DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation

Jeong Joon Park[1,3†] Peter Florence [2,3†] Julian Straub[3] Richard Newcombe[3] Steven Lovegrove[3]

[1]University of Washington [2]Massachusetts Institute of Technology [3]Facebook Reality Labs

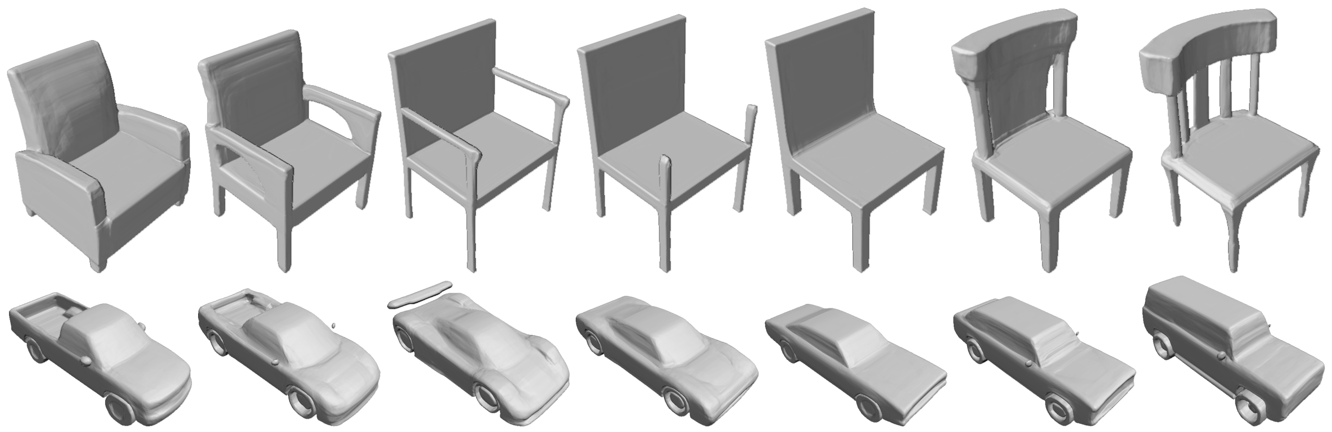


Figure 1: DeepSDF represents signed distance functions (SDFs) of shapes via latent code-conditioned feed-forward decoder networks. Above images are raycast renderings of DeepSDF interpolating between two shapes in the learned shape latent space. Best viewed digitally.

Abstract

Computer graphics, 3D computer vision and robotics communities have produced multiple approaches to representing 3D geometry for rendering and reconstruction. These provide trade-offs across fidelity, efficiency and compression capabilities. In this work, we introduce DeepSDF, a learned continuous Signed Distance Function (SDF) representation of a class of shapes that enables high quality shape representation, interpolation and completion from partial and noisy 3D input data. DeepSDF, like its classical counterpart, represents a shape's surface by a continuous volumetric field: the magnitude of a point in the field represents the distance to the surface boundary and the sign indicates whether the region is inside (-) or outside (+) of the shape, hence our representation implicitly encodes a shape's boundary as the zero-level-set of the learned function while explicitly representing the classification of space as being part of the shapes interior or not. While classical SDF's both in analytical or discretized voxel form typically represent the surface of a single shape, DeepSDF can represent an entire class of shapes. Furthermore, we show state-of-the-art performance for learned 3D shape representation and completion while reducing the model size by an order of magnitude compared with previous work.

† Work performed during internship at Facebook Reality Labs.

1. Introduction

Deep convolutional networks which are a mainstay of image-based approaches grow quickly in space and time complexity when directly generalized to the 3rd spatial dimension, and more classical and compact surface representations such as triangle or quad meshes pose problems in training since we may need to deal with an unknown number of vertices and arbitrary topology. These challenges have limited the quality, flexibility and fidelity of deep learning approaches when attempting to either input 3D data for processing or produce 3D inferences for object segmentation and reconstruction.

In this work, we present a novel representation and approach for generative 3D modeling that is efficient, expressive, and fully continuous. Our approach uses the concept of a SDF, but unlike common surface reconstruction techniques which discretize this SDF into a regular grid for evaluation and measurement denoising, we instead learn a generative model to produce such a continuous field.

The proposed continuous representation may be intuitively understood as a learned shape-conditioned classifier for which the decision boundary is the surface of the shape itself, as shown in Fig. 2. Our approach shares the generative aspect of other works seeking to map a latent space to a distribution of complex shapes in 3D [54], but critically differs in the central representation. While the notion of an

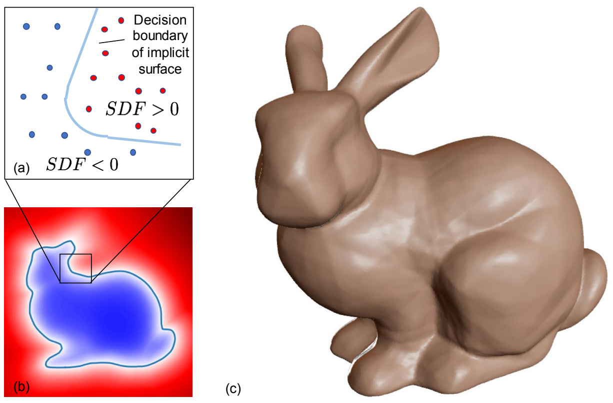


Figure 2: Our DeepSDF representation applied to the Stanford Bunny: (a) depiction of the underlying implicit surface $SDF = 0$ trained on sampled points inside $SDF < 0$ and outside $SDF > 0$ the surface, (b) 2D cross-section of the signed distance field, (c) rendered 3D surface recovered from $SDF = 0$. Note that (b) and (c) are recovered via DeepSDF.

implicit surface defined as a SDF is widely known in the computer vision and graphics communities, to our knowledge no prior works have attempted to directly learn continuous, generalizable 3D generative models of SDFs.

