TRAINING DATA GENERATING NETWORKS: LINKING 3D SHAPES AND FEW-SHOT CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a novel 3d shape representation for 3d shape reconstruction from a single image. Rather than predicting a shape directly, we train a network to generate a training set which will be feed into another learning algorithm to define the shape. Training data generating networks establish a link between few-shot learning and 3d shape analysis. We propose a novel meta-learning framework to jointly train the data generating network and other components. We improve upon recent work on standard benchmarks for 3d shape reconstruction, but our novel shape representation has many applications.

1 INTRODUCTION

Neural networks have shown promising results for shape reconstruction (Wang et al. (2018); Groueix et al. (2018); Mescheder et al. (2019); Genova et al. (2019)). Different from the image domain, there is no universally agreed upon way to represent 3d shapes. There exist many explicit and implicit representations. Explicit representations include point clouds (Qi et al. (2017a;b); Lin et al. (2017)), grids (Wu et al. (2015); Choy et al. (2016); Riegler et al. (2017); Wang et al. (2017); Tatarchenko et al. (2017)), and meshes (Wang et al. (2018); Groueix et al. (2018); Hanocka et al. (2019)). Implicit representations (Mescheder et al. (2019); Michalkiewicz et al. (2019); Park et al. (2019)) define shapes as iso-surfaces of functions. Both types of representations are important as they have different advantages.

In this work, we propose a novel implicit representation. Most implicit representations approximate a 3d function directly by a deep neural network and extract an iso-surface. By contrast, our model learns to predict a labelled point set which is fed into a binary classifier to form a decision boundary defining the surface of the shape. We use this representation to reconstruct a 3d shape from a single image. We improve the reconstruction accuracy compared to the state of the art.

Our framework is inspired by few-shot learning for images. Specifically, our solution combines the idea of a data set generation network with a meta-learning framework for few-shot classification. Few-shot learning can be described as tackling a collection of supervised learning tasks with few training samples. To map 3d shape reconstruction from a single image to few shot learning, we treat each shape in a collection as a separate task and propose a PointGen network to generate the training data for the task.

In Fig. 1, we show a summary of our pipeline and the idea. The model takes as input an image, which is then mapped to a feature vector. Then we use a point generation network to create a *labelled* point set which will be used as training set for another machine learning algorithm (Kernel-SVM). Another important component is an embedding network that warps the 3D space conditioned on the input image. The embedding network warps the space so that the decision boundary is simplified and a 3d shape can be represented by a smaller number of points. Finally, we build a decision surface by feeding the point set (in embedding space) as training samples. This model is able to output the inside/outside label of a query point.

Contributions.

- We propose a new type of shape representation, where we train a network to output a training set (a set of labeled points) for another machine learning algorithm (Kernel-SVM).

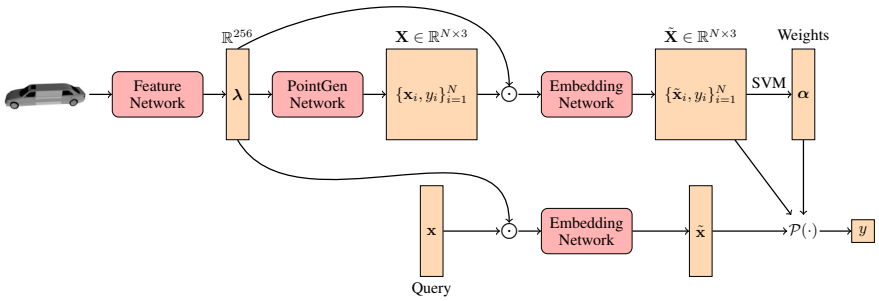


Figure 1: **Pipeline.** Networks with trainable parameters are shown in red boxes with round corners. Outputs are shown in yellow boxes. Arrows show how the data flows in the network. The sign $\odot$ means we are concatenating multiple inputs.

- We found an elegant way to map the problem of shape reconstruction from a single image to the problem of few-shot classification by introducing a network to generate training data.
- We validate our model using the problem of 3d shape reconstruction from single image and improve upon the state of the art.

2 Related works

Single image 3D reconstruction There are various ways to represent shapes, which can be divided into two major categories: *explicit* representations, where a shape can be explicitly defined; and *implicit* representations, where a shape can be defined as iso-surface of a function (signed distance function or indicator function). In the past decade, we have seen great success in neural network based explicit shape representation analysis: voxel representations (Wu et al. (2015); Choy et al. (2016); Riegler et al. (2017); Wang et al. (2017); Tatarchenko et al. (2017)), point representations (Qi et al. (2017a;b); Fan et al. (2017); Lin et al. (2017)), mesh representations(Wang et al. (2018); Groueix et al. (2018); Hanocka et al. (2019)). On the other hand, modeling implicit representations with neural networks has been a current trend, where usually the signed distance function or indicator function is parameterized by a neural network (Mescheder et al. (2019); Chen & Zhang (2019); Michalkiewicz et al. (2019); Park et al. (2019). More recent works learn a network that outputs intermediate parameters, *e.g.* CvxNet (Deng et al., 2019) and BSP-Net (Chen et al., 2019) learns to output half-spaces. We propose a novel type of shape representation, where the model outputs a training set.

Few-shot learning There are two common meta learning approaches for few-shot learning: metric-based (Koch et al. (2015); Vinyals et al. (2016); Snell et al. (2017)), which aims to learn a metric for each task; optimization-based (Ravi & Larochelle (2017); Finn et al. (2017); Nichol et al. (2018)), which is designed to learn with a few training samples by adjusting optimization algorithms. These approaches commonly have two parts, an embedding model for mapping an input to an embedding space, and a base learner for prediction. Bertinetto et al. (2019) showed that using a light-weight and differentiable base learner (*e.g.* ridge regression) leads to better results. To further developing the idea, Lee et al. (2019) used multi-class support vector machine (Crammer & Singer (2001)) as base learner and incorporated differentiable optimization (Amos & Kolter (2017); Gould et al. (2016)) into the framework. In our work, we propose a shape representation that is compatible with a few-shot classification framework so that we can utilize existing meta learning approaches. Specifically, we will use SVM as the base learner.

