ZEROFORGE: FEEDFORWARD TEXT-TO-SHAPE WITHOUT 3D SUPERVISION

Kelly O. Marshall*, **Minh Pham***, **Ameya Joshi***, **Anushrut Jignasu**†, **Aditya Balu**†,
Adarsh Krishnamurthy†, **Chinmay Hegde***
*New York University †Iowa State University

ABSTRACT

Current state-of-the-art methods for text-to-shape generation either require supervised training using a labeled dataset of pre-defined 3D shapes, or perform expensive inference-time optimization of implicit neural representations. In this work, we present ZeroForge, an approach for zero-shot text-to-shape generation that avoids both pitfalls. To achieve open-vocabulary shape generation, we require careful architectural adaptation of existing feed-forward approaches, as well as a combination of data-free CLIP-loss and contrastive losses to avoid mode collapse. Using these techniques, we are able to considerably expand the generative ability of existing feed-forward text-to-shape models such as CLIP-Forge. We support our method via extensive qualitative and quantitative evaluations[1].

1 Introduction

1.1 Motivation

High-quality 3D shape generation is an important primitive in numerous applications spanning video games, animation, scientific visualization, and the metaverse. Consequently, AI-based models for 3D shape generation have garnered significant recent attention. A large majority of such approaches pursue feedforward generative models (such as GANs), producing 3D shapes in various representations: pointclouds Achlioptas et al. [1], Li et al. [2], Yang et al. [3], voxels Wu et al. [4], and meshes [5][6]. A major drawback of the above class of approaches is that they require a labeled dataset of 3D shapes (such as ShapeNet [7]), which typically only consist of a few categories. But for real-world applications, we seek models that can generate a wide variety—ideally, an open set—of shapes, succeeding with scarce (few-shot) or even no (zero-shot) labeled training data.

A solution to the data scarcity problem is by leveraging *vision-language* pretraining. Models such as CLIP [8] are pretrained on web-scale data to jointly learn correlated embeddings for both the visual features of an object and its textual description. Such models demonstrate spectacular zero-shot generalization capabilities, and have recently led to breakthroughs in image classification [9–11], retrieval [12, 13], and image generation [14–16].

Starting from Tang et al. [17], several recent efforts have adapted vision-language models (VLMs) for text-to-shape generation. Two important, zero-shot approaches along this direction are CLIP-Forge [18] and Dream Fields [19]. CLIP-Forge is a feed-forward network trained on the 3D ShapeNet dataset to learn a text-to-shape mapping. It uses a normalizing flow model conditioned on CLIP text encodings coupled with a geometry decoder to produce diverse shapes represented by voxel grids. However, *CLIP-Forge generalizes poorly* outside the categories in ShapeNet. This deficiency is addressed by Dream Fields, which trains a Neural Radiance Field (NeRF) for each object using only the supervision provided by an open-vocabulary natural language caption, plus a sparsity-based loss to encourage coherent geometries. However, NeRFs only implicitly represent shapes and are not generative feed-forward models; synthesizing even a single new concept with *NeRFs require expensive inference-time optimization*, i.e., learning the weights of a network from scratch.

[1]Code available at: https://github.com/Km3888/ZeroForge

Figure 1: *Example 3D shapes (above) with corresponding text inputs (below) from our proposed feedforward 3D generative text-to-shape model, ZeroForge. We are able to generate 3D voxel grids from arbitrary text queries without requiring 3D labeled shape data or NeRF-style optimization.*

1.2 Our Contributions

We present **ZeroForge**, a significant extension of the CLIP-Forge architecture for 3D shape generation that resolves the above two pitfalls, thereby achieves the best-of-all-worlds:

- ZeroForge demonstrates considerable shape generalization abilities beyond ShapeNet.
- ZeroForge does not require any 3D shape supervision, and instead trained using (frozen) CLIP embeddings of open-vocabulary, natural language captions.
- ZeroForge incurs considerably lesser inference-time costs as compared with Dream Fields.

For the above reasons, we envision that ZeroForge can be used in several applications: quickly generating new families of paired shape-image datasets (beyond ShapeNet); 3D visualization of novel concepts described using natural language; probing geometric properties of specific 3D shapes using their voxel representations; and other related tasks in 3D vision. We present a host of 3D visualization results in support of ZeroForge. We also report the competitive performance of ZeroForge in terms of quantitative human evaluation metrics. See Section 4 for details.

1.3 Techniques

At a high level, the core ideas behind ZeroForge are conceptually very simple. Start with a pre-trained CLIP-Forge architecture (with a frozen CLIP text encoder); feed a novel text prompt through CLIP-Forge, render the output shape into 2D, and compute the embeddings of the resulting images (with a frozen CLIP image encoder); compare the similarity between the text and image embeddings; and use this as training signal to update CLIP-Forge decoder and flow model weights.

However, care must be taken to make things work. The cosine similarity used by itself (without additional regularization) leads to mode collapse. Therefore, we train using batches of prompts, and augment the training objective with a contrastive loss term that encourages generator outputs to have higher similarity with their corresponding prompts than with all other text inputs. CLIP-Forge produces binarized voxel grids, which leads to non-differentiability. Instead, we use approximate binarization, coupled with a neural voxel renderer (NVR) to enable backpropagation of gradients from the similarity objective to the decoder weights. Finally, training on only new prompts (outside ShapeNet) leads to forgetting phenomena. We find that training convergence—as well as retention of *original* ShapeNet generation capability—is improved by adding a zero-convolution adapter to the decoder, similar to the very recent work of Zhang and Agrawala [20]. See Section 3 for details.

2 Background

Multi-modal Learning: Multi-modal Learning (MML) has been an important research area in recent decades. The objective of this approach is to enhance learning and predictive performance by integrating information from multiple sources. Ngiam et al. [21] focused on combining audio and visual data and demonstrated that when more than one modality is present during learning, deep networks can learn to leverage the correlative information, leading to improved performance in situations where one or more modalities may be missing. Baltrusaitis et al. [22] focused on human communication and developed a method for understanding it as a combination of verbal (speech) and non-verbal (gesture, facial expression, etc.) signals. Their work underscored the importance of multi-modal learning in real-world applications such as human-computer interaction and affective computing.

Recently, vision-language model (VLM) learning has attracted a lot of attention from various research communities. Depending on the downstream task, different model architectures have been proposed, including the dual-encoder architecture [8, 23], the unified transformer architecture [24, 25], the fusion-encoder architecture [26, 27], and the encoder-decoder architecture [28, 29] Additionally, various pre-training objectives have also been studied, which have progressively converged to a few time-tested ones: image-text contrastive learning [8, 30], image-text matching [27], and (masked) language modeling [31]. VLMs have found considerable success in various tasks such as (zero-shot) classification [9–11], retrieval [12, 13], and image generation [14–16].

CLIP-Forge: A core component of our method is the CLIP-Forge model, which uses the ShapeNet dataset to learn a text-to-shape mapping [7, 18]. The CLIP-Forge pipeline is fully feed-forward and, once trained, requires no additional optimization to generate new shapes belonging to the classes in the dataset. This dataset can be represented as $\mathcal{S} = \{(\mathbf{V}_n, \mathbf{I}_n, \mathbf{O}_n)\}_{n=1}^{N}$ where V_n is a voxel grid of fixed resolution, I_n is a rendered image of the shape, and O_n is a set of space occupancies. We note that in addition to voxels, CLIP-Forge can also generate a point cloud, which we do not consider in our paper and leave as future work.

Due to the relatively small amount of 3D shape data, CLIP-Forge uses a multi-stage training process similar to [32–34]. The first step is learning a latent distribution over all the shapes in the training set. This uses a voxel encoder E, which produces latent code $z_i = E(x_i) \in \mathbb{R}^h$. These latent codes are then passed to an Occupancy-Net [35] like decoder D, which predicts $\mathbf{O}_n$, producing a reconstruction loss. In the second stage, a latent-flow model $\mathcal{F}$ [36] learns the conditional distribution of latent codes z_n based on the CLIP embedding of the rendered image I_n .

During inference, the image embedding is replaced by the CLIP embedding of a text query to give a sample from the distribution over latents conditioned on the text, which can be decoded to predict occupancy values at any points specified. To get a voxel-grid-shaped output, one can simply choose the grid vertices as the occupancy points. These occupancy predictions are then converted into binary voxel values using a constant threshold parameter γ (typically 0.05).