Our contributions include: (i) the formulation of generative shape-conditioned 3D modeling with a continuous implicit surface, (ii) a learning method for 3D shapes based on a probabilistic auto-decoder, and (iii) the demonstration and application of this formulation to shape modeling and completion. Our models produce high quality continuous surfaces with complex topologies, and obtain state-of-the-art results in quantitative comparisons for shape reconstruction and completion. As an example of the effectiveness of our method, our models use only 7.4 MB (megabytes) of memory to represent entire classes of shapes (for example, thousands of 3D chair models) – this is, for example, less than half the memory footprint (16.8 MB) of a single uncompressed 512^3 3D bitmap.

2. Related Work

We review three main areas of related work: 3D representations for shape learning (Sec. 2.1), techniques for learning generative models (Sec. 2.2), and shape completion (Sec. 2.3).

2.1. Representations for 3D Shape Learning

Representations for data-driven 3D learning approaches can be largely classified into three categories: point-based, mesh-based, and voxel-based methods. While some applications such as 3D-point-cloud-based object classification are well suited to these representations, we address their limitations in expressing continuous surfaces with complex topologies.

Point-based. A point cloud is a lightweight 3D representation that closely matches the raw data that many sensors (i.e. LiDARs, depth cameras) provide, and hence is a natural fit for applying 3D learning. PointNet [38, 39], for example, uses max-pool operations to extract global shape features, and the technique is widely used as an encoder for point generation networks [57, 1]. There is a sizable list of related works to the PointNet style approach of learning on point clouds. A primary limitation, however, of learning with point clouds is that they do not describe topology and are not suitable for producing watertight surfaces.

Mesh-based. Various approaches represent classes of similarly shaped objects, such as morphable human body parts, with predefined template meshes and some of these models demonstrate high fidelity shape generation results [2, 34]. Other recent works [3] use poly-cube mapping [51] for shape optimization. While the use of template meshes is convenient and naturally provides 3D correspondences, it can only model shapes with fixed mesh topology.

Other mesh-based methods use existing [48, 36] or learned [22, 23] parameterization techniques to describe 3D surfaces by morphing 2D planes. The quality of such representations depends on parameterization algorithms that are often sensitive to input mesh quality and cutting strategies. To address this, recent data-driven approaches [57, 22] learn the parameterization task with deep networks. They report, however, that (a) multiple planes are required to describe complex topologies but (b) the generated surface patches are not stitched, i.e. the produced shape is not closed. To generate a closed mesh, sphere parameterization may be used [22, 23], but the resulting shape is limited to the topological sphere. Other works related to learning on meshes propose to use new convolution and pooling operations for meshes [17, 53] or general graphs [9].

Voxel-based. Voxels, which non-parametrically describe volumes with 3D grids of values, are perhaps the most natural extension into the 3D domain of the well-known learning paradigms (i.e., convolution) that have excelled in the 2D image domain. The most straightforward variant of voxel-based learning is to use a dense occupancy grid (occupied / not occupied). Due to the cubically growing compute and memory requirements, however, current methods are only able to handle low resolutions (128^3 or below). As such, voxel-based approaches do not preserve fine shape details [56, 14], and additionally voxels visually appear significantly different than high-fidelity shapes, since when rendered their normals are not smooth. Octree-based methods [52, 43, 26] alleviate the compute and memory limitations of dense voxel methods, extending for example the ability to learn at up to 512^3 resolution [52], but even this resolution is far from producing shapes that are visually compelling.

Aside from occupancy grids, and more closely related to our approach, it is also possible to use a 3D grid of voxels to represent a signed distance function. This inherits from the success of fusion approaches that utilize a truncated SDF (TSDF), pioneered in [15, 37], to combine noisy depth maps into a single 3D model. Voxel-based SDF representations have been extensively used for 3D shape learning [59, 16, 49], but their use of discrete voxels is expensive in memory. As a result, the learned discrete SDF approaches generally present low resolution shapes. [30] reports various wavelet transform-based approaches for distance field compression, while [10] applies dimensionality reduction techniques on discrete TSDF volumes. These methods encode the SDF volume of each individual scene rather than a dataset of shapes.

2.2. Representation Learning Techniques

Modern representation learning techniques aim at automatically discovering a set of features that compactly but expressively describe data. For a more extensive review of the field, we refer to Bengio et al. [4].

