Implementation of TransGAN

Aditya Kumar [1] Vatsalraj Rathod [1] Vedanshi Raiyani [1]

[1]Department of Computer Science and Engineering, Indian Institute of Technology, Gandhinagar

Introduction

Point clouds are widely used to represent objects and environments in fields like robotics, AR/VR, and autonomous vehicles. Techniques such as PointNet, TreeGAN, and PointFlow have greatly improved tasks like point cloud classification, segmentation, and generation. Transformer-based models, like TransGAN, have shown great potential in 2D image generation, inspiring exploration into their use for 3D data.

In this project, we implemented TransGAN to generate 3D point clouds. The model uses Transformers to create realistic 3D shapes and addresses challenges in producing high-quality point clouds. While the training was limited by time and dataset size, the results show that TransGAN is effective for generating point clouds. This work lays the foundation for further improvements and broader applications in 3D modeling.

Existing Work

- 3D point cloud processing initially relied on traditional geometric methods like **Iterative Closest Point (ICP)** for tasks such as alignment and manipulation.
- Over time, deep learning transformed the field, enabling direct processing of raw point clouds and improving performance on complex tasks.
- **PointNet** and **PointNet++** were pivotal in extracting features for tasks like classification and segmentation, marking the start of deep learning-based point cloud processing.
- Generative models like **TreeGAN** and **PointFlow** advanced 3D shape generation by leveraging hierarchical structures and probabilistic frameworks, respectively.
- Recent approaches, such as Transformer-based models like TransGAN, offer potential for handling complex dependencies in 3D data, further refining reconstruction and synthesis tasks.

Table 1. Result Comparison between PointFlow and TreeGCN-ED

Metric	PointFlow	TreeGCN-ED
Chamfer Distance (CD)	1.74 (Chair, ShapeNet)	1.21 (Average, ShapeNet)
Frechet Point Cloud Distance (FPD)	Not reported	Lower than FoldingNet
ModelNet Classification Accuracy	93.7% (ModelNet10)	85% (ModelNet10)
	86.8% (ModelNet40)	73% (ModelNet40)
Point Cloud Generation Diversity	60% (1-NNA, lower is better)	Not available
Number of Parameters	1.06M	Not explicitly reported
Applications	Generation, Reconstruction	Completion, Reconstruction

Our Work: Transformer-Based GAN for Point Cloud Generation

- In this project, we researched various methods for 3D point cloud generation and explored the potential of adapting TransGAN for this task.
- TransGAN, originally designed for 2D image generation, was studied and modified to process 3D data, requiring: The design capable of handling the unique structure and sparsity of point clouds.
- The goal was to generate high-quality, detailed 3D point clouds by training on the ModelNet10 dataset, focusing on creating realistic object shapes.
- The Transformer-based architecture was chosen for its ability to model long-range dependencies and global context, which are crucial for capturing fine-grained details in 3D shapes.

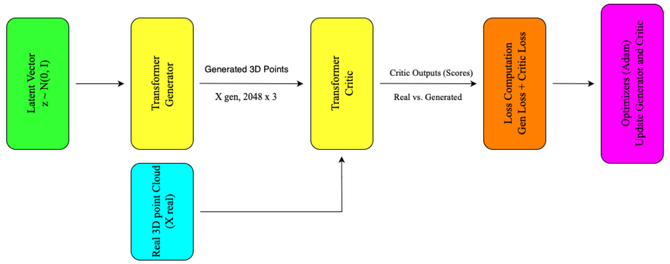

Figure 1. TransGAN Architecture

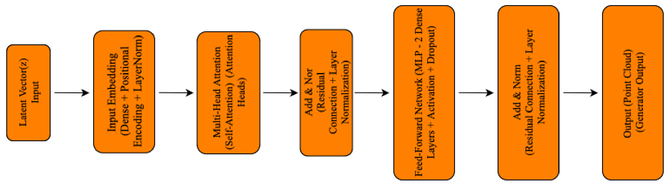

Figure 2. Transformer Architecture

Mathematical Framework for TransGAN

1. Generator and Critic: The generator G maps a latent vector $\mathbf{z} \in \mathbb{R}^d$ to a 3D point cloud:

$$\mathbf{x}_{\text{gen}} = G(\mathbf{z}; \theta_G),$$

while the critic D evaluates the realism of point clouds:

