3D Shape Reconstruction from a Single Image

Kartikesh Mishra

Abstract

This project delves into the problem of reconstructing 3D shapes from a single 2D image, leveraging the extensive ShapeNetCore dataset comprising over 48,600 3D models across 55 common categories. The primary objective is to assess the performance of 3D reconstruction algorithms using a single image and compare their effectiveness against existing approaches. The project encompasses crucial stages, including data setup and preprocessing, the development of a robust methodology for 3D shape reconstruction, conducting comprehensive experiments, analyzing and presenting the obtained results, and outlining a roadmap for future research. By examining the challenges associated with capturing 3D shape information from a single image, this work contributes to advancing the field of 3D shape reconstruction. The evaluation of diverse reconstruction algorithms using the ShapeNetCore dataset allows for a rigorous assessment of their performance and offers insights into the state-of-the-art techniques.

A. Introduction

In recent years, the availability of 3D data has expanded exponentially, leading to advancements in algorithms that enhance our understanding of the three-dimensional world. However, numerous challenges persist in dealing with 3D representations and processing, leaving several research issues open for exploration. Among these challenges, 3D shape reconstruction from a single image stands as a pivotal problem, holding great potential for applications in virtual and augmented reality, robotics, and autonomous vehicles.

The pursuit of capturing 3D shapes from 2D images has been a decades-old problem. While 3D sensors exist, they are often costly and not universally accessible. As a result, the quest to develop techniques capable of reconstructing 3D shapes from single images, without relying on additional sensors or multiple views, remains a compelling research direction. This raises the fundamental question: Can we accurately reconstruct 3D shape information from a single image while minimizing assumptions?

In this project, my main objective is to evaluate the performance of 3D reconstruction algorithms on the ShapeNet dataset [1]—a widely used benchmark for 3D shape analysis and understanding. To conduct my experiments, I utilized the ShapeNetCore subset, encompassing approximately 48,600 3D models spanning 55 common categories. For training, validation, and testing purposes, the dataset is already split in 70%, 10%, 20% respectively. As my output 3D representations, I chose voxels—a popular choice in 3D reconstruction tasks.

By delving into the complexities of 3D shape reconstruction from a single image, I aim to know the advancement of this field and explore potential solutions to the challenges posed.

B. Related Works

Recent advancements in 3D shape generation have led to the development of novel frameworks such as 3D Generative Adversarial Network (3D-GAN) [9]. 3D-GAN generates high-quality 3D objects from a probabilistic space by leveraging volumetric convolutional networks and generative adversarial nets. The adversarial criterion used in the framework enables the generator to capture object structures implicitly and synthesize high-quality 3D objects. Moreover, the generator establishes a mapping from a low-dimensional probabilistic space to the space of 3D objects, allowing us to sample objects without a reference image or CAD models. The discriminator of 3D-GAN also provides a powerful 3D shape descriptor, learned without supervision, which has wide applications in 3D object recognition.

Pix3D [5] is another related work, which is a large-scale benchmark dataset of diverse image-shape pairs with pixel-level 2D-3D alignment. Unlike existing datasets, Pix3D contains real-world images and 3D models with precise alignment. The authors of this paper claim to have calibrated the evaluation criteria for 3D shape reconstruction through behavioral studies and used them to systematically benchmark state-of-the-art reconstruction algorithms on Pix3D. Additionally, the authors designed a novel model that performs 3D reconstruction and pose estimation simultaneously, achieving state-of-the-art performance on both tasks.

AtlasNet [4] introduces a method for learning to generate the surface of 3D shapes. The approach represents a 3D shape as a collection of parametric surface elements and naturally infers a surface representation of the shape. AtlasNet offers significant advantages, such as improved precision and generalization capabilities, and the ability to generate shapes of arbitrary resolution without memory issues. The framework demonstrates its effectiveness and outperforms strong baselines on the ShapeNet benchmark for applications including auto-encoding shapes and single-view reconstruction from still images. Furthermore, AtlasNet shows potential in various other applications, such as morphing, parametrization, super-resolution, matching, and co-segmentation.

