# AutoSDF: Shape Priors for 3D Completion, Reconstruction, and Generation

Paritosh Mittal*,[1] Yen-Chi Cheng*,[1] Maneesh Singh[2] Shubham Tulsiani[1]
[1]Carnegie Mellon University [2]Verisk Analytics

https://yccyenchicheng.github.io/AutoSDF/

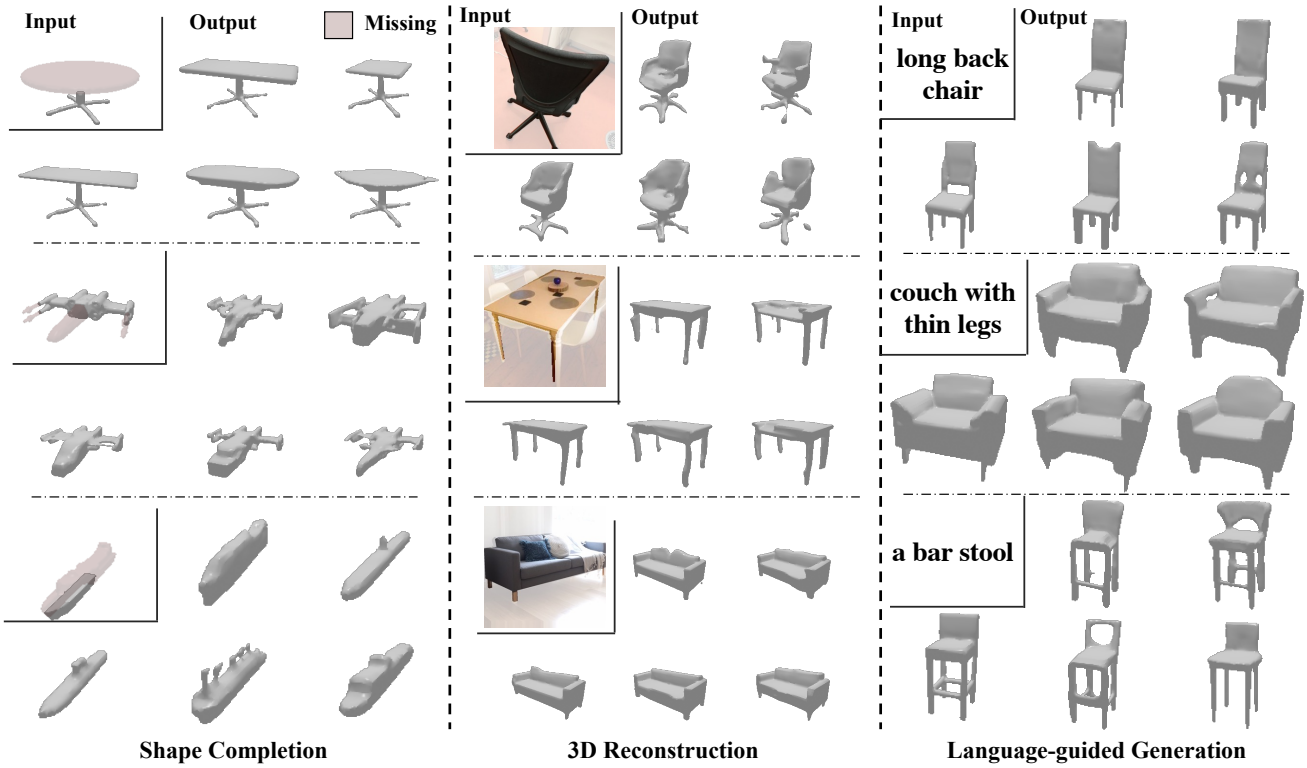



Figure 1. Our approach combines a novel, non-sequential autoregressive prior, capturing the distribution over 3D shapes, with task-specific conditionals, to generate multiple plausible and high-quality shapes consistent with input conditioning. We show the efficacy of our approach across diverse tasks such as shape completion, single-view reconstruction and language-guided generation.

## Abstract

*Powerful priors allow us to perform inference with insufficient information. In this paper, we propose an autoregressive prior for 3D shapes to solve multimodal 3D tasks such as shape completion, reconstruction, and generation. We model the distribution over 3D shapes as a non-sequential autoregressive distribution over a discretized, low-dimensional, symbolic grid-like latent representation of 3D shapes. This enables us to represent distributions over 3D shapes conditioned on information from an arbitrary set of spatially anchored query locations and thus perform shape completion in such arbitrary settings* (e.g. *generating a complete chair given only a view of the back leg). We also show that the learned autoregressive prior can be leveraged for conditional tasks such as single-view reconstruction and language-based generation. This is achieved by learning task-specific 'naive' conditionals which can be approximated by light-weight models trained on minimal paired data. We validate the effectiveness of the proposed method using both quantitative and qualitative evaluation and show that the proposed method outperforms the specialized state-of-the-art methods trained for individual tasks.*

*indicates equal contribution

## 1. Introduction

3D representations are essential for applications in robotics, self-driving, virtual/augmented reality, and online marketplaces. This has led to an increasing number of diverse tasks that rely on effective 3D representations – a robot might need to predict the shape of the objects it encounters, an artist may want to imagine what a 'thin couch' would look like, or a woodworker may want to explore possible tabletop designs to match the legs they carved. A common practice for tackling these tasks, such as 3D completion or single-view prediction is to utilize task-specific data and train individual systems for each task, requiring a large amount of compute and data resources.

While tasks such as shape completion or image-conditioned prediction are seemingly different, they require similar outputs – a distribution over the plausible 3D structure conditioned on the corresponding input. A generalized notion of what 'tables' are is useful for both predicting the full shape from the left half and imagining what a 'tall round table' may look like. In this work, we operationalize this observation and show that a generic shape prior can be leveraged across these different inference tasks. In particular, we propose to learn an expressive autoregressive shape prior from abundantly available raw 3D data. This prior can then help augment the task-specific conditional distributions which require paired training data (*e.g.* language-shape pairs), and significantly improve performance when such paired data is difficult to acquire.

Learning such a prior directly over the continuous and high-dimensional space of 3D shapes is computationally intractable. Inspired by recent approaches that overcome similar challenges for image synthesis, we first leverage discrete representation learning to compute discretized and low-dimensional representations for 3D shapes. This not only preserves the essential information for decoding high-quality outputs but also makes the training of autoregressive models tractable. Moreover, to learn such a prior for a broad set of tasks such as shape completion where arbitrary subsets maybe observed *e.g.* 4 legs of a chair, we propose to learn a 'non-sequential' autoregressive prior *i.e.* one capable of using random subsets as conditioning. To enable this, we also enforce that the discrete elements over which this prior is learned are encoded independently.

We then present a common framework for leveraging our learned prior for conditional generation tasks *e.g.* single-view reconstruction or language-guided generation (see Figure 1). Instead of modeling the complex conditional distribution directly, we propose to approximate it as a product of the prior and task-specific 'naive' conditionals, the latter of which can be learned without extensive training data. Combined with the rich and expressive shape prior, we find that this unified and simple approach leads to improvements over task-specific state-of-the-art methods.

## 2. Related Work

**Autoregressive Modeling.** Autoregressive models [20, 21] factorize the joint distribution over structured outputs into products of conditional distributions ($p(\mathbf{x}) = \Pi p(x_i|x_{<i})$). Unlike GANs [15], these can serve as powerful density estimators [33], are more stable during training [25,33], and can generalize well on held-out data. They have been successfully leveraged for modeling distributions across domains, such as images [8, 20, 25, 34], audio [21], video [18], or language [45], and our work explores their benefits across a broad range of 3D generation tasks.

Following their recent successes in autoregressive modeling [3,6,23,45], our work adapts a Transformer-based [35] architecture. While these approaches typically assume a sequential sampling order, closer to our work, Tulsiani and Gupta [31] extend these to allow non-sequential conditioning, which is important for tasks like completion. However, as they model distributions over low-level pixels, their approach cannot synthesize high-resolution outputs due to the quadratic complexity of Transformers. We therefore propose to first reduce high-dimensional 3D shapes to lower-dimensional discrete representations, and learn an autoregressive prior over this latent space.

We build on the work by van den Oord *et al*. [22] who proposed a method to learn quantized and compact latent representations for images using Vector-Quantized Variational AutoEncoder (VQ-VAE), and later also introduced a hierarchical version [24]. Inspired by Esser *et al*. [12] who learned autoregressive generation over the discrete VQ-VAE representations, our work extends these ideas to the domain of 3D shapes. Different from these prior methods, we learn a non-sequential autoregressive prior while modifying the VQ-VAE architecture to independently encode the symbols, and show that this prior can be leveraged for downstream conditional inference tasks.

**Shape Completion.** Completing full shapes from partial inputs such as discrete parts, or single-view 3D, is an increasingly important task across robotics and graphics. Most recent approaches [1, 7, 30, 47–49] formulate it as performing completion on point clouds and can infer plausible global shapes but have difficulty in either capturing fine-grained details, conditioning on sparse inputs, or generating diverse samples. Our non-sequential autoregressive prior provides an alternative approach for shape completion. Given observations for an arbitrary (and possibly sparse) subregion of the 3D shape, we can sample diverse and high-quality shapes from our learned distribution, and we show that this generic approach performs comparably, if not better than previous specialized methods.

**Single-view Reconstruction.** Inferring the 3D shape from a single image is an inherently ill-posed task – an image of a chair from the back does not remove ambiguities about the shape of its seat. Several approaches

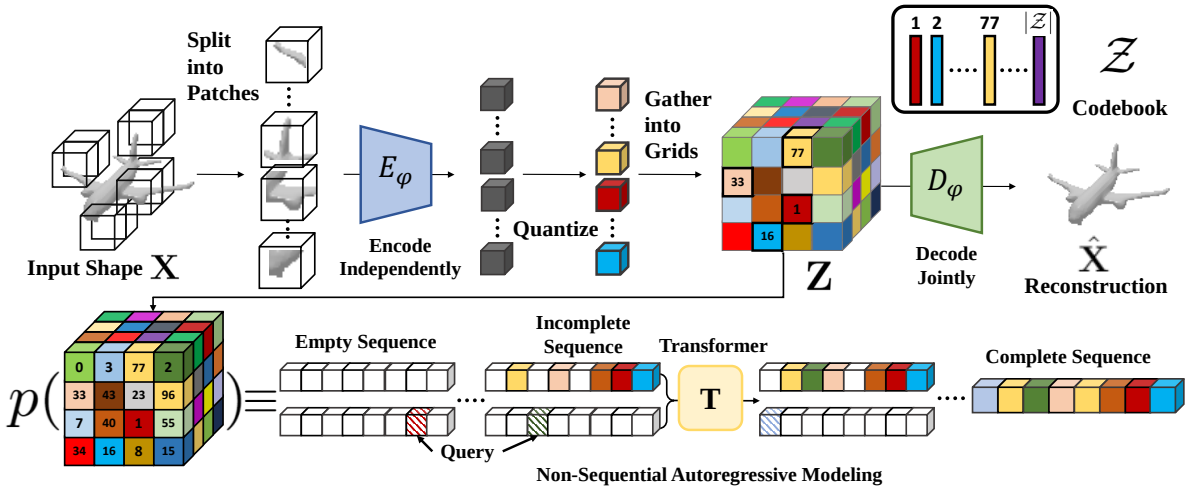



Figure 2. **Overview of Autoregressive Modeling. (top)** We use a VQ-VAE to extract a low-dimensional discrete representation of 3D shapes. Using a patch-wise encoder enables independently encoding local context and allows downstream tasks with partial observations. **(bottom)** We learn a transformer-based autoregressive model over the latent representation. Using randomized sampling orders allows learning a 'non-sequential' autoregressive shape prior that can condition on arbitrary sets of partial latent observations.

have shown impressive single-view reconstruction results using voxels [10, 14, 32, 39, 40], point clouds [13, 19, 42], meshes [37, 38], and most recently implicit representations of 3D surfaces like SDFs [17, 44], UDFs [9] and CSPs [36]. However, these are often deterministic in nature and only generate a 3D single output. By treating image-based prediction as conditional distributions that can be combined with a generic autoregressive prior, our work provides a simple and elegant way of inferring multiple plausible outputs, while also yielding empirical improvements.

