CLIP-Forge: Towards Zero-Shot Text-to-Shape Generation

Aditya Sanghi[1] Hang Chu[1] Joseph G. Lambourne[1] Ye Wang[2]
Chin-Yi Cheng[1] Marco Fumero[1] Kamal Rahimi Malekshan[1]
[1]Autodesk AI Lab [2]Autodesk Research
{aditya.sanghi,hang.chu,joseph.lambourne,ye.wang,chin-yi.cheng,marco.fumero,kamal.malekshan}@autodesk.com
https://github.com/AutodeskAILab/Clip-Forge

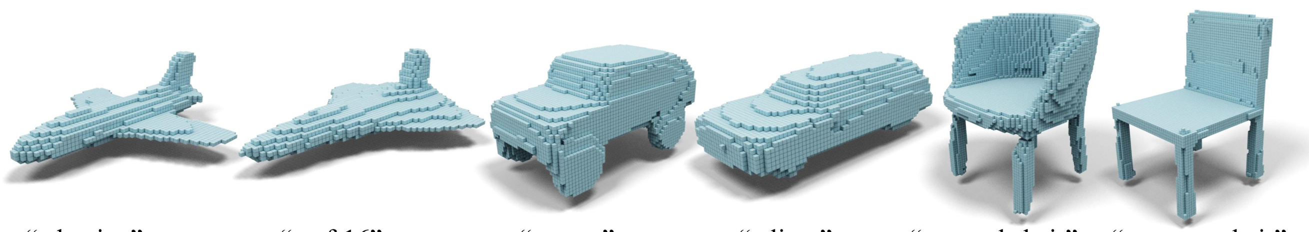

Figure 1. We propose a zero-shot text-to-shape generation method named CLIP-Forge. Without training on any shape-text pairing labels, our method generates meaningful shapes that correctly reflect the common name, (sub-)category, and semantic attribute information.

Abstract

Generating shapes using natural language can enable new ways of imagining and creating the things around us. While significant recent progress has been made in text-to-image generation, text-to-shape generation remains a challenging problem due to the unavailability of paired text and shape data at a large scale. We present a simple yet effective method for zero-shot text-to-shape generation that circumvents such data scarcity. Our proposed method, named CLIP-Forge, is based on a two-stage training process, which only depends on an unlabelled shape dataset and a pre-trained image-text network such as CLIP. Our method has the benefits of avoiding expensive inference time optimization, as well as the ability to generate multiple shapes for a given text. We not only demonstrate promising zero-shot generalization of the CLIP-Forge model qualitatively and quantitatively, but also provide extensive comparative evaluations to better understand its behavior.

1. Introduction

Generating 3D shapes from text input has been a challenging and interesting research problem with both significant scientific and applied value [19, 22, 23, 34]. In the artificial intelligence and cognitive science research communities, researchers have long sought to bridge the two modalities of natural language and geometric shape [3, 55]. In practice, text-to-shape generation models are a key enabling component to new smart tools in creative design and manufacture as well as animation and games [6].

Significant progress has been made to connect text and image modalities [10, 18, 26, 27, 52, 54]. Recently, DALL-E [49] and its associated pre-trained visual-textual embedding model CLIP [48] has shown promising results on the problem of text-to-image generation [44]. Notably, they have demonstrated strong zero-shot generalization while evaluated on tasks the model has not been specifically trained on. Shape generation is a more fundamental problem than image generation, because images are projections and renderings of the inherently 3D physical world. Therefore, one may wonder if the success in 2D can be transferred to the 3D domain. This turns out to be a non-trivial problem. Unlike the text-to-image case, where paired data is abundant, it is impractical to acquire huge paired datasets of texts and shapes.

Leveraging the progress of text-to-image generation, we present CLIP-Forge. As shown in Figure 2, we overcome the limitation of shape-text pair data scarcity via a simple and effective approach. We exploit the fact that 3D shapes can be easily and automatically rendered into images using standard graphics pipelines. We then utilize pre-trained image-text joint embedding models such as [25, 48], which bring text and image embeddings in a similar latent space so that they can be used interchangeably. Hence, we can train

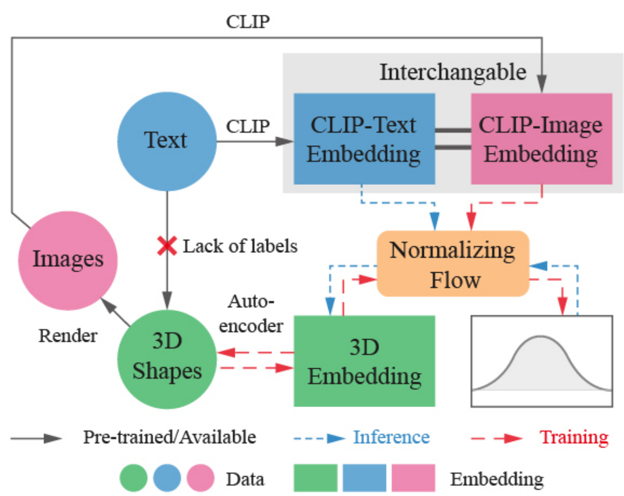

Figure 2. Illustration of the main idea. It is difficult to directly learn text-to-shape generation due to the lack of paired data. Instead, we use renderings of shapes with a pre-trained image-text joint embedding model to bridge the data gap between 3D shapes and natural language.

a model using image embeddings, but at inference time replace it with text embeddings.