NeRFs: An emerging class of approaches implicitly represents 3D shapes in the form of neural radiance fields (NeRFs) [37]. Original work compared rendered images to ground truth images, but recent work uses vision-language models to do text-to-shape modeling [19, 38, 39]. These can generate visually compelling outputs but require a new network for each shape. This necessitates training a network from scratch for each new text prompt. Such an "inference-time" computational cost makes these approaches unsuitable for generating arbitrary shapes on demand.

Differentiable Rendering: Using CLIP supervision to learn 3D shape generation requires that we backpropagate from RGB images into their corresponding shapes. To do this, we must define a rendering layer $\omega : \mathbb{R}^{N^3} \times [0, 360]^2 \rightarrow \mathbb{R}^{3 \times N^2}$ which maps a 3-dimensional voxel grid (along with a perspective angle) to its 2D rendering while also having a well-defined gradient.

Henzler et al. [40] leverage image supervision to learn a 3D Generative Adversarial Network (GAN) by using a differentiable formulation of classical ray-tracing [41] algorithms. However, these renderings struggle to capture object depth and lighting effects. Later works [42] have achieved better results by training a neural network model which is differentiable by design on large amounts of shape-rendering pairs to predict the images from 3D input.

ControlNet Finetuning: With the widespread success of large foundation models, a recent area of research has focused on enabling the efficient adaptation of large models for specialized purposes [43, 44]. A recent such method is ControlNet [20], which addresses the specific problem of introducing conditional control to pre-trained diffusion models [45]. ControlNet learns a modified version of an original diffusion model by augmenting certain layers. Consider an arbitrary layer written as:

$$y = \mathcal{M}(x, \theta) \tag{1}$$

with input x and parameters θ. To get an easily adaptable version of this layer, ControlNet makes a locked copy of the parameters θ_L, which retains the capabilities of the original trained model as well as a trainable copy θ_C. To allow conditioning on an additional input, this finetuned model accepts an additional parameter c. To combine these, they introduce a pair of 1×1 convolution kernels (that they call zero-convolution, or ZeroConv) with parameters $\theta_z 1, \theta_z 2$ initialized to zero and define a new controllable version of the output:

$$y_c = \mathcal{M}(x, \theta_L) + \mathcal{Z}(\mathcal{M}(x + \mathcal{Z}(c; \theta_{z1}, \theta_C), \theta_{z2}) \tag{2}$$

Because the convolution layers are initialized with zero entries, we have that $y_c = y$ for all inputs at the beginning of training. However, introducing additional trainable parameters allows the network to rapidly adapt to the new finetuning task. This is achieved while maintaining performance on the original pre-training task, as at each layer, the convolution operator can filter out the output from the finetuned parameters.

3 Methods

Our method ZeroForge (see Figure 2) can be viewed as an adaptation of CLIP-Forge to generate an arbitrary distribution over text queries $p(t)$. We begin by using pre-trained CLIP-Forge architectures that have been trained on the ShapeNet dataset; these are well-suited text-to-shape generation for the classes within this dataset. Using this initialization gives a sensible starting point from which the networks can learn novel 3D concepts of interest to us.

To enable generalization to concepts outside ShapeNet, we first leverage a similarity loss based on maximizing an inner product between image and (open vocabulary) text embeddings provided by CLIP [8]. We also define an additional contrastive loss term to discourage mode collapse. To minimize these loss terms, we formulate a differentiable layer for getting binary voxel occupancies from the model's real-valued outputs. Finally, we describe how we can apply the fine-tuning methods used in ControlNet [20] to rapidly learn new shapes while preserving performance on shapes from the ShapeNet dataset.

3.1 Similarity Loss

For performing our adaptation, we can set aside CLIP-Forge's encoder E and define our generator $G = D \circ \mathcal{F}$ as the composition of the latent flow network and voxel decoder (In contrast, recall that the CLIP-Forge model is $E \circ D$). We define a new training objective for this generator which measures how well the renderings of its voxel grid outputs match the text descriptions given as input. To do this, we sample angles randomly from the distribution $q(\theta, \phi)$ over the unit sphere using Shao and Badler [46] and differentiably render the shape from these angles using the renderer ω [42]. These 2D images can then be embedded into CLIP space using CLIP's image encoder head and compared to the encoded text input. This allows us to define the similarity loss $\mathcal{L}_{sim}$ which encourages voxel outputs matching the text queries drawn from $p(t)$:

$$\mathcal{L}_{sim}(G) = -\mathbb{E}_{t\sim p,(\theta,\phi)\sim q}\Big[h(\omega(G(g(t)), \theta, \phi)) \cdot g(t)\Big]. \tag{3}$$

3.2 Contrastive Loss

While this simple similarity loss can achieve 3D shapes which have high similarity in CLIP embedding space, we observe that using it by itself can result in mode collapse. This occurs when the input distribution contains text queries that have high similarity with one another in CLIP space. In this scenario, the generator will converge to the degenerate solution of producing whichever shape is simplest, regardless of the input query. For instance, the queries ‘‘`spoon`’’ and ‘‘`fork`’’ have very high similarity, despite these shapes having very different appearances. As a result, if the generator outputs spoons for the input ‘‘`fork`’’, it will still partially satisfy the similarity loss.

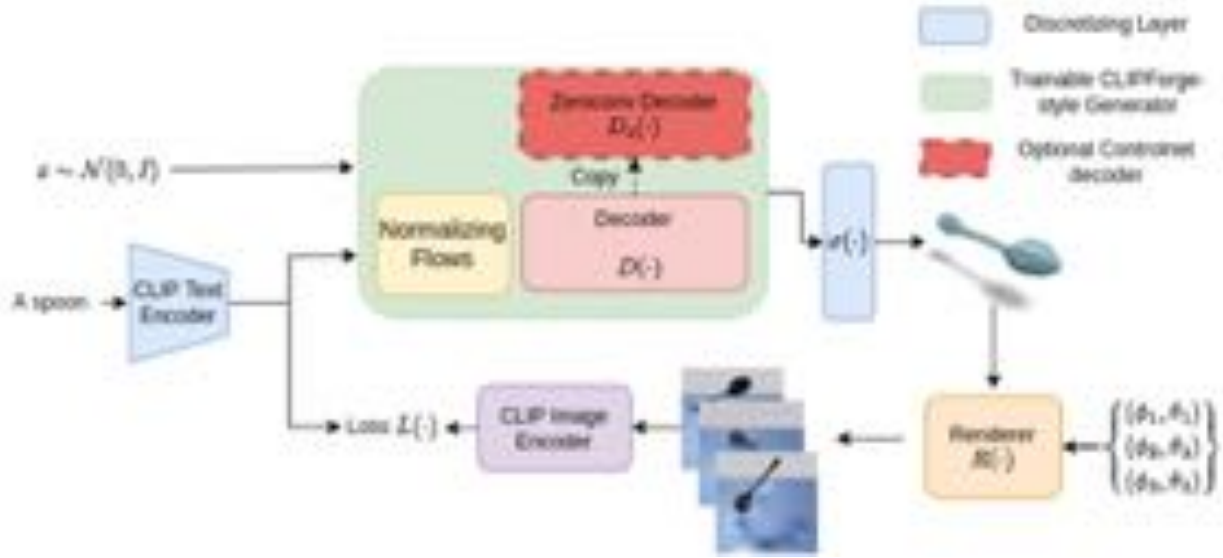

Figure 2: ***ZeroForge Block Diagram.*** *Illustration of our novel method for learning text-to-shape models. Outputs from a pre-trained CLIP-Forge model are rendered into images and encoded into CLIP space. By computing their similarity with the input text queries we obtain a tractable loss function for arbitrary text queries*

To remedy this, we augment $\mathcal{L}_{sim}$ a contrastive loss term based on InfoNCE [47] to our training objective. This encourages each of the generator's outputs to have higher similarity with its corresponding text input than with all the other inputs. We define this loss for a batch of size B with $\mathcal{I}, \mathcal{T} \in \mathbb{R}^{B \times H}$ representing the image and text embeddings, respectively:

$$\mathcal{L}_{contrast}(\mathcal{I}, \mathcal{T}, \tau) = \frac{1}{M} \sum_{i=1}^{B} \log \left(\frac{\exp(\tau \mathcal{I}_i \, \mathcal{I}_i)}{\sum_{j=1}^{B} \exp(\tau \mathcal{I}_j \, \mathcal{T}_i)} \right), /. \tag{4}$$

This loss prevents mode collapse by penalizing similarity between text-image pairs that do not correspond to one another. The temperature hyperparameter τ controls how steeply the penalization is distributed amongst the different non-matching text queries. When τ is high, only the text queries with the highest similarity are punished, and when it is zero, all queries are punished equally regardless of similarity to the rendered image. We add this contrastive loss to our original training objective as a regularizer multiplied by the hyperparameter λ_C. We tune this hyperparameter to avoid mode collapse while allowing the generator to accurately generate shapes that are similar to one another.

