Compress3D: a Compressed Latent Space for 3D Generation from a Single Image

Bowen Zhang[1*], Tianyu Yang[2†], Yu Li[2], Lei Zhang[2], and Xi Zhao[1†]

[1] Xi'an Jiaotong University
[2] International Digital Economy Academy (IDEA)

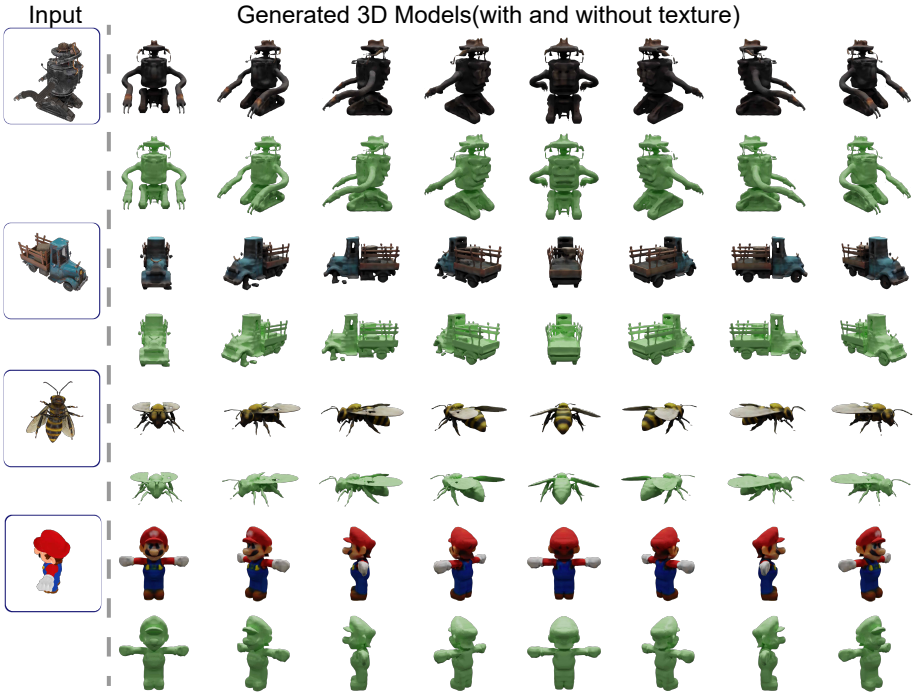

Fig. 1: Given a single-view image, our method can generate high-quality 3D Models.

Abstract. 3D generation has witnessed significant advancements, yet efficiently producing high-quality 3D assets from a single image remains challenging. In this paper, we present a triplane autoencoder, which encodes 3D models into a compact triplane latent space to effectively compress both the 3D geometry and texture information. Within the autoencoder framework, we introduce a 3D-aware cross-attention mechanism, which utilizes low-resolution latent representations to query features from a high-resolution 3D feature volume, thereby enhancing the

* Work done during the internship at IDEA.
† Corresponding authors.

representation capacity of the latent space. Subsequently, we train a diffusion model on this refined latent space. In contrast to solely relying on image embedding for 3D generation, our proposed method advocates for the simultaneous utilization of both image embedding and shape embedding as conditions. Specifically, the shape embedding is estimated via a diffusion prior model conditioned on the image embedding. Through comprehensive experiments, we demonstrate that our method outperforms state-of-the-art algorithms, achieving superior performance while requiring less training data and time. Our approach enables the generation of high-quality 3D assets in merely 7 seconds on a single A100 GPU. More results and visualization can be found on our project page: https://compress3d.github.io/.

Keywords: 3D Generation · Diffusion Model

1 Introduction

3D assets are widely used and have huge demand in the fields of gaming, AR/VR, and films. However, 3D modeling is a time-consuming and labor-intensive job and requires a long period of learning and mastering a variety of tools. Although there are already some image generation algorithms that can assist designers in 3D modeling, directly generating high-quality 3D assets is still challenging.

Benefiting from the emergence of the large-scale image-text pairs dataset LAION, image generation algorithms have made great progress in both generation quality and diversity. DreamFusion [26] proposed score distillation sampling(SDS) for the first time, and used pre-trained 2D diffusion models to guide the generation of 3D models. Subsequent works replace the 3D scene representation with DMtet or Gaussian Splatting and improve the optimization process, which speeds up the generation process and improves the mesh quality. Learning-based 3D generation is also a promising direction, and our method also falls into this category. There have been some works [5, 10, 21] training latent diffusion models on large-scale 3D datasets, achieving impressive results. However, none of these methods has a highly compressed latent space, which reduces the training speed and generation speed of latent diffusion. Moreover, current 3D generation methods use text or images as conditions to directly generate 3D models. However, these generated models usually do not conform to text or images, and the generated mesh geometry is low-quality.

To tackle the problems above, we propose a triplane autoencoder that takes colored point clouds as input to compress 3D model into a low-dimensional latent space on which a two-stage diffusion model is trained to generate 3D contents. [5, 21] directly project 3D point-wise features to triplanes through mean pooling. As this process involves no learnable parameters, it inevitably leads to the loss of 3D information. [5, 21] use UNet to further refine the triplane, which however greatly increases computation due to the high-resolution of triplanes. We instead add learnable parameters in the process of projecting 3D point cloud to 2D triplanes, which mitigates the information loss while avoiding significant

computational overhead. Specifically, we first convert 3D point-wise features into 3D feature volume and then use 3D convolution neural networks in 3 directions to obtain high-resolution triplane features. We use a series of ResNet blocks and downsample layers to get a low-resolution triplane. To further enhance the representation ability of latents, Shap-E [10] uses multi-view images as additional input and injects multi-view information via cross-attention. However, multi-view images lack accuracy in representing 3D information and computing attention weights between image patch embeddings and latent tokens consumes significant time, resulting in inefficiency in encoder training. In contrast, we leverage a 3D feature volume to augment the representation capability of triplane features. Specifically, we use triplane latent to query the 3D feature volume. This operation constitutes a local cross-attention mechanism that not only facilitates rapid computation but also significantly enhances the expressive capacity of triplane features.

