20/4/2020

Learning Implicit Fields for Generative Shape Modeling

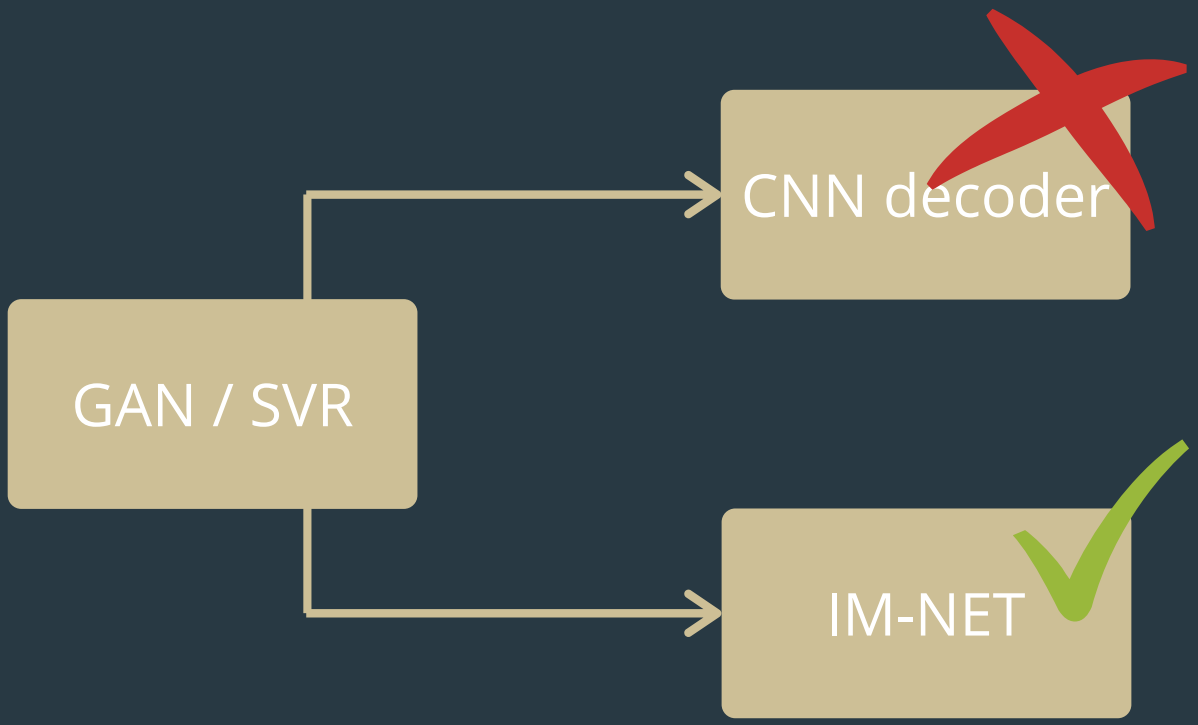
Team 12: Super Phantom CV Hyperforce Go

Problem Statement

- Typical state-of-the-art architectures for 3D generative shape modelling using standard CNNs and GANs are poor in terms of visual quality.
- They exhibit discontinuous and overly smoothed surfaces, provide low resolution outputs and are susceptible to irregularities in training data.
- CNNs learn voxel distributions over a volume, rather than the shape boundary itself

The Solution

- Implicit Fields used in Decoder feeds the point coordinates along with shape feature vector to determine whether a certain point lies on the inside or the outside, relative to the shape.
- The method allows to learn shape boundaries and output at multiple resolutions, irrespective of the resolution of the training data.
- This shape aware network produces shapes of higher visual quality on interpolation through latent GANs



Our Next Steps:

This is the procedure we followed while implementing and understanding the project.

1 Understanding the Paper

Knowledge of network structures, definitions, and purpose.

2 Data Gathering and Prep

Preparing input to AE and GAN. Sampling from HSP dataset

3 Implementation of AE

Implementation of the CNN Auto-encoder

4 Implementation of Decoder

IM-NET decoder unique to the paper is implemented

5 Implementation of GAN

GAN to demonstrate generation of models based on input models.

6 Comparison of Results

Qualitatively compare results of our model with implementations and outputs of other models.