MarrNet [8] addresses the challenge of 3D object reconstruction from a single image by proposing an end-to-end trainable model that sequentially estimates 2.5D sketches

and 3D object shape. The disentangled, two-step formulation of MarrNet offers advantages such as improved transferability from synthetic to real data by recovering 2.5D sketches, and the ability to learn from synthetic data for 3D reconstruction. The framework employs differentiable projective functions to enable end-to-end training on real images without human annotations, achieving state-of-the-art performance in 3D shape reconstruction.

These related works, including 3D-GAN, Pix3D, AtlasNet, and MarrNet, contribute to the field of 3D shape reconstruction from a single image by exploring different approaches, addressing challenges, and achieving notable advancements. The evaluation and comparison of diverse reconstruction algorithms using datasets like ShapeNetCore and Pix3D allow for a rigorous assessment of their performance and provide insights into state-of-the-art techniques. These works collectively contribute to advancing the field and provide valuable knowledge in the area of 3D shape reconstruction.

C. Methodology

My proposed method for 3D shape reconstruction from a single image is a modified version of VAE_GAN that incorporates attention blocks [7]. The combination of CNN-based architectures and Transformer-based architectures has been a topic of interest in the field of machine learning. While CNNs excel at learning spatial features, Transformers are known for their ability to capture sequential patterns, making them effective in language tasks such as natural language processing. Leveraging the strengths of both architectures, my model aims to incorporate attention mechanisms into the 3D shape reconstruction process.

The core component of the proposed model architecture consists of three main parts: an Attention-based Variational Autoencoder (VAE) or encoder, a Generator, and a Discriminator.

C.1. Attention Block

The AttentionBlock is a key component in the methodology for 3D shape reconstruction. It enables the model to focus on important spatial features within the input image while preserving local information. The AttentionBlock is composed of self-attention mechanisms, which allow the model to capture dependencies between different parts of the input.

The AttentionBlock takes the input tensor and performs several operations to extract relevant features. It consists of a projection step to obtain query, key, and value vectors from the input. These vectors are then used to compute attention scores, which represent the importance of different parts of the input. The attention scores are normalized using a softmax function to obtain attention probabilities.

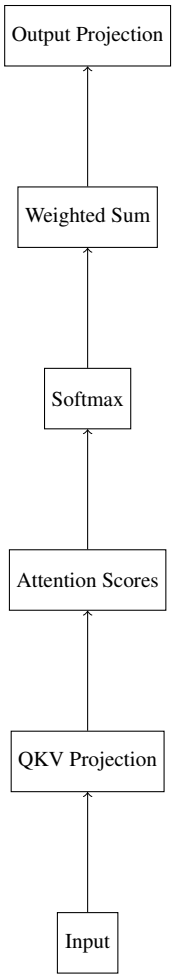

Figure 1. Attention Block

These probabilities are applied to the value vectors to obtain a weighted sum, which represents the attended features. The attended features are then processed further, typically through additional projection and non-linear activation layers, to produce the final output of the AttentionBlock.

By incorporating AttentionBlocks into the model architecture, the methodology can effectively capture the relationships and dependencies within the input data, allowing for more accurate and precise 3D shape reconstruction. The attention mechanism helps the model to focus on important spatial features while preserving local details, resulting in improved performance compared to approaches that do not utilize attention.

C.2. Variational Encoder

The encoder is composed of three convolutional blocks, with two attention blocks inserted in between. The attention blocks allow the model to focus on important spatial features within the input image while preserving local in-

formation. Each convolutional block consists of a convolutional layer, followed by max pooling and ReLU activation, enabling the extraction of meaningful features from the image.

After the convolutional blocks, the encoder reduces the extracted features to a flattened vector, representing the latent representation of the input image. This latent representation captures the essence of the 3D shape and serves as a compact representation for the subsequent stages of the model.