Generative Adversial Networks. GANs [21] and their variants [13, 41] learn deep embeddings of target data by training discriminators adversarially against generators. Applications of this class of networks [29, 31] generate realstic images of humans, objects, or scenes. On the downside, adversarial training for GANs is known to be unstable. In the 3D domain, Wu et al. [54] trains a GAN to generate objects in a voxel representation, while the recent work of Hamu et al. [23] uses multiple parameterization planes to generate shapes of topological spheres.

Auto-encoders. Auto-encoder outputs are expected to replicate the original input given the constraint of an information bottleneck between the encoder and decoder. The ability of auto-encoders as a feature learning tool has been evidenced by the vast variety of 3D shape learning works in the literature [16, 49, 2, 22, 55] who adopt auto-encoders for representation learning. Recent 3D vision works [6, 2, 34] often adopt a variational auto-encoder (VAE) learning scheme, in which bottleneck features are perturbed with Gaussian noise to encourage smooth and complete latent spaces. The regularization on the latent vectors enables exploring the embedding space with gradient descent or random sampling.

Optimizing Latent Vectors. Instead of using the full auto-encoder for representation learning, an alternative is to learn compact data representations by training *decoder-only* networks. This idea goes back to at least the work of Tan et al. [50] which simultaneously optimizes the latent vectors assigned to each data point and the decoder weights through back-propagation. For inference, an optimal latent vector is searched to match the new observation with fixed decoder parameters. Similar approaches have been extensively studied in [42, 8, 40], for applications including noise reduction, missing measurement completions, and fault detections. Recent approaches [7, 20] extend the technique by applying deep architectures. Throughout the paper we refer to this class of networks as *auto-decoders*, for they are trained with *self*-reconstruction loss on decoder-only architectures.

2.3. Shape Completion

3D shape completion related works aim to infer unseen parts of the original shape given sparse or partial input observations. This task is anaologous to image-inpainting in 2D computer vision.

Classical surface reconstruction methods complete a point cloud into a dense surface by fitting radial basis function (RBF) [11] to approximate implicit surface functions, or by casting the reconstruction from oriented point clouds as a Poisson problem [32]. These methods only model a single shape rather than a dataset.

Various recent methods use data-driven approaches for the 3D completion task. Most of these methods adopt encoder-decoder architectures to reduce partial inputs of occupancy voxels [56], discrete SDF voxels [16], depth maps [44], RGB images [14, 55] or point clouds [49] into a latent vector and subsequently generate a prediction of full volumetric shape based on learned priors.

3. Modeling SDFs with Neural Networks

In this section we present DeepSDF, our continuous shape learning approach. We describe modeling shapes as the zero iso-surface decision boundaries of feed-forward networks trained to represent SDFs. A signed distance function is a continuous function that, for a given spatial point, outputs the point's distance to the closest surface, whose sign encodes whether the point is inside (negative) or outside (positive) of the watertight surface:

$$SDF(\boldsymbol{x}) = s : \boldsymbol{x} \in \mathbb{R}^3,\, s \in \mathbb{R}\,. \tag{1}$$

The underlying surface is implicitly represented by the iso-surface of $SDF(\cdot) = 0$. A view of this implicit surface can be rendered through raycasting or rasterization of a mesh obtained with, for example, Marching Cubes [35].

Our key idea is to directly regress the continuous SDF from point samples using deep neural networks. The resulting trained network is able to predict the SDF value of a given query position, from which we can extract the zero level-set surface by evaluating spatial samples. Such surface representation can be intuitively understood as a learned binary classifier for which the decision boundary is the surface of the shape itself as depicted in Fig. 2. As a universal function approximator [27], deep feed-forward networks in theory can learn the fully continuous shape functions with arbitrary precision. Yet, the precision of the approximation in practice is limited by the finite number of point samples that guide the decision boundaries and the finite capacity of the network due to restricted compute power.

The most direct application of this approach is to train a single deep network for a given target shape as depicted in Fig. 3a. Given a target shape, we prepare a set of pairs X composed of 3D point samples and their SDF values:

$$X := \{(\boldsymbol{x}, s) : SDF(\boldsymbol{x}) = s\}\,. \tag{2}$$

We train the parameters θ of a multi-layer fully-connected neural network f_θ on the training set S to make f_θ a good approximator of the given SDF in the target domain Ω:

$$f_\theta(\boldsymbol{x}) \approx SDF(\boldsymbol{x}),\, \forall \boldsymbol{x} \in \Omega\,. \tag{3}$$

The training is done by minimizing the sum over losses between the predicted and real SDF values of points in X under the following L_1 loss function:

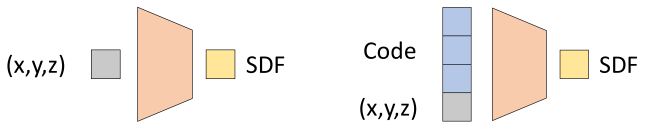
$$\mathcal{L}(f_\theta(\boldsymbol{x}), s) = |\,\text{clamp}(f_\theta(\boldsymbol{x}), \delta) - \text{clamp}(s, \delta)\,|, \tag{4}$$

where $\text{clamp}(x, \delta) := \min(\delta, \max(-\delta, x))$ introduces the parameter δ to control the distance from the surface over which we expect to maintain a metric SDF. Larger values of δ allow for fast ray-tracing since each sample gives information of safe step sizes. Smaller values of δ can be used to concentrate network capacity on details near the surface.