Recovering 3D model from a single-view image is inherently an ill-posed problem. Instead of solely relying on image embedding for generating 3D, we propose leveraging both image embedding and shape embedding as conditions simultaneously for 3D content generation. Shape embedding inherently contains more 3D information compared to image embedding. Therefore, incorporating shape embedding as an additional condition for 3D generation is expected to yield better results than conditioning solely on image embedding. To obtain shape embedding during generation, we train a diffusion prior model to generate shape embedding conditioned on the image embedding. Specifically, we first use a pretrained shape-text-image alignment model OpenShape [17] to extract the shape embedding of 3D model and the image embedding of its corresponding rendering image. We then train a diffusion prior model that can estimate shape embedding conditioned on the corresponding image embedding. Since these embeddings are aligned in the same space, it is easy to learn a model to convert image embedding into shape embedding. Finally, we train a triplane latent diffusion model to generate triplane latent conditioned on the image embedding and the predicted shape embedding.

To summarize, our contributions are:

- We design an autoencoder capable of efficiently compressing 3D models into a low-dimensional triplane latent space and accurately decoding them back to high-quality colored 3D models.
- We introduce a triplane latent diffusion model that can be conditioned on both image embeddings and shape embeddings estimated from image embeddings, thereby facilitating the generation of 3D models.
- We conduct extensive ablations studies to verify the effectiveness of different components of our method and demonstrate that our method achieves high-quality 3D generation from a single image.

2 Related Work

2.1 Optimization-based Methods

Different from image generation, the size of datasets for 3D generation is much smaller than that of 2D generation. The largest 3D dataset Objaverse-XL [3] contains 10 million 3D objects, which is far smaller than LAION [30] that is used to train text-to-image generation models. To alleviate the problem of lacking 3D data, DreamFusion [26] proposes score distillation sampling (SDS), which enables the use of a 2D pre-trained diffusion model as a prior for 3D optimization. However, the optimization process takes around 2 hours for one 3D asset. Make-it-3D [39] incorporates constrain in the reference image and employs a two-stage optimization to achieve high-quality 3D generation. Magic3D [16] also adopts coarse to fine two-stage optimization, and it replaces the 3D scene representation from NeRF [22] to DMTet [31] in the refining stage, which allows it to efficiently render high-resolution images, greatly speeding up the optimization process and reducing the optimization time from 2 hours to 40 minutes. Recently, with the emergence of a new 3D scene representation Gaussian Splatting [11], there are also some works [1,38,42] that introduce this 3D representation into the field of optimization-based 3D generation. However, generating high-quality 3D assets using these optimization-based methods still takes several minutes.

2.2 Learning-based Methods

Limited by the scale of the 3D dataset, early learning-based 3D generation methods were limited to generating 3D geometry only. And there has been a large number of methods tried to explore generating point clouds [14, 41, 43], mesh [20,24,35] and signed distance field(SDF) [2,13,23,33,34,44,45]. Due to its sparse nature, point clouds are difficult to reconstruct fine 3D geometry. Computing the signed distance field requires preprocessing of the 3D mesh, and the geometry quality of the processed mesh will decrease. With the emergence of new 3D scene representations (NeRF [22], DMTet [31], Gaussian Splatting [11], FlexiCubes [32]) and large-scale 3D datasets, it is possible to replicate the successes of image generation in the field of 3D generation. Point-E [25] train a diffusion transformer with CLIP [28] image embedding as a condition on a large-scale 3D dataset to generate coarse colored point cloud, and then use a point cloud upsampler to upsamle coarse colored point cloud. Compared to optimization-based 3D generation methods, it is one to two orders of magnitude faster to sample from. However, since the generated point cloud contains only 4K points, it is difficult to reconstruct high-quality 3D mesh. To generate high-quality 3D mesh, Shpa-E [10] uses a transformer encoder to encode colored point cloud and multi-view images into parameters of an implicit function, through which mesh and neural radiance fields can be generated. Shpa-E then trains a conditional latent diffusion transformer to generate the parameters of the implicit function. Shap-E demonstrates the potential of latent representation in the field of 3D generation. Subsequent works [5, 21] also train the diffusion model on the latent

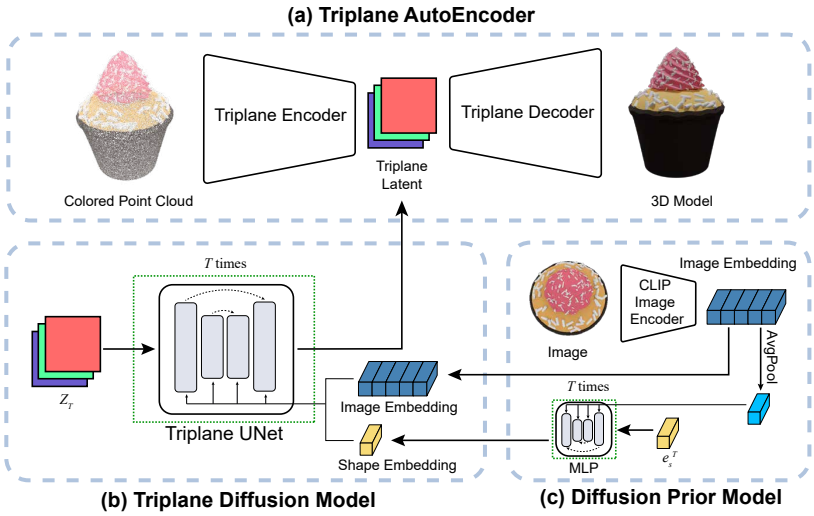

Fig. 2: Method overview. Compress3D mainly contains 3 components. (a) Triplane AutoEncoder: Triplane Encoder encodes color point cloud on a low-resolution triplane latent space. Then we use a Triplane Decoder to decode 3D model from a triplane latent. (b) Triplane Diffusion Model: we use shape embedding and image embedding as conditions to generate triplane latent. (c) Diffusion Prior Model: generate shape embedding conditioned on the image embedding.

space, but use DMTet [31] as the 3D scene representation, which improves the training speed and geometry quality. However, how to compress 3D model into a low-dimensional latent space is still an open problem.