3 Method

The framework is shown in Fig. 1. The network is mainly composed by 3 sub-networks. The Feature Network maps an input image to feature space. The resulting feature vector $\boldsymbol{\lambda}$ is then decoded by the PointGen Network to a labeled point set $\{\mathbf{x}_i, y_i\}_{i=1}^N$. After that the Embedding Network projects

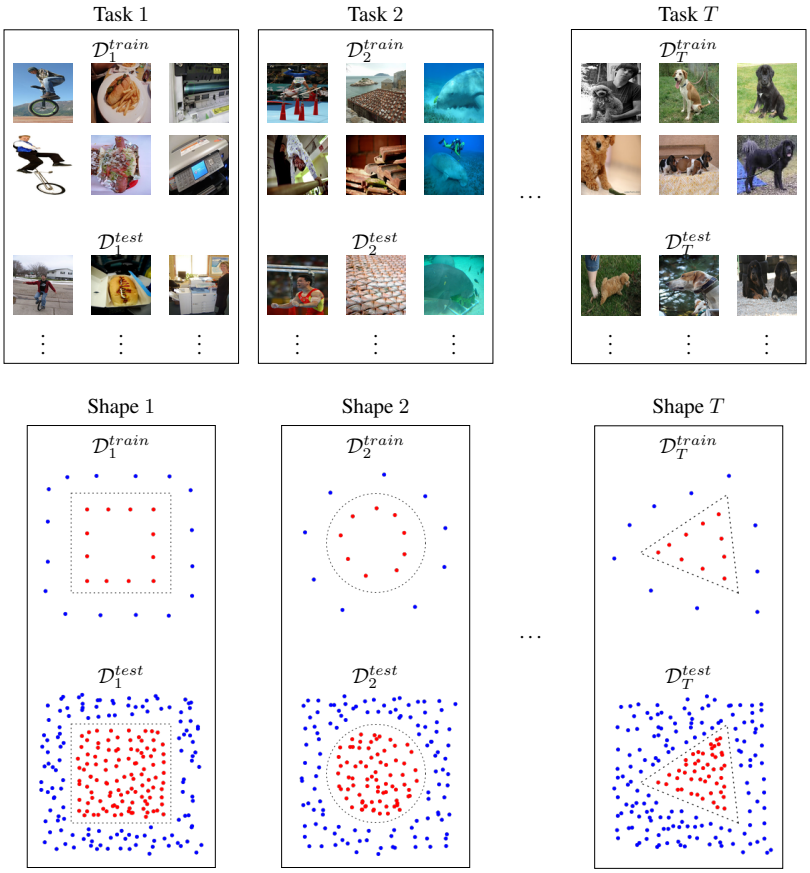


Figure 2: **Few-shot learning v.s. shape representation. Top row:** few-shot classification (images taken from the dataset miniImageNet). In this figure, we show that 2-shot-3-way few-shot classification in which every training task contains 3 categories and 2 training samples in each categories. Thus we use $2 \times 3 = 6$ training samples to build classifiers which are expected to work well on $\mathcal{D}_t^{test}$. **Bottom row:** shape representation. Dashed lines are the surface of shapes. Suppose we can find sets of points with labels blue or red for shapes. Then we use them as training data to build the surfaces (classification boundaries) which should be approximations to the ground-truth surfaces (dashed lines).

the point set into embedding space along with $\boldsymbol{\lambda}$. The projected points $\tilde{\mathbf{x}}_i$ and the label are taken as the input of a binary classifer (SVM) parameterized by $\boldsymbol{\alpha}$. Finally, the network is able to output the inside/outside label y of a query point.

In the following subsections, we describe our method in more detail. First, we introduce the background of meta learning approaches for few-show learning (Sec. 3.1) and establish a link between single image 3D reconstruction and few-shot learning (Sec. 3.2). We propose a problem formulation inspired by few-shot learning (Sec. 3.3) and propose a solution in the following subsections. Specifically, we apply recently developed differentiable optimization.

3.1 Background

Supervised learning Given training set $\mathcal{D}^{train} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, supervised learning learns a predictor $y = \mathcal{P}(\mathbf{x})$ which is able to predict the labels of test set $\mathcal{D}^{test} = \{\mathbf{x}_i, y_i\}_{i=1}^M$ (assuming both $\mathcal{D}^{train}$ and $\mathcal{D}^{test}$ are sampled from the same distribution).

Few-shot learning In few-shot learning, the size N of the training set is typically small. The common learning algorithms on a single task usually cause problems like overfitting. However, we

	Few-shot classification	3D shape reconstruction
$\mathbf{I}$	-	input images
$\mathbf{x}_i$	images	points
y_i	categories	inside/outside labels
learner $\mathcal{P}(\cdot)$	classifier	surface boundary
$\mathcal{D}^{train}$	$\{\mathbf{x}_i, y_i\}_{i=1}^N$	-
$\mathcal{D}^{test}$	$\{\mathbf{x}_i, y_i\}_{i=1}^M$	$\{\mathbf{x}_i, y_i\}_{i=1}^M$
$\mathcal{D}^{meta-train}$	$\{\mathcal{D}_t^{train}, \mathcal{D}_t^{test}\}_{t=1}^T$	$\{\mathbf{I}_t, \mathcal{D}_t^{test}\}_{t=1}^T$
f	-	$f(\mathbf{I}_t) = \mathcal{D}_t^{train}$

Table 1: Symbols for few-shot classification and 3D reconstruction

are given a collection of tasks, the meta-training set $\mathcal{D}^{meta-train} = \{\mathcal{D}_t^{train}, \mathcal{D}_t^{test}\}_{t=1}^T$, on which we train a *meta-learner* which produces a predictor on every task $\{\mathcal{D}_t^{train}, \mathcal{D}_t^{test}\}$ and generalizes well on the meta-testing $\mathcal{D}^{meta-test} = \{\mathcal{D}_s^{train}, \mathcal{D}_s^{test}\}_{s=1}^S$.

Consider a K-class classification task, each training set $\mathcal{D}^{train}$ consists of N/K labelled examples for each of K classes. The meta-training task is often referred to as N/K-*shot*-K-*way*. Refer to Figure 2 for an example visualization of 2-shot-3-way few-shot classification.

