Feature Space Transfer as Data Augmentation for Few-shot Classification and Single-View Reconstruction

Bo Liu, *Student Member, IEEE,* Xudong Wang, *Student Member, IEEE,* Mandar Dixit, *Member, IEEE,* Roland Kwitt, *Member, IEEE,* and Nuno Vasconcelos, *Fellow, IEEE*

Abstract—The problem of data augmentation in feature space is considered. A new architecture, denoted the FeATure TransfEr Network (FATTEN), is proposed for the modeling of feature trajectories induced by variations of object pose. This architecture exploits a parametrization of the pose manifold in terms of pose and appearance. This leads to a deep encoder/decoder network architecture, where the encoder factors into an appearance and a pose predictor. Unlike previous attempts at trajectory transfer, FATTEN can be efficiently trained end-to-end, with no need to train separate feature transfer functions. This is realized by supplying the decoder with information about a target pose and the use of a multi-task loss that penalizes category- and pose-mismatches. In result, FATTEN discourages discontinuous or non-smooth trajectories that fail to capture the structure of the pose manifold, and generalizes well on object recognition tasks involving large pose variation. For few-shot recognition, meta-learning is used to further stabilize the model when applied on unseen classes. Experimental results on the artificial ModelNet database show that it can successfully learn to map source features to target features of a desired pose, while preserving class identity. Most notably, by using feature space transfer for data augmentation (w.r.t. pose and depth) on SUN-RGBD objects, we demonstrate considerable performance improvements on one/few-shot object recognition in a transfer learning setup, compared to current state-of-the-art methods. The method is also applied on single-view reconstruction. By augmenting shape codes in terms of poses, it boosts the performance of the auto-encoder based reconstruction method.

Index Terms—Feature Augmentation, Few-Shot Learning, Meta-Learning, Single-View Reconstruction.

1 Introduction

Convolutional neural networks (CNNs) trained on large datasets, such as ImageNet [1], have shown significant gains for computer vision problems like object recognition over the last few years. These models not only achieve human level performance in recognition challenges, but are also easily transferable to other data domains or tasks, by fine tuning. Many recent works have shown that ImageNet trained CNNs, like AlexNet [2], VGG [3], GoogLeNet [4], or ResNet [5] can be used as feature extractors for the solution of many other problems. Nevertheless, there are still challenges to CNN-based recognition. One limitation is that existing CNNs still have limited ability to handle pose variability. This is, in part, due to limitations of existing datasets, which are usually collected on the web and are biased towards a certain type of images. For example, objects that have a well defined "frontal view," such as "couch" or "clock," are rarely available from viewing angles that differ significantly from frontal.

This is problematic for applications like robotics, where a robot might have to navigate around or manipulate such objects. When implemented in real time, current CNNs tend to produce object labels that are unstable with respect to viewing angle. The resulting object recognition can vary from nearly perfect under some views to much weaker for neighboring, and very similar, views. One potential solution to the problem is to rely on larger datasets with a much more

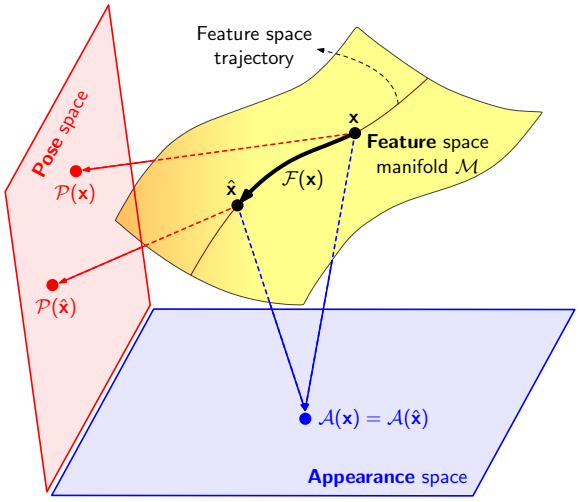


Fig. 1. Schematic illustration of *feature space transfer* for variations in *pose*. The input feature $\mathbf{x}$ and transferred feature $\hat{\mathbf{x}}$ are projected to the same point in *appearance space*, but have different mapping points in *pose space*.

- *B. Liu and N. Vasconcelos are with the Department of Electrical and Computer Engineering, University of California, San Diego, CA, 92093. E-mail:{boliu, nvasconcelos}@ucsd.edu*
- *M. Dixit is with the Microsoft, Redmond, WA, 98052. E-mail:madixit@microsoft.com*
- *X. Wang is with the EECS Department at University of California, Berkeley.*
- *R. Kwitt is with University of Salzburg, Austria.*

Manuscript received April 19, 2005; revised August 26, 2015.

dense sampling of the viewing sphere. This, however, is not trivial, for a number of reasons. *First*, for many classes, such images are not easy to find on the web in large enough quantities. *Second*, because existing recognition methods are weakest at recognizing "off-view" images, the process cannot be easily automated. *Third*, the alternative of collecting these images in the lab is quite daunting. While this has been done in the past, *e.g.*, the COIL [6], NORB [7], or Yale face dataset, these datasets are too small by modern standards. The set-ups used to collect them, by either using a robotic table and several cameras, or building a camera dome, can also not be easily replicated and do not lend themselves to distributed dataset creation efforts, such as crowd sourcing. Finally, even if feasible to assemble, such datasets would be massive and difficult to process. For example, the NORB recommendation of collecting 9 elevations, 36 azimuths, and 6 lighting conditions per object, results in 1944 images per object. Applying this standard to ImageNet would result in a dataset of close to 2 billion images!

Some of these problems can be addressed with computer generated images. This is indeed an established practice for problems that require multiple object views, such as shape recognition, where synthetic image datasets [8], [9] are routinely used. However, the application of networks trained on synthetic data to real images raises a problem of *domain adaptation*. Despite a vast literature on the topic [10], [11], [12], [13], [14], [15], [16], adaptation methods are usually not tailored for the transfer of object poses. In particular, they do not account the fact that, as illustrated in Fig. 1, objects subject to pose variation span low-dimensional manifolds of image space, or corresponding spaces of CNN features. This is because objects can be decomposed into appearance and pose components. While the appearance component is fixed, the pose component varies with viewing angle. Since the latter has few degrees of freedom, the *pose trajectories* spanned by the object for different angles define low-dimensional surfaces in image or feature space. The mapping between locations along a pose trajectory is denoted *pose transfer.* This has only been considered by a few works [17], [18], who have proposed models with explicit pose inputs to transfer objects along the pose manifold.

Besides explicit pose transfer, it is also possible to resort to few-shot learning methods. One popular solution is to rely on meta-learning [19]. This is a generic term for methods that try to train a "learner". Many recent works show that meta-learning is useful for the few-shot learning of a parameterized function that maps limited labeled data to a classifier [20], [21]. For pose transfer, this function can be meta-trained from data with many poses. However, all current meta-learning methods treat the training procedure as a black box, and rely on meta-training to learn all unknown structures implicitly. In particular, they don't have an explicit model for pose transfer. Moreover, most methods are only trained on original data, without augmentation. One possibility is to add an hallucination module [22], by using a generator to leverage prior visual knowledge and hallucinate additional training data for better classification. However, the generator of [22] fails to take the advantage of pose transfer to generate more effective data.

The benefits of data augmentation along pose trajectories are not limited to classification. For example, single-view reconstruction is another prime application target. This follows from the ill-defined nature of 3D shape recovery from a single object image, since part of the shape is never seen. Traditional augmentations are not effective for this problem and can even be misleading if combining information from different objects. We find that pose transfer is specifically suitable for this topic, because augmentations across pose trajectory leverage information from multiple views. This provides extra 3D structure while maintaining the semantic object information.

Contribution. In this work, we propose a universal framework, termed *FeATure TransfEr Network* (FATTEN), that addresses these problems. Essentially, FATTEN is an encoder-decoder architecture, inspired by Fig. 1. We parametrized pose trajectory transfer in terms of an *appearance map*, which captures properties invariant on pose, such as object texture and structure, and a *pose map*, which is pose dependent. Th *encoder* maps an input feature $\mathbf{x}$ into a pair of appearance $\mathcal{A}(\mathbf{x})$ and pose $\mathcal{P}(\mathbf{x})$ parameters. The *decoder* then takes the appearance parameter together with a target pose parameter $\mathbf{t} = \mathcal{P}(\hat{\mathbf{x}})$ and generate a feature vector $\hat{\mathbf{x}}$ with the new pose.

The FATTEN model is applied to few-shot recognition and single-view reconstruction. For recognition, a classifier is learned by meta-learning, using a parametric learner that takes the pose-generated data as labeled data to produce a classifier. To avoid mismatching, both pose transfer module and classifier are trained end-to-end, using a multi-task loss that accounts for both classification and feature transfer errors. For single-view reconstruction, FATTEN is used to augment shape codes. A 3D auto-encoder is used for 3D reconstruction, producing a shape code that is used as a bridge between a 2D image and the corresponding 3D shape. FATTEN is trained to augment shape codes along pose trajectories. The combination of these augmentations is shown to improve 3D reconstruction quality.

The performance of FATTEN is investigated in two stages. In the first stage, we examine the effectiveness of pose transfer. We utilize the model in a *multi-view retrieval* task, where generated features are used to retrieve features by category and pose. These experiments are carried out on a synthetic 3D dataset ModelNet [23]. Results show that our pose transfer module hallucinates features with good quality along both object category and pose dimensions, for applications involving computer graphics imagery. This could be of use for a now large body of 3D shape classification works [23], [24], [25], [26], where such datasets are predominant. Second, FATTEN is embeded with other models to work on few-shot recognition and single-view reconstruction. For few-shot recognition, we combine the pose transfer module and a classifier and train the whole model end-to-end. We show that the proposed architecture outperfroms both pure meta-learning methods and meta-learning methods with generic feature hallucinator. For single-view reconstruction, we combine FATTEN with a popular 3D reconstruction model, the BSP-Net, whose performance is also shown to benefit from the addition of FATTEN.