**Language-based Generation.** Language is a highly effective and parsimonious modality for describing real-world shapes and objects. Chen *et al*. [5] proposed a method to learn a joint text-shape embedding, followed by a GAN [15] based generator for synthesizing 3D from texts. However, generating shapes from texts is a fundamentally multimodal task, and a GAN-based approach struggles to capture the multiple output modes. In contrast, learning naive-language guided conditional distributions from text aimed at disambiguation shapes [2] and combining these with a generic prior, our work can generate diverse and plausible shapes.

## 3. Approach

We propose an autoregressive method to learn the distribution $p(\mathbf{X})$ over possible 3D shapes $\mathbf{X}$. Our method uses a volumetric Truncated-Signed Distance Field (T-SDF) for representing a 3D shape and learns a Transformer-based [35] neural autoregressive model. However, as the computational complexity of transformers increases quadratically with the input dimension, we first map the high dimensional 3D shape to a corresponding low dimensional, discretized latent space. We then learn a 'non-sequential' autoregressive prior over this compressed discrete representation, and show that this learned prior can be leveraged across diverse conditional generation tasks.

### 3.1. Discretized Latent Space for 3D Shapes

To learn an effective autoregressive model, we aim to reduce the high-dimensional continuous 3D shape representation to a lower-dimensional discrete latent space. Towards this, we adapt the VQ-VAE [22] framework and learn a 3D-VQ-VAE whose encoder $E_\psi$ can compute the desired low-dimensional representation, and the decoder $D_\psi$ can map this latent space back to 3D shapes. Given a 3D shape $\mathbf{X}$ with spatial dimension of $D^3$, we have

$$\mathbf{Z} = VQ(E_\psi(\mathbf{X})), \quad \mathbf{X}' = D_\psi(\mathbf{Z}), \\ \text{where } \mathbf{Z} \in \{1, \cdots, |\mathcal{Z}|\}^{d^3} \tag{1}$$

where $VQ$ is the Vector Quantization step that maps a vector to the nearest element in the codebook $\mathcal{Z}$ which is jointly learned while training the VQ-VAE [22]. The latent representation $\mathbf{Z}$ is thus a 3D grid of elements from the codebook, and can equivalently be thought of as a grid of indices referring to the corresponding codebook entry. We use $z_\mathbf{i}$ to denote the latent variable in the grid $\mathbf{Z}$ at position $\mathbf{i}$.

While the above framework allows learning a compact and quantized latent space, the encoder *jointly* processes an input shape, and thus can use a large receptive field to encode each latent symbol. Unfortunately, this is not a desirable property for tasks such as shape completion since the latent codes for encoded partial shapes may differ significantly from those of the encoded full shape – thus partial observations of shape may not correspond to partial obser-

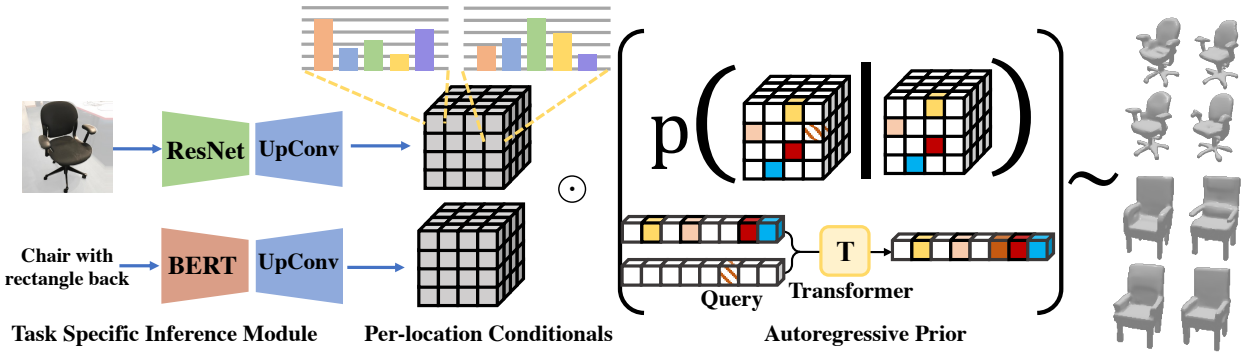



Figure 3. **Overview of conditional generation.** The proposed autoregressive prior can be used across diverse conditional generation tasks. For each task, we use a domain specific encoder followed by 3D up-convolutions to learn task specific conditional distributions. During inference, we can sample from the product distribution of the predicted conditionals and the learned autoregressive prior.

vations of latent variables. To overcome this challenge, we propose Patch-wise Encoding VQ-VAE or P-VQ-VAE that encodes the local shape regions *independently*, while decoding them *jointly* – this allows the discrete encodings to only depend on local context, while still allowing the decoder to reason more globally when generating a 3D shape. We visualize this proposed architecture in Figure 2, and train it using a combination of three losses proposed by van den Oord *et al*. [22]: reconstruction loss, the vector quantization objective, and the commitment loss.

### 3.2. Non-sequential Autoregressive Modeling

The latent space $\mathbf{Z}$ is a 3D grid of tokens representing the original 3D shape. We can thus reduce the task of learning the distribution over continuous 3D shapes to learning $p(\mathbf{Z})$, which is a distribution over the lower-dimensional discrete space. Assuming some ordering of the latent variables *e.g.* a raster scan, typical autoregressive model can approximate this distribution by factorizing it as a product of location specific conditionals: $p(\mathbf{Z}) = \prod_{\mathbf{i}=[1,1,1]}^{[d,d,d]} p(z_\mathbf{i}|z_{<\mathbf{i}})$.

However, this factorization assumes a fixed ordering in which the tokens are observed/generated. More specifically, this factorization implies that we need to know all $z_{<\mathbf{i}}$ before we predict the 'next' symbol $z_\mathbf{i}$. However, such conditioning is not always possible. For example, if we only observe the wheels of a car, the corresponding symbols would not be the first $k$ elements in a predefined sequence but rather occupy some spatially arbitrary locations. To allow for such arbitrary conditioning in our autoregressive model, we propose an autoregressive model which can predict a categorical distribution over tokens conditioned on a random input sequence, and use the term 'non-sequential' autoregressive model to highlight this capability.

We follow the observation from [31] that the joint distribution $P(\mathbf{Z})$ can be factorized into terms of the form $p(z_\mathbf{i}|\mathbf{O})$, where $\mathbf{O}$ is a random set of observed variables. As illustrated in Figure 2, instead of using a rasterized sampling order, we use a randomly permuted sequence of latent variables $\{z_{\mathbf{g}_1}, z_{\mathbf{g}_2}, z_{\mathbf{g}_3}, \cdots\}$ for autoregressively modeling the distribution $P(\mathbf{Z})$:

$$p(\mathbf{Z}) = p_\theta(z_{\mathbf{g}_1}|\varnothing) \cdot p_\theta(z_{\mathbf{g}_2}|z_{\mathbf{g}_1}) \cdot p_\theta(z_{\mathbf{g}_3}|z_{\mathbf{g}_1}, z_{\mathbf{g}_2}) \cdots \quad (2)$$

We model the distribution $p_\theta(z_\mathbf{i}|\mathbf{O})$ using a transformer-based architecture which is parameterized by $\theta$ and takes in an arbitrary set $\mathbf{O} \equiv (\{\mathbf{g}_j\}_{j=1}^k)$ of observed latent variables with known locations and predicts the categorical distribution for an arbitrary query location $i$. We learn this model by simply maximizing the log-likelihood of the encoded latent representations using randomized orders for autoregressive generation. The non-sequential autoregressive network models the distribution over the latent variables $\mathbf{Z}$, which can be mapped to full 3D shapes $\hat{\mathbf{X}} = D_\psi(\mathbf{Z})$. Please see appendix for details.

### 3.3. Conditional Generation

Given the autoregressive model trained to predict the distribution over the latent representation of 3D shapes, we can leverage it to solve various conditional prediction tasks like shape completion, or generation based on modalities like image and language.

**Shape Completion.** The proposed P-VQ-VAE encodes local regions independently. This enables us to map partially observed shape $\mathbf{X_p}$ to corresponding observed latent variables $\mathbf{O} = \{z_{\mathbf{g}_1}, z_{\mathbf{g}_2}, \cdots, z_{\mathbf{g}_k}\}$. Although these observations can be at arbitrary spatial locations, our transformer-based autoregressive model is specifically trained to handle such inputs. In particular, we can formulate the task of shape completion as:

$$p(\mathbf{X}|\mathbf{X_p}) \approx p(\mathbf{Z}|\mathbf{O}) = \prod_{j>k} p_\theta(z_{\mathbf{g}_j}|z_{\mathbf{g}_{<j}}, \mathbf{O}) \quad (3)$$

Based on the above formulation, we can directly use our model from Section 3.2 to autoregressively sample complete latent codes from partial observations. These can then be converted to 3D shapes via the P-VQ-VAE decoder.

**Approximating generic conditional distributions.** While shape completion could be reduced to conditional inference given a partially observed latent code, this reduction does not apply to other generation tasks. More generally, we are interested in inferring shape distributions $p(\mathbf{X}|C)$, where $C$ represents some conditioning *e.g.* an image, or a text description. Approximating this as a distribution over the latent space, our goal is to learn models for $p(\mathbf{Z}|C)$. A possible approach would be to model the terms of the full joint distribution $p(\mathbf{Z}|C) = \prod_i p(z_\mathbf{i}|z_{<\mathbf{i}}, C)$. However, in absence of abundant training data, learning this complex joint distribution may not be feasible.

Instead of modeling this complex distribution, we make a simplifying assumption and propose to model this joint distribution as a product of the shape prior, coupled with independent 'naive' conditional terms that weakly capture the dependence on the conditioning $C$:

$$\prod_j p(z_{\mathbf{g}_j}|z_{\mathbf{g}_{<j}}, C) \approx \prod_j p_\theta(z_{\mathbf{g}_j}|z_{\mathbf{g}_{<j}}) \cdot \prod_j p_\phi(z_{\mathbf{g}_j}|C)$$

This factorization corresponds to assuming a factor graph where the conditioning $C$ is connected to each latent variable $z_\mathbf{i}$ with only a pairwise potential $p(\mathbf{z_i}|C)$. While this is an approximation of the more general case, it enables efficient learning and inference.

**Learning Naive Conditionals.** This per-location distribution intuitively corresponds to an independent 'naive' conditional distribution for each variable in the latent representation. E.g., if the language described a 'thin chair', this term may capture that we expect thin structures around legs. We model this distribution using a neural network parameterized by $\phi$, and can train this network using task-specific paired supervision. In particular, given $(\mathbf{X}, C)$ pairs, we learn $\phi$ by maximizing the log-likelihood $\log p_\phi(z_\mathbf{i}|C)$ for each variable in the encoded shape $\mathbf{Z}$. As illustrated in Figure 3, our task-specific network ($\phi$) comprises of domain-specific Encoders (*e.g.* ResNet [16] for images; BERT [11] for language etc.) followed by up-convolutional decoders to the predict the conditional distribution over elements in $\mathbf{Z}$ *i.e.* $p_\phi(z_\mathbf{i}|C)$.

**Prior-guided conditional inference.** Using the learned task-specific network to model the naive distributions $p_\phi(z_\mathbf{i}|C)$, we can use to combine it with our autoregressive prior to obtain a conditional distribution over shapes which can be used for multimodal generation.

# 4. Experiments

To demonstrate the efficacy of our autoregressive prior for generic 3D tasks, we quantitatively and qualitatively evaluated our method on three tasks – a) shape completion, b) 3D reconstruction, and, c) language-guided generation.

