Synthesizing Dense and Colored 3D Point Clouds for Training Deep Neural Networks

Mohammad Samiul Arshad and William J. Beksi

Department of Computer Science and Engineering, University of Texas at Arlington

The Goal

Generate dense and colored 3D point clouds for multiple classes of objects in an unsupervised manner.

Introduction

Generating synthetic 3D point cloud data is an open area of research with the intention of facilitating the learning of non-Euclidean point representations. Currently, synthetic data aids the solution of 3D computer vision tasks such as classification, segmentation, and reconstruction. Researchers make use of point clouds sampled from the mesh of manually designed objects as data for training deep learning models. However, the geometry and texture of these point clouds is bounded by the resolution of the modeled objects. Moreover, due to the complexity of the design process, the number of composed objects can fail to satisfy the enormous data needs of deep learning research. Automatically synthesizing point clouds can solve this problem by providing a source of potentially infinite amounts of diverse data.

Background

Wasserstein Distance: Given two distributions P_r and P_g in a metric space $\mathcal{M}$, the Wasserstein distance calculates the minimal cost to transform P_r into P_g or vice-versa. A distance of order p can be expressed as

$$W_p(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\|x - y\|\big],$$

where $\Pi(P_r, P_g)$ is the set of all joint distributions $\gamma(x, y)$ with marginals P_r and P_g.

Generative Adversarial Network (GAN): A GAN is a special type of neural network that focuses on learning the underlying distribution P_r of a dataset in order to generate new samples. The objective of a GAN can be written as

$$\min_G \max_D (G, D) = \mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{z\sim P_g}[\log(1 - D(G(z)))],$$

where G and D denote the generator and discriminator, respectively.

Wasserstein GAN (WGAN): To improve the convergence of a GAN, a WGAN minimizes the distance between the real data distribution P_r and the generated data distribution P_g using the dual form of the Wasserstein distance with a GAN objective,

$$\min_G \max_{D\in\mathcal{D}} (G, D) = \mathbb{E}_{x\sim P_r}[D(x)] - \mathbb{E}_{z\sim P_g}[D(G(z))],$$

where $\mathcal{D}$ is the set of 1-Lipschitz functions achieved through weight clipping.

WGAN with Gradient Penalty: To improve the stability of a WGAN, a WGAN-GP constrains the norm of the gradient to be at most 1 by applying a penalty term to the gradient of the discriminator,

$$\min_G \max_{D\in\mathcal{D}} (G, D) = \mathbb{E}_{x\sim P_r}[D(x)] - \mathbb{E}_{z\sim P_g}[D(G(z))] + \lambda\, \mathbb{E}_{\tilde{x}\sim P_{\tilde{x}}}\big[(\|\nabla_{\tilde{x}} D(\tilde{x})\|_2 - 1)^2\big],$$

where $\tilde{x} \sim P_{\tilde{x}}$ are the points uniformly sampled along the straight line between pairs of points from P_r and P_g.

Architecture

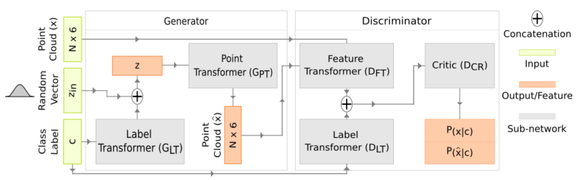

Figure 1: The PCGAN architecture.

Our progressive conditional generative adversarial network (PCGAN) architecture consists of a generator and discriminator which are optimized via the gradient penalty techniques of WGAN-GP, Figure 1. For multiclass generation both the generator and discriminator are conditioned on a randomly chosen class label, $c \in C$, which is represented as one hot vector and presented as input to both networks.

Generator: Given a random vector $z_{in} \in \mathcal{N}(0, 1)$ and class label c, the generator aims to create realistic point clouds of class c. The label transformer (G_{LT}) sub-network, converts c into a vector that is concatenated to z_{in} to produce a point vector z. The generator uses a tree-structured graph convolutional network in the point transformer (G_{PT}) sub-network that transforms z into point clouds with color.

