arXiv:2006.10705v1 [cs.LG] 18 Jun 2020

Set Distribution Networks: a Generative Model for Sets of Images

Shuangfei Zhai, Walter Talbott, Miguel Angel Bautista, Carlos Guestrin, Josh M. Susskind
Apple Inc.
{szhai,wtalbott,mbautistamartin,guestrin,jsusskind}@apple.com

Abstract

Images with shared characteristics naturally form sets. For example, in a face verification benchmark, images of the same identity form sets. For generative models, the standard way of dealing with sets is to represent each as a one hot vector, and learn a conditional generative model $p(\mathbf{x}|\mathbf{y})$. This representation assumes that the number of sets is limited and known, such that the distribution over sets reduces to a simple multinomial distribution. In contrast, we study a more generic problem where the number of sets is large and unknown. We introduce Set Distribution Networks (SDNs), a novel framework that learns to autoencode and freely generate sets. We achieve this by jointly learning a set encoder, set discriminator, set generator, and set prior. We show that SDNs are able to reconstruct image sets that preserve salient attributes of the inputs in our benchmark datasets, and are also able to generate novel objects/identities. We examine the sets generated by SDN with a pre-trained 3D reconstruction network and a face verification network, respectively, as a novel way to evaluate the quality of generated sets of images.

1 Introduction

Generative modeling of natural images has seen great advances in recent years. State of the art models such as GANs [3], EBMs [9] and VAEs [19] can generate single images with high perceptual quality. In many applications, however, images often come in sets with shared characteristics. For example, a set might be constructed from images that belong to the same semantic category, or those that share the same attribute. When the number of sets is limited and known, one is able to easily extend a generative model to its conditional version by representing the set information as a one hot vector [15, 17]. We refer to these models as class conditional generative models (CCGMs). CCGMs are fundamentally limited by the pre-defined enumeration of all possible sets in their encoding, which prevents recognizing or generating sets that are not specifically encoded from the training distribution. Also, the number of parameters needed for a CCGM grows linearly w.r.t the number of sets (classes) during training, which limits its scalability.

In this paper, we study the problem of generative modeling of sets of images with a generic approach. We propose Set Distribution Networks (SDNs), a probabilistic model that is capable of learning to stochastically reconstruct a given set and generate novel sets at the same time. Stochastic reconstruction of a set means that the individual images in the input set will not be reproduced exactly, but the generated images will share set-defining attributes with the input images. We achieve this by jointly training a set encoder, a set discriminator, a set generator, and a set prior. We train an SDN by following the MLE objective, which results in an adversarial game where the encoder, discriminator and prior are trained against the generator.

We evaluate SDNs on two benchmarks: 1. ShapeNet [5], which consists of objects in various viewpoints; 2. VGGFace2 [4], which consists of various faces of human identities. We show that

Preprint. Under review.

the same SDN architecture can be successfully trained on the two datasets, and can learn to both reconstruct an unseen set, and generate a novel set. We measure the quality of the set generative model by examining the generated samples with a pre-trained 3D reconstruction network, and face verification network, respectively, and show that SDNs learn to generate faithful and coherent sets.

2 Methodology

2.1 Set Distribution Networks

We denote an image set of size n as $\mathbf{X} \in \mathcal{X}$, with $\mathbf{X} = \{\mathbf{x}_i\}_{i=1...n}$, $\mathbf{x}_i \in R^d$, and we are interested in learning a probabilistic model $p_\theta(\mathbf{X})$. To do so, we propose a novel encoder-decoder styled model, dubbed Set Distribution Networks (SDNs), that takes the form:

$$p_\theta(\mathbf{X}) = p_\theta(z(\mathbf{X};\theta))p_\theta(\mathbf{X}|z(\mathbf{X};\theta)). \tag{1}$$

Here $z : \mathcal{X} \to \mathcal{Z}$ is a deterministic function that maps a set $\mathbf{X}$ to an element $\mathbf{z}$ in a discrete space $\mathcal{Z}$. $p_\theta(\cdot)$ is a prior distribution, with the support given by $supp(z) = \{z(\mathbf{X};\theta) : \mathbf{X} \in \mathcal{X}\}$ which is a subset of $\mathcal{Z}$. $p_\theta(\mathbf{X}|z(\mathbf{X}))$ is a conditional distribution which is defined as :

$$p_\theta(\mathbf{X}|\mathbf{z}) = \frac{I(z(\mathbf{X}) = \mathbf{z})e^{-\sum_{\mathbf{x}\in\mathbf{X}} E(\mathbf{x},\mathbf{z};\theta)}}{\int_{\mathbf{X}'\in\mathcal{X}} I(z(\mathbf{X}') = \mathbf{z})e^{-\sum_{\mathbf{x}\in\mathbf{X}'} E(\mathbf{x},\mathbf{z};\theta)} d\mathbf{X}'}. \tag{2}$$

Here $I(\cdot)$ is the indicator function. The conditional probability takes the form of an energy based model (EBM), where density is only assigned to sets $\mathbf{X}'$ that are mapped to the same $\mathbf{z}$.

We first show that Equation 1 indeed defines a valid distribution of $\mathbf{X}$. To see this, we have:

$$\begin{aligned}\int_{\mathbf{X}\in\mathcal{X}} p_\theta(\mathbf{X})d\mathbf{X} &= \int_{\mathbf{X}\in\mathcal{X}} p_\theta(z(\mathbf{X};\theta))p_\theta(\mathbf{X}|z(\mathbf{X}))d\mathbf{X} = \sum_{\mathbf{z}\sim supp(z)} \int_{\mathbf{X}\in\mathcal{X}_\mathbf{z}} p_\theta(\mathbf{z})p_\theta(\mathbf{X}|\mathbf{z})d\mathbf{X} \\ &= \sum_{\mathbf{z}\in supp(z)} p_\theta(\mathbf{z}) \int_{\mathbf{X}\in\mathcal{X}_\mathbf{z}} p_\theta(\mathbf{X}|\mathbf{z})d\mathbf{X} = \sum_{\mathbf{z}\in supp(z)} p_\theta(\mathbf{z}) = 1,\end{aligned} \tag{3}$$

where $\mathcal{X}_\mathbf{z} \triangleq \{\mathbf{X} : z(\mathbf{X};\theta) = \mathbf{z}, \mathbf{X} \in \mathcal{X}\}$. Here we first partition the integration by grouping $\mathbf{X}$ that share the same $\mathbf{z}$, then move $p_\theta(\mathbf{z})$ out of the integral as it's a constant within the partition, and lastly recognize that the integration evaluates to 1 according to the definition of Equation 2.

Intuitively, the SDN is an encoder-decoder model with discrete latent variables, with $z(\cdot;\theta)$ being the encoder and $p_\theta(\cdot|\cdot)$ being the probabilistic decoder. The encoder serves the role of partitioning the input space, while the decoder defines a normalized distribution over sets mapped to the same partition.

2.2 Approximate Inference with Learned Prior and Generator

We apply MLE to estimate the parameters of $p_\theta(\mathbf{X})$, where the negative log likelihood loss for an observed set $\mathbf{X}$ in the training split is:

$$-\log p_\theta(z(\mathbf{X};\theta)) - \log p_\theta(\mathbf{X}|z(\mathbf{X};\theta)), \tag{4}$$

which is decomposed into two parts, corresponding to the prior and decoder distribution respectively. We adopt a simple parameterization of the prior distribution as: $p_\theta(\mathbf{z}) = \frac{\bar{p}_\theta(\mathbf{z})}{\sum_{\mathbf{z}\in supp(z),} \bar{p}_\theta(\mathbf{z})}, \forall \mathbf{z} \in supp(z)$; $p_\theta(\mathbf{z}) = 0$ otherwise. Here $\bar{p}_\theta(\mathbf{z})$ is a normalized distribution over $\mathcal{Z}$. Because $supp(z)$ is always a subset of $\mathcal{Z}$, we have that $p_\theta(\mathbf{z}) \geq \bar{p}_\theta(\mathbf{z}), \forall \mathbf{z} \in supp(z)$. We then achieve an upper bound of -$\log p_\theta(z(\mathbf{X};\theta))$ given by

$$\mathcal{L}_0(\theta) \triangleq -\log \bar{p}_\theta(z(\mathbf{X};\theta)), \tag{5}$$

which gets rid of the need for the intractable $supp(z)$. For -$\log p_\theta(\mathbf{X}|z(\mathbf{X};\theta))$, we apply variational inference with a learned generator, as in [6, 22, 24, 25], where we have:

