneck for my capstone research project. Generating 3D scenes is a relatively new field with the latest SOTA coming out every week. A caveat here is that the codebases of the latest work are not released along with the paper which leads to reproducibility challenges. Through this project, I gained the relevant skills and ability to code parts of newer architectures directly from papers and use it for my specific tasks. To summarize

- I learned how to set up transformer diffusion architectures
- Learned how to create a VAE to generate samples from large amounts of data
- Gained experience submitting jobs to the NYU HPC cluster
- Gained valuable writing practice through this paper

References

[1] Preetum Nakkiran, Arwen Bradley, Hattie Zhou, and Madhu Advani. Step-by-step diffusion: An elementary tutorial, 2024.

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

[4] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017.

[5] Barbara Roessle, Norman Müller, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, Angela Dai, and Matthias Nießner. L3dg: Latent 3d gaussian diffusion. In *SIGGRAPH Asia 2024 Conference Papers*, SA '24, page 1–11. ACM, December 2024.

[6] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts, 2022.