Table 1. **Quantitative comparison on Shape Completion**.

| | Bottom Half | | Octant | |
|---|---|---|---|---|
| Method | UHD ↓ | TMD ↑ | UHD ↓ | TMD ↑ |
| MPC [41] | 0.0627 | 0.0303 | 0.0579 | 0.0376 |
| PoinTr [46] | 0.0572 | N/A | **0.0536** | N/A |
| Ours | **0.0567** | **0.0341** | 0.0599 | **0.0693** |

## 4.1. Multi-modal Shape Completion

Our learned non-sequential autoregressive prior can naturally be adapted to the task of shape completion. We encode partial observations (in the form of local TSDFs) to obtain discrete symbols via the patchwise VQ-VAE encoder for the seen regions and sample full shapes conditioned on these observed symbols using the autoregressive prior.

**Baselines and Evaluation Setup.** We evaluate our approach on the ShapeNet [4] dataset using the train/test splits provided by Xu *et al*. [44]. We use two completion settings for evaluation with varying fraction of observed shapes:

- *Bottom half* of complete shape as input
- *Octant* with front, left and bottom half of complete shape as input

We compare our generations against two state-of-the-art point-cloud completion methods, MPC [41] and PoinTr [46]. The former can generate multiple plausible shapes given the partial input, whereas the latter generates only a single (more accurate) completion. Both the publicly available PointTr [46] and our approach can use a single model to handle genetic shape completion scenarios. However, MPC needs to train a *separate* model to handle completion in each different scenario. As our approach uses a partial TSDF input, it can potentially 'see' information (up to a small threshold) beyond the boundaries. For a fair comparison, we also give the baseline methods additional points within the truncation threshold.

**Evaluation Metrics.** We adopt the metrics from MPC [41] for the quantitative evaluation. These are given below. For each partial shape, we generate $k(= 10)$ complete shapes.

- *Completion fidelity*: we compute the average of *Unidirectional Hausdorff Distance* (UHD) from the input partial shape to the $k$ generated shapes. This measures the completion fidelity given the partial inputs.
- *Completion diversity*: given $k$ generated results for each shape, we compute the average Chamfer distance to other $k-1$ shapes. The sum of the average distance among $k$ generation assesses the completion diversity and is denoted as *Total Mutual Difference* (TMD).

**Results.** To compare the performance of our approach with the baselines on the task of shape completion, we use a set of held-out chairs from the ShapeNet dataset. Please note that while all the methods are trained across all

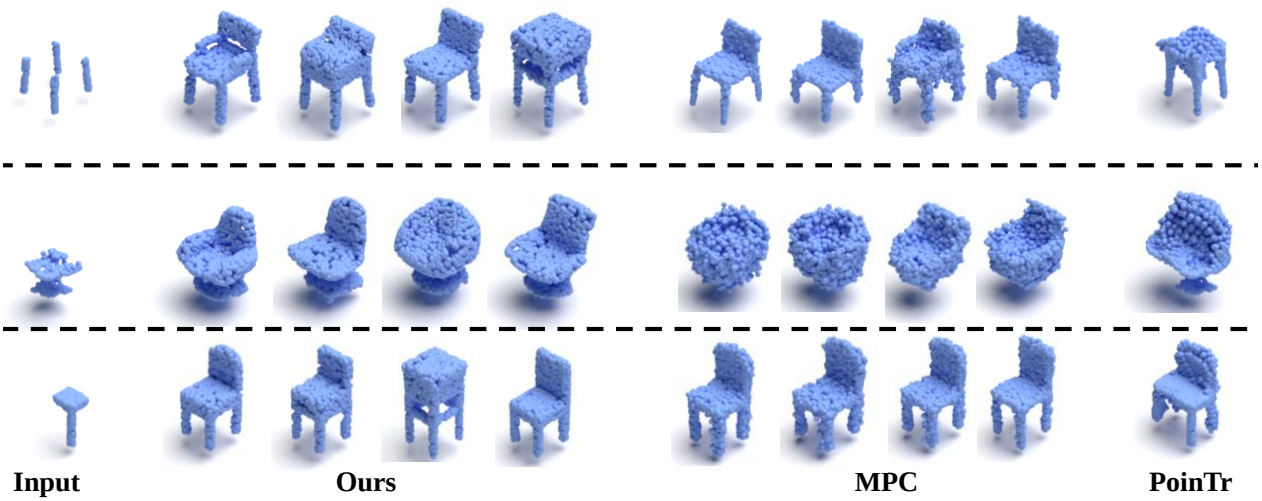



Figure 4. **Comparative results for Shape Completion.** Given the partial inputs, we visualize the generated results from different methods. Our approach yields more diverse generations, while also better preserving the originally observed structure. For example, in the first row, given 4 slanted legs of a chair, some MPC generations make them straighter in the full point cloud, while they are preserved in our approach

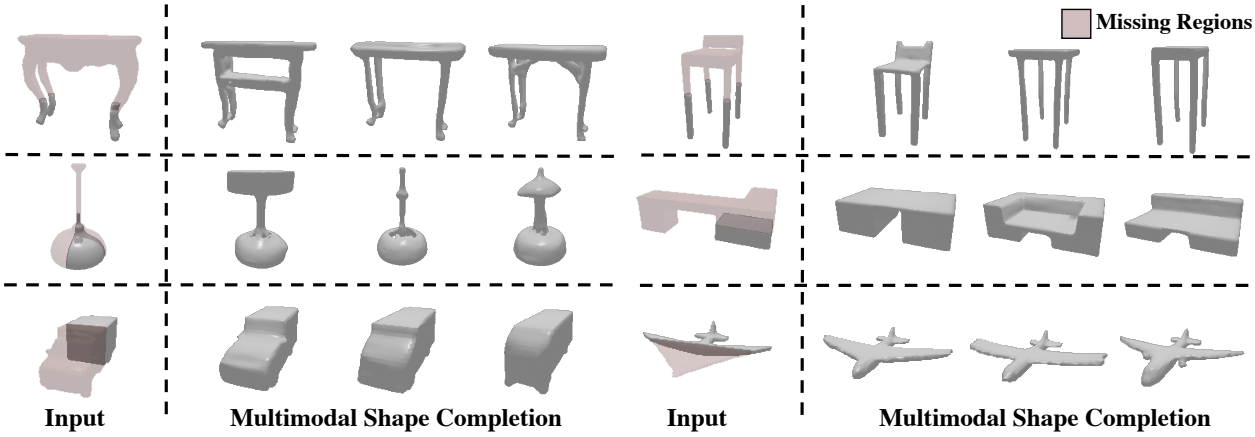



Figure 5. **Qualitative results for Shape Completion.** Our proposed approach is able to generate diverse plausible 3D shapes consistent with the partial input. The generated shapes are visually consistent with realistic shapes even with significantly missing parts( in **Red**)

ShapeNet classes, the evaluation is performed on a limited subset for computational reasons. The quantitative results reported in Table 1 demonstrate that our autoregressive-prior based completion method performs favorably against the baselines, both in terms of fidelity and diversity on the two protocols.

We also show qualitative comparisons to these baselines in Figure 4 and observe that our approach yields more diverse generations, while also better preserving the originally observed structure *e.g.* given 4 slanted legs of a chair, some MPC generations make them straighter in the full point cloud, while the slants are preserved across the shapes generated by our approach. More shape completion results across other diverse shapes are shown in Figure 5. Our approach, while appropriately conditioning on the partial observations, generates a rich variety of diverse, high-quality and realistic 3D shapes. It is noteworthy that although our autoregressive model is trained only on random observation sequences, it is able to condition on structured partial observations (anchored on a correlated set of anchored locations) that it was never trained explicitly to complete (unlike the point cloud completion baselines which are specifically trained for this task).

### 4.2. Single-view 3D Prediction

We next show that the learned prior can be leveraged for the task of single-view 3D reconstruction. To obtain the per-location image-conditioning, we train a modified ResNet [16] using pairs of images and corresponding encoded 3D models from the training dataset.

**Evaluation Setups.** We evaluate the proposed method on the ShapeNet rendered images [10], and the real-world

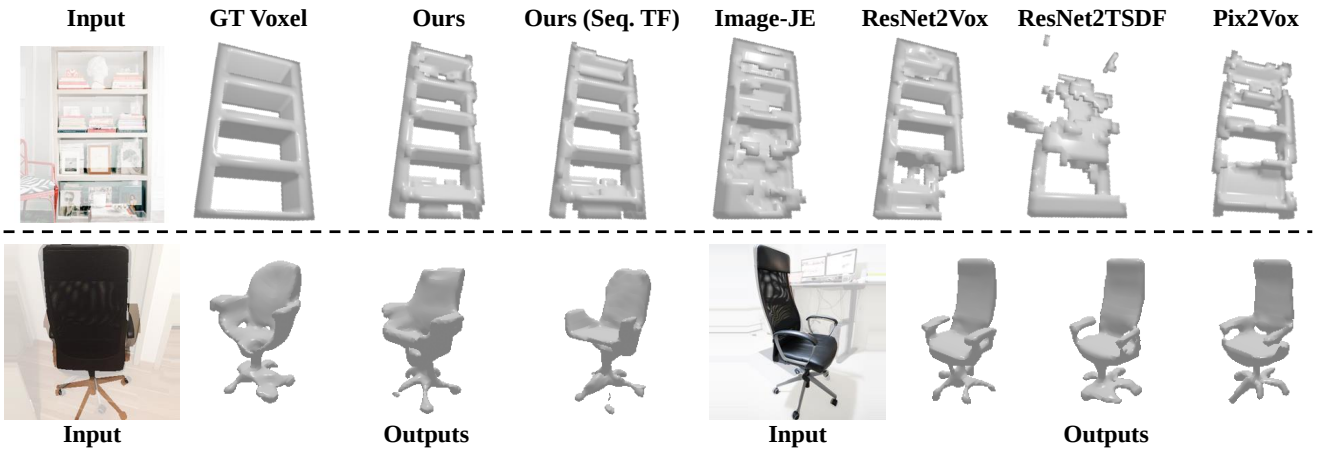



Figure 6. **Single-view 3D reconstruction**. (**Top**) We show a sample single-view reconstruction using our approach and other baselines – our shape prior helps the generated shape be more globally coherent. (**Bottom**) We visualize multiple shapes predicted by our approach given the input images. We observe meaningful shape variation in the unobserved regions *e.g.* front of the chair in the left image.

Table 2. **Quantitative evaluation of single-view reconstruction**.

| | ShapeNet | | | Pix3D | | |
|---|---|---|---|---|---|---|
| Method | IoU ↑ | CD ↓ | F-Score ↑ | IoU ↑ | CD ↓ | F-Score ↑ |
| Pix2Vox | 0.467 | 2.521 | 0.335 | 0.504 | 3.001 | 0.385 |
| ResNet2TSDF | 0.478 | 2.684 | 0.320 | 0.475 | 4.582 | 0.351 |
| ResNet2Voxel | 0.457 | 2.501 | 0.316 | 0.505 | 4.670 | 0.357 |
| Image-JE | 0.486 | 1.972 | 0.338 | 0.480 | 2.983 | 0.394 |
| Ours (Sequential) | 0.554 | 1.448 | 0.393 | 0.516 | **2.254** | 0.412 |
| Ours | **0.577** | **1.331** | **0.414** | **0.521** | 2.267 | **0.415** |

benchmark Pix3D [27], using cropped and segmented images as input for reconstruction. For ShapeNet, we use the same train/test split provided by Xu *et al*. [44], and evaluate against the voxelized models provided by Choy *et al*. [10]. For Pix3D, we use the provided train/test splits for the chair category. In the absence of official splits for other categories, we randomly split the dataset into disjoint 3D shapes for training and testing. We evaluate all methods on the ground truth voxels in Pix3D and follow the official implementation to downsample all predictions into $32^3$ of voxels for evaluation. We use 3D IoU, Chamfer Distance (CD), and F-score@$1\%$ [29] as the metrics to measure the performance across different methods.