Scope of Work

- Our implicit field decoder, IM-NET, can be embedded into different shape analysis and synthesis frameworks to support various applications.
- For our project, we demonstrate auto-encoding and generation of 3D objects or shape generation.

- For auto-encoding 3D shapes, we used a 3D CNN as encoder to extract 128-dimensional features from 64x64x64 voxel models.
- For 3D shape generation, we employed latent-GANs on feature vectors learned by a 3D autoencoder.

Major Definitions

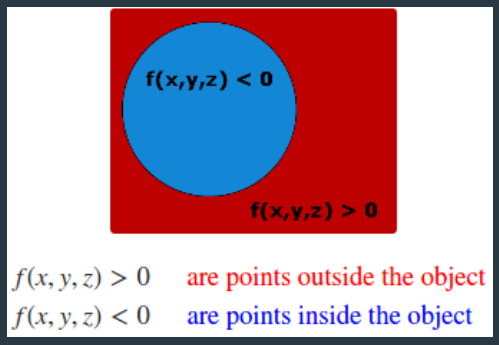
A look through of definitions of terms used in the report.

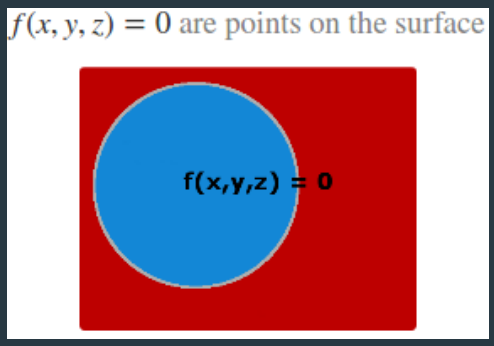
voxel

A pixel is a 2-dimensional Raster graphic, therefore having the values of width and length, with colour placed inside the coordinate. A voxel is a raster graphic on a 3-dimensional grid, with the values of length, width and depth. It also contains multiple scalar values such as opacity, color and density.

implicit surface

Implicit surfaces are actually explicit volumes they explicitly define what is the inside and outside of the object.
Consider the function which takes a 3D point (x,y,z) as input and returns a single value. This value tells us explicitly if we are outside or inside the volume.
However, **explicit surfaces** directly extract points on the surface. Eg: A function takes in input parameters (u,v) and returns a point (x,y,z) directly on the surface.



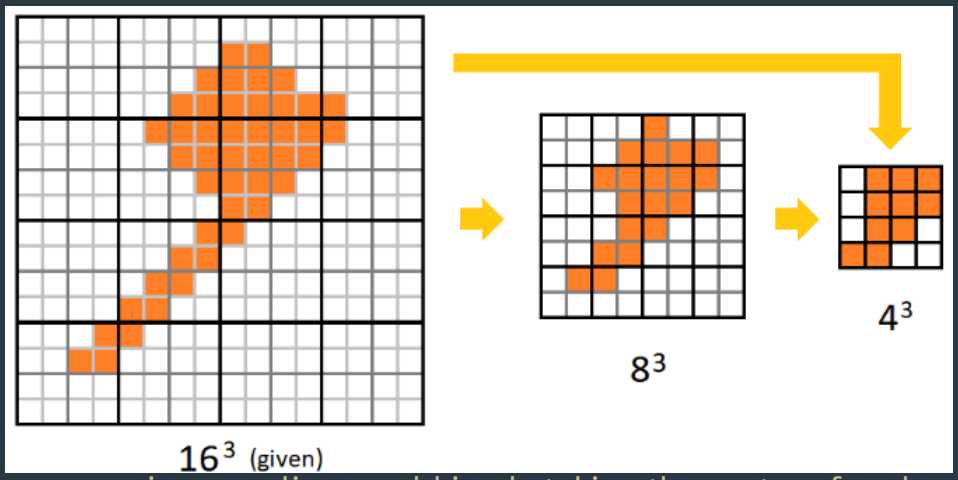


iso-surface

An isosurface is a three-dimensional surface that represents points of a constant value within a volume of space. Thus for our purpose, the object we need to construct has a zero ISO-surface.
A mesh surface can be reconstructed by finding the zero-isosurface of the implicit field.