$$
\begin{aligned}
&-\log p_\theta(\mathbf{X}|z(\mathbf{X};\theta)) = -\log \frac{I(z(\mathbf{X};\theta)=\mathbf{z})e^{-\sum_{\mathbf{x}\in\mathbf{X}} E(\mathbf{x},\mathbf{z};\theta)}}{\int_{\mathbf{X}'\in\mathcal{X}} I(z(\mathbf{X}';\theta)=\mathbf{z})e^{-\sum_{\mathbf{x}\in\mathbf{X}'} E(\mathbf{x},\mathbf{z};\theta)}d\mathbf{X}'}\Bigg|_{\mathbf{z}=z(\mathbf{X};\theta)} \\
&= \sum_{\mathbf{x}\in\mathbf{X}} E(\mathbf{x},\mathbf{z};\theta) + \log \int_{\mathbf{X}'\in\mathcal{X}} I(z(\mathbf{X}';\theta)=\mathbf{z})e^{-\sum_{\mathbf{x}\in\mathbf{X}'} E(\mathbf{x},\mathbf{z};\theta)}d\mathbf{X}'\Bigg|_{\mathbf{z}=z(\mathbf{X};\theta)} \\
&\geq \sum_{\mathbf{x}\in\mathbf{X}} E(\mathbf{x},\mathbf{z};\theta) + \mathrm{E}_{\mathbf{X}'\sim p_\psi(\mathbf{X}|\mathbf{z})} \log[I(z(\mathbf{X}';\theta)=\mathbf{z})e^{-\sum_{\mathbf{x}\in\mathbf{X}'} E(\mathbf{x},\mathbf{z};\theta)}] + H(p_\psi)\Bigg|_{\mathbf{z}=z(\mathbf{X};\theta)} \\
&\triangleq \mathcal{L}_1(\theta,\psi)
\end{aligned}
\tag{6}
$$

Here we have derived a lower bound of -$\log p_\theta(\mathbf{X}|\mathbf{z})$[1] by introducing a variational distribution $p_\psi(\mathbf{X}|\mathbf{z})$, which we parameterize in the form of a generator:

$$
\int_{\mathbf{X}\sim p_\psi(\mathbf{X}|\mathbf{z})} f(\mathbf{X})d\mathbf{X} \triangleq \int_{\{\mathbf{z'}_i\sim p_\psi(\mathbf{z}')\}_{i=1\ldots n}} f(\{G(\mathbf{z},\mathbf{z}'_\mathbf{i};\psi)\}_{i=1\ldots n})d\{\mathbf{z'}_i\}_{i=1\ldots n}, \forall f. \tag{7}
$$

Here $p_\psi(\mathbf{z}')$ is a simple distribution (e.g., Isotropic Gaussian) over $\mathcal{Z}'$, and $G : \mathcal{Z}\times\mathcal{Z}' \to R^d$ is a deterministic function (the generator) that maps a $(\mathbf{z},\mathbf{z}')$ tuple to the image space. The lower bound $\mathcal{L}_1(\theta,\psi)$ can be tightened by solving $\max_\psi \mathcal{L}_1(\theta,\psi)$ [22].

2.3 Model Architectures

There are four interacting modules for SDNs: the prior $p_\theta(\mathbf{z})$, the encoder $z(\mathbf{X};\theta)$, the discriminator (energy) $E(\mathbf{x},\mathbf{z};\theta)$ and the generator $G(\mathbf{z},\mathbf{z}';\psi)$. We now explain the architecture and design choice for each of them. See Figure 1 for an illustration.

Prior. In all of our implementations, we adopt binary set codes, i.e., letting $\mathcal{Z} = \{-1,1\}^{d_z}$. In theory, one can choose any prior distribution over discrete variables. We use a standard auto-regressive model MADE [10] with three fully connected layers, mainly for its simplicity and robustness.

Encoder. As a necessary condition, an encoder for a set needs to satisfy the permutation invariant property [21]. We opt to use a simple architecture design where we let $z(\mathbf{X};\theta) = binarize(\frac{1}{n}\sum_{i=1}^n c(\mathbf{x}_i;\theta))$, where $c(\cdot;\theta)$ is a standard CNN image encoder. The outputs of the image codes are averaged across the set and then passed to a differentiable binarization operator (with straight through gradient estimation) to produce the final set code.

Discriminator. The discriminator's job is to assign low energy to observed images and high energy to generated images, given a set code $\mathbf{z}$. We use an autoencoder based energy function implementation, similar to [25]. We have found that this choice is important as it enables effective learning in early stages of training. We reuse $c(\cdot;\theta)$ as the encoder, and separately learn a decoder $d(\mathbf{z},c(\mathbf{x}))$, which takes the concatenation of the binary set code $\mathbf{z}$ and dense image code $c(\mathbf{x})$ as input, and outputs a "reconstruction" of $\mathbf{x}$. We additionally learn a unary energy term of $\mathbf{x}$ with a small MLP $d_0 : R^{d_z} \to R$ that takes $c(\mathbf{x})$ as input and outputs a scalar. This gives us the final output of the discriminator as:

$$
E(\mathbf{x},\mathbf{z};\theta) = \|\mathbf{x} - d(\mathbf{z},c(\mathbf{x};\theta);\theta)\|_2^2 + d_0(c(\mathbf{x};\theta);\theta). \tag{8}
$$

Generator. The generator generates a set conditioned on a set code $\mathbf{z}$ by sampling n random variables $\{\mathbf{z'}_i \sim p_\psi(\mathbf{z}')\}_{i=1\ldots n}$, each of which is concatenated with $\mathbf{z}$ and generates an image independently.

2.4 Losses

SDNs consist of two sets of parameters to be optimized, θ and ψ, where θ denotes the combined parameters for the prior, encoder and discriminator; and ψ denotes those for the generator. During training, with θ fixed, we first optimize ψ to tighten the lower bound by solving $\max_\psi \mathcal{L}_1(\theta,\psi)$. In order to make this practical, we make two simplifications. First, we do not explicitly include the

[1] we use $\mathbf{z}$ in place of $z(\mathbf{X};\theta)$ when there is no ambiguity

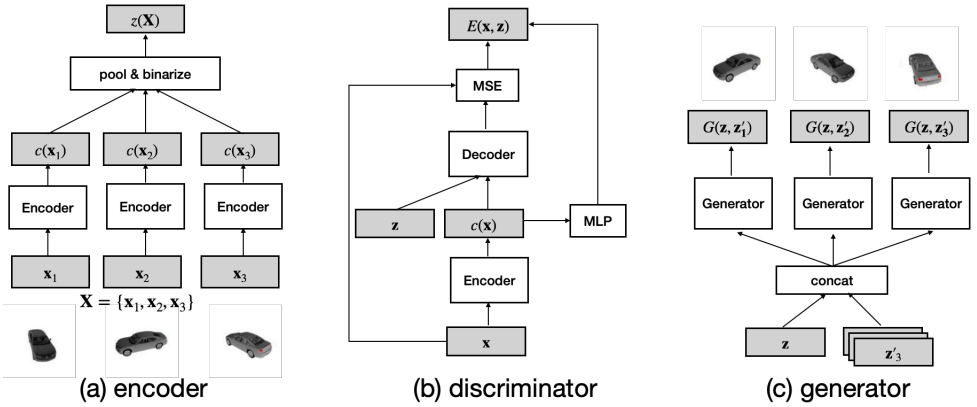


Figure 1: Model architectures of SDNs. SDNs consist of three modules: (a) a set encoder that maps a set of images into a discrete code, with a shared convolutional encoder encoding each image followed by average pooling and discretization; (b) a conditional discriminator (energy) for each image that takes the form of an autoencoder (similar to [25]); (c) a conditional generator that generates images conditioned on a set code $\mathbf{z}$.

