REPORT PROJECT COMPUTER VISION

Predict model 3D from 2D used ResNet18

[1]Trinh Thanh An
[2]Nguyen Minh Quan
[3]Phan Nguyet Minh
[4]Pham Ngoc Trong
[5]Nguyen Ba Tung Duong

FPT University, TP.HCM, Viet Nam

e-mail: [1] trinhthanhan00789@gmail.com,
e-mail: [2] minhquan02102005@gmail.com,
e-mail: [3] pnguyetmiinh@gmail.com,
e-mail: [4] phamngoctrong13082005@gmail.com
e-mail: [5] duong2mailegend@gmail.com

July 24, 2025

Abstract

Reconstructing 3D models from 2D images is a highly challenging inverse problem in the field of computer vision, especially when only a single RGB image is provided as input. This project proposes a complete pipeline for single-image 3D reconstruction using ResFesT — a state-of-the-art deep learning architecture that combines Convolutional Neural Networks (CNN) and Transformers. The system is trained on the Pix3D dataset, with voxel grids as the output representation for object geometry. After preprocessing the dataset by converting voxel files from '.mat' to '.npy', and building an appropriate training pipeline, the model demonstrates promising performance in reconstructing common household objects such as chairs, tables, and beds. This report presents the data processing pipeline, model architecture, experimental results, limitations, and potential directions for future improvement.

Contents

1 Introduction

Reconstructing 3D models from 2D images is a fundamental but challenging problem in computer vision. This task, known as single-view 3D reconstruction, is inherently ill-posed due to missing depth and spatial information. With growing demand in fields like graphic design, AR/VR, and robotics, automating 3D modeling from a single RGB image can significantly reduce manual effort in creating digital content.

In this project, we aim to build a complete pipeline that generates 3D voxel-based models from a single 2D image. We utilize the ResFesT architecture — which combines convolutional networks and Transformers — to reconstruct common household objects such as chairs, tables, and beds. The Pix3D dataset is used for training and evaluation, and we focus on implementing a simplified yet effective system that supports practical 3D object generation from minimal input.

2 Theoretical Background

2.1 3D Reconstruction from 2D Images – ResFesT

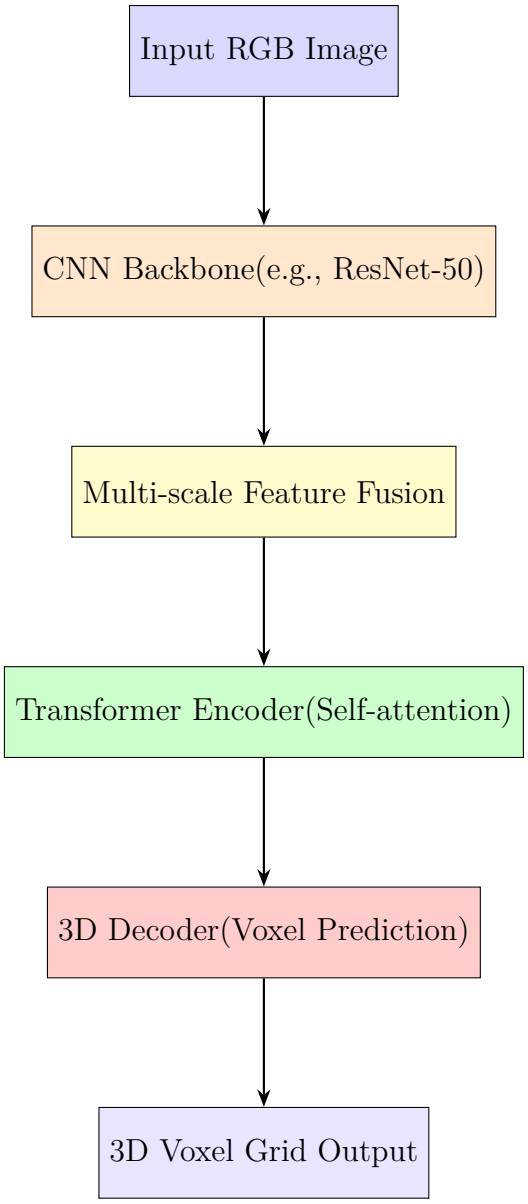
Reconstructing 3D models from a single RGB image is a classic inverse problem in computer vision. The task aims to infer the 3D spatial structure—such as geometry, volume, and depth—based solely on a flat 2D image. This is inherently ill-posed, as 2D images lack explicit depth information and may suffer from occlusion, limited viewpoint, and lighting variations.