3.3 Differentiable Binarization

Unlike CLIP-Forge, we do not fully binarize the output voxel grid based on whether the voxel values are above a fixed threshold. Instead, we use approximate Binarization using a sigmoidal transfer function B, producing a grid representation that is differentiable, following Liao et al. [48]. During training, we use the same threshold value γ that was used for the CLIP-Forge initialization and get a near-binary voxel output.

$$B(x, \beta, \gamma) = \frac{1}{1 + \exp(\beta(x - \gamma))} \tag{5}$$

The temperature parameter, β, controls the tradeoff between a tight approximation and smooth gradients, with B recovering the regular step function in the limit as $\beta \to \infty$. This "soft thresholding" allows us to backpropagate gradients through the binarization operation at train time while ensuring that our generator will produce sensible outputs when we use the regular thresholding operation during inference.

3.4 ZeroConv

We implement a variant of the zero-convolution layer from Zhang and Agrawala [20] to improve training convergence and improve the tradeoff between the performance on the fine-tuning queries and those in the original ShapeNet dataset. Because we are not concerned with adding conditional control to the model, we can simplify Equation 2 to:

$$y_c = F(x, \theta_L) + \mathcal{Z}(F(\theta_z)). \tag{6}$$

We replace each residual block in the pre-trained decoder with an augmented version based on this formulation, with a 3D convolutional layer to connect the frozen CLIP-Forge weights and the fine-tuned weights learned using the loss we define. This additional set of weights can be referred to as D_c, as illustrated in Figure 2.

4 Results

4.1 Generating Novel Shapes

To test our method, we perform qualitative evaluations on several distributions over text queries. First, we gradually expand the support of CLIP-Forge's distribution from the classes in ShapeNet by training ZeroForge on 10 text queries. Five (Table, Airplane, Boat, Chair, Television) are present in the ShapeNet dataset, and the other five (Umbrella, Barbell, Christmas tree, Knife, Spoon) are novel. We set $\lambda_c = 0.01, \tau = 50$ and report results from both with and without the ZeroConv modification of the decoder network. We adopt the same voxel rendering technique as CLIP-Forge for visualization purposes.

The results of this experiment are shown in Figure 3. Both choices of decoder architecture allow ZeroForge to learn the task distribution with minimal mode collapse, capturing the majority of shapes. However, the addition of ZeroConv shows a clear improvement in shapes within the original ShapeNet distribution. Because the original CLIP-Forge parameters are frozen, and the convolution kernels can easily learn to filter out the results of the fine-tuned parameters at each layer, these shapes are largely identical to those that would have been produced by the CLIP-Forge model. For

visual clarity, we also show the volume renderings of various shapes generated by ZeroForge in Figure 4. Volume renderings were generated using GPU-accelerated ray-marching of the voxel grid [49].

A key observation from our experiments is the importance of initializing ZeroForge with CLIP-Forge weights. Without this inductive bias, we find that gradient descent cannot find a sensible solution to the data-free shape generation process. Shapenet-based training of CLIP-Forge gives our network a solid foundation for representing basic primitive shapes, which it can learn to reform into new shapes that satisfy the CLIP-based loss. We demonstrate this with additional figures in the appendix, showing ZeroForge's attempt to learn the same ten shapes shown in Figure 3 using the same architecture as before but with randomly initialized weights.

To demonstrate the problem of mode collapse and how it can be addressed by the contrastive loss term defined in Equation 4, we also provide an example of a set of text queries that are prone to collapse, seen in Equation 6. These four queries {`spoon`, `fork`, `knife`, and `wineglass`} are all tableware and, as a result, have a significant commonality in CLIP's text embedding space. This makes it particularly difficult to avoid collapsing to a single-shape output in this case. As a result, we can observe the role of the λ_C and τ hyperparameters. Even setting $\lambda_c = 0.01$, which we find to be sufficient for other text queries, there is little difference between the outputs for `fork` and `spoon`. Getting the

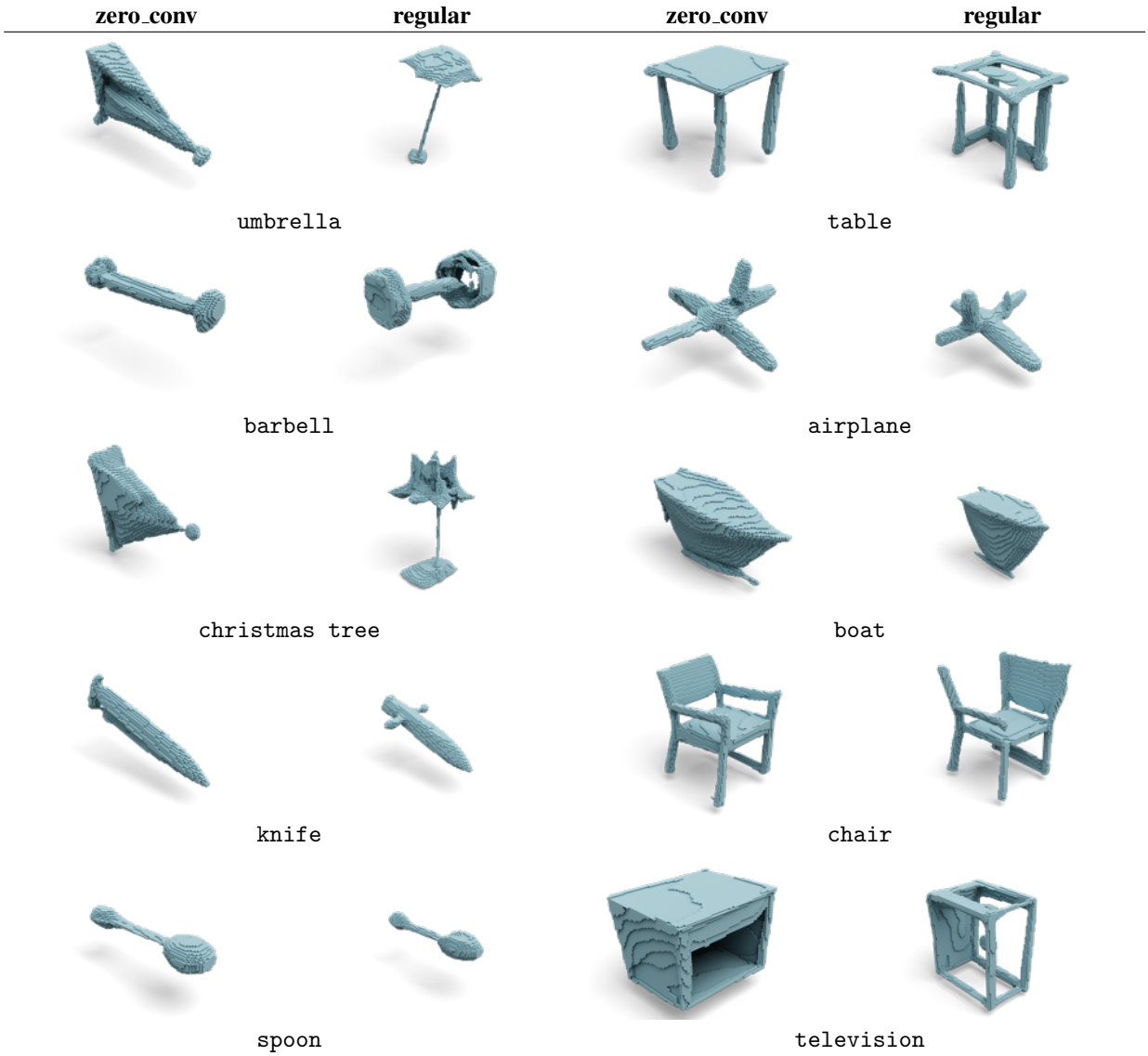

Figure 3: *Shapes generated from ZeroForge after finetuning on 5 queries which were not seen in pre-training data (umbrella, barbell, Christmas tree, knife, spoon) and 5 which were (table, airplane, boat, chair, television). Here we use* $\lambda_C = 0.01$