entropy term $H(p_\psi)$. This seems problematic at first glance as $H(p_\psi)$ plays the role of encouraging $p_\psi(\mathbf{X}|\mathbf{z})$ to have high entropy which prevents the mode collapse problem. However, we have found that we can implicitly achieve high diversity of the generated samples with careful learning scheduling and architectural design choices as commonly used in GAN training, see Sec. 4.2 for more details. Also refer to [22] for more justification of this choice. Second, the indicator function $I(z(\mathbf{X}') = \mathbf{z})$ which ensures that the generated sets are mapped to the same set code $\mathbf{z}$ is not differentiable. We thus choose to use a soft approximation: $I(z(\mathbf{X}') = \mathbf{z}) \approx e^{-\|[z(\mathbf{X}';\theta)\odot\mathbf{z}]_-\|_1}$, where $\odot$ is the element-wise product, $[\cdot]_-$ is the operator that zeros out the positive elements, and $\|\cdot\|_1$ is the ℓ_1 norm. Intuitively, this approximation equates the indicator function when $sign(z(\mathbf{X}';\theta)) = \mathbf{z}$, and induces a value in $(0, 1)$ otherwise. This leads us to the final form of the loss for the generator:

$$\begin{aligned}\mathcal{L}_\psi &= -\mathrm{E}_{\{\mathbf{z'}_i\sim p_\psi(\mathbf{z'})\}_{i=1\ldots n}} \log[e^{-\|[z(\mathbf{X}';\theta)\odot\mathbf{z}]_-\|_1} e^{-\sum_{\mathbf{x}\in\mathbf{X}'} E(\mathbf{x},\mathbf{z};\theta)}]|_{\mathbf{X}'=\{G(\mathbf{z},\mathbf{z'}_i;\psi)\}_{i=1\ldots n}} \\ &= \mathrm{E}_{\{\mathbf{z'}_i\sim p_\psi(\mathbf{z'})\}_{i=1\ldots n}}[\|[z(\mathbf{X}';\theta)\odot\mathbf{z}]_-\|_1 + \sum_{\mathbf{x}\in\mathbf{X}'} E(\mathbf{x},\mathbf{z};\theta)]|_{\mathbf{X}'=\{G(\mathbf{z},\mathbf{z'}_i;\psi)\}_{i=1\ldots n}}.\end{aligned} \tag{9}$$

With ψ fixed, we can optimize θ with a loss that combines $\mathcal{L}_0(\theta)$ and $\mathcal{L}_1(\theta,\psi)$. As is common practice in GAN and deep EBM training [3, 9, 25], we apply a margin based loss on $E(\mathbf{x},\mathbf{z};\theta)$. In particular, we have found that it's beneficial to apply two separate margins on the two terms of E, this leads to the loss function of θ update as:

$$\begin{aligned}\mathcal{L}_\theta = &-\log\bar{p}_\theta(\Omega(z(\mathbf{X};\theta))) \\ &+\sum_{\mathbf{x}\in\mathbf{X}}\|\mathbf{x}-d(z(\mathbf{X};\theta),c(\mathbf{x};\theta);\theta)\|_2^2 + [\gamma_0 + d_0(\mathbf{x};\theta)]_+ \\ &+\mathrm{E}_{\mathbf{X}'\sim p_\psi(\mathbf{X}|z(\mathbf{X}))}\sum_{\mathbf{x}\in\mathbf{X}'}[\gamma_1 - \|\mathbf{x}-d(z(\mathbf{X};\theta),c(\mathbf{x};\theta);\theta)\|_2^2]_+ + [\gamma_0 - d_0(\mathbf{x};\theta)]_+.\end{aligned} \tag{10}$$

Here Ω is the stop gradient operator, indicating that we do not pass gradient from the prior to the encoder. $[\cdot]_+$ is the operator that zeros out the negative portions of the input. $\gamma_0, \gamma_1 \in R^+$are two margin parameters.

3 Related Work

Generative Modeling of sets. Generative modeling of sets is a challenging task. Hierarchical Bayes models have previously been widely applied to document modeling, which can be considered as a set of words, represented by the LDA model [2]. Recently, there has been a a body of work dedicated to point cloud generative modeling, where sets are composed of 3D points in Euclidean space [1, 13, 20]. However, image sets considered in our work contain much richer structure than 3D points. It is thus

unclear how methods developed from the point cloud or document modeling lines of work generalize to more complex image data.

Autoencoding GANs. SDNs are similar to a handful of existing works which involve an encoder, generator/decoder and a discriminator [7, 8, 12]. Notably, BiGAN [7] trains a two channel discriminator that tries to separate the two joint distributions $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ and $p(\mathbf{x})p(\mathbf{z}|\mathbf{x})$. Aside from the set modeling aspect, in a BiGAN the encoder and decoder work jointly against the discriminator, whereas in SDNs, the encoder and the discriminator work jointly against the generator. Intuitively, SDNs encourage the encoder to learn discriminative representations by not allowing it to cooperate with the generator, which in turn improves the generator's quality as well.

Conditional GANs. A conditional GAN (cGAN) [15, 17] modifies the generator and discriminator of a GAN to be conditional on class labels. SDNs can be considered a generalization of cGANs, as it learns the parametric set representation jointly with the generator and discriminator.

VQ-VAEs. VQ-VAEs [18, 19] are related to SDNs in that they learn encoders that output discrete and deterministic codes. They also learn auto-regressive priors which greatly improves their capability of generative modeling. SDNs differ from VQ-VAEs in that they have a more sophisticated encoder and discriminator, which are essential to cope with set structures.

4 Experiments

4.1 Datasets

The first dataset we use is ShapeNet [5], which consists of 3D objects with projected 2d views. We use a subset which contains 13 popular categories, namely *airplane, bench, cabinet, car, chair, monitor, lamp, speaker, gun, couch, table, phone, ship*. The training set consists of 32,837 unique objects and 788,088 images in total. The second dataset is VGGFace2 [4], which is a face dataset containing 9,131 subjects and 3.31M images.

For both datasets, we randomly construct fixed-size sets of 8 elements for training. For ShapeNet, this is done by randomly selecting an object and 8 of its views; for VGGFace2, we similarly select one identity and 8 of her/his faces. A mini-batch is constructed by selecting N objects/subjects, which amounts to $N \times 8$ images in total. All images are scaled to a size of 128×128.

4.2 Implementation Details

Our implementation resembles that of SAGAN [23] w.r.t. base architecture and learning scheduling. The encoder is identical to the discriminator in SAGAN, namely a CNN enhanced with Spectral Normalization (SN) [16] and Self Attention (SA), except that the number of output units is changed to d_z instead of 1. The dimension of $\mathbf{z}$ is 2048 and 256 for ShapeNet and VGGFace2, respectively. The generator is also the same as that in SAGAN, which is a CNN with SN, Batch Norm [11] and SA. The decoder for the discriminator resembles the generator, but without BN and SA; the MLP for the unary energy term is a simple two layer ReLU neural network with d_z hidden units. All images are normalized to range $[-1, 1]$. We set the two margin parameters as $\gamma_0 = 1, \gamma_1 = 0.1$, which we have found to work robustly across various settings.

During training, we use Adam with a learning rate 1e-4 for the generator , and learning rate 4e-4 for the encoder, discriminator, and prior. The momentum terms are set as $\{0, 0.999\}$ for both optimizers. We use a batch size of 32 (consisting of 256 images), spread across 4 V100 GPUs. We alternatively update ψ and θ one step per mini-batch, and train the models for approximately 600K iterations, which takes roughly two weeks.

4.3 Qualitative Results

By construction, SDNs are able to reconstruct (stochastically) an input set, and generate new sets by sampling from the prior. We show example inputs, reconstructions and samples in Fig. 2, 4, 6, 7. We see that SDNs produce semantically meaningful, consistent reconstructions of unseen input sets. The samples from the prior are also coherent. For faces, the SDN captures variations in hairstyle, expression, and pose, while retaining gender and ethnicity of the input identities. For ShapeNet, we

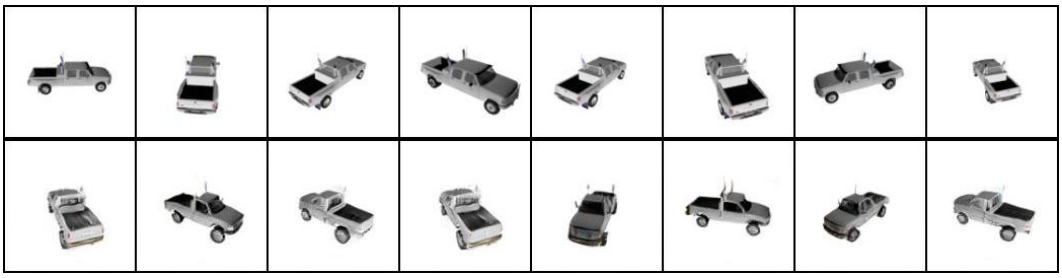


Figure 2: Top: example set from the test split of ShapeNet; bottom: reconstructed set. There is no correspondence between images from the two sets.