Organization. In Section 2, we review related work. Section 3 introduces the proposed FATTEN architecture. Section 4 and 5 discuss the application of FATTEN to few-shot recognition and single-view reconstruction. Section 6

presents experimental results on ModelNet, SUN-RGBD, and ShapeNet. Finally, Section 7 concludes the paper with a discussion of the main points and an outlook on open issues.

2 Related Work

Objects describe smooth trajectories in image space, where they span a 3D manifold, parameterized by the viewing angle. Hence, many of the manifold modeling methods proposed in the literature [27], [28], [29] could, in principle, be used to develop trajectory transfer algorithms. However, many of these methods are transductive, *i.e.*, they do not produce a function that can make predictions for images outside of the training set, and do not leverage recent advances in deep learning. While deep learning could be used to explicitly model pose manifolds, it is difficult to rely on CNNs pre-trained on ImageNet for this purpose. This is because these networks attempt to collapse the manifold into a space where class discrimination is linear. On the other hand, the feature trajectories in response to pose variability are readily available. These trajectories are also much easier to model. For example, if the CNN is successful in mapping the pose manifold of a given object into a single point, *i.e.*, exhibits total pose invariance for that object, the problem is already solved and trajectory leaning is trivial for that object.

On the other hand, trajectory transfer is popular for problems involving multi-view recognition. Due to the increased cost and difficulty of multi-view image collecting, such problems usually consider some level of learning from synthetic images. In fact, there is an established practice in shape recognition, where synthetic 3D datasets [8], [9] are widely used. A rich literature in shape recognition from synthetic datasets has produced many 3D representations [23], [24], [25], [26], [30], [31]. Nevertheless, it has also been shown that the 3D recognition problem can be efficiently solved as multi-view 2D recognition by using simple multi-view extensions of current CNNs [25]. However, it is not evident how these conclusions can generalize to real world datasets.

An important component of the proposed model is a pose transfer module that augments a dataset with feature responses of unseen object poses. In this sense, the problem is related to novel view synthesis [32], [33], [34], [35], [36], [37], [38]. [36] uses appearance flows to synthesize novel views of both objects and scenes. [34] combines information from multiple views. [37] and [38] mainly focus on scene images. All these works aim to generate realistic images, but do not discuss potential benefits for image classification. This mostly due to the difficulty of generating images, leading to models that only work well on limited categories. This is unlike the proposed model, which simplifies the task by only generating features. This generalizes to larger sets of categories and improves feature robustness.

With the introduction of large scale 3D datasets, such as ShapeNet [39], 3D modeling has been widely studied. One of the important tasks is single-view reconstruction, which recovers a 3D shape model from a single view of an object. Early works, such as 3D-R2N2 [40], generalize 2D convolution to 3D and model 3D shapes with voxels. This usually has low resolution, due to limits in computation and memory. Later on, methods such as [41], directly generate meshes or surfaces of the shape. This, however, could lead to over complicated surfaces, and sometimes produces open surfaces. Recently, there has been a trend of using implicit models [42], [43], [44], which model the shape with a 3D point classifier. In this work, we adopt one of the most recent and successful implicit models, the BSP-Net [44].

Instead, trajectory transfer is more closely related to the topic of transfer learning, where, there extensive work has been devoted to problems such as zero-shot [10], [11], [12] or few-shot learning [13], [14], [15]. Meta-learning has recently shown its effectiveness for few-shot recognition. Some methods, such as MAML and its variants [20], [45], or LEO [46], are gradient based. These methods take advantage of second derivatives to optimize the model from few-shot samples. Another group of methods, including the matching network [14], prototypical network [21], relation network [47], and category traversal [48], aims to learn robust metrics. Some few-shot methods have also proposed to augment training data by combining GANs with meta-learning [22], synthesizing features across object views [18] or using other forms of data hallucination [49]. Learning without forgetting [50] aims to transfer the existing classifier to novel classes without loss of performance on the base ones. [51] extends the few-shot problem from recognition to detection by feature reweighting. Self-supervised learning or representationg learning is a technique aiming for knowledge transfer. It has recently been studied for general classification tasks [52], [53], [54] However, these methods tend to be of general purpose. None of them exploits specific properties of the pose manifold, such as the parametrization of Fig. 1. The introduction of networks that enforce such parameterizations is a form of regularization that improves on the transfer performance of generic procedures.

3 The Feature Transfer Meta-Learner Architecture

In this section, we describe the proposed architecture for *feature space transfer*.

3.1 Feature Transfer Motivation

In this work, we assume the availability of a training set with pose annotations, *i.e.*, $\mathcal{S}_{train} = \{(\mathbf{x}_n, \mathbf{p}_n, y_n)\}_n$, where $\mathbf{x}_n \in \mathbb{R}^D$ is the feature vector (*e.g.*, a CNN activation at some layer) extracted from an image, $\mathbf{p}_n$ is the corresponding pose value and y_n a category label. The pose value could be a scalar p_n, *e.g.*, the azimuth angle on the viewing sphere, but is more generally a vector, *e.g.*, also encoding an elevation angle or even the distance to the object (object depth). The feature transfer problem is to learn the transfer function $\mathcal{F}(\mathbf{x}_n, \mathbf{p})$ that maps the source feature vector $\mathbf{x}_n$ to a target feature vector $\hat{\mathbf{x}}_n$ corresponding to a new pose $\mathbf{p}$.

3.2 The Feature Transfer Network Architecture

The FeATure TransfEr Network (FATTEN) architecture is illustrated by Fig. 1, which depicts a pose manifold spanned by an object under pose variation, and parameterized by two variables. One of them is an *appearance descriptor* $\mathbf{a} \in \mathbb{R}^A$ that represents pose invariant properties such as color or

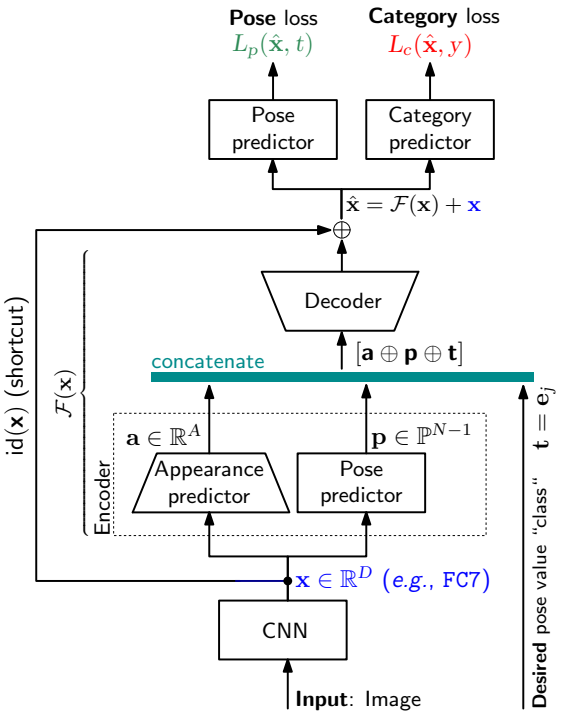


Fig. 2. The FATTEN architecture. Here, id denotes the identity shortcut connection, D the dimensionality of the input feature space, A the dimensionality of the appearance space and $\mathbb{P}^{N-1}$ the $N-1$ probability simplex. Both pose predictors are pre-trained and share parameters.

texture. This means that it has the same value for all points on the manifold. It can be thought of as an object code that distinguishes the manifold spanned by one object from those spanned by others. It is worth noting that appearance is not the same as category. Some objects of different categories may share more similar appearance than those of the same category. The other variable is a *pose descriptor* $\mathbf{p} \in \mathbb{R}^N$ that captures the point $\mathbf{x}$ that corresponds to a particular pose $\mathbf{p}$. In form, a feature point $\mathbf{x}$ on the manifold could be considered as the implementation of a mapping

$$\phi : \mathbb{R}^A \times \mathbb{R}^N \to \mathcal{M}, \quad \phi(\mathbf{a}, \mathbf{p}) \mapsto \mathbf{x} \ . \tag{1}$$

The FATTEN architecture, as a encoder-decoder architecture shown in Fig. 2, models the relationship between the feature vectors extracted from object images and its associated appearance and pose parameters. The encoder is designed to invert the mapping of (1), namely implement ϕ^{-1}. Given a feature vector $\mathbf{x}$, it produces an estimate of the appearance $\mathbf{a}$ and pose $\mathbf{p}$ parameters. A *target* pose parameter $\mathbf{t}$ specifies the corresponding pose of the desired feature vector $\hat{\mathbf{x}}$, which will be then generated by a decoder that processes the concatenation of $\mathbf{a}$, $\mathbf{p}$ and $\mathbf{t}$. While, in principle, it would be sufficient to derive $\hat{\mathbf{x}}$ from $\phi(\mathbf{a}, \mathbf{t})$, *i.e.*, to use the inverse of the encoder as a decoder, we have obtained the best results with the following modifications.

First, to prevent the encoder/decoder pair from learning a mapping that simply "matches" feature pairs, FATTEN implements the residual learning paradigm of [5]. In particular, the encoder-decoder is only used to learn the residual

$$\mathcal{F}(\mathbf{x}) = \hat{\mathbf{x}} - \mathbf{x} \tag{2}$$

between the target and source feature vectors. *Second*, two modules that explicitly predict the appearance $\mathbf{a}$ and pose $\mathbf{p}$ parameters are used instead of a single encoder. In our experience this decomposition facilitates learning, since the pose predictor can be learned with full supervision. *Third*, the source $\mathbf{p}$ and target $\mathbf{t}$ pose parameters are encoded as one-hot vectors instead of continuous scalars. This makes the dimensionality of the pose parameters closer to that of the appearance parameter, and triggers a more balanced training procedure. We have noted that, otherwise, the learning algorithm can have a tendency to ignore the pose parameters and produce a limited diversity of target feature vectors. *Finally*, the decoder can leverage the source pose $\mathbf{p}$, in addition to $\mathbf{a}$ and $\mathbf{t}$ in (1). This again guarantees that the *intermediate* representation is higher dimensional and accelerates the learning of the decoder. We next discuss the details of the various network modules.