Meta learning approaches for few-shot learning often involves an embedding network g, and a base learner $\mathcal{P}(\cdot)$. The embedding network g maps training samples to an embedding space. We explain in later subsections how the 3d reconstruction is connected to meta learning in these two aspects.

3.2 Single image 3D reconstruction

A watertight shape can be represented by an indicator (or occupancy) function $\mathcal{O} : \mathbb{R}^3 \to \{0, 1\}$. We define $\mathcal{O}(\mathbf{x}) = 1$ if $\mathbf{x} \in \mathbb{R}^3$ is inside the object, $\mathcal{O}(\mathbf{x}) = 0$ otherwise. We can sample a set of points in $\mathbb{R}^3$ and evaluate the indicator $\mathcal{O}$, then we have the labeled point set $\{\mathbf{x}_i, y_i\}_{i=1}^M$ where $y_i \in \{0, 1\}$. The number M needs to be large enough to approximate the shape. In this way, we rewrite the target ground-truth as a point set. This strategy is also used by Mescheder et al. (2019) and Deng et al. (2019). Also see Figure 2 for an illustration.

The goal of single image 3D reconstruction is to convert an input image $\mathbf{I}$ to the indicator function $\mathcal{O}$. Previous work either directly learns $\mathcal{O}$ (Mescheder et al., 2019)) or trains a network to predict an intermediate parametric representation (*e.g.* collection of convex primitives (Deng et al., 2019), half-spaces (Chen et al., 2019)). Different from any existing methods, our shape representation is to generate training data for a few-shot classification problem. In order to make the connection clear, we denote the ground-truth $\{\mathbf{x}_i, y_i\}_{i=1}^M$ as $\mathcal{D}^{test}$.

The training data of single image 3D reconstruction are a collection of images $\mathbf{I}_t$ and their corresponding shapes $\mathcal{D}_t^{test}$ which we denote as $\mathcal{D}^{meta-train} = \{\mathbf{I}_t, \mathcal{D}_t^{test}\}_{t=1}^T$. The goal is to learn a network which takes as input an image $\mathbf{I}$ and outputs a functional (predictor) $\mathcal{P}(\mathbf{x})$ which works on $\mathcal{D}^{test}$.

We summarize the notation and the mapping of few-shot classification to 3D shape reconstruction in Table 1. Using the proposed mapping, we need to find a network to convert the input $\mathbf{I}$ to a set of labeled points $\mathcal{D}^{train} = \{\mathbf{x}_i, y_i\}_{N=1}^N$ (usually N is far smaller than M), the $\mathcal{D}^{meta-train}$ can be rewritten as $\{\mathcal{D}_t^{train}, \mathcal{D}_t^{test}\}_{t=1}^T$. It can be seen that, this formulation has a high resemblance to few-shot learning. Also see Figure 2 for a visualization. As a result, we can leverage techniques from the literature of few-shot learning to jointly train the data generation and the classification parts this problem.

3.3 Formulation

Similar to few-shot learning, the problem can be written as a bi-level optimization. The inner optimization is to train the predictor $\mathcal{P}(\mathbf{x})$ to estimate the inside/outside label of a point,

$$\min_{\mathcal{P}} \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{train}} \left[\mathcal{L}\left(y, \mathcal{P}(\mathbf{x})\right)\right], \tag{1}$$

where $\mathcal{L}(\cdot,\cdot)$ is a loss function such as cross entropy and $\mathcal{D}^{train}$ is generated by a network f, $\mathcal{D}^{train} = f(\mathbf{I})$. To reconstruct the shape $(\mathbf{I}, \mathcal{D}^{test})$, the predictor $\mathcal{P}$ should work as an approximation of the indicator $\mathcal{O}$ and is expected to minimize the term,

$$\mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{test}}\left[\mathcal{L}\left(y, \mathcal{P}(\mathbf{x})\right)\right]. \tag{2}$$

The final objective across all shapes (tasks) is

$$\begin{aligned} \min_{\theta} \quad & \mathbb{E}_{(\mathbf{I},\mathcal{D}^{test})\in\mathcal{D}^{meta-train}}\left[\mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{test}}\left[\mathcal{L}\left(y, \mathcal{P}(\mathbf{x})\right)\right]\right], \\ \text{s.t.} \quad & \mathcal{P} = \min_{\mathcal{P}} \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{train}}\left[\mathcal{L}\left(y, \mathcal{P}(\mathbf{x})\right)\right], \quad \mathcal{D}^{train} = f(\mathbf{I}). \end{aligned} \tag{3}$$

In meta learning approaches for few-shot classification, an embedding network is used to map the training samples to an embedding space, $g(\mathbf{x}) = \tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}}$ is the embedding vector of the input $\mathbf{x}$. Similarly, here we also migrate the idea to 3d representation, $g(\mathbf{x}|\mathbf{I}) = \tilde{\mathbf{x}}$, where the embedding network is also conditioned on the task input $\mathbf{I}$. It is equivalent to considering the whole training set of a task in few-shot learning. This setting can also be found in meta-learning (Achille et al., 2019) while most algorithms do not do this. To avoid clutter, we omit the tilde sign of the point embedding $\tilde{\mathbf{x}}$ as $\mathbf{x}$ in later sections.

3.4 Differentiable Predictor

The Eq. 1 is the inner loop of the final objective Eq. 3. Similar to Lee et al. (2019), we choose SVM as the predictor $\mathcal{P}$ for the following reasons: 1) It is well known that SVM has a high generalization ability; 2) SVM can be formulated as quadratic programming which is convex and differentiable with respect to its parameters.

In practice, we use the dual form of kernel SVM,

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{i=1}^{N}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{N}\alpha_i y_i = 0, \quad 0 \le \alpha_i \le C,\ i = 1,\ldots,N, \end{aligned} \tag{4}$$

where $K(\cdot,\cdot)$ is the Gaussian kernel $K(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\frac{\mathbf{x}_i-\mathbf{x}_j}{2\sigma^2})$.

The discriminant function becomes,

$$\mathrm{SVM}(\mathbf{x};\mathcal{D}^{train}) = \sum_{i=1}^{N}\alpha_i y_i K(\mathbf{x}_i,\mathbf{x}) + b, \tag{5}$$

which will be used as our predictor $\mathcal{P}(\mathbf{x})$. Using recent advances in differentiable optimization by Amos & Kolter (2017), the discriminant function is differentiable with respect to each $\mathbf{x}_i$ in $\mathcal{D}^{train}$.