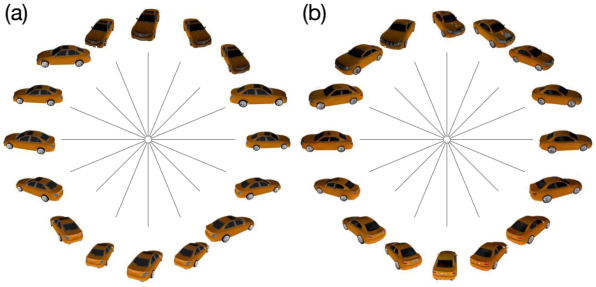


Figure 3: (a) Input, and (b) reconstructed sets from a ShapeNet car where samples are arranged on a circle by manually estimated pose (see Supp. Material for details.) This visualization shows consistency in terms of appearance and pose variability between input and reconstructed sets.

show in Fig. 3 that the reconstructed sets match their input counterparts in terms of object consistency and pose diversity. See Supp. Materials for more qualitative results.

4.4 ShapeNet Evaluations

In order to show that ShapeNet sets reconstructed by SDNs are consistent quantitatively, we take a pre-trained Occupancy Network [14] (OccNet) and use it to generate 3D meshes from images of both real and SDN-reconstructed sets, where SDNs and OccNets are trained on the same train/test splits and rendered views. We extend OccNets [14] to deal with multiple views by a simple average pooling of its encoder output across views, and refer readers to [14] and the Supp. Material for more details.

We empirically evaluate SDNs by computing Chamfer Distance[2] (CD) [1] and IoU as in [14] between corresponding 3D meshes obtained from real and SDN-reconstructed sets. We denote these meshes as $\mathcal{M}$ and $\hat{\mathcal{M}}$, respectively. We compute the CD on 2048 points sampled from the surface of each mesh. To approximate the IoU we sample 2048 points on the volume of each mesh and compute the average proportion of points of $\mathcal{M}$ contained in the volume of $\hat{\mathcal{M}}$ and vice-versa. We expect that if SDN-reconstructed sets represent accurate and consistent views of the same object, both metrics will improve as the difference between reconstructed and real sets decreases.

[2]We report CD$\times 10^3$

Figure 4: Two sets sampled from the prior on ShapeNet.

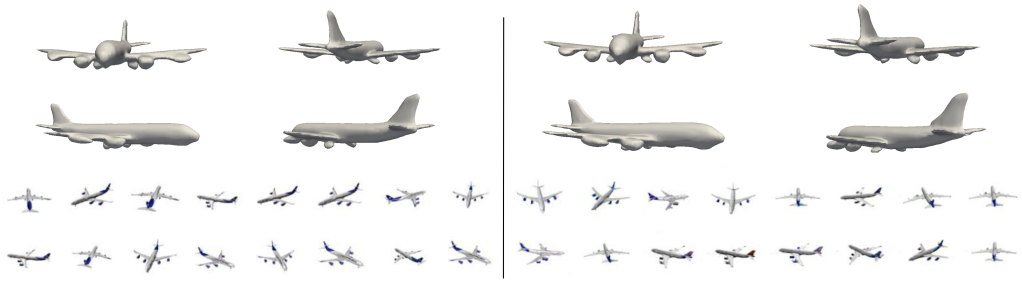


Figure 5: Left: real set and 3D reconstruction. Right: reconstructed set and 3D reconstruction. For this example the IoU and CD are 0.784 and 4.289, respectively.

Figure 6: Top: an example set from the test split of VggFace2; bottom: the reconstructed set. There is no correspondence between images from the two sets.

We verify this hypothesis with two experiments, both conducted on the test split of ShapeNet. In the first experiment we fix the size of real sets to 16 views and vary the number of views sampled from the reconstructed set[3]. We then use the sets to compute meshes $\mathcal{M}$ and $\hat{\mathcal{M}}$. We see that as the number of sampled reconstructed views increases, we consistently get better reconstructions, as shown in Tab. 1 (a).

# Recon. views	1	4	8	16
CD ↓	4.458	3.682	3.501	3.433
IoU ↑	0.679	0.711	0.719	0.722

(a)

# Input views	1	4	8	16
CD ↓	6.333	3.912	3.590	3.434
IoU ↑	0.633	0.701	0.715	0.722

(b)

Table 1: (a) 3D reconstruction results as a function of the reconstructed set size. (b) 3D reconstruction results as a function of the number of samples n used in the compute the set code $z(\mathbf{X})$.

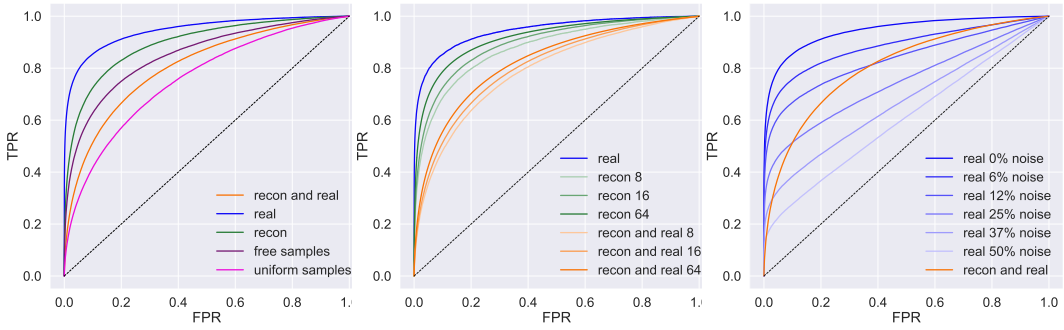
For our second experiment we vary the number of input views used to compute the set code $z(\mathbf{X})$, while keeping the number of reconstructed views constant (16). We compute the reconstructed meshes and compare them with the corresponding mesh obtained with all 16 input views. Our hypothesis is that the set representation gets better as the number of samples in $\mathbf{X}$ increases, and therefore sets can be more accurately reconstructed, which will lead to better 3D reconstructions, as shown in Tab. 1(b). Finally, in Fig. 5 we show qualitative comparisons of 3D meshes obtained from real vs. reconstructed sets, where both contain 16 elements.

4.5 VGGFace2 Evaluations

To provide quantitative results of set reconstruction performance for faces, we examine identity verification ROC curves on the VGGFace2 dataset using a pre-trained face verification model from [4]. We evaluate both reconstructed samples and samples generated from the prior, corresponding to Fig. 6 and Fig. 7, respectively. All input sets are taken from the test split of VGGFace2.

[3]Note that the model is trained with sets of 8 but naturally generalizes to different set sizes at inference time, due to the use of average pooling in the encoder.

Figure 7: Two sets sampled from the prior on VGGFace2.



(a) Image type (b) Context size (c) Label noise

Figure 8: Identity verification performance with a pre-trained VGGFace2 model. 'recon and real': matching reconstructed images to real images; 'real': matching real images; 'recon': matching reconstructed images; 'free samples': matching samples generated from the learned prior. 'uniform samples': matching samples generated from a uniform prior.

As a preprocessing step, we convert each image (real and generated) into a fixed dimensional embedding with the VGGFace2 model. We then use the distances in the embedding space to compare same and different identities' true-positive rate (TPR) and false-positive rate (FPR). We plot the ROC curve for different types of input images in Figure 8a. We see that the curve for reconstructed images ('recon') is close to that of real images ('real'), suggesting that the reconstructed sets are diverse and also that the images within a set are consistent with each other. Comparing the reconstruction to real images ('recon and real'), we see that the reconstructions are diverse and self-consistent, although the SDN does not strictly preserve identity in the reconstructed sets. As a better calibration of this difference, we also show the ROC for real images with different proportions of label noise in Figure 8c. Here we randomly contaminate the sets of real images with increasing numbers of images from different identities. We can see that the reconstruction performance at different FPR thresholds approximates different levels of label noise. For example, at a 10% FPR, we see that the reconstruction sets perform similarly to real identities with 25% label noise.

Additionally, we look at the effect of context size on the reconstructed images in Figure 8b. Even though the SDN was trained with a constant context size of 8, increasing the size of the input set improves the performance, and here we see reconstructed images approach even closer the performance of the model on the real images, which is also consistent with Table 1.