In CLIP-Forge, we first obtain a latent space for shapes via training an autoencoder, and we then train a normalizing flow network [12] to model the distribution of shape embeddings conditioned on the image features obtained from the pre-trained image encoder [48]. We use the renderings of 3D shapes and hence, no labels are required for training our model. During inference, we obtain text features of the given text query via the pre-trained text encoder. We then condition the normalizing flow network with text features to generate a shape embedding, which is converted into 3D shape through the shape decoder. In this process, CLIP-Forge requires no text labels for shapes, which means it can be extended easily to larger datasets. Since our method is fully feed-forward, it also has the advantage of avoiding the expensive inference time optimizations as employed in existing 2D approaches [17, 56].

The main contributions of this paper are as follows:

- We present a new method, CLIP-Forge, that generates 3D shapes directly from text as shown in Figure 1, without requiring paired text-shape labels.
- Our method has an efficient generation process requiring no inference time optimization, can generate multiple shapes for a given text and can be easily extended to multiple 3D representation.
- We provide extensive evaluation of our method in various zero-shot generation settings qualitatively and quantitatively.

2. Related Work

Zero-Shot Learning. Zero-shot learning is an important paradigm of machine learning, which typically aims to make predictions on classes that have never been observed during training, by exploiting certain external knowledge source. The literature originates from the image classification problem [32, 41], and has been recently extended to generative models, in particular, the task of synthesizing images from text [49]. As far as our best knowledge, our method is the first to bring this paradigm to the 3D shape domain, which enables efficient shape generation from natural language text input.

Applications of CLIP. A major building block of our method is CLIP [48], which shows groundbreaking zero-shot capability using a mechanism to connect text and image by bringing them closer in the latent space. Previous work such as ALIGN [25], has used a similar framework on noisy datasets. Recently, pre-trained CLIP has been used for several zero-shot downstream applications [15, 17, 25, 40, 44, 53]. The most similar previous work to ours is zero-shot image and drawing synthesis [17, 44]. Typically, these methods involve iteratively optimizing a random image to increase certain CLIP activations. There is still no clear way to apply them to 3D due to the significantly higher complexity. Our approach conditions a shape prior network with CLIP features, which has the advantages of significant speed-up and the capability to generate multiple shapes from a single text.

3D Shape Generation and Language. Recently, there has been tremendous progress in 3D shape generating in different data formats such as point cloud [2, 33, 58], voxel [57], implicit representation [7, 36, 43] and mesh [38]. While our method is not limited to produce one 3D data format, we mainly adopt the implicit representation in this work due to their simplicity and superior quality. More recently, methods that use text to localize objects in 3D scene have been explored [1, 5]. A metric learning method for text-to-shape generation is presented in [6]. The main difference and advantage of our approach is its zero-shot capability which requires no text-shape labels.

Multi-Stage Training. In this work we follow a multi-stage training approach, where we first learn the embeddings of the target data and then learn a probabilistic encoding model for the learned embeddings. Such an approach as been explored in image generation [14, 35, 39] and 3D shape generation [2, 7]. Concretely for CLIP-Forge, we first train a 3D shape autoencoder and then model the embeddings using normalizing flow.

Normalizing Flow. Generative models have extensive use cases such as content creation and editing. Flow-based generative networks [11, 12, 50] is able to perform exact likelihood evaluation, while being efficient to sample from. They have been widely applied to a variety of tasks rang-

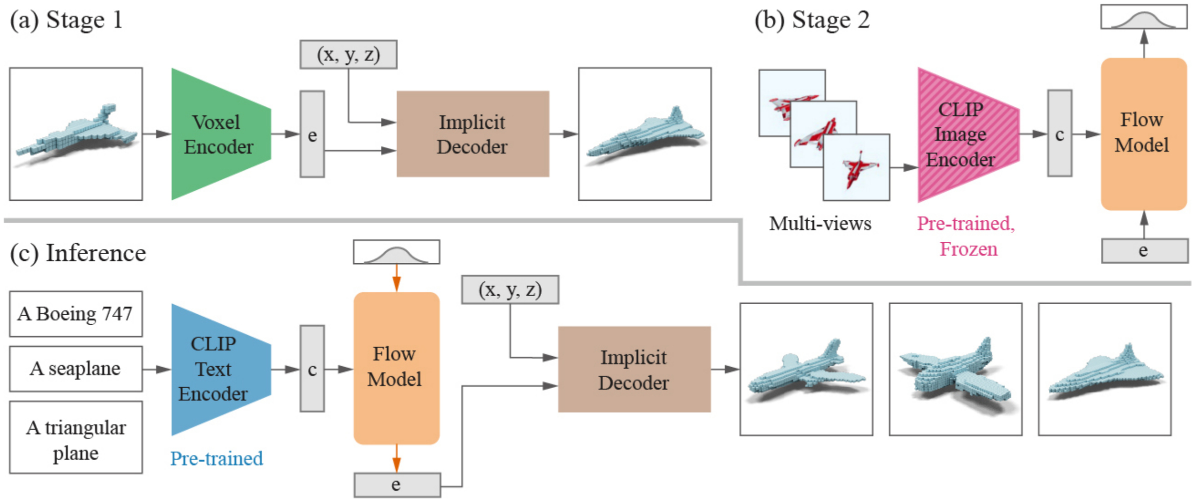

Figure 3. An overview of the CLIP-Forge method. The top row shows the two training stages of shape autoencoder training, and the conditional normalizing flow training. The bottom row shows how text-to-shape inference is conducted.

ing from image generation [30], audio synthesis [28] and video generation [31]. Recently, normalizing flow has been brought to the 3D domain enabling fast generation of point clouds [46,58]. In this paper, we employ a normalizing flow model [12] to model the conditional distribution of latent shape representations given text and image embeddings.

