Learning Semantic Deformation Flows with 3D Convolutional Networks

M. Ersin Yumer[1(✉)] and Niloy J. Mitra[2]

[1] Adobe Research, San Jose, CA, USA
yumer@adobe.com
[2] University College London, London, UK
n.mitra@cs.ucl.ac.uk

Abstract. Shape deformation requires expert user manipulation even when the object under consideration is in a high fidelity format such as a 3D mesh. It becomes even more complicated if the data is represented as a point set or a depth scan with significant self occlusions. We introduce an end-to-end solution to this tedious process using a volumetric Convolutional Neural Network (CNN) that learns deformation flows in 3D. Our network architectures take the voxelized representation of the shape and a semantic deformation intention (*e.g.,* make more sporty) as input and generate a deformation flow at the output. We show that such deformation flows can be trivially applied to the input shape, resulting in a novel deformed version of the input without losing detail information. Our experiments show that the CNN approach achieves comparable results with state of the art methods when applied to CAD models. When applied to single frame depth scans, and partial/noisy CAD models we achieve ∼60 % less error compared to the state-of-the-art.

1 Introduction

Shape deformation is a core component in 3D content synthesis. This problem has been well studied in graphics where low level, expert user manipulation is required [2,36]. It is acknowledged that this is an open and difficult problem, especially for deformations that follow semantic meaning, where very sparse high level information (*e.g.,* make this shoe more durable) need to be extrapolated to a complex deformation. One way to solve this problem using traditional editing paradigms is through highly customized template matching [44], which does not scale. In this paper, we introduce a novel volumetric CNN, end-to-end trained for learning deformation flows on 3D data, which generalizes well to low fidelity models as well.

CNNs have been shown to outperform hand-crafted features and domain knowledge engineered methods in many fields of computer vision. Promising applications to classification [23], dense segmentation [26], and more recently direct synthesis [8] and transformation [39,43] have been demonstrated.

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-46466-4_18) contains supplementary material, which is available to authorized users.

B. Leibe et al. (Eds.): ECCV 2016, Part VI, LNCS 9910, pp. 294–311, 2016.
DOI: 10.1007/978-3-319-46466-4_18

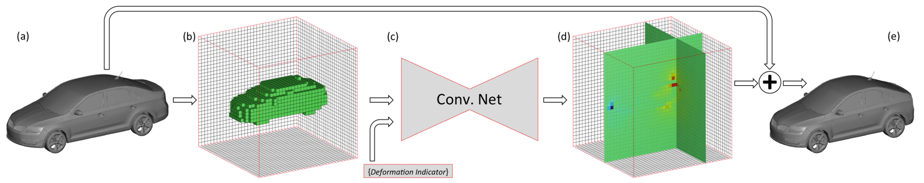

Fig. 1. Our 3D convolutional network (c) takes a volumetric representation (b) of an object (a) and high-level deformation intentions as input and predicts a deformation flow (d) at the output. Applying the predicted deformation flow to the original object yields a high quality novel deformed version (e) that displays the high-level transformation intentions (In this illustration, the car is deformed to be more compact).

Encouraged by these advances, we propose using the reference shape's volumetric representation, and high-level deformation intentions (*e.g.,* make the shape more sporty) as input to our 3D convolutional neural network, where both channels get mixed through fully connected layers and is consequently 'upconvolved'[1] into a volumetric deformation flow at the output. When such a deformation flow is applied to the original reference shape, it yields a deformed version of the shape that displays the high-level deformation intentions (Fig. 1).

We train and test end-to-end networks for four different object categories: cars, shoes, chairs, and airplanes with five high-level relative attribute based deformation controls for each category. In addition to Yumer *et al.* [44]'s dataset (referred to as the SemEd dataset), we also use additional data for the same categories from ShapeNet [3,35]. We use more than 2500 unique shapes for each category, which yields in ∼2.5M training pairs with additional data types (point set and depth scan), as well as data augmentation.

We introduce two novel deformation flow CNN architectures. We compare with state of the art semantic deformation methods, as well as a data-driven baseline and two direct shape synthesis CNN baselines where the output is replaced with volumetric representation of the deformed shape instead of the deformation flow in our architectures. Even though the ultimate goal is generating a deformed version of the shape, we opt to learn a deformation flow instead of directly learning to generate the shape in volumetric format. We show that our deformation flow approach results in ∼70 % less error compared to such direct synthesis approaches using the same CNN architectures. Moreover, we quantitatively and qualitatively show that deformation flow based CNN perform significantly better than the state-of-the-art semantic deformation [44]: we achieve ∼60 % less error on depth scans and noisy/partial CAD models.

Our main contributions are:

– Introducing the first 3D volumetric generative network that learns to predict per-voxel dense 3D deformation flows using explicit high level deformation intentions.

[1] *Upconvolution* in our context is unpooling followed by convolution. Refer to Sect. 3.1 for more details.

– Demonstrating semantic 3D content deformation exploiting structural compatibility between volumetric network grids and free-form shape deformation lattices.

2 Background

3D Deep Learning. 3D ShapeNets [40] introduced 3D deep learning for modeling shapes as volumetrically discretized (*i.e.,* in voxel form) data, and showed that intuitive 3D features can be learned directly in 3D. Song *et al.* [37] introduced an amodal 3D object detector method for RGB-D images using a two 3D convolutional networks both for region proposal and object recognition. Maturana and Scherer demonstrated the use of 3D convolutional networks in object classification of point clouds [30] and landing zone detection [29], specifically from range sensor data. 3D feature extraction using fully connected autoencoders [10,41] and multi-view based CNNs [38] are also actively studied for classification and retrieval. Although volumetric convolution is promising for feature learning, due to the practically achievable resolution of the voxel space prevents high quality object synthesis [40]. We circumvent this by learning a deformation flow instead of learning to generate the transformed object directly. Such deformation flows exhibit considerably less high frequency details compared to the shape itself, and therefore are more suitable to be generated by consecutive convolution and upsampling layers.