5 Conclusion

In this paper, we presented SDNs, a novel probabilistic generative model for image sets. We demonstrated that SDNs can be successfully trained on two real world datasets, ShapeNet and VGGFace2, at 128×128 resolution. We proposed to evaluate trained SDNs with a pretrained 3D reconstruction network and a face verification network, respectively, and showed that the trained SDNs generate high quality image sets both qualitatively and quantitatively.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[4] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 67–74. IEEE, 2018.

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

[6] Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating energy-based generative adversarial networks. *arXiv preprint arXiv:1702.01691*, 2017.

[7] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[8] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *Advances in Neural Information Processing Systems*, pages 10541–10551, 2019.

[9] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

[10] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889, 2015.

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[12] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[13] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point cloud gan. *arXiv preprint arXiv:1810.05795*, 2018.

[14] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[15] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[16] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[17] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018.

[18] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017.

[19] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*, 2019.

[20] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4541–4550, 2019.

[21] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

[22] Shuangfei Zhai, Walter Talbott, Carlos Guestrin, and Joshua Susskind. Adversarial fisher vectors for unsupervised representation learning. In *Advances in Neural Information Processing Systems*, pages 11156–11166, 2019.

[23] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[24] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[25] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

Appendix

A More samples

We first show more samples obtained with SDNs. Reconstructions of unseen sets can be found in Fig. 9 and Fig. 10 for ShapeNet and VGGFace2, respectively. We also demonstrate that the learned prior produces good and consistent samples in Figure 11 and Figure 12, where the uniform prior produces significantly worse samples. All the samples shown are uncurated.

B Varying input set size

SDNs are trained with fixed set sizes (8 in our experiments). But because of the use of average pooling in the encoder, it is possible to test a trained SDN with varied input sizes (Table 1(b) in main text). We show four such results in Fig. 13. we see that an SDN is able to effectively utilize more input samples within the set and produce inreasingly more consistent reconstructions.

C 3D reconstruction on ShapeNet

Our 3D reconstruction experiments show that SDN-reconstructed sets are accurate and consistent. In this section, we describe in more detail our extension of Occupancy Networks [14] to deal with multiple input RGB views, which is not discussed in [14]. Occupancy Networks use a convolutional encoder to compute a d-dimensional latent representation $\mathbf{c} \in \mathbb{R}^d$ for a given RGB view. This latent representation is then used as conditioning for an MLP $f_\theta : \mathbb{R}^3 \times \mathbb{R}^d \rightarrow \mathbb{R}$ that takes 3D points and predicts their occupancy (ie. whether the points lie inside or outside the object mesh).

Our hypothesis is that by average pooling the latent representations $\mathbf{c}$ obtained by each element on a set (remember each element on a set corresponds to a different viewpoint of the same object) we can increase the amount of information about the object contained in $\mathbf{c}$. However, that is only true if the latent representations $\mathbf{c}$ of different views are in agreement, in other words, if they are different views of the *same* object. To check this assumption we take a pre-trained Occupancy Network trained on single views (check [14] for more details) and show in Tab. 2 that reconstruction accuracy as measured by IoU increases when the latent representations $\mathbf{c}$ are pooled across views of the *same* object[4]. These results show that if elements of a SDN-reconstructed set are in agreement (eg. if a set contains different viewpoints of the same underlying object) reconstruction accuracy should improve when average pooling across elements in the set.

	↑**IoU**	
	1 view	5 views
Mean	0.593	0.621

Table 2: Aggregating multiple views for Occupancy Networks improves reconstruction accuracy.

In Fig. 14 we show uncurated pairs of real and SDN-reconstructed sets together with their respective 3D reconstructions obtained by running our Occupancy Networks extension. For each grid the two first rows correspond to real set and an orbit of its predicted 3D reconstruction and the two last rows correspond to SDN-reconstructed set and its corresponding 3D reconstruction.

Finally, Fig. 15 shows sets sampled from the SDN prior and their corresponding 3D reconstruction. For each grid in the top row we show the sampled sets of 8 elements from the prior, and the bottom two rows show and orbit of the 3D mesh.

[4]This evaluation was done on real sets of object views and following the evaluation protocol of [14].

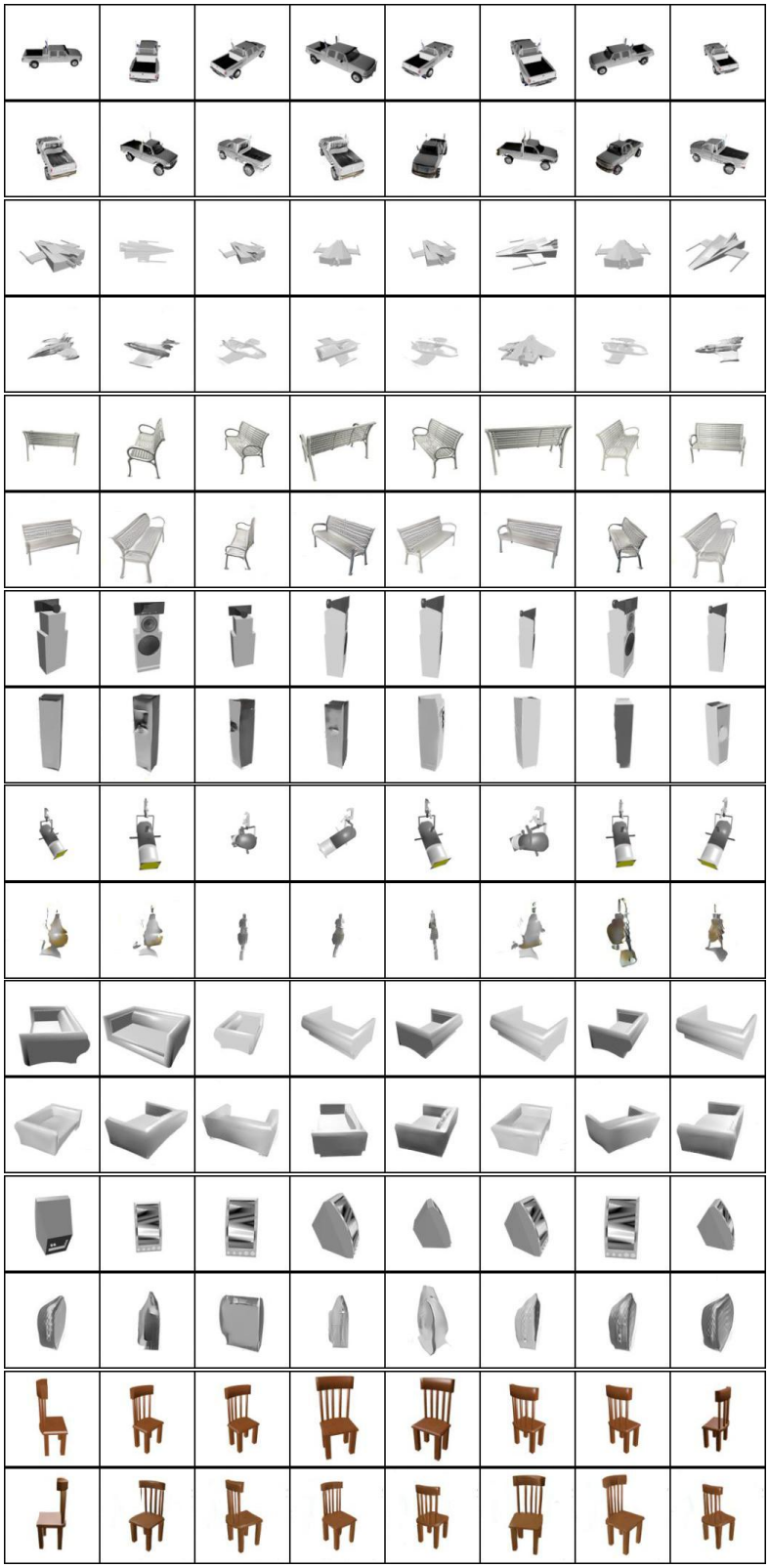


Figure 9: More reconstructions of objects from the test split ShapeNet. For each block the top row is the input set and the bottom row is the reconstructed set.