3. Method

Our method requires a collection of 3D shapes without any associated text labels, which takes the format of $\mathcal{S} = \{(\mathbf{I}_n, \mathbf{V}_n, \mathbf{P}_n, \mathbf{O}_n)\}_{n=1}^N$. Each shape in the collection $\mathcal{S}$ is comprised of a rendered image $\mathbf{I}_n$, a voxel grid $\mathbf{V}_n$, a set of query points in the 3D space $\mathbf{P}_n$, and space occupancies $\mathbf{O}_n$. As an overview, the CLIP-Forge training has two stages. In the first stage, we train an autoencoder with a voxel encoder and an implicit decoder. Once the autoencoder training is completed we obtain a shape embedding $\mathbf{e}_n$ for each 3D shape in $\mathcal{S}$. In the second stage, we train a conditioned normalizing flow network to model and generate $\mathbf{e}_n$, which is conditioned with image features obtained from the CLIP image encoder using $\mathbf{I}_n$. During inference, we first convert the text to the interchangeable text-image latent space using the CLIP text encoder. We then condition the normalizing flow network with the given text features and a random vector sampled from the uniform Gaussian distribution to obtain a shape embedding. Finally, this shape embedding is converted to a 3D shape using the implicit decoder. The overall architecture is shown in Figure 3.

3.1. Stage 1: Shape Autoencoder

The autoencoder consists of an encoder and a decoder. We use an encoder f_V to extract the shape embedding $\mathbf{e}_n$ for the training shape collection, using $\mathbf{V}_n$ of resolution 32^3 as the input. We use a simple voxel network that comprises of a series of batch-normalized 3D convolution layers followed by linear layers. This can be written as:

$$\mathbf{e}_n = f_V(\mathbf{V}_n) + \epsilon, \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, 0.1) \tag{1}$$

where $\mathbf{e}_n$ is augmented with a Gaussian noise. We find empirically injecting this noise improves the generation quality as later shown in the ablation study. This is also theoretically verified to improve results for conditional density estimation [51]. We then pass $\mathbf{e}_n$ through an implicit decoder. Our decoder architecture is inspired by the Occupancy Networks [36], which takes concatenated $\mathbf{e}_n$ and $\mathbf{P}_n$ as input. Our implicit decoder consists of linear layers with residual connections and predicts $\mathbf{O}_n$. We use a mean squared error loss between the predicted occupancy and the ground truth occupancy. Our framework is flexible and can be adapted to different forms of architectures. To showcase this, we use a PointNet [47] as the encoder and a FoldingNet [59] as the decoder that generates point clouds instead of occupancies, which are trained with a Chamfer loss [2].

3.2. Stage 2: Conditional Normalizing Flow

We train a normalizing flow network using $\mathbf{e}_n$ and its corresponding rendered images $\mathbf{I}_n$. Note that each $\mathbf{I}_n$ can include multiple images of the same shape from different rendering settings, such as changing camera viewpoints. We model the conditional distribution of $\mathbf{e}_n$ using a Real-NVP network [12] with five layers, which transforms the distribution of $\mathbf{e}_n$ into a normal distribution. We obtain the condition vector $\mathbf{c}_n$ by passing $\mathbf{I}_n$ through the ViT [13] based CLIP image encoder f_I, whose weights are frozen after pre-training. $\mathbf{c}_n$ is concatenated with the transformed feature vector at each scale and translation coupling layers of RealNVP:

$$\mathbf{c}_n = f_I(\mathbf{I}_n), \quad \mathbf{z}_n^{1:d} = \mathbf{e}_n^{1:d} \quad \text{and} \tag{2}$$

$$\mathbf{z}_n^{d+1:D} = \mathbf{e}_n^{d+1:D} \odot \exp\left(s([\mathbf{c}_n; \mathbf{e}_n^{1:d}])\right) + t([\mathbf{c}_n; \mathbf{e}_n^{1:d}]) \tag{3}$$

where s and t stand for the scale and translation function parameterized by a neural network. The intuition here is we split the object embedding $\mathbf{e}_n$ into two parts where one part is modified using a neural network that is simple to invert, but still dependent on the remainder part in a non-linear manner. The splitting can be done in several ways by using a binary mask [12]. In particular, we investigate two strategies: *checkerboard* masking and *dimension-wise* masking. The checkerboard masking has value 1 where the sum of spatial coordinates is odd, and 0 otherwise. The dimension-wise mask has value 1 for the first half of latent vector, and 0 for the second half. The masks are reversed after every layer. Finally, we impose a density estimation loss on the shape embeddings as:

$$\log\left(p(\mathbf{e}_n)\right) = \log\left(p(\mathbf{z}_n)\right) + \log\left(\left|\det\left(\frac{\partial f(\mathbf{e}_n)}{\partial \mathbf{z}_n^T}\right)\right|\right)$$

where f is the normalizing flow model, and $\partial f(\mathbf{e}_n)/\partial \mathbf{z}_n^T$ is the Jacobian of f at $\mathbf{e}_n$ [12]. We model the latent distribution $p(\mathbf{z}_n)$ as an unit Gaussian distribution.

3.3. Inference

During the inference phase, we convert a text query **t** into the text embedding using the CLIP text encoder, f_T. As the CLIP image and text encoders are trained to bring the image and text embeddings in a joint latent space, we can simply use the text embedding as the condition vector for the normalizing flow model, i.e. $\mathbf{c}$=$f_T(\mathbf{t})$. Once we obtain the condition vector we can sample a vector from the normal distribution and use the reverse path of the flow model to obtain a shape embedding in $p(\mathbf{e}_n)$. The normal distribution allows us to sample multiple times to obtain multiple shape embeddings for a given text query. We obtain the mean shape embedding by using the mean of the normal distribution. The mean shape embedding represents the prototype for a given text query. These shape embeddings are then converted to 3D shapes using the implicit decoder trained in stage 1.