Generative Learning. There has been several recent methods introduced to generate or alter objects in images using deep networks. Such methods generally utilize 3D CAD data by applying various transformations to objects in images in order to synthesize controlled training data. Dosovitskiy *et al.* [8] introduced a CNN to generate object images from a particular category (chairs in their case) via controlling variation including 3D properties that affect appearance such as shape and pose. Using a semi-supervised variational autoencoder [20], Kingma *et al.* [19] utilized class labels associated to part of the training data set to achieve visual analogies via controlling the utilized class labels. Similar to the variational autoencoder [20], Kulkarni *et al.* [24] introduced the deep convolutional inverse graphics network, which aims to disentangle the object in the image from viewing transformations such as light variations and depth rotations. Yang *et al.* [43] introduced a recurrent neural network to exploit the fact that content identity and transformations can be separated more naturally by keeping the identity constant across transformation steps. Note that the generative methods mentioned here tackle the problem of separating and/or imposing transformations in the 2D image space. However, such transformations act on the object in 3D, whose representation is naturally volumetric. As the applied transformation gets more severe, the quality and sharpness of the generated 2D image diminishes. On the other hand, our volumetric convolution based deformation flow applies the transformation in 3D, therefore does not directly suffer from the discrepancy between 2D and 3D data.

3D Deformation. 3D shape deformation is an actively studied research area, where many energy formulations that promote smoothness and minimize shear on manifolds have been widely used (see [2] for an extensive review). With the increasing availability of 3D shape repositories, data-driven shape analysis and synthesis methods have been recently receiving a great deal of attention. Mitra *et al.* [31] and Xu *et al.* [42] provide extensive overviews of related techniques. These methods aim to decipher the geometric principles that underlie a product family in order to enable deformers that are customized for individual models, thereby expanding data-driven techniques beyond compositional modeling [12, 44, 46]. Yumer *et al.* [44, 46] present such a method for learning statistical shape deformation handles [45] that enable 3D shape deformation. The problem with such custom deformation handles are two folds: (1) Limited generalization due to dependency on correct registration of handles between template and the model, (2) Being capable to only operate on fully observed data (*e.g.,* complete 3D shapes) and not generalizing well for partially observed data (*e.g.,* depth scans, range sensor output). We circumvent the registration problem by training an end-to-end volumetric convolutional network for learning a volumetric deformation field. We show that our method outperforms the previous methods when the input is partially observed by providing experiments on depth sensor data.

Relative Attributes. We incorporate explicit semantic control of the deformation flow using relative attributes [4, 32, 33, 44]. Relative attributes have been demonstrated useful for high level semantic image search [22], shape assembly [4], and human body shape analysis [1]. Recently, Yumer *et al.* [44] showed that relative attributes can be directly used in a shape editing system to enable semantic deformation (*e.g.,* make this car sportier) using statistical shape deformation handles. We use their system to generate training data with CAD models. We show that our end-to-end method generalizes better compared to [44], especially for low quality, higher variance, and incomplete data (*e.g.,* partial shapes, depth sensor output).

3 Approach

3.1 Network Architectures

Convolutional neural networks are known to perform well in learning input-output relations given sufficient training data. Hence, we are motivated to introduce an end-to-end approach for semantically deforming shapes in 3D (*e.g.,* deform this shoe to be more comfortable). This is especially useful for raw and incomplete data such as depth scans, which previous methods have not addressed. One might think that a complete network to generate the deformed shape at the output of the network is a better solution. While this is a reasonable thought, the resulting shape will be missing high frequency details due to the highest resolution that is achievable with a volumetric network. Results from such a network fail to capture intricate shape details (see Sect. 5 for comparison).

Dense Prediction with CNNs. Krizhevsky *et al.* [23] showed that convolutional neural networks trained with backpropagation [25] perform well for image classification in the wild. This paved the way to recent advancements in computer vision where CNNs have been applied to computer vision problems at large by enabling end-to-end solutions where feature engineering is bypassed. Rather, features are learned implicitly by the network, optimizing for the task and data at hand. Our volumetric convolution approach is similar to CNNs that operate in 2D and generate dense prediction (*i.e.,* per pixel in 2D). To date, such CNNs have been mainly used in semantic segmentation [11,14,26], key point prediction [16], edge detection [13], depth inference [9], optical flow prediction [7], and content generation [8,34,39]. Below, we introduce our 3D convolutional network architecture that derives inspiration from these recent advances in dense prediction approaches.

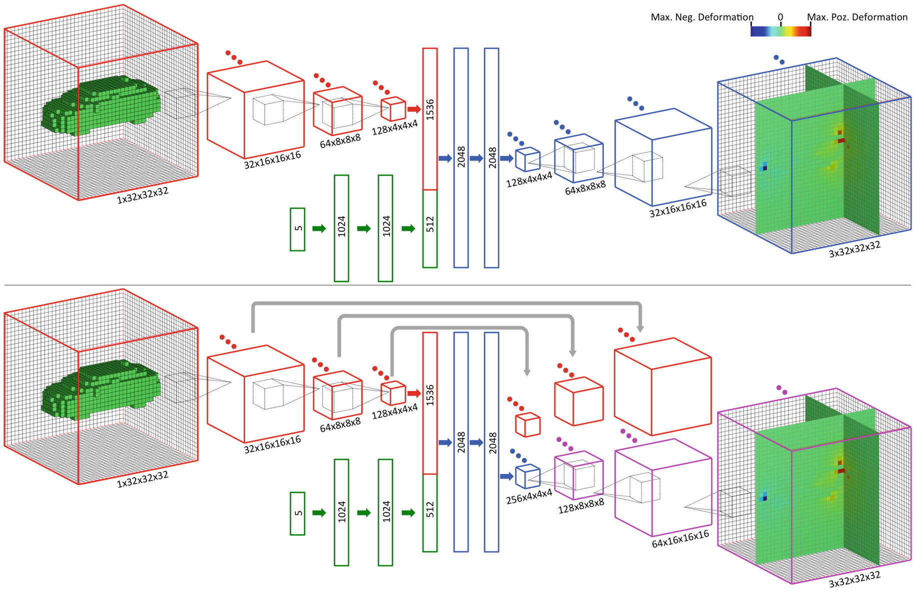

Fig. 2. Top: Volumetric convolutional encoder (**red**)'s third set of filter responses ($128^{*}4 \times 4 \times 4$) are fully connected to a layer of 1536 neurons, which are concatenated with the 512 codes of deformation indicator vector (**green**). After three fully connected layer mixing, convolutional decoder part (**blue**) generates a volumetric deformation flow ($3^{*}32 \times 32 \times 32$). Bottom: We add all filter responses from the encoder part to the decoder part at corresponding levels. (Only the far faces of input - output volume discretization is shown. The deformation flow is computed in the entire volume, where only two slices are shown for visual clarity. Arrows indicate fully connected layers, whereas convolution and upconvolution layers are indicated with appropriate filters.) (Color figure online)

3D Deformation Flow CNN Architecture. We propose two network architectures for learning deformation flows (Fig. 2). Our first network architecture (Fig. 2-top) integrates ideas from Tatarchenko *et al.* [39] where explicit control over transformation parameters (deformation attributes in our case) are fed into the network as a separate input channel. Each element of the input channel demarcates the deformation indicator based on the semantic attribute: 0: generate a deformation flow to decrease this attribute, 1.0: generate a deformation flow to increase this attribute, and 0.5: keep this attribute same. This simpler architecture is easier and faster to train, but fails to capture some of the sharp details in the deformation flow when the structure is volumetrically thin (Fig. 3).