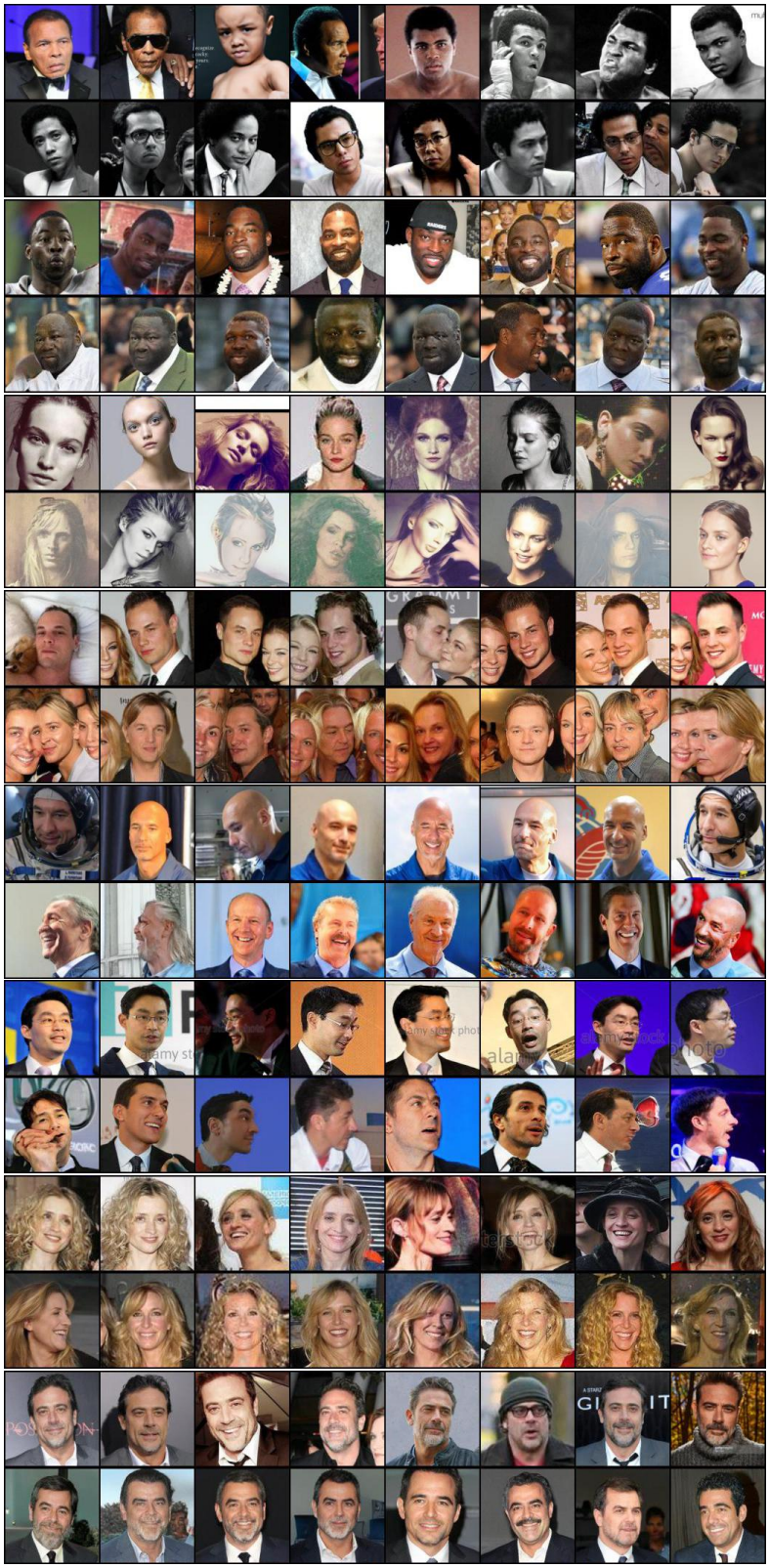


Figure 10: More reconstructions of objects from the test split VggFace2. For each block the top row is the input set and the bottom row is the reconstructed set.

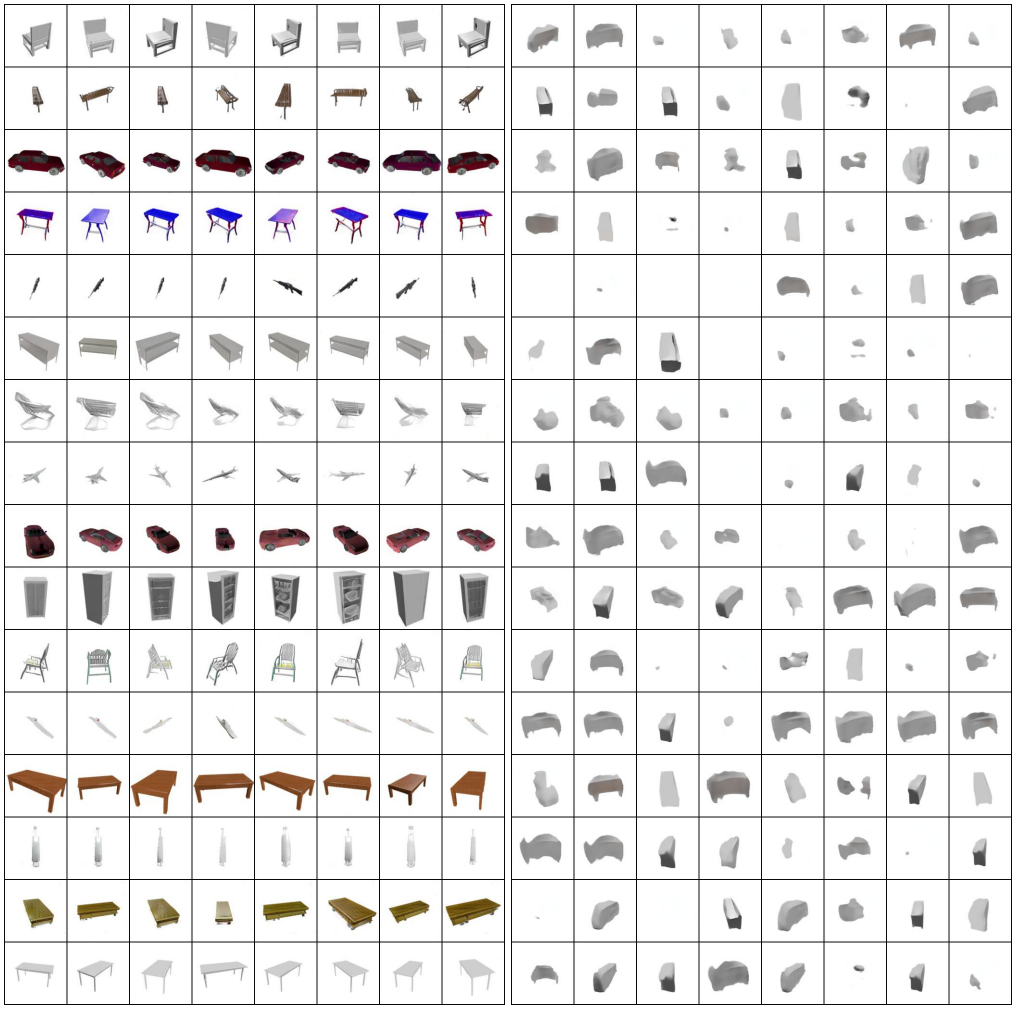


Figure 11: Uncurated ShapeNet samples from the learned autoregressive prior (left) and a uniform prior (right).

D Qualitative Analysis of Within Set Variation

In this section, we visualize the consistency and diversity of samples within a set produced by our SDN trained on ShapeNet object instances. To do so, for several object instances, we manually order samples by their pose (as estimated by a human annotator) to generate a continuum. We then map this continuum to a circle representing how the pose varies along the viewing sphere. By visualizing the input and reconstructed sets in this way we can examine how well the reconstructions capture the pose variability of the inputs. The reconstructions qualitatively capture the appearance of the input set (consistency) and the pose variation around the circle (diversity), with no sign of mode collapse. For the input set, we sampled 16 images randomly for that object. For the reconstructed set, we sampled 64 images, ordered them manually by visual inspection, and then decimated the continuum to obtain 16 reconstruction images.