To address this challenge, modern deep learning models attempt to learn the mapping between 2D image features and their corresponding 3D structures. One such architecture is **ResFesT** (Resolution-aware Feature-fusion Transformer), which combines CNNs and Transformers to produce accurate 3D voxel reconstructions from a single image.

Key components of ResFesT:

- **Backbone CNN (ResNet-50):** A deep convolutional neural network is used to extract spatial features from the input image. The CNN layers enable the model to recognize edges, shapes, and textures at different levels of abstraction.
- **Multi-scale Feature Fusion:** Features are aggregated from multiple layers (multi-level) at different resolutions, allowing the model to capture both fine details and the overall structure of the object. This is especially important for objects with complex geometry, such as chairs or bookshelves.
- **Transformer Encoder:** It learns long-range dependencies and correlations between different regions of the image. The use of self-attention in the Transformer enables the model to identify relationships between spatially distant parts of the image in 2D space.
- **3D Decoder:** The output is reconstructed in the form of a voxel grid — a 3D volume divided into small cells (voxels). Each voxel is assigned a probability indicating the likelihood of occupancy at that specific location.

ResFesT has shown superior performance over previous models such as 3D-R2N2 and Pix2Vox, especially in reconstructing small-scale geometric details and complex structures.



2.2 Pix3D Dataset

Training 3D reconstruction models requires high-quality datasets that provide real-world images along with accurate 3D models. **Pix3D** is one of the most widely used datasets for this purpose.

Key characteristics of Pix3D:

- **Real RGB images:** Captured from real-world scenes instead of synthetic environments, ensuring better generalization.
- **Accurate 3D models:** Each image has a corresponding 3D model in formats like `.obj` or voxel.
- **Detailed annotations:** Includes camera pose, object bounding boxes, keypoints, and JSON mappings between 2D images and 3D volumes.
- **Diverse object categories:** Includes chairs, beds, tables, and other household items to ensure variety in geometric forms.

Pix3D is used in this project to train and evaluate the ResFesT model. Each RGB image is linked to a ground-truth voxel volume, enabling supervised learning of the 3D reconstruction task.

2.3 3D Representations

Choosing an appropriate 3D representation is essential for model architecture and computational feasibility. Common types of 3D representations include:

2.3.1 Voxel Grid

- Represents 3D space as a dense grid of voxels, similar to pixels in 2D.
- Each voxel holds a binary or probabilistic value indicating occupancy.
- Compatible with convolutional networks and easy to integrate into training pipelines.
- Consumes a large amount of memory, especially at higher resolutions (e.g., 64^3 or 128^3).

In this project, ResFesT uses voxel grids due to their simplicity and compatibility with 3D CNN-based decoders.

2.3.2 Mesh

- Represents object surfaces using vertices, edges, and triangular faces.
- Ideal for visualization and rendering, and compact in storage.
- Difficult to learn using deep networks due to its irregular graph-based structure.

2.3.3 Implicit Representation

- Defines 3D shape using a continuous function $f(x, y, z)$ (e.g., Signed Distance Functions or Occupancy Networks).
- Allows sub-voxel precision and fine surface reconstruction.
- Computationally demanding and harder to train compared to voxel-based methods.

While voxel grids are chosen for this project, future extensions could explore mesh or implicit formats to improve surface fidelity and efficiency.

3 Data Preprocessing

3.1 Raw Dataset (Pix3D)

The original Pix3D dataset is composed of four key components:

- **RGB images:** Stored in `.jpg` or `.png` format within the `img/` directory.

- **Segmentation masks:** Optional `.png` or `.jpg` masks located in a separate `masks/` folder. These masks are used to isolate the foreground object from the background during preprocessing or training.
- **Voxel files:** Each object is represented as a 3D voxel grid and stored in `.mat` format (MATLAB files). These voxel grids indicate whether a given point in 3D space is occupied or not, forming the ground truth for supervised training.
- **JSON metadata file:** The `pix3d.json` file provides annotations that link each RGB image to its corresponding 3D voxel file, mask, object category, camera parameters, keypoints, and bounding boxes.