Our second network architecture introduces additional feature maps from the encoder part of the network, as well as upconvolving coarse predictions added to the corresponding resolution layers in the decoder part (analogous to Long *et al.* [26] and Dosovitskiy *et al.* [7] but in 3D). This approach performs better at reconstructing higher frequency details in the deformation flow due to the low level features introduced at corresponding layers. As such, it enables us to perform subtle deformations that are not possible with the first architecture. Figure 3 shows that this architecture captures the shoe sole thickness transformation that corresponds to a 'more durable' deformation.

In the following parts of this paper, we denote the first and second architecture with F1-32 and F2-32. Additionally, we compare with a lower resolution, easier to train version of the networks denoted by F1-16 and F2-16, where 16 denotes the lower volumetric resolution at the input and output ($16 \times 16 \times 16$ instead of $32 \times 32 \times 32$). These low resolution variations are architecturally identical to the ones in Fig. 2 except the fact that the volumetric encoder and the decoder have one less number of layers but same number of convolution filters. For comparison purposes, we also train direct volumetric content synthesis versions of high resolution networks by replacing the deformation flow at the output with the voxelized deformed target shape ($1{*}32 \times 32 \times 32$) and denote these variations as: S1-32 and S2-32.

We use leaky rectified nonlinearities [28] with negative slope of 0.2 after all layers. For both convolution and upconvolution layers, we use $5 \times 5 \times 5$ filters. After each convolution layer we use a $2 \times 2 \times 2$ max pooling layer, whereas

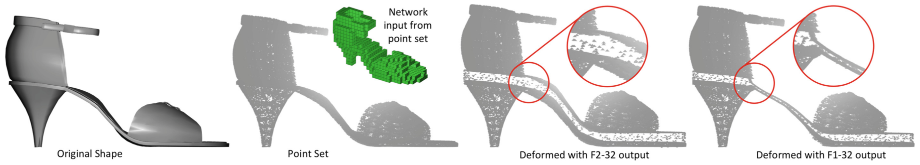
Original Shape
Point Set
Network input from point set
Deformed with F2-32 output
Deformed with F1-32 output

Fig. 3. Our 3D convolutional network takes a volumetric representation of an object ('the point set' in this example) and high-level deformation intentions as input ('durable' in this example) and predicts a deformation flow that can be used to deform the underlying object. Note that the our F2-32 architecture gracefully deforms all parts of the object, whereas the simpler F1-32 might miss the thin regions.

upconvolution layers use an unpooling layer preceding them. Following [8], we simply replace each entry of a feature map with a $2 \times 2 \times 2$ block with entry value at the top left corner and zeros everywhere else. Hence, each upconvolution results in doubled height, width and depth of the feature map. In our second architecture (Fig. 2-bottom), these upconvolved feature maps are concatenated with the corresponding feature maps from the encoder part of the CNN, resulting in doubled the number of feature maps compared to the simpler network illustrated in Fig. 2-top.

3.2 Deformation Flow Computation

Since the volumetric convolution is computed in a regular 3D grid, it conforms naturally to free-form deformation (FFD) using lattices [6,36]. FFD embeds a shape in a lattice space and enables the embedded shape to be deformed using the FFD lattice vertices, which act as control points in the local volumetric deformation coordinate system. The FFD lattice vertices are defined at the voxel centers of the last layer of the CNN (Fig. 4), since the prediction is per voxel.

Formally, the local lattice space for each deformation volume is given by 64 control points, whose position are denoted with $\mathbf{P}_{ijk}$, are the vertices of 27 sub-deformation volumes. Deformed positions of arbitrary points in the center sub-deformation lattice can be directly computed using control point positions:

$$\mathbf{P}(u,v,w) = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} \mathbf{P}_{ijk} B_i(u) B_j(v) B_k(w) \quad \substack{0<u<1 \\ 0<v<1 \\ 0<w<1} \tag{1}$$

where $B_n(x)$ is a Bernstein polynomial of degree n [21], that acts as a blending function. For the sake of completeness, we include a detailed formulation of Bernstein polynomials in our supplementary material.

Since our data is in the form of *undeformed-deformed* shape or point set pairs, we first compute a binary voxel mask for the *undeformed* shape as network input, and a deformation flow for each pair as network output for the training dataset. To compute the deformation flow, we solve the following optimization problem to compute the deformed lattice vertex positions for the input-output pairs:

$$\underset{\mathbf{d}' \in \mathcal{D}}{\arg\min} \quad \sum_i l(\mathbf{p}'_i - \mathbf{F}(\mathbf{p}_i)). \tag{2}$$

where $\mathbf{p}'_i$ and $\mathbf{p}_i$ are the *deformed* and *undeformed* positions of points in the shape or point set data, $\mathbf{d}'$ is the deformation lattice vertex position in the deformed state, and $\mathcal{D}$ is the set of all deformation lattice vertices. $\mathbf{F}$ is the FFD deformation flow operator applied on the undeformed positions using Eq. 1. The deformation lattice vertices are the voxel centers in the network output (Fig. 4). Hence, the deformation flow vector in $\mathbb{R}^3$ for each voxel is given by $\mathbf{v} = \mathbf{d}' - \mathbf{d}$, where $\mathbf{d}$ is the undeformed lattice vertex position.

One can argue that instead of computing the deformation flow a priori and using an Euclidean loss on the dense deformation flow as we do, an alternative is to deform the input shape using the deformation flow at each forward pass, and compute an Euclidean loss over points in the deformed positions of the shape or point cloud. The problem with such an optimization is that only a sparse number of voxels contribute to the deformation. In our approach, the non-contributing voxel values are set to zero to enforces correct dense prediction. We experimented with both, and observed that dense deformation flow Euclidean loss resulted in ∼5x faster convergence, without any performance difference on test sets.