To generate the 3D model shown in Fig. 3a, we use $\delta = 0.1$ and a feed-forward network composed of eight fully connected layers, each of them applied with dropouts. All internal layers are 512-dimensional and have ReLU non-linearities. The output non-linearity regressing the SDF value is tanh. We found training with batch-normalization [28] to be unstable and applied the weight-normalization technique instead [46]. For training, we use the Adam optimizer [33]. Once trained, the surface is implicitly represented as the zero iso-surface of $f_\theta(\boldsymbol{x})$, which can be visualized through raycasting or marching cubes. Another nice property of this approach is that accurate normals can be analytically computed by calculating the spatial derivative $\partial f_\theta(\boldsymbol{x})/\partial \boldsymbol{x}$ via back-propogation through the network.

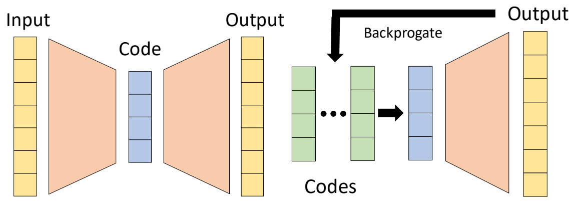
4. Learning the Latent Space of Shapes

Training a specific neural network for each shape is neither feasible nor very useful. Instead, we want a model that can represent a wide variety of shapes, discover their common properties, and embed them in a low dimensional latent space. To this end, we introduce a latent vector $\boldsymbol{z}$, which can be thought of as encoding the desired shape, as a second input to the neural network as depicted in Fig. 3b. Conceptually, we map this latent vector to a 3D shape represented



(a) Single Shape DeepSDF **(b)** Coded Shape DeepSDF

Figure 3: In the single-shape DeepSDF instantiation, the shape information is contained in the network itself whereas the coded-shape DeepSDF, the shape information is contained in a code vector that is concatenated with the 3D sample location. In both cases, DeepSDF produces the SDF value at the 3D query location,



(a) Auto-encoder **(b)** Auto-decoder

Figure 4: Different from an auto-encoder whose latent code is produced by the encoder, an auto-decoder directly accepts a latent vector as an input. A randomly initialized latent vector is assigned to each data point in the beginning of training, and the latent vectors are optimized along with the decoder weights through standard backpropagation. During inference, decoder weights are fixed, and an optimal latent vector is estimated.

by a continuous SDF. Formally, for some shape indexed by i, f_θ is now a function of a latent code $\boldsymbol{z}_i$ and a query 3D location $\boldsymbol{x}$, and outputs the shape's approximate SDF:

$$f_\theta(\boldsymbol{z}_i, \boldsymbol{x}) \approx SDF^i(\boldsymbol{x}). \tag{5}$$

By conditioning the network output on a latent vector, this formulation allows modeling multiple SDFs with a single neural network. Given the decoding model f_θ, the continuous surface associated with a latent vector $\boldsymbol{z}$ is similarly represented with the decision boundary of $f_\theta(\boldsymbol{z}, \boldsymbol{x})$, and the shape can again be discretized for visualization by, for example, raycasting or Marching Cubes.

Next, we motivate the use of encoder-less training before introducing the 'auto-decoder' formulation of the shape-coded DeepSDF.

4.1. Motivating Encoder-less Learning

Auto-encoders and encoder-decoder networks are widely used for representation learning as their bottleneck features tend to form natural latent variable representations.

Recently, in applications such as modeling depth maps [6], faces [2], and body shapes [34] a full auto-encoder is trained but only the decoder is retained for inference, where they search for an optimal latent vector given some input observation. However, since the trained encoder is unused at test time, it is unclear whether using the encoder is the most effective use of computational resources during training. This motivates us to use an *auto*-**decoder** for learning a shape embedding without an encoder as depicted in Fig. 4.

We show that applying an auto-decoder to learn continuous SDFs leads to high quality 3D generative models. Further, we develop a probabilistic formulation for training and testing the auto-decoder that naturally introduces latent space regularization for improved generalization. To the best of our knowledge, this work is the first to introduce the auto-decoder learning method to the 3D learning community.

4.2. Auto-decoder-based DeepSDF Formulation

To derive the auto-decoder-based shape-coded DeepSDF formulation we adopt a probabilistic perspective. Given a dataset of N shapes represented with signed distance function $SDF^{i}{}_{i=1}^{N}$, we prepare a set of K point samples and their signed distance values:

$$X_i = \{(\boldsymbol{x}_j, s_j) : s_j = SDF^i(\boldsymbol{x}_j)\}\,. \tag{6}$$

For an auto-decoder, as there is no encoder, each latent code $\boldsymbol{z}_i$ is paired with training shape X_i. The posterior over shape code $\boldsymbol{z}_i$ given the shape SDF samples X_i can be decomposed as:

$$p_\theta(\boldsymbol{z}_i|X_i) = p(\boldsymbol{z}_i)\prod_{(\boldsymbol{x}_j,\boldsymbol{s}_j)\in X_i} p_\theta(\boldsymbol{s}_j|z_i;\boldsymbol{x}_j)\,, \tag{7}$$

where θ parameterizes the SDF likelihood. In the latent shape-code space, we assume the prior distribution over codes $p(\boldsymbol{z_i})$ to be a zero-mean multivariate-Gaussian with a spherical covariance $\sigma^2 I$. This prior encapsulates the notion that the shape codes should be concentrated and we empirically found it was needed to infer a compact shape manifold and to help converge to good solutions.