2.3 Reconstruction-based Methods

There are also some methods that use 3D reconstruction techniques to generate 3D assets. Zero-1-to-3 [19] proposes that for a single-view image of a given object, images of other specific views of the object are generated through fine-tuning a 2D diffusion model, and then reconstruct 3D assets through the generated multi-view images. One-2-3-45 [18] further improves view consistency and reconstruction efficiency. LRM [9] and Instant3d [15] use a transformer to encode images into a triplane and use NeRF to reconstruct the 3D assets. Some recent work has introduced the gaussian splatting technique into the field of reconstruction-based 3D generation to achieve more efficient and high-quality reconstruction. [46] uses a hybrid triplane-gaussian intermediate representation for single-view reconstruction that efficiently generates a 3D model from a single image via feed-forward inference. More recently, LGM [37] proposes to encode multi-view images into multi-view gaussian features for high-quality 3D model generation.

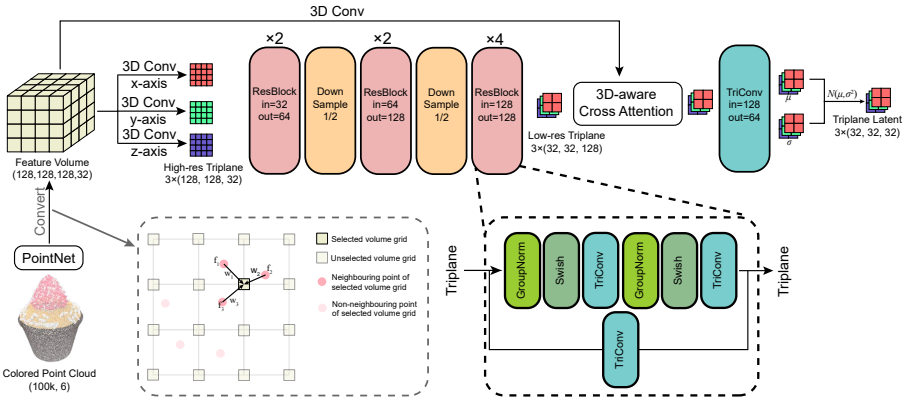

Fig. 3: Triplane Encoder. TriConv is the 3D-aware convolution proposed in [40].

3 Method

Our approach uses latent diffusion models to generate 3D assets from a single image. Instead of generating on the latent space of 3D models directly, we first generate shape embedding conditioned on the image embedding, then we generate triplane latent conditioned on both image embedding and previously generated shape embedding. The overview of our method is shown in Fig. 2.

Specifically, our method consists of three stages. In the first stage, we train a triplane variational autoencoder which takes as input the colored point clouds. The triplane encoder encodes 3D geometry and texture on a compressed triplane latent space. Subsequently, a triplane decoder reconstructs colored 3D model from the triplane latent space. In the second stage, we train a diffusion prior model to generate shape embedding conditioned on the image embedding. To obtain shape and image embedding pairs, we use OpenShape [17] to extract the shape embedding of 3D model and the image embedding of its rendered image. In the third stage, we train a triplane diffusion model to generate triplane latent conditioned on the image embedding and shape embedding.

3.1 Triplane AutoEncoder

Encoder The triplane encoder is shown in Fig. 3. The encoder takes colored point clouds as input and outputs a distribution on the triplane latent space. We represent the colored point cloud as $P \in \mathbb{R}^{N\times 6}$, where N is the number of points. The first three channels represent point 3D position (x, y, z) and the last three channels represent its corresponding (R, G, B) colors. We use PointNet [27] with position embedding and local max pooling as our point cloud encoder to extract 3D point-wise features. Then we project 3D point-wise features onto triplanes to achieve feature compression.

Previous methods [5, 21] directly project 3D point-wise features to triplanes through mean pooling, which inevitably leads to the loss of 3D information due to no learnable parameters in this process. Other works, such as 3DGen [5],

employ a UNet to further refine the triplane features and mitigate the loss of 3D information. However, incorporating an additional UNet does not compress the triplane and may increase computational demands. We instead add learnable parameters in this process. Specifically, given point-wise features $F = \{f_i \in \mathbb{R}^c\}_N$, the feature volume $V = \{v_j \in \mathbb{R}^c\}_{r\times r\times r} \in \mathbb{R}^{r\times r\times r\times c}$ is calculated as

$$v_j = \sum_{i\in\mathcal{N}(j)} w_i \cdot f_i \tag{1}$$

where r is the resolution of feature volume and c is the number of channels. $\mathcal{N}(j)$ is a set which contains the neighbor points indices of the jth feature volume grid, and $w_i = (1 - |p_j^x - p_i^x|)(1 - |p_j^y - p_i^y|)(1 - |p_j^z - p_i^z|)$ is a weight that is inversely proportional to the distance between p_i and p_j. The 2D illustration of the conversion is shown in Fig. 3. As the point cloud density is usually uneven, we need to normalize v_j to cancel out the impact of the point cloud density. We obtain the normalized feature volume $V^n = \{v_j^n \in \mathbb{R}^c\}_{r\times r\times r} \in \mathbb{R}^{r\times r\times r\times c}$ by,

$$v_j^n = \frac{v_j}{\sum_{i\in N(j)} w_i} \tag{2}$$

After obtaining normalized feature volume V^n, We employ 3D convolution in three directions to convolve the normalized feature volume and obtain high-resolution triplane features $T_{xy}, T_{yz}, T_{zx} \in \mathbb{R}^{r\times r\times c}$, respectively.

$$T_{xy} = \texttt{3DConv}(V^n, k = (1, 1, r), s = (1, 1, r)) \tag{3}$$

$$T_{yz} = \texttt{3DConv}(V^n, k = (r, 1, 1), s = (r, 1, 1)) \tag{4}$$

$$T_{zx} = \texttt{3DConv}(V^n, k = (1, r, 1), s = (1, r, 1)) \tag{5}$$

where k is the kernel size and s is the stride. Then the triplane features are passed through a series of ResBlocks and down-sample layers to obtain low-resolution triplane latents T_{xy}^l, T_{yz}^l, $T_{zx}^l \in \mathbb{R}^{r'\times r'\times c'}$.