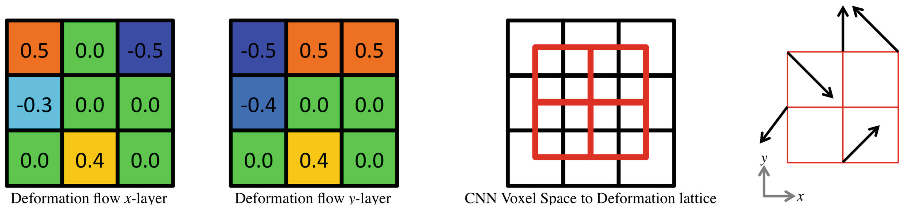

Fig. 4. CNN voxels to FFD lattice illustrated with a 2D example for clarity. The flow values predicted for each voxel (left), correspond to the deformation lattice vertices (middle), which result in the deformation vectors applied to the control points (right).

Deformation of the Shape using the Predicted Flow. At test time, we use the trained network to predict a volumetric deformation flow for the input shape or point set. We apply the flow to the deformation lattice vertices where the input is embedded, which results in the final deformed shape/range scan.

4 Implementation and Training Details

4.1 Training Data Generation

We build eight datasets (four object classes × two data types) from two sources: (1) ShapeNet [3], (2) SemEd [44]. We train and test on three different data types: 3D shapes in mesh representation, point sets sampled directly on the shape, and simulated single frame depth scans from a kinect sensor [15] in point cloud representation. The four object classes we include are: Cars, Shoes, Chairs, Airplanes. Note that a network is trained per object class.

3D Shapes. We collect 2500 shapes from ShapeNet for each category[2] in addition to the data provided by Yumer *et al.* [44] for each category. We randomly separate ∼20 % from each group to be used as tests.

Point Sets and Simulated Depth Scans. One of the powerful aspects of our method is that, it can gracefully handle low fidelity and incomplete data. We achieve this by sampling points on the shape set and also simulating single view depth scans for the shapes mentioned above from arbitrary viewpoints

[2] We collect additional data from 3D Warehouse where ShapeNet counts fall short.

and include these in the training set as well. Note that for the simulated scans, only the input of the training pair changes such that the network is trained to predict complete deformation flow from single viewpoint depth scans in order to correctly deform the points.

Undeformed-Deformed Pairs. We generate deformed shapes using the method introduced by Yumer *et al.* [44]. Their approach semantically deforms shapes by first fitting a class dependent template to the shape, and subsequently deforming the shape using the template whose degrees of freedom (DOF) are linked to semantic attribute controllers. Semantic controller - template DOF mapping is learned through user studies: they provide five different semantic controllers for four shape categories, all of which are utilized in our networks. Yumer *et al.* [44] deform shapes at a continuous scale. We opt to divide their range into five *severity* steps, and use each successive step as a training pair. In our experiments we observed that this yielded a better performance allowing the network to learn a less severe deformation field. A shape can be passed through the network multiple times by using the output from the previous step as the input, resulting in more severe deformation (Fig. 5). Note that, their dependency on customized deformation templates and correct registration of template labels for successful deformation limits the variety of shapes they can operate on. Their system will mostly fail to fit correct templates to depth scans and point sets (refer to Sect. 5 for details of such experiments). We therefore utilize the following strategy to generate the deformed counter part for point sets and depth scans: (1) register the nearest point on the shape for each point in the simulated depth scan, (2) deform the original shape with Yumer *et al.* [44], (3) transform each point in the depth scan to a deformed position by using the relative distance to the corresponding nearest point on the shape.

Data Augmentation. We utilize two data augmentations to increase the robustness of the network, and to provide additional training data: (1)part removal from input, (2) translation and rotation transforms. Note that part removal only applied to the input (*i.e.,* the network is trained to predict the full deformation flow), whereas transformations are applied to both the input and the output (*i.e.,* the network is expected to predict the flow both at globally and locally correct positions relative to the shape data) We assume that the object up direction is known, which is available in both data sources we utilize, and easy to obtain for shapes where such information is not available a priori. Therefore, the rotation transformation applies only to in-plane rotation on the ground.

Training Data Statistics. Table 1 show detailed statistics of the datasets used both in training. Although the number of shapes in each category we use in training is ∼2000, this quickly multiplies: each category has 5 different semantic deformation modes, and 5 different severity steps, yielding 25 input-output pairs for each shape. Moreover, we simulated depth scans from 6 arbitrary views for each shape, and apply 7 random data augmentations. This results in ∼2.5 million input-output training pairs per object category.