4. Experiments

In this section, we first describe the experimental setup and then show qualitative and quantitative results. More results can be found in the supplementary material.

Dataset. For all of our experiments, we use the ShapeNet(v2) dataset [4] which consists of 13 rigid object classes. We use the processed version of the data which consists of rendered images, voxel grids, query points and their occupancies from shapes as provided in [9, 36].

Implementation Details. For both training stages, we use the Adam optimizer [29] with a learning rate of 1e-4 and a batch size of 32. We train the stage 1 autoencoder for 300 epochs whereas we train the stage 2 conditional normalizing flow model for 100 epochs. For all the experiments below we use a latent size of 128 with a BatchNorm [24] based voxel encoder and a ResNet based decoder inspired by the Occupancy Network [36]. We use a RealNVP [12] based network with dimension-wise masking for the flow model. The design decisions are discussed in the ablation study section and further details are provided in the supplemental material.

Evaluation Metrics. To evaluate our method thoroughly, we consider four criteria and several metrics for those criteria respectively. Furthermore, for some criteria we manually define a set of 234 text queries (or prompts). These queries include direct hyponyms for the ShapeNet categories from the WordNet [16] taxonomy, sub-categories and relevant shape attributes for a given category (e.g. a round chair, a square table, etc.) across the ShapeNet(v2) dataset. The text queries are listed in the appendix. The criteria are as follows:

1. **Reconstruction Quality.** This criteria is mainly used to check the reconstruction capabilities of the stage 1 autoencoder on the test set. We use two metrics: Mean Square Error (MSE) on 30,000 sample query points and Intersection over Union (IOU) with 32^3 voxel shapes.

2. **Generation Quality.** We use this criteria to evaluate the quality of generated shapes on text queries. We consider two metrics: Fréchet inception distance (FID) [21] and Maximum Measure Distance (MMD) using IOU . To calculate FID and MMD, we first take 224 text queries as mentioned above and generate a mean shape embedding for each text query. We then generate 32^3 resolution 3D objects for all the text queries. For FID, we compare the generated 3D shapes with the test dataset of ShapeNet. FID depends on a pre-trained network, for which we train a voxel classifier on the 13 ShapeNet classes and use the feature vector from the fourth layer. We provide more details in the appendix. In the case of MMD, for each generated shape we match a shape in the test dataset based on the highest IOU. We then average the IOU across all the text queries. Note, MMD is a variation of the Minimum Measure Distance as described in [2], which we believe is more suitable for implicit representations as we do not need to sample the surface.

3. **Diversity Across Categories.** To make sure we gen-

method	FID↓	MMD↑	Acc.↑
text2shape-CMA [6]	16078.05	0.4992	4.27
text2shape-supervised [6]	14881.96	0.1418	6.84
‘CLIP-Forge (ours)	**2425.25**	**0.6607**	**83.33**

Table 1. Comparing CLIP-Forge with supervised models using the text2shape dataset.

erate shapes across categories we design a new criteria. First, we generate the shapes based on the text queries as mentioned above. For each text query we have an assigned label. We then pass the generated voxels through the same classifier used to calculate the FID metric. We then report the accuracy based on the assigned label. We refer to this metric as Acc. throughout the text. Also note that the FID metric gives a good measure for diversity as we compare it with the test distribution.

4. **Human Perceptual Evaluation.** To evaluate CLIP-Forge's ability to provide control over the generated shape using attribute, common name, and sub-category information from the text prompt, we conducted a perceptual evaluation using Amazon SageMaker Ground Truth and crowd workers from Mechanical Turk [37]. More detail is provided in section 4.3.

4.1. Comparison with Supervised Models

We compare CLIP-Forge with text-to-shape generation models that are trained with direct supervision signals. The only existing paired text-shape dataset is provided by Text2Shape [6], which contains 56,399 natural language descriptions for ShapeNet objects within the chair and table categories. We train two supervised models using the Text2Shape dataset: text2shape-CMA uses the cross-modality alignment loss described in [6] and text2shape-supervised uses a direct MSE loss in the embedding space. For both supervised baseline methods, we use the same CLIP text encoder and occupancy network shape encoder and decoder to ensure fair comparison. Table 1 shows the result on our text query set. It can be seen that CLIP-Forge significantly outperforms both supervised baselines in all evaluation metrics. In particular, we observe that text2shape-CMA generates generic shapes such as boxes and spheres that do not resemble specific objects. The text2shape-supervised baseline fails to generalize and tends to generate chair- and table- like shapes that it is trained on, despite the text query is irrelevant to these two categories.

4.2. Qualitative Results

We qualitatively evaluate generative capabilities of our method. First, in Figure 4 we show that our network can generate multiple and diverse shapes using a single text query. This can be useful in a design process for imagining new variations. Next, we show that our network can generate shapes based on category, sub-category, common semantic words, and common shape attributes as shown in Figure 5. It can be seen that our network captures semantic notion of the text query. Finally, we show the generated shapes from interpolation between two text inputs in Figure 7. The interpolation results imply that the conditioning space is smooth.