Data Preparation

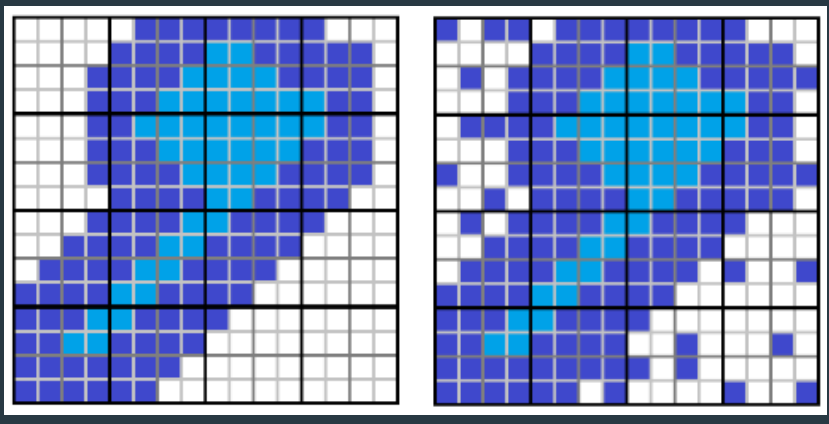
- We need a point value cloud for the training of our implicit decoder.
- For 3D shapes, to get voxel models in different resolutions (16x16x16 to 128x128x128), we sample points on each resolution in order to train the model progressively.



- However, a naive sampling would imply taking the center of each voxel and thus produce n points in each dimension ie, nxnxn points.
- We aim to get nxn points and thus sample more points closer to the shape surfaces and neglect points far away.
- To compensate for a density change, we assigned weights of all sampled points to 1, because we want the model to pay more attention to the surface and allow small errors in the void area.

Voxel grid resolution	16^3	32^3	64^3	128^3
Number of points	16^3	$16^3 \times 2$	32^3	$32^3 \times 4$

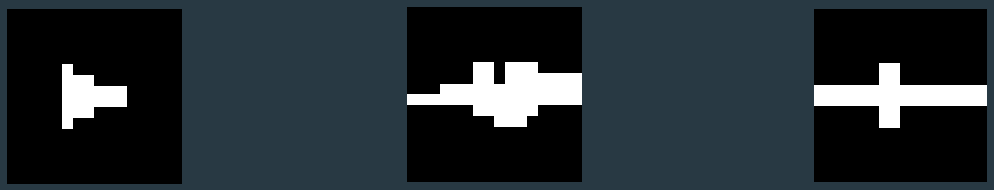
- Sample points which are within 3 voxel (in all x, y, z directions) from shape boundaries are taken. If the number of sampled points does not exceed the limit, randomly sample more points up to a limit.