Note that MetaOptSVM (Lee et al., 2019) also use SVM, but we differ in these aspects: 1) MetaOptSVM uses the dual form of SVM without kernels; 2) MetaOptSVM trains SVM on high-dimensional embedding, while we use $\mathbb{R}^3$ embedding.

3.5 Indicator approximation

The SVM predicator $\mathcal{P}(\mathbf{x})$ outputs a positive value if $\mathbf{x}$ is inside the shape otherwise a negative value. So we apply a sigmoid function to convert it to the range $[0,1]$, $\hat{\mathcal{O}} = \mathrm{Sigmoid}(\beta\mathcal{P}(\mathbf{x}))$, where β is a learned scale. Then Eq. 2 is written as follows:

$$\mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{test}}\left[\left\|\hat{\mathcal{O}}(\mathbf{x}) - y\right\|_2^2\right]. \tag{6}$$

4 IMPLEMENTATION

SVM There are two hyperparameters for SVM to be tuned. First, we choose $C = 1$ in all experiments. Second, σ in the kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ is learned during training and is shape-specific, i.e., each shape has its own σ.

In addition to the isotropic Gaussian kernel, we also consider the anisotropic Gaussian kernel,

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-(\mathbf{x}_i - \mathbf{y}_i)^\intercal \mathrm{diag}^{-1}(\boldsymbol{\sigma}^2)(\mathbf{x}_i - \mathbf{y}_i)\right), \tag{7}$$

where $\boldsymbol{\sigma}$ is a vector and $\boldsymbol{\sigma}^2$ represents the element-wise square. Similar to recent works on instance segmentation (Liang et al. (2017); Kendall et al. (2018); Novotny et al. (2018); Zhang & Wonka (2019)), we also find that a simple $\mathbb{R}^3$ spatial embedding works well, i.e., $\mathbf{x} \in \mathbb{R}^3$ and $\boldsymbol{\sigma} \in \mathbb{R}^3$. See Appendix for an explanation of the embedding space.

Networks Our framework is composed of three sub-networks: Feature Network, Point Generator and Embedding Network (see Fig. 1). For the Feature Network, We use the same ResNet18 backbone as in OccNets (Mescheder et al. (2019)) and CvxNets (Deng et al. (2019)) to generate a 256-dimensional feature vector $\boldsymbol{\lambda}$ for an input image. Both the Point Generator and Embedding Network are implemented with MLPs. The Point Generator outputs $\boldsymbol{\sigma}$ and points $\{\mathbf{x}_i\}^N$ (half of which have $+1$ inside label and the other half have -1 outside label) where $N = 32$. The Embedding Network takes as input the concatenation of both the point $\mathbf{x}$ and the feature vector $\boldsymbol{\lambda}$.

Data We perform single image 3d reconstruction on the ShapeNet Chang et al. (2015) dataset. The rendered RGB images and data split are taken from Choy et al. (2016). We sample 100k points uniformly from the shape bounding box as in OccNet (Mescheder et al. (2019)) and also 100k "near-surface" points as in CvxNets (Deng et al. (2019) and SIF (Genova et al. (2019)). Along with the corresponding inside/outside labels, we construct the $\mathcal{D}^{test}$ for each shape offline to increase the training speed. At training time, 1024 points are drawn from the bounding box and 1024 "near-surface".

5 RESULTS

Evaluation metrics We use the volumetric IoU, the Chamfer-L1 distance and F-Score (Tatarchenko et al. (2019)) for evaluation. Volumetric IoU is obtained on 100k uniformly sampled 100k points. The Chamfer-L1 distance is estimated by randomly sampling 100k points from the ground-truth mesh and predicted mesh which is generated by Marching Cubes (Lorensen & Cline (1987)). F-Score is calculated with $d = 2\%$ of the side length of the reconstructed volume. Note that following the discussions by Tatarchenko et al. (2019), F-Score is a more robust and important metric for 3d reconstruction compared to IoU and Chamfer.

Quantitative results We compare our method with a list state-of-the-art methods quantitavely in Table 2. We improve the most important metric, F-score, from 51.75% to 55.91% compared to the previous state of the art OccNet (Mescheder et al. (2019)). We also improve upon OccNet in the two other metrics. According to the L1-Chamfer metric, AtlasNet (Groueix et al. (2018)) has a slight edge, but we would like to reiterate that this metric is less important and we list it mainly for completeness.

Qualitative results We show qualitative reconstruction results in Fig. 3. We also show shape interpolation results in Fig. 4. Interpolation is done by acquiring two features $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ from two different images. Then we reconstruct meshes while linearly interpolating the two features.

Visualization of embedding space In Fig. 5, we first show $\mathcal{D}^{train}$ along with the corresponding meshes. Since our embedding space is also $\mathbb{R}^3$, we can visualize the embedding space. It can be easily seen that, all meshes are transformed to ellipsoid-like shapes in embedding space. That also explains how the embedding network and point generation network collaborate. The embedding network transforms the shape into a simple surface no matter how complicated the shape is. Still, the subtle differences between the ellipsoid like shapes are critical for the performance as verified in tests.

Figure 3: **Top:** ground-truth meshes shown in vermilion red. **Bottom:** predicted meshes shown in blue.

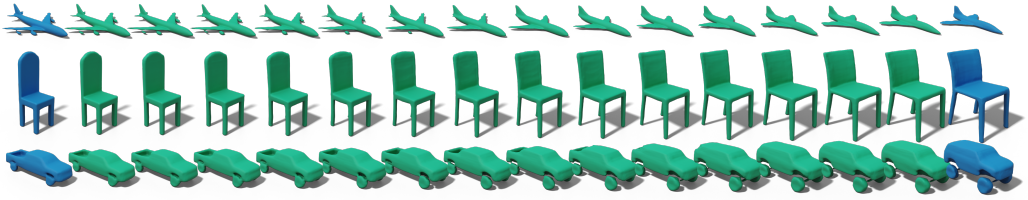


Figure 4: **Reconstruction with interpolated feature vectors.** Meshes in blue are original, green meshes are interpolated.