**Baselines** We compare with the following methods:

- Pix2Vox [43]: a state-of-the-art approach for 3D reconstruction.
- ResNet2TSDF / ResNet2Voxel: baselines which use a similar ResNet to ours but directly decode the output shape without using any shape prior.
- Image-JE: a transformer-based baseline to predict $P(S_i|S_{<i}, I)$ jointly, where $I$ is the input image.
- Ours (Sequential): a variant of the proposed method where the transformer

**Results.** Quantitative results in Table 2 demonstrate that our proposed method performs favorably across almost all categories. Please refer to In Figure 6, we show some representative results with comparisons against the competing methods (additional results are in the supplementary). More crucially, as shown in the second row of Figure 6, unlike baselines, our approach can generate multiple plausible shapes given an input image. For example, given an image with a back-view of a chair , our model produces diverse reconstruction results with meaningful variation in unobserved regions, such as different armrests or cushions with varied shapes.

### 4.3. Language-guided Generation

Achlioptas *et al*. [2] released a dataset containing text utterances describing the distinction between a target chair and the two distractors from Shapenet [4]. We repurpose this data to train a text-conditioned generative model as described in Section 3.3. The distribution from this conditional when combined with our autoregressive prior, allows us to generate diverse shapes given a language description.

We compare our method with Text2Shape (T2S) [5] on the text-guided 3D shape generation task. While T2S was originally trained to generate color and shape from text descriptions, after finetuning it on our dataset [2], we only use the generated shape for comparison. In addition, we also compare our method with a transformer-based Encoder-Decoder Model (JE) trained on [2] to predict $P(S_i|S_{<i}, T)$ jointly which serves as a baseline where a joint distribution is learned as opposed to our factored approach. Our approach combines two factors: a generic prior factor with an input modality dependent conditional factor. The latter may be potentially weak depending on the parsimony of the input modality and the amount of training data available.

**Quantitative Comparison.** To enable a comparison on this task, we train a neural evaluator similar to the one proposed in [2]. The evaluator is trained to distinguish the

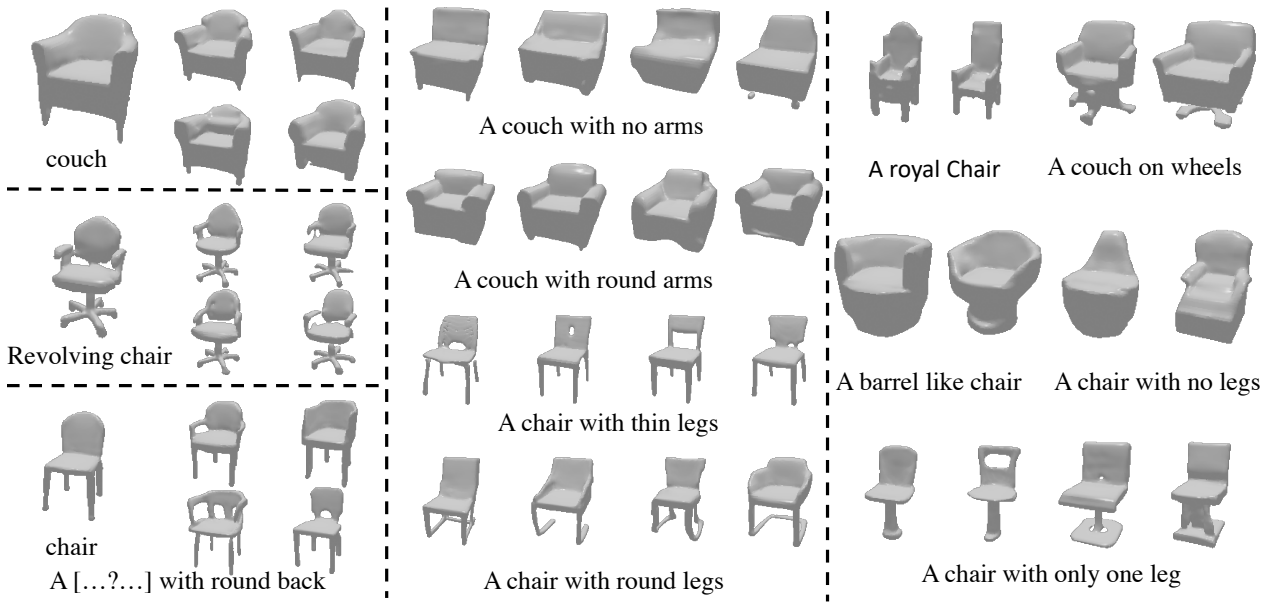



Figure 7. **Language Conditioned Generation**. The results signify that our approach can meaningfully estimate the correlation between input description and correspondingly plausible shapes while simultaneously generate the missing context required to generate them.

target shape from a distractor given the specified text, and achieves a $\sim$ 83% accuracy on this binary classification task. We use descriptions corresponding to held out test objects for evaluation. For each description, we provide the evaluator with two generations and report the quantitative results in Table 3. We label an instance as 'confused' when the absolute difference between the evaluator's confidence is $\leq 0.2$. We perform a seven way comparison by reporting the following comparisons – Ours vs T2S, JE and Ours(Sequential), and, GT vs Ours, Ours (Sequential) , JE, and, T2S. We find that our approach is the preferred choice, with a very large margin, over either of the baselines (66:18 over T2S and 61:23 over JE). In a direct comparison with the GT, our generations are preferred 30% of the time while the GT is preferred 49%, and is significantly better than the other three baselines.

**Qualitative Results.** While the quantitative evaluation above has been conducted for a smaller set of descriptions in the dataset, the model is trained over a larger set and can conditionally generate shapes for many more generic descriptions. We present an exemplary set of generated samples conditioned on a variety of input text in Figure 7. The results clearly demonstrate that our approach can (a) generate highly plausible and realistic shapes correlated to the input description, (b) the shapes co-vary in a reasonable fashion with variations in the input text, *e.g.* couch, revolving chair *etc*., and, (c) that even when the descriptions refer only to a specific part *e.g.* 'a chair with one leg', our model generates coherent global shapes consistent with this description. Please see appendix for additional results.

Table 3. **Language-guided Generation**.

| Target (Tr) | Distractor (Dis) | $P(Tr)$ | $P(Dis)$ | $P(Conf)$ |
|---|---|---|---|---|
| Ours | Text2Sshape [5] | 66% | 18% | 16% |
| Ours | Joint-Encoding | 61% | 23% | 16% |
| Ours | Ours (Sequential) | 34% | 27% | 39% |
| Ours | GT | 30% | 49% | 21% |
| Ours (Sequential) | GT | 28% | 52% | 20% |
| Text2Sshape [5] | GT | 15% | 74% | 11% |
| Joint-Encoding | GT | 19% | 67% | 14% |

## 5. Discussion

We proposed an approach for learning a generic non-sequential autoregressive prior over 3D shapes useful for *multi-modal* generation for a diverse set of tasks *e.g.* shape completion, single-view reconstruction, and language-guided synthesis. We find it encouraging that our unifying approach yields compelling results across these different tasks and is competitive with specifically designed baselines. However, a limitation of our conditional inference formulation is that it can only approximate the joint distribution – while this helps in the low-paired data regime, this would be suboptimal with large-scale task-specific data. Moreover, our approach only applies to spatially structured 3D representations *e.g.* TSDF or voxels, and it is not obvious whether our autoregressive modeling framework can be adapted to other 3D representations such as meshes or neural implicit functions [26]. Second, the proposed method might also be sensitive with respect to shape alignment. Finally, our learned shape prior is biased towards artificial categories with abundantly available CAD models, and cannot be leveraged for 3D generation beyond these.

## References


[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018. 2, 14

[2] Panos Achlioptas, Judy Fan, Robert Hawkins, Noah Goodman, and Leonidas J Guibas. Shapeglot: Learning language for shape differentiation. In *CVPR*, 2019. 3, 7, 14, 15

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020. 2

[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. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 5, 7, 11, 14, 15

[5] 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 *ACCV*, 2018. 3, 7, 8, 15

[6] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020. 2

[7] Xuelin Chen, Baoquan Chen, and Niloy J Mitra. Unpaired point cloud completion on real scans using adversarial training. In *ICLR*, 2020. 2

[8] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. In *ICML*, 2018. 2

[9] Julian Chibane and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In *NeurIPS*, 2020. 3

[10] 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 *ECCV*, 2016. 3, 6, 7

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019. 5, 11, 12, 13, 15

[12] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 2

[13] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2017. 3

[14] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016. 3

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 2, 3

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 6, 13, 14

[17] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *CVPR*, 2020.

[18] Nal Kalchbrenner, Aäron Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *ICML*, 2017. 2

[19] Priyanka Mandikal, Navaneet K. L., Mayank Agarwal, and Venkatesh Babu Radhakrishnan. 3d-lmnet: Latent embedding matching for accurate and diverse 3d point cloud reconstruction from a single image. In *BMVC*, 2018. 3

[20] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016. 2

[21] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016. 2

[22] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017. 2, 3, 4, 11

[23] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *ICML*, 2018. 2

[24] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *NeurIPS*, 2019. 2

[25] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017. 2

[26] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34, 2021. 8

[27] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *CVPR*, 2018. 7

[28] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS*, 2020. 11

[29] Maxim Tatarchenko*, Stephan R. Richter*, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? 2019. 7

[30] Lyne P Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. Topnet: Structural point cloud decoder. In *CVPR*, 2019. 2

[31] Shubham Tulsiani and Abhinav Gupta. Pixeltransformer: Sample conditioned signal generation. In *ICML*, 2021. 2, 4, 11

[32] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, 2017. 3

[33] Benigno Uria, Iain Murray, and Hugo Larochelle. Rnade: The real-valued neural autoregressive density-estimator. In *NeurIPS*, 2013. 2

[34] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In *NeurIPS*, 2016. 2

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2, 3

[36] Rahul Venkatesh, Tejan Karmali, Sarthak Sharma, Aurobrata Ghosh, R. Venkatesh Babu, László A. Jeni, and Maneesh Singh. Deep implicit surface point prediction networks. In *ICCV*, 2021. 3

[37] 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*, pages 52–67, 2018. 3

[38] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 3dn: 3d deformation network. In *CVPR*, 2019. 3

[39] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, William T Freeman, and Joshua B Tenenbaum. MarrNet: 3D Shape Reconstruction via 2.5D Sketches. In *NeurIPS*, 2017. 3

[40] Jiajun Wu, Chengkai Zhang, Xiuming Zhang, Zhoutong Zhang, William T Freeman, and Joshua B Tenenbaum. Learning shape priors for single-view 3d completion and reconstruction. In *ECCV*, 2018. 3

[41] Rundi Wu, Xuelin Chen, Yixin Zhuang, and Baoquan Chen. Multimodal shape completion via conditional generative adversarial networks. In *ECCV*, 2020. 5

[42] Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. Pq-net: A generative part seq2seq network for 3d shapes. In *CVPR*, 2020. 3

[43] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In *CVPR*, pages 2690–2698, 2019. 7

[44] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *NeurIPS*, 2019. 3, 5, 7, 11

[45] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *NeurIPS*, 2019. 2

[46] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. PoinTr: Diverse point cloud completion with geometry-aware transformers. In *CVPR*, 2021. 5

[47] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *3DV*, 2018. 2

[48] Junzhe Zhang, Xinyi Chen, Zhongang Cai, Liang Pan, Haiyu Zhao, Shuai Yi, Chai Kiat Yeo, Bo Dai, and Chen Change Loy. Unsupervised 3d shape completion through gan inversion. In *CVPR*, 2021. 2

[49] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *ICCV*, 2021. 2

# Supplementary Materials:
# AutoSDF: Shape Priors for 3D Completion, Reconstruction and Generation

We provide the implementation details of the P-VQ-VAE, transformer, image/text inference module, and the baselines such as ResNet2TSDF, ResNet2Voxel, Joint Text-Shape Baseline. For the visualization of the generated 3D shapes, please check our project page at https://yccyenchicheng.github.io/AutoSDF/.

## A. Implementation Detail

We will release our code and learned models for reproducibility, but also describe here the experiments in additional detail.

