# Construction of 3D models from 2D images

*A THESIS*

*submitted by*

**SMRUTI PATRO**

*for the award of the degree*

*of*

**DUAL DEGREE (B.TECH. & M.TECH.)**


**DEPARTMENT OF ELECTRICAL ENGINEERING**
**INDIAN INSTITUTE OF TECHNOLOGY MADRAS.**

**JUNE 2021**

# THESIS CERTIFICATE

This is to certify that the thesis titled **Construction of 3D models from 2D images**, submitted by **Smruti Patro**, to the Indian Institute of Technology, Madras, for the award of the degree of **Dual Degree (B.Tech. & M.Tech.)**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

**Prof. Anurag Mittal**
Research Guide
Professor
Dept. of Computer Science and Engineering
IIT Madras, 600 036

**Prof. Kaushik Mitra**
Department Co-Guide
Assistant Professor
Dept. of Electrical Engineering
IIT Madras, 600 036

Place: Chennai

Date: June 2021

# ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my project guide **Prof. Anurag Mittal** for providing invaluable guidance, comments and suggestions throughout the course of the project. I would like to thank **Prof. Kaushik Mitra** for agreeing to be my department co-guide for this project. I would like to thank my parents and friends for the unceasing encouragement and support. I would also like to offer my sincere thanks to all other persons who knowingly or unknowingly helped me in completing this project.

# ABSTRACT

KEYWORDS: 3D models; GANs; 3D-GAN; 3D-IWGAN

This project aims at constructing 3D models from their corresponding 2D images. We propose a generative model for construction of the 3D models by leveraging the advances in the development and stable training of generative adversarial nets(GANs). Since it is well known that training GANs is elusive, we propose the use of a 3D-IWDCGAN(3D- Improved Wasserstein Deep Convolutional Generative Adversarial Nets) with the objective to attain stable training for complex data distributions. Considering DCGANs have become the de facto standard for stable GAN training, our generator network has deep convolutional layers instead of the fully connected layers as in traditional GANs. The 3D-DCGAN architecture maps each embedding it has seen in the latent space to a 3D voxel grid. The remainder of this latent space can also generate novel 3D objects. GAN training may not converge especially when they are trained to learn complex data distributions. So, we use the Wasserstein training objective, which has shown promising results in the past for GAN training. For a single-viewpoint image, the ground-truth shape is ambiguous, i.e. there can be multiple shapes that fit the given 2D image quite well. We make use of an additional 3D encoder along with a 2D encoder. Both of these encoders are trained to take the image and its 3D model to one specific latent embedding. We also make use of techniques like feature matching, virtual batch normalization and minibatch discrimination to obtain stability while training GANs.

# TABLE OF CONTENTS

# LIST OF FIGURES

# ABBREVIATIONS

| | |
|---|---|
| **GAN** | Generative Adversarial Network |
| **IWGAN** | Improved Wasserstein Generative Adversarial Network |
| **VAE** | Variational Auto Encoder |

# CHAPTER 1

# INTRODUCTION

Modelling 3D objects from their corresponding 2D counterparts has been an important problem in the field of Computer Vision as it broadens the sense of perception and understanding of a scene and has found several application in design-related industries. With the advent of Deep Learning and the breakthroughs it has achieved on the performance of various 2D Computer Vision tasks, this problem finds new light especially with the availability of huge amounts of 3D data. However, the extension of 2D to 3D modelling isn't a simple and straight-forward task and is dependent on several factors including the task at hand and the way 3D data is represented.

The earliest set of methods for solving the problem like shape from silhouette require calibrated cameras capturing multiple RGB images of the scene (Junyuan Xie, 2016). Humans however can solve this ill-posed problem easily because all the scenes they have experienced in the past which led them to develop a prior understanding of how the objects would like in 3D. The second generation of methods make use of this prior knowledge by posing this problem as a 3D identification problem, thereby eliminating the complex process of calibration.

For 3D objects to have variations and look realistic, the model needs to produce novel shapes and not just reproduce or recombine parts from previously seen 3D objects or models like the traditional heuristic models, that have been used in the past, do. Hence, we use generative adversarial models for the 3D object generation which captures the 3D object structure implicitly. The usage of GANs for the task also allows the generation of new 3D models from a random input in the lower dimensional probabilistic space, thereby completely eliminating the use of reference images or models.

We make use of an encoder-decoder architecture to solve the problem where the decoder is the 3D-GAN mapping from an embedding in the lower dimensional space

to the space of 3D voxel grids. The latent embedding is obtained from a VAE using a single RGB image. We also make use of Wasserstein training objective to obtain stability while learning complex data distributions. An additional 3D encoder is added along with the 2D encoder to ensure that the two data points which are close in the 2D space are mapped to two points which are close in the latent space and vice versa. This removes ambiguity to a certain degree in the output obtained for a given single RGB image.

Training GANs has been hard because of several reasons (Weng, 2019). One of the primary reasons for instability while training GANs is non-convergence. Generator and discriminator are in a minimax game trained to find the Nash equilibrium. Convergence implies finding Nash equilibrium here. The instability increases with every gradient update because of the huge oscillations happening with every update. Mode collapsing might also occur while training GANs where multiple inputs to the generator result in the generation of the same output. This is the case where generator is able to trick the discriminator but is actually stuck in a small space(oscillates) with very small variations and is unable to learn any complex target distribution. GANs have also shown high sensitivity to hyperparameter selections. Diminishing gradients is another cause of instability. With a good discriminator, imbalance between generator and discriminator training leading to overfitting in one. This happens because generally the discriminator learns faster as discrimination is an easier task than generation. The loss function falls to $0$ and we have no gradient updates. We use several methods like virtual batch normalization, mini-batch discrimination, feature selection and adding noise to tackle some of these issues and obtain stability while training GANs for complex data distributions.

# CHAPTER 2

# RELATED WORK

3D object modelling has been extensively studied and researched. Deep networks has been used in the 3D research community for tasks like 3D object recognition (Yangyan Li and Guibas, 2015), learning joint embedding of 3D shapes and image synthesis (Hang Su and Learned-Miller, 2015), single image 3D object reconstruction (Abhishek Kar and Malik, 2015), 3D shape classification (Andrew Brock and Weston, 2016*a*) and 3D shape retrieval (Andrew Brock and Weston, 2016*b*) , to name a few. Some notable architectures like Convolutional Deep Belief Net(CDBN) (Honglak Lee and Ng., 2009), Convolutional Volumetric Auto-Encoder(VConv-DAE) (Abhishek Sharma and Fritz, 2016) and Voxception-ResNet (VRN) (Andrew Brock and Weston, 2016*c*) made use of 3D volumetric representations of 3D objects to exploit the full geometry of the object. These models come with the limitation of being computationally extensive because the filters are of volumetric type which increases the computational complexity cubically. However, with adequate resources at hand, the performance of the models has to proven to be very effective and practical. A lot of the existing architectures to solve the problem require the supervision or the generation followed by supervision of depth maps to guide the network. Convolutional Neural Networks (CNN) along with Kinect data (N. Silberman and Fergus, 2012) has been used to obtain features and refined depth maps to generate the 3D models. A unified CNN framework was proposed (Jiyoung Lee and Sohn, 2017) which used a spatial transformer module for the automatic 2D-to-3D conversion without separately generating or requiring the use of depth maps. A similar notion is used for an end-to-end training of network to automatically convert 2D images to 3D models with the help of video pairs(Junyuan Xie, 2016) making use of the temporal information.

More specific applications of this problem include 3D human body reconstruction which has found uses in gaming and visual effects industry. Some of the parametric methods to solve this problem make use of statistical models like SMPL (M. Loper and Black, 2015) and boiling down the problem to the estimation of these parameters.

A volumetric approach to infer the 3D human body from a single RGB image called BodyNet (G. Varol and Schmid, 2018) used a cascade of networks for 2D and 3D pose inference and the final 3D shape estimation network using the estimates of these previous networks. 3D human face reconstruction has also found light in recent years aiming to capture shape, pose and expressions of the human face. A popular representation used to model human faces in 3D is the 3DMM which captures the variability of the human face in terms of geometry and texture. Parametric methods make use of architectures like VGG-face (O. M. Parkhi, 2015) and Face-Net (F. Schroff and Philbin, 2015) using the 3DMM representation (Blanz and Vetter, 1999) and regress to obtain the parameters with L2 loss trained under 3D supervision.