Ablation study We show the metrics under different hyper-parameter choices in Table 3, including different N and isotropic and anisotropic SVM kernels. We find that increasing N from 16 to 32, generally improves the results, while from 32 to 64, we do no see much performance gain. Further, the results with anisotropic kernel are better than the ones with isotropic kernel in almost all cases.

Table 2: **Reconstruction results on ShapeNet.** We compare our results with P2M (Wang et al. (2018)), AN (Groueix et al. (2018)), OccNet (Mescheder et al. (2019)), SIF (Genova et al. (2019)) and CvxNet (Deng et al. (2019)). Best results are shown in bold.

Category	IoU ↑					Chamfer ↓						F-Score ↑				
	Ours	P2M	OccNet	SIF	CvxNet	Ours	P2M	AN	OccNet	SIF	CvxNet	Ours	AN	OccNet	SIF	CvxNet
airplane	**0.607**	0.420	0.571	0.530	0.598	0.135	0.187	0.104	0.147	0.167	**0.093**	68.08	67.24	62.87	52.81	**68.16**
bench	0.477	0.323	**0.485**	0.333	0.461	0.159	0.201	0.138	0.155	0.261	**0.133**	**64.07**	54.50	56.91	37.31	54.64
cabinet	**0.733**	0.664	**0.733**	0.648	0.709	**0.143**	0.196	0.175	0.167	0.233	0.160	**62.65**	46.43	61.79	31.68	46.09
car	0.733	0.552	**0.737**	0.657	0.675	0.129	0.180	0.141	0.159	0.161	**0.103**	**61.54**	51.51	56.91	37.66	47.33
chair	**0.502**	0.396	0.501	0.389	0.491	0.230	0.265	**0.209**	0.228	0.380	0.337	**44.99**	38.89	42.41	26.90	38.49
display	0.520	0.490	0.471	0.491	**0.576**	0.216	0.239	**0.198**	0.278	0.401	0.223	**43.14**	42.79	38.96	27.22	40.69
lamp	**0.373**	0.323	0.371	0.260	0.311	0.406	0.308	**0.305**	0.479	1.096	0.795	**40.15**	33.04	38.35	20.59	31.41
speaker	**0.647**	0.599	**0.647**	0.577	0.620	0.259	0.285	**0.245**	0.300	0.554	0.462	**43.22**	35.75	42.48	22.42	29.45
rifle	0.508	0.402	0.474	0.463	**0.515**	0.117	0.164	0.115	0.141	0.193	**0.106**	**65.99**	64.22	56.52	53.20	63.74
sofa	0.678	0.613	**0.680**	0.606	0.677	0.174	0.212	0.177	0.194	0.272	**0.164**	**50.75**	43.46	48.62	30.94	42.11
table	**0.519**	0.395	0.506	0.372	0.473	**0.183**	0.218	0.190	0.189	0.454	0.358	**61.79**	44.93	58.49	30.78	48.10
phone	**0.754**	0.661	0.720	0.658	0.719	0.108	0.149	0.128	0.140	0.159	**0.083**	**71.98**	58.85	66.09	45.61	59.64
vessel	0.547	0.397	0.530	0.502	**0.552**	0.199	0.212	**0.151**	0.218	0.208	0.173	48.41	**49.87**	42.37	36.04	45.88
mean	**0.585**	0.480	0.571	0.499	0.567	0.189	0.217	**0.175**	0.215	0.349	0.245	**55.91**	48.58	51.75	34.86	47.36

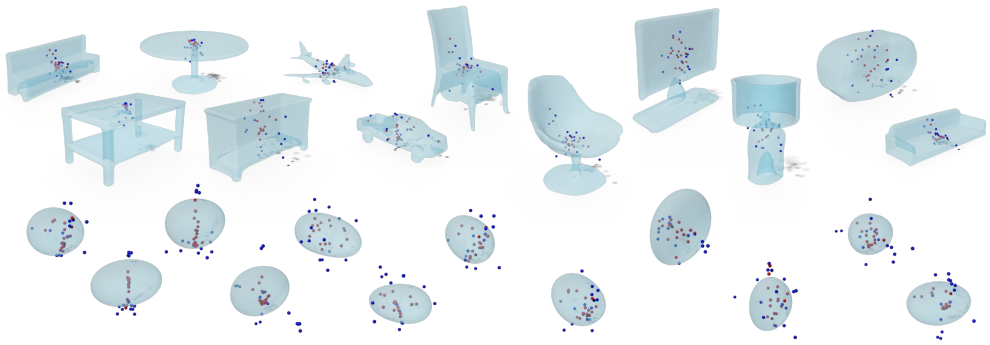


Figure 5: **Top:** $\mathcal{D}^{train}$. **Bottom:** $\mathcal{D}^{train}$ in embedding space. We show positive points in red and negatives points in blue. Also, we visualize the meshes and the meshes in embedding space. These visualizations can be better reviewed in the attached video.

Table 3: **Ablation study.** This table shows results given different N (16, 32 and 64) and the kernel (isotropic or anisotropic). The best model are shown in light gray.