3.3 FATTEN Details

Encoder. The encoder consists of a pose and an appearance predictor. The pose predictor implements the mapping $\mathbf{p} = \mathcal{P}(\mathbf{x})$ from input feature vectors $\mathbf{x}$ to pose descriptors $\mathbf{p}$. The poses, as azimuth angle, are first internally regulated into a code vector $\mathbf{c} \in \mathbb{R}^N$ of dimensionality comparable to that of the appearance vector $\mathbf{a}$. In the current implementation of FATTEN this is achieved in three steps. First, the full angle space is quantized into N cells with centroids $\mathbf{m}_i$. Second, Each pose is then assigned to the cell of the nearest representative $\mathbf{m}^*$ and converted by a N-dimensional one-hot vector that identifies $\mathbf{m}^*$. The pose mapping $\mathcal{P}$ is finally implemented with a N-way classifier that maps $\mathbf{x}$ into a vector of posterior probabilities

$$\mathbf{p} = [p(\mathbf{m}_1|\mathbf{x}), \ldots, p(\mathbf{m}_N|\mathbf{x})] \tag{3}$$

on the $N-1$ probability simplex $\mathbb{P}^{N-1}$. The classifier consists of a two-layer neural network, composed of a fully-connected layer, batch normalization, and a ReLU, followed by a softmax layer.

The appearance predictor implements the mapping $\mathbf{a} = \mathcal{A}(\mathbf{x})$ from input feature vectors $\mathbf{x}$ to appearance descriptors $\mathbf{a}$. This is implemented with a two-layer network, where each layer consists of a fully-connected layer, batch normalization, and a ELU layer.

The outputs of the pose and appearance predictors are concatenated with a one-hot encoding of the *target* pose. The encoding is implemented in the same way as the pose code vector of the pose predictor.

Decoder. The decoder maps the vector of concatenated appearance and pose parameters

$$[\mathbf{a} \ \oplus \ \mathbf{p} \ \oplus \ \mathbf{t}] \tag{4}$$

where $\oplus$ denotes vector concatenation, into the residual $\hat{\mathbf{x}} - \mathbf{x}$ of (2). It is implemented with a two layer network, where the first layer contains a sequence of fully-connected layer, batch normalization, and ELU, and the second is a fully connected layer. The decoder output $\mathcal{F}(\mathbf{x})$ is then added to the input $\mathbf{x}$ to produce the target $\hat{\mathbf{x}}$.

Although conceptually similar, our architecture is different from AGA [17] and solves some key limitations of the latter. In particular, the feature synthesis function $f(\mathbf{x}, \mathbf{p}, \mathbf{t})$

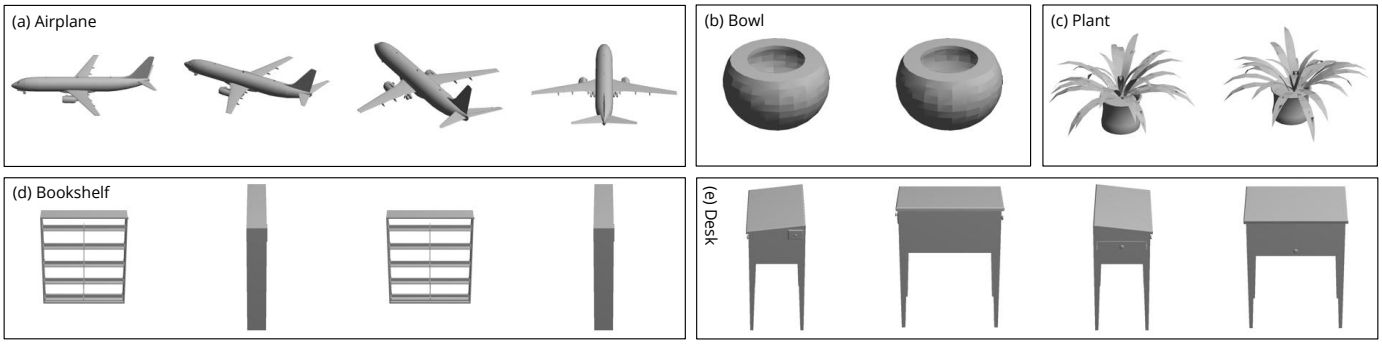


Fig. 3. Exemplary ModelNet [23] views: (a) different views of one object (airplane); (b)-(c) different views of two *symmetric* objects (bowl, plant); (d)-(e) four views (bookshelf, desk) with 90 degrees difference.

of AGA is implemented as a series of encoder-decoder modules $f_{\mathbf{p},\mathbf{t}}(x)$, one for each pair of $(\mathbf{p},\mathbf{t})$. The number of these functions grows exponentially and AGA needs to learn a different f for each $\mathbf{p} \rightarrow \mathbf{t}$ and $\mathbf{t} \rightarrow \mathbf{p}$; there is no provision to share information. Since FATTEN uses a *single* network to solve this task, (1) model complexity scales favorably with pose quantization and (2) due to weight sharing, pose translations are informed by each other. Also, AGA uses an L_2 regularizer in feature space, which may not preserve class identity; FATTEN uses a category loss to help address this problem.

3.4 FATTEN Training

FATTEN is trained as a data generator, independently of the downstream tasks to which it may be applied. A multi-task loss is designed to ensure *end-to-end* training that accomplishes two goals. The *first* is that the synthesized feature vector $\hat{\mathbf{x}}$ should correspond to the desired pose $\mathbf{t}$. This constraint is enforced by the pose loss, which is the cross-entropy loss commonly used for classification

$$L_p(\hat{\mathbf{x}},\mathbf{t}) = -\mathbf{t}^T \log \rho(\mathcal{P}(\hat{\mathbf{x}})), \tag{5}$$

where ρ is the softmax function, i.e. $\rho_j(v) = \frac{e^{v_j}}{\sum_k e^{v_k}}$. Note that, as shown in Fig. 2, this requires feeding the target feature vector $\hat{\mathbf{x}}$ into a pose predictor $\mathcal{P}$. It is worth noting that, while this is only needed for training, the loss of (5) can also be computed during inference, since the target pose $\mathbf{t}$ is known. This can be used as a diagnostic measure for the performance of FATTEN.

The *second* goal is that the generated feature vector $\hat{\mathbf{x}}$ maintains all the semantic information of the source vector $\mathbf{x}$. This semantic information can vary from task to task. For classification, it is the category label y. Ideally, the synthesized feature vectors should achieve the same recognition results as the network used to extract the feature vectors, denoted as CNN in Fig. 2, in the original problem. To guarantee this, FATTEN uses the linear classifier obtained by training the CNN backbone as a category predictor. This predictor is fixed during the training of the remaining FATTEN blocks to avoid overfitting. Its accuracy is measured by the cross-entropy loss

$$L_c(\hat{\mathbf{x}},y) = -\log \rho_y(\hat{\mathbf{x}}), \tag{6}$$

where $\rho(v)$ is the softmax output of the category predictor. The *multi-task loss* is finaly defined as

$$L_{\text{FATTEN}}(\hat{\mathbf{x}},\mathbf{t},y) = L_p(\hat{\mathbf{x}},\mathbf{t}) + L_c(\hat{\mathbf{x}},y). \tag{7}$$

This leads to three stages of training. In the first, the CNN backbone and class predictor are trained for category prediction, without pose information. This step can be skipped if a pre-trained CNN is available. In the second stage, the pose predictor $\mathcal{P}(\mathbf{x})$ is trained with pose supervision. This is then *embedded* into the encoder-decoder structure and pose loss structure. In the final training stage, both pose and category predictors are fixed, and the appearance predictor and decoder are trained end-to-end without further appearance supervision. We found this training strategy to be beneficial in two ways. First, embedding the pre-trained pose predictor reduces the number of degrees of freedom in the network, minimizing the ambiguity inherent to the fact that a given feature vector could be consistent with multiple pairs of pose and appearance parameters. Unlike pose parameters, FATTEN allows a certain flexibility on the appearance prediction for further robustness. For example, while all feature vectors $\mathbf{x}$ extracted from views of the same object should map into the same appearance parameter value $\mathbf{a}$, we do not enforce such constraint. This endows the network with invariance to small variations of the appearance descriptor, due to occlusions, variations in lighting, etc. Second, by pre-training the pose and category predictor, *only* weights of the encoder/decoder need to be learned end to end. The weights of the sub-networks used by the loss function(s) are fixed. This minimizes the chance that FATTEN will over-fit to specific poses or object categories.

4 Few-Shot Recognition

A one/few-shot classifier is a classifier trained with one or few examples per class. In this section we investigate the use of FATTEN as a data augmentation technique for this problem. We consider different classification strategies, from the traditional support vector machine (SVM) to modern meta-learners.

4.1 SVM

A classical solution to the few-shot problem is to rely on the good generalization ability of the SVM [55], which is

typically used as a binary classifier. Given a set of training examples $\{\mathbf{x}_i, y_i\}$, the SVM training algorithm minimizes the hinge loss

$$L(\mathbf{x}_i, y_i) = \max(0, 1 - y_i(\mathbf{w}\mathbf{x}_i - b)), \tag{8}$$

where $\mathbf{w}$ and b are the classifier parameters to be learned. This can be used to induce a soft margin by introduction of a regularization term, leading to the loss

$$L_{SVM} = \frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i(\mathbf{w}\mathbf{x}_i - b)) + \lambda||\mathbf{w}||^2. \tag{9}$$

A multi-class SVM is then built by learning binary classifiers that oppose one class to all others [56].