The usage of adversarial discriminator in the generative models proposed as Generative Adversarial Nets(GANs) (Ian Goodfellow and Bengio, 2014) opened up a plethora of applications where GANs were used from image synthesis (Emily L Denton and Fergus, 2015), image style modelling (Wang and Gupta, 2016), texture synthesis and image editing (Jun-Yan Zhu and Efros, 2018), etc. There have also been several attempts at using GANs for 3D object generation like PrGANs (Gadelha and Wang, 2016) and the usage of conditional GANs (Ong¨un and Temizel, 2018). However, the training of GANs has been difficult due to several reasons like non-convergence, mode collapsing, high sensitivity to hyperparameter selections, diminishing gradients and imbalance between generator and discriminator training leading to overfitting in one. There have been several advances towards achieving a stable training methodology for GANs like usage of regularization (Kevin Roth and Hofmann, 2017), historical averaging, virtual batch normalization and mini-batch discrimination (Tim Salimans and Chen, 2016).

# CHAPTER 3

# PROBLEM STATEMENT

## 3.1 Problem statement

Let $I = \{I_j, j = 1, 2, \ldots, k\}$ be a set of $k \geq 1$ images of an object. We can pose the problem of generation of 3D models from $I$ as learning an estimator $y_\theta$ that takes $I$ and the ground truth model $X$ and learns a 3D shape $\hat{X}$ with the overall objective of minimizing $\ell(I) = d(y_\theta(I), X)$ where $\theta$ is the set of learnable parameters of $y$, $X$ is the original/ground truth 3D shape and $d(.,.)$ is some metric to measure the distance between the original 3D model $X$ and the reconstructed model $f_\theta(I)$ and $\ell$ is the loss function. The predictor $y$ consists of $h$ and $g$ with $y = g \circ h$. $g$ and $h$ differ during training and testing phases and are called the decoder and encoder respectively.

In the following sections, we discuss the choice of $f_\theta$, $\ell$ and $d$. Note we train the our model $f$ using multiple 3D images $I$ obtained by rendering the ground truth 3D model of ShapeNet dataset as images in various orientations. However, the testing is done only on a single RGB image. We also discuss the representation of the 3D model chosen.

## 3.2 Representation of 3D models

We choose volumetric representation of the 3D models instead of surface-based representations because 2D convolutions can be simply extended to 3D with the replacement of up-convolutions in 2D to 3D. This combined with the fact that this representation can be very extensively adapted in network architectures based on deep learning in the three dimensional space allowing the parameterization of 3D shapes using regular voxel grids, makes it a suitable representation. One disadvantage of using the volumetric representation is that they lean towards the heavier side in terms of demands of memory space and can quickly exhaust the memory as the size of the output 3D voxel grid grows.

# CHAPTER 4

# METHODS

In this section, we discuss the encoding-decoding architecture needed to obtain 3D models from 2D images. For the decoder, we use two main models namely 3D-GAN and 3D-IWGAN for 3D object generation from a random input in the low-dimensional probabilistic space (latent vector). For the encoder, we discuss the use of a Variational Auto Encoder(VAE) with the decoder so that the entire resultant framework can learn the mapping of 2D-images to 3D-objects.

## 4.1 Enconding stage - VAE

The input image(s) $I$ need to be encoded into a feature vector $m = h(I) \in \chi$ where $\chi$ is a continuous latent space. We have chosen the latent space to be continuous because a good encoder should satisfy:

- A good encoder $h$ maps two input images $I_1$ and $I_2$ with 3D models close in the 3D voxel space to latent embeddings $m_1, m_2 \in \chi$ that are close to each other.
- A small change in the encoding in $m$ should correspond to a small change in the input image $I$ and vice-versa.

Hence, we use 3D-VAE because their latent spaces are implicitly continuous which helps in generative modelling and easy sampling. In VAE, the input is not mapped directly to a point in the latent space. Instead, it is mapped to mean vector $\mu$ and a standard deviation vector $\sigma$ of a multivariate Gaussian distribution. We then take random samples of these two vectors to obtain the final encoding $x$ in the continuous latent space. One additional advantage of using a VAE as an encoder is that we can use random samples from these lower dimensional space which are used as embeddings to obtain multiple variations of the input which can be used to obtain multiple possible 3D models for a given input image.

## 4.2 Decoding stage - 3D-GAN architecture

The encoded representation of the input image $I$ has to be decoded by $g$ which is the generator in the generator-discriminator architecture. This will map the obtain vector encoding into a volumetric voxel grid. The space around the 3D object is discretized to give this voxel grid. We want the discretization to be as fine as possible to get more accurate representation. This is done using 3D-GANs which have a generator architecture essentially acting as an up-convolutional network or convolutional decoder mirroring the convolutional encoder. However, architectures involving volumetric 3D representations have high computational complecity and memory requirements. As a result, they produce low resolution grids($32 \times 32 \times 32$ in our case). We can try to upscale the architecture to obtain high resolution volumetric grids. However, this comes at the cost of risking exhausting the memory available as the memory requirement will grow cubically with the increase in the output 3D voxel grid resolution. Also, GANs become unstable for high-resolution shapes (Weng, 2019) with mode collapsing and gradient vanishing.

## 4.3 3D-GAN

The Generative Adversarial Network(GAN) (Ian Goodfellow and Bengio, 2014) consists of two networks, a generator $G$ and a discriminator $D$, which together play a mini-max game to learn a target distribution, $P_{\text{data}}$. The generator learns to convert latent vector $z$ into samples as close to a sample $x$ from the target distribution $P_{\text{data}}$ so as to confuse the discriminator and the discriminator learns to differentiate correctly between these synthesized samples and the original data samples. In our context, $z$ is a 200-dimensional randomly sampled latent vector from a probabilistic space and $G$ maps $z$ to $G(z)$ of dimension $64 \times 64 \times 64$ which represents the 3D object in the 3D-voxel space (J. Wu and Tenenbaum, 2016). The discriminator $D$ produces a confidence value $D(G(z))$ to assert how real or synthetic the data is. We have used a binary cross-entropy loss as the overall loss of the 3D-GAN model $L_{\text{3D-GAN}}$ given as:

$$L_{\text{3D-GAN}} = \log D(x) + \log(1 - D(G(z))) \tag{4.1}$$

This model is called 3D-GAN (J. Wu and Tenenbaum, 2016). However, training GANs is very tricky as it might not converge to the desired saddle point optimum. Inspired by Deep Convolutional Generative Adversarial Networks(DCGAN) (Alec Radford and Chintala, 2016), which have become the de facto standard for stable GAN training, we use all-convolutional layers in our generator model. Fig. 4.1 below shows the generator architecture of the model:

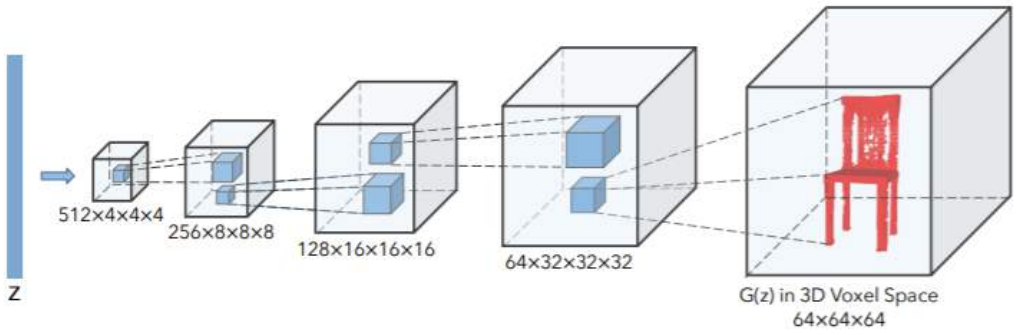


Figure 4.1: The generator architecture of 3D-GAN

The architecture consists of five fully convolutional layers where each kernel is of size $4 \times 4 \times 4$ and stride 2 with using ReLU activation and sigmoid activation at the end. Every layer also uses batch-normalization. It will be seen later on that we later find the use of virtual batch normalization providing better stability during training GANs than vanilla batch normalization. The disciminator network mimics the generator network with a slight difference of using Leaky ReLU instead of ReLU.