4.3. Human Perceptual Evaluation

In this study we measure whether providing additional detail in the text prompt gives rise to semantically appropriate changes in the generated shape. To evaluate if the shape changes are semantically correct, we used human evaluators from Amazon Mechanical Turk [37]. The human evaluators were presented with pairs of images as shown in Figure 6(a). One image was generated using the ShapeNet(v2) category name (for example “a car”) while the other was generated using text which described a sub-category or shape attribute (for example “a truck” or “a round car”). The human evaluators were asked to identify which image best matched the sub-category or attribute text prompt. Each image pair was shown to 9 independent human evaluators. We record the fraction of image pairs for which more than half of the evaluators selected the image generated using the sub-category or attribute augmented prompt.

The results of the perceptual study are shown in Figure 6(b). The human evaluators correctly identified the model generated by the detailed prompt for 70.83% of the image pairs, showing that our method is able to utilize attribute and sub-category information in a way which is recognizable for humans. We see the attribute prompts produced shapes which were more easily identified than those from the sub-category prompts. One reason for this result is that the attribute augmented prompts give a clear description of how the object should look, while many of the sub-categories are less easily recognized given the quality of generation. For example “A circular bench” was correctly identified by 8/9 evaluators while “A laboratory bench” was not recognized by any of the 9 humans.

4.4. Choice of Prefix in Text Prompt

Designing a prompt can be challenging as small changes in words can potentially have a impact on our downstream task. In this experiment, we investigate how much does prompt selection effect the performance of our method. We specifically investigate what prefix to choose before a text query. The investigations are shown in Table 2. We find that prefix selection indeed has a effect on generation quality and diversity. A interesting avenue of future research would be to investigate prompt engineering [60].

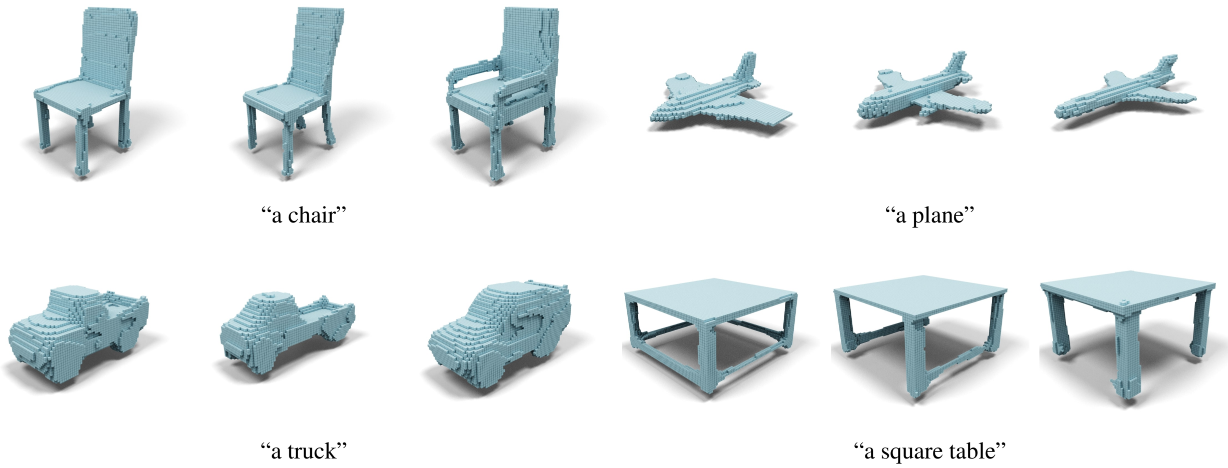

Figure 4. Our method can generate multiple examples given a text query. In this case, we are generating 3 shapes for a given text prompt.