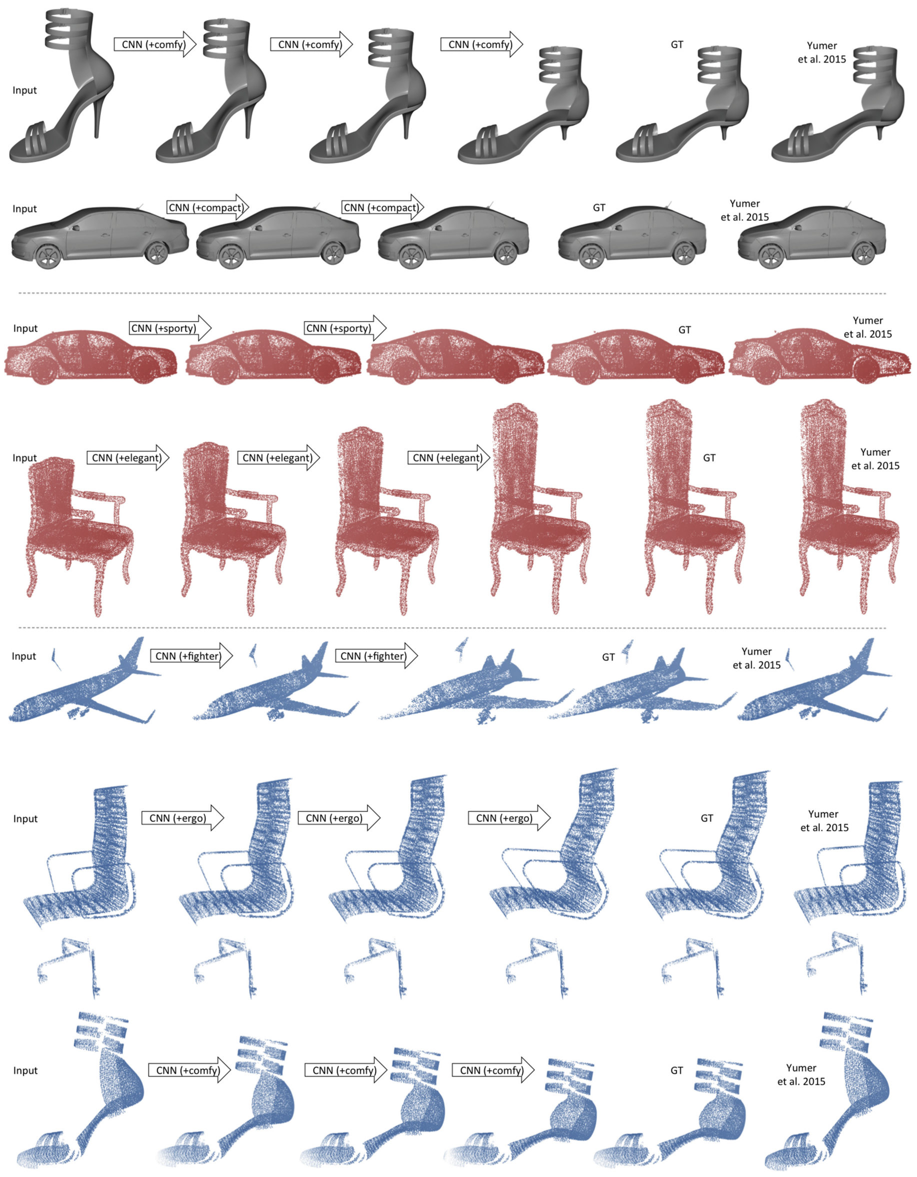

Fig. 5. Results using our finetuned networks (F2-32{-fm, -fp, -fs}) for 3D shapes in mesh representation (grey), point sets (red), and depth scan data (blue). Note that the shape is processed through the network a few number of times to achieve the desired deformation effect. Comparisons with Yumer *et al.* [44] show that our method achieves similar results for high quality mesh data. On the other hand, both for point sets and depth scans, our method outperforms Yumer *et al.* [44] significantly. For additional visual comparisons, please refer to our supplementary material. (Color figure online)

Table 1. Dataset statistics. Although the number of models in SemEd [44] is smaller, we include them to compare with Yumer *et al.* [44] under more challenging conditions as well (*e.g.,* train on ShapeNet – test on SemEd).

	Cars		**Shoes**		**Chairs**		**Airplanes**	
	ShapeNet	SemEd	ShapeNet	SemEd	ShapeNet	SemEd	ShapeNet	SemEd
All Shapes	2500	131	2500	127	2500	61	2500	53
Training	2000	100	2000	100	2000	45	2000	40
+ Depth Scan	14700		14700		14315		14280	
× Attributes	73500		73500		71575		71400	
× Def. Severity Steps	$\sim 350 \times 10^3$		$\sim 350 \times 10^3$		$\sim 350 \times 10^3$		$\sim 350 \times 10^3$	
× Data Augmentation	$\sim 2.5 \times 10^6$		$\sim 2.5 \times 10^6$		$\sim 2.5 \times 10^6$		$\sim 2.5 \times 10^6$	

4.2 Network Training

Training Procedure. We train the networks using the Torch7 framework [5]. We initialize the weights from a normal distribution: $\mathcal{N}(\mu = 0, \sigma^2 = 0.015)$. We utilize Adam [18] optimizer using a mini-batch stochastic gradient descent (with a mini-batch size of 128). Kingma *et al.* [18] proposed momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and a learning rate of 0.001. However, we found out that the learning rate was too high and used 0.0002 instead. Together with reducing the first momentum term β_1 to 0.5 as suggested by [34] resulted in a more stable and repeatable training procedure. We also used batch normalization [17], which proved to be very useful in training the deeper versions of the network ($32 \times 32 \times 32$ resolution at input and output). We use 10 % of the training data for validation, to monitor and prevent overfitting.

Finetuning. Note that the target input data types we want to the method to handle are significantly different: (1) complete 3D mesh models, (2) point sets, (3) depth scans. We therefore finetune the networks on these three data types separately. We use corresponding subset of the training data for each data type within our training set (Table 1), with a low learning rate (1×10^{-6}). We denote the finetuned networks with '-fm', '-fp', and '-fs', which correspond to finetuned on mesh data, point set data, and scan data, respectively.

5 Results

5.1 Experiments

Each shape category (Cars, Shoes, Chairs and Airplanes) database have five different associated attributes given in Table 2. We choose these four object types, and corresponding attributes due to the fact that we can generate ground truth data for comparison and training using Yumer *et al.* 's method [44]. We train the multiple network architectures as introduced in Sect. 3.1 for these four shape categories: two low resolution deformation flow networks (F1-16, F2-16), two deformation flow networks (F1-32, F2-32), and an additional data type specific finetuned version (F2-32{-fm, -fp, -fs}).

Baselines. In addition to comparing our method with Yumer *et al.* [44], we also introduce three baseline methods which we compare with our deformation

Table 2. Mesh deformation error [voxelized space edge length $\times 10^{-2}$]. In each row: **Lowest error - best performance**. **Highest error**.

	[44]	*k*NN	S1-32	S2-32	F1-16	F2-16	F1-32	F2-32	F2-32-fm
Car-luxurious	**0.0**	7.43	**9.45**	8.10	4.54	3.72	1.93	1.13	0.82
Car-sporty	**0.0**	10.4	**15.8**	12.0	7.02	5.45	3.76	2.65	1.83
Car-compact	**0.0**	11.6	**13.7**	11.6	6.52	5.82	3.54	2.49	1.79
Car-muscular	**0.0**	6.98	**9.21**	7.90	4.28	3.26	1.87	1.07	0.77
Car-modern	**0.0**	6.43	**10.2**	8.67	5.78	4.23	2.04	1.02	0.79
Shoe-fashionable	**0.0**	9.64	**15.4**	14.2	7.43	7.21	4.32	3.24	3.20
Shoe-durable	**0.0**	4.28	**4.71**	4.28	4.78	4.34	0.23	0.21	0.19
Shoe-comfy	**0.0**	7.45	**13.5**	12.5	6.93	6.87	3.97	3.10	2.45
Shoe-feminine	**0.0**	10.4	**15.3**	13.2	8.29	7.64	4.81	4.06	3.50
Shoe-active	**0.0**	7.43	**14.2**	13.8	7.26	7.12	4.20	3.57	3.07
Chair-comfy	**0.0**	8.43	**9.12**	8.10	7.05	6.18	4.15	3.20	2.01
Chair-ergonomic	**0.0**	8.02	**10.6**	9.12	7.43	6.01	3.28	3.01	1.98
Chair-elegant	**0.0**	**9.11**	9.10	8.52	6.18	5.43	3.05	2.29	2.00
Chair-antique	**0.0**	0.24	**9.58**	8.54	0.45	0.49	0.10	0.08	0.10
Chair-sturdy	**0.0**	4.10	**4.51**	4.24	4.80	3.34	2.30	2.20	1.49
Airplane-fighter	**0.0**	**14.3**	14.2	13.8	10.5	9.54	6.32	5.90	5.06
Airplane-fast	**0.0**	**16.5**	13.4	12.6	9.67	8.75	5.71	4.88	4.27
Airplane-stealth	**0.0**	7.84	**10.6**	9.47	8.43	7.86	5.61	4.83	3.30
Airplane-sleek	**0.0**	8.59	**9.41**	9.02	7.40	7.02	6.04	5.31	3.29
Airplane-civilian	**0.0**	12.7	**15.3**	11.6	6.73	6.12	5.23	5.47	4.01