### A.1. VQ-VAE Training

**Dataset Details.** We train our proposed P-VQ-VAE using the objects from 13 categories of Shapenet [4] data. These categories are [airplane, bench, cabinet, car, chair, display, lamp, speaker, rifle, sofa, table, phone, watercraft]. To extract SDF, we follow the preprocessing steps in DISN [44] and PixelTransformer [31]. The shapes are normalized to lie in a origin-centered cube in $[-1, 1]^3$, and their signed distance function is evaluated at locations in a uniformly sampled $64^3$ grid. We use $0.2$ as the threshold to further obtain the Truncated-SDF (T-SDF) representations.

**Training Details.** Given a 3D shape $\mathbf{X} \in \mathbb{R}^{64^3}$, we first split $\mathbf{X}$ into 512 patches of $\mathbf{X}_p \in \mathbb{R}^{8^3}$. After $E_\psi$ encodes each patch independently, we have the latent code $\hat{z}_\mathbf{i}$ for each patch at location $\mathbf{i}$. We perform vector quantize step $VQ$ for all $\hat{z}_\mathbf{i}$ to obtain $z_\mathbf{i}$. Gathering $\hat{z}_\mathbf{i}$ and $z_\mathbf{i}$ for all location $\mathbf{i}$ into grids gives us the complete latent space for $\hat{\mathbf{Z}}$ and $\mathbf{Z}$ respectively. Finally, $D_\psi$ decodes $\mathbf{Z}$ to output the reconstruction $\mathbf{X}'$. We then use the training objective proposed in VQ-VAE work [22]:

$$\begin{aligned}\mathcal{L}_{\text{VQ}-\text{VAE}} = -\log p(X|\mathbf{Z}) + \left\|\text{sg}[\hat{\mathbf{Z}}] - \mathbf{Z}\right\|^2 \\ + \left\|\hat{\mathbf{Z}} - \text{sg}[\mathbf{Z}]\right\|^2 .\end{aligned} \tag{4}$$

where the first term is the reconstruction loss and $\text{sg}[\cdot]$ denotes the stop gradients. The second and third term in Eq. 4 is the $VQ$ objective and the commitment loss respectively.

**Architecture Details.** We describe the details for P-VQ-VAE's encoder $E_\psi$ at Table 4, decoder $D_\psi$ at Table 5. For the hyperparameters of the Codebook $\mathcal{Z}$, we use number of codebook entries 512, and the dimensionality of $z \in \mathbb{R}^{256}$.

### A.2. Non-sequential Autoregressive Modeling

**Training Details.** Given the learned P-VQ-VAE, we train our non-sequential autoregressive model over the low-dimensional shape space. Given a 3D shape $\mathbf{X}$, we first use $E_\psi$ and the codebook $\mathcal{Z}$ to obtain its discrete representation $\mathbf{Z}$ in the latent space. Given this low-dimensional and discrete representation, we autoregressively model $p(\mathbf{Z})$ using a randomly permuted order of latent variables $\{z_{\mathbf{g}_1}, z_{\mathbf{g}_2}, z_{\mathbf{g}_3}, \cdots\}$ for factorizing the joint distribution:

$$p(\mathbf{Z}) = p_\theta(z_{\mathbf{g}_1}|\varnothing) \cdot p_\theta(z_{\mathbf{g}_2}|z_{\mathbf{g}_1}) \cdot p_\theta(z_{\mathbf{g}_3}|z_{\mathbf{g}_1}, z_{\mathbf{g}_2}) \cdots \tag{5}$$

In our implementation, the input to the transformer $\mathbf{T}$ are all the elements from $\{z_{\mathbf{g}_1}, z_{\mathbf{g}_2}, \cdots, z_{\mathbf{g}_{511}}\}$, and the query locations for the next elements $\{\mathbf{i}_{\mathbf{g}_2}, \mathbf{i}_{\mathbf{g}_3}, \cdots, \mathbf{i}_{\mathbf{g}_{512}}\}$. The outputs of $\mathbf{T}$ are the probability distributions over the codebook elements. For instance, to predict the $\mathbf{g}_i$-th element in the sequence, we have

$$p(z_{\mathbf{g}_i}) = \mathbf{T}(\{z_{\mathbf{g}_1}, z_{\mathbf{g}_2}, \cdots, z_{\mathbf{g}_{<i}}\}, \mathbf{i}_{\mathbf{g}_i}) . \tag{6}$$

In practice, the training is done in parallel and the targets are the elements starting from $\mathbf{g}_2$: $\{z_{\mathbf{g}_2}, z_{\mathbf{g}_3}, \cdots, z_{\mathbf{g}_{512}}\}$. We feed the attention mask with upper-triangular matrix of $-\infty$, and zeros on the diagonal to make sure the information do not leak from the future elements. We use fourier features for the positional embedding for all locations $\mathbf{i}$ following Tancik *et al*. [28]. The training objective is minimizing the expected negative log-likelihood, where we sample random orders or variables in every iteration:

$$\mathcal{L}_\mathbf{T} = \mathbb{E}_{\mathbf{X}\sim p(\mathbf{X})} - [\log p(\mathbf{Z})] . \tag{7}$$

**Inference.** During inference, the transformer can achieve autoregressive generation by completing either an empty sequence (unconditional generation) or incomplete sequence (shape completion). Given an arbitrary set $\mathbf{O} \equiv (\{\mathbf{g}_j\}_{j=1}^k)$ of observed latent variables and their locations, we repeated apply

$$p(z_{\mathbf{g}_t}) = \mathbf{T}(\{\mathbf{O}, \hat{z}_{\mathbf{g}_{k+1}}, \cdots, \hat{z}_{\mathbf{g}_{t-1}}\}, \mathbf{i}_{\mathbf{g}_t}) . \tag{8}$$

for $t = k+1, k+2, \cdots, 512$. $\hat{z}_t$ is obtained by sampling from the probability distribution $p(z_{\mathbf{g}_t})$. Once we have the complete sequence, we feed the estimated $\mathbf{Z}$ into $D_\psi$ to output the 3D shape.

**Architecture Detail.** We adopt a transformer based architecture to learn the autoregressive prior over 3D shapes. Our model consists of 12 Encoder layers, with 12 multi-head attention heads and a hidden dimension of 768. In similar spirit as BERT [11] our transformer does not contain a 'Decoder' i.e. all attention layers are self-attention. We directly reuse the learned codebook for the input embedding of each token.

Table 4. **Architecture for the P-VQ-VAE's encoder $E_\psi$.**

| Layer name | Weights/Parameters | Input size | Output size |
|---|---|---|---|
| Conv3D | kernel $3 \times 3$, stride 1, padding 1 | $B \times 1 \times 8 \times 8 \times 8$ | $B \times 64 \times 8 \times 8 \times 8$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 64 \times 8 \times 8 \times 8$ | $B \times 64 \times 8 \times 8 \times 8$ |
| Downsample | kernel $3 \times 3$, stride 2, padding 0 | $B \times 64 \times 8 \times 8 \times 8$ | $B \times 64 \times 4 \times 4 \times 4$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 64 \times 4 \times 4 \times 4$ | $B \times 128 \times 4 \times 4 \times 4$ |
| Downsample | kernel $3 \times 3$, stride 2, padding 0 | $B \times 128 \times 4 \times 4 \times 4$ | $B \times 128 \times 2 \times 2 \times 2$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 128 \times 2 \times 2 \times 2$ | $B \times 128 \times 2 \times 2 \times 2$ |
| Downsample | kernel $3 \times 3$, stride 2, padding 0 | $B \times 128 \times 2 \times 2 \times 2$ | $B \times 128 \times 1 \times 1 \times 1$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 128 \times 1 \times 1 \times 1$ | $B \times 256 \times 1 \times 1 \times 1$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 256 \times 1 \times 1 \times 1$ | $B \times 256 \times 1 \times 1 \times 1$ |
| 3D Attention Block | - | $B \times 256 \times 1 \times 1 \times 1$ | $B \times 256 \times 1 \times 1 \times 1$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 256 \times 1 \times 1 \times 1$ | $B \times 256 \times 1 \times 1 \times 1$ |
| GroupNorm | num_groups=32 | $B \times 256 \times 1 \times 1 \times 1$ | $B \times 256 \times 1 \times 1 \times 1$ |
| Swish | - | $B \times 256 \times 1 \times 1 \times 1$ | $B \times 256 \times 1 \times 1 \times 1$ |
| Conv3D | kernel $3 \times 3$, stride 1, padding 1 | $B \times 256 \times 1 \times 1 \times 1$ | $B \times 256 \times 1 \times 1 \times 1$ |

Table 5. **Architecture for the P-VQ-VAE's decoder $D_\psi$.**

| Layer name | Weights/Parameters | Input size | Output size |
|---|---|---|---|
| Conv3D | kernel $3 \times 3$, stride 1, padding 1 | $B \times 256 \times 1 \times 1 \times 1$ | $B \times 256 \times 1 \times 1 \times 1$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 256 \times 1 \times 1 \times 1$ | $B \times 256 \times 1 \times 1 \times 1$ |
| 3D Attention Block | - | $B \times 256 \times 1 \times 1 \times 1$ | $B \times 256 \times 1 \times 1 \times 1$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 256 \times 1 \times 1 \times 1$ | $B \times 256 \times 1 \times 1 \times 1$ |
| Upsample | kernel $3 \times 3$, stride 2, padding 0 | $B \times 256 \times 1 \times 1 \times 1$ | $B \times 256 \times 2 \times 2 \times 2$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 256 \times 2 \times 2 \times 2$ | $B \times 128 \times 2 \times 2 \times 2$ |
| 3D Attention Block | - | $B \times 128 \times 2 \times 2 \times 2$ | $B \times 128 \times 2 \times 2 \times 2$ |
| Upsample | kernel $3 \times 3$, stride 2, padding 0 | $B \times 128 \times 2 \times 2 \times 2$ | $B \times 128 \times 4 \times 4 \times 4$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 128 \times 4 \times 4 \times 4$ | $B \times 64 \times 4 \times 4 \times 4$ |
| Upsample | kernel $3 \times 3$, stride 2, padding 0 | $B \times 64 \times 4 \times 4 \times 4$ | $B \times 64 \times 8 \times 8 \times 8$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 64 \times 8 \times 8 \times 8$ | $B \times 64 \times 8 \times 8 \times 8$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 64 \times 8 \times 8 \times 8$ | $B \times 64 \times 8 \times 8 \times 8$ |
| GroupNorm | num_groups=32 | $B \times 64 \times 8 \times 8 \times 8$ | $B \times 64 \times 8 \times 8 \times 8$ |
| Swish | - | $B \times 64 \times 8 \times 8 \times 8$ | $B \times 64 \times 8 \times 8 \times 8$ |
| Conv3D | kernel $3 \times 3$, stride 1, padding 1 | $B \times 64 \times 8 \times 8 \times 8$ | $B \times 1 \times 8 \times 8 \times 8$ |

## A.3. Naive Condition Predictions

**Image Inference Module: Training Details.** Given a pair of image and its corresponding 3D shape $(I, \mathbf{X})$, we first use a pretrained ResNet-18 to extract the image features. After that, we adopt a linear layer to lift the image features into 3D followed by several UpConv3D layers to output $8^3$ grids. Each grid contains the probability distribution $p(\mathbf{z_i}|I)$ over the codebook entries learned by the P-VQ-VAE. We train the image inference module using a cross entropy loss, where the ground truth classes is given by $\mathbf{Z} = E_\psi(\mathbf{X})$. The architecture details for image inference module is shown at Table 6.