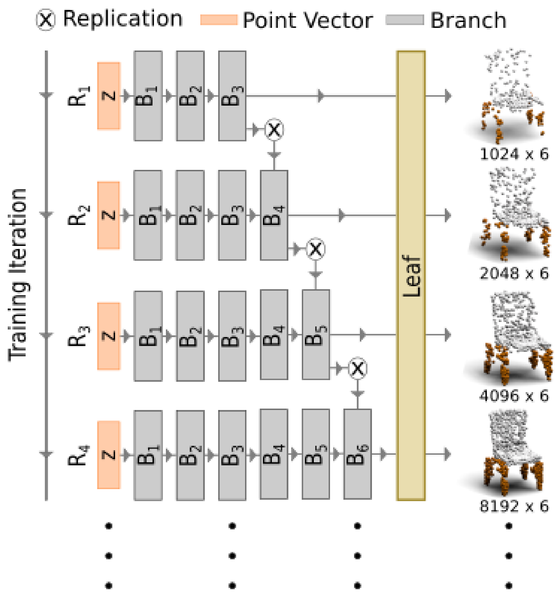

Figure 2: The progressive growing of the point transformer.

To reduce the complexity of the generation process, the generator is trained in a progressive manner. PCGAN first learns to generate low-resolution point clouds and after a fixed number of iterations the size of the network is increased via replication of the last branch, Figure 2.

Discriminator: Given a class label c, real point cloud x, and generated point cloud $\hat{x}$, the discriminator tries to identify between the real and synthesized point cloud of c. Through the feature transformer (D_{FT}) sub-network, the point clouds are converted into a vector and concatenated to a class vector produced by the label transformer (D_{LT}). The concatenated vector is given to the critic (D_{CR}) sub-network as input. D_{CR} predicts the probability of its input being real or generated. D_{FT} incorporates dynamic graph convolutions to collect local and global features from point clouds.

Generated Samples

PCGAN learns the basic structure of an object in low resolutions and gradually builds up to higher resolutions. The relationship between the object parts and their colors (e.g., the legs of the chair/table are the same color while seat/top are dissimilar) is also

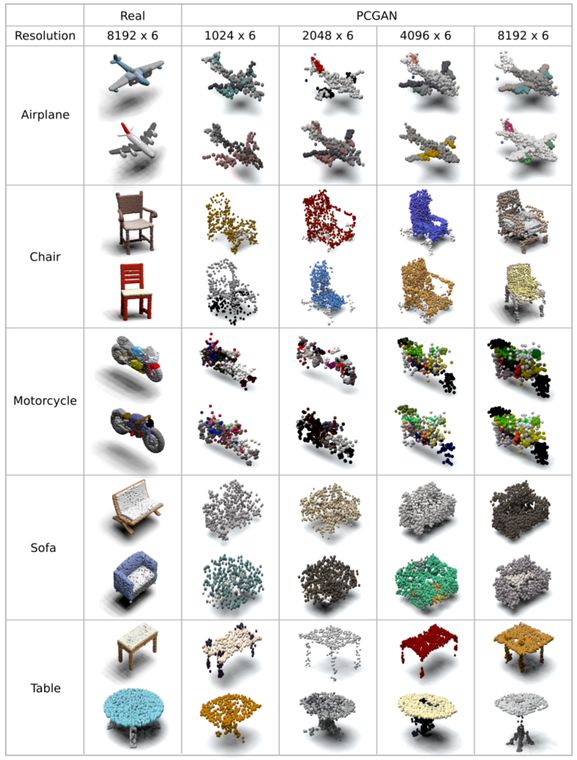

Figure 3: Examples of 3D point clouds synthesized by PCGAN.

learned by the network, Figure 3. We use the ShapeNetCore CAD model dataset for training our network to create synthetic point clouds. PCGAN was trained on TACC's Maverick2 resource. With approximately 20,000 point clouds from five classes, PCGAN takes roughly 15 minutes per iteration on four Nvidia GTX 1080 GPUs to generate point clouds $\hat{x} \in \mathbb{R}^{1024\times 6}$.

Acknowledgements

The authors acknowledge TACC at the University of Texas at Austin for providing software, computational, and storage resources that have contributed to the research results reported within this project.

UNIVERSITY OF TEXAS ARLINGTON