To enhance the representation ability of triplane latents, we propose a 3D-aware cross-attention mechanism, which takes triplane features as queries to query features from 3D feature volume. The 3D-aware cross-attention computation process is shown in Fig. 4. We first use a 3D convolutional layer to down sample V^n to obtain a low-resolution feature volume $V_d^n \in \mathbb{R}^{r''\times r''\times r''\times c''}$.

$$V_d^n = \texttt{3DConv}(V^n, k = (o, o, o), s = (o, o, o)) \tag{6}$$

where o is the down-sample factor. Then, leveraging low-resolution triplane latents T_{xy}^l, T_{yz}^l, and T_{zx}^l, we employ 3D-aware cross-attention on the feature volume V_d^n to extract a residual feature. This residual feature is then added to the original triplane latent to compose the enhanced triplane latent.

$$(T_{xy}^e, T_{yz}^e, T_{zx}^e) = (A_{xy}, A_{yz}, A_{zx}) + (T_{xy}^l, T_{yz}^l, T_{zx}^l) \tag{7}$$

where T_{xy}^e, T_{yz}^e, T_{zx}^e are enhanced triplane latents. A_{xy}, A_{yz}, A_{zx} are the residual feature obtained by 3D-aware cross-attention. We empirically found that querying on low-resolution feature volume does not hurt the performance while saving

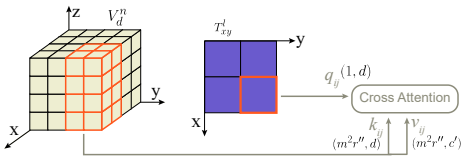

Fig. 4: 3D-aware cross attention. We use each point feature on the triplane to query the corresponding cube region (red) of feature volume. In addition, we add a position embedding to the volume feature.

lots of computation as shown in Table 3. To compute the residual features, we need first calculate the triplane queries Q_{xy}, Q_{yz}, $Q_{zx} \in \mathbb{R}^{r' \times r' \times d}$ and feature volume keys $K \in \mathbb{R}^{r'' \times r'' \times r'' \times d}$ and values $V \in \mathbb{R}^{r'' \times r'' \times r'' \times c'}$ by,

$$\begin{aligned}(Q_{xy}, Q_{yz}, Q_{zx}) &= \texttt{TriConv}((T_{xy}^l, T_{yz}^l, T_{zx}^l), k = (1,1), s = (1,1)) \\ K &= \texttt{3DConv}(V_d^n, k = (1,1,1), s = (1,1,1)) \\ V &= \texttt{3DConv}(V_d^n, k = (1,1,1), s = (1,1,1))\end{aligned} \tag{8}$$

where `TriConv` is the 3D-aware convolution proposed in [40]. For simplicity, we take A_{xy} as an example to illustrate 3D-aware cross-attention process. A_{yz}, A_{zx} can be calculated in a similar way. We define $Q_{xy} = \{q_{ij} \in \mathbb{R}^{1 \times d}\}_{r' \times r'}$ where q_{ij} is one point feature at position (i, j). We then extract its corresponding key and value by,

$$k_{ij} = K(mi : mi + m - 1, mj : mj + m - 1, :, :) \in \mathbb{R}^{m \times m \times r'' \times d} \tag{9}$$

$$v_{ij} = V(mi : mi + m - 1, mj : mj + m - 1, :, :) \in \mathbb{R}^{m \times m \times r'' \times c'} \tag{10}$$

where $m = \texttt{round}(\frac{r''}{r'})$ is the scale ratio between volume size and triplane size. We then reshape k_{ij} and v_{ij} to $\mathbb{R}^{m^2 r'' \times d}$ and $\mathbb{R}^{m^2 r'' \times c'}$ repectively for ease of attention computation. The cross-attention feature $A_{xy} = \{a_{ij} \in \mathbb{R}^{1 \times c'}\}_{r' \times r'}$ can be calculated by,

$$a_{ij} = \texttt{sotfmax}(\frac{q_{ij} k_{ij}^T}{\sqrt{d}}) v_{ij} \tag{11}$$

Decoder As shown in Fig. 5, the decoder consists of a series of ResBlocks and up-sample layers. The decoder is responsible for decoding the low-resolution triplane latent into a high-resolution triplane feature. The high-resolution triplane feature contains the geometry and texture information of the 3D model.

To recover geometry information from triplane features, we adopt FlexiCubes [32] representation, an isosurface representation capable of generating high-quality mesh with low-resolution cube grids. For each cube in FlexiCubes, we predict the weight, signed distance function (SDF), and vertex deformation at each cube vertex. Specifically, we concatenate the triplane features of the eight vertices of each cube and predict the cube weight using an MLP layer. Similarly,

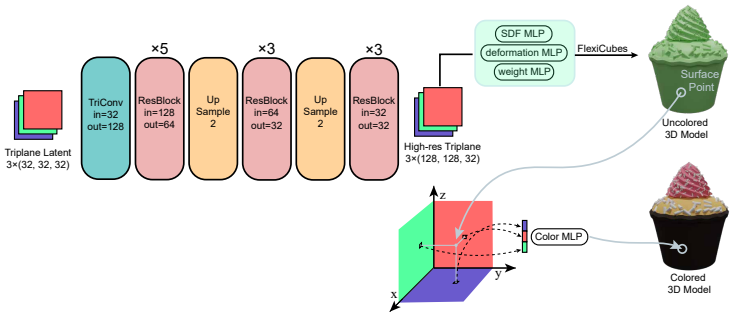

Fig. 5: Triplane Decoder.

we concatenate the triplane features of each vertex to predict the SDF and deformation using another 2 MLP layers. With the cube weights, SDF, and vertex deformations determined, the mesh can be extracted using the dual marching cubes method [29]. To recover texture information from the triplane features, we take the triplane features of the mesh surface points and predict the color of each surface point through an MLP layer.