	N	aniso-	sofa	airplane	lamp	phone	vessel	speaker	chair	cabinet	table	display	car	bench	rifle	mean	Δ
IoU ↑	16		0.666	0.596	0.379	0.770	0.511	0.659	0.494	0.718	0.513	0.463	0.740	0.440	0.487	0.572	-0.014
		✓	**0.685**	0.609	0.379	0.774	0.529	0.664	0.505	0.718	0.527	0.475	0.736	**0.483**	0.397	0.575	-0.011
	32		0.680	0.615	0.389	0.780	**0.542**	0.661	0.503	0.722	0.527	0.469	0.740	0.471	0.281	0.568	-0.019
		✓	0.681	**0.621**	0.392	**0.787**	0.530	0.661	**0.508**	0.717	0.524	0.477	0.736	0.474	**0.514**	**0.586**	0.000
	64		0.681	0.550	0.391	0.779	0.537	0.658	0.502	**0.727**	0.528	0.484	**0.745**	0.473	**0.514**	0.582	-0.004
		✓	0.670	0.612	**0.400**	0.779	0.529	**0.668**	0.504	0.721	**0.529**	**0.485**	0.721	0.476	0.512	0.585	-0.001
Chamfer ↓	16		0.179	0.140	0.427	0.110	0.222	0.252	0.255	0.147	0.177	0.261	0.128	0.187	0.206	0.207	0.023
		✓	0.167	0.127	0.375	0.110	0.210	0.248	0.240	0.145	0.171	0.245	0.127	0.144	0.191	0.192	0.008
	32		**0.165**	0.131	0.377	0.105	0.202	0.250	0.238	0.144	0.170	0.252	0.129	0.150	0.730	0.234	0.050
		✓	**0.165**	0.128	**0.369**	**0.102**	0.206	**0.232**	0.236	0.148	0.165	0.242	0.130	0.146	0.130	**0.184**	0.000
	64		0.171	0.186	**0.369**	0.108	**0.193**	0.243	0.240	**0.141**	0.165	0.247	**0.124**	**0.143**	**0.124**	0.189	0.004
		✓	0.173	**0.126**	0.371	0.109	0.195	0.242	**0.232**	0.146	**0.159**	**0.238**	0.141	0.150	**0.124**	0.185	0.001
F-Score ↑	16		49.9	67.9	41.8	71.8	44.4	45.2	43.4	62.0	62.9	39.4	61.8	60.1	57.5	54.5	-2.3
		✓	51.0	69.9	42.2	72.9	46.6	45.7	45.0	61.5	64.8	39.0	63.2	**65.8**	50.7	55.2	-1.5
	32		50.4	69.4	43.4	74.0	47.4	45.0	44.9	62.3	64.0	39.4	62.8	65.0	33.7	54.0	-2.8
		✓	**51.6**	70.0	**43.5**	**76.4**	46.6	**46.3**	**45.7**	60.7	64.3	39.7	61.9	64.3	**66.5**	**56.7**	0.0
	64		51.2	59.7	43.2	76.2	**47.7**	45.2	45.3	**62.5**	65.2	**40.9**	**63.9**	64.6	66.4	56.3	-0.4
		✓	49.4	**70.1**	42.9	74.1	46.8	45.7	45.4	61.5	**65.3**	39.0	57.5	63.5	65.9	55.9	-0.8

6 CONCLUSION

In this paper, we presented a shape representation for deep neural networks. Training data generating networks established a connection between few-shot learning and shape representation by converting a shape into a training set for a supervised task. Training can be solved with meta-learning approaches for few-shot learning. While our solution is inspired by few-shot learning, it is different in: 1) our training datasets are generated by a separate network and not given directly; 2) our embedding network is conditioned on the task but traditional few-shot learning employs unconditional embedding networks; 3) our test dataset is generated by sampling and not directly given. The experiments are evaluated on a single image 3D reconstruction dataset and improve over the SOTA. Our idea is general and can be used in many applications in 3d shape analysis.

REFERENCES

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6430–6439, 2019. 5

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 136–145. JMLR. org, 2017. 2, 5

Luca Bertinetto, João F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=HyxnZh0ct7`. 2

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

Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5939–5948, 2019. 2

Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. *arXiv preprint arXiv:1911.06971*, 2019. 2, 4

Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pp. 628–644. Springer, 2016. 1, 2, 6

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292, 2001. 2

Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnets: Learnable convex decomposition. *arXiv preprint arXiv:1909.05736*, 2019. 2, 4, 6, 7

Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 605–613, 2017. 2

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017. 2

Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7154–7164, 2019. 1, 6, 7

Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016. 2

Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 216–224, 2018. 1, 2, 6, 7

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 1, 2

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7482–7491, 2018. 6

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015. 2

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019. 2, 5

Xiaodan Liang, Liang Lin, Yunchao Wei, Xiaohui Shen, Jianchao Yang, and Shuicheng Yan. Proposal-free network for instance-level object segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2978–2991, 2017. 6

Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. *arXiv preprint arXiv:1706.07036*, 2017. 1, 2

William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 6

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*, pp. 4460–4470, 2019. 1, 2, 4, 6, 7

Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Deep level sets: Implicit surface representations for 3d shape inference. *arXiv preprint arXiv:1901.06802*, 2019. 1, 2

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 2

David Novotny, Samuel Albanie, Diane Larlus, and Andrea Vedaldi. Semi-convolutional operators for instance segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 86–102, 2018. 6

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 165–174, 2019. 1, 2

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017a. 1, 2

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pp. 5099–5108, 2017b. 1, 2

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=rJY0-Kcll`. 2

Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3577–3586, 2017. 1, 2

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pp. 4077–4087, 2017. 2

Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2088–2096, 2017. 1, 2

Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3405–3414, 2019. 6, 13

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pp. 3630–3638, 2016. 2

Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 52–67, 2018. 1, 2, 7

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36 (4):1–11, 2017. 1, 2

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015. 1, 2

Biao Zhang and Peter Wonka. Point cloud instance segmentation using probabilistic embeddings. *arXiv preprint arXiv:1912.00145*, 2019. 6

A Appendix

A.1 Input images

We show input rendered images in Fig. 3.

A.2 Visualization of feature space

We visualize the feature space in two ways: 1) tSNE in Fig. 7; 2) L1-norm statistics in Fig. 8.

A.3 More ablation study results

Additionaly, we show reconstruction results when $N = 4, 8$ and with/without the embedding network g (see 3.3) in Table 4. Besides the conclusion we have made in our main paper, the metrics with embedding network are generally higher than the ones without embedding network.