One way to implement few shot classification is to rely on a pre-trained CNN backbone, which is used a feature extractor. The softmax layer used for category prediction is then replaced by an SVM, which is trained with the features produced by the CNN for the few examples available from the target task. In our experiments, FATTEN is used as a feature augmentation technique for this second stage, where feature vectors of unavailable poses are synthesized to produce more data for SVM training. This allows us to increase the number of training examples, enabling more robust SVM training. While the simple use of an SVM is not a state of the art method for few-shot classification, the fact that it decouples feature synthesis from classification enables an effective evaluation of the benefits of data augmentation.

4.2 Meta-Learning Models

Recently, a number of model generalization approaches have been proposed or aggregated under the meta-learning term. These methods aim to produce a generalized classification method that can learn a model independently of the data domain, and adapt to new domains easily with limited or even no data. This property makes meta-learning models particularly suited for few-shot learning problems, where a training set S_{tr} of class space Y_{tr} and a testing set S_{te} of class space Y_{te} are given, with disjoint Y_{tr} and Y_{te}. While many samples are provided for S_{tr}, only a small number of samples are available in S_{te}. The model is evaluated by its classification accuracy in class space Y_{te}. More specifically, a *query set* Q with the same class space Y_{te} as S_{te}, which is also known is *support set*, is given. The model is evaluated on Q, with the help of samples and labels supported by S_{te}. When the support set S_{te} has N classes and K samples per class, the problem is called *N-way K-shot few-shot classification*.

Unlike traditional mini-batch training, meta-training methods employ episode training on S_{tr}. For each episode, a task is created by sampling a meta-training set M_{tr} from S_{tr} and a meta testing set M_{te} with overlapping classes from S_{te}. The meta-learning model F is then updated using all samples from M_{tr} before classifying samples in M_{te}. Given the updated model $F(\mathbf{x}, M_{tr})$ and example $(\mathbf{x}, y)$ in M_{te}, the model prediction $\hat{y} = F(\mathbf{x}, M_{tr})$ is then chosen to minimize a classification loss $L(\hat{y}, y)$.

The training of each episode mimics the scenario where a model is transferred to a new task. Because a new task is sampled and optimized per episode, the model is less overfitted to the given training set S_{tr}, and will be more effective when transferred to a new task. During testing, given a new task of support set S_{te} and query set Q, the prediction for a sample $\mathbf{x} \in Q$ is made with $F(\mathbf{x}, S_{te})$.

Different meta-learning methods use different models and even different learning strategies. They can be broadly grouped into two types: gradient-based and metric-based. In gradient-based methods, M_{tr} is used to update the model F by directly back-propagating gradients. However, in metric-based learning, M_{tr} is used to provide class prototypes. Since FATTEN is a data generator, it is more suitable for use with metric-based methods. We consider the following two metric-learning approaches.

Prototypical Networks. Prototypical networks [21] leverages the distance from class center in an M-dimensional feature space. Given an embedding function f_ϕ, The meta-learning model is a classifier

$$F(\mathbf{x}) = \arg\max_{y \in Y_{te}} \frac{e^{-d(\mathbf{x}, \mathbf{c}_y)}}{\sum_{k \in Y_{te}} e^{-d(\mathbf{x}, \mathbf{c}_k)}} \tag{10}$$

where

$$d(\mathbf{x}, \mathbf{c}_k) = ||f_\phi(\mathbf{x}) - \mathbf{c}_k||_2 \tag{11}$$

is the Euclidean distance between example $\mathbf{x}$ and a prototype $\mathbf{c}_k$ of class k. During meta-training with M_{tr} or testing with S_{te}, F is updated by computing class prototypes as

$$\mathbf{c}_k = \frac{1}{|S_k|}\sum_{\mathbf{x}_i \in S_k} f_\phi(\mathbf{x}_i), \tag{12}$$

where S_k is the set of all points with label $y = k$ in M_{tr} or S_{te}. During meta-training, given meta-training set M_{tr} and a sample ($\mathbf{x}$, y) from meta-testing set M_{te}, the classification loss is defined as

$$\begin{aligned} L_{\text{META}}(\mathbf{x}, y, M_{tr}) &= d(f_\phi(\mathbf{x}), \mathbf{c}_y) \\ &+ \log\sum_k \exp[-d(f_\phi(\mathbf{x}), \mathbf{c}_k)]. \end{aligned} \tag{13}$$

Relation Networks. Similar to prototypical networks, relation networks [47] leverage an embedding function f_ϕ to produce a feature map for example $\mathbf{x}$. However, the Euclidean distance is replaced by a relation module g_φ that produces a relation score in $[0, 1]$. The relation score of example $\mathbf{x}$ to class k is defined as

$$r(\mathbf{x}, k) = g_\varphi\left(\sum_{\mathbf{x}_i \in S_k} f_\phi(\mathbf{x}_i), f_\phi(\mathbf{x})\right), \tag{14}$$

where S_k is a set contains all points with label $y = k$ in M_{tr} (during meta-training) or S_{te} (during testing). The loss function is the mean square error (MSE)

$$L_{\text{META}}(\mathbf{x}, y, M_{tr}) = \sum_k (r(\mathbf{x}, k) - \mathbf{1}_{k=y})^2, \tag{15}$$

where $\mathbf{1}_{k=y}$ is the indicator function of $k = y$, matched pairs receive a relation score of 1 and mismatched ones a score of 0.

4.3 Meta Training

Recent work [22] has shown that training a data augmentation module end-to-end with a meta-learning algorithm can improve the performance of the latter. For pose augmentation, this requires that FATTEN and the meta-learner be

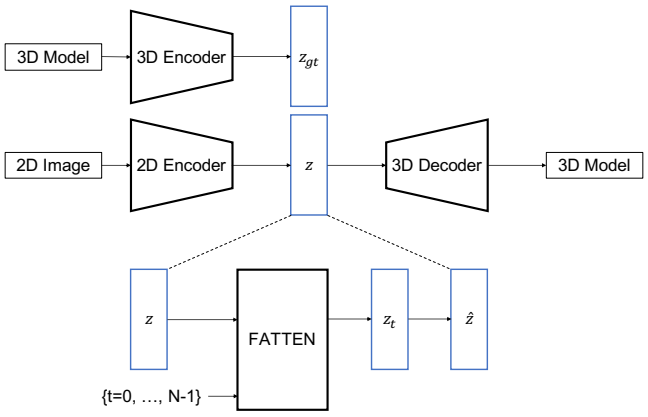


Fig. 4. Single view reconstruction with the BSP-Net. A 3D encoder extracts a ground-truth shape code from a 3D object, a 2D encoder extracts a shape code from a 2D image, and a decoder reconstructs a 3D model. FATTEN is applied to the shape code, which it augments along pose trajectories. The resulting pose-trajectory codes are combined into a shape code that is input to the BSP-Net decoder.

jointly trained. To accomplish this, a feature extractor and the FATTEN model are first pre-trained on training set S_{tr}, as discussed in Section 3.4, and then embedded into the meta-training procedure. In a second stage, FATTEN and the classification model are trained together with meta-learning. For this, in each episode, meta-training M_{tr} and meta-testing M_{te} sets are first assembled from S_{tr}. Then, every example in M_{tr} and M_{te} is fed through FATTEN to generate features with a set of target poses $\{\mathbf{t} = 1, \ldots, N\}$. These generated features together with the original features from M_{tr} and M_{te} create augmented meta-training/-testing sets M_{tr}^{aug} and M_{te}^{aug}. The meta-learning model is learned with the meta-learning loss of (13) or (15), but the meta-learning sets are replaced by M_{tr}^{aug} and M_{te}^{aug}. FATTEN is updated according to the FATTEN loss of (7) with the classification term replaced by the meta-learning classification loss. The overall loss of each episode is

$$L(\hat{\mathbf{x}}, y, \mathbf{t}, M_{tr}^{aug}) = L_{\text{META}}(\hat{\mathbf{x}}, y, M_{tr}^{aug}) + L_p(\hat{\mathbf{x}}, \mathbf{t}). \quad (16)$$

for any $(\hat{\mathbf{x}}, y) \in M_{te}^{aug}$. Because the feature extractor is fine-tuned, the pose predictor should be changed accordingly. However, we find that fine-tuning the pose predictor does not improve classification performance. This implies that without explicit supervision, the meta-learning fine-tuning does not change the latent pose trajectory. In result, we fix the pose predictor in all settings.

5 Single-View Reconstruction

Single-view reconstruction aims to reconstruct the 3D shape of an object from a single image of the object. This is quite difficult because most of the 3D structure of the object is occluded and has to be hallucinated by the reconstruction model. Since FATTEN is designed to augment features along pose trajectories it can simplify this hallucination, by synthesizing a set of views from the single available image.

5.1 Single-View Reconstruction Model

We consider the recent BSP-Net [44], one of the most recent and successful 3D shape generators. This is an implicit model that, given a shape feature vector and a point in 3D, classifies the point as being inside or outside the shape. The object surface is modelled as the combination of a set of *binary space partitions*, *i.e.* hyper-planes in 3D space.

For single-view reconstruction, the BSP-Net is implemented with the auto-encoder structure of Figure 4. A 2D convolutional neural network is used as an image encoder that maps the input image I into a shape code

$$\mathbf{z} = \mathcal{E}_{\text{image}}(I). \quad (17)$$

The decoder takes this code and a set of 3D point coordinates $\mathcal{X} = \{\mathbf{x}_i\}$, classifying each point as being inside or or outside the shape, producing a set of binary labels

$$\mathcal{Y} = \mathcal{D}_{\text{shape}}(\mathbf{z}, \{x_i\}) \quad (18)$$

that, together with $\mathcal{X}$, characterize the shape. For example, the sets $\mathcal{S}$ and $\mathcal{Y}$ can be used to create a surface mesh using the marching cubes algorithm [57].