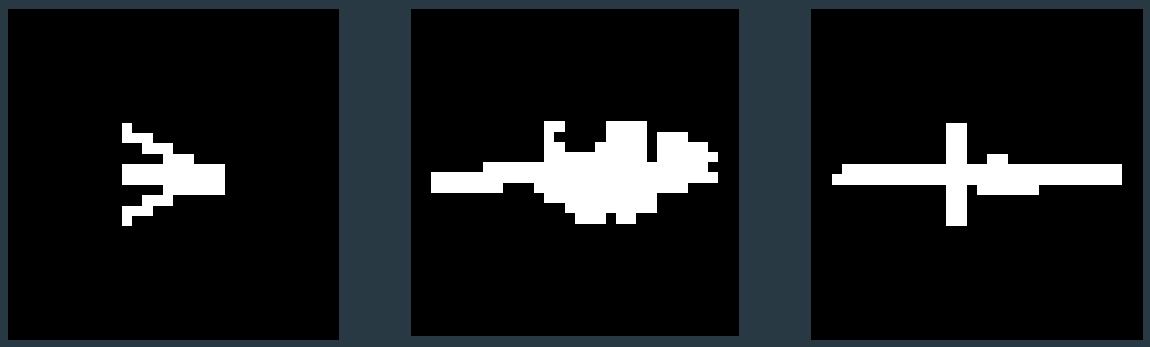
Dataset Sample

A rifle dataset image sample at different resolutions and visualized at 3 axes

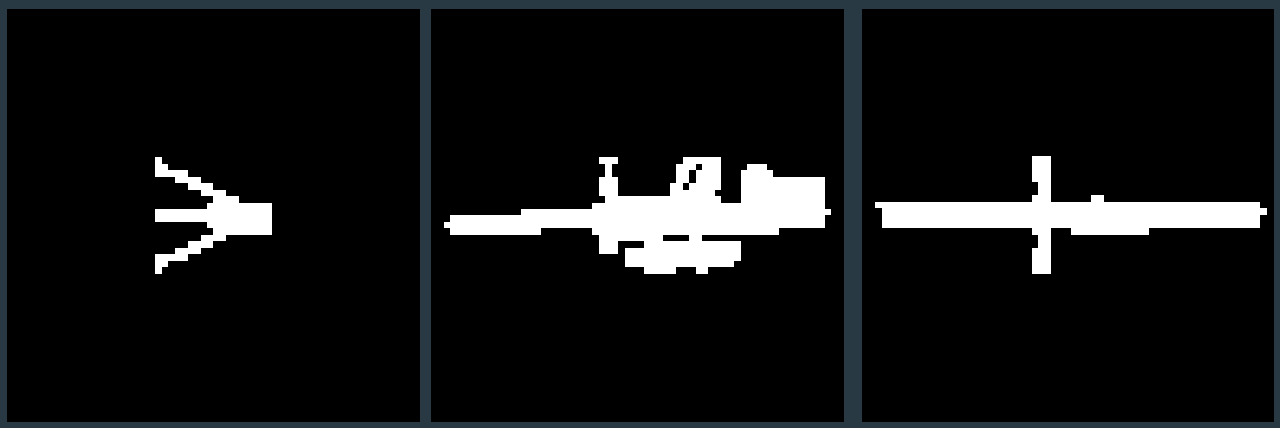
Res: 16



Res: 32



Res: 64



Decoder: IM-NET

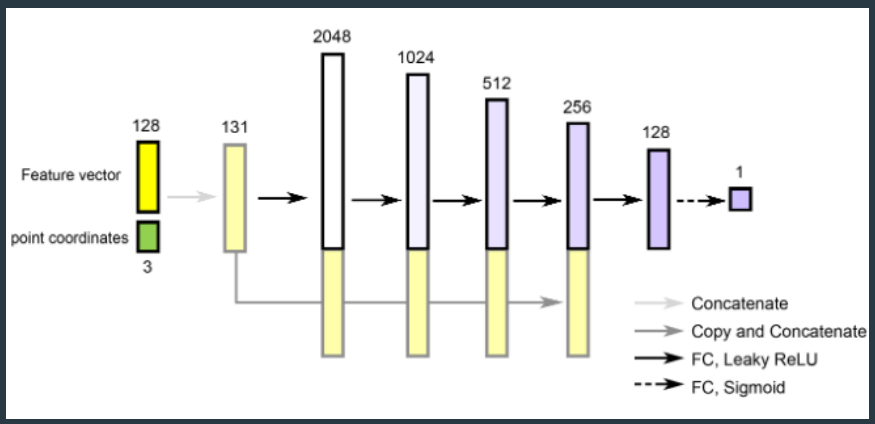
Used instead of CNN-decoder

The network takes as input a feature vector extracted by a shape encoder, as well as a 3D or 2D point coordinate, and it returns a value indicating the inside/outside status of the point relative to the shape.

The loss function is a weighted mean squared error between ground truth labels and predicted labels of each point. Taking S as sampled set of points:

$$\mathcal{L}(\theta) = \frac{\sum_{p \in S} |f_\theta(p) - \mathcal{F}(p)|^2 \cdot w_p}{\sum_{p \in S} w_p}$$

- In a typical application setup, our decoder, which is coined IM-NET, would follow an encoder which outputs the shape feature vectors and then return an implicit field to define an output shape.
- The skip connections (copy and concatenate) in the model can make the learning progress faster in the experiments. They can be removed when the feature vector is too long, so as to prevent the model from becoming too large.
- The implicit field decoder, IM-NET, can be embedded into different shape analysis and synthesis frameworks to support various applications.