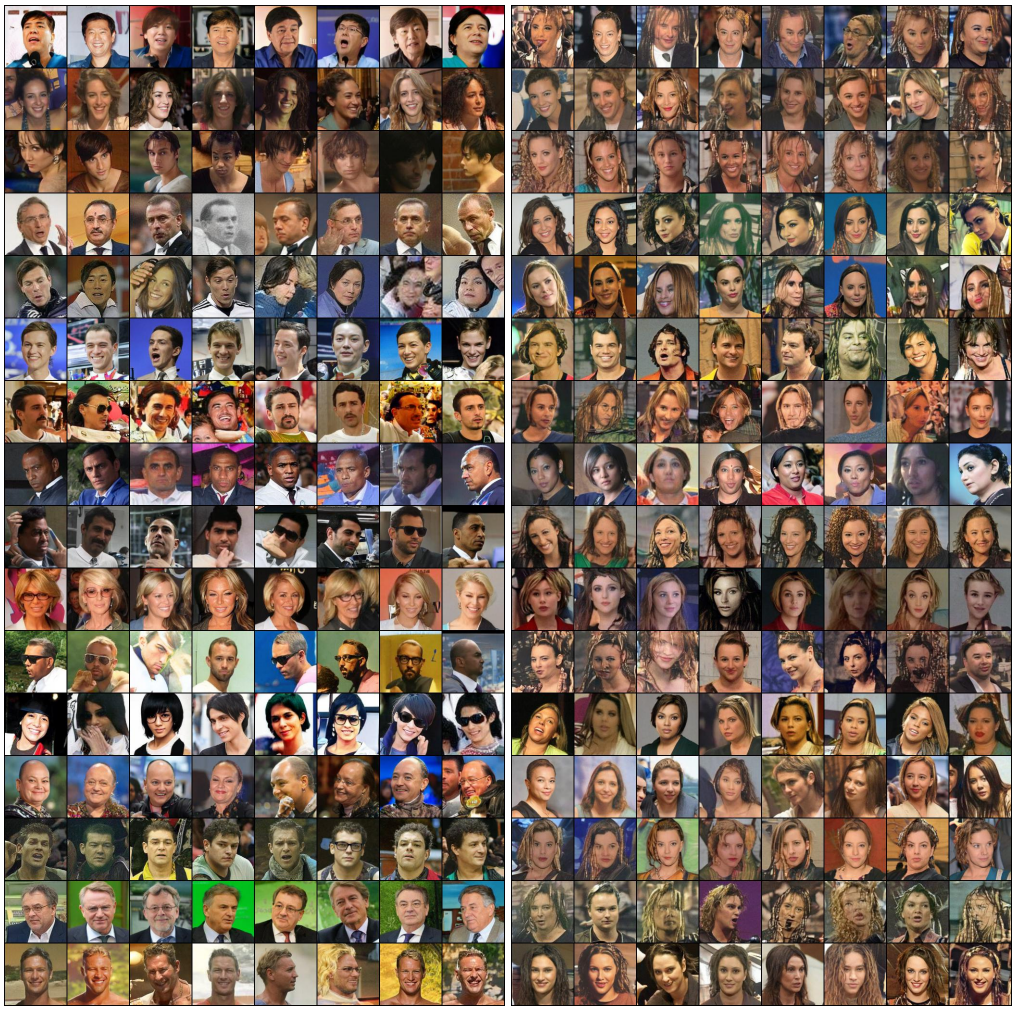


Figure 12: Uncurated VGGFace2 samples from the learned autoregressive prior (left) and a uniform prior (right).

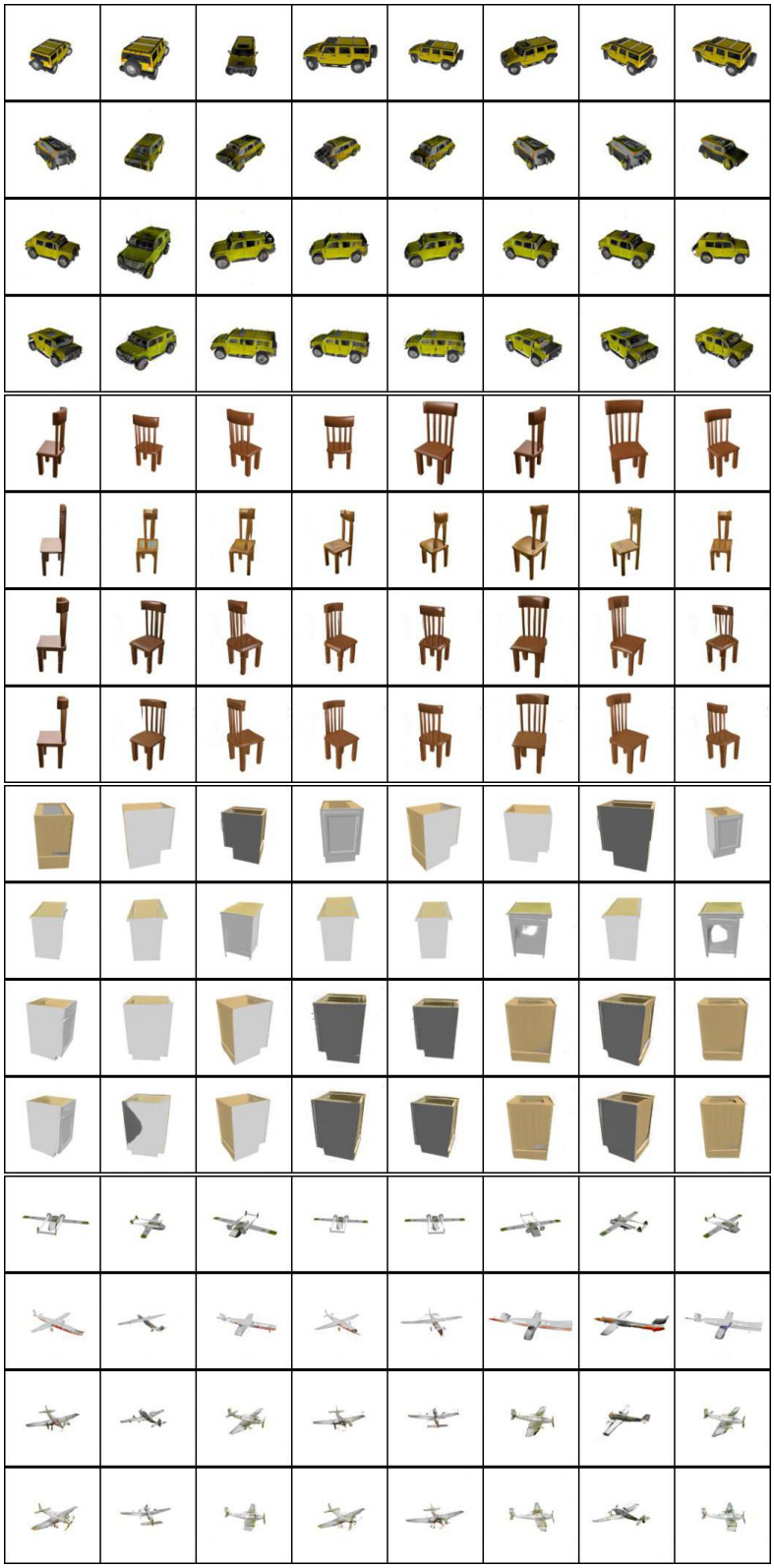


Figure 13: Reconstructions by varying input set size (see Table 1(b) in main text). For each block, the top row is the input set, with the following rows showing reconstructions obtained with the first, first 4 and all 8 views, respectively.