### 4.3.1 Limitations of 3D-GAN

3D-GANs, like GANs, suffer from instability in training. In particular, we need to have independent models each trained on a single class as training a single 3D-GAN model with multiple classes and varied poses is difficult and leads to instability.

## 4.4 3D-IW-GAN

To overcome the limitation posed by 3D-GAN, we use Wasserstein training objective with gradient penalty, with the aim of attaining stable joint training over multiple classes with complex distributions. The model is called 3D Improved Wasserstein Generative

Adversarial Network 3D-IWGAN (Smith and Meger, 2017).
$G$ and $D$ play a mini-max game as discussed above with a value function $V(G, D)$ given by(following the notations from the previous section on 3D-GANs):

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim P_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim P_{\text{noise}}}[\log(1 - D(G(z)))] \tag{4.2}$$

GAN training training tries to minimize the KL divergence between the generated and true data distributions which should ideally converge. However, the use of gradient descent while training GANs using this objective leads to instability. There will be an imbalance between the learning of the generator and discriminator which would lead to vanishing gradients. To solve this we use the Wasserstein GAN algorithm (M. Arjovsky and Bottou, 2017) which minimizes the Wasserstein distance between the two distributions. This is defined as:

$$W(P_r, P_g) = \inf_{\psi \sim \Pi(P_r, P_g)} \mathbb{E}_{(x,y) \sim \psi}[||x - y||] \tag{4.3}$$

where $\Pi(P_r, P_g)$ is the set of all joint distributions with marginals $P_r$ and $P_g$. This scheme clips the discriminator's weights to lie within a compact space. However, weight clipping might lead the model towards learning more simple distributions and can lead to gradients either exploding or vanishing. The Improved Wasserstein GAN training scheme(IWGAN), instead of clipping the discriminator's gradients, penalizes the discriminator when they diverge from unity. Let $P_g$ and $P_r$ be defined as the generator and target distributions and $P_x$ is the distribution sampling uniformly on a straight line between $P_g$ and $P_r$, then the resultant loss function of the discriminator becomes:

$$\mathbb{E}_{\hat{x} \sim P_g}[D(\hat{x})] - \mathbb{E}_{x \sim P_r}[D(x)] + \lambda \mathbb{E}_{\hat{x} \sim P_x}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2] \tag{4.4}$$

This scheme, unlike WGAN, shows more promising results in terms of stable convergance and learning more complex distributions.

### 4.4.1 Network architecture:

The generator consists of a fully connected layer with $2048$ nodes which takes a $200$-dimensional vector as an input. The resultant output is then passed through four $3D$

deconvolutional layers each having kernels of size $4 \times 4 \times 4$ and stride 2 which have ReLU activation. Every layer also uses batch-normalization and as we will see later on is changed to use virtual batch normalization to increase stability. The final activation layer is tanh which outputs a $32 \times 32 \times 32$ 3D-voxel object.

The discriminator takes this object and passes it through four convolutional layers each having kernels of size $4 \times 4 \times 4$ and stride 2. There are batch normalization and Leaky ReLU layers in between. The final layer is a fully-connected layer with 2048 nodes and outputs a single value which indicates the confidence of the discriminator of how real the passed sample is. Fig. 4.2 below shows the generator and discriminator architecture of the model:

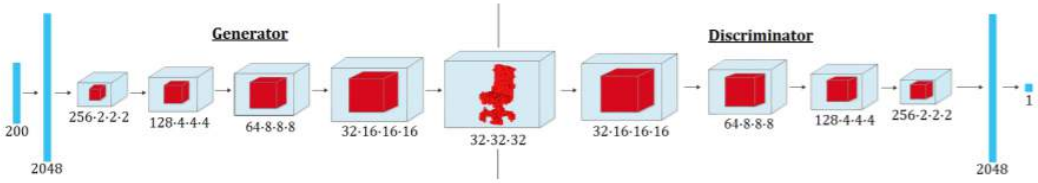


Figure 4.2: The generator and discriminator architecture of 3D-IWGAN

## 4.5 Mapping 2D images to 3D objects: 3D-VAE-IWGAN

Inspired by VAE-GAN (Anders Boesen Lindbo Larsen and Winther, 2016), we propose the use of a Variational Auto Encoder(VAE) $E$ which learns the mapping from a 2D image(samples from target distribution) to a latent representation $z$. This is then used by 3D-IWGAN as discussed, for the generation of a 3D object corresponding to the 2D image. Essentially, the entire network combines the encoding power of VAE and generative power of GANs to synthesize samples conditioned on prior data. We call the overall model 3D-VAE-IWGAN. This model consists of three networks: Variational Auto Encoder $E$ which is essentially an image encoder, generator $G$(also acting as image decoder) and discriminator $D$.

**Architecture of VAE:** The VAE consists of five spatial convolutional layers with ReLU activation. Each layer uses batch-normalization. It finally samples a 200-dimensional latent vector $z$ used by the generator $G$ to synthesize the 3D object.

**Loss function:** Let $y$ be the 2D image and $x$ be its corresponding 3D object. Let

the variational distribution of $z$ given the 2D image $y$ be given by $q(z|y)$. We want this variational distribution $q(z|y)$ to be as close as possible to the prior distribution $p(z)$(chosen to be multivariate Gaussian with zero mean and unit variance). Here, $p(z|y) \sim N(\mu, \Sigma))$ and $p(z) \sim N(0, I)$ where $\mu$ is the means $\Sigma$ is variances given by the VAE. To achieve this, we add a loss function measuring the KL divergence between them and is given as:

$$L_{\text{KL}} = D_{\text{KL}}(q(z|y)||p(z)) = D_{\text{KL}}(N(\mu, \Sigma)||N(0, I)) \tag{4.5}$$

We also add a reconstruction loss $L_{\text{reconst}}$ which minimizes the Euclidean distance between generated and the ground-truth 3D object and is given as:

$$L_{\text{reconst}} = ||G(E(y)) - x||_2 \tag{4.6}$$

So the overall loss function of $E$ given by $L_{\text{VAE}}$ is the linear combination of these two loss and is written as:

$$L_{\text{VAE}} = \alpha_1 L_{\text{KL}} + \alpha_2 L_{\text{reconst}} = \alpha_1 D_{\text{KL}}(q(z|y)||p(z)) + \alpha_2 ||G(E(y)) - x||_2 \tag{4.7}$$

However, the obtained shape of the 3D object might be ambiguous especially when the number of views observed is small because the testing is only done on a single view of the image. Therefore, the generated object might appear correct only from the observed viewpoint and blurry/incorrect from the unobserved ones especially when training hasn't been done for a good number of viewpoints for images of the class from where test image comes from. To overcome this limitation, we propose the addition of perceptual loss component to the overall loss function. Addition of perceptual loss has shown to improve the visual performance of GANs (He Zhang and Patel, 2020) in 2D Computer Vision tasks when the new input doesn't come from the training distribution. Let the ground-truth 3D object corresponding to the 2D image $y$ be given by $X$ and $\hat{x}$ be the synthesized output of the generator which is given as:

$$\hat{x} = G(E(y))$$

We render both the 3D objects $x$ and $\hat{x}$ as images in the same set of orientations. Let the

number of orientations be $K$. Let $Y_R = \{y_1, y_2, \ldots, y_K\}$ and $\hat{Y}_R = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_K\}$ denote the set of rendered ground-truth and synthesized images in these orientations and $W_i$, $H_i$ and $C_i$ be the width, height and number of channels in the $i$th rendered images respectively. Then the perceptual loss $L_{\text{percept}}$ is given by:

$$L_{\text{percept}} = \Sigma_{k=1}^{K} \Sigma_{c=1}^{C_k} \Sigma_{w=1}^{W_k} \Sigma_{h=1}^{H_k} \frac{1}{C_k W_k H_k} ||V(y_k)^{c,w,h} - V(\hat{y}_k)^{c,w,h}||_2^2$$

where $V$ is a non-linear CNN transformation of the image. Here, we have the used the output at layer relu2_2 in VGG-16 model.

The overall loss of 3D-VAE-IWGAN is the summation of the adversarial loss, VAE loss and the perceptual loss.