**Language Inference Module.** We encode text descriptions using a pre-trained BERT [11] encoder. Since text

Table 6. **Architecture for the image inference module**.

| Layer name | Weights/Parameters | Input size | Output size |
|---|---|---|---|
| ResNet-18 [16] | - | $B \times 3 \times 256 \times 256$ | $B \times 512 \times 8 \times 8$ |
| Linear_to_3D | - | $B \times 512 \times 64$ | $B \times 512 \times 512$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 512 \times 8 \times 8 \times 8$ | $B \times 256 \times 8 \times 8 \times 8$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 256 \times 8 \times 8 \times 8$ | $B \times 128 \times 8 \times 8 \times 8$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 128 \times 8 \times 8 \times 8$ | $B \times 64 \times 8 \times 8 \times 8$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 64 \times 8 \times 8 \times 8$ | $B \times 64 \times 8 \times 8 \times 8$ |
| Conv3D | kernel $3 \times 3$, stride 1, padding 1 | $B \times 64 \times 8 \times 8 \times 8$ | $B \times 512 \times 8 \times 8 \times 8$ |

Table 7. **Architecture for the language inference module**.

| Layer name | Weights/Parameters | Input size | Output size |
|---|---|---|---|
| BERT [11] | 12 Layers, 12 Attention heads, 768 hidden dim | $B \times Seq$ | $B \times 768$ |
| Linear_up | - | $B \times 768$ | $B \times 1024$ |
| Linear_down | - | $B \times 1024$ | $B \times 512$ |
| Conv3D | kernel $3 \times 3$, stride 1, padding 1 | $B \times 1 \times 8 \times 8 \times 8$ | $B \times 64 \times 8 \times 8 \times 8$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 64 \times 8 \times 8 \times 8$ | $B \times 128 \times 8 \times 8 \times 8$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 128 \times 8 \times 8 \times 8$ | $B \times 256 \times 8 \times 8 \times 8$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 256 \times 8 \times 8 \times 8$ | $B \times 256 \times 8 \times 8 \times 8$ |
| Conv3D | kernel $3 \times 3$, stride 1, padding 1 | $B \times 256 \times 8 \times 8 \times 8$ | $B \times 512 \times 8 \times 8 \times 8$ |

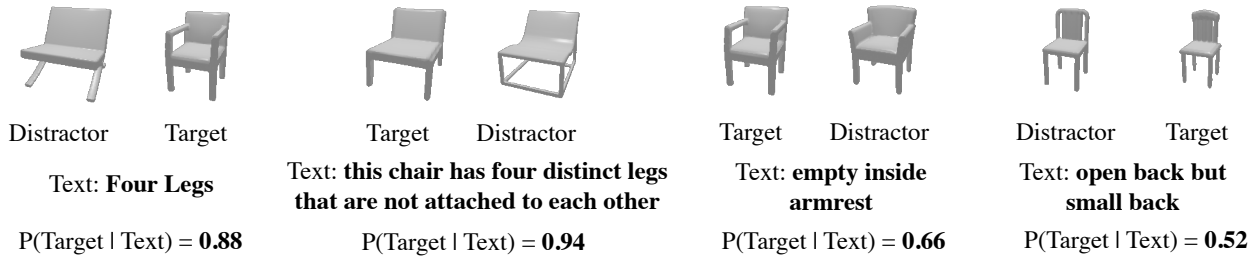



Figure 8. **Prediction from neural evaluator** Neural evaluator is trained to predict the target shape from two candidate shapes given a text description. Target indicates the true shape and P(Target — Text) indicates the prediction confidence for a given input. This highlights that neural evaluator can do a meaningful job in associating text descriptions to shapes. Low confidence in right most input instance indicates that the model confuses when input shapes are somewhat similar

can be of varying length, we follow a common practice in language classification tasks and use the pooler output: a linear transformation over first token's representation as the output for learning the conditional. Following a similar architecture as image conditioning, we adopt linear layers to lift text-features into 3D. A group of UpConv3D layers are further used to produce the $8^3$ grid. This module is trained with cross entropy loss and exact architecture details are included in Table 7.

**Weight parameter $\alpha$ between the shape prior and the conditional marginal.** For the per-location condition, we use a hyperparameter $\alpha$ to balance the condition between the shape prior and the conditional marginals. Specifically, in the equation we use to approximate the conditional distribution,

$$\prod_j p(z_{\mathbf{g}_j}|z_{\mathbf{g}_{<j}}, C) \approx \prod_j p_\theta(z_{\mathbf{g}_j}|z_{\mathbf{g}_{<j}})^{1-\alpha} \cdot \prod_j p_\phi(z_{\mathbf{g}_j}|C)^\alpha$$

we control the dependence on the conditioning $C$ by raising power of the shape prior to $1-\alpha$ and the condition marginal to $\alpha$, where $\alpha \in [0, 1]$. We use $\alpha = 0.75$ across all experiments on image conditioning. For language conditional generation, we find $\alpha = 0.5$ to work better.

## A.4. Experimental Details

**Details of the Baselines for Single-view 3D Prediction.** We describe the training and architecture details in this

Table 8. **Architecture for the ResNet2TSDF**.

| Layer name | Weights/Parameters | Input size | Output size |
|---|---|---|---|
| ResNet-18 [16] | - | $B \times 3 \times 256 \times 256$ | $B \times 512 \times 8 \times 8$ |
| Linear_to_3D | - | $B \times 512 \times 64$ | $B \times 512 \times 512$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 512 \times 8 \times 8 \times 8$ | $B \times 256 \times 8 \times 8 \times 8$ |
| Upsample | kernel $3 \times 3$, stride 2, padding 0 | $B \times 256 \times 8 \times 8 \times 8$ | $B \times 256 \times 16 \times 16 \times 16$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 256 \times 16 \times 16 \times 16$ | $B \times 128 \times 16 \times 16 \times 16$ |
| Upsample | kernel $3 \times 3$, stride 2, padding 0 | $B \times 128 \times 16 \times 16 \times 16$ | $B \times 128 \times 32 \times 32 \times 32$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 128 \times 32 \times 32 \times 32$ | $B \times 64 \times 32 \times 32 \times 32$ |
| Upsample | kernel $3 \times 3$, stride 2, padding 0 | $B \times 64 \times 32 \times 32 \times 32$ | $B \times 64 \times 64 \times 64 \times 64$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 64 \times 64 \times 64 \times 64$ | $B \times 32 \times 64 \times 64 \times 64$ |
| GroupNorm | num_groups=32 | $B \times 32 \times 64 \times 64 \times 64$ | $B \times 32 \times 64 \times 64 \times 64$ |
| Conv3D | kernel $1 \times 1$, stride 1, padding 0 | $B \times 32 \times 64 \times 64 \times 64$ | $B \times 1 \times 64 \times 64 \times 64$ |

Table 9. **Architecture for the ResNet2Voxel**.

| Layer name | Weights/Parameters | Input size | Output size |
|---|---|---|---|
| ResNet-18 [16] | - | $B \times 3 \times 256 \times 256$ | $B \times 512 \times 8 \times 8$ |
| Linear_to_3D | - | $B \times 512 \times 64$ | $B \times 512 \times 512$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 512 \times 8 \times 8 \times 8$ | $B \times 256 \times 8 \times 8 \times 8$ |
| Upsample | kernel $3 \times 3$, stride 2, padding 0 | $B \times 256 \times 8 \times 8 \times 8$ | $B \times 256 \times 16 \times 16 \times 16$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 256 \times 16 \times 16 \times 16$ | $B \times 128 \times 16 \times 16 \times 16$ |
| Upsample | kernel $3 \times 3$, stride 2, padding 0 | $B \times 128 \times 16 \times 16 \times 16$ | $B \times 128 \times 32 \times 32 \times 32$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 128 \times 32 \times 32 \times 32$ | $B \times 64 \times 32 \times 32 \times 32$ |
| 3D ResNet Block | kernel $3 \times 3$, stride 1, padding 1 | $B \times 32 \times 32 \times 32 \times 32$ | $B \times 32 \times 32 \times 32 \times 32$ |
| GroupNorm | num_groups=32 | $B \times 32 \times 32 \times 32 \times 32$ | $B \times 32 \times 32 \times 32 \times 32$ |
| Conv3D | kernel $1 \times 1$, stride 1, padding 0 | $B \times 32 \times 32 \times 32 \times 32$ | $B \times 1 \times 32 \times 32 \times 32$ |

paragraph.

- *ResNet2TSDF*. Given a pair of image and its corresponding 3D shape $(I, \mathbf{X})$, we use the similar architecture as the image inference module to extract the image features and lift them to 3D. However, we use extra UpConv3D layers to output $64^3$ grids instead, where each grid predicts the TSDF values. We train ResNet2TSDF with the L1 loss.
- *ResNet2Voxel*. We adopt the similar architecture of ResNet2TSDF, but the resulting outputs are the voxels prediction with resolution of $32^3$. ResNet2Voxel is trained with binary cross entropy loss where the targets are the ground truth voxels.

The architectures details for ResNet2TSDF and ResNet2Voxel are shown at Table 8 and Table 9.

**Details on Neural Evaluator** We leverage a Neural Listener architecture proposed by Achlioptas *et al*. [2] to perform discriminative quantitative comparisons. The Neural Listener uses 2D rendered images and Point-Clouds (PC) to represent the corresponding 3D shape in Shapenet [4]. Following the proposed implementation by Achlioptas *et al*. [2], we (1) finetune a VGG network for 8-way classification task of Shapenet objects: The 8 classes are [airplane, car, chair, lamp, rifle, sofa, table, watercraft] and gives $\sim 95\%$ accuracy on held out data; (2) train a self-supervised Point-Cloud AutoEncoder [1] with 2048 points under Chamfer loss [1] on chairs from Shapenet [4]; (3) train a 3 layer LSTM to process VGG-features and text description; (4) train a MLP which takes in the PC-AE latent representation (128D) and output of LSTM as input and performs classification. We mainly digress from the original implementation by performing a 2-way classification with one target and one distractor as opposed to the original setting of on target and two distactors. The Neural Listener has a classification accuracy of $\sim 83\%$ on held out data.

Figure 8 contains the visualizations indicating meaningful performance from neural evaluator.

**Joint Text-Shape Baseline** Our method proposes an approximate decomposition of joint probability of shapes and conditioning, into a product of an autoregressive prior over shapes and an independent conditional. In order to best highlight the efficacy of our proposed approach we create a baseline trained to directly learn this joint distribution. In further discussions, we call this baseline as JE. We imagine the task of text-to-shape generation as a Sequence to Sequence Translation task where a transformer Encoder (pre-trained using BERT [11]) is used to encode language. An autoregressive decoder is tasked with generating the latent representation which P-VQ-VAE can map back to 3D shapes. The Transformer Decoder uses the output of Encoder as *memory* while making generations. For simplicity we assume a fixed rasterized order of generation (as opposed to the random order with which our proposed method is trained) and only use the chair category from Shapenet [4] (i.e. the latent space from P-VQ-VAE was also only trained on Shapenet chairs). Figure 15 shows that these assumptions help in generating higher-fidelity shapes however even then the proposed method's outputs align better with text query.

## B. Autoregressive Results

**Reconstruction of P-VQ-VAE.** In Figure 9, we demonstrate that P-VQ-VAE can faithfully reconstruct the given input shape $\mathbf{X}$. Although P-VQ-VAE is trained on the ShapeNet, it can reconstruct the input shape from Pix3D as well.

**Random Generation of the Non-sequential Transformer.** We show the unconditional generation results with the transformer at Figure 10. The generation is achieved by autoregressive generation starting from an empty sequence. The diverse results across all categories show that the non-sequential autoregressive prior learns the representation of the generic shapes.

## C. Shape Completion Results

**More Comparison of Multimodal Shape Completion.** We show more comparisons of shape completion with baselines at Figure 11.

**More Results of Multimodal Shape Completion.** We show more results of shape completion at Figure 12.

## D. Single-view Results

We show more single-view reconstruction results in this section. We provide more comparisons with the competing method at Figure 13, and more results from the proposed method at Figure 12.

## E. Language-guided Generation Results

**Comparisons with Baseline** Figure 15 contains qualitative results for comparing our proposed approach with the two baselines (JE and Text2Shape (T2S) [5]). For every text description, we include three random generations for each of the methods. The visualized images for ours and JE are rendered from $64^3$ T-SDFs while for T2S they are rendered from $32^3$ voxels. We observe that results from our approach are more aligned to corresponding text. We also observe the lack of diversity in the generations from T2S. We suspect this could be because of Mode Collapse while fine-tuning the GAN in T2S [5] on data from ShapeGlot [2].