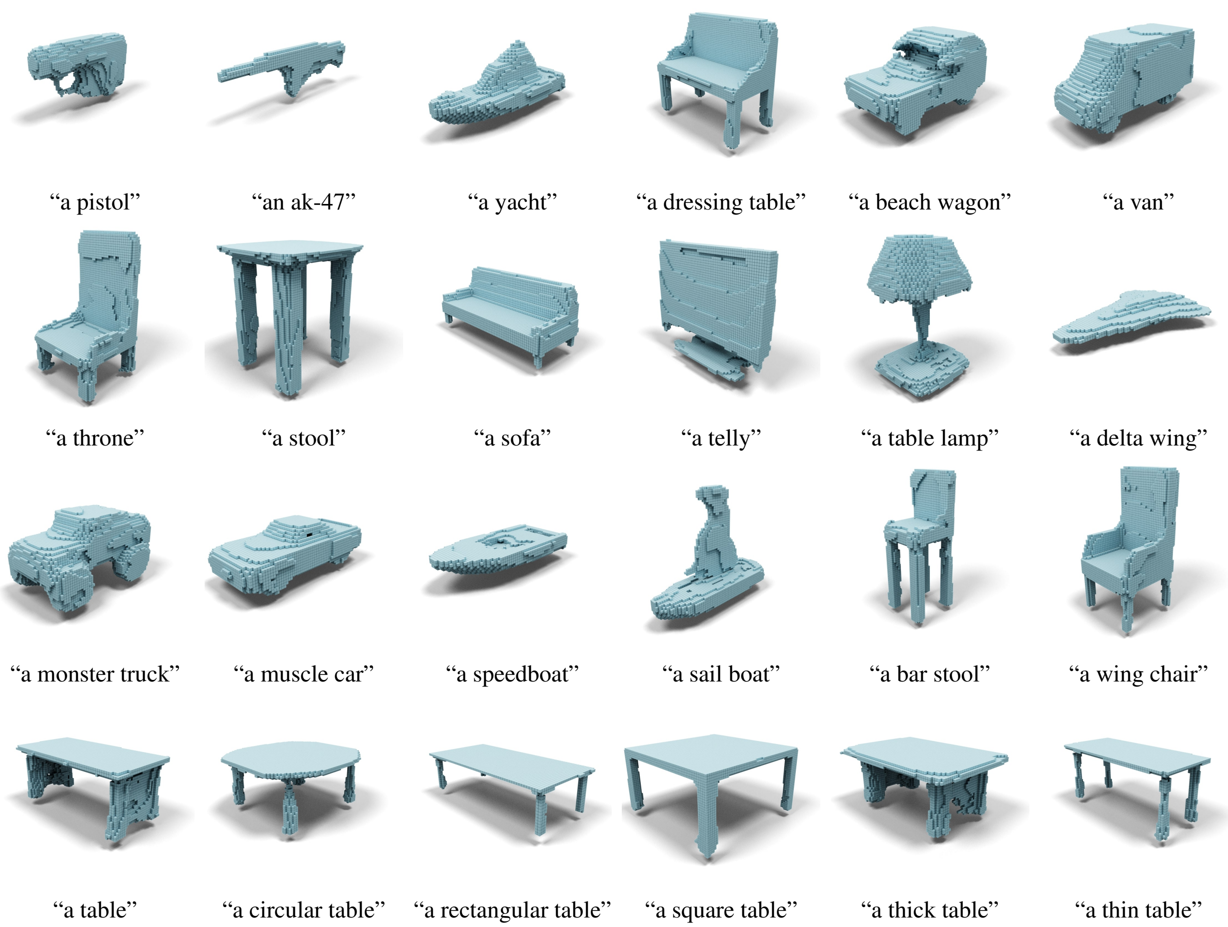

Figure 5. Illustrating our method can generate shapes via text containing common names, sub-category and attribute. The first two rows show shape generation based on using common names to describe an object. The next row shows two shapes for sub-category in car, boat and chair category. The final row illustrates the different shape attributes (circular, square, etc.) for the table class.

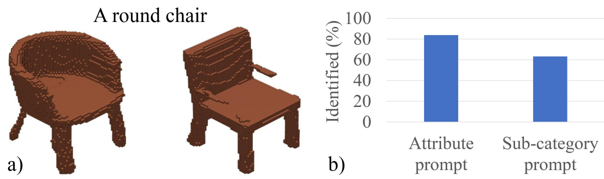

Figure 6. a) An example of an image pair and text prompt shown to the human evaluators. b) The percentage of image pairs for which the model generated by the text prompt was correctly identified.

prefix	FID↓	MMD↑	Acc.↑
"a"/"an"	2425.25	0.6607	83.33
"a photo of"	**2400.36**	0.6490	78.63
"a photo of a"	2484.49	0.6620	80.77
"a picture of a"	2560.98	**0.6681**	81.20
"a rendering of"	3029.92	0.6311	76.50
"a photo of one"	2715.45	0.6597	82.48
"one"	3142.07	0.6608	**87.18**

Table 2. Different Prompts and their effect.

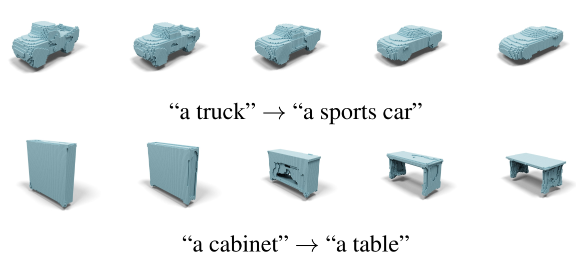

Figure 7. Interpolation results between two text queries.

4.5. CLIP-Forge for Point Cloud

In this section, we investigate if our method can be simply applied to a different representation, namely, a point cloud. As stated earlier we use the PointNet encoder [47] and FoldingNet decoder [59]. We use the same flow architecture as mentioned above. We train the network on the ShapeNet(v2) dataset. The results are shown in Fig 8. It can be seen that our method does a satisfactory job in generating 3D point clouds using text queries while using off the shelf point cloud encoders and decoders.

5. Ablation Studies

In this section, we discuss how different components of our algorithm affect our model. For all our ablation studies, we use the above mentioned hyperparameters for the autoencoder unless otherwise stated. For the flow model, we use RealNVP model with Checkerboard masking for most experiments unless otherwise stated.

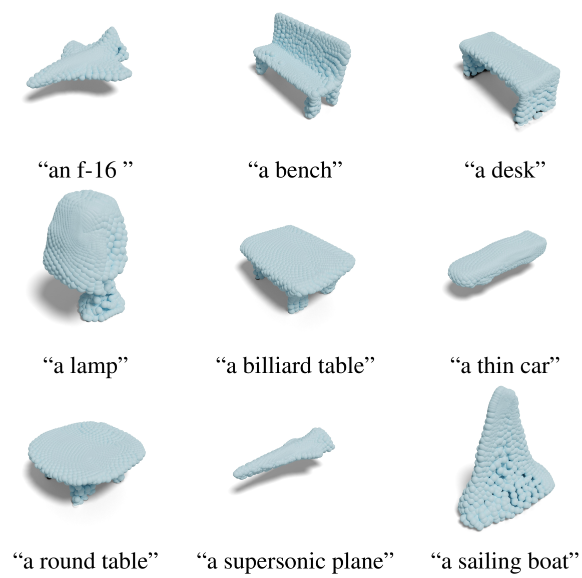

Figure 8. CLIP-Forge point cloud generations.