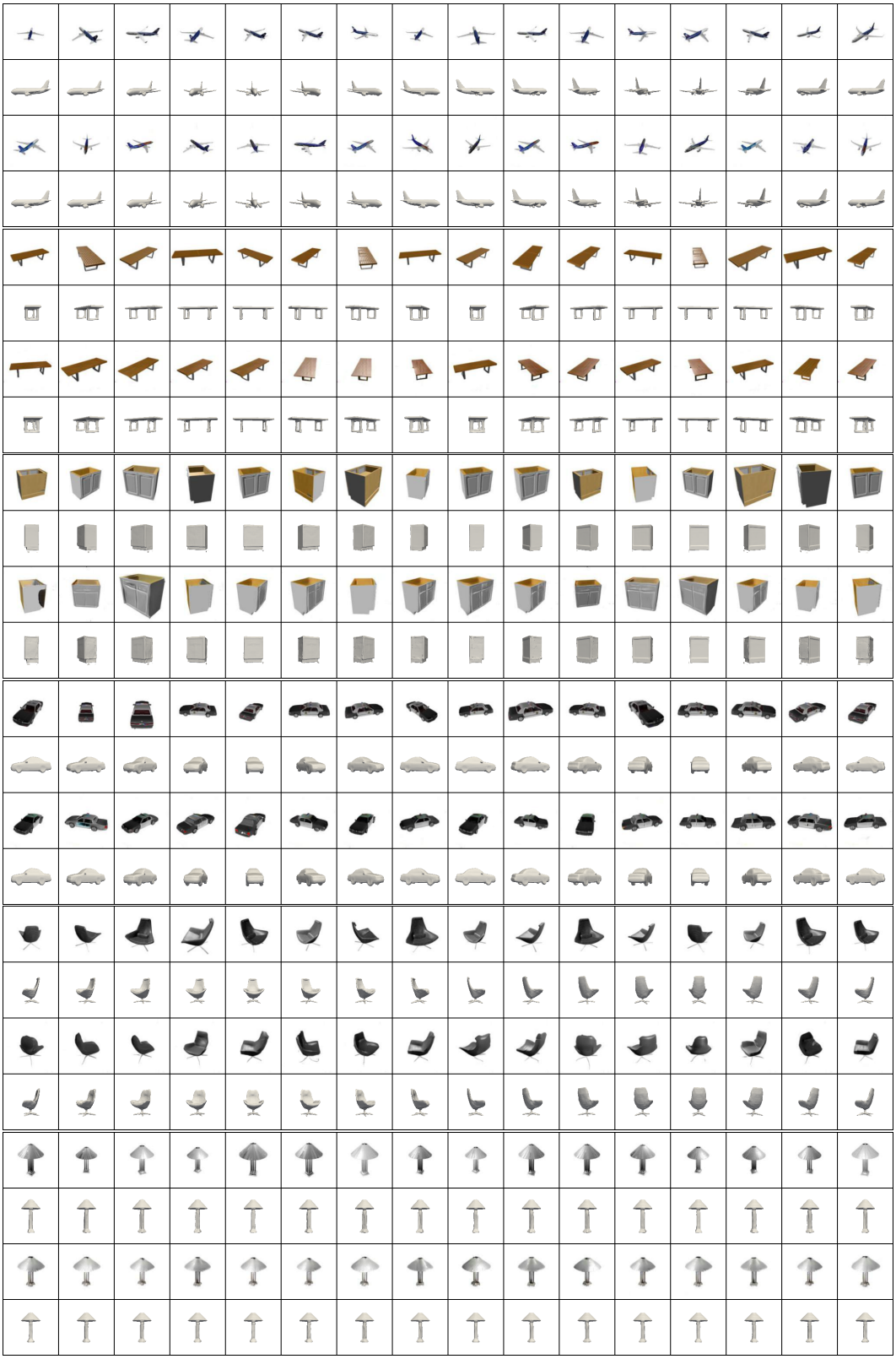


Figure 14: Random sampling of real and SDN-reconstructed sets with their corresponding 3D reconstructions obtained by our Occupancy Network extension. Each block shows an input views, input 3D meshes, reconstructed views, reconstructed 3D meshes, from top to bottom.

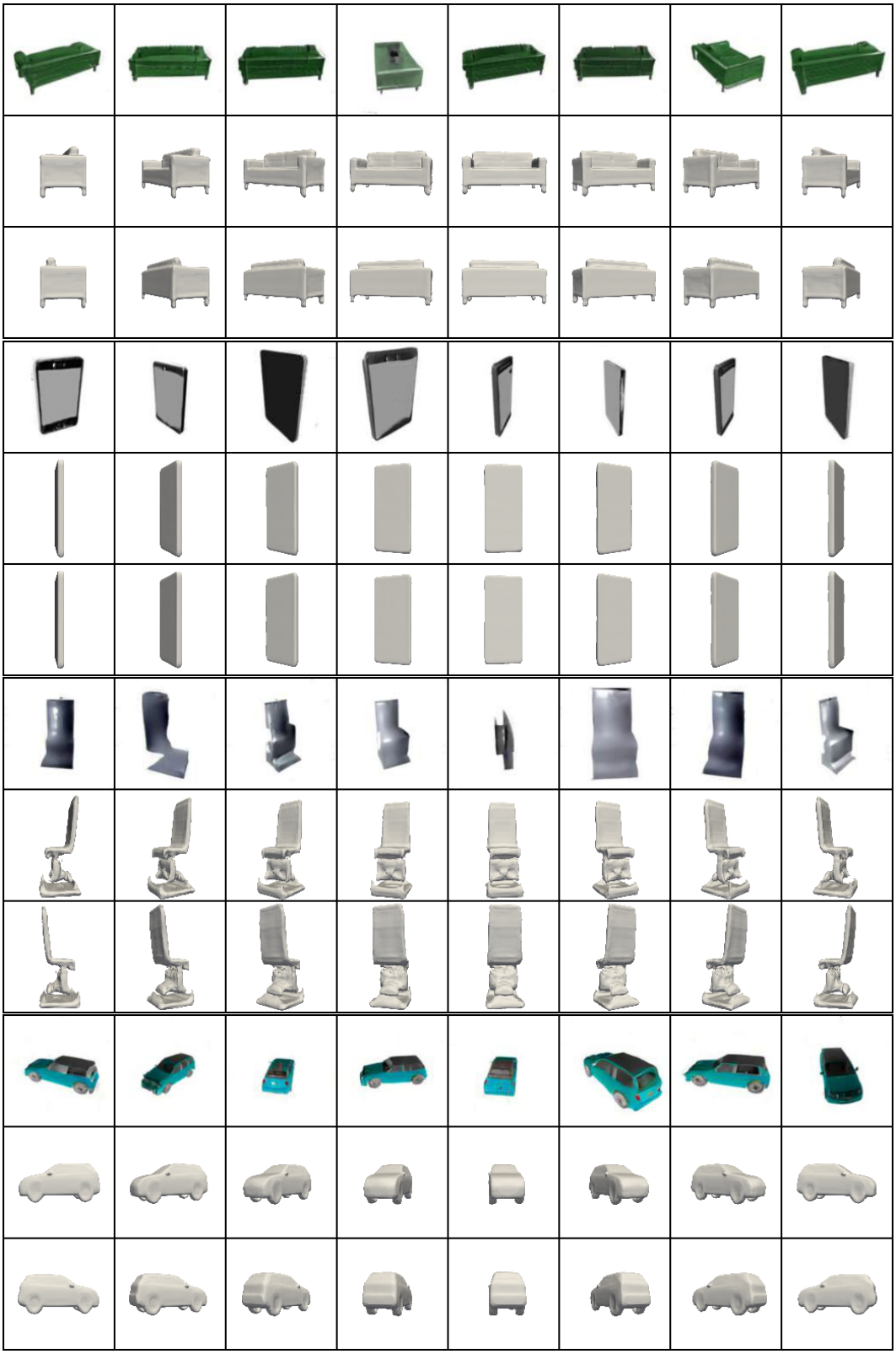


Figure 15: Sets sampled from the SDN prior with their corresponding 3D reconstruction. Each block shows a generated set consisting of 8 views and 16 of its reconstructed 3D mesh projections.

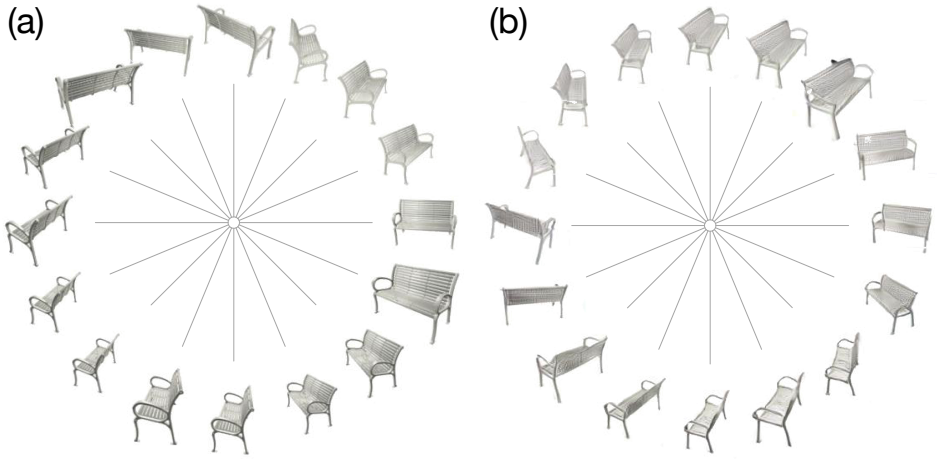


Figure 16: Input (a) and reconstructed (b) sets from a ShapeNet bench where samples are arranged on a circle by manually estimated pose (see Supplementary Material for details). This visualization shows consistency in terms of appearance and pose variability between input and reconstructed sets.