Encoder:

- For auto-encoding 3D shapes, we used a 3D CNN as encoder to extract 128-dimensional features from 64x64x64 voxel models. We adopt progressive training techniques, to first train the model on 16x16x16 resolution data, then increase the resolution gradually.
- **Note:** Higher resolution models can be trained with pre-trained weights on low-res data. In experiments, progressive training can stabilize training process and reduce training time significantly.

IM-NET: Novel Features

Properties which impact the visual quality of generated shapes.

- **Resolution:** Decoder output can be samples at any resolution and thus not hindered by the resolution of the training samples. In our case, we sampled the output shapes at 512x512x512 or 256x256x256 resolution and rendered after Marching Cubes.

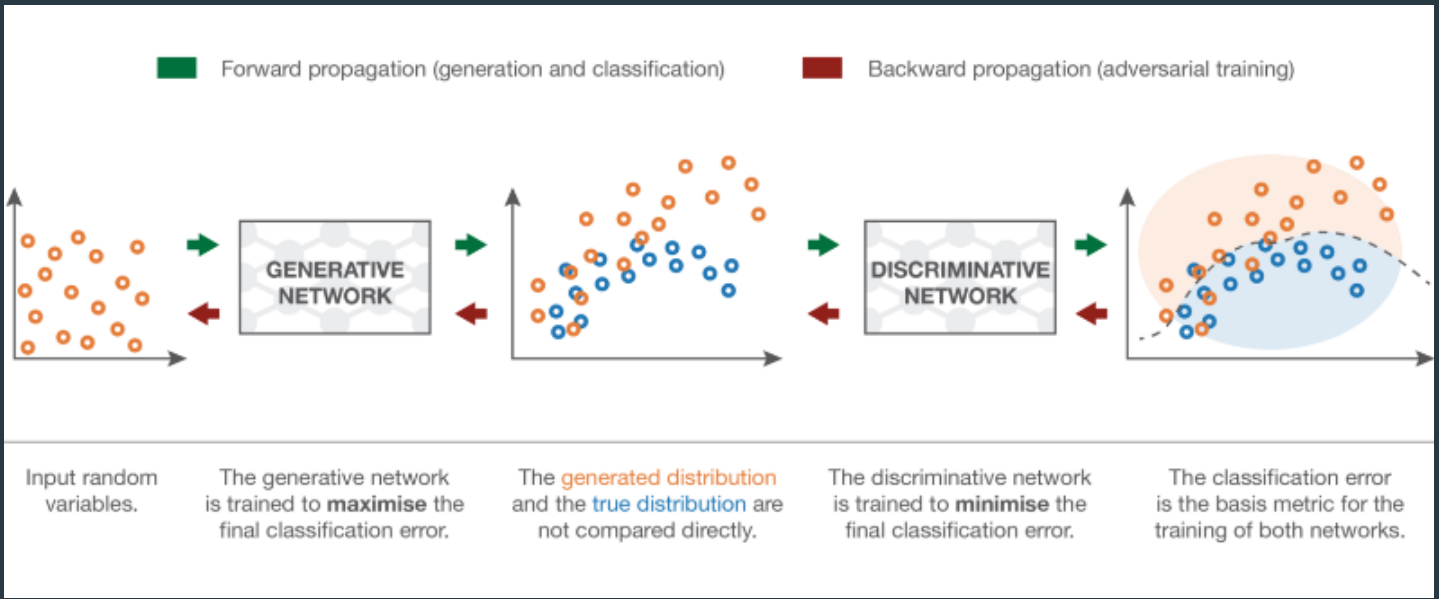
- **Properties Learned:** Since point coordinates are also concatenated with 128-feature vector, the network learns the inside/outside status of any point relative to a shape. In contrast, a CNN network predicts possibility of each pixel to be on/off relative to extent of bounding volume of a shape.

- **GANs:** IM-NET learns *shape boundaries* while CNN learns *voxel distributions* over a volume. CNN computes voxels as weighted averages, where the kernel windows are not "shape-aware." Thus, IM-NET is particularly useful in case of GANs as shape evolution is a direct result of changing the assignments of point coordinates to their inside/outside status.