Challenge: Cross-modality mapping. One of the core challenges in using the Pix3D dataset is that each 2D image is associated with a unique 3D object instance, captured under a specific camera pose and viewpoint. Consequently, the RGB image, segmentation mask, and voxel file for each sample are **not stored together or named consistently**. Instead, these components must be accurately connected through the metadata in the `pix3d.json` file.

Each entry in the JSON file contains:

- The file path to the corresponding RGB image
- The object class and instance ID
- The path to the voxel file (in `.mat` format)
- Camera intrinsics and extrinsics
- 2D keypoints and 3D model alignment information

This mapping is essential because each image represents the object from a different angle, perspective, and lighting condition. Without correct alignment via the JSON annotations, the model would incorrectly learn mismatched image-voxel pairs, leading to poor performance during training and inference. Therefore, careful preprocessing is required to ensure that the 2D and 3D data are semantically and spatially consistent.

3.2 Processed Dataset

Voxel Conversion (`.mat` to `.npy`)

The voxel files in the raw dataset are stored in MATLAB's `.mat` format, which is not natively supported in most Python-based frameworks. Therefore, each `.mat` file was parsed and converted into a NumPy array (`.npy`) using a conversion script. These `.npy` files can be easily loaded using `numpy` or integrated into PyTorch `Dataset` classes without the need for third-party libraries like `scipy.io`.

This step greatly improves:

- **Loading speed:** NumPy arrays can be accessed directly in RAM.
- **Compatibility:** Easier integration with neural network models.
- **Storage consistency:** All files share the same format and dimensions.

Image–Voxel–Mask Mapping via JSON

Because each RGB image, segmentation mask, and voxel file in Pix3D represents a unique object instance and viewpoint, it is critical to link them correctly. This mapping is accomplished using the `pix3d.json` metadata file.

During preprocessing, a Python script traverses each JSON entry and extracts:

- The file path to the image (e.g., `img/xxx.png`)
- The corresponding segmentation mask (if available)
- The linked voxel file (`.mat`, converted to `.npy`)
- Object category (e.g., chair, table, bed)
- Viewpoint and keypoints metadata

From this information, a unified dataset index is created. Each training sample now consists of:

```
(image, mask, voxel, label)
```

Why JSON-based linking is necessary:

- Each image is captured from a unique viewpoint and has different camera intrinsics.
- Each object instance may appear in multiple images but with different poses.
- There is no 1-to-1 filename convention across image, mask, and voxel directories.

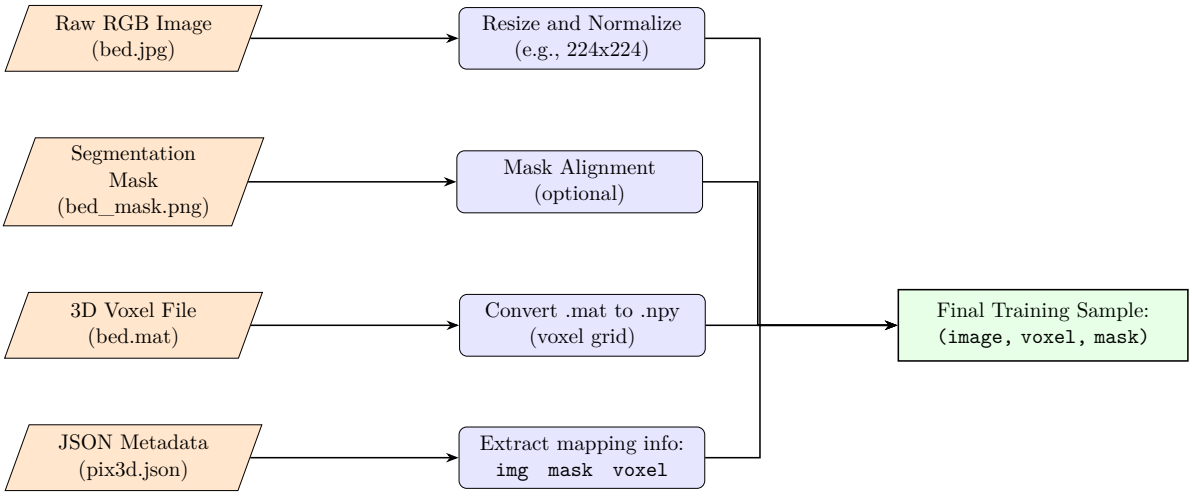
Without JSON-based alignment, mismatches between 2D and 3D data would occur—resulting in incorrect supervision during model training.