In the auto-decoder-based DeepSDF formulation we express the SDF likelihood via a deep feed-forward network $f_\theta(\boldsymbol{z}_i, \boldsymbol{x}_j)$ and, without loss of generality, assume that the likelihood takes the form:

$$p_\theta(\boldsymbol{s}_j|z_i;\boldsymbol{x}_j) = \exp(-\mathcal{L}(f_\theta(\boldsymbol{z}_i,\boldsymbol{x}_j), s_j))\,. \tag{8}$$

The SDF prediction $\tilde{s}_j = f_\theta(\boldsymbol{z}_i, \boldsymbol{x}_j)$ is represented using a fully-connected network. $\mathcal{L}(\tilde{s}_j, s_j)$ is a loss function penalizing the deviation of the network prediction from the actual SDF value s_j. One example for the cost function is the standard L_2 loss function which amounts to assuming Gaussian noise on the SDF values. In practice we use the clamped L_1 cost from Eq. 4 for reasons outlined previously.

At training time we maximize the joint log posterior over all training shapes with respect to the individual shape codes $\{z_i\}_{i=1}^N$ and the network parameters θ:

$$\underset{\theta,\{\boldsymbol{z}_i\}_{i=1}^N}{\arg\min} \sum_{i=1}^{N}\left(\sum_{j=1}^{K}\mathcal{L}(f_\theta(\boldsymbol{z}_i,\boldsymbol{x}_j), s_j) + \frac{1}{\sigma^2}||\boldsymbol{z}_i||_2^2\right). \tag{9}$$

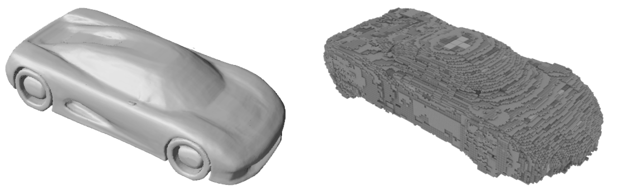


Figure 5: Compared to car shapes memorized using OGN [52] (right), our models (left) preserve details and render visually pleasing results as DeepSDF provides oriented surace normals.

At inference time, after training and fixing θ, a shape code $\boldsymbol{z}_i$ for shape X_i can be estimated via Maximum-a-Posterior (MAP) estimation as:

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z}}{\arg\min} \sum_{(\boldsymbol{x}_j,\boldsymbol{s}_j)\in X}\mathcal{L}(f_\theta(\boldsymbol{z},\boldsymbol{x}_j), s_j) + \frac{1}{\sigma^2}||\boldsymbol{z}||_2^2\,. \tag{10}$$

Crucially, this formulation is valid for SDF samples X of arbitrary size and distribution because the gradient of the loss with respect to $\boldsymbol{z}$ can be computed separately for each SDF sample. This implies that DeepSDF can handle any form of partial observations such as depth maps. This is a major advantage over the auto-encoder framework whose encoder expects a test input similar to the training data, e.g. shape completion networks of [16, 58] require preparing training data of partial shapes.

To incorporate the latent shape code, we stack the code vector and the sample location as depicted in Fig. 3b and feed it into the same fully-connected NN described previously at the input layer and additionally at the 4th layer. We again use the Adam optimizer [33]. The latent vector $\boldsymbol{z}$ is initialized randomly from $\mathcal{N}(0, 0.01^2)$.

Note that while both VAE and the proposed auto-decoder formulation share the zero-mean Gaussian prior on the latent codes, we found that the the stochastic nature of the VAE optimization did not lead to good training results.

5. Data Preparation

To train our continuous SDF model, we prepare the SDF samples X (Eq. 2) for each mesh, which consists of 3D points and their SDF values. While SDF can be computed through a distance transform for any watertight shapes from real or synthetic data, we train with synthetic objects, (e.g. ShapeNet [12]), for which we are provided complete 3D shape meshes. To prepare data, we start by normalizing each mesh to a unit sphere and sampling 500,000 spatial points $\boldsymbol{x}$'s: we sample more aggressively near the surface of the object as we want to capture a more detailed SDF near the surface. For an ideal oriented watertight mesh, computing the signed distance value of $\boldsymbol{x}$ would only involve finding the closest triangle, but we find that human designed meshes are commonly not watertight and contain undesired internal structures. To obtain the *shell* of a

Method	Type	Discretization	Complex topologies	Closed surfaces	Surface normals	Model size (GB) (s)	Inf. time (s)	Eval. tasks
3D-EPN [16]	Voxel SDF	32^3 voxels	✓	✓	✓	0.42	-	C
OGN [52]	Octree	256^3 voxels	✓	✓		0.54	0.32	K
AtlasNet -Sphere [22]	Parametric mesh	1 patch		✓		0.015	**0.01**	K, U
AtlasNet -25 [22]	Parametric mesh	25 patches	✓			0.172	0.32	K, U
DeepSDF (ours)	Continuous SDF	**none**	✓	✓	✓	**0.0074**	9.72	K, U, C