A.4 Analysis

Here, we analyze how the generated $\mathcal{D}^{train}$ contributes to the reconstruction results. We already know that the the final (decision) surface is decided by $\mathcal{D}^{train}$ and a base learner. Instead of using SVM as the base learner, consider a much simpler learner, k-nearest neighbors with $K = 1$. The label of $\mathbf{x}$ is the same as $(\mathbf{x}_t, y_t) \in \mathcal{D}^{train}$ where $(\mathbf{x}_t, y_t) = \arg\min_{(\mathbf{x}_i, y_i) \in \mathcal{D}^{train}} d(\mathbf{x}, \mathbf{x}_i)$ and $d(\mathbf{x}, \mathbf{x}_i)$ is a distance metric function. In other words, we can find a region (or Voronoi cell),

$$\mathcal{R}_t = \{\mathbf{x} | d(\mathbf{x}, \mathbf{x}_t) \leq d(\mathbf{x}, \mathbf{x}_i), (\mathbf{x}_i, y_i) \in \mathcal{D}^{train} \setminus \{\mathbf{x}_t, y_t\}\}, \tag{8}$$

Figure 6: Input images used in Fig. 3

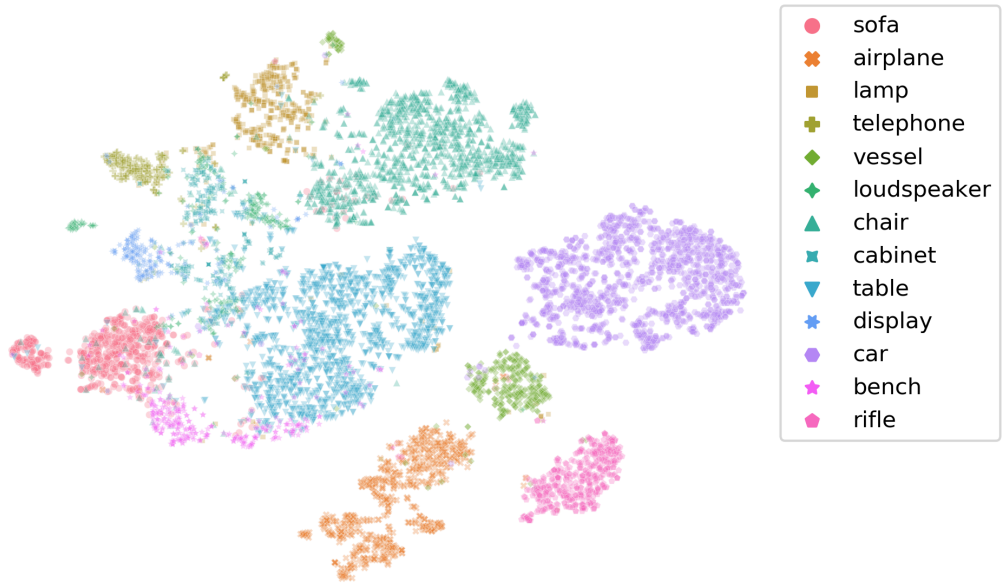


Figure 7: **T-SNE of the feature of shapes.**

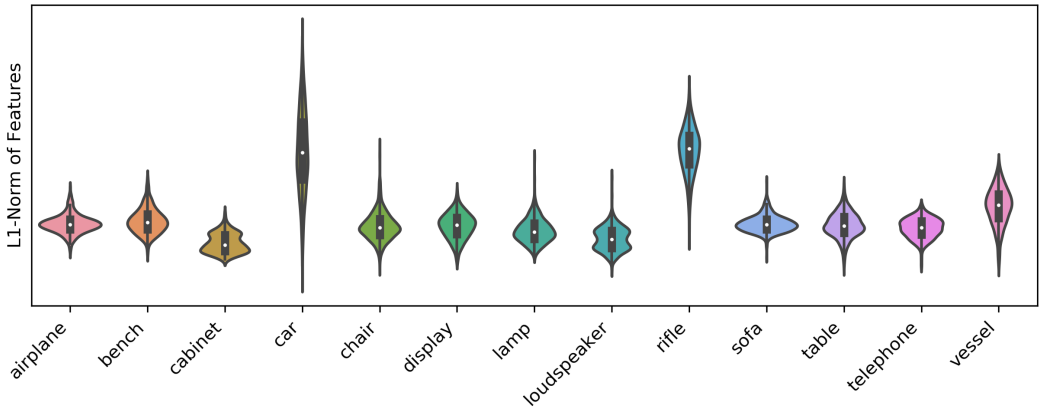


Figure 8: **L1-Norm of features of shapes.**

where for all $\mathbf{x} \in \mathcal{R}_t$ has the same label y_t. Thus each positive point $(\mathbf{x}_i, y_i) \in \mathcal{D}^{train}$ where $y_i = +1$ is responsible to reconstruct a part of the shape. This gives an explanation to $\mathcal{D}^{train}$.

However, SVM is much more complicated. According to Eq. 5, the discriminant function is a linear combination of kernels $K(\mathbf{x}_i, \mathbf{x})$. Therefore, each $(\mathbf{x}_i, y_i) \in \mathcal{D}^{train}$ is responsible for the prediction label of $\mathbf{x}$. A way to find how $\mathcal{D}^{train}$ affects the reconstruction, is to train the SVM with a subset of $\mathcal{D}^{train}$. Although, there are 2^N subsets which makes the analysis nearly impossible. So we use the following greedy process: we sequentially remove a positive point from $\mathcal{D}^{train}$, where each deletion step must make sure the resulting mesh has the best metric (IoU). The partial reconstructed meshes can be seen in Fig. 9. We start with the full set $\mathcal{D}^{train} = \{\mathbf{x_i}, y_i\}_{i=1}^{N}$ where $N = 32$. We only show the first 10 deletion steps. In the first 5 deletion steps, we do not see large changes for the reconstruction, which verifies the robustness of SVM. Then in the last 5 steps, parts are missing gradually. Similarly, we also present the process of removing negative points. In this way, we show a simple relationship between $\mathcal{D}^{train}$ and the final reconstruction.

Table 4: **Ablation study.** This table shows results (volumetric IoU) given different N (4, 8, 16, 32 and 64), the kernel (isotropic or anisotropic) and with/without embedding network.