- **Continuity:** The perception of continuity for CNN decoders is the lightness change in each pixel and therefore when object with sharp boundaries are used for training the WGAN-CNN, the generated results are not continuous and broken. Blurring the dataset to imply continuity is usually used. However, WGAN-IM can be trained well for sharp and blurred versions, as lightness changes and shape movements are continuous.

GANs

The problem of generating a new image of an object can be rephrased into a problem of generating a random vector in the N dimensional vector space that follows the "targeted object probability distribution"



The downstream task of GANs is a discrimination task between true and generated samples. Or we could say a "non-discrimination" task as we want the discrimination to fail as much as possible. So, in a GAN architecture, we have a discriminator, that takes samples of true and generated data and that try to classify them as well as possible, and a generator that is trained to fool the discriminator as much as possible.

Once defined, the two networks can then be trained jointly (at the same time) with opposite goals:

- the goal of the generator is to fool the discriminator, so the generative neural network is trained to maximise the final classification error (between true and generated data). It takes as input a simple random variable and must return, once trained, a random variable that follows the targeted distribution.
- the goal of the discriminator is to detect fake generated data, so the discriminative neural network is trained to minimise the final classification error. It takes as input a point (ie, a N dimensional vector) and returns as output the probability of this point to be a "true" one.

IM-GAN

METRIC	DESCRIPTION	DISADV
KL DIVERGENCE	**D-**KL achieves the minimum zero when p(x) == q(x) everywhere.	Asymmetric
JENSON-SHANNON	Bounded by [0,1], Symmetric, More smooth	
WASSERSTEIN Measure of distance between 2 probability distributions	Clamps weights after gradient update to [-c,c], no logs calculated	Unstable training, size of clipping window causes problems

When using the shape feature vectors from IM-NET to train a latent GAN (one that is trained on encoded vectors), we create the IM-GAN.

For 3D shape generation, we employed **latent-GANs** on feature vectors learned by a 3D auto-encoder. We did not apply traditional GANs trained on voxel grids since the training set is considerably smaller compared to the size of the output. Therefore, the pre-trained AE would serve as a means for **dimensionality reduction**, and the latent-GAN was trained on high-level features of the original shapes. We used two hidden fully-connected layers for both the generator and the discriminator, and the **Wasserstein GAN loss** with gradient penalty.

WGAN:

Instead of using a discriminator to classify or predict the probability of generated images as being real or fake, the WGAN changes or replaces the discriminator model with a critic that scores the realness or fakeness of a given image. Disadvantages of WGAN can be overcome to a certain extent by using gradient penalty to the loss function, converges faster, stable score.

Structure

We assess and evaluate improvements made by the implicit decoder for generative modeling of 3D shapes. We trained latent-GANs on both CNN-AE and IM-AE to ob-tain CNN-GAN and IM-GAN.

The paper also details results obtained when compared with state-of-the-art methods: PCGAN and 3DGAN. PCGAN uses generative point cloud methods.

CNN-GAN and IM-GAN are using the same latent-GAN structure.

Generator:

Layer	*Activation function*	*Output shape*
latent vector	-	(128)
fully-connected	LReLU	(2048)
fully-connected	LReLU	(2048)
fully-connected	Sigmoid	(128)

Discriminator:

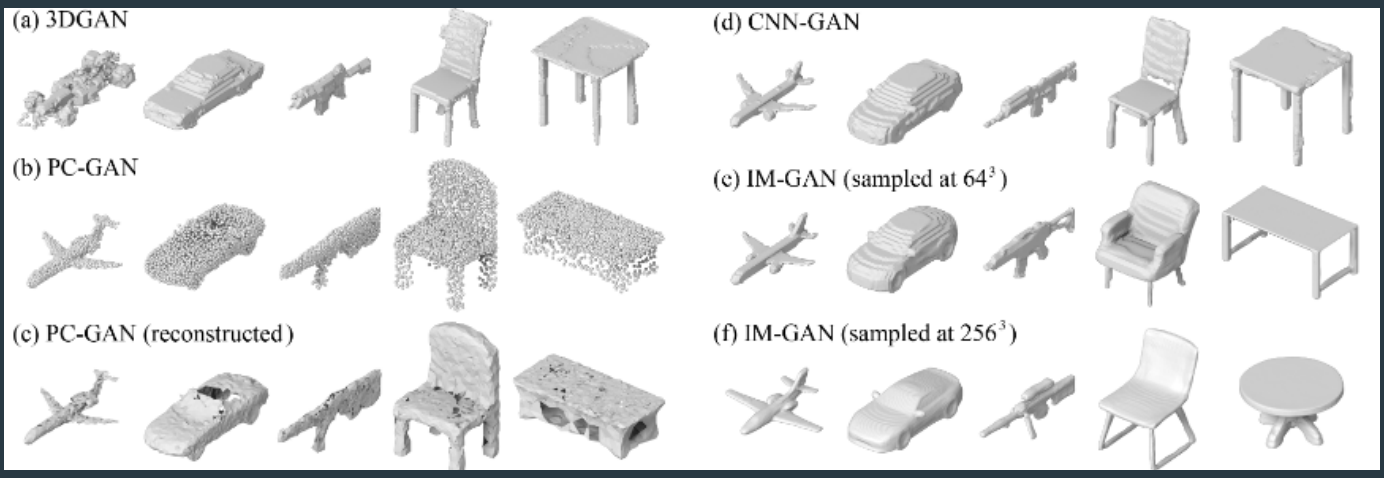
Layer	*Activation function*	*Output shape*
feature code	-	(128)
fully-connected	LReLU	(2048)
fully-connected	LReLU	(2048)
fully-connected	-	(1)

TRAINING CONFIGURATION

- The networks were implemented with Tensorflow and using Adam optimizer (learning_rate=5e-5, beta1=0.5, beta2=0.999, epsilon=1e-8).
- For leaky ReLU, alpha=0.02.
- For batch normalization, decay=0.999, epsilon=1e-5.
- The training batch size is: 32 for 3D CNN-based models, 1 for implicit-decoder-based models, 50 for latent-GANs.
- Notice that for implicit-decoder-based models, the batch size is one shape, the actual batch size for implicit decoder varies according to the resolution of the training data.

Results

IM-GAN generates shapes with better visual quality compared to other methods, in particular, with smoother and more coherent surfaces. 3DGAN appears to suffer from mode collapse on several categories, leading to lower coverage. Point clouds generated by PC-GAN are recognizable but lack detailed features; high-quality reconstruction from only 2048 generated points would be challenging. In addition IM-GAN exhibits superior capability in 3D shape interpolation.



Quantitative results

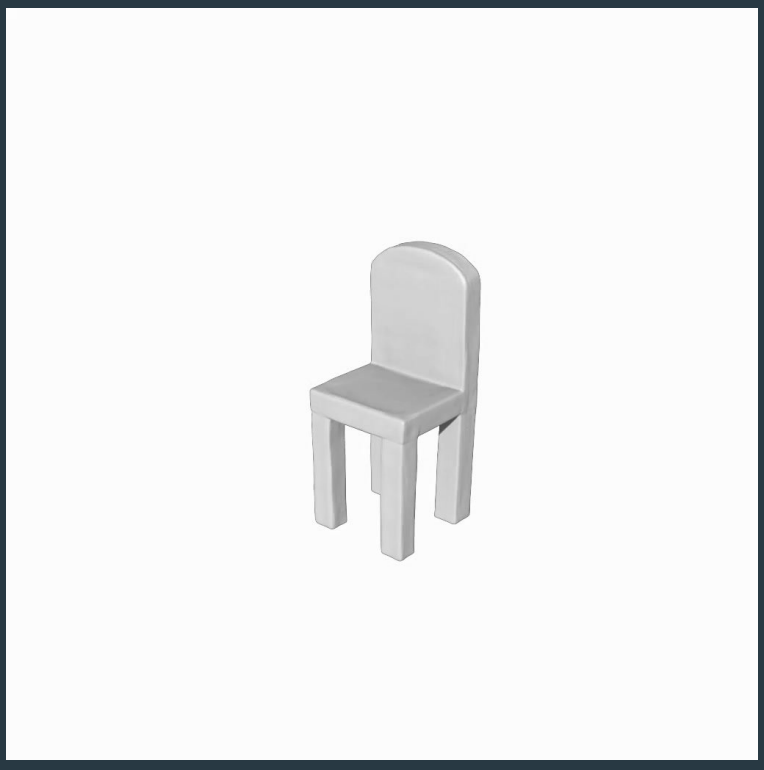
- Light Field Descriptor captures radiance at a 3D point along a 2D direction. The Light Field Descriptor (LFD) of a 3D shape is a set of 2D images of it, taken from a 2D array of cameras
  - 20 cameras positioned at the vertices of a regular dodecahedron.
- Suppose that we have a testing set G and a sample set A, for each shape in A, we find its closest neighbor in G using LFD, say g, and mark g as “matched”. In the end, we calculate the percentage of G marked as “matched” to obtain the coverage score (COV-LFD) that roughly represents the diversity of the generated shapes.
- However, a random set may have a high coverage, since matched shapes need not be close. Therefore, we match every shape in G to the one in A with the minimum distance and compute the mean distances in the matching as Minimum Matching Distance(MMD-LFD). Ideally, a good generative model would have higher COV-LFD and lower MMD-LFD values.