Table 1: Overview of the benchmarked methods. AtlasNet-Sphere can only describe topological-spheres, voxel/octree occupancy methods (i.e. OGN) only provide 8 directions for normals, and AtlasNet does not provide oriented normals. Our tasks evaluated are: (K) representing known shapes, (U) representing unknown shapes, and (C) shape completion.

mesh with proper orientation, we set up equally spaced virtual cameras around the object, and densely sample surface points, denoted P_s, with surface normals oriented towards the camera. Double sided triangles visible from both orientations (indicating that the shape is not closed) cause problems in this case, so we discard mesh objects containing too many of such faces. Then, for each $\boldsymbol{x}$, we find the closest point in P_s, from which the $SDF(\boldsymbol{x})$ can be computed. We refer readers to supplementary material for further details.

6. Results

We conduct a number of experiments to show the representational power of DeepSDF, both in terms of its ability to describe geometric details and its generalization capability to learn a desirable shape embedding space. Largely, we propose four main experiments designed to test its ability to 1) represent training data, 2) use learned feature representation to reconstruct unseen shapes, 3) apply shape priors to complete partial shapes, and 4) learn smooth and complete shape embedding space from which we can sample new shapes. For all experiments we use the popular ShapeNet [12] dataset.

We select a representative set of 3D learning approaches to comparatively evaluate aforementioned criteria: a recent octree-based method (OGN) [52], a mesh-based method (AtlasNet) [22], and a volumetric SDF-based shape completion method (3D-EPN) [16] (Table 1). These works show state-of-the-art performance in their respective representations and tasks, so we omit comparisons with the works that have already been compared: e.g. OGN's octree model outperforms regular voxel approaches, while AtlasNet compares itself with various points, mesh, or voxel based methods and 3D-EPN with various completion methods.

6.1. Representing Known 3D Shapes

First, we evaluate the capacity of the model to represent *known* shapes, i.e. shapes that were in the training set, from only a restricted-size latent code — testing the limit of expressive capability of the representations.

Method \metric	CD, mean	CD, median	EMD, mean	EMD, median
OGN	0.167	0.127	**0.043**	**0.042**
AtlasNet-Sph.	0.210	0.185	0.046	0.045
AtlasNet-25	0.157	0.140	0.060	0.060
DeepSDF	**0.084**	**0.058**	**0.043**	**0.042**

Table 2: Comparison for representing known shapes (K) for cars trained on ShapeNet. CD = Chamfer Distance ($30,000$ points) multiplied by 10^3, EMD = Earth Mover's Distance (500 points).

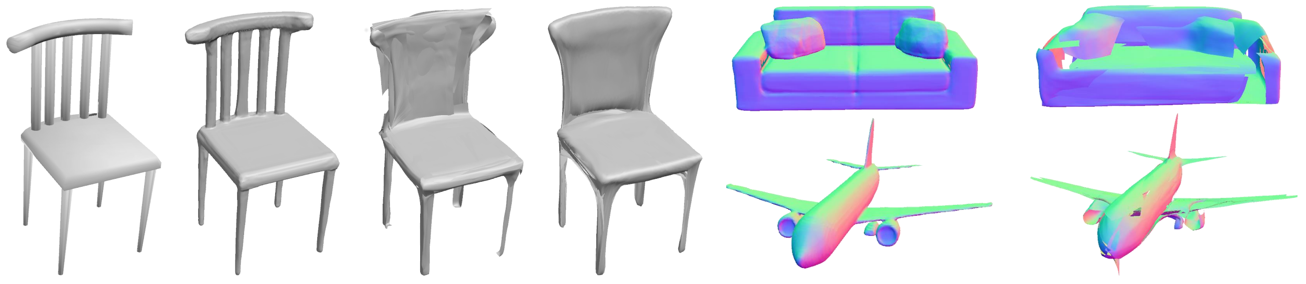
Quantitative comparison in Table 2 shows that the proposed DeepSDF significantly beats OGN and AtlasNet in Chamfer distance against the true shape computed with a large number of points (30,000). The difference in earth mover distance (EMD) is smaller because 500 points do not well capture the additional precision. Figure 5 shows a qualitative comparison of DeepSDF to OGN.

6.2. Representing Test 3D Shapes (auto-encoding)

For encoding *unknown* shapes, i.e. shapes in the held-out test set, DeepSDF again significantly outperforms AtlasNet on a wide variety of shape classes and metrics as shown in Table 3. Note that AtlasNet performs reasonably well at classes of shapes that have mostly consistent topology without holes (like planes) but struggles more on classes that commonly have holes, like chairs. This is shown in Fig. 6 where AtlasNet fails to represent the fine detail of the back of the chair. Figure 7 shows more examples of detailed reconstructions on test data from DeepSDF as well as two example failure cases.

6.3. Shape Completion

A major advantage of the proposed DeepSDF approach for representation learning is that inference can be performed from an arbitrary number of SDF samples. In the DeepSDF framework, shape completion amounts to solving for the shape code that best explains a partial shape observation via Eq. 10. Given the shape-code a complete shape can be rendered using the priors encoded in the decoder.