**Qualitative results** Figure 16 includes results of Text Conditioned Shape Generation on diverse text descriptions from held out data. For each text we include three random generations. The bottom row of Figure 16 includes some complex descriptions and highlight that generation quality is impacted with such complex and ambiguous descriptions.

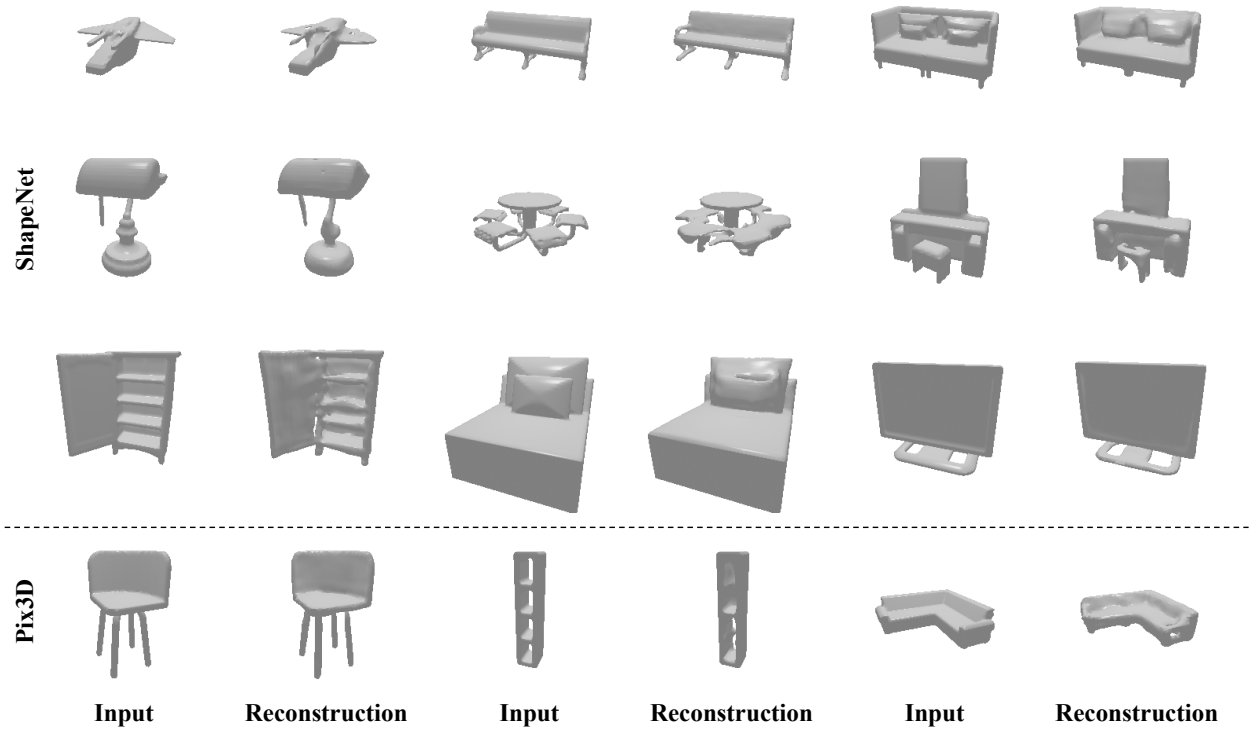
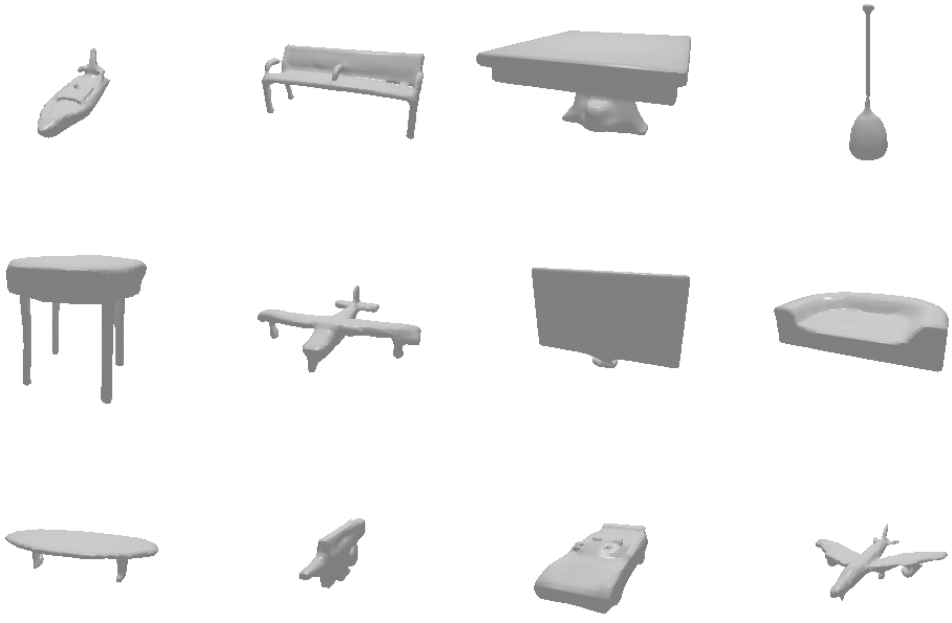



Figure 9. **Qualitative results for P-VQ-VAE's reconstruction.** We analyze the expressivity of our low-dimensional shape representation by visualizing the input shape (left) and the shape decoded from the encoded latent representation (right).



**Autoregressive Generation**

Figure 10. **Qualitative results of autoregressive generation from the transformer.** We validate that the transformer learns the representation of generic shapes. We achieve the generation by autoregressively predicting the next tokens starting from an empty sequence.

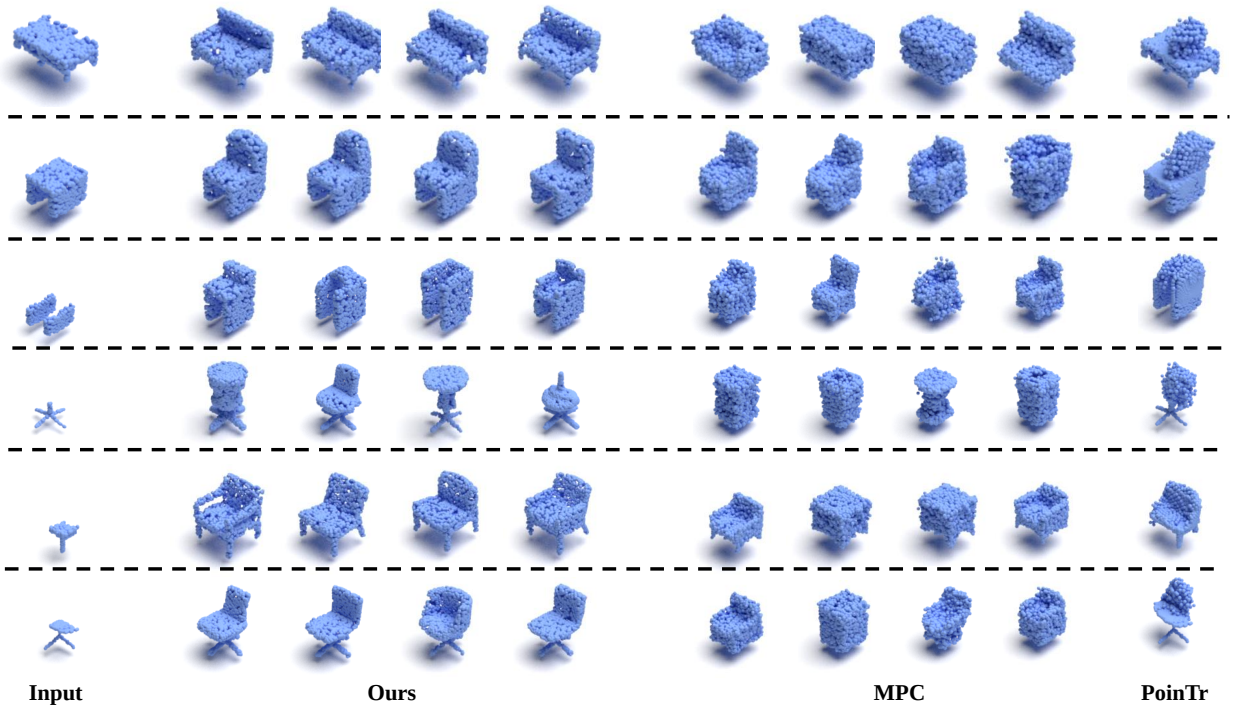



Figure 11. **More comparisons with competing methods for the shape completion.**

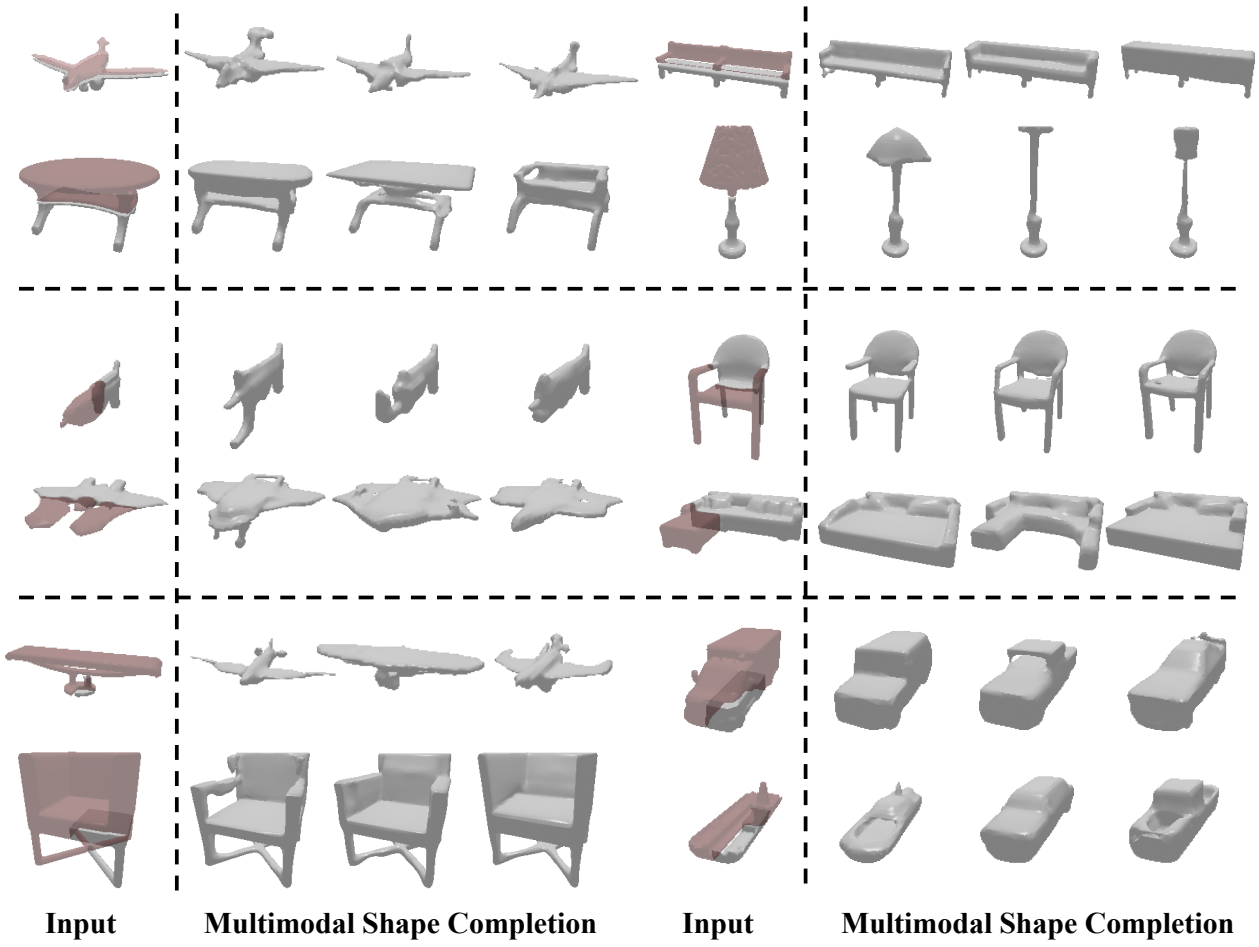



Figure 12. **More shape completions results with the proposed method.** Given the partial inputs, the proposed approach is able to generate diverse plausible 3D shapes consistent with the partial input. For example in row-5, a table like structure is reconstructed as an aeroplane. Red cuboid denotes the missing parts.