5.2 Shape Code Augmentation

To apply FATTEN to the single view reconstruction problem, $\mathbf{z}$ can be seen as a feature vector that encodes shape information but is sensitive to pose information. FATTEN can then be used to augment $\mathbf{z}$ with feature vectors corresponding to other views of the object and these feature vectors then combined to achieve a more robust shape code. For this, the shape code predicted from the input image is first augmented along a pose trajectory of N pre-defined view angles $\mathbf{t}_k$, as shown in Figure 4. The shape code $\mathbf{z}$ and a one-hot encoded target pose $\mathbf{t}_k$ are input to the FATTEN module, which synthesizes a new shape code

$$\mathbf{z}_k = \text{FATTEN}(\mathbf{z}, \mathbf{t}_k) \quad (19)$$

corresponding to pose $\mathbf{t}_k$ but maintaining the semantic information of $\mathbf{z}$. In the second step, the set of feature vectors $\{\mathbf{z}_k, k = 1, \ldots, N\}$ produced by FATTEN is then mapped into a single shape code $\hat{\mathbf{z}}$, as required by the BSP-Net decoder $\mathcal{D}_{\text{shape}}$. In this work, this is implemented with a simple average pooling operation

$$\hat{\mathbf{z}} = \frac{1}{N} \sum_{k=1}^{N} \mathbf{z}_k. \quad (20)$$

Other reduction methods are discussed and compared in the experimental section.

5.3 Training

The BSP-Net is pre-trained as in [44] and fixed. The 2D encoder is used to extract the shape code $\mathbf{z}$ from input images. The training of FATTEN follows the procedure of Section 3.4 with one exception. For reconstruction, the goal is not to classify the input images, i.e. the semantic information is not in the form of class labels. The shape code should encode the shape of the object, instead of just its class. In the reconstruction task, ground-truth shapes are also available during training. In fact, they are used to train the image encoder $\mathcal{E}_{\text{image}}$. The corresponding shape codes $\mathbf{z}_{\text{gt}}$ summarize the semantic information required for this task. Hence, the classification loss of Section 3.4 is replaced by a regression loss, consisting of the L_2 distance

between the feature vector $\mathbf{z}_k$ synthesized by FATTEN and the ground-truth code

$$L_{\mathrm{s}}(\mathbf{z}_k, \mathbf{z}_{\mathrm{gt}}) = ||\mathbf{z}_k - \mathbf{z}_{\mathrm{gt}}||^2. \quad (21)$$

The FATTEN loss is finally given by

$$L(\mathbf{z}_k, \mathbf{t}_k, \mathbf{z}_{\mathrm{gt}}) = \lambda L_p(\mathbf{z}_k, \mathbf{t}_k) + L_{\mathrm{s}}(\mathbf{z}_k, \mathbf{z}_{\mathrm{gt}}). \quad (22)$$

6 Experiments

We first train and evaluate the FATTEN model on the artificial ModelNet [23] dataset (Sec. 6.1), and then assess its feature augmentation performance on the one/few-shot object recognition task of [17] (Sec. 6.2). This is done by both training an SVM on augmented data, to evaluate augmentation performance separately, and integrating classifier design and FATTEN training with meta-learning (Sec. 6.3). Finally, we evaluate the FATTEN model on the single-view reconstruction task, using ShapeNet [39] (Sec. 6.4).

6.1 Feature Quality Evaluation

6.1.1 Dataset

ModelNet [23] is a 3D synthetic data set of 3D voxel grids. It contains 4000 shapes from 40 object categories. Given a 3D shape, it is possible to render 2D images from any pose. In our experiments, we follow the rendering strategy of [25]. 12 virtual cameras are placed around the object, in increments of 30 degrees along the z-axis, and 30 degrees above the ground. Example rendered views are shown in Fig. 3. The training and testing splits are those proposed in the ModelNet benchmark, namely 80 objects per category for training and 20 for testing. However, the dataset contains some categories of symmetric objects, such as "bowl", which produce identical images from all views (see Fig. 3(b)) and some that lack any distinctive information across views, such as "plant" (see Fig. 3(c)). These objects are eliminated and only the remaining 28 object categories are used.

6.1.2 Implementation

To verify the generality of FATTEN, both VGG16 [3] and ResNet-101 [5] are adopted as backbone architectures in feature transfer experiments. All feature vectors $\mathbf{x}$ are collected from activations of the last fully-connected layer of networks fine-tuned on the training set, namely `fc7` in VGG16 and `pool5` in ResNet101. The pose predictor is trained with a learning rate of 0.01 for 1000 epochs, and evaluated on the testing corpus. The complete FATTEN model is then trained for 1000 epochs with a learning rate of 0.01. The angle range of $[0°, 360°]$ is split into 12 non-overlapping intervals of size $30°$ each, labeled as 0-11. Each angle is then converted to a classification label based on the interval it belongs to.

6.1.3 Feature transfer results

The feature transfer performance of FATTEN is assessed in two steps. The accuracy of the pose predictor is evaluated *first*, with the results listed in Table. 1. The large majority of the errors have magnitude of $180°$. This is not surprising, since ModelNet images have no texture. As as shown in Fig. 3(d)-(e), object views that differ by $180°$ can be similar or even identical for some objects. However, this is not a substantial problem for transfer. Since two feature vectors corresponding to the $180°$ difference are close to each other in feature space, to the point where the loss cannot distinguish them clearly, FATTEN will generate target features close to the source, which is the goal anyway. If these errors are disregarded, the pose prediction has accuracy 90.8% for VGG16 and 85.8% for ResNet-101.

TABLE 1
Top: Pose prediction error (in %); *Bottom:* Pose & category accuracy (in %) of generated features.

Degrees →	0	30	60	90	120	150	180
VGG16	72.3	2.2	1.1	4.0	0.9	1.0	18.5
ResNet-101	64.4	2.9	2.3	5.1	2.3	1.6	21.4

	Pose	**Object category**
VGG16	96.20	83.65
ResNet-101	99.95	84.13

The *second* evaluation step measures the feature transfer performance of the whole network, given the pre-trained pose predictor. During training, each feature in the training set is transferred to all 12 views (including identity). During testing, this is repeated for each test feature. The accuracy of the pose and category prediction of the features generated on the test corpus, is listed in Table 1. Note that, here, category refers to object category or class. It is clear that on a large synthetic dataset, such as ModelNet, FATTEN can generate features of good quality, as indicated by the pose prediction accuracy of 99.95% and the category prediction accuracy of 84.13% of the ResNet-101. Further, pose prediction error as well as pose and category accuracy of synthesized features is similar for the two backbones.

6.1.4 Retrieval with generated features

A set of retrieval experiments is performed on ModelNet to further assess the effectiveness of FATTEN generated features. These experiments address the question of whether the latter can be used to retrieve instances of (1) the same class or (2) the same pose. Since all features are extracted from the VGG16 `fc7` layer, the Euclidean distance

$$d_1(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_2 \quad (23)$$

is a sensible similarity measure for the purpose of retrieving images of the same *object category*. This is because the model is trained to map features with equal category labels to the same partitions of the feature space (enforced by the category loss L_c). However, d_1 is inadequate for *pose* retrieval. Instead, retrieval is based on the activation of the second fully-connected layer of the pose predictor $\mathcal{P}$, denoted by $\gamma(\mathbf{x})$. The *pose distance function* is then defined as

$$d_2(\mathbf{x}, \mathbf{y}) = ||\gamma(\mathbf{x}) - \gamma(\mathbf{y})||_2. \quad (24)$$

Finally, the performance of *joint* category & pose retrieval is measured with a combined distance, *i.e.*,

$$d_c(\mathbf{x}, \mathbf{y}) = d_1(\mathbf{x}, \mathbf{y}) + \lambda d_2(\mathbf{x}, \mathbf{y}). \quad (25)$$

All queries and instances to be retrieved are based on features *synthesized* for the *testing* corpus of ModelNet. For each synthesized feature vector, three queries are performed: (1) *Category*, (2) *Pose*, and (3) *Category & Pose*. This

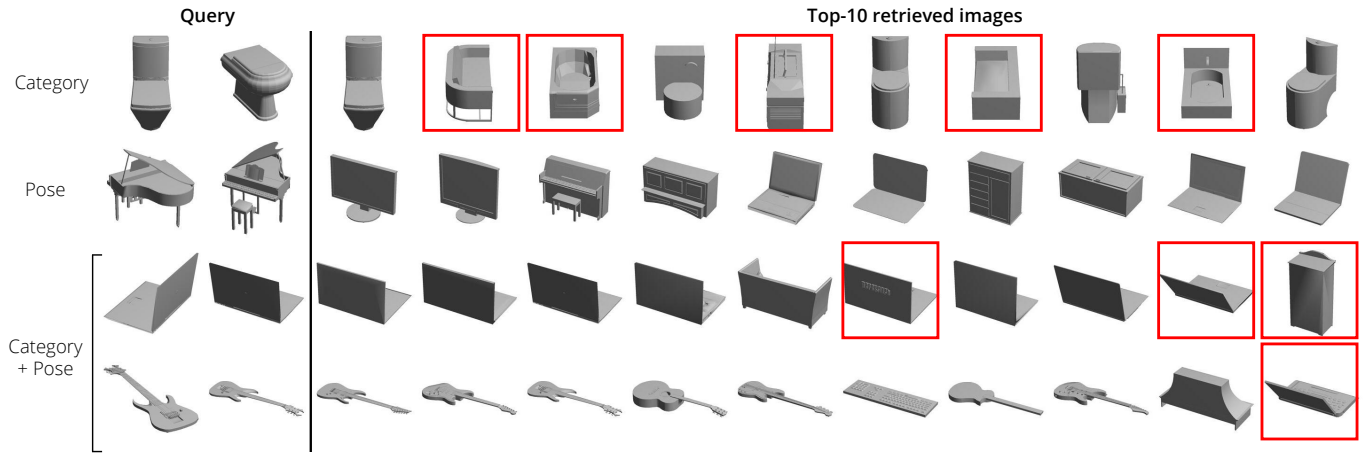


Fig. 5. Exemplary retrieval results for the experiments of Sec. 6.1.4. Rows are annotated by the retrieval *type* and errors are highlighted in red. In the **query**, *(left)* shows the original image, *(right)* shows the original image corresponding to the pose of the generated feature.

