Unconditional Generation of 3D Models: Study on ShapeNet dataset

This report outlines the approach and experiments conducted for the unconditional generation of 3D models using voxel grids. The focus is on choosing the right 3D representation and generative method for the given ShapeNet dataset, which contains ~44k 3D objects as voxel grids.

Please check the readme.md file attached for installation and setup instructions. I have provided 2 best performing models:

1. With only ‘chair’ dataset
2. With chair and airplane objects

Choice of 3D Representation

3D representations vary widely depending on the task at hand, and each has its strengths and weaknesses:

- **Point Clouds:** Are flexible and adaptive, particularly when optimized using gradient descent. However, they lack the concept of solid surfaces, which are crucial for rendering.
- **Meshes:** They effectively model solid surfaces and associated properties, making them efficient for certain tasks. However, they are difficult to optimize as they are inherently non-differentiable.
- **Implicit Representations:** Techniques like neural fields and tri-planes are compact and flexible but often lack explicit geometric bias.

For the ShapeNet dataset, which provides 32x32x32 voxel grids, using voxel grids seems reasonable for our task. However, voxel grids can be inefficient for unconstrained generation tasks. Sparse voxel grids have been used in literature to improve efficiency, such as in the XCube paper, which achieved 1024^3 resolution generation in under 30 seconds using a hierarchical sparse voxel grid.

For this project, I started with a dense voxel grid. If training time or memory becomes a constraint, we could explore sparse representations combined with SparseConv.

Choosing Object Categories

Initial experiments were conducted using all available categories in the dataset, but the inference results were not satisfactory. Upon visual inspection, the "chair" category consistently had better samples than others. Due to limitations in training time and GPU memory, I decided to focus solely on the "chair" category for all experiments.

Including more classes could potentially degrade the performance in an unconditional generation scenario. For better guidance, conditional generation could be more effective.

The dataset includes four categories: chair, airplane, table, and car. I plan to incorporate the airplane category in the final experiments to test if the model can capture diversity across different classes.

Choice of Generative Modeling

Generative modeling in 2D image generation has seen a variety of methods, such as GANs, VAEs, Normalizing Flows, and Diffusion models. Recently, diffusion models have gained popularity, so I chose to begin the experiments with this approach, using a DDIM sampler for its faster sampling speed.

The initial model architecture used was a 3D UNet. If this proves too slow, we might switch to a Latent Diffusion Model to improve efficiency.

Objective Function

The XCube model employs a multi-stage generation process that begins with sampling a latent voxel space, followed by sampling additional attributes like normals. However, the dataset we're using does not contain point clouds or similar attributes, so this approach is not applicable.

Instead, I directly apply diffusion with an MSE/Huber loss. In one of the experiments, I also introduced an 'plane of symmetry' constraint to enhance the model's performance.

Experimental Details

UNet Architecture

The UNet architecture was modified from the one proposed in the DDPM paper to adapt all convolution operations to 3D. Each block in the network contains a ResNet structure that incorporates the timestep embedding, an attention layer, and final downsample or upsample blocks. The upsampling block was modified from using transposed convolution to a combination of nearest interpolation and Conv3d to avoid the "checker-board effect."

The architecture is structured as follows:
1. Downsampling Path (32->2):
   - Each down-sample block operation has (ResNet, ResNet, Attention, and downsample)
   - For `dim_multiples =(1, 2, 4, 8)' channels are increased with multiples.
2. Bottleneck (2x2x2):
   -ResNet block + attention + ResNet block.

3. Upsampling Path (2->32):
   - Each upsample operation doubles the spatial dimensions and halves channels, and has skip connections.

Note: *The DDPM and DDIM formulations are identical during training, differing only during inference. DDIM is used across experiments for inference and evaluation.*

Experiments

Exp 0: Naive Implementation with DDIM Configuration

A config for the noise schedule from a popular image diffusion DDIM setup was used. The initial inference results were poor (for both the full dataset and just the chair). After stripping down and using default parameters, I discovered that `clip_sample` is crucial for numerical stability in our setup. All subsequent experiments use `clip_sample`.

Exp 1: DDPM with 0 Terminal SNR

This experiment showed better denoising results, but there were still issues with higher timestep denoising. I tried fine-tuning the model on just higher timesteps, but it did not help.

(I have created a denoising training visualization script in infer.ipynb, which helped with insight and not rely on just inference results)

Theory suggests that rescaling beta SNR to zero should be used with v-prediction, as pure noise at the end can't be recovered by the epsilon prediction method.

Exp 2: DDPM with Clip Output, Cosine Schedule, and Attention Dim = 64

This experiment did not yield good results on denoising. An initial experiment with attention dim = 32 and zero terminal SNR performed better. I refactored the code to include SNR weight [as suggested by Paper].

Exp 3: DDPM with v-Prediction and Zero Terminal SNR [Velocity prediction as suggest by Paper]

For velocity prediction, the intermediate state is an interpolation between **x_0** (the original sample) and **x_T** (pure noise) depending on the timestep.
For different prediction types:

1. *Noise Prediction:* The model predicts noise and MSE with actual noise
2. *Sample Prediction:* The model directly predicts the sample.
3. *Velocity Prediction:* Computed using the velocity equation.

This setup improved denoising but still faced issues at higher timesteps. Another experiment with 32-dim attention and no SNR weighting was conducted to compare results.