Final Structure

After preprocessing, the dataset is organized as:

- `images/` – Resized and normalized RGB images
- `voxels/` – Converted voxel grids in `.npy` format
- `masks/` – Optional binary masks (aligned with images)
- `metadata.json` – Unified mapping of all components

This structure supports efficient loading, random sampling, and batched training in PyTorch.



3.3. Ground Truth Generation (Occupancy / Voxel / Mesh)

In 3D reconstruction tasks, **ground truth** refers to the labeled data that provides a reference shape for training and evaluating models. This ground truth is essential for supervised learning, where the model learns to map input 2D images to their corresponding 3D structures.

Depending on the design of the system, ground truth 3D data can be represented in several formats:

- **Occupancy Grid (Voxel):** The 3D space is divided into a regular grid of small cubes called voxels (volumetric pixels). Each voxel is assigned a binary value:
  - 1 (occupied) if the voxel lies within the object's volume
  - 0 (empty) if the voxel lies outside the object

  This format is easy to integrate into convolutional neural networks (3D CNNs), but it consumes large memory, especially at high resolutions (e.g., 64^3, 128^3).

- **Mesh:** A mesh represents an object's surface as a collection of vertices, edges, and triangular faces. This format is very compact and suitable for visualization or physical simulation. However, since mesh data is irregular and not grid-aligned, it is harder to handle using standard deep learning architectures.

- **Implicit Field:** Instead of storing explicit geometry, an implicit function $f(x, y, z)$ is trained to describe the object surface. For example:
  - Signed Distance Functions (SDF): Store the shortest distance from each point to the surface.
  - Occupancy Networks: Store the probability that a point is inside the object.

  These methods offer high flexibility and continuous resolution, but require special architectures and large datasets to train effectively.

In this project:

We use the **voxel grid (occupancy)** format as the ground truth representation. This is due to several reasons:

- It provides a structured and regular format that aligns well with 3D CNN-based decoders.
- It is easier to visualize and evaluate (e.g., using IoU scores).
- It integrates smoothly with PyTorch and NumPy during training and inference.

Initially, voxel data is stored in MATLAB `.mat` files. These are not directly compatible with PyTorch pipelines. Therefore, as part of the preprocessing step, all `.mat` voxel files are parsed and converted into NumPy `.npy` format.

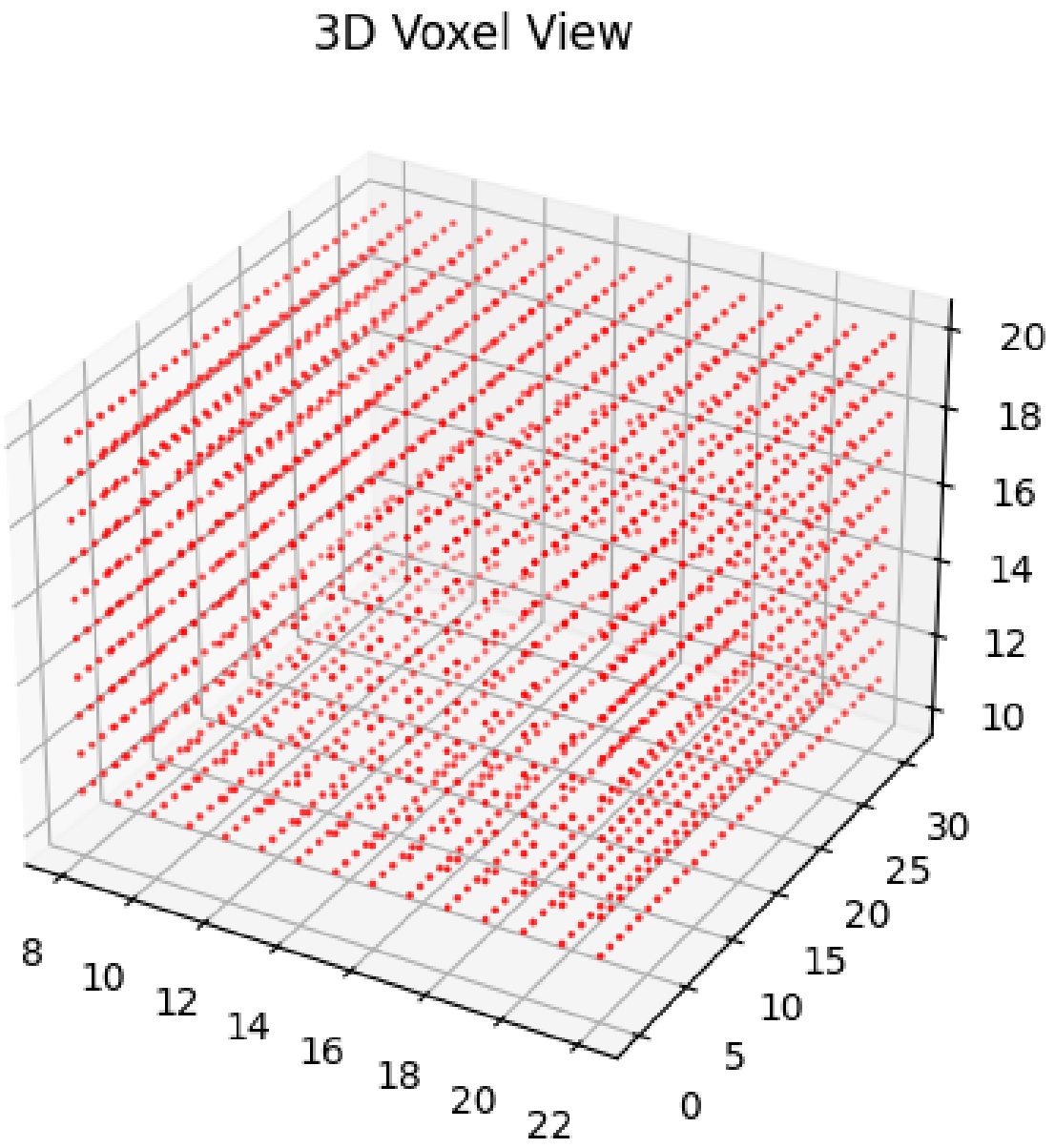
This conversion ensures:

- Faster data loading during training
- Elimination of dependency on MATLAB toolboxes
- Compatibility with PyTorch's `torch.utils.data.Dataset` APIs

As a result, each training sample includes:

`(RGB image, segmentation mask, voxel grid, label)`

This ground truth representation plays a crucial role in enabling the model to learn accurate 3D shapes from only a single 2D input.



In addition to verifying the 2D input data, we also visualized the corresponding 3D voxel grid. Displays a 3D scatter plot of occupied voxels. This check ensures that the voxel representation correctly describes the object geometry and aligns with the input image and segmentation mask.

4 Proposed Method

4.1 Overview of the Pipeline

Figure 1: Pipeline for 3D reconstruction from a single RGB image. Features are extracted using a 2D CNN encoder, unprojected into 3D space, and decoded via MLP into an occupancy voxel grid.

4.2 Main Steps of the Pipeline

1. **Input Normalization:** The input RGB image is resized and normalized to a fixed resolution (e.g., 224×224). The coordinate system of the voxel space is also aligned with the camera viewpoint provided in the dataset annotations.

2. **3D Grid Generation:** A regular 3D voxel grid is generated to serve as the spatial domain for reconstruction. Each voxel in the grid is a candidate for occupancy prediction.

3. **Feature Extraction:** A 2D CNN encoder (such as ResNet18) extracts high-level spatial features from the input image. These features encode semantic and geometric information.

4. **Unprojection (View-based Fusion):** The 2D features are projected back into 3D space using known camera intrinsics and extrinsics (available in Pix3D annotations). This associates each voxel position with the corresponding feature vector from the image.

5. **Occupancy Prediction:** A fully connected decoder (MLP) predicts whether each voxel in the grid is occupied (value = 1) or empty (value = 0). The output is a 3D occupancy grid that defines the object shape.

4.3 Model Architecture: Encoder–Decoder

- **Encoder:** A convolutional neural network (ResNet18) is used to extract hierarchical features from the input RGB image. The final feature maps are downsampled representations encoding both texture and geometry.

- **Decoder:** A multilayer perceptron (MLP) takes as input the projected features along with the voxel coordinates and produces an occupancy score for each voxel. The output is a 3D binary volume indicating the shape of the object.