$$D(\mathbf{x}; \theta_D).$$

2. Loss Functions:
Generator Loss:

$$\mathcal{L}_G = -\mathbb{E}_{\mathbf{z} \sim P_{\mathbf{z}}}[D(G(\mathbf{z}))].$$

Critic Loss with Gradient Penalty:

$$\mathcal{L}_D = \mathbb{E}_{\mathbf{x}_{\text{gen}}}[D(\mathbf{x}_{\text{gen}})] - \mathbb{E}_{\mathbf{x}_{\text{real}}}[D(\mathbf{x}_{\text{real}})] + \lambda \mathcal{L}_{\text{GP}},$$

where

$$\mathcal{L}_{\text{GP}} = \mathbb{E}_{\hat{\mathbf{x}}}\left[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2\right].$$

Interpolation:

$$\hat{\mathbf{x}} = \epsilon \mathbf{x}_{\text{real}} + (1 - \epsilon)\mathbf{x}_{\text{gen}}, \quad \epsilon \sim \text{Uniform}(0, 1).$$

3. Training Updates:
Generator Update:

$$\theta_G \leftarrow \theta_G - \eta \nabla_{\theta_G} \mathcal{L}_G.$$

Critic Update:

$$\theta_D \leftarrow \theta_D - \eta \nabla_{\theta_D} \mathcal{L}_D.$$

Results

- We ran and analyzed the TreeGAN model, expecting to achieve high-quality 3D point clouds similar to its results for realistic object shapes.
- The actual results did not fully match expectations, as we were unable to generate clear, well-defined shapes in the 3D point clouds.

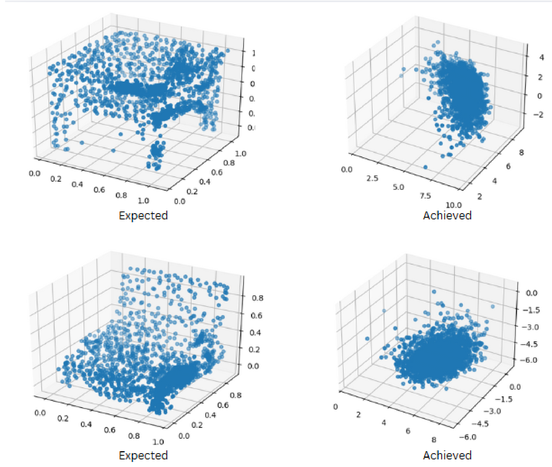

Figure 3. results

Future Prospects

- Performing a detailed analysis of the current model's performance to identify areas for improvement.
- Fixing the code by addressing any bugs or inefficiencies that may have affected the results.
- Optimizing parameter values (e.g., learning rate, batch size) to enhance model performance and stability.
- Running the model for more epochs to allow for better learning and more accurate results.
- Feeding the model with a larger dataset to help the model generalize better and capture finer details in the generated 3D shapes.

References

[1] VITA Group. (n.d.). TransGAN. GitHub repository.

[2] 3D Point Cloud Generative Adversarial Network. (n.d.). Papers with Code.

[3] Cui, R. (2021). Partial2Complete: Transformer Model Implementation. GitHub repository.

[4] Xucheng, V. (2020). PointStyleGAN: A Generative Adversarial Network for 3D Point Clouds. GitHub repository.

[5] Sato, H., Takahashi, M. (2023). 3D Point Cloud Generation with Transformers. arXiv:2303.16450.