flow based approach: two direct synthesis network baselines (S1-32, S2-32), and a nearest neighbor baseline (demarcated as *k*NN in results Tables 2, 3, and 4). The synthesis baselines replace the last layer of the networks with voxelized representation of the deformed model instead of the deformation flows. We then use marching cubes [27] for reconstructing a surface mesh representation for the deformed shape. For the *k*NN baseline, we build three databases each registering training inputs for one data type (mesh, point set, depth scan) to the corresponding deformation flows. We include all rotation and translation augmentations introduced in Sect. 4. At test time, we find the k nearest neighbors to the input using average nearest point Euclidean distance (for meshes we use randomly sampled points on the shape), blend the corresponding deformation flows of the k neighbors proportionally weighted by their inverse distance to the input. We experimented with various k between 3 and 25, however we report our results using $k = 15$ which performed best. More details on the *k*NN and synthesis CNN baselines can be found in our supplementary material.

Table 3. Point set deformation test error [voxelized space edge length $\times 10^{-2}$]. In each row: **Lowest error - best performance**. **Highest error**.

	[44]	*k*NN	S1-32	S2-32	F1-16	F2-16	F1-32	F2-32	F2-32-fp
Car-luxurious	2.12	7.43	**9.88**	8.27	4.50	3.70	1.86	1.17	**0.76**
Car-sporty	1.98	10.4	**13.9**	11.6	7.43	5.60	3.68	2.59	**1.90**
Car-compact	**0.96**	11.6	**14.1**	12.0	6.60	5.62	3.62	2.32	1.84
Car-muscular	1.24	6.98	**9.76**	7.63	4.74	3.31	1.97	1.11	**0.72**
Car-modern	1.57	6.43	**10.9**	8.75	5.75	4.10	2.43	1.18	**0.75**
Shoe-fashionable	**2.23**	9.64	14.3	**14.8**	7.81	7.93	4.51	2.99	2.60
Shoe-durable	0.31	4.28	4.40	4.44	**4.76**	4.55	0.22	0.25	**0.21**
Shoe-comfortable	2.76	7.45	**13.1**	12.2	6.63	6.25	4.24	3.42	**2.33**
Shoe-feminine	**3.20**	10.4	**15.8**	12.9	8.97	7.70	4.19	4.64	3.28
Shoe-active	**3.25**	7.43	**13.1**	13.0	7.76	7.25	4.01	3.70	3.36
Chair-comfortable	2.54	8.43	**9.77**	8.35	7.24	6.21	4.28	3.47	**2.24**
Chair-ergonomic	2.36	8.02	**10.2**	8.93	7.67	6.43	3.65	3.23	**2.10**
Chair-elegant	2.71	9.11	**9.65**	8.80	6.34	5.22	3.69	2.75	**2.54**
Chair-antique	**0.06**	0.24	**9.90**	8.71	0.66	0.60	0.12	0.09	0.12
Chair-sturdy	1.32	4.10	**4.75**	4.41	4.48	3.12	2.37	2.25	**1.25**
Airplane-fighter	5.67	**14.3**	13.4	13.2	11.4	9.70	6.20	5.64	**4.96**
Airplane-fast	5.21	**16.5**	15.7	12.4	9.07	8.72	5.67	4.70	**4.41**
Airplane-stealth	3.78	7.84	**11.3**	9.76	8.86	7.89	5.74	4.65	**3.20**
Airplane-sleek	4.02	8.59	**9.73**	9.47	7.61	6.90	6.23	5.10	**3.68**
Airplane-civilian	4.85	12.7	**14.6**	11.2	6.40	6.34	5.53	5.20	**4.36**

Ground Truth. As mentioned in Sect. 4, Yumer *et al.* [44] cannot deform the shape when their deformation template does not register correctly. This is a major problem for point sets and depth scans. hence we generate the deformed counter part for these input types by using the underlying 3D shape mesh: (1) register the nearest point on the shape for each point in the simulated depth scan, (2) deform the original shape with Yumer *et al.* [44], (3) transform each point in the depth scan to a deformed position by using the relative distance to the corresponding nearest point on the shape. The point sets are sampled directly on the shape, whereas depth scans are simulated by using a virtual kinect sensor [15].

Deformation Test Error. We compute the test error relative to the network input 3D volume edge length. We report average Euclidean error comparing point positions with respect to the ground truth (for the mesh representation we compute the error using randomly sampled points on the mesh since mesh vertices might be sparse and inconsistently distributed). Tables 2, 3, and 4 show all results for the mesh, point set, and depth scan data, respectively.

Table 4. Depth scan deformation test error [voxelized space edge length $\times 10^{-2}$]. In each row: **Lowest error - best performance**. **Highest error**.