## 4.6 Joint 2D and 3D embedding

In order to have a good latent representation we need to make sure of the following:

- The enoceded representation should map to the 3D model by the generator
- It is inferrable from the 2D image that is the 2D image can be encoded into that representation

In order to achieve these two goals, we use TL-embedding networks (R. Girdhar and Gupta, 2016) which consists of a 2D-encoder which is the usual 3D-VAE and an additional 3D encoder. Both of these encoders are trained to encode an image and its 3D model to the same embedding. Figure below shows the TL embedding network during the training and testing phases:

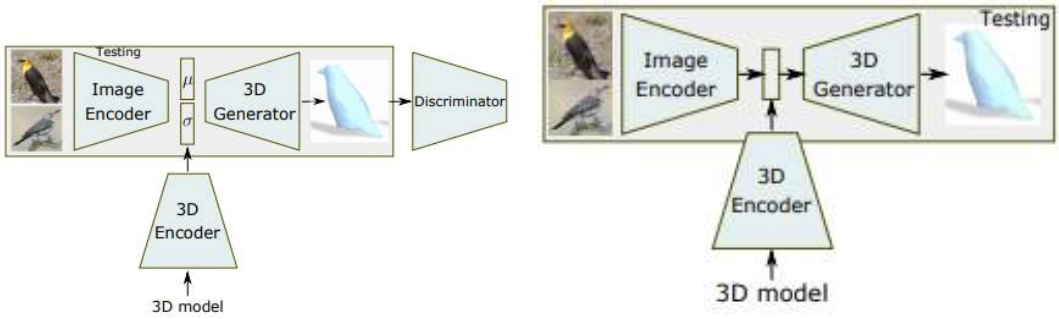


Figure 4.3: 3D-VAE-GAN architecture along with 3D encoder during: (a) Training phase, (b) Testing phase

### 4.6.1 Loss function

Instead of using the Euclidean distance between generated and the ground-truth 3D object as the reconstruction loss, we use Intersection over Union (IoU) which measures the fraction of the intersection of the ground truth and predicted 3D voxel grid to the union of them. Let $X_k$ be the $k$th voxel of the ground truth 3D model and $\hat{X}_k$ be the $k$th voxel of the predicted 3D model. Then the IoU loss is given formally as:

$$IoU_\epsilon = \frac{\hat{X} \cap X}{\hat{X} \cup X} = \frac{\sum_k \{I(\hat{X}_k > \epsilon) * I(X_k)\}}{\sum_k \{I(I(\hat{X}_k > \epsilon) + I(X_k))\}} \tag{4.8}$$

### 4.6.2 Projections for perceptual loss

A modified version of the perceptual loss (M. Gadelha and Wang, 2017) is tried where we assume an orthographic projection. Let $X$ be the 3D grid and $Pr(.)$ is the projection operator. It is given by:

$$Pr((i,j), X) = 1 - \exp^{-\sum_m X(i,j,m)} \tag{4.9}$$

We use the render and compare loss (A. Kundu and Rehg, 2018) as the perceptual loss given as:

$$L_{\text{percept}} = 1 - IoU(y, \hat{y}) + d_{L2}(y, \hat{y}) \tag{4.10}$$

where $IoU$ is the Intersection over Union loss and $d_{L2}$ is the L2 loss between the rendered(projected) images of ground truth and obtained 3D models.

### 4.6.3 Adding noise

Given a single-viewpoint image, the ground-truth shape is ambiguous, i.e. there can be multiple shapes that fit the given 2D image quite well. The current model we have fails to address this issue. We use Min-of-N(MoN) loss (H. Fan and Guibas, 2017) in order to mitigate the issue. The idea is to use a multivariate normal distribution $\mathcal{N}(0, I)$ and sample a random vector $p$ from it. This $p$ is used as noise to perturb the input. All these variations of the noisy inputs are used to train the network which is expected to generate

possible 3D models for each of them. The loss functions is given as:

$$\ell_{MoN} = \sum_{i} \min_{p \sim \mathcal{N}(0,I)} \{d(f(I,p), X)\} \tag{4.11}$$

We can generate various plausible reconstructions for any given input image $I$ using different samples of the random vector $p$ drawn from $\mathcal{N}(0, I)$. We also add noise to the input of the discriminator which has shown to increase the stability during training (Tim Salimans and Chen, 2016).

### 4.6.4 Feature matching

In this technique, we have a new objective which requires the generator to generate data that matches the statistics of real data and this is used to update the weights of the generator. More specifically, we train the generator to match the expected values of the features of an intermediate layer of the discriminator architecture. Training on this new objective enforces the discriminator to find the most discriminative features between the real and synthesized data. The new objective function for the generator can be written as:

$$\arg\max_{\theta} ||\mathbb{E}_{x \sim p_{\text{data}}} f(x) - \mathbb{E}_{z \sim p(z)} f(G_\theta(z))||_2^2 \tag{4.12}$$

where f(x) is the activations on some intermediate layer of the discriminator architecture. The discriminator is trained as vanilla GANs.

### 4.6.5 Virtual Batch normalization

Batch normalization is widely used to stablize the training of GANs as it normalizes the activations of the previous layer to have zero mean and unit variance. However, it leads to the samples within the same batch to be dependent on each other. For example, it has been seen that images generated within the same mini-batch to have peculiarities of the same type like a greenish or orange tint. To avoid interdependence of samples within each mini-batch, we make use of virtual batch normalization (Alec Radford, 2015). In this technique, each sample $x$ is normalized based on a reference batch of samples fixed at the beginning of training rather than normalizing it against samples from its own

mini-batch. While training, we have chosen the first batch as the reference batch.

### 4.6.6 Minibatch discrimination

A very common cause of failure in GANs is mode collapsing in which the generator produces the same data and fools the discriminator. The discriminator recognizes the synthesized data eventually and forces the generator to find another mode. This goes on indefinitely, resulting in the generator not learning any meaningful distribution of data. To solve this, we make use of minibatch discrimination (Tim Salimans and Chen, 2016). The idea is to feed batches of real and synthetic data and then produce a similarity score instead of directly producing a score based on a single input. Whenever the samples within a batch are similar, the similarity score shoots up and the discriminator can accordingly penalize the generator to avoid the mode collapsing.
More formally, let $o(x_i)$ be the similarity score between the image $x_i$ and the other images in the same btach. We use a transformation matrix $T$ to transform the features in image $x_i$ to $M_i$ and is given as:

$$M(x_i) = f(x_i)T \tag{4.13}$$

where $f(x_i)$ is the output in some intermediate layer of the discriminator for image $x_i$. Now the similarity between images $x_i$ and $x_j$ denoted as $c(x_i, x_j)$ is given by the L1-norm as:

$$c(x_i, x_j) = \exp\left(-||M_i, M_j||_1\right) \in \mathbb{R} \tag{4.14}$$

Now the total similarity between $x_i$ and other images in the batch is given by:

$$o(x_i) = \sum_{j=1}^{K} c(x_i, x_j) \in \mathbb{R} \tag{4.15}$$

where $K$ is the number of images in one batch.

# CHAPTER 5

# Experiments and results

In this section, we discuss the training details and obtained 3D objects for both 3D-GANs and 3D-VAE-IWGANs.

## 5.1 3D-GANs

In the subsequent subsections, we discuss the training methodology and results for 3D-GAN.

### 5.1.1 Training:

We train the model on ShapeNet dataset with the following classes: car, chair, desk, gun and sofa. The problem of image discrimination is much easier than image generation. As a result the discriminator learns much faster than the generator leading to unstable training. Hence, we update the discriminator weights after every $p$ updates of the generator weights. $p$ is a hyperparameter which after tuning is set to $5$. ADAM optimization with $\beta = 0.5$ and a batch size of $50$ is used. The learning rates of $G$ and $D$ are set at $0.0025$ and $10^{-5}$ respectively.