Results (IM-GAN)

Some results of chair and rifle dataset after running IM-GAN and rendering at 256x256x256 resolution with marching cubes algo.

Results (IM-GAN)

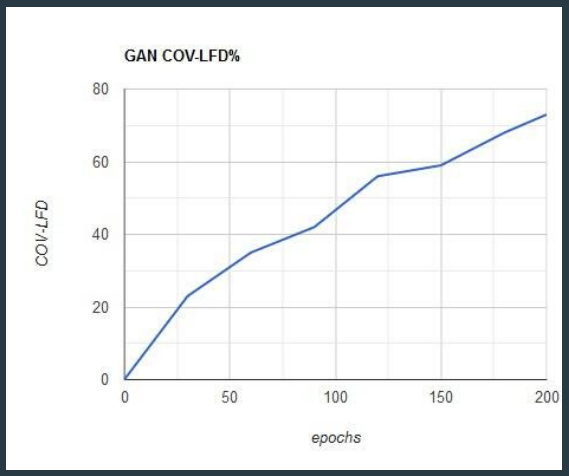
Video Renderings are available on our github.



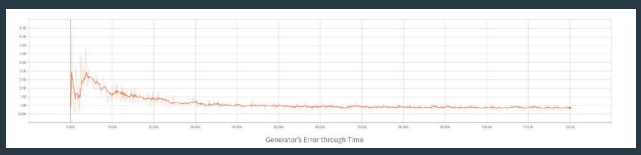
COV-LFD	Plane	Car	Chair	Rifle	Table
CNN-GAN	69.22	**73.00**	**77.73**	61.26	83.73
IM-GAN	70.33	69.33	75.44	**65.26**	**86.43**

MMD-LFD	Plane	Car	Chair	Rifle	Table
CNN-GAN	3,745	1,288	3,012	3,819	2,594
IM-GAN	**3,689**	**1,287**	**2,893**	**3,760**	**2,527**

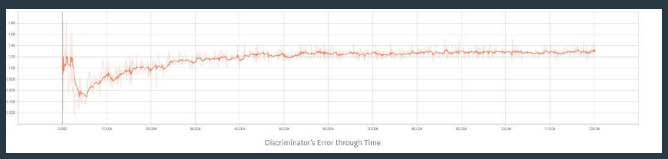
Coverage does not indicate exactly how well the covered examples (point-clouds) are represented in set A; matched examples need not be close. We need a way to measure the fidelity of A with respect to B. To this end, we match every point cloud of B to the one in A with the minimum distance (MMD) and report the average of distances in the matching.



Generator: Error with Time



Discrim.: Error with Time



Drawbacks

Also detailing further optimizations that can be utilized.

- **Very long training time:** CNN-AE is much faster, even in the case of progressive training.
- **Dissimilar Shapes:** Cannot ensure a meaningful morph between highly dissimilar shapes ie, from different ccategories.
- **Low Frequency Errors:** Like global thinning and thickening of meshes.