4.4 Training Details

- **Language:** Python

- **Framework:** PyTorch

- **Dataset:** Pix3D – includes RGB images, segmentation masks, and voxel ground truth in `.mat` format (converted to `.npy`)

- **Loss Function:** Binary Cross Entropy (BCE) is used to compare predicted and ground truth occupancy values at each voxel.
- **Optimizer:** Adam optimizer with a learning rate scheduler to ensure stable convergence.

4.5 Advantages of This Design

- The encoder–decoder structure is easy to extend and can incorporate multi-view or temporal features in future versions.
- Using voxel-based representation simplifies the output structure and allows direct supervision with ground truth.
- Leveraging camera parameters from the JSON metadata ensures spatial alignment between the image and 3D structure.

5 Discussion

5.1 Strengths and Limitations of the Model

Our 3D reconstruction model demonstrates promising results on common household objects such as chairs, tables, and beds. The encoder–decoder pipeline is lightweight and easy to implement, while still capable of learning meaningful voxel representations.

However, the model has several limitations:

- It performs poorly on uncommon objects or complex shapes not well represented in the training data.
- Due to the low resolution of voxel outputs (e.g., 32^3 or 64^3), fine geometric details are often lost or smoothed out.
- The model struggles to generalize to unseen categories or unseen camera angles, especially when there is only one view per object.

5.2 Error Analysis

A key limitation arises from the dataset itself: many object categories in Pix3D contain only a single image–voxel pair. This makes it difficult for the model to learn meaningful shape priors or to infer unseen viewpoints.

Additionally, incorrect voxel predictions often occur in parts of the object that are occluded or not clearly visible in the input image. Since the network cannot hallucinate geometry from missing views, it tends to produce incomplete or inaccurate reconstructions.

5.3 Component Contribution Analysis (Ablation Study)

Due to time constraints, a full ablation study was not performed. However, future work could investigate:

- The impact of adding a segmentation mask as an input channel

- Comparing ResNet18 versus deeper architectures (e.g., ResNet50, EfficientNet)
- Varying the resolution of voxel outputs and loss balancing

These insights can guide future improvement and help isolate which modules contribute most to model performance.

6 Conclusion and Future Work

6.1 Summary of Results

This project presented a full pipeline for reconstructing 3D shapes from a single 2D image, using a voxel-based representation and a ResNet18-based encoder. Our system leverages the Pix3D dataset and converts .mat voxel files into .npy format for easier training. Results show that the model can successfully learn object shapes in a supervised manner and generalize to some unseen instances.

6.2 Future Directions

To improve this system, several directions are proposed:

- **Data augmentation:** Increase the diversity of training data by adding synthetic views or augmenting camera poses.
- **Model improvements:** Use more advanced decoders such as 3D U-Net or Pix2Vox++, or adopt implicit functions for higher fidelity.
- **Interactive applications:** Extend this model into a web-based interface where users can upload a 2D image and visualize the reconstructed 3D model in real time.
- **Scene-level reconstruction:** Move beyond single-object reconstruction toward whole-scene understanding from images of indoor rooms.

7 References

References

[1] Sun, X., Wu, J., Zhang, X., Zhang, Z., Zhang, C., & Tenenbaum, J. B. (2018). *Pix3D: Dataset and methods for single-image 3D shape modeling*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2974–2983.

[2] Xu, W., Wang, C., Jiang, H., Cao, Y., Wang, C., & Li, Y. (2022). *ResFesT: Resolution-aware feature-fusion transformer for single image 3D reconstruction*. Proceedings of the European Conference on Computer Vision (ECCV), 276–293.

[3] Choy, C. B., Xu, D., Gwak, J., Chen, K., & Savarese, S. (2016). *3D-R2N2: A unified approach for single and multi-view 3D object reconstruction*. Proceedings of the European Conference on Computer Vision (ECCV), 628–644.

[4] Wu, J., Zhang, C., Xue, T., Freeman, B., & Tenenbaum, J. (2016). *Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling.* Advances in Neural Information Processing Systems (NeurIPS), 82–90.

[5] Tatarchenko, M., Dosovitskiy, A., & Brox, T. (2017). *Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs.* Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2088–2096.

[6] Xiang, Y., Mottaghi, R., & Savarese, S. (2016). *ObjectNet3D: A large scale database for 3D object recognition.* Proceedings of the European Conference on Computer Vision (ECCV), 160–176.

[7] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., & Geiger, A. (2019). *Occupancy Networks: Learning 3D reconstruction in function space.* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4460–4470.