### 5.1.2 Results:

For testing, we randomly sampled a 200-dimensional vector $z$ where each dimension is i.i.d uniformly distributed in $[0, 1]$. Fig. 5.1- 5.5 below shows the obtained 3D objects for each class:

Figure 5.1: Objects from ‘chair’ class generated by 3D-GAN trained on the corresponding class in ShapeNet dataset. The images are obtained from a 200-dimensional randomly sampled latent vector $z$ from I[0,1]

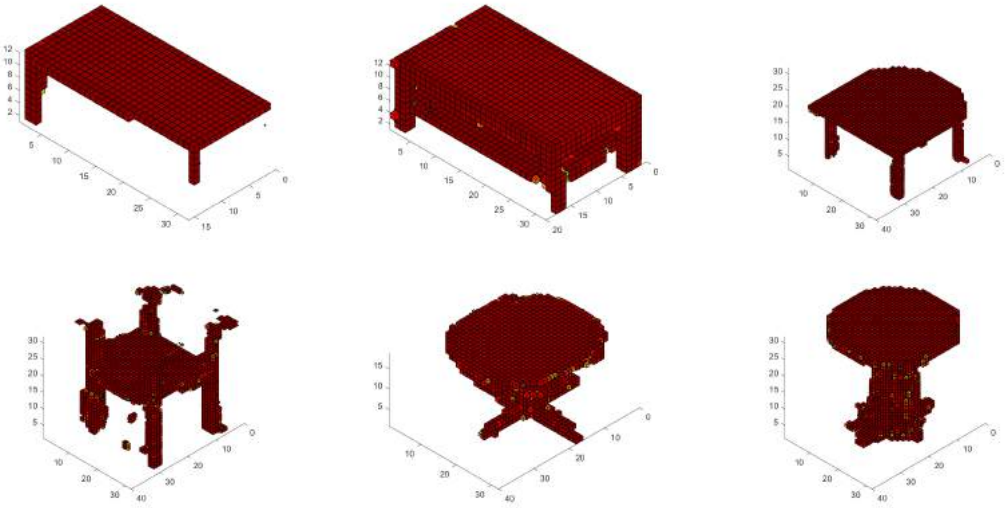

Figure 5.2: Objects from ‘desk’ class generated by 3D-GAN trained on the corresponding class in ShapeNet dataset. The images are obtained from a 200-dimensional randomly sampled latent vector $z$ from I[0,1]

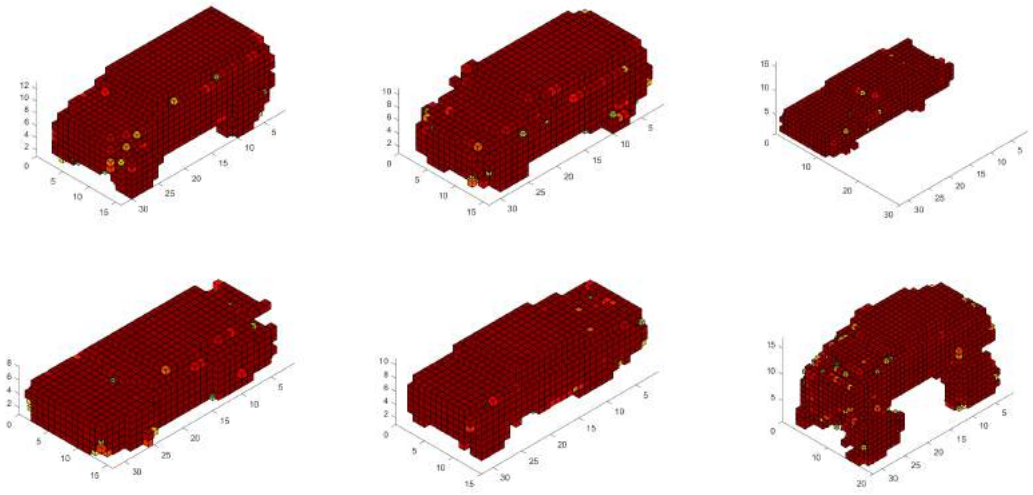

Figure 5.3: Objects from ‘car’ class generated by 3D-GAN trained on the corresponding class in ShapeNet dataset. The images are obtained from a 200-dimensional randomly sampled latent vector $z$ from I[0,1]

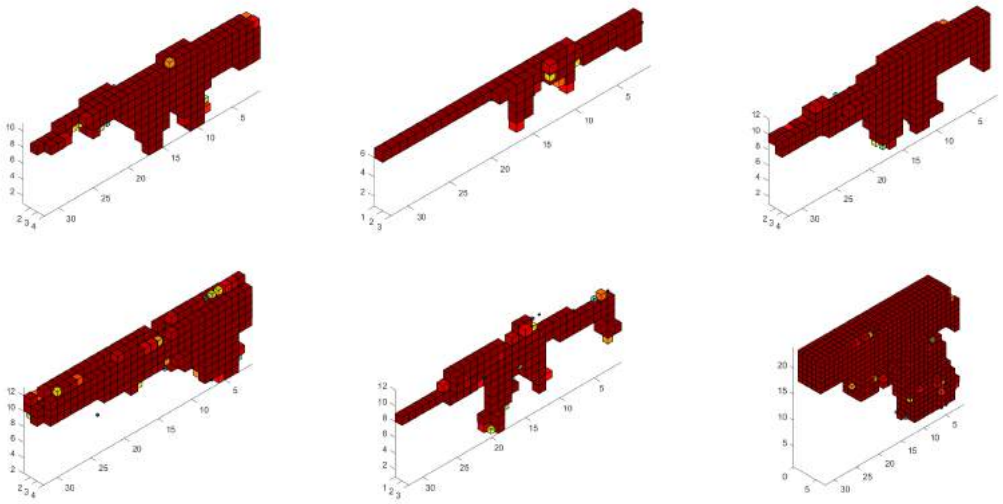

Figure 5.4: Objects from ‘gun’ class generated by 3D-GAN trained on the corresponding class in ShapeNet dataset. The images are obtained from a 200-dimensional randomly sampled latent vector $z$ from I[0,1]

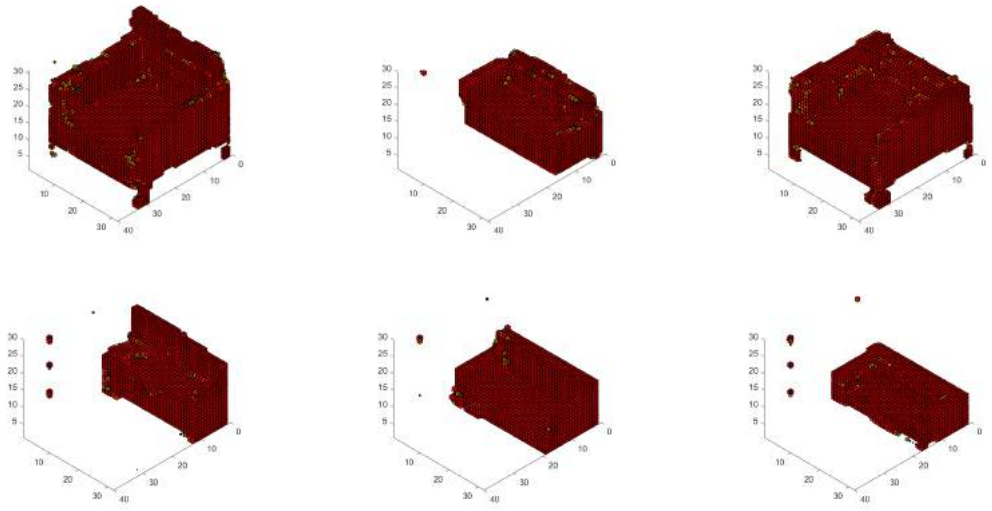

Figure 5.5: Objects from 'sofa' class generated by 3D-GAN trained on the corresponding class in ShapeNet dataset. The images are obtained from a 200-dimensional randomly sampled latent vector $z$ from I[0,1]

## 5.2 3D-VAE-IWGAN

In the subsequent subsections, we discuss the training methodology and results for 3D-VAE-IWGAN. Note: Only one model is used for joint training of multi-class inputs in various poses as opposed to 3D-GAN(where a separate model had to be trained for each class) because of the stability provided by Wasserstein training objective to learn complex distributions.

### 5.2.1 Training:

Both 2D images and their ground truth 3D models are required to train 3D-VAE-IWGAN. The dataset consists of images obtained by rendering 3D shapes from the ShapeNet dataset consisting of object classes: chair, desk, gun, sofa and car, at random poses and lighting conditions. A subset of it is used to train the model for $1100$ epochs. We also adapt our training to include historical averaging for better stability during training. We add an extra term $||\Theta - \frac{1}{t}\Sigma_{i=1}^{t}\Theta_i||^2$ into the loss function where $\Theta$ is the weight for generator and discriminator and $\Theta_i$ are their weights in the past training update $i$ (Weng, 2019). This is done to penalize large changes in $\Theta$. We tune $t$ and set it to 3. The remaining subset of rendered images is used to test to the model

and generate corresponding 3D objects.