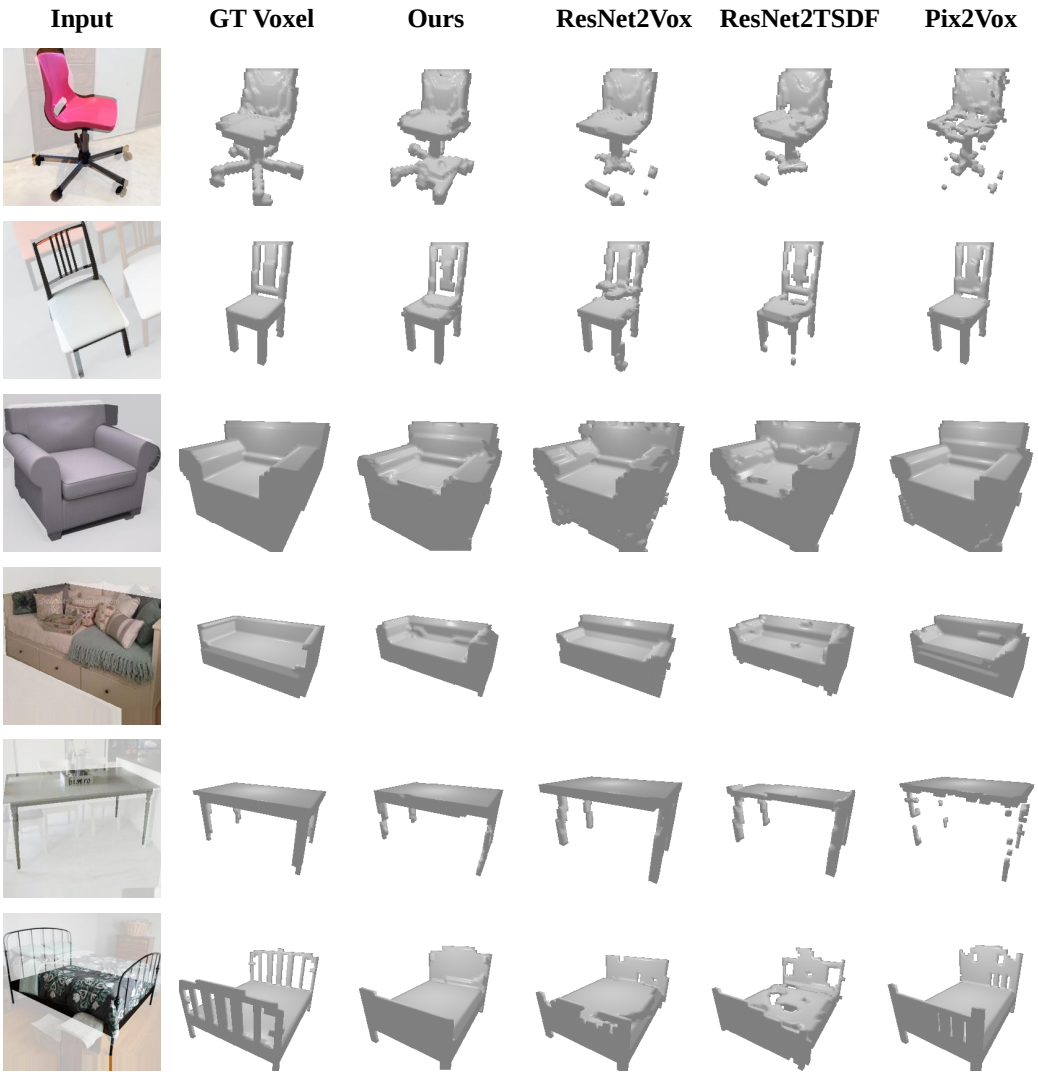



Figure 13. **More comparisons with the competing methods on single-view reconstruction.** Given an image as input, we show the single-view reconstruction results with the proposed method and how it compares against other competing methods.

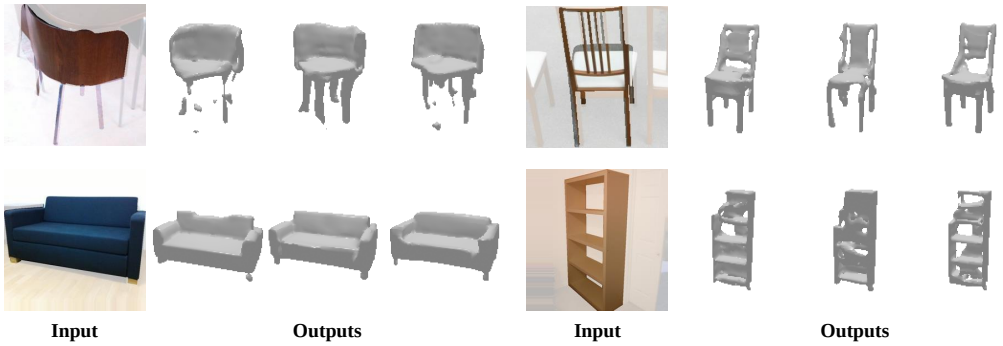



Figure 14. **More single-view reconstruction results from the proposed methods.** Given an image as input, we show the multimodal generation results with the proposed method.

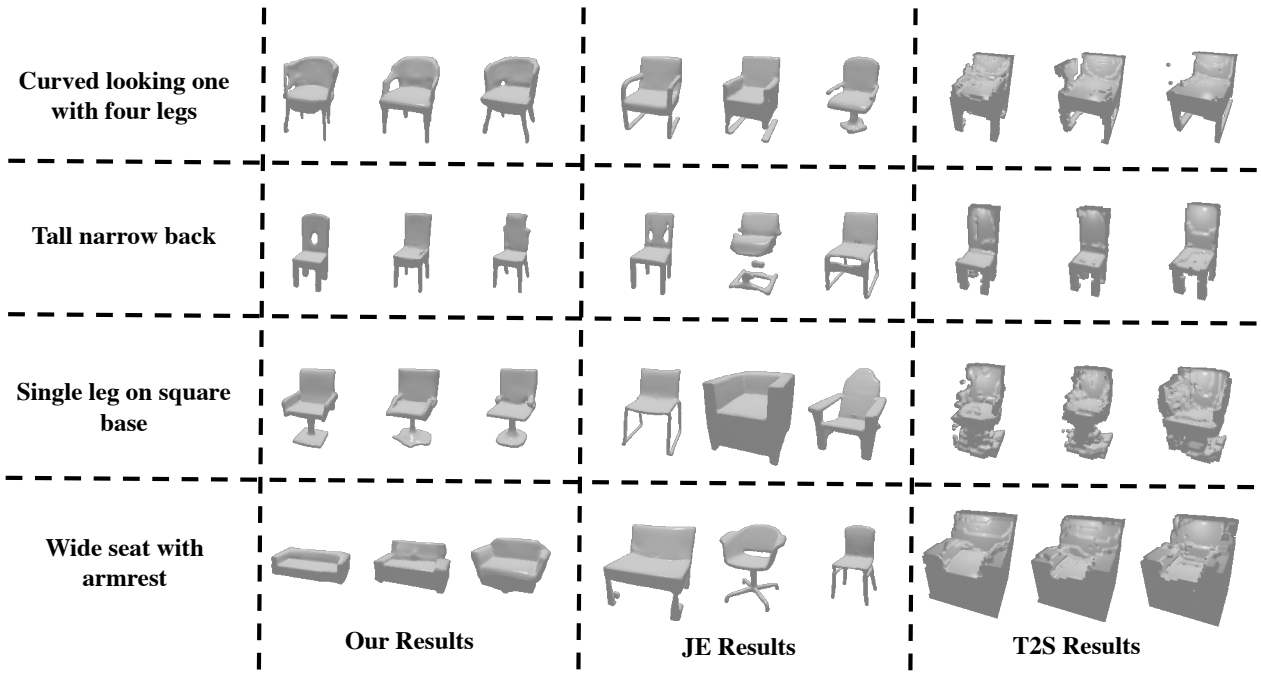



Figure 15. **Qualitative Comparison with baselines for language based generation.** While comparing our proposed method (Ours) with two baselines (JE and T2S) we report three random generations from each model for every text description. We observe that T2S shape generations are better aligned with text as compared to JE. Our proposed method is able to do both (a) align better with conditioning and (b) generate realistic 3D shapes.

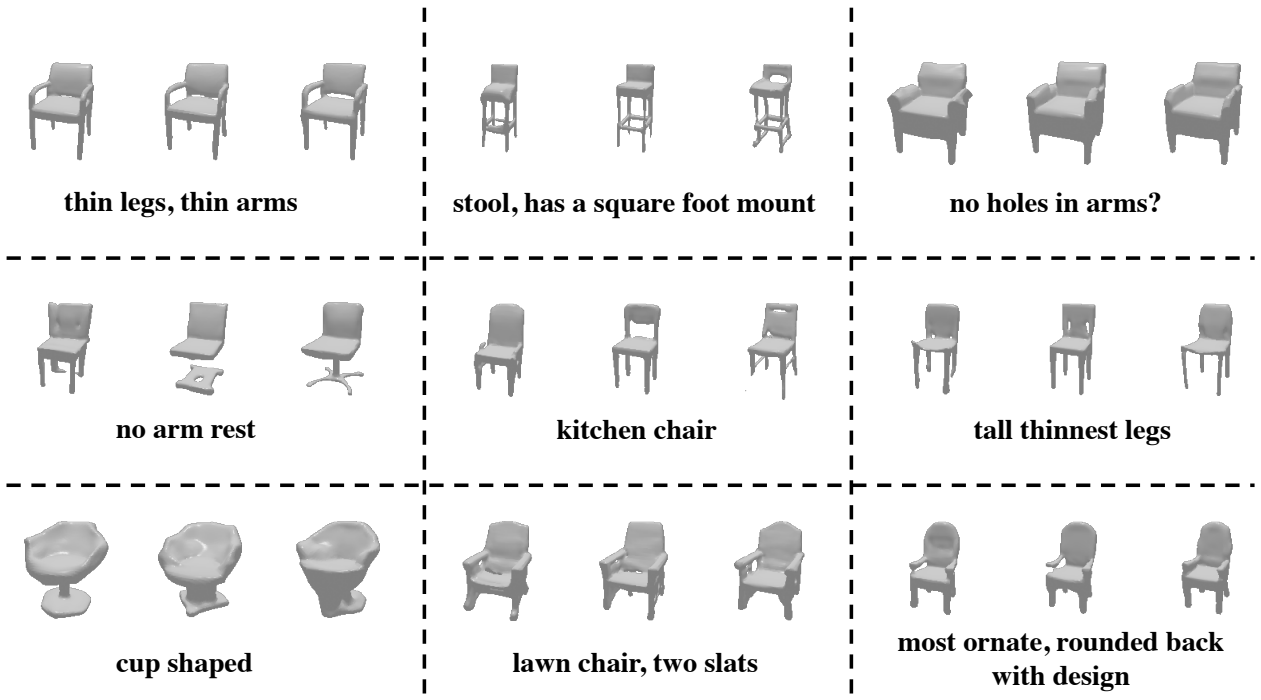



Figure 16. **Qualitative results of language based generations from our proposed approach** We report 3 random shapes generated for a given text description (in bold). We observe that while generated shapes are often meaningful, then can sometimes be random, example a flying chair in Row-2 Column-2. Bottom row includes some arguably rare text descriptions from held out data.