Renderer We train the encoder and decoder using a differentiable renderer [12]. Compared with previous methods [23, 44, 45], we do not need to pre-compute the signed distance field of each 3D mesh, which demands a huge computation and storage space. Moreover, our method based on differentiable rendering also avoids information loss during data pre-processing. For the mesh output by the decoder, we first render the 3D model at a certain view and then compare it with the rendering images of the ground truth model from the same perspective. The rendering images contains RGB image I_{rgb}, silhouette image I_{mask} and depth image I_{depth}. Finally, we calculate the loss in the image domain and train the encoder and decoder jointly through rendering loss L_R. The rendering loss is as follows:

$$L_R = \lambda_1 L_{rgb} + \lambda_2 L_{mask} + \lambda_3 L_{depth} - \lambda_{kl} D_{KL}(N(\mu, \sigma)|N(0, 1)) \tag{12}$$

where $L_{rgb} = ||I_{rgb} - I^{gt}_{rgb}||^2$, $L_{mask} = ||I_{mask} - I^{gt}_{mask}||^2$, $L_{depth} = ||I_{depth} - I^{gt}_{depth}||^2$, $N(\mu, \sigma)$ is the distribution of the low resolution triplane latent. Moreover, we add KL penalty to ensure that the distribution of the triplane latent $N(\mu, \sigma)$ is close to the standard Gaussian distribution $N(0, 1)$.

3.2 Diffusion Prior Model

Generating a 3D model directly from an image is a difficult task because the image embedding of a single view image only contains 2D geometry and texture information of the 3D model. Compared to image embedding, shape embedding contains richer 3D geometry and texture information. Generating 3D model with

shape embedding as a condition is easier and more accurate than using image embedding as a condition. To train this diffusion prior model, we first use the OpenShape [17] model pre-trained on large-scale 3D dataset, a shape-text-image alignment model, to extract the shape embedding $e_s \in \mathbb{R}^{1280}$ of the 3D model and the image embedding $e_i \in \mathbb{R}^{1280}$ of the single-view rendering image. Then we design an MLP with skip connections between layers at different depths of the network as the diffusion backbone to generate shape embedding. This diffusion backbone consists of multiple MLP ResBlocks. In each block, image embedding is injected into the MLP block through concatenation, and the timestep embedding is injected through addition. Instead of using ϵ-prediction formulation as used in [7], we train our prior diffusion model to predict the denoised e_s directly with 1000 denoising steps, and use a L1 loss on the prediction:

$$L_{prior} = \mathbb{E}_{t\sim[1,T],e_s^{(t)}\sim q_t}[||f_\theta^p(e_s^{(t)}, t, e_i) - e_s||] \tag{13}$$

where f_θ^p is the learned prior model.

3.3 Triplane Diffusion Model

After we obtain the prior model, we then train a triplane diffusion model, which uses the shape embedding estimated by the prior model and image embedding as conditions, to generate 3D models. The diffusion backbone is a UNet, which contains multiple ResBlocks and down/up sample layers. The input and output of each ResBlock are triplanes, and we use 3D-aware convolution [40] in each ResBlock. Shape embedding e_s and image embedding e_p are injected into ResBlocks through cross attention. We train the triplane diffusion model to predict the noise ϵ added to the triplane latent with 1000 denoising steps, and use an L1 loss on the prediction,

$$L_{tri} = \mathbb{E}_{t\sim[1,T],\epsilon\sim N(0,1)}[||f_\theta(z^t, t, e_s, e_p) - \epsilon||] \tag{14}$$

where f_θ is the learned triplane diffusion model. To improve the diversity and quality of generated samples, we introduce classifier free guidance [8] by randomly dropout conditions during training. Specifically, we randomly set only $e_p = \varnothing_p$ for 5%, only $e_s = \varnothing_s$ for 5%, both $e_p = \varnothing_p$ and $e_s = \varnothing_s$ for 5%. During the inference stage, the score estimate is defined by,

$$\begin{aligned}\widetilde{f}_\theta(z^t, t, e_s, e_p) =& f_\theta(z^t, t, \varnothing_s, \varnothing_p) \\ &+ s_p \cdot (f_\theta(z^t, t, \varnothing_s, e_p) - f_\theta(z^t, t, \varnothing_s, \varnothing_p)) \\ &+ s_s \cdot (f_\theta(z^t, t, e_s, e_p) - f_\theta(z^t, t, \varnothing_s, e_p))\end{aligned} \tag{15}$$

4 Experiments

4.1 Dataset Curation

We train our model on a filtered Objaverse dataset [4]. As there are many low-quality 3D models in the origin Objaverse dataset. To obtain high-quality 3D

Table 1: Quantitative Comparison with other methods.