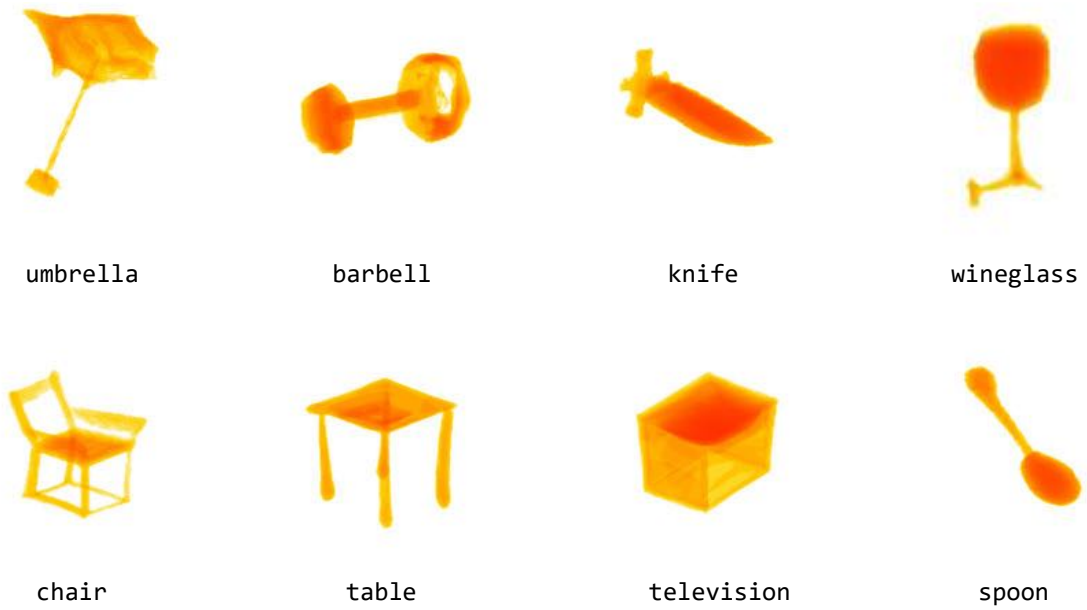

Figure 4: *Volume renderings for various shapes generated by ZeroForge (umbrella, barbell, knife, wineglass, chair, table, television, and spoon).*

model to produce the fork tines requires not only increasing the value of λ_c, but also the τ to properly penalize the errant similarity between the renderings of the fork and spoon outputs.

To test ZeroForge's limits regarding the size of the query distribution, we train ZeroForge on a set of 12 challenging shape descriptions, all of which are outside the original CLIP-Forge dataset. These queries are: { `fork`, `hammer`, `knife`, `mushroom`, `pencil`, `sandal`, `screw`, `screwdriver`, `sombrero`, `spoon`, `umbrella`, `wineglass` }.

Figure 5: *Samples from a ZeroForge model trained on 12 challenging and complex shape classes not seen in the original data distribution.*

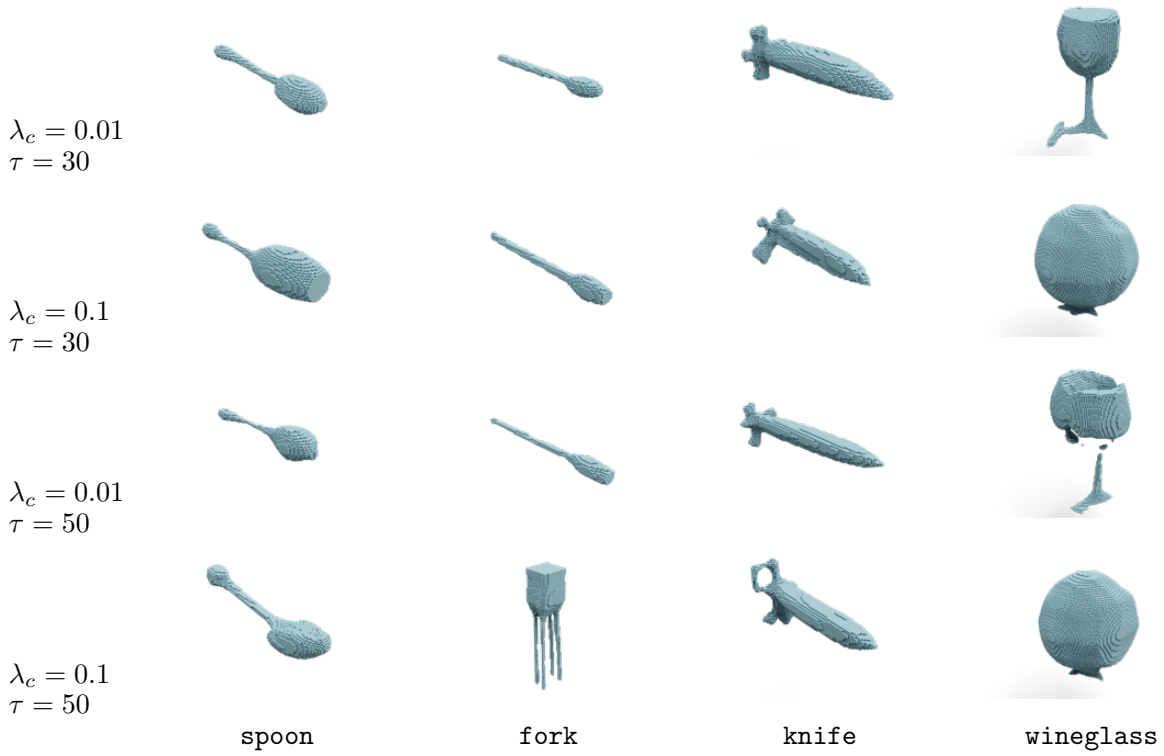

Figure 6: *Ablation study on shapes generated from ZeroForge using different values of λ_c and τ.*

In-distribution	Out-of-Distribution	Overall
95.0±11.9 %	95.7±13.3%	$95.4\% \pm 5.0\%$

Table 1: *Accuracy of ZeroForge in a human evaluation survey.*

To measure how well the ZeroForge-generated shapes appear to human beings, we conducted a human perceptual evaluation via an online survey. We mirrored the protocol outlined in the paper describing CLIP-Forge [18]. Participants were shown a pair of shapes generated by ZeroForge and asked to determine which of the two objects matched the text query used to generate one of them. For this experiment, we used the objects displayed in Figure 3 and the model trained using a ZeroConv decoder. A total of 65 participants were queried, with each one judging 17 pairs of queries.

The average accuracy was 95.4% (Table 1) across all queries, with over half of surveyed participants identifying the correct answer for all 17 questions. The standard deviation of the accuracy rate across participants was 5 percentage points. Subdividing this accuracy between shapes that were in the ShapeNet dataset and those which were not, we get that the former grouping yielded an average of 95.0% correct, and the latter gave an average of 95.7% correct, well below the threshold of statistical significance.

Please see the appendix for several more experimental results and ablation studies.

5 Conclusions

In this paper, we presented ZeroForge, a novel feedforward text-to-shape model capable of generating open-vocabulary shapes (in terms of their voxel grid representation) without 3D labeled supervision. We validated ZeroForge using several qualitative and quantitative evaluations.

Several avenues for future work remain. We have focused on voxel grid representations for generated shapes, but extensions to point clouds and mesh-based representations are a natural next step. A better understanding of the impact of various architectural choices: pre-training weights, prompt context length, and the capacity of flow model and decoder, would pave the way for better zero-shot 3D generation. Finally, a hybrid combination with NeRF-style methods would enable high spatial resolution, as well as endow generated shapes with improved texture properties.