Fig. 5.6- 5.10 below shows some of these rendered images for each class with a single image in different orientations:

Figure 5.6: Rendered images in different orientations using 3D shapes from ShapeNet dataset for class ‘chair’

Figure 5.7: Rendered images in different orientations using 3D shapes from ShapeNet dataset for class ‘desk’

Figure 5.8: Rendered images in different orientations using 3D shapes from ShapeNet dataset for class ‘car’

Figure 5.9: Rendered images in different orientations using 3D shapes from ShapeNet dataset for class ‘gun’

Figure 5.10: Rendered images in different orientations using 3D shapes from ShapeNet dataset for class ‘sofa’

### 5.2.2 Results:

For testing, we used the remaining subset of rendered images(from 3D objects of ShapeNet).

Fig. 5.11- 5.15 below shows the obtained 3D objects for each class:

Figure 5.11: Input image and its obtained 3D model using 3D-VAE-IWGAN for class ‘chair’

Figure 5.12: Input image and its obtained 3D model using 3D-VAE-IWGAN for class ‘desk’

Figure 5.13: Input image and its obtained 3D model using 3D-VAE-IWGAN for class ‘car’

Figure 5.14: Input image and its obtained 3D model using 3D-VAE-IWGAN for class ‘gun’

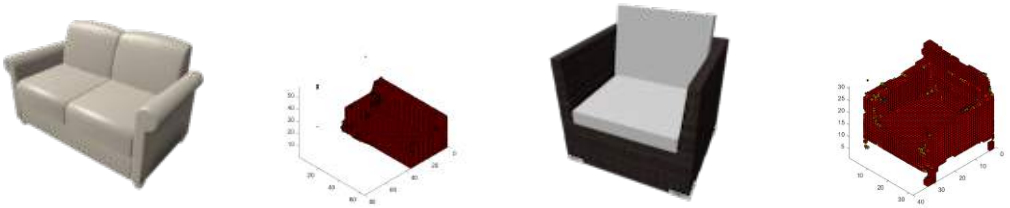

Figure 5.15: Input image and its obtained 3D model using 3D-VAE-IWGAN for class ‘sofa’

## 5.3 3D-VAE-GAN with 3D encoder

In the subsequent sections, we discuss the training methodolgy and results obtained with the use of an additional 3D encoder along with the 3D-VAE-GAN.

### 5.3.1 Training

The model is trained in three phases for each epoch:

- The 2D encoder along with the decoder is trained independent of the 3D encoder in the first phase.
- The 3D encoder is trained to regress the latent embedding obtained for each input image in the next phase.
- In the final phase, the entire network is fine-tuned together.

### 5.3.2 Results

Fig. 5.16- 5.19 below shows the obtained 3D objects for each class along with their corresponding 3D ground truth models:

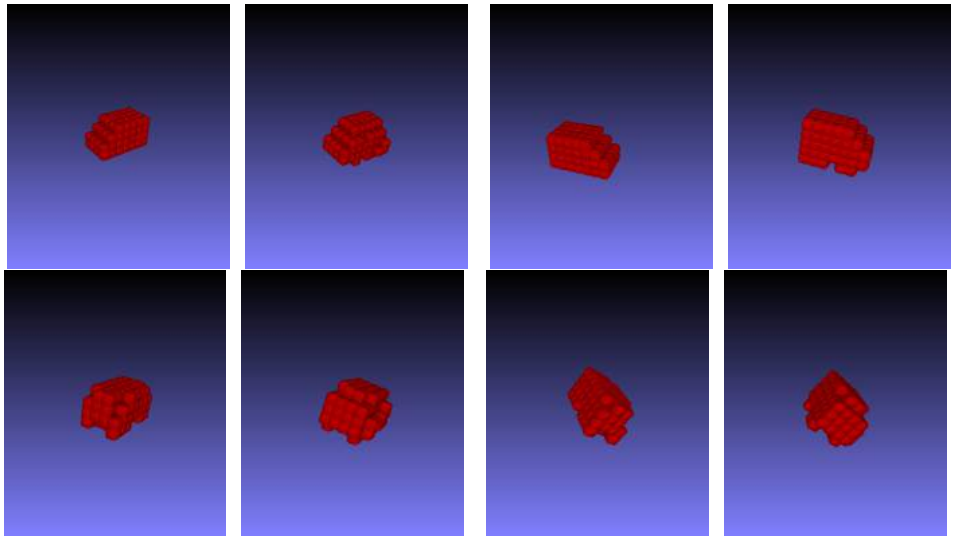

Figure 5.16: Two examples of ground truth model(on the left) and its obtained 3D model(on the right) in different orientations for the class ‘car’

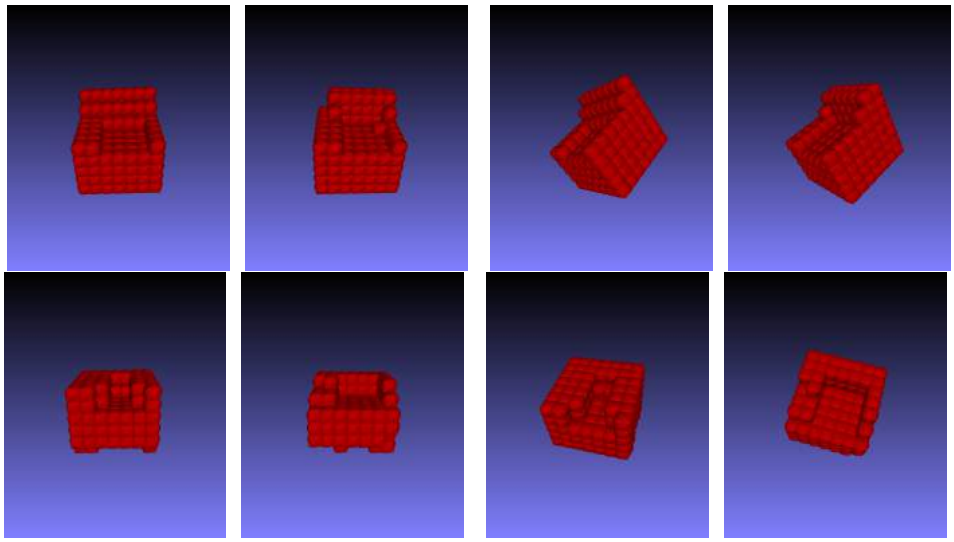

Figure 5.17: Two examples of ground truth model(on the left) and its obtained 3D model(on the right) in different orientations for the class ‘chair’

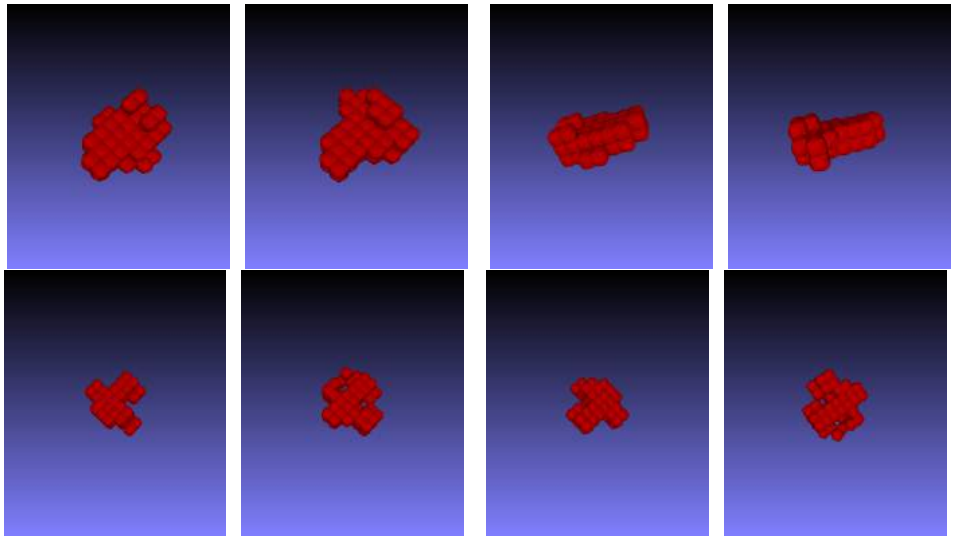

Figure 5.18: Two examples of ground truth model(on the left) and its obtained 3D model(on the right) in different orientations for the class ‘plane’