Metric	Shap-E [10]	OpenLRM [6]	LGM [37]	Ours
FID(↓)	146.14	86.93	88.64	**53.21**
CLIP Similarity(↑)	0.731	0.764	0.743	**0.776**
PSNR(↑)	14.54	14.36	13.23	**16.82**
LPIPS(↓)	0.350	0.335	0.381	**0.272**
Latent space dimension(↓)	1.05M	0.98M	-	**0.10M**
Seconds per shape(↓)	11	**5**	55	7
Training dataset size	≥1M	0.951M	**0.080M**	0.095M
Training time (A100 GPU hours)	-	9200	3072	**1900**

data for training, we manually annotated approximately 2500 3D models, categorizing them as either good or bad. A 'good' 3D model exhibits realistic textures and intricate geometric structures, whereas a 'bad' 3D model is characterized by single-color textures or simple shapes. We randomly select five random views and use the pre-trained CLIP model to extract their image embeddings. Then we concatenate these image embeddings and feed them into a shallow MLP network for classification. Despite the limited annotation data, we find that the trained MLP classification network can correctly classify 3D models in most cases. We use this MLP classification network to filter the entire Objaverse dataset and obtain 100k high-quality 3D models. We randomly select 95% 3D models for training and 5% for testing.

4.2 Training Details

Triplane AutoEncoder For the encoder, the number of input points N is 100k, the resolution r of the V_{norm} is 128, the resolution r'' of the V_d^n used in 3D-aware cross attention is 32. The resolution r' of the triplane latent is 32, and its channel number is 32. For the decoder, the decoded triplane has a resolution of 128, and its channel number is 32, we set the grid size of FlexiCubes as 90. For the Renderer, we render 512×512 RGB, mask and depth images from 40 random views to supervise the training process, and we set $\lambda_1 = 10$, $\lambda_2 = 10$, $\lambda_3 = 0.1$, $\lambda_{kl} = 1e^{-6}$ for the rendering loss. The triplane autoencoder has 32M parameters in total, and it is trained with the AdamW optimizer. The learning rate gradually decreases from 3×10^{-5} to 3×10^{-6}. We train it on 8 A100 GPUs for 6 days.

Diffusion Prior Model To stabilize the training process of the prior diffusion network, we scale the shape embedding e_s by 0.25, and image embedding e_i by 0.85, making their variance approximate to 1. The Diffusion Prior Model has 25.8M parameters, and we train it on 2 A100 GPUs for 18 hours. The learning rate gradually decreases from 1×10^{-5} to 1×10^{-6}.

Triplane Diffusion Model The triplane diffusion model has 864M parameters, We train the model on 8 A100 GPUs for 4 days. The learning rate gradually decreases from 3×10^{-5} to 3×10^{-6}.

4.3 Comparison with Other Methods

We compare our method with Shap-E [10] , OpenLRM [6] and LGM [37]. To generate 3D model efficiently, We use DDIM [36] sampler with 50 steps. The guidance scale for shape embedding and image embedding are 1.0 and 5.0 respectively.

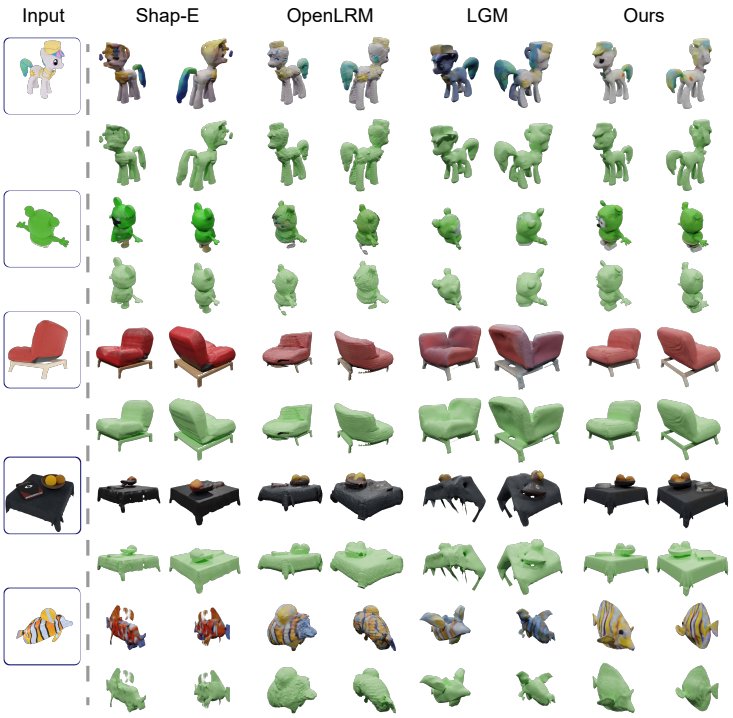

Fig. 6: Qualitative comparison with other methods.

Quantitative Comparison We use FID and CLIP similarity as evaluation metrics for generation quality. For the computation of FID, we randomly select 200 images in our test set that have not been seen during training, and generate 200 3D models using our method. Then we render each generated 3D model and its corresponding ground truth 3D model from 40 random views. We compute FID of the generated images set and ground truth image set. For the CLIP similarity, we calculate the cosine similarity of the CLIP image embedding of the generated 3D model and GT 3D model at the same viewpoint. We calculate FID and CLIP similarity five times and take the average. The quantitative comparison is reported in Table 1. Our method achieves lower FID and higher CLIP similarity than Shap-E and OpenLRM, while using less training data and time.
Qualitative Comparison The qualitative comparison is shown in Fig. 6. Compared with other methods, Compress3D can generate 3D models with good texture and fine geometric details. Benefiting from the two-stage generation,

our method can generate high-quality results under various viewing angles, while OpenLRM and Shpa-E are more sensitive to viewing angles. For example, OpenLRM and Shpa-E usually fail to generate 3D models with fine geometric details given top and bottom views as input. In addition, the up-axis of the 3D model generated by OpenLRM often does not coincide with the z-axis, which needs to be manually rotated to align with the z-axis This is time-consuming and laborious. In comparison, our method could generate 3D models whose up-axis coincides with the z-axis, which makes it easier to use.

4.4 Ablation Studies

To evaluate the design of our method, we conduct a series of ablation studies on several key designs.