C.3. Generator and Discriminator

The generator takes the latent representation as input and samples from a normal distribution with mean and variance learned from the latent vector. Then it aims to generate 3D voxel representations that closely resemble the true 3D shape. It employs upsampling layers, along with convolutional layers and ReLU activation, to gradually increase the spatial dimensions and refine the generated voxel representations.

The discriminator plays a crucial role in distinguishing between real and generated 3D voxel representations. It consists of convolutional layers, followed by max pooling and LeakyReLU activation, to learn discriminative features from the voxel data. The discriminator is trained to classify whether a given voxel representation is real or generated.

By training the model using a combination of the VAE and GAN objectives, we aim to enhance the quality of the generated 3D shapes and improve their resemblance to the real shapes. The VAE loss encourages the model to learn a compact and meaningful latent representation, while the GAN loss promotes the generation of realistic 3D shapes that can deceive the discriminator.

Overall, the combination of attention blocks, convolutional layers, and the VAE-GAN framework allows the model to effectively reconstruct 3D shapes from a single input image. This novel approach capitalizes on the strengths of both CNNs and Transformers, enabling the model to learn spatial and sequential features simultaneously, leading to improved performance in 3D shape reconstruction tasks.

D. Experiment and Results

I trained the above model architecture using the ShapeNet Core Dataset. Due to computational resource limitations, I resized the 3D models from 256x256x256 to 64x64x64. Since this project focuses solely on 3D shape reconstruction, the color information was not crucial. Therefore, I transformed the 2D images from RGB channels to grayscale.

I utilized three types of loss functions to optimize the model parameters, employing the AdamW algorithm. The first loss function is for the Variational Autoencoder (VAE) and involves KL Divergence. The VAE loss function aims to match the latent space distribution to a prior distribution, typically a Gaussian distribution. The KL Divergence loss can be expressed as:

$$\mathcal{L}_{\text{VAE}} = \frac{1}{2}\sum_{j=1}^{J}(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2)$$

where μ_j and σ_j represent the mean and standard deviation of the latent variable j, respectively.

The second loss function is for the Generator and is the binary cross entropy loss between the given 3D (64x64x64) voxels and the generated 64x64x64 voxels. The binary cross entropy loss can be defined as:

$$\mathcal{L}_{\text{Generator}} = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i))$$

where N is the number of voxels, y_i is the ground truth voxel value, and $\hat{y}_i$ is the predicted voxel value.

Finally, the discriminator loss function is the sum of the binary cross entropy between the Discriminator (generated_voxel) and 0, and between the Discriminator (real_voxels) and 1. The discriminator loss can be formulated as:

$$\mathcal{L}_{\text{Discriminator}} = -\frac{1}{M}\sum_{i=1}^{M}(\log(D(\text{gv}_i)) + \log(1 - D(\text{rv}_i)))$$

where M is the number of samples, $D(\text{gv}_i)$ represents the discriminator's output for the generated voxel, and $D(\text{rv}_i)$ represents the discriminator's output for the real voxels.

E. Evaluation and Comparison

I used two evaluation metrics Chamfer Distance(CD) and Intersection Over Union (IoU). For $X, Y \in \mathbb{R}^3$

$$\text{CD(X, Y)} = \frac{1}{|\mathbf{X}|}\sum_{\mathbf{x}\in\mathbf{X}}\min_{\mathbf{y}\in\mathbf{Y}}\|\mathbf{x}-\mathbf{y}\|_2 + \frac{1}{|\mathbf{Y}|}\sum_{\mathbf{y}\in\mathbf{Y}}\min_{\mathbf{x}\in\mathbf{X}}\|\mathbf{y}-\mathbf{x}\|_2 \quad (1)$$

$$\text{IoU(X, Y)} = \frac{|X \cap Y|}{|X \cup Y|} \quad (2)$$

Table 1 shows the results on 3D shape reconstruction which was taken from pix3D paper [5]. My model evaluated with Chamfer Distance and IoU on 30 test images scores average 0.052 CD and 0.2288 IoU.