	[44]	kNN	S1-32	S2-32	F1-16	F2-16	F1-32	F2-32	F2-32-fs
Car-luxurious	**13.5**	11.2	10.2	9.70	4.99	4.23	2.34	1.40	**1.18**
Car-sporty	**15.4**	14.5	14.2	12.7	7.86	6.25	4.02	2.66	**2.04**
Car-compact	**19.6**	15.3	14.6	13.1	7.21	6.09	3.94	2.89	**2.01**
Car-muscular	**15.0**	11.6	10.5	8.50	5.19	4.28	2.40	1.50	**0.95**
Car-modern	**12.7**	9.30	11.4	9.42	5.82	5.09	2.40	1.36	**0.98**
Shoe-fashionable	**14.8**	11.2	14.9	13.5	7.94	8.51	4.87	3.45	**3.08**
Shoe-durable	0.52	**4.99**	4.80	4.67	4.81	4.42	0.32	**0.20**	**0.20**
Shoe-comfortable	**16.4**	8.86	13.2	11.5	6.92	6.75	4.41	3.87	**2.86**
Shoe-feminine	11.9	12.8	**15.6**	12.7	9.44	7.98	4.53	4.84	**3.70**
Shoe-active	**13.5**	9.30	12.6	12.4	8.03	7.59	4.28	3.89	**3.51**
Chair-comfortable	**11.5**	9.97	10.8	9.51	9.30	6.70	4.60	4.06	**3.07**
Chair-ergonomic	10.6	9.75	**11.6**	9.03	8.57	6.87	3.41	3.74	**2.74**
Chair-elegant	9.05	10.2	**12.0**	8.75	7.36	5.98	3.87	**3.12**	3.18
Chair-antique	**0.09**	0.21	**8.52**	8.40	0.98	0.56	0.29	0.14	0.16
Chair-sturdy	**10.8**	5.40	6.10	5.69	5.33	4.10	3.03	2.89	**1.90**
Airplane-fighter	**18.7**	13.5	14.1	12.7	12.5	10.5	6.93	5.33	**5.23**
Airplane-fast	14.1	15.6	**15.8**	12.0	9.96	9.82	6.26	4.94	**4.78**
Airplane-stealth	**15.2**	9.78	12.0	10.6	9.50	8.90	5.91	4.47	**3.45**
Airplane-sleek	**13.6**	10.4	11.4	11.0	7.93	7.54	6.98	5.39	**3.79**
Airplane-civilian	12.4	**14.1**	12.5	12.5	8.05	7.02	6.68	5.61	**4.52**

5.2 Discussions

We present average deformation error results on all test datasets in Tables 2, 3, and 4 for mesh, point set, and depth scan data separately. Note that these tables show aggregated error from both ShapeNet and SemEd (Table 1 (shapes that were not used for training)). Specifically, Cars: 500 ShapeNet + 31 SemEd, Shoes: 500 ShapeNet + 27 SemEd, Chairs: 500 ShapeNet + 16 SemEd, and Airplanes: 500 ShapeNet + 13 SemEd mesh models were used in the tests. Point sets also used the same models resulting in the same number of test subjects, whereas depth scans used twice the number of input by generating test date from two randomly chosen sensor locations around each shape. We challenge our method by training only on ShapeNet dataset and testing on SemEd (this is more challenging because the previous work we compare with [44] is trained on SemEd and utilizes a mixture of experts approach). We present detailed comparison of this additional experiment in our supplementary material, where our method outperformed the comparison similarly as in Tables 2, 3, and 4.

For the mesh representation deformation (Table 2), Yumer *et al.* 's [44] results are null, since their method is used to compute ground truth, and complete shapes are the ideal case. Our finetuned network (F2-32-fm) performs best among the convolutional network approaches, and outperforms the kNN baseline as well. Note that, the synthesis network baselines (S1-32, S2-32) are not as good as the kNN baseline in most cases. Note also that, kNN results for point sets are the same with kNN for mesh data since kNN for mesh also uses the points sampled on the shape as described in Sect. 5.1.

When there is a considerable amount of uncertainty is existent in the input (such as point set or depth scan input type), our method outperforms the results of [44] (Tables 3, and 4). Moreover, since depth scans incorporate a significant amount of missing data (Fig. 5), Yumer *et al.* [44] is not able to generate plausible deformations due to their template matching step failing to fit correspondences.

Figure 5 shows visual results for all three data types. Our method achieves similar results to that of Yumer *et al.* [44] with respect to the ground truth for the mesh data type. Note the significant loss of deformation quality with the previous work for point set and depth scan data. Our method gracefully generates deformation flows that meaningfully deform such low fidelity data. Table 4 shows that deformation using our best performing network (F2-32-fs) achieves $\sim 5\times$ less error compared to the four benchmark methods (Yumer *et al.* [44], kNN, S1-32, S2-32). Refer to our supplemental material for visual comparisons with kNN and synthesis network baselines.

Robustness to Noise and Partial Data. We further test our method by introducing noise and partially missing parts to the input (Fig. 6(a–c)). To add noise, we randomly select a point on the surface from a uniform distribution and generate a sphere that has a radius drawn from $\mathcal{N}(d/20, d/100)$, where d is the diagonal of the shape bounding box. The number of spheres added affects the amount of noise, which is measured in added voxel percentage in the voxelized representation of the shape. For introducing missing parts (Fig. 6(b)), we randomly select a point on the surface from a uniform distribution and remove polygons from the shape to match the percentage of missing voxels required. Figure 6(d–e) show the results of this experiment on 1000 randomly selected shapes from our test data, with 1000 randomly selected deformation indications from a uniform distribution. Figure 6 shows that the CNN methods are more robust to noise in general, with our flow based networks outperforming other methods including the synthesis networks.

Limitations. Our deformation flow is continuous in the voxel grid, and this introduces some limitations when the optimal deformation can only be achieved with a discontinuous flow. An example of this can be observed in the airplanes dataset with the 'fighter' deformation indicator. The airplane wings stretch significantly whereas the sides of the fuselage where the wings attach does not stretch with it in the original deformation. Our deformation for such extreme discontinuity requirements is not as good as globally continuous cases as seen in Table 2.

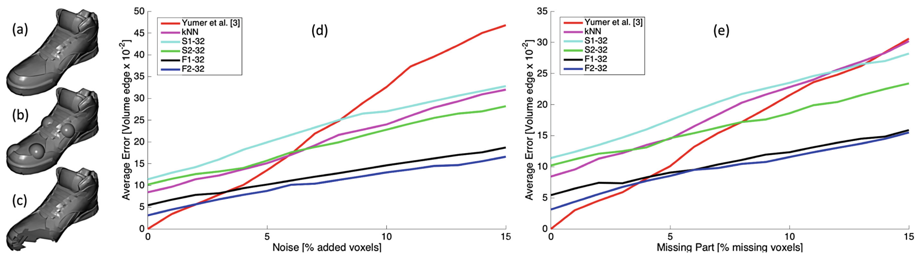

Fig. 6. Right: Example augmentation on test data: (a) original shape, (b) noise, (c) missing parts. (d) Noise vs. Average test error, (e) Missing data vs. Average test error. Both noise and missing data percentages are measured with respect to the original input's voxelized representation number of filled voxels.