3D-aware cross-attention. As described in Section 3.1, to enhance the representation ability of the triplane latent, we use triplane to query a feature volume via 3D-aware cross-attention. Table 2 shows that 3D-aware cross-attention improves the geometric and texture reconstruction quality greatly. Although the training time for each step increases slightly, from 0.789s to 0.824s, this is acceptable. As shown in Table 3, we find using a down-sampled feature volume in 3D-aware cross-attention improves reconstruction quality slightly and greatly decreases the training time.

Table 2: Ablation study on 3D-aware cross attention.

Method	$L_{rgb} \times 10^3$	$L_{mask} \times 10^3$	$L_{depth} \times 10^2$	$CD \times 10^2$	seconds per step
w/o atten	3.798	6.953	2.637	2.11	**0.789**
w atten	**2.485**	**5.059**	**2.095**	**1.64**	0.824

Table 3: Ablation study on volume resolution r'' used in 3D-aware cross attention.

Resolution	$L_{rgb} \times 10^3$	$L_{mask} \times 10^3$	$L_{depth} \times 10^2$	$CD \times 10^2$	seconds per step
128	2.551	5.234	2.187	1.72	2.295
64	2.497	5.134	2.124	1.67	0.961
32(ours)	**2.485**	**5.059**	**2.095**	**1.64**	**0.824**

Diffusion Prior Model. To validate the importance of diffusion prior model, we train a triplane diffusion model conditioned only on the image embedding and compare it with our method. As shown in Table 4, using prior model further improves the quality of generated 3D model. As shown in Figure 7, our method can still produce correct shapes under some unusual viewing angles, while the one without prior model fails.

Guidance scales. To increase the quality of generated 3D models, we adopt classifier-free guidance during inference. There are multiple combinations of guidance scales for shape embedding and image embedding. Overall, we find that an appropriate guidance scale for s_p or s_s can improve the generation quality. As

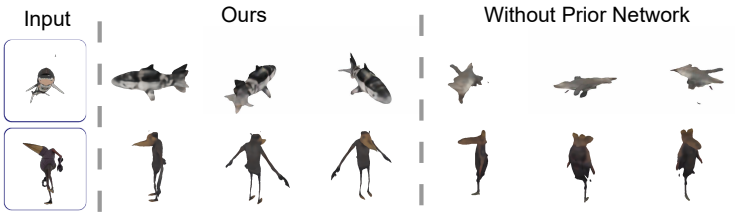

Fig. 7: Ablation Study: Compare our method with the version that do not use prior diffusion network.

Table 4: Ablation study on using diffusion prior model.

Method	FID(↓)	CLIP Similarity(↑)
w/o prior	66.46	0.745
w prior	**53.21**	**0.776**

shown in Table 5, when $s_p = 5.0$, $s_s = 1.0$, the model achieves the best FID. Although its CLIP similarity is slightly lower than the best one, they are very close.

Table 5: Ablation study on shape embedding guidance scale s_s and image embedding guidance scale s_p. The values are [FID/ CLIP similarity].

s_p \ s_s	1.0	3.0	5.0	10.0
1.0	65.18/0.75934	61.20/0.76435	57.05/0.76149	58.80/0.75882
3.0	55.09/0.77800	55.18/0.77538	53.60/**0.77803**	53.30/0.77494
5.0	**53.21**/0.77642	55.00/0.77524	53.43/0.77683	53.86/0.77343
10.0	54.82/0.77611	54.82/0.77543	54.63/0.77643	54.91/0.77689

5 Conclusion

This paper proposes a two-stage diffusion model for 3D generation from a single image, that was trained on a highly compressed latent space. To obtain a compressed latent space, we add learnable parameters in the projecting process from 3D to 2D, and we use 3D-aware cross-attention to further enhance the latent. Instead of generating latent conditioned solely on image embedding, we additionally condition on the shape embedding predicted by the diffusion prior model. Compress3D achieves high-quality generation results with minimal training data and training time, showcasing its versatility and adaptability across diverse scenarios.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (62072366, U23A20312).

References

1. Chen, Z., Wang, F., Liu, H.: Text-to-3d using gaussian splatting. arXiv preprint arXiv:2309.16585 (2023)
2. Cheng, Y.C., Lee, H.Y., Tulyakov, S., Schwing, A.G., Gui, L.Y.: Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4456–4465 (2023)
3. Deitke, M., Liu, R., Wallingford, M., Ngo, H., Michel, O., Kusupati, A., Fan, A., Laforte, C., Voleti, V., Gadre, S.Y., et al.: Objaverse-xl: A universe of 10m+ 3d objects. Advances in Neural Information Processing Systems **36** (2024)
4. Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., Farhadi, A.: Objaverse: A universe of annotated 3d objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13142–13153 (2023)
5. Gupta, A., Xiong, W., Nie, Y., Jones, I., Oğuz, B.: 3dgen: Triplane latent diffusion for textured mesh generation. arXiv preprint arXiv:2303.05371 (2023)
6. He, Z., Wang, T.: Openlrm: Open-source large reconstruction models. `https://github.com/3DTopia/OpenLRM` (2023)
7. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)
8. Ho, J., Salimans, T.: Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598 (2022)
9. Hong, Y., Zhang, K., Gu, J., Bi, S., Zhou, Y., Liu, D., Liu, F., Sunkavalli, K., Bui, T., Tan, H.: Lrm: Large reconstruction model for single image to 3d. arXiv preprint arXiv:2311.04400 (2023)
10. Jun, H., Nichol, A.: Shap-e: Generating conditional 3d implicit functions. arXiv preprint arXiv:2305.02463 (2023)
11. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
12. Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., Aila, T.: Modular primitives for high-performance differentiable rendering. ACM Transactions on Graphics **39**(6) (2020)
13. Li, M., Duan, Y., Zhou, J., Lu, J.: Diffusion-sdf: Text-to-shape via voxelized diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12642–12651 (2023)
14. Li, R., Li, X., Hui, K.H., Fu, C.W.: Sp-gan: Sphere-guided 3d shape generation and manipulation. ACM Transactions on Graphics (TOG) **40**(4), 1–12 (2021)
15. Li, S., Li, C., Zhu, W., Yu, B., Zhao, Y., Wan, C., You, H., Shi, H., Lin, Y.: Instant-3d: Instant neural radiance field training towards on-device ar/vr 3d reconstruction. In: Proceedings of the 50th Annual International Symposium on Computer Architecture. pp. 1–13 (2023)
16. Lin, C.H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.Y., Lin, T.Y.: Magic3d: High-resolution text-to-3d content creation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 300–309 (2023)
17. Liu, M., Shi, R., Kuang, K., Zhu, Y., Li, X., Han, S., Cai, H., Porikli, F., Su, H.: Openshape: Scaling up 3d shape representation towards open-world understanding. Advances in Neural Information Processing Systems **36** (2024)