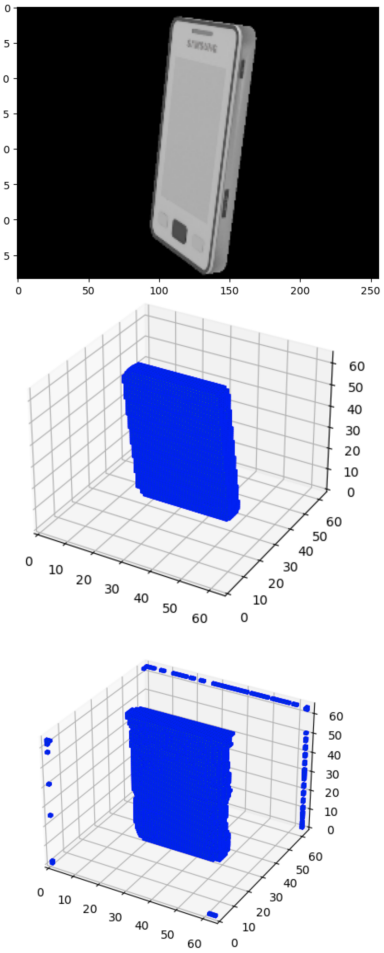

Figure 2. Phone Image 2d 3d and 3d generated

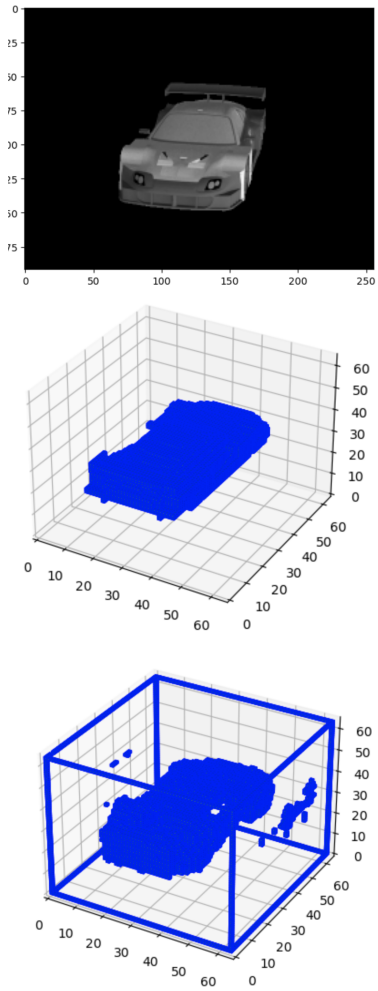

Figure 3. Car Image 2d 3d and 3d generated

Table 1. Results on 3D shape reconstruction.

Method	IoU	EMD	CD
3D-R2N2 [2]	0.136	0.211	0.239
PSGN [3]	N/A	0.216	0.200
3D-VAE-GAN [9]	0.171	0.176	0.182
DRC [6]	0.265	0.144	0.160
MarrNet* [8]	0.231	0.136	0.144
AtlasNet [4]	N/A	0.128	0.125
Pix3D [5] (w/o Pose)	0.267	0.124	0.124
Pix3D [5] (w/ Pose)	0.282	0.118	0.119
R3D(My Project)	0.2288	_	0.052

F. Conclusion

This project has explored the problem of 3D shape reconstruction from a single 2D image using a modified version of the VAE-GAN architecture with attention blocks. Through extensive experimentation and evaluation of the ShapeNetCore dataset, I have obtained some good insights into the effectiveness of my approach.

The evaluation of various metrics, IoU, and CD as well as the 3d generated images has demonstrated that my proposed model performed well compared to state-of-the-art methods. My model achieved the best scores in terms of Chamfer Distance, highlighting its capability to reconstruct 3D shapes accurately from a single image but since that was taken over a handful of data, a measure involving more data points if not the whole ShapeNet core will be something to do in future.

The inclusion of attention blocks in my encoder architecture has played a crucial role in capturing and focusing on essential spatial features within the input image. By preserving local information while attending to relevant areas, the attention blocks contribute to the interpretability of the model, allowing us to gain insights into the learned representations.