References

1. Allen, B., Curless, B., Popović, Z.: The space of human body shapes: reconstruction and parameterization from range scans. ACM Trans. Graph. **22**, 587–594 (2003)
2. Botsch, M., Sorkine, O.: On linear variational surface deformation methods. IEEE TVCG **14**(1), 213–230 (2008)
3. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: an information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
4. Chaudhuri, S., Kalogerakis, E., Giguere, S., Funkhouser, T.: Attribit: content creation with semantic attributes. In: Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology, pp. 193–202. ACM (2013)
5. Collobert, R., Kavukcuoglu, K., Farabet, C.: Torch7: a matlab-like environment for machine learning. In: BigLearn, NIPS Workshop. No. EPFL-CONF-192376 (2011)
6. Coquillart, S.: Extended free-form deformation: a sculpturing tool for 3D geometric modeling. ACM SIGGRAPH Comput. Graph. **24**, 187–196 (1990). ACM
7. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: Flownet: learning optical flow with convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2758–2766 (2015)
8. Dosovitskiy, A., Tobias Springenberg, J., Brox, T.: Learning to generate chairs with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1538–1546 (2015)
9. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: Advances in Neural Information Processing Systems, pp. 2366–2374 (2014)
10. Fang, Y., Xie, J., Dai, G., Wang, M., Zhu, F., Xu, T., Wong, E.: 3d deep shape descriptor. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2319–2328 (2015)
11. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1915–1929 (2013)
12. Fish, N., Averkiou, M., Van Kaick, O., Sorkine-Hornung, O., Cohen-Or, D., Mitra, N.J.: Meta-representation of shape families. ACM Trans. Graph. **33**(4), 34–1 (2014)

13. Ganin, Y., Lempitsky, V.: N^4-Fields: neural network nearest neighbor fields for image transforms. In: Cremers, D., Reid, I., Saito, H., Yang, M.-H. (eds.) ACCV 2014. LNCS, vol. 9004, pp. 536–551. Springer, Heidelberg (2015). doi:10.1007/978-3-319-16808-1_36
14. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)
15. Gschwandtner, M., Kwitt, R., Uhl, A., Pree, W.: BlenSor: blender sensor simulation toolbox. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Wang, S., Kyungnam, K., Benes, B., Moreland, K., Borst, C., DiVerdi, S., Yi-Jen, C., Ming, J. (eds.) ISVC 2011. LNCS, vol. 6939, pp. 199–208. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24031-7_20
16. Hariharan, B., Arbeláez, P., Girshick, R., Malik, J.: Hypercolumns for object segmentation and fine-grained localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 447–456 (2015)
17. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
18. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
19. Kingma, D.P., Mohamed, S., Rezende, D.J., Welling, M.: Semi-supervised learning with deep generative models. In: Advances in Neural Information Processing Systems, pp. 3581–3589 (2014)
20. Kingma, D.P., Welling, M.: Stochastic gradient vb and the variational autoencoder. In: ICLR (2014)
21. Korovkin, P.: Bernstein polynomials. In: Hazewinkel, M. (ed.) Encyclopedia of Mathematics. Springer (2001). ISBN: 978-1-55608-010-4
22. Kovashka, A., Parikh, D., Grauman, K.: Whittlesearch: image search with relative attribute feedback. In: IEEE CVPR, pp. 2973–2980 (2012)
23. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
24. Kulkarni, T.D., Whitney, W.F., Kohli, P., Tenenbaum, J.: Deep convolutional inverse graphics network. In: Advances in Neural Information Processing Systems, pp. 2530–2538 (2015)
25. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4), 541–551 (1989)
26. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
27. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3d surface construction algorithm. In: ACM SIGGRAPH Computer Graphics, vol. 21, pp. 163–169. ACM (1987)
28. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of ICML, vol. 30, p. 1 (2013)
29. Maturana, D., Scherer, S.: 3d convolutional neural networks for landing zone detection from lidar. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 3471–3478. IEEE (2015)
30. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 922–928. IEEE (2015)

31. Mitra, N.J., Wand, M., Zhang, H., Cohen-Or, D., Bokeloh, M.: Structure-aware shape processing. In: Eurographics STARs, pp. 175–197 (2013)
32. Parikh, D., Grauman, K.: Interactively building a discriminative vocabulary of nameable attributes. In: IEEE CVPR, pp. 1681–1688. IEEE (2011)
33. Parikh, D., Grauman, K.: Relative attributes. In: IEEE Conference on Computer Vision, pp. 503–510. IEEE (2011)
34. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
35. Savva, M., Chang, A., Hanrahan, P.: Semantically-enriched 3d models for common-sense knowledge. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 24–31 (2015)
36. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. ACM SIGGRAPH Comput. Graph. **20**(4), 151–160 (1986)
37. Song, S., Xiao, J.: Deep sliding shapes for amodal 3d object detection in rgb-d images. arXiv preprint arXiv:1511.02300 (2015)
38. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 945–953 (2015)
39. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Single-view to multi-view: reconstructing unseen views with a convolutional network. arXiv preprint arXiv:1511.06702 (2015)
40. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: a deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1912–1920 (2015)
41. Xie, J., Fang, Y., Zhu, F., Wong, E.: Deepshape: deep learned shape descriptor for 3d shape matching and retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1275–1283 (2015)
42. Xu, K., Kim, V.G., Huang, Q., Kalogerakis, E.: Data-driven shape analysis and processing. Computer Graphics Forum (to appear)
43. Yang, J., Reed, S.E., Yang, M.H., Lee, H.: Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In: Advances in Neural Information Processing Systems, pp. 1099–1107 (2015)
44. Yumer, M.E., Chaudhuri, S., Hodgins, J.K., Kara, L.B.: Semantic shape editing using deformation handles. ACM Trans. Graph. (TOG) **34**(4), 86 (2015)
45. Yumer, M.E., Kara, L.B.: Co-abstraction of shape collections. ACM Trans. Graph. (TOG) **31**(6), 166 (2012)
46. Yumer, M.E., Kara, L.B.: Co-constrained handles for deformation in shape collections. ACM Trans. Graph. (TOG) **33**(6), 187 (2014)