TABLE 2
Retrieval performance in mAP [%] of real and synthesized features, on the testing portion of ModelNet.

Feature type	(P)ose	(C)ategory	P + C
Real	54.58	32.71	23.65
Synthesized	77.62	28.89	11.07

is compared to the performance, on the same experiment, of the real features extracted from the testing corpus by the backbone CNN. Retrieval results are listed in Table 2 and some retrieval examples are shown in Fig. 5. The synthesized features enable a very high mAP for *pose retrieval*, even higher than the mAP of real features. This is strong evidence that FATTEN successfully encodes pose information in the transferred features. The mAP of the synthesized features is lower for *category retrieval* and the combination of both. However, the performance of the real features is also weak on these tasks. This could be due to a failure of mapping features from the same category into well defined neighborhoods, or to the distance metric used for retrieval. While retrieval performs a nearest neighbor search under these metrics, the network optimizes the cross-entropy loss on the softmax output(s) of both output branches of Fig. 2. The distance of (25) may be a particularly poor way to assess joint category and pose distances. In the following section, we will see that using a strong classifier (*e.g.*, a SVM) on the generated features produces significantly better results.

6.2 Data Augmentation on Few-shot Recognition

The experiments above provide no insight on whether FATTEN generates meaningful features for tasks involving real world data. In this section, we assess feature transfer performance on a one/few-shot object recognition problem. On this task, feature transfer is used for feature space "fattening" or *data augmentation*. The benchmark data is collected from SUN-RGBD [58], following the setup of [17].

6.2.1 Dataset

The whole SUN-RGBD dataset contains $10,335$ images and their corresponding depth maps. Additionally, 2D and 3D bounding boxes are available as ground truth for object detection. *Depth* (distance from the camera plane) and *Pose* (rotation around the vertical axis of the 3D coordinate system) are used as pose parameters in this task. The depth range of $[0, 5)$ m is broken into non-overlapping intervals of size 0.5m. An additional interval $[5, +\infty)$ is included for larger depth values. For pose, the angular range of $[0°, 180°]$ is divided into 12 non-overlapping intervals of size $15°$ each. These intervals are used for one-hot encoding and system training. However, to allow a fair comparison with AGA during testing, we restrict the desired pose t to take the values $45°$, $75°$, ..., $180°$, prescribed in [17]. This is mainly to ensure that our system generates 11 synthetic points along the *Depth* trajectory and 7 along the *Pose* trajectory.

The first $5,335$ images of SUN-RGBD are used for training and the remaining 5000 images for testing. If only ground truth bounding boxes were used for object extraction, the instances would not be balanced w.r.t. categories, nor w.r.t. pose/depth values. To remedy this issue, a fast R-CNN [59] object detector is fine-tuned on the dataset and proposals with IoU > 0.5 (to ground truth boxes) and detection scores > 0.7 are used to extract object images for training. Since this strategy produces a large amount of data, the training set can be easily balanced per category, as well as pose and depth. In the testing set, only ground truth bounding boxes are used to exact objects. All source features are exacted from the penultimate (*i.e.*, `fc7`) layer of the fine-tuned fast R-CNN detector for all instances from both training and testing sets.

Evaluation is based on the source and target object classes in [17]. We define a source dataset $\mathcal{S}$ and two different (disjoint) target datasets, $\mathcal{T}_1$ and $\mathcal{T}_2$. A third target dataset is defined as the union of the first two, $\mathcal{T}_3 = \mathcal{T}_1 \cup \mathcal{T}_2$. Table 3 lists all the object categories in each set. The instances in $\mathcal{S}$ are collected from the training portion of SUN-RGBD only, while those in $\mathcal{T}_1$ and $\mathcal{T}_2$ are collected from the testing set. Further, $\mathcal{S}$ does not have class overlap with any $\mathcal{T}_i$, which ensures that FATTEN has no access to shared knowledge between training/testing images or classes.

TABLE 3
List of object categories in the source $\mathcal{S}$ training set and the two target/evaluation sets $\mathcal{T}_1$ and $\mathcal{T}_2$.

$\mathcal{S}$ (19, Source)				$\mathcal{T}_1$ (10)		$\mathcal{T}_2$ (10)	
bathtub	counter	lamp	sofa	picture	stove	mug	microwave
bed	desk	monitor	table	whiteboard	cabinet	telephone	coffee table
bookshelf	door	night stand	tv	fridge	printer	bowl	recycle bin
box	dresser	pillow	toilet	counter	computer	bottle	cart
chair	garbage bin	sink		books	ottoman	scanner	bench

TABLE 4
One-/Five-shot recognition accuracy for three recognition problems (from SUN-RGBD). Accuracies (in %) are averaged over 500 random runs. **Baseline** denotes the accuracy of a linear SVM, when trained on *single* instances of each class only.

		Baseline	**Hal.** [49]	**AGA** [17]	**FATTEN**
1–shot	$\mathcal{T}_1$ (10)	33.74	35.43	39.10	44.99
	$\mathcal{T}_2$ (10)	23.76	21.12	30.12	34.70
	$\mathcal{T}_3$ (20)	22.84	21.67	26.67	32.20
5–shot	$\mathcal{T}_1$ (10)	50.03	50.31	56.92	58.82
	$\mathcal{T}_2$ (10)	36.76	38.07	47.04	50.69
	$\mathcal{T}_3$ (20)	37.37	38.24	42.87	47.07

6.2.2 Implementation

Predictors of *pose* and *depth* are trained with a learning rate of 0.01 for 1000 epochs. The FATTEN network is fine-tuned, starting from the weights obtained from the ModelNet experiment of Sec. 6.1, with a learning rate of 0.001 for 2000 epochs. The classification problems on $\mathcal{T}_1$ and $\mathcal{T}_2$ are 10-class problems, whereas $\mathcal{T}_3$ is a 20-class problem. As a *baseline* for one-shot learning, we train a linear SVM using *only* a single instance per class. We then feed those same instances into the FATTEN network to generate artificial features for different values of depth and pose, in particular, 11 values for depth and 7 for pose. After feature synthesis, a linear SVM is trained with the same parameters on the *augmented* ("fattened") feature set (source and target features).

6.2.3 Results

Table 4 lists the averaged one-shot (and five-shot) recognition accuracies (over 500 random runs) for all three evaluation sets $\mathcal{T}_i$. These are compared to the recognition accuracies of two data augmentation methods from the literature, feature hallucination [49] and AGA [17]. Table 4 supports the following conclusions. *First*, when compared to the SVM baseline, FATTEN achieves a remarkable and consistent improvement of around 10 percentage points on all evaluation sets. This indicates that FATTEN can actually embed the pose information into features and effectively "fatten" the data used to train the linear SVM. *Second*, and most notably, FATTEN achieves a significant improvement (about 5 percentage points) over AGA, and an even larger improvement over the feature hallucination approach of [49]. The improved performances of FATTEN over AGA and AGA over hallucination show that it is important (1) to exploit the structure of the pose manifold (which only FATTEN and AGA do), and (2) to rely on models that can capture defining properties of this manifold, such as continuity and smoothness of feature trajectories (which AGA does not).

While feature hallucination works well in the ImageNet1k low-shot setup of [49], Table 4 shows only marginal gains over the baseline (especially in the one-shot case). There may be several reasons as to why it fails in this setup. *First*, the number of examples per category (k in the notation of [49]) is a hyper-parameter set through cross-validation. To make the comparison fair, we chose to use the same value in all methods, which is $k = 19$. This may not be the optimal setting for [49]. *Second*, we adopt the same number of clusters as used by the authors when training the generator. However, the best value may depend on the dataset (ImageNet1k in [49] *vs.* SUN-RGBD here). Without clear guidelines of how to set this parameter, it seems challenging to adjust it appropriately. *Third*, all results of [49] list the top-5 accuracy, while we use top-1 accuracy. *Finally*, FATTEN takes advantage of pose and depth to generate features, while the hallucination feature generator is *non-parametric* and does not explicitly use this information.

The improvement of FATTEN over AGA can most likely be attributed to (1) the fact that AGA uses separate synthesis functions (trained independently) and (2) failure cases of the pose/depth predictor that determines *which* particular synthesis function is used. In case of the latter, generated features are likely to be less informative, or might even confound any subsequent classifier.

6.3 Augmentation with Meta-Learning

The experiments above show the effectiveness of pose guided data augmentation. However, the classifier is only trained a posteriori, on the augmented data. In this section, we evaluate the few-shot recognition problem with meta-learning, where the two are optimized jointly. Following the notations of Section 6.2, we use $\mathcal{S}$ during training, and $\mathcal{T}_i (i = 1, 2, 3)$ during testing.

6.3.1 Implementation

Meta-training involves the training of both feature extractor and classifier. We adopt the structure and fine-tune the parameters of the feature extractor used in Section 6.2 except the last fc layer, which is dicussed in Section 6.3.3. During meta-training, per episode, we sample M_{tr} and M_{te} from $\mathcal{S}$, and calculate the meta loss. To thoroughly evaluate the effectiveness of meta-learning with data augmentation, we propose three different meta-learning setups. **M**: a meta-learner is trained on $\mathcal{S}$ without data augmentation. The meta loss is calculated on M_{tr} and M_{te} with (13) or (15). **G+M**: a meta-learner is trained on augmented data. All data points in M_{tr} are first augmented by FATTEN to assemble M_{tr}^{aug}.