Exp 4: DDPM with Trailing Timespacing - SNR Weighting Loss

This experiment provided the best results: sample clipping, zero terminal SNR, velocity prediction, and no SNR weighting loss. Inference was conducted with 100 steps and trailing timespacing.

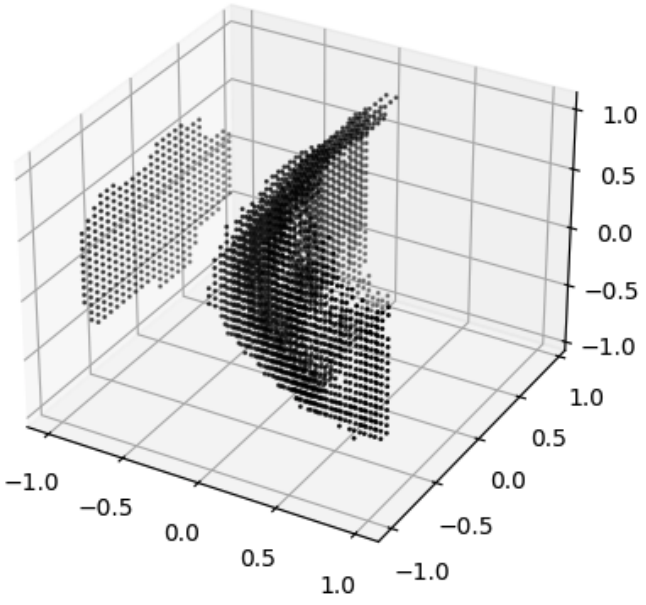
Exp 5: Plane of Symmetry Constraint

Given that the chair dataset predominantly features objects with a canonical pose (facing -z plane), I introduced a constraint to enforce symmetry between the upper and lower halves of the objects. While not all samples strictly follow this symmetry, the goal was to see if it improved overall quality.

This constraint should be applied to the *"denoised prediction of the original sample."* For different prediction types:

4. *Noise Prediction:* The denoised prediction is the current noised version minus the predicted noise.
5. *Sample Prediction:* The model directly predicts the sample.
6. *Velocity Prediction:* Computed using the velocity equation.

On visual inspection, I found consistent sampling results.

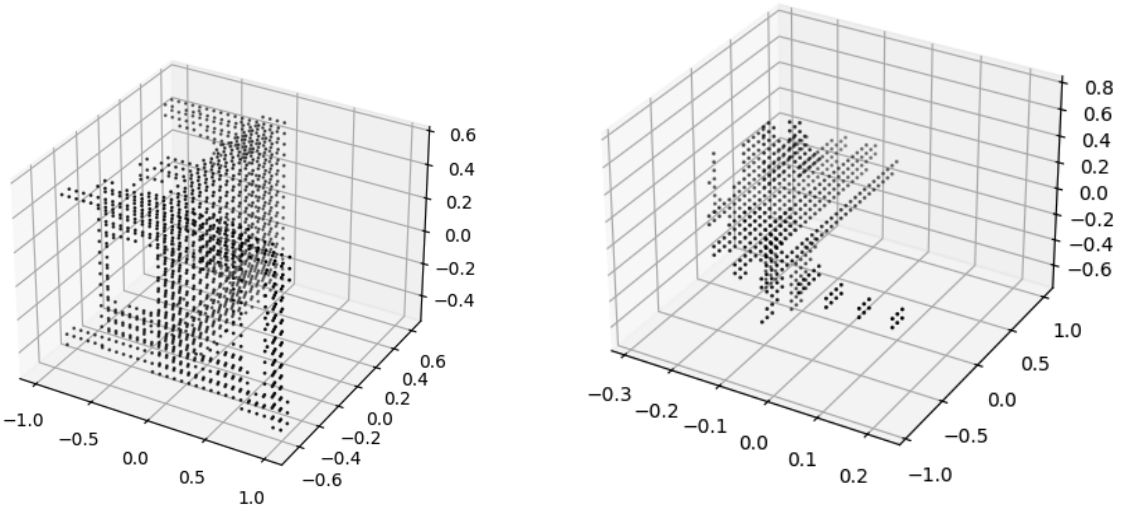


Exp 6: Chair + Airplane for Capturing Diversity

To test the model's ability to capture diversity, I incorporated the airplane category into the experiments. Fortunately, all object categories have the same canonical orientation, so the plane of symmetry constraint remains valid.

Contrary to my assumption,this proved to improve results in terms of MSE loss and visual inspection, where model was able to generate distinctive samples from both categories.

Some samples are included in the notebook provided.



Due to time + compute constraints, I am not exploring further expanding categories, but the effect of adding classes for unconditional generation performance seems a promising exploration.

Future Work

1. **Objective function:** MSE/ Huber loss reconstruction error was insufficient to judge model performance and compare, I had to rely on visual inspection. I believe a better objective founded in geometric constraints (plane of symmetry, a constraint on parts in objects, surface normals etc) is crucial. I explored the plane of symmetry due to canonical poses in train set, which can further be improved by only constraining the object (voxel value > 0) and not empty space.
2. **Evaluation Criteria:** I briefly checked the literature for quantitative evaluation criteria, but as different 3D representations were employed, I found it much easier just to rely on visual inspection. This needs to be further studied.
3. **Miscellaneous:** Further explore the representation and diffusion process itself, e.g.
    a. hierarchical latent diffusion like XCube,
    b. Progressive distillation, Score-based Generative Modeling, SDEs
    c. I couldn't implement the EMA model due to time constraints, which is known to give stable results.