5.1. Stage 1 Autoencoder Design Choice

In Table 3, we experiment with different parts of the autoencoder architecture. The first subsection of Table 3 investigates how adding noise in the latent space helps our model. Empirically it can be seen from the table that adding noise not only helps the reconstruction but also improves the generation and diversity of the shapes generated. Next we investigate the size of the latent space and find that our model works reasonably well while using a smaller latent size of 128. Finally, we explore different encoders and decoders for our model. The results indicate that our model can take different representations, point cloud, as the input to the encoder. We provide more details regarding the encoder and decoder in appendix.

5.2. Stage 2 Prior Design Choice

In this section, we investigate the design choice for the prior network. First, we investigate different conditioning mechanisms, namely, conditioning the affine coupling layers and conditioning the prior network. From Table 4, it can be seen the choice of conditioning matters and conditioning the affine layers is the most effective. This intuitively makes sense as we are conditioning multiple times as we are concatenating each coupling layer whereas we just condition the prior once. A similar phenomena is observed in the case of an architectures like [7,43], where they concatenated the condition vector multiple times.

In Table 4, we also investigate different masking techniques (Dimension masking and Checkered masking) [12] and a distinct flow architecture: Masked Autoregressive

noise	*latent*	*encoder*	*decoder*	**IOU↑**	**MSE↓**	**FID↓**	**MMD↑**	**Acc.↑**
×	128	VoxEnc	RN-OccNet	0.7275	0.01120	3871.48	0.6559	71.94
✓				0.7374	0.01159	2688.72	0.6732	**79.34**
✓	256	VoxEnc	RN-OccNet	0.7375	0.01158	3177.92	0.6535	78.77
	512			0.7362	0.01155	3577.72	0.6374	74.50
✓	128	PointNet	RN-OccNet	0.7082	0.01051	**2646.93**	**0.6746**	76.50
		ResVoxEnc	RN-OccNet	0.7371	0.01075	3146.94	0.6509	74.64
		VoxEnc	CBN-OccNet	**0.7674**	**0.01025**	2956.78	0.6645	78.77

Table 3. Effects of different autoencoder design choices in stage 1, including the usage of Gaussian noise, the latent vector size, as well as various encoder and decoder architectures.

condition	*prior*	**FID↓**	**MMD↑**	**Acc.↑**
affine coupling	RealNVP-C	2688.72	0.6732	79.34
prior network		5227.32	0.6600	62.39
affine coupling	RealNVP-D	**2591.87**	**0.6751**	**82.19**
	MAF [42]	6052.62	0.6273	59.40

Table 4. Effects of different conditional normalizing flow design choices in stage 2.

Flow (MAF) [42]. It can be seen from the table that both masking techniques are effective but Dimension masking (RealNVP-D) seems to be more effective than Checkered masking (RealNVP-C). Furthermore, we find that MAF flow prior network is not as effective as RealNVP. For the remainder of the ablation studies we use the Dimension Masking for RealNVP.

5.3. Number of Renderings

Next, we evaluate if using more views helps the generation quality and diversity. We report the results in Table 5. The views are randomly selected from the renderings as prepared in [9]. It can be seen that using more views in general help improve the generation quality and diversity. As we are using a pre-trained CLIP model which is trained on natural images from different viewpoints, training using multiple views of shape renderings allows us to better capture the output distribution of CLIP model.

5.4. CLIP Architecture

In this section, we evaluate using different CLIP models to see how increasing the size of CLIP model and using ResNet [20] or ViT [13] based clip model effects our downstream task. We empirically observe from Table 5 that increasing the size of the model, i.e from ViT-B/32 to ViT-B/16, does not effect the text based generation too much. A more surprising result is that ResNet based CLIP model performs inferior to Visual Transformers. We hypothesize that patch based methods such as ViT focus more on the foreground object rather than the background. This is especially helpful in the case of image renderings.

rendering	*CLIP*	**FID↓**	**MMD↑**	**Acc.↑**
1	ViT-B/32	2983.72	0.6586	79.77
5		2776.28	0.6655	80.20
10		2622.71	0.6652	80.63
20	ViT-B/32	2591.87	**0.6751**	**82.19**
	ViT-B/16	**2515.81**	0.6573	80.48
	RN50x16	2906.75	0.6591	75.93

Table 5. Effects of different numbers of renderings and CLIP architectures.

6. Limitations and Future Work

We believe our method can be improved in several ways. Firstly, the quality of generation is still lacking and we believe a novel future avenue would be to combine ideas from local implicit methods [8, 45]. Furthermore, our work currently focuses on geometry and it would be interesting to integrate texture to our model. Finally, we are limited by CLIP's trained data distribution and a potential future direction would be to fine tune it for a specfic dataset.

In terms of potential negative impact, language-driven 3D modeling tools enabled by CLIP-Forge might lower the technical barriers to 3D modeling and potentially reduce some tedious 3D modeling tasks for 3D modelers and animators. However, it brings a greater benefit of democratizing 3D content creation to the general public, similar to that everyone can take photos and make videos today.

7. Conclusion

We presented a method, CLIP-Forge, that can efficiently generate multiple 3D shapes while preserving the semantic meaning from a given text prompt. Our method requires no text-shape labels as training data, offering an opportunity to leverage shape-only datasets such as ShapeNet. Finally, we showed that our model can generate results on other representations such as point clouds and we thoroughly studied different components of the method.