TABLE 5
One-shot recognition accuracy under three meta-learning setups. Accuracies (in %) are averaged over 500 random runs. **SVM** denotes a linear SVM trained on *single* instances of each class. **G+SVM** denotes a linear SVM trained on *augmented* instances of each class.

		M	G+M	GM+M	Imag. [22]	SVM	G+SVM
$\mathcal{T}_1$ (10)	**Prototypical**	44.52	33.26	46.78	43.13	33.74	44.99
	Relation	41.25	36.75	44.45	43.86		
$\mathcal{T}_2$ (10)	**Prototypical**	34.22	23.69	37.08	31.64	23.76	34.70
	Relation	31.74	26.73	35.99	34.80		
$\mathcal{T}_3$ (20)	**Prototypical**	28.79	19.30	35.89	32.52	22.84	32.20
	Relation	34.02	27.71	33.74	29.01		

TABLE 6
One-shot recognition accuracy of the "**GM+M**" setup for different embedding dimensions. Accuracies (in %) are averaged over 500 random runs.

Dimension	**64**	**256**	**1024**	**4096**
Prototypical	45.07	46.78	43.00	40.39
Relation	42.59	43.39	43.82	44.45

A meta loss is calculated on M_{tr}^{aug} and M_{te} with (13) or (15). Only the feature extractor and the meta-learning classifier are updated according to the loss. **GM+M**: A meta-learner is trained together with a data generator. All data points in M_{tr} and M_{te} are first augmented by FATTEN to assemble M_{tr}^{aug} and M_{te}^{aug}. The loss is calculated on M_{tr}^{aug} and M_{te}^{aug} with (16). Both the meta-learning classifier and the generator are updated according to the loss.

All meta-learning setups are tested with both the Prototypical [21] and the Relation Network [47]. All meta-training parameters are those in the original papers with exception of the feature space dimension, which is discussed in Section 6.3.3. During testing, 1 instance per class is randomly sampled from $\mathcal{T}_i$ to form a 1-shot problem, and the remaining samples are used as testing set. The average accuracy over 500 episodes of different random samples is reported, to reduce the effects of randomness.

6.3.2 Results

Table 5 lists the averaged results for all three setups and two meta-learners. These are compared to the recent Imaginary [22] meta-learning based data augmentation method. Two baselines, SVM on 1-shot and SVM on augmentation, are also listed for comparison.

The Table 5 supports several conclusions. *First*, both the Prototypical and the Relation Network perform much better than the SVM, even without data augmentation. *Second*, when synthesized features are directly applied to the classifier, both meta-learning structures have a big performance drop. We believe this is due to mismatching. Because the data augmentation module is not optimized for the classifier, the synthesized features are not useful for meta-learning. *Third*, the performance of both meta-learners improves substantially, when data augmentation module and meta-learner are trained jointly. This again illustrates the benefits of data augmentation with FATTEN.

Surprisingly, the imaginary structure does not always outperform the meta-learner itself. This observation is not consistent with the results of [22]. We believe this is due to the difficulty of the dataset. Unlike ImageNet1k, our sun-rgbd dataset only has 19 training classes. The may not create the data diversity needed by Imaginary to learn a good generator. This is less of a problem for FATTEN since, unlike unsupervised generators, it uses pose information as extra supervision. In result, it can generate more meaningful augmented data even for a limited meta-training set.

6.3.3 Ablation Study on Meta-learner Structure

Both prototypical and relation network use a feature embedding f_ϕ to map the high-dimensional input features into a low-dimension latent feature space. While the dimensions of this space are usually small (64) for meta-learning, this may not be ideal when the data augmentation module is introduced. Since the synthesized features are 4096 dimensional, we ablate the latent space dimension of the "**GM+M**" configuration for the four dimensions reported in Table 6. All experiments are carried out on test set $\mathcal{T}_1$. These results show that the relation network benefits from a larger dimension, while the prototypical network prefers a lower dimensional embedding. It can also be seen that the prototypical network is much more sensitive to embedding dimension. For fair comparison, we use a 256-D prototypical and a 4096-D relation network in all experiments of Table 5.

6.4 Single-View Reconstruction

We next consider the task of single-view reconstruction.

6.4.1 Dataset

We use the 13 categories of ShapeNet [39] with more than $1,000$ shapes each, and the rendered views from 3D-R2N2 [40]. 80% of the objects from all categories are used for training, and the remaining for testing. Each object has 24 2D images rendered from random camera angles. During testing, the last generated view of each object is used as input. A symmetric Chamfer Distance is used for quantitative evaluation.

6.4.2 Implementation

The BSP-Net is first trained as in [44] and then frozen. A set of shape codes $\{\mathbf{z}_i\}$ is extracted from all 2D views in the training set and a set of ground-truth shape codes $\{\mathbf{z}_{i,\text{gt}}\}$ from the corresponding 3D shapes. For each 2D view, the azimuth angular range of $[0°, 360°]$ is divided into 12 non-overlapping intervals of size $30°$ each. These intervals are

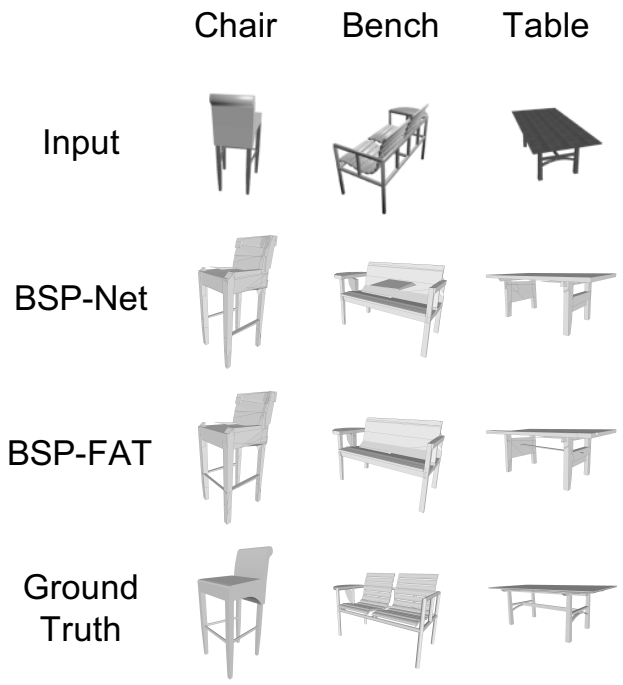


Fig. 6. Examples of 3D reconstruction. A chair, a bench and a table object are shown. For all objects, the figure shows the input image together with the original BSP-Net results, results of BSP-FAT, and the ground truth.

used by the cross-entropy loss for pose training and the one-hot encoding of FATTEN.

A pose training set $\{(\mathbf{z}_i, p_i)\}$ is composed by shape codes and the corresponding discrete pose labels. The pose predictor is trained on $\{(\mathbf{z}_i, p_i)\}$ with a learning rate of 0.01 for 100 epochs. The training set $\{(\mathbf{z}_i, \mathbf{t}_p, \mathbf{z}_{i,\text{gt}})|p = 0, \ldots, 11\}$ of FATTEN, where $\mathbf{t}_p$ is the one-hot encoding of pose p, is composed by all possible combinations of shape codes and target poses. FATTEN is then trained with a learning rate of 0.01 for 200 epochs, using $\lambda = 0.01$ in (22).

During inference, a shape code $\mathbf{z}$ is combined with all possible target poses $\{\mathbf{t}_p|p = 0, \ldots, 11\}$ to obtain a set of synthesized features $\{\mathbf{z}_p|p = 0, \ldots, 11\}$. The average of the features in this set is used by the BSP-Net decoder to reconstruct the 3D shape.

6.4.3 Results

The combination of BSP-Net and FATTEN, denoted BSP-FAT, is compared to Atlas [41], OccNet [42], IM-Net [43], and the baseline BSP-Net model [44]. The pytorch code provided by the authors of the BSP-Net is used to implement the model. This gives slightly different results from those reported in their paper, which are produced with tensorflow code. We report the pytorch results for comparison, and also include the original results for reference.

The overall and category performance are shown in Table 7. BSP-FAT achieves the best overall performance and the best or the second best result in 10 of the 13 categories. When compared to BSP-Net, from which it differs only by addition of FATTEN feature augmentation, it has improved performance in 12 of the 13 categories. This shows that FATTEN can stably improve single-view reconstruction.

6.4.4 Ablations

We first compare different choices of shape code reduction. In addition to average pooling, we test max pooling, an MLP with two fully-connected layers, and an LSTM similar to that of [40]. In the cases of MLP and LSTM, the networks are included in the training. Table 8 shows that none of these choices outperforms average pooling. Further examination of the MSE distance between predicted and ground truth shape code, shows that only average pooling decreases the MSE distance of the original BSP-Net. We next consider how the training MSE loss varies with λ in (22). Table 9 shows that when λ is too large, giving the pose prediction too much weight, the shape code cannot be trained to produce a low MSE error, which is important for good reconstruction performance.

6.4.5 Visualization

Figure 6 shows some typical examples of how pose transfer improves single-view reconstruction. All three objects are reconstructed from a view where part of the object is barely visible or ambiguous. For example, the chair has a footrest beam on the front, of which only a very small portion is visible in the image. The original BSP-Net reconstruction fails to recover this shape feature. This is not the case when FATTEN is used, since the beam can be recovered by pose transfer from other views. Similarly, the table has an underlying beam that is hard to identify in the image, but recovered by FATTEN. For the bench, the input view is such that the right part of the bench can be confused for a small table, which results in the wrong reconstruction by the BSP-Net. FATTEN corrects this problem by pose transfer, leveraging the fact that this ambiguity does not occur in other views. These examples show that the perceptual gains of FATTEN are even larger than the gains in Chamfer distance suggest. Frequently, the reconstructions improve by addition of shape features, such as the footrest above, that can occupy few voxels but are semantically significant.