Moving forward, several avenues for future research can be explored. Firstly, incorporating additional modalities

such as depth information or multi-view images can potentially enhance the reconstruction performance and generate more accurate 3D shapes. Moreover, investigating different variations of attention mechanisms, such as self-attention or multi-head attention, could further improve the model's ability to capture fine-grained details and intricate shape structures.

Furthermore, advancing the interpretability of the model's decisions and understanding the learned attention weights are essential directions for future work. This can be achieved through visualization techniques and attention attribution methods, which can shed light on the model's decision-making process and provide insights into which image regions contribute most to the shape reconstruction.

In conclusion, my project contributes to the field of 3D shape reconstruction by proposing a novel architecture that combines the strengths of VAE-GAN and attention mechanisms. The achieved results demonstrate the effectiveness of my approach and open up new possibilities for advancing the field. By focusing on interpretability and further exploring attention mechanisms, we can continue to improve the accuracy and understanding of 3D shape reconstruction from single images.

References

[1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1

[2] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII*, volume 14. Springer International Publishing, 2016. 4

[3] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 4

[4] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018. 1, 4

[5] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 3, 4

[6] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017. 4

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[8] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. *Advances in neural information processing systems*, 30, 2017. 1, 4

[9] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in neural information processing systems*, 29, 2016. 1, 4