5.1 Limitations

Despite the several encouraging results, we find several key areas in which future work could improve upon our results.

One such area is in improving the tradeoff introduced by the contrast coefficient. While higher values of this hyperparameter are necessary for preventing mode collapse and allowing the generation of certain shapes, it can also result in dissimilarity amongst shapes which ought to be similar. This can be seen in Figure 6, where to avoid generating spoons in place of forks, the contrastive loss degrades the wineglass shape to avoid it resembling a spoon.

Our results also show that the inductive bias provided by using pre-trained CLIP-Forge weights as initialization for ZeroForge can also affect the performance of the final trained ZeroForge model. For instance, even when ZeroForge avoids mode collapse for the `fork` query, its output still somewhat resembles a chair, as this is part of the ShapeNet dataset that CLIP-Forge is trained on. This can be seen in the training process as early on, the outputs for queries not seen in the training data look almost identical to ShapeNet objects and then *continuously deform* into the correct shapes. This raises interesting questions of whether ZeroForge can generate shapes with completely new (and intricate) topological properties, and we defer a more thorough characterization of this question to future work.

In several of the generated shapes, there are visible holes and areas of the shape not filled in. This is likely attributable to the differentiable binarization layer (Equation 5), which assigns non-zero occupancy values to the generated voxels just below the thresholding value γ. If there are several such voxels in the same area, they can form a low opaqueness region which is rendered as part of the shape. This can be addressed by increasing the binarization temperature β, but doing so also adversely affects our optimization procedure as it causes the loss gradients to become less smooth. Another potential method for improving this would be to use a smoothing method such as bilateral filtering [50] instead of a fixed threshold value.

Finally, like CLIP-Forge, we do not modify the CLIP text encoder and therefore are limited by CLIP's context length (currently, 77 tokens). But we envision that text encoders from future vision-language models with longer context lengths can be integrated within ZeroForge to resolve this limitation.

5.2 Broader Impacts

By building and validating ZeroForge, we have demonstrated rapid (feedforward) generative capability of complex 3D shapes using very weak (natural language) supervision. This opens the door to several new computational tools: interactive 3D visualization; understanding geometric properties by probing their voxel representations; and novel dataset generation and curation. These tools open the door for several fruitful applications, including improved design and manufacturing of novel industrial tools, as well as visualizing new material geometries.

However, more sophisticated versions of ZeroForge could also be used to realistically hallucinate 3D objects, and, combined with augmented reality tools, could be used as a misinformation tool when used inappropriately. On balance, we feel that techniques such as ZeroForge pave the way for a better overall understanding of the scope of modern 3D shape generation tools, which is a necessary step for their safe deployment.

Acknowledgements

This work was supported in part by NSF CAREER grants CCF-2005804 and OAC-1750865, NSF grant LEAP-HI-2053760, and NSF/USDA-NIFA grant 2021-67021-35329. KOM was also supported by a US Department of Education GAANN Fellowship.

A Appendix

A.1 Implementation Details

Architecture:

Our model architecture built off of the default implementation provided by CLIP-Forge[18]. The RealNVP [36] latent flow model consisted of 5 coupling layer blocks, each with translation and scaling followed by batch norm and using the inverse of the masking scheme from the previous block. The blocks themselves each contained 2-layer MLPs with a hidden size of 1024 followed by an additional linear layer. The decoder model uses 5 3-dimensional ResNet decoder blocks. The shape embedding outputted by the flow model and given to the decoder has 128 dimensions.

For training ZeroForge, we use an Adam optimizer with a learning rate of 10^{-5} and $\beta_1 = 0.9, \beta_2 = 0.999$. We set the batch size to be three times the number of queries to try to obtain accurate gradients. All ZeroForge visualizations come from models trained for 15k training iterations unless otherwise noted.

Hparams: Unless otherwise specified we use the following hyperparameters in each ZeroForge experiment: $\beta = 200, \tau = 50, \gamma = 0.05$. We generate our voxels with a resolution of 128 in each dimension and resize our rendered images to 224×224 pixels in each dimension to make them inputtable to CLIP. All models for experiments were trained on 4 NVIDIA A100 GPUs.

A.2 Additional Experiments

The first set of additional experiments we present is a test of the importance of using CLIP-Forge weights to initialize our ZeroForge method. We mimic the training procedure and architecture used in training ZeroForge to produce the images in Figure 3, but instead of initializing the latent flow and decoder models using CLIP-Forge weights we instead initialize them randomly. We show in Figure 7 that without the inductive bias provided by CLIP-Forge's initialization, the networks are unable to generate sensible shapes using only the similarity-based CLIP loss. Instead, the model converges to a degenerate solution in which it outputs blockish shapes that lack the recognizable features of typical ZeroForge outputs.

As an additional ablation, we examine the effect of the β hyperparameter on ZeroForge's performance, showing the outputs of models trained using $\beta = 100$ Figure 8 and $\beta = 300$ in Figure 9 (our default setting for this hyperparameter was 200). We see that ZeroForge is relatively robust to this choice of hyperparameter, with distinguishable outputs still being generated at lower and higher values. We do however observe some degradation in quality at $\beta = 100$ compared to the default setting and in training the model with $\beta = 300$ there is a slower rate of convergence.

Finally, we demonstrate the progression of ZeroForge's outputs during training by including shapes generated by a partially trained model in Figure 10. We select the model used to generate Figure 3 after 0 iterations and 5000 iterations of ZeroForge training (note that the first is just the weights of the CLIP-Forge initialization). In this case we select the variant without ZeroConv so that that the progressive degradation on ShapeNet shapes can also be visualized. It can be seen here how ZeroForge learns to represent novel shapes by deforming and rearranging shapes that were in CLIP-Forge's training distribution. For instance, "barbell" and "Christmas tree" both start off resembling lamps as that is a shape present in ShapeNet.

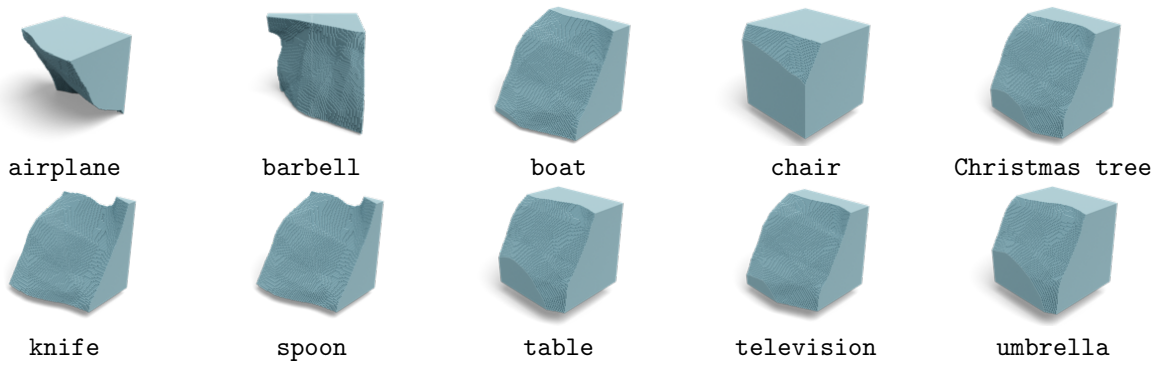

Figure 7: *Samples from a ZeroForge model trained without using CLIP-Forge weights as an initialization. Notice that these outputs lack definition as compared with typical ZeroForge outputs*

A.3 Visualizations

For visual clarity purposes, we generate volume renderings for each of the shapes shown in Figure 1, Figure 3, Figure 5, and Figure 6.