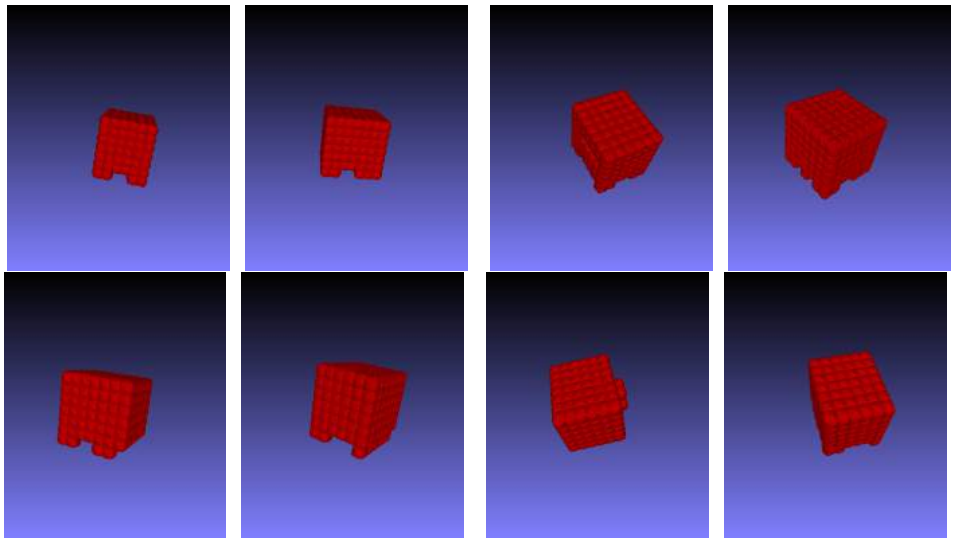

Figure 5.19: Two examples of ground truth model(on the left) and its obtained 3D model(on the right) in different orientations for the class ‘table’

Fig. 5.20- 5.23 below shows the obtained 3D model(on the right) along with the ground truth(in the middle) for a given input image(on the left):

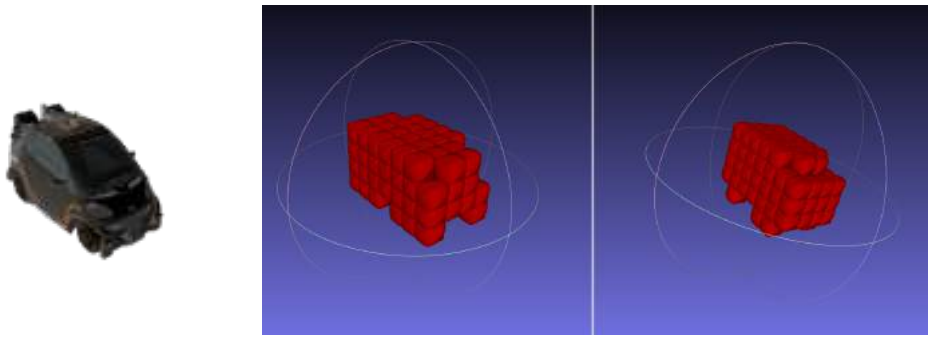

Figure 5.20: Input image(on the left), its ground truth 3D model(in the middle) and the obtained 3D model(on the right) for the class ‘car’

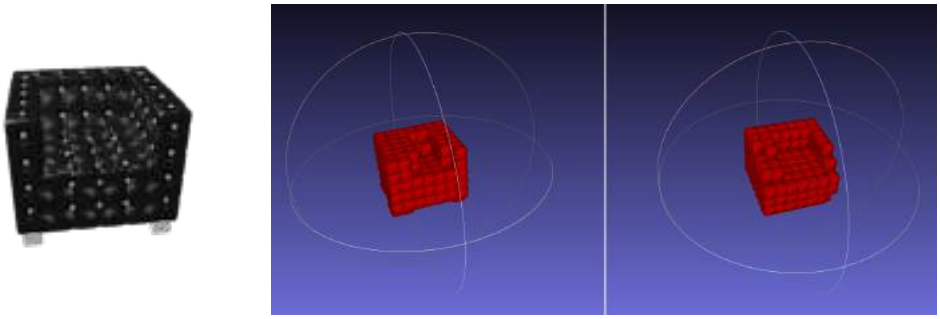

Figure 5.21: Input image(on the left), its ground truth 3D model(in the middle) and the obtained 3D model(on the right) for the class ‘chair’

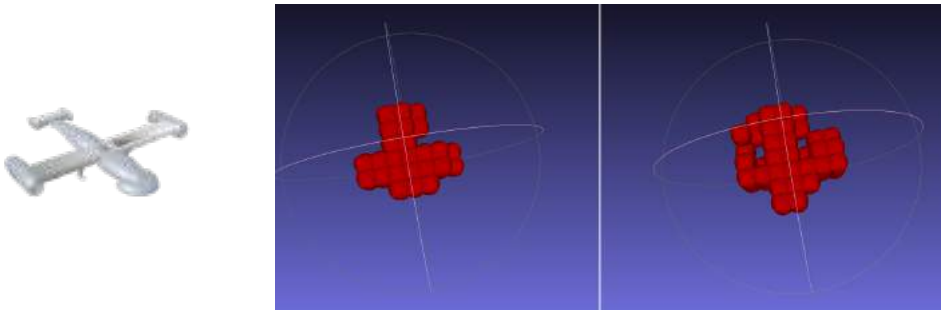

Figure 5.22: Input image(on the left), its ground truth 3D model(in the middle) and the obtained 3D model(on the right) for the class ‘plane’

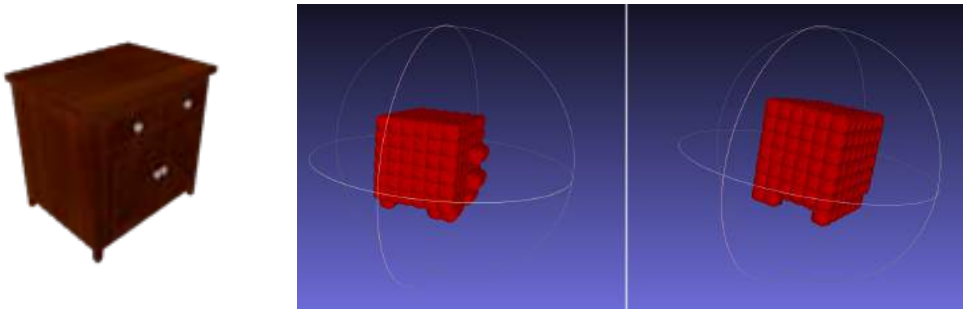

Figure 5.23: Input image(on the left), its ground truth 3D model(in the middle) and the obtained 3D model(on the right) for the class ‘table’

# CHAPTER 6

# Future work

The work done in this project along with current state-of-the-art methods obtain coarse 3D voxel grids. Several improvements can be made to obtain high resolution 3D models. Some of the techniques used are space partitioning (P.-S. Wang and Tong, 2017), occupancy networks (Chen and Zhang, 2019) and shape partioning (J. Li and Guibas, 2017).

Several of current techniques use single object images without any occlusion or cluttering in the background Weng (2019). However, most of the images obtained in real-life contain multiple objects with cluttered background. Previous works to solve this have used regions of interest to detect the objects. The identification and generate modules in these works are trained independent of each other. However, since clearly these tasks are inter-dependent, a jointly trained model can be expected to have better performance.

In our project and most research done in the area, the same dataset is split as training, validation and test data. The performance is reported on the data coming from the same dataset. However, an ideal goal would be to see the generalization of the model on unseen images/objects. An ideal model should be able to generate reasonable 3D models from any arbitrary image containing aribtrary objects. It has been explored recently (I. Cherabier and Geiger, 2018) and would be a good direction to further the focus on.

The final goal would be able to do parse an entire scene into 3D using one or more RGB images. This would require capturing the spatial interaction between multiple objects and their joint detection and modelling. However, not much advancement has been done in this area. Existing work related to full scene 3D parsing have made some

inviolable and harsh assumptions like only considering objects in indoor scenes. Weng (2019).