(a) Ground-truth **(b)** Our Result **(c)** [22]-25 patch **(d)** [22]-sphere **(e)** Our Result **(f)** [22]-25 patch

Figure 6: Reconstruction comparison between DeepSDF and AtlasNet [22] (with 25-plane and sphere parameterization) for test shapes. Note that AtlasNet fails to capture the fine details of the chair, and that (f) shows holes on the surface of sofa and the plane.

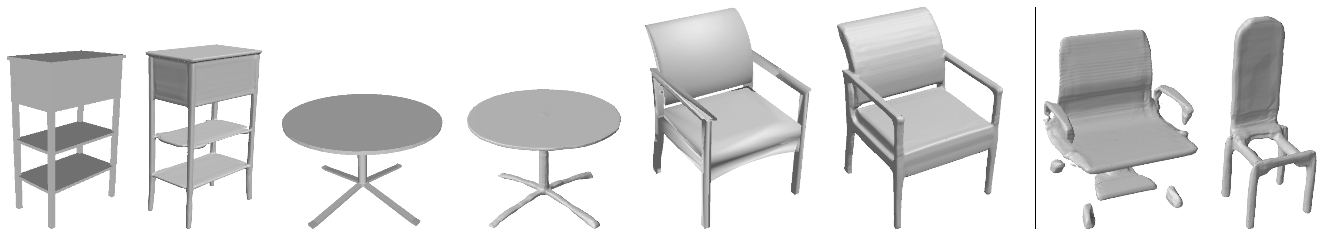


Figure 7: Reconstruction of test shapes. From left to right alternating: ground truth shape and our reconstruction. The two right most columns show failure modes of DeepSDF. These failures are likely due to lack of training data and failure of minimization convergence.

CD, mean	chair	plane	table	lamp	sofa
AtlasNet-Sph.	0.752	0.188	0.725	2.381	0.445
AtlasNet-25	0.368	0.216	**0.328**	1.182	0.411
DeepSDF	**0.204**	**0.143**	0.553	**0.832**	**0.132**
CD, median					
AtlasNet-Sph.	0.511	0.079	0.389	2.180	0.330
AtlasNet-25	0.276	0.065	0.195	0.993	0.311
DeepSDF	**0.072**	**0.036**	**0.068**	**0.219**	**0.088**
EMD, mean					
AtlasNet-Sph.	0.071	0.038	0.060	0.085	0.050
AtlasNet-25	0.064	0.041	0.073	0.062	0.063
DeepSDF	**0.049**	**0.033**	**0.050**	**0.059**	**0.047**
Mesh acc., mean					
AtlasNet-Sph.	0.033	0.013	0.032	0.054	0.017
AtlasNet-25	0.018	0.013	0.014	0.042	0.017
DeepSDF	**0.009**	**0.004**	**0.012**	**0.013**	**0.004**

Table 3: Comparison for representing unknown shapes (U) for various classes of ShapeNet. Mesh accuracy as defined in [47] is the minimum distance d such that 90% of generated points are within d of the ground truth mesh. Lower is better for all metrics.

	lower is better				*higher is better*	
Method \Metric	CD, med.	CD, mean	EMD	Mesh acc.	Mesh comp.	Cos sim.
chair						
3D-EPN	2.25	2.83	0.084	0.059	0.209	0.752
DeepSDF	**1.28**	**2.11**	**0.071**	**0.049**	**0.500**	**0.766**
plane						
3D-EPN	1.63	2.19	0.063	0.040	0.165	0.710
DeepSDF	**0.37**	**1.16**	**0.049**	**0.032**	**0.722**	**0.823**
sofa						
3D-EPN	2.03	2.18	0.071	0.049	0.254	0.742
DeepSDF	**0.82**	**1.59**	**0.059**	**0.041**	**0.541**	**0.810**

Table 4: Comparison for shape completion (C) from partial range scans of unknown shapes from ShapeNet.

We test our completion scheme using single view depth observations which is a common use-case and maps well to our architecture without modification. Note that we currently require the depth observations in the canonical shape frame of reference.

To generate SDF point samples from the depth image observation, we sample two points for each depth observation, each of them located η distance away from the measured surface point (along surface normal estimate). With small η we approximate the signed distance value of those points to be η and $-\eta$, respectively. We solve for Eq. 10 with loss function of Eq. 4 using clamp value of η. Additionally, we incorporate free-space observations, (i.e. empty-space between surface and camera), by sampling points along the freespace-direction and enforce larger-than-zero constraints. The freespace loss is $|f_\theta(\boldsymbol{z}, \boldsymbol{x}_j)|$ if $f_\theta(\boldsymbol{z}, \boldsymbol{x}_j) < 0$ and 0 otherwise.