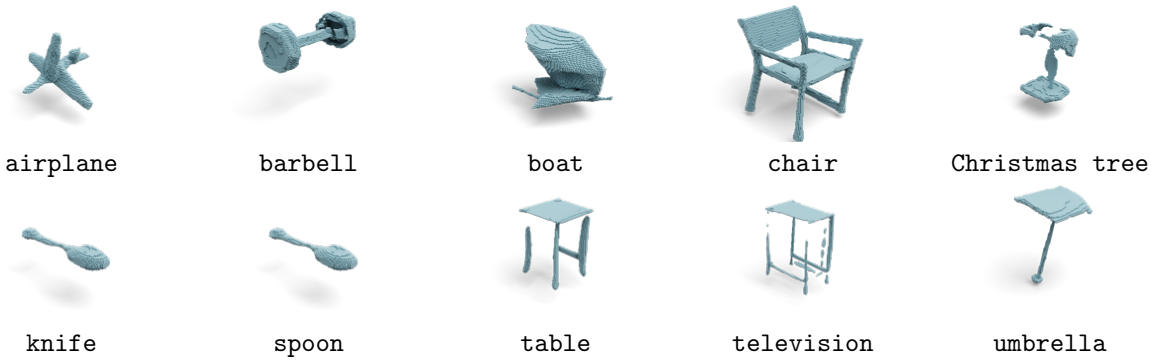

Figure 8: *Samples from a ZeroForge model trained with* $\beta = 100$

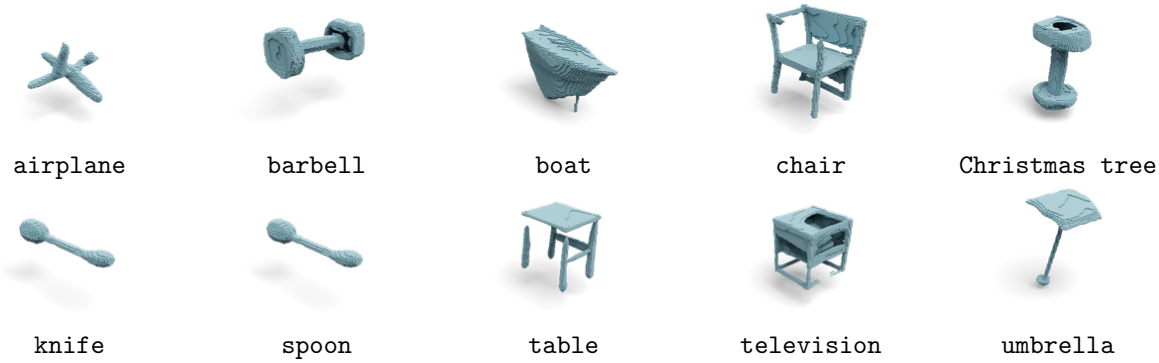

Figure 9: *Samples from a ZeroForge model trained with* $\beta = 300$

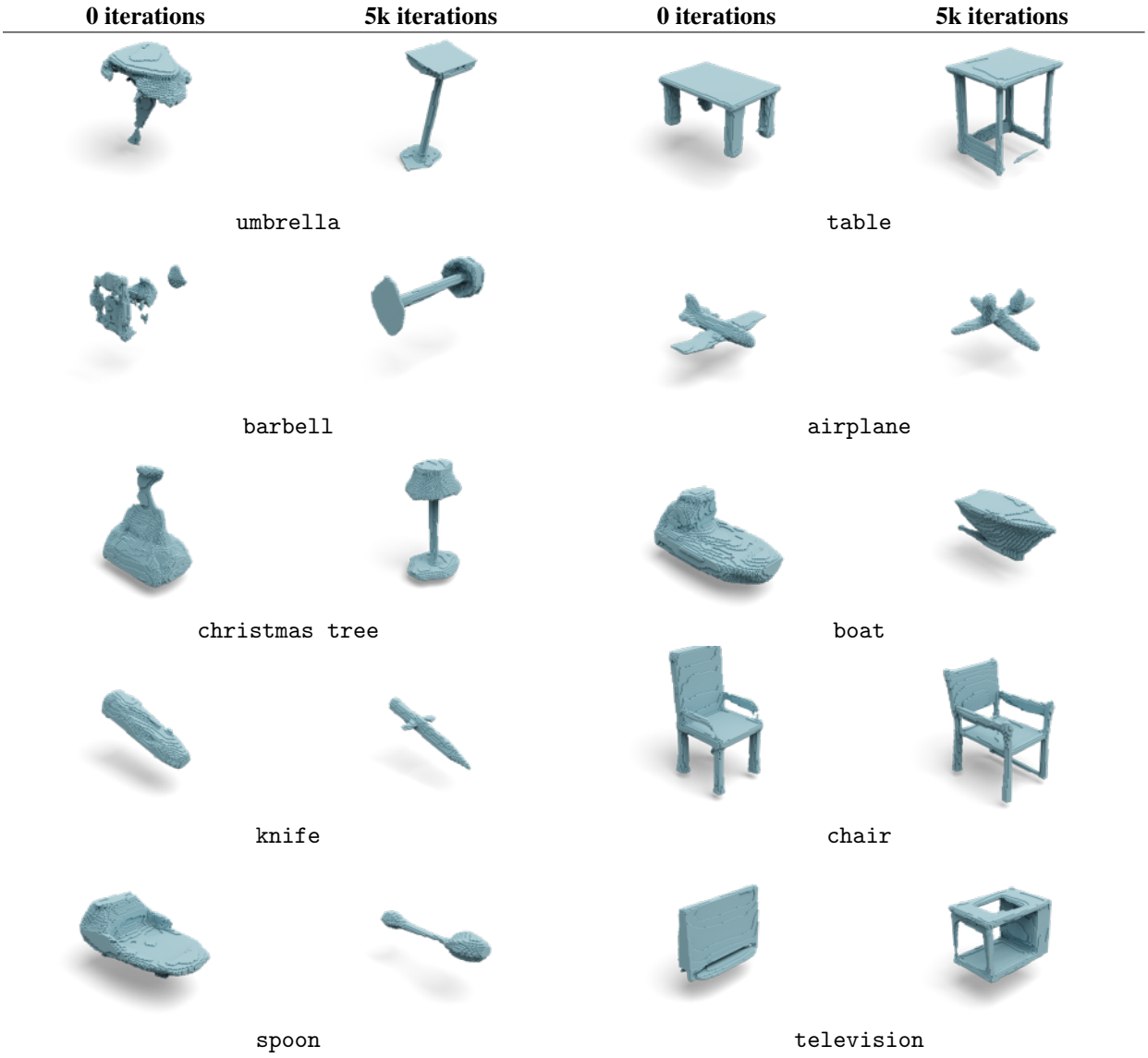

Figure 10: *Shapes generated by untrained and partially trained ZeroForge models. Note how several shapes outside CLIP-Forge's are initially represented by shapes inside the distribution which resemble them.*