References

[1] Panos Achlioptas, Ahmed Abdelreheem, Fei Xia, Mohamed Elhoseiny, and Leonidas Guibas. Referit3d: Neural listeners for fine-grained 3d object identification in real-world scenes. In *European Conference on Computer Vision*, pages 422–440. Springer, 2020. 2

[2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018. 2, 3, 4

[3] Panos Achlioptas, Judy Fan, Robert X. D. Hawkins, Noah D. Goodman, and Leonidas J. Guibas. Learning to refer to 3d objects with natural language. 2018. 1

[4] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository, 2015. cite arxiv:1512.03012. 4

[5] Dave Zhenyu Chen, Angel X Chang, and Matthias Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 202–221. Springer, 2020. 2

[6] Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. In *Asian Conference on Computer Vision*, pages 100–116. Springer, 2018. 1, 2, 5

[7] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 7

[8] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020. 8

[9] 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 *European conference on computer vision*, pages 628–644. Springer, 2016. 4, 8

[10] Karan Desai and Justin Johnson. Virtex: Learning visual representations from textual annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11162–11173, 2021. 1

[11] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation, 2015. 2

[12] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 2, 3, 4, 7

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3, 8

[14] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12873–12883, 2021. 2

[15] Han Fang, Pengfei Xiong, Luhui Xu, and Yu Chen. Clip2video: Mastering video-text retrieval via image clip. *arXiv preprint arXiv:2106.11097*, 2021. 2

[16] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. The MIT Press, 05 1998. 4

[17] Kevin Frans, L. B. Soros, and Olaf Witkowski. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders, 2021. 2

[18] Andrea Frome, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. 2013. 1

[19] Zhizhong Han, Mingyang Shang, Xiyang Wang, Yu-Shen Liu, and Matthias Zwicker. Y2seq2seq: Cross-modal representation learning for 3d shape and text by joint reconstruction and prediction of view and word sequences. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):126–133, Jul. 2019. 1

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 8

[21] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 4

[22] Pin-Hao Huang, Han-Hung Lee, Hwann-Tzong Chen, and Tyng-Luh Liu. Text-guided graph neural networks for referring 3d instance segmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(2):1610–1618, May 2021. 1

[23] Faria Huq, Anindya Iqbal, and Nafees Ahmed. Holistic static and animated 3d scene generation from diverse text descriptions. *CoRR*, abs/2010.01549, 2020. 1

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 4

[25] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. *arXiv preprint arXiv:2102.05918*, 2021. 1, 2

[26] Armand Joulin, Laurens Van Der Maaten, Allan Jabri, and Nicolas Vasilache. Learning visual features from large weakly supervised data. In *European Conference on Computer Vision*, pages 67–84. Springer, 2016. 1

[27] Andrej Karpathy, Armand Joulin, and Li Fei-Fei. Deep fragment embeddings for bidirectional image sentence mapping. *arXiv preprint arXiv:1406.5679*, 2014. 1

[28] Sungwon Kim, Sang gil Lee, Jongyoon Song, Jaehyeon Kim, and Sungroh Yoon. Flowavenet : A generative flow for raw audio, 2019. 3

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4
[30] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions, 2018. 3
[31] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. Videoflow: A conditional flow-based model for stochastic video generation, 2020. 3
[32] Christoph Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. 06 2009. 2
[33] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point cloud gan. *arXiv preprint arXiv:1810.05795*, 2018. 2
[34] Angela S. Lin, Lemeng Wu, Rodolfo Corona, Kevin W. H. Tai, Qixing Huang, and Raymond J. Mooney. Generating animated videos of human activities from natural language descriptions. 2018. 1
[35] Jinlin Liu, Yuan Yao, and Jianqiang Ren. An acceleration framework for high resolution image synthesis. *arXiv preprint arXiv:1909.03611*, 2019. 2
[36] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 3, 4
[37] Abhishek Mishra. Machine learning in the aws cloud: Add intelligence to applications with amazon sagemaker and amazon rekognition, 2019. 5
[38] Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. Polygen: An autoregressive generative model of 3d meshes. *ICML*, 2020. 2
[39] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017. 2
[40] Daniil Pakhomov, Sanchit Hira, Narayani Wagle, Kemar E Green, and Nassir Navab. Segmentation in style: Unsupervised semantic image segmentation with stylegan and clip. *arXiv preprint arXiv:2107.12518*, 2021. 2
[41] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. 2
[42] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017. 8
[43] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 7
[44] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. *arXiv preprint arXiv:2103.17249*, 2021. 1, 2
[45] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer, 2020. 8
[46] Albert Pumarola, Stefan Popov, Francesc Moreno-Noguer, and Vittorio Ferrari. C-flow: Conditional generative flow models for images and 3d point clouds, 2020. 3
[47] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 3, 7
[48] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 1, 2
[49] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021. 1, 2
[50] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015. 2
[51] Jonas Rothfuss, Fabio Ferreira, Simon Boehm, Simon Walther, Maxim Ulrich, Tamim Asfour, and Andreas Krause. Noise regularization for conditional density estimation. *arXiv preprint arXiv:1907.08982*, 2019. 3
[52] Mert Bulent Sariyildiz, Julien Perez, and Diane Larlus. Learning visual representations with caption annotations. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pages 153–170. Springer, 2020. 1
[53] Sheng Shen, Liunian Harold Li, Hao Tan, Mohit Bansal, Anna Rohrbach, Kai-Wei Chang, Zhewei Yao, and Kurt Keutzer. How much can clip benefit vision-and-language tasks? *arXiv preprint arXiv:2107.06383*, 2021. 2
[54] Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218, 2014. 1
[55] Chuan Tang, Xi Yang, Bojian Wu, Zhizhong Han, and Yi Chang. Part2word: Learning joint embedding of point clouds and text by matching parts to words, 2021. 1
[56] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018. 2
[57] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 82–90, Red Hook, NY, USA, 2016. Curran Associates Inc. 2
[58] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud

generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4541–4550, 2019. 2, 3

[59] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018. 3, 7

[60] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *arXiv preprint arXiv:2109.01134*, 2021. 5