Given the SDF point samples and empty space points, we similarly optimize the latent vector using MAP estimation. Tab. 4 and Figs. (22, 9) respectively shows quantitative and qualitative shape completion results. Compared to one of the most recent completion approaches [16] using volu-

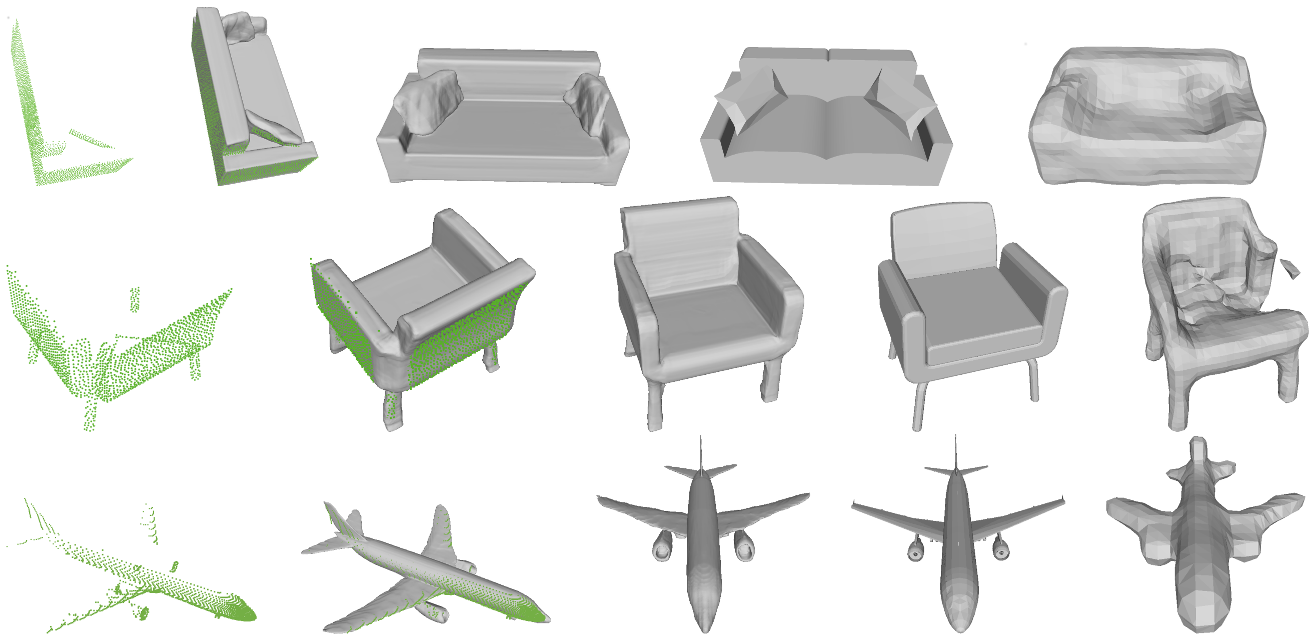

(a) Input Depth **(b)** Completion (ours) **(c)** Second View (ours) **(d)** Ground truth **(e)** 3D-EPN

Figure 8: For a given depth image visualized as a green point cloud, we show a comparison of shape completions from our DeepSDF approach against the true shape and 3D-EPN.

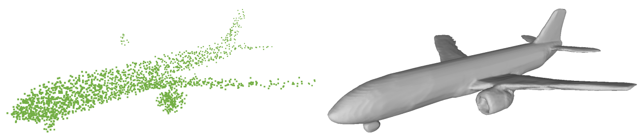

(a) Noisy Input Point Cloud **(b)** Shape Completion

Figure 9: Demonstration of DeepSDF shape completion from a partial noisy point cloud. Input here is generated by perturbing the 3D point cloud positions generated by the ground truth depth map by 3% of the plane length. We provide a comprehensive analysis of robustness to noise in the supplementary material.

metric shape representation, our continuous SDF approach produces more visually pleasing and accurate shape reconstructions. While a few recent shape completion methods were presented [24, 55], we could not find the code to run the comparisons, and their underlying 3D representation is voxel grid which we extensively compare against.

6.4. Latent Space Shape Interpolation

To show that our learned shape embedding is complete and continuous, we render the results of the decoder when a pair of shapes are interpolated in the latent vector space (Fig. 1). The results suggests that the embedded continuous SDF's are of meaningful shapes and that our representation extracts common interpretable shape features, such as the arms of a chair, that interpolate linearly in the latent space.

7. Conclusion & Future Work

DeepSDF significantly outperforms the applicable benchmarked methods across shape representation and completion tasks and simultaneously addresses the goals of representing complex topologies, closed surfaces, while providing high quality surface normals of the shape. However, while point-wise forward sampling of a shape's SDF is efficient, shape completion (auto-decoding) takes considerably more time during inference due to the need for explicit optimization over the latent vector. We look to increase performance by replacing ADAM optimization with more efficient Gauss-Newton or similar methods that make use of the analytic derivatives of the model.

DeepSDF models enable representation of more complex shapes without discretization errors with significantly less memory than previous state-of-the-art results as shown in Table 1, demonstrating an exciting route ahead for 3D shape learning. The clear ability to produce quality latent shape space interpolation opens the door to reconstruction algorithms operating over scenes built up of such efficient encodings. However, DeepSDF currently assumes models are in a canonical pose and as such completion in-the-wild requires explicit optimization over a $SE(3)$ transformation space increasing inference time. Finally, to represent the true *space-of-possible-scenes* including dynamics and textures in a single embedding remains a major challenge, one which we continue to explore.