7 Conclusion

The proposed architecture to data augmentation in feature space, FATTEN, aims to learn trajectories of feature responses, induced by variations in image properties (such as pose). These trajectories can then be easily traversed via *one* learned mapping function which, when applied to instances of novel classes, effectively enriches the feature space by additional samples corresponding to a desired change, *e.g.*, in pose. This "fattening" of the feature space is highly beneficial in situations where the collection of large amounts of adequate training data to cover these variations would be time-consuming, if not impossible. In principle, FATTEN can be used for any kind of desired (continuous) variation, so long as the trajectories can be learned from external data. By discretizing the space of variations, *e.g.*, the rotation angle in case of pose, we also effectively reduce the dimensionality of the learning problem and ensure that the approach scales favorably w.r.t. different resolutions of desired changes. Finally, it is worth pointing out that feature space transfer via FATTEN is not limited to object images; rather, it is a generic architecture in the sense that any variation could, in principle, be learned and transferred.

TABLE 7
Single-view reconstruction results per category and overall. Chamfer Distance is scaled by $1,000$. BSP-Net* denotes results from original paper, BSP-Net those of our implementation. Best result in green, second best the best in red.

Method	airplane	bench	cabinet	car	chair	display	lamp	speaker	rifle	couch	table	phone	vessel	Overall
Atlas	0.587	1.086	1.231	0.799	1.629	1.516	3.858	2.328	1.001	1.471	1.996	1.048	1.179	1.487
OccNet	1.534	3.220	1.099	0.870	1.484	2.171	12.528	2.662	2.015	1.246	3.734	1.183	1.691	2.538
IM-Net	2.211	1.933	1.902	1.390	1.783	2.370	6.387	3.120	2.052	2.344	2.778	2.268	2.385	2.361
BSP-Net*	0.759	1.226	1.188	0.841	1.340	1.856	3.480	2.616	0.888	1.645	1.643	1.383	1.585	1.432
BSP-Net	0.713	1.362	1.184	0.864	1.386	1.929	4.179	2.585	0.881	1.627	1.641	1.536	1.420	1.477
BSP-FAT	0.636	1.251	1.132	0.831	1.274	1.692	4.518	2.398	0.797	1.464	1.499	1.321	1.301	1.391

TABLE 8
Single-view reconstruction with different shape code reduction procedures.

Method	BSP-Net	av pool	max pool	MLP	LSTM
Chamfer	1.477	1.391	1.723	1.456	1.472
MSE	0.0092	0.0083	0.0102	0.0091	0.0092

TABLE 9
MSE loss of the shape code for different λ.

λ	1	0.5	0.1	0.05	0.01
MSE	0.0252	0.0234	0.0155	0.0126	0.0090

REFERENCES

[1] J. Deng, W. Dong, R. S. L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. arXiv:1409.1556v6, 2014.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[5] K. He, X. Zhang, S.Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[6] S. Nene, S. Nayar, and H. Murase, "Columbia object image library," Columbia University, Tech. Rep. CUCS-006-96, 1996.

[7] Y. LeCun, F. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *CVPR*, 2004.

[8] X. Peng, B. Sun, K. Ali, and K. Saenko, "Learning deep object detectors from 3D models," in *ICCV*, 2015.

[9] H. Su, C. Qi, Y. Li, and L. Guibas, "Render for CNN: Viewpoint estimation in images using cnns trained with rendered 3D model views," in *ICCV*, 2015.

[10] R. Socher, M. Ganjoo, C. D. Manning, and A. Y. Ng, "Zero-shot learning through cross-modal transfer," in *NIPS*, 2013.

[11] C. H. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," *TPAMI*, vol. 36, no. 3, pp. 453–465, 2014.

[12] B. Romera-Paredes and P. Torr, "An embarrassingly simple approach to zero-shot learning," in *ICML*, 2015.

[13] K. Tang, M. Tappen, R. Sukthankar, and C. Lampert, "Optimizing one-shot recognition with micro-set learning," in *CVPR*, 2010.

[14] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *NIPS*, 2016.

[15] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *ICML*, 2016.

[16] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *ICLR*, 2017.

[17] M. Dixit, R. Kwitt, M. .Niethammer, and N. Vasconcelos, "AGA: Attribute-guided augmentation," in *CVPR*, 2017.

[18] B. Liu, X. Wang, M. Dixit, R. Kwitt, and N. Vasconcelos, "Feature space transfer for data augmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9090–9098.

[19] S. Thrun, "Lifelong learning algorithms," in *Learning to learn*. Springer, 1998, pp. 181–209.

[20] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1126–1135.

[21] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4077–4087.

[22] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan, "Low-shot learning from imaginary data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7278–7286.

[23] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shape modeling," in *CVPR*, 2015.

[24] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," *CoRR*, vol. arXiv:1612.00593v2, 2016. [Online]. Available: http://arxiv.org/abs/1612.00593

[25] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *ICCV*, 2015.

[26] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3D data," *CoRR*, vol. abs/1604.03265, 2016. [Online]. Available: http://arxiv.org/abs/1604.03265

[27] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000. [Online]. Available: http://science.sciencemag.org/content/290/5500/2323

[28] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003. [Online]. Available: https://doi.org/10.1162/089976603321780317

[29] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-SNE," *JMLR*, vol. 9, pp. 2579–2605, 2008.

[30] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool, "Hough transform and 3D SURF for robust three dimensional classification," in *ECCV*, 2010.

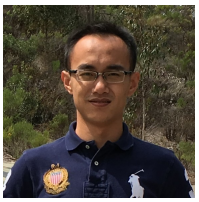
[31] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, "3D shape segmentation with projective convolutional networks," *CoRR*, vol. arXiv:1612.02808v3, 2016. [Online]. Available: http://arxiv.org/abs/1612.02808

[32] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *arXiv preprint arXiv:1606.03657*, 2016.

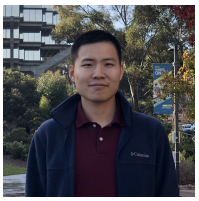
[33] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg, "Transformation-grounded image generation network for novel 3d view synthesis," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017, pp. 3500–3509.

[34] S.-H. Sun, M. Huh, Y.-H. Liao, N. Zhang, and J. J. Lim, "Multi-view to novel view: Synthesizing novel views with self-learned

confidence," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 155–171.
[35] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Multi-view 3d models from single images with a convolutional network," in *European Conference on Computer Vision*. Springer, 2016, pp. 322–337.
[36] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, "View synthesis by appearance flow," in *European conference on computer vision*. Springer, 2016, pp. 286–301.
[37] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson, "Synsin: End-to-end view synthesis from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7467–7477.
[38] R. Tucker and N. Snavely, "Single-view view synthesis with multiplane images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 551–560.
[39] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.
[40] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *European conference on computer vision*. Springer, 2016, pp. 628–644.
[41] T. Groueix, M. Fisher, V. Kim, B. Russell, and M. Aubry, "Atlasnet: a papier-mâché approach to learning 3d surface generation (2018)," *arXiv preprint arXiv:1802.05384*, vol. 11, 2018.
[42] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4460–4470.
[43] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5939–5948.
[44] Z. Chen, A. Tagliasacchi, and H. Zhang, "Bsp-net: Generating compact meshes via binary space partitioning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 45–54.
[45] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 9516–9527.
[46] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=BJgklhAcK7
[47] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1199–1208.
[48] H. Li, D. Eigen, S. Dodge, M. Zeiler, and X. Wang, "Finding task-relevant features for few-shot learning by category traversal," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1–10.
[49] B. Hariharan and R. Girshick, "Low-shot visual recognition by shrinking and hallucinating features," *CoRR*, vol. `arXiv:1606.02819v4`, 2016. [Online]. Available: https://arxiv.org/abs/1606.02819
[50] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4367–4375.
[51] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell, "Few-shot object detection via feature reweighting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8420–8429.
[52] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," *arXiv preprint arXiv:1803.07728*, 2018.
[53] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.
[54] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.
[55] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
[56] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.
[57] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM siggraph computer graphics*, vol. 21, no. 4, pp. 163–169, 1987.
[58] S. Song, S. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D scene understanding benchmark suite," in *CVPR*, 2015.
[59] R. Girshick, "Fast R-CNN," in *ICCV*, 2015.



Bo Liu Bo Liu is currently a PhD candidate at Statistical Visual Computing Lab, Department of Electrical and Computer Engineering at University of California, San Diego. His current research is focused on computer vision and machine learning. Specifically, he is working on few-shot learning, meta-learning and long-tailed recognition.



Xudong Wang Xudong (Frank) Wang is a Ph.D. student in the EECS Department at University of California, Berkeley, co-advised by Prof. Stella X. Yu and Prof. Michael Lustig. He received his Master's Degree in Intelligent Systems, Robotics and Control at University of California, San Diego, advised by Prof. Nuno Vasconcelos, and Bachelor's Degree from Tang Aoqing Honors Program in Science at Jilin University. He was a Staff Researcher at International Computer Science Institute from 2019-2020.

Mandar Dixit Mandar Dixit received his PhD from the University of California at San Diego in 2018. He is currently a researcher in the computer vision technology group at Microsoft, Redmond. His research interests include in large scale visual recognition, transfer learning and meta learning.

Roland Kwitt Roland Kwitt is full professor for machine learning in the Department of Computer Science at the University of Salzburg, Austria. His research spans multiple areas, but mostly focusses on theoretical