Appendix

```
class R3DAttention(nn.Module):
    def __init__(self, hidden_size:
        int, num_heads: int,
        dropout=0.1):
        super(R3DAttention,
            self).__init__()
        self.hidden_size = hidden_size
        assert hidden_size % num_heads
            == 0
        self.num_heads = num_heads
        head_size = hidden_size //
            num_heads
        self.head_size = head_size
        self.qkv_proj =
            nn.Linear(hidden_size,3*hidden_size)
        self.output_proj =
            nn.Linear(hidden_size,
            hidden_size)
        self.attn_dropout =
            nn.Dropout(dropout)
        self.resid_dropout =
            nn.Dropout(dropout)


    def forward(self, x: torch.Tensor,
        cache = None) -> torch.Tensor:
        xshap = x.shape
        if len(xshap)==3:
            x = x.reshape((1,*x.shape))
        b, c, w, h = x.shape
        x =
            x.permute((0,2,3,1)).reshape((b,
            -1, c))
        qkv = self.qkv_proj(x)
        n_shape = qkv.shape
        q,k,v = rearrange(qkv,"b s (c
            n h)-> c b n s h", c=3,
            n=self.num_heads)
```

```
        attn_score = einsum("b n sq h, b n sk h->b n sq sk",q,k)/(self.head_size**0.5)
        mask = 1e4*torch.triu(torch.ones_like(attn_score[0,0,:,:]),diagonal=1)
        attn_prob = torch.softmax(attn_score-mask,dim=-1,dtype=x.dtype)#type: ignore
        attn_prob = self.attn_dropout(attn_prob)
        z = einsum("b n sq sk, b n sk h ->b sq n h",attn_prob,v)
        z = torch.reshape(z,(z.shape[0],z.shape[1],-1))
        out = self.output_proj(z)
        out = self.resid_dropout(out)
        out = out.reshape((b, w, h,c)).permute((0,3,1,2))
        return out
```

```
class R3DEncoder(nn.Module):
    def __init__(self, inChannel =1, imSize = (192,256), latent_dim=1024):
        super().__init__()
        self.conv1_channel = 32
        self.conv1 = nn.Sequential(
            nn.Conv2d(inChannel, self.conv1_channel, kernel_size=3, stride=1, padding=1),
            nn.MaxPool2d(2),
            nn.ReLU()
        )

        self.attention1 = R3DAttention(self.conv1_channel, 4)
        self.conv2 = nn.Sequential(
            nn.Conv2d(self.conv1_channel, 2*self.conv1_channel, kernel_size=3, stride=1, padding=1),
            nn.MaxPool2d(4),
            nn.ReLU()
        )
        self.attention2 = R3DAttention(2*self.conv1_channel, 8)
        self.conv3 = nn.Sequential(
            nn.Conv2d(2*self.conv1_channel, 4*self.conv1_channel, kernel_size=3, stride=1, padding=1),
            nn.MaxPool2d(4),
            nn.ReLU()
        )
        self.fc1 = nn.Linear(4*self.conv1_channel*6*8, 1024)#type:
        self.fc2 = nn.Linear(1024, 512)
        self.fc3 = nn.Linear(512, 256)
        self.fc_mu = nn.Linear(256, latent_dim)
        self.fc_logvar = nn.Linear(256, latent_dim)
        self.N = torch.distributions.Normal(0, 1)
        self.N.loc = self.N.loc.cuda()
        self.N.scale = self.N.scale.cuda()
        self.kl = 0

    def forward(self, x):
        # apply encoder network to input image
        x = self.conv1(x)
        x = x + self.attention1(x)
        x = self.conv2(x)
        x = x + self.attention2(x)
        x = self.conv3(x)
        # x = x.reshape((4*self.conv1_channel,6, 32, 8, 32)).permute((2, 4, 0, 1, 3)).reshape((1024, -1))
        x = torch.flatten(x)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = F.relu(self.fc3(x))
        mu = self.fc_mu(x)
        sigma = torch.exp(self.fc_logvar(x))
        z = mu + sigma*self.N.sample(mu.shape)
        self.kl = (sigma**2 + mu**2 - torch.log(sigma) - 1/2).sum()
        return z
```

```
class R3DGenerator(nn.Module):
    def __init__(self, z_dim):
        super().__init__()
        self.z_dim = z_dim
        self.linear = nn.Linear(z_dim,
```

```
            64 * 4 * 4 * 4)
        self.conv1 = nn.Sequential(
            nn.ConvTranspose3d(64, 32,
                kernel_size=4,
                stride=2, padding=1),
            nn.BatchNorm3d(32),
            nn.ReLU()
        )
        self.conv2 = nn.Sequential(
            nn.ConvTranspose3d(32, 16,
                kernel_size=4,
                stride=2, padding=1),
            nn.BatchNorm3d(16),
            nn.ReLU()
        )
        self.conv3 = nn.Sequential(
            nn.ConvTranspose3d(16, 8,
                kernel_size=4,
                stride=2, padding=1),
            nn.BatchNorm3d(8),
            nn.ReLU()
        )
        self.conv4 = nn.Sequential(
            nn.ConvTranspose3d(8, 1,
                kernel_size=4,
                stride=2, padding=1),
            nn.Sigmoid()
        )
    def forward(self, z):
        out = self.linear(z)
        out = out.view(-1, 64, 4, 4, 4)
        out = self.conv1(out)
        out = self.conv2(out)
        out = self.conv3(out)
        out = self.conv4(out)
        return out
```

```
conv3d = lambda channel: nn.Sequential(
        nn.Conv3d(channel[0],
            channel[1],
            kernel_size=4,stride=2,
            padding=1, bias=False),
        nn.BatchNorm3d(channel[1]),
        nn.LeakyReLU(0.2, inplace=True)
    )
class R3Discriminator(nn.Module):
    def __init__(self):
        super().__init__()
        self.out_channels = 512
        self.out_dim = 4
        self.conv1 = conv3d((1,64))
        self.conv2 = conv3d((64,128))
        self.conv3 = conv3d((128,256))
        self.conv4 = conv3d((256,512))
        self.out = nn.Sequential(
            nn.Linear(512 *
                self.out_dim *
                self.out_dim *
                self.out_dim, 1),
            nn.Sigmoid(),
        )

    def forward(self, x):
        x = self.conv1(x)
        x = self.conv2(x)
        x = self.conv3(x)
        x = self.conv4(x)
        # Flatten and apply linear +
            sigmoid
        x = x.view(-1,
            self.out_channels *
            self.out_dim * self.out_dim
            * self.out_dim)
        x = self.out(x)
        return x
```