Figure 11: *Volume renderings for shapes in Figure 1.*

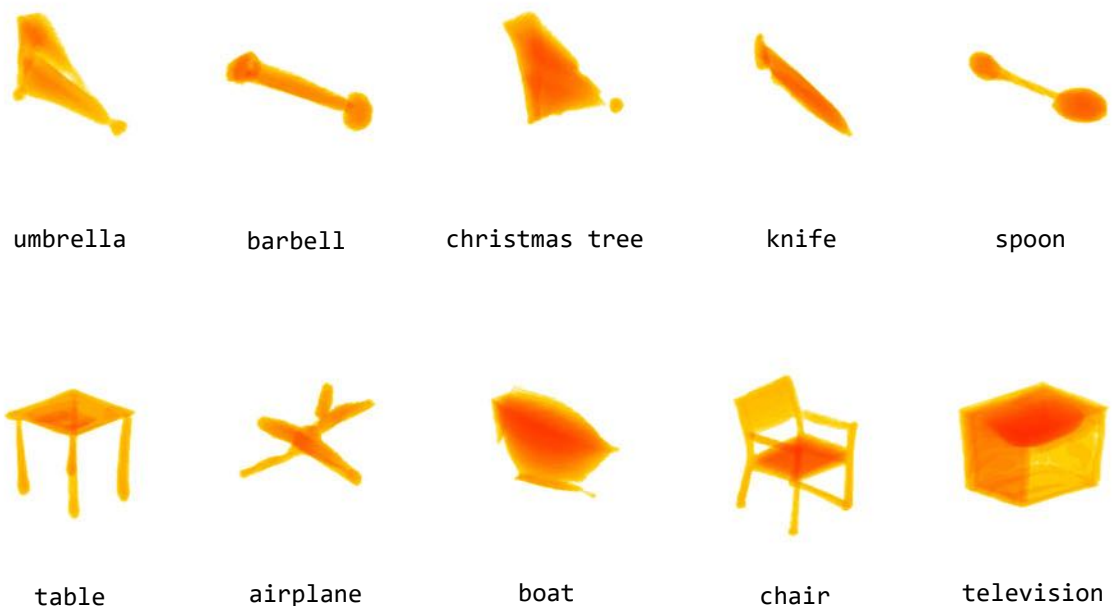

Figure 12: *Volume renderings for **zero_conv** config from Figure 3.*

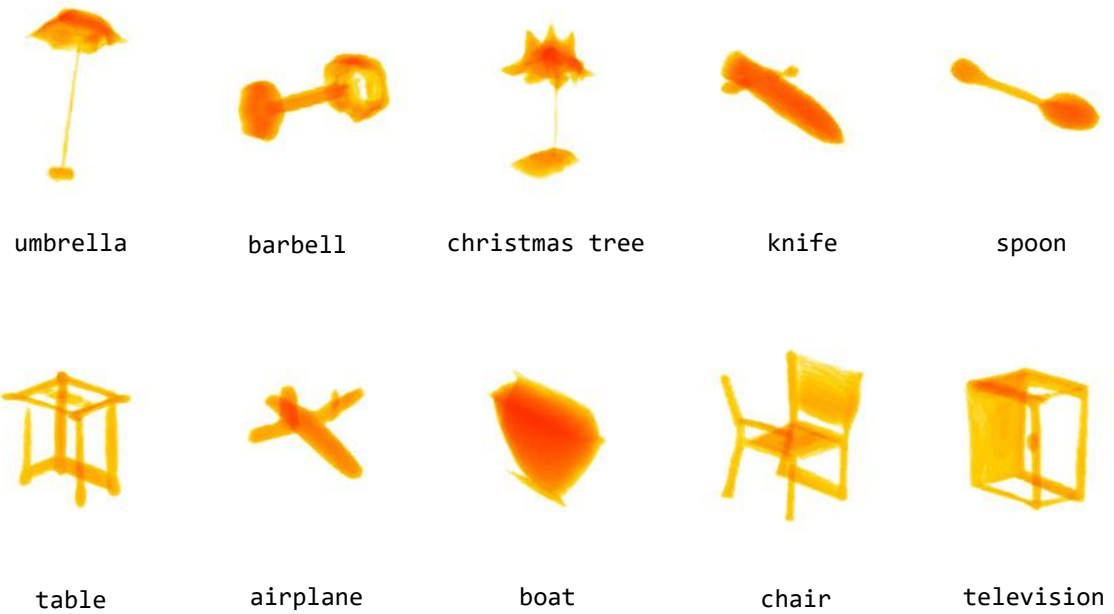

Figure 13: *Volume renderings for **regular** config from Figure 3.*

Figure 14: *Volume renderings for shapes shown in Figure 5.*

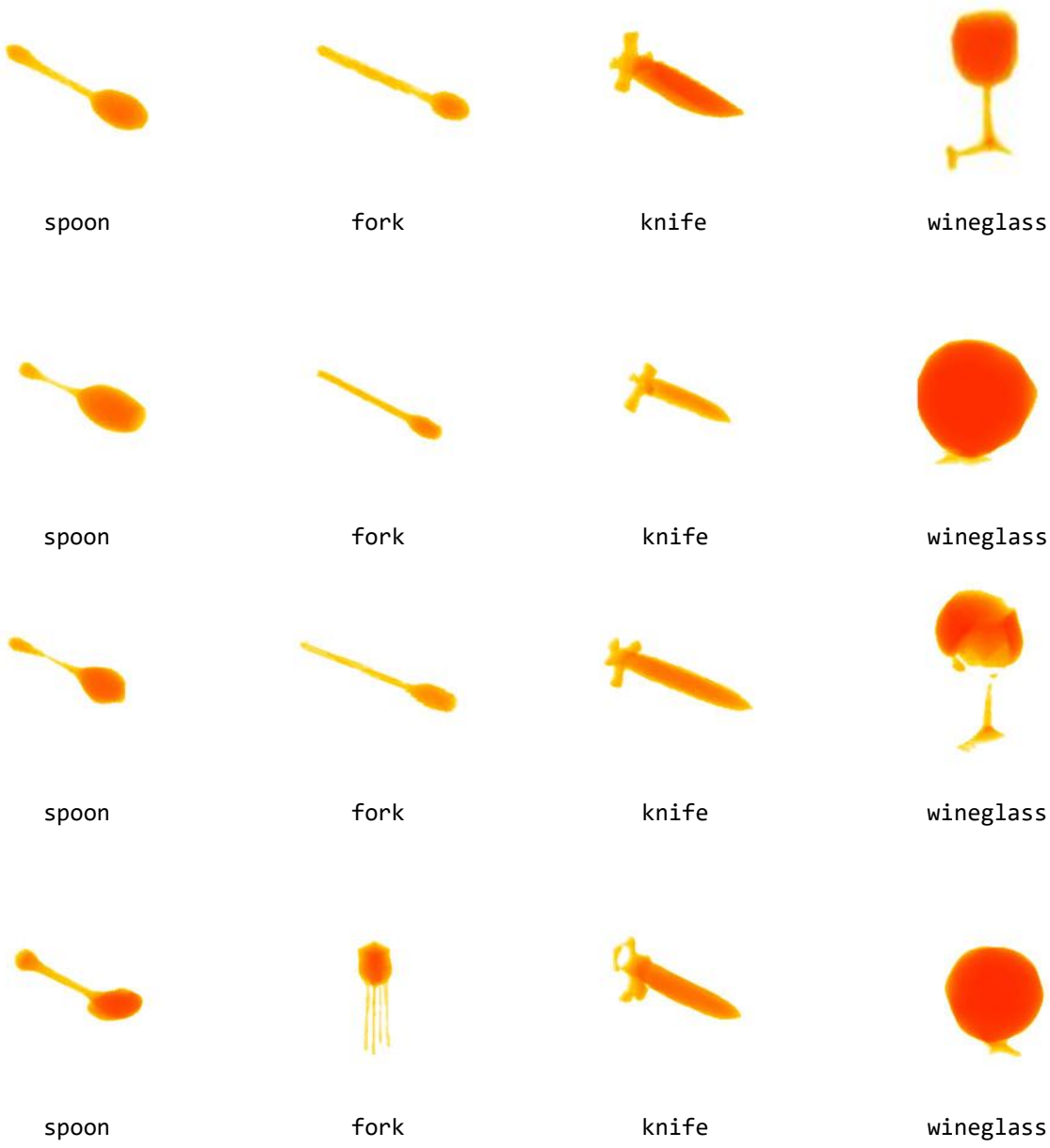

Figure 15: *Volume renderings for shapes shown in Figure 6. The top row corresponds to* $\lambda_c = 0.01$ *and* $\tau = 30$*, the second row corresponds to* $\lambda_c = 0.1$ *and* $\tau = 30$*, the third row corresponds to* $\lambda_c = 0.01$ *and* $\tau = 50$*, and the fourth row corresponds to* $\lambda_c = 0.1$ *and* $\tau = 50$*.*

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3d point clouds. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=r14RP5AUz.

[2] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabás Póczos, and Ruslan Salakhutdinov. Point cloud GAN. In *Deep Generative Models for Highly Structured Data, ICLR 2019 Workshop, New Orleans, Louisiana, United States, May 6, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=S1xim8UFuV.

[3] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge J. Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 4540–4549. IEEE, 2019. doi: 10.1109/ICCV.2019.00464. URL https://doi.org/10.1109/ICCV.2019.00464.

[4] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 82–90, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/44f683a84163b3523afe57c2e008bc8c-Abstract.html.

[5] Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 7220–7229. PMLR, 2020. URL http://proceedings.mlr.press/v119/nash20a.html.

[6] Maria Vakalopoulou, Guillaume Chassagnon, Norbert Bus, Rafael Marini, Evangelia I. Zacharaki, Marie-Pierre Revel, and Nikos Paragios. Atlasnet: Multi-atlas non-linear deep networks for medical image segmentation. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger, editors, *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part IV*, volume 11073 of *Lecture Notes in Computer Science*, pages 658–666. Springer, 2018. doi: 10.1007/978-3-030-00937-3_75. URL https://doi.org/10.1007/978-3-030-00937-3_75.

[7] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing 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. *CoRR*, abs/1512.03012, 2015. URL http://arxiv.org/abs/1512.03012.