18. Liu, M., Xu, C., Jin, H., Chen, L., Varma T, M., Xu, Z., Su, H.: One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. Advances in Neural Information Processing Systems **36** (2024)
19. Liu, R., Wu, R., Van Hoorick, B., Tokmakov, P., Zakharov, S., Vondrick, C.: Zero-1-to-3: Zero-shot one image to 3d object. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9298–9309 (2023)
20. Liu, Z., Feng, Y., Black, M.J., Nowrouzezahrai, D., Paull, L., Liu, W.: Meshdiffusion: Score-based generative 3d mesh modeling. arXiv preprint arXiv:2303.08133 (2023)
21. Mercier, A., Nakhli, R., Reddy, M., Yasarla, R., Cai, H., Porikli, F., Berger, G.: Hexagen3d: Stablediffusion is just one step away from fast and diverse text-to-3d generation. arXiv preprint arXiv:2401.07727 (2024)
22. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
23. Mittal, P., Cheng, Y.C., Singh, M., Tulsiani, S.: Autosdf: Shape priors for 3d completion, reconstruction and generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 306–315 (2022)
24. Nash, C., Ganin, Y., Eslami, S.A., Battaglia, P.: Polygen: An autoregressive generative model of 3d meshes. In: International conference on machine learning. pp. 7220–7229. PMLR (2020)
25. Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., Chen, M.: Point-e: A system for generating 3d point clouds from complex prompts. arXiv preprint arXiv:2212.08751 (2022)
26. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. arXiv preprint arXiv:2209.14988 (2022)
27. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
28. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
29. Schaefer, S., Warren, J.: Dual marching cubes: Primal contouring of dual grids. In: 12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings. pp. 70–76. IEEE (2004)
30. Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al.: Laion-5b: An open large-scale dataset for training next generation image-text models. Advances in Neural Information Processing Systems **35**, 25278–25294 (2022)
31. Shen, T., Gao, J., Yin, K., Liu, M.Y., Fidler, S.: Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. Advances in Neural Information Processing Systems **34**, 6087–6101 (2021)
32. Shen, T., Munkberg, J., Hasselgren, J., Yin, K., Wang, Z., Chen, W., Gojcic, Z., Fidler, S., Sharp, N., Gao, J.: Flexible isosurface extraction for gradient-based mesh optimization. ACM Transactions on Graphics (TOG) **42**(4), 1–16 (2023)
33. Shim, J., Kang, C., Joo, K.: Diffusion-based signed distance fields for 3d shape generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20887–20897 (2023)

34. Shue, J.R., Chan, E.R., Po, R., Ankner, Z., Wu, J., Wetzstein, G.: 3d neural field generation using triplane diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20875–20886 (2023)
35. Siddiqui, Y., Alliegro, A., Artemov, A., Tommasi, T., Sirigatti, D., Rosov, V., Dai, A., Nießner, M.: Meshgpt: Generating triangle meshes with decoder-only transformers. arXiv preprint arXiv:2311.15475 (2023)
36. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502 (2020)
37. Tang, J., Chen, Z., Chen, X., Wang, T., Zeng, G., Liu, Z.: Lgm: Large multi-view gaussian model for high-resolution 3d content creation. arXiv preprint arXiv:2402.05054 (2024)
38. Tang, J., Ren, J., Zhou, H., Liu, Z., Zeng, G.: Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. arXiv preprint arXiv:2309.16653 (2023)
39. Tang, J., Wang, T., Zhang, B., Zhang, T., Yi, R., Ma, L., Chen, D.: Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior. arXiv preprint arXiv:2303.14184 (2023)
40. Wang, T., Zhang, B., Zhang, T., Gu, S., Bao, J., Baltrusaitis, T., Shen, J., Chen, D., Wen, F., Chen, Q., et al.: Rodin: A generative model for sculpting 3d digital avatars using diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4563–4573 (2023)
41. Wu, L., Wang, D., Gong, C., Liu, X., Xiong, Y., Ranjan, R., Krishnamoorthi, R., Chandra, V., Liu, Q.: Fast point cloud generation with straight flows. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9445–9454 (2023)
42. Yi, T., Fang, J., Wu, G., Xie, L., Zhang, X., Liu, W., Tian, Q., Wang, X.: Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. arXiv preprint arXiv:2310.08529 (2023)
43. Zeng, X., Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., Kreis, K.: Lion: Latent point diffusion models for 3d shape generation. arXiv preprint arXiv:2210.06978 (2022)
44. Zhang, B., Nießner, M., Wonka, P.: 3dilg: Irregular latent grids for 3d generative modeling. Advances in Neural Information Processing Systems **35**, 21871–21885 (2022)
45. Zhang, B., Tang, J., Niessner, M., Wonka, P.: 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. arXiv preprint arXiv:2301.11445 (2023)
46. Zou, Z.X., Yu, Z., Guo, Y.C., Li, Y., Liang, D., Cao, Y.P., Zhang, S.H.: Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. arXiv preprint arXiv:2312.09147 (2023)