# CHAPTER 7

# Conclusion

In this project, we use three models for generation of 3D objects from 2D images. First, we train the 3D-GAN model(separate model for each class) and show its limitations for joint training of multi-class complex distributions. For stable training and better convergence, we then use the improved Wasserstein training objective with the 3D-IWGAN model and add a VAE to learn the mapping from 2D images to 3D-voxel space. We also increase the novelty of the established framework by adding a perceptual loss component to the model which improves the visual performance of the generated outputs as the new input to the model is a single image(single viewpoint). An extra loss component is added for historical averaging which ensures the parameters of the generator and discriminator do not change drastically in between consecutive updates to partially mitigate the problem of exploding or vanishing gradients. For a single viewpoint of a scene, there can be multiple 3D models which explain the scene. A good encoder which maps two points close in the 2D space to two points in the 3D voxel space which are close and vice versa, is needed to reduce the ambiguity. This is done with the help of an additional 3D encoder which is jointly trained with VAE encoder. Several other techniques like minibatch discrimination and virtual batch normalization are used to obtain stablility in GAN training.

# REFERENCES


1. **A. Kundu, Y. L.** and **J. M. Rehg**, 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. *In IEEE CVPR, pp. 3559–3568*. 2018.

2. **Abhishek Kar, J. C., Shubham Tulsiani** and **J. Malik**, Category-specific object reconstruction from a single image. *In CVPR*. 2015.

3. **Abhishek Sharma, O. G.** and **M. Fritz**, Vconv-dae: Deep volumetric shape learning without object labels. *In In Geometry Meets Deep Learning Workshop at European Conference on Computer Vision (ECCV-W)*. 2016.

4. **Alec Radford, L. M.** and **S. Chintala**, Unsupervised representation learning with deep convolutional generative adversarial networks. *In ICLR*. 2016.

5. **Alec Radford, S. C., Luke Metz**, Unsupervised representation learning with deep convolutional generative adversarial networks. *In CVPR*. 2015.

6. **Anders Boesen Lindbo Larsen, S. K. S.** and **O. Winther**, Autoencoding beyond pixels using a learned similarity metric. *In ICML*. 2016.

7. **Andrew Brock, J. M. R., Theodore Lim** and **N. Weston**, Generative and discriminative voxel modeling with convolutional neural networks. *In arXiv preprint arXiv:1608.04236*. 2016*a*.

8. **Andrew Brock, J. M. R., Theodore Lim** and **N. Weston**, Generative and discriminative voxel modeling with convolutional neural networks. *In arXiv preprint arXiv:1608.04236*. 2016*b*.

9. **Andrew Brock, J. M. R., Theodore Lim** and **N. Weston**, Generative and discriminative voxel modeling with convolutional neural networks. *In arXiv preprint arXiv:1608.04236*. 2016*c*.

10. **Blanz, V.** and **T. Vetter**, A morphable model for the synthesis of 3d faces. *In Siggraph, pp. 187–194*. 1999.

11. **Chen, Z.** and **H. Zhang**, Learning implicit fields for generative shape modeling. *In IEEE CVPR, pp. 5939–5948.*. 2019.

12. **Emily L Denton, S. C.** and **R. Fergus**, Deep generative image models using a laplacian pyramid of adversarial networks. *In NIPS*. 2015.

13. **F. Schroff, D. K.** and **J. Philbin**, Facenet: A unified embedding for face recognition and clustering. *In IEEE CVPR, pp. 815–823.*. 2015.

14. **G. Varol, B. R. J. Y. E. Y. I. L., D. Ceylan** and **C. Schmid**, Bodynet: Volumetric inference of 3d human body shapes. *In ECCV*. 2018.

15. **Gadelha, S., M.; Maji** and **R. Wang**, 3d shape induction from 2d views of multiple objects. *In arXiv:1612.05872*. 2016.

16. **H. Fan, H. S.** and **L. Guibas**, A point set generation network for 3d object reconstruction from a single image. *In IEEE CVPR*. 2017.

17. **Hang Su, E. K., Subhransu Maji** and **E. Learned-Miller**, Multi-view convolutional neural networks for 3d shape recognition. *In ICCV*. 2015.

18. **He Zhang, V. S.** and **V. M. Patel**, Image de-raining using a conditional generative adversarial network. *In IEEE Transactions on Circuits and Systems for Video Technology*. 2020.

19. **Honglak Lee, R. R., Roger Grosse** and **A. Y. Ng.**, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *In Proceedings of the 26th annual international conference on machine learning. ACM, 609–616.*. 2009.

20. **I. Cherabier, M. R. O. M. P., J. L. Schonberger** and **A. Geiger**, Learning priors for semantic 3d reconstruction. *In ECCV*. 2018.

21. **Ian Goodfellow, M. M. B. X. D. W.-F. S. O. A. C., Jean Pouget-Abadie** and **Y. Bengio**, Generative adversarial nets. *In NIPS*. 2014.

22. **J. Li, S. C. E. Y. H. Z., K. Xu** and **L. Guibas**, Grass: Generative recursive autoencoders for shape structures. *In ACM TOG, vol. 36, no. 4, p. 52*. 2017.

23. **J. Wu, T. X. B. F., C. Zhang** and **J. Tenenbaum**, Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *In In Advances in Neural Information Processing Systems, pages 82–90*. 2016.

24. **Jiyoung Lee, Y. K., Hyungjoo Jung** and **K. Sohn**, Automatic 2d-to-3d conversion using multi-scale deep neural network. *In IEEE International Conference on Image Processing (ICIP)*. 2017.

25. **Jun-Yan Zhu, E. S., Philipp Krähenbühl** and **A. A. Efros**, Generative visual manipulation on the natural image manifold. *In CVPR*. 2018.

26. **Junyuan Xie, A. F., Ross Girshick**, Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. *In ECCV*. 2016.

27. **Kevin Roth, S. N., Aurelien Lucchi** and **T. Hofmann**, Stabilizing training of generative adversarial networks through regularization. *In NIPS*. 2017.

28. **M. Arjovsky, S. C.** and **L. Bottou**, Wasserstein gan. *In ArXiv e-prints*. 2017.

29. **M. Gadelha, S. M.** and **R. Wang**, 3d shape induction from 2d views of multiple objects. *In 3D Vision, pp. 402–411.*. 2017.

30. **M. Loper, J. R. G. P.-M., N. Mahmood** and **M. J. Black**, Smpl: A skinned multi-person linear model. *In ACM TOG, vol. 34, no. 6, p. 248*. 2015.

31. **N. Silberman, D. H., P. Kohli** and **R. Fergus**, Indoor segmentation and support inference from rgb-d images. *In Proc. Eur. Conf. Comput. Vis., pp. 746-760*. 2012.

32. **O. M. Parkhi, A. Z. e. a., A. Vedaldi**, Deep face recognition. *In BMVC, vol. 1, no. 3, p. 6.*. 2015.

33. **Ong¨un, C.** and **A. Temizel**, Paired 3d model generation with conditional generative adversarial networks. *In CVPR*. 2018.

34. **P.-S. Wang, Y.-X. G. C.-Y. S., Y. Liu** and **X. Tong**, Ocnn: Octree-based convolutional neural networks for 3d shape analysis. *In ACM TOG, vol. 36, no. 4, p. 72*. 2017.

35. **R. Girdhar, M. R., D. F. Fouhey** and **A. Gupta**, Learning a predictable and generative vector representation for objects. *In ECCV, pp. 484–499*. 2016.

36. **Smith, E. J.** and **D. Meger**, Improved adversarial systems for 3d object generation and reconstruction. *In CVPR*. 2017.

37. **Tim Salimans, W. Z. V. C. A. R., Ian Goodfellow** and **X. Chen**, Improved techniques for training gans. *In CVPR*. 2016.

38. **Wang, X.** and **A. Gupta**, Generative image modeling using style and structure adversarial networks. *In ECCV*. 2016.

39. **Weng, L.**, From gan to wgan. *In arXiv:1904.08994*. 2019.

40. **Yangyan Li, C. R. Q. N. F. D. C.-O., Hao Su** and **L. J. Guibas**, oint embeddings of shapes and images via cnn image purification. *In ACM TOG, 34(6):234*. 2015.

# LIST OF PAPERS BASED ON THESIS


1. J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling, *In Advances in Neural Information Processing Systems, pages 82–90*, (2016).

2. Edward J. Smith and David Meger Improved Adversarial Systems for 3D Object Generation and Reconstruction *CVPR*, (2017).

3. Xian-Feng Han, Hamid Laga, Mohammed Bennamoun Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era *CVPR*, (2019).