[8] 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. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021. URL http://proceedings.mlr.press/v139/radford21a.html.

[9] Karan Desai and Justin Johnson. Virtex: Learning visual representations from textual annotations. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 11162–11173. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01101. URL https://openaccess.thecvf.com/content/CVPR2021/html/Desai_VirTex_Learning_Visual_Representations_From_Textual_Annotations_CVPR_2021_paper.html.

[10] Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomás Mikolov. Devise: A deep visual-semantic embedding model. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2121–2129, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/7cce53cf90577442771720a370c3c723-Abstract.html.

[11] Armand Joulin, Laurens van der Maaten, Allan Jabri, and Nicolas Vasilache. Learning visual features from large weakly supervised data. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 67–84, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46478-7.

[12] Han Fang, Pengfei Xiong, Luhui Xu, and Yu Chen. Clip2video: Mastering video-text retrieval via image CLIP. *CoRR*, abs/2106.11097, 2021. URL https://arxiv.org/abs/2106.11097.

[13] Andrej Karpathy, Armand Joulin, and Li Fei-Fei. Deep fragment embeddings for bidirectional image sentence mapping. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 1889–1897, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/84d2004bf28a2095230e8e14993d398d-Abstract.html.

[14] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 2021. URL http://proceedings.mlr.press/v139/ramesh21a.html.

[15] Kevin Frans, Lisa B. Soros, and Olaf Witkowski. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/21f76686538a5f06dc431efea5f475f5-Abstract-Conference.html.

[16] Kevin Chen, Christopher B. Choy, Manolis Savva, Angel X. Chang, Thomas A. Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. In C. V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler, editors, *Computer Vision - ACCV 2018 - 14th Asian Conference on Computer Vision, Perth, Australia, December 2-6, 2018, Revised Selected Papers, Part III*, volume 11363 of *Lecture Notes in Computer Science*, pages 100–116. Springer, 2018. doi: 10.1007/978-3-030-20893-6_7. URL https://doi.org/10.1007/978-3-030-20893-6_7.

[17] Chuan Tang, Xi Yang, Bojian Wu, Zhizhong Han, and Yi Chang. Parts2words: Learning joint embedding of point clouds and texts by bidirectional matching between parts and words, 2023.

[18] Aditya Sanghi, Hang Chu, Joseph G. Lambourne, Ye Wang, Chin-Yi Cheng, Marco Fumero, and Kamal Rahimi Malekshan. Clip-forge: Towards zero-shot text-to-shape generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 18582–18592. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01805. URL https://doi.org/10.1109/CVPR52688.2022.01805.

[19] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. *CVPR*, 2022.

[20] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *CoRR*, abs/2302.05543, 2023. doi: 10.48550/arXiv.2302.05543. URL https://doi.org/10.48550/arXiv.2302.05543.

[21] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. Multimodal deep learning. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 689–696. Omnipress, 2011. URL https://icml.cc/2011/papers/399_icmlpaper.pdf.

[22] Tadas Baltrusaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(2):423–443, 2019. doi: 10.1109/TPAMI.2018.2798607. URL https://doi.org/10.1109/TPAMI.2018.2798607.

[23] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 4904–4916. PMLR, 2021. URL http://proceedings.mlr.press/v139/jia21b.html.

[24] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi. BLIP: bootstrapping language-image pre-training for unified vision-language understanding and generation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 12888–12900. PMLR, 2022. URL https://proceedings.mlr.press/v162/li22n.html.

[25] Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, and Furu Wei. Image as a foreign language: Beit pretraining for all vision and vision-language tasks. *CoRR*, abs/2208.10442, 2022. doi: 10.48550/arXiv.2208.10442. URL https://doi.org/10.48550/arXiv.2208.10442.

[26] Hao Tan and Mohit Bansal. LXMERT: learning cross-modality encoder representations from transformers. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language*

Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 5099–5110. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1514. URL https://doi.org/10.18653/v1/D19-1514.

[27] Junnan Li, Ramprasaath R. Selvaraju, Akhilesh Gotmare, Shafiq R. Joty, Caiming Xiong, and Steven Chu-Hong Hoi. Align before fuse: Vision and language representation learning with momentum distillation. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 9694–9705, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/505259756244493872b7709a8a01b536-Abstract.html.

[28] Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. Unifying vision-and-language tasks via text generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1931–1942. PMLR, 2021. URL http://proceedings.mlr.press/v139/cho21a.html.

[29] Zirui Wang, Jiahui Yu, Adams Wei Yu, Zihang Dai, Yulia Tsvetkov, and Yuan Cao. Simvlm: Simple visual language model pretraining with weak supervision. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=GUrhfTuf_3.

[30] Lewei Yao, Runhui Huang, Lu Hou, Guansong Lu, Minzhe Niu, Hang Xu, Xiaodan Liang, Zhenguo Li, Xin Jiang, and Chunjing Xu. FILIP: fine-grained interactive language-image pre-training. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=cpDhcsEDC2.

[31] Mert Bülent Sariyildiz, Julien Perez, and Diane Larlus. Learning visual representations with caption annotations. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VIII*, volume 12353 of *Lecture Notes in Computer Science*, pages 153–170. Springer, 2020. doi: 10.1007/978-3-030-58598-3_10. URL https://doi.org/10.1007/978-3-030-58598-3_10.

[32] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6306–6315, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/7a98af17e63a0ac09ce2e96d03992fbc-Abstract.html.

[33] Jinlin Liu, Yuan Yao, and Jianqiang Ren. An acceleration framework for high resolution image synthesis. *CoRR*, abs/1909.03611, 2019. URL http://arxiv.org/abs/1909.03611.

[34] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 12873–12883. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01268. URL https://openaccess.thecvf.com/content/CVPR2021/html/Esser_Taming_Transformers_for_High-Resolution_Image_Synthesis_CVPR_2021_paper.html.

[35] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4460–4470. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00459. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Mescheder_Occupancy_Networks_Learning_3D_Reconstruction_in_Function_Space_CVPR_2019_paper.html.

[36] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

[37] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65(1):99–106, 2022. doi: 10.1145/3503250. URL https://doi.org/10.1145/3503250.

[38] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion, 2022. URL https://arxiv.org/abs/2209.14988.

[39] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition,*

CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pages 3825–3834. IEEE, 2022. doi: 10.1109/CVPR52688.2022.00381. URL https://doi.org/10.1109/CVPR52688.2022.00381.

[40] Philipp Henzler, Niloy J. Mitra, and Tobias Ritschel. Escaping plato's cave: 3d shape from adversarial rendering. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[41] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, page 65–74, New York, NY, USA, 1988. Association for Computing Machinery. ISBN 0897912756. doi: 10.1145/54852.378484. URL https://doi.org/10.1145/54852.378484.

[42] Konstantinos Rematas and Vittorio Ferrari. Neural voxel renderer: Learning an accurate and controllable rendering tool. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 5416–5426. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00546. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Rematas_Neural_Voxel_Renderer_Learning_an_Accurate_and_Controllable_Rendering_Tool_CVPR_2020_paper.html.

[43] Zhaojiang Lin, Andrea Madotto, and Pascale Fung. Exploring versatile generative language model via parameter-efficient transfer learning. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 441–459. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.41. URL https://doi.org/10.18653/v1/2020.findings-emnlp.41.

[44] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[45] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

[46] Min-Zhi Shao and Norman Badler. Spherical sampling by archimedes' theorem. *Technical Reports (CIS)*, 01 1996.

[47] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

[48] Minghui Liao, Zhaoyi Wan, Cong Yao, Kai Chen, and Xiang Bai. Real-time scene text detection with differentiable binarization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11474–11481, Apr. 2020. doi: 10.1609/aaai.v34i07.6812. URL https://ojs.aaai.org/index.php/AAAI/article/view/6812.

[49] Randima Fernando et al. *GPU gems: programming techniques, tips, and tricks for real-time graphics*, volume 590. Addison-Wesley Reading, 2004.

[50] Giuseppe Papari, Nasiru Idowu, and Trond Varslot. Fast bilateral filtering for denoising large 3d images. *IEEE Transactions on Image Processing*, 26(1):251–261, 2017. doi: 10.1109/TIP.2016.2624148.