$N = 4$	✓					✓					✓				
$N = 8$		✓					✓					✓			
$N = 16$			✓					✓					✓		
$N = 32$				✓					✓					✓	
$N = 64$					✓					✓					✓
anistropic	✓	✓	✓	✓	✓						✓	✓	✓	✓	✓
embedding	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓					
IoU	0.545	0.541	0.575	**0.586**	0.585	0.545	0.560	0.572	0.568	0.582	0.443	0.460	0.485	0.507	0.520
Δ	-0.041	-0.045	-0.011	0.000	-0.001	-0.041	-0.026	-0.014	-0.018	-0.004	-0.143	-0.126	-0.101	-0.079	-0.066

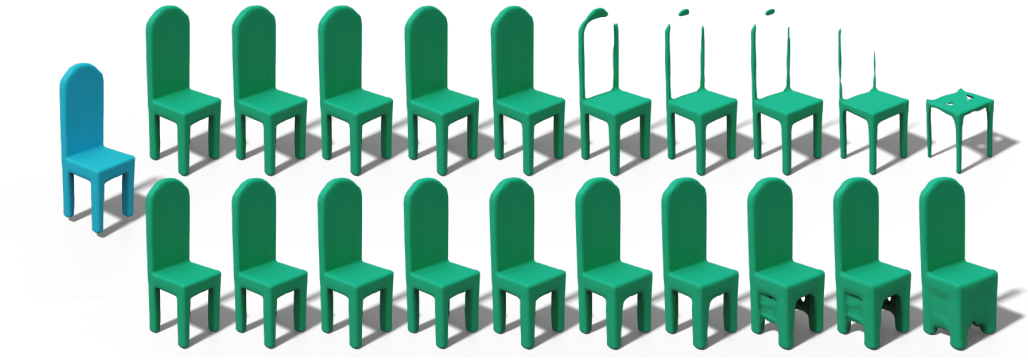
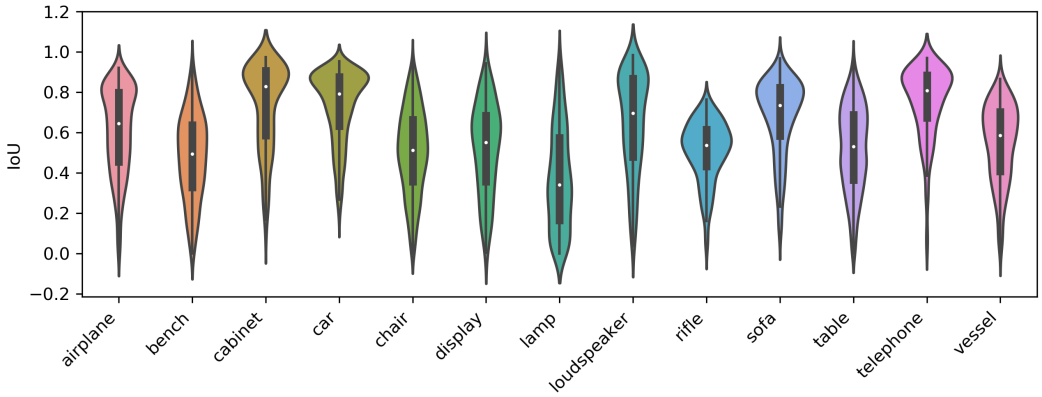


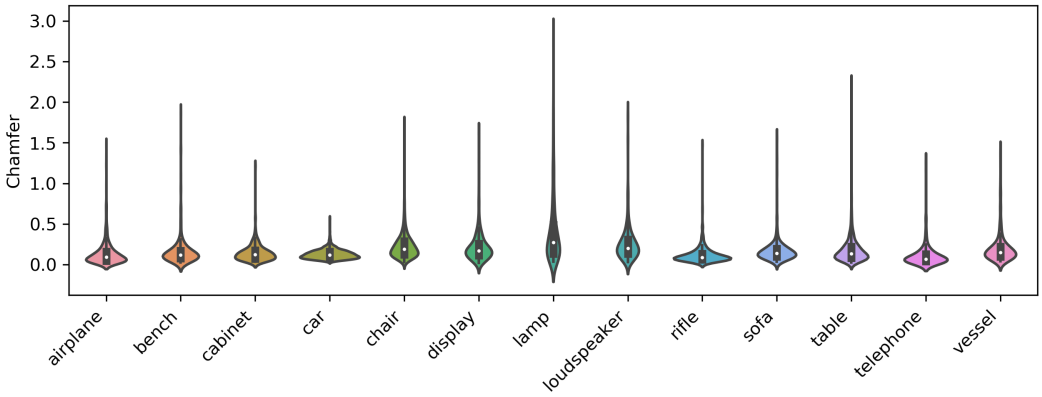
Figure 9: **Reconstruction results with $\mathcal{D}^{train}$. Blue:** full set of $\mathcal{D}^{train}$. **Green:** subset of $\mathcal{D}^{train}$. **Top row:** we sequentially remove positive points from $\mathcal{D}^{train}$ one by one. **Bottom row:** we sequentially remove negative points from $\mathcal{D}^{train}$ one by one.

A.5 STATISTICS OF METRICS

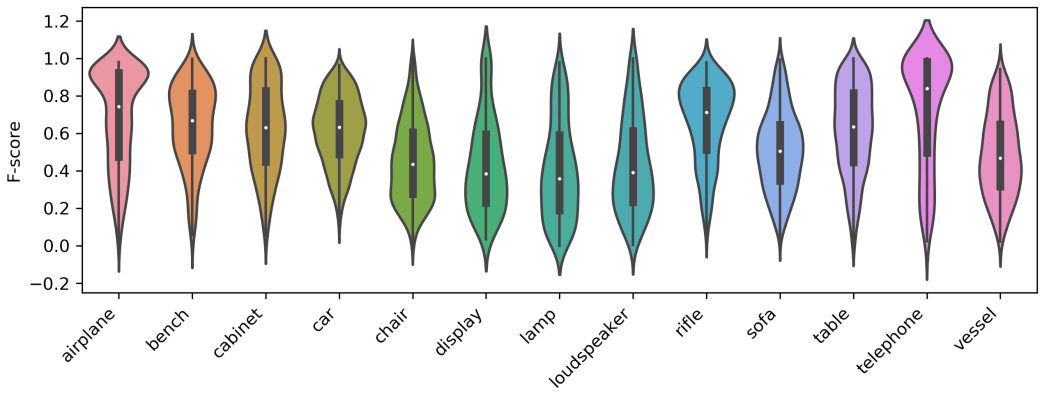
In Table. 2 we have shown the summary of metrics, here we show more detailed statistics in Fig. 10. Comparing to IoU and F-score, most of the values of Chamfer concentrate around the mean value and rare values very far from the mean. The "peakedness" property of Chamfer implies it is unstable to a certain extent. The conclusion is also in consistent with Tatarchenko et al. (2019).



(a) IoU



(b) Chamfer-L1



(c) F-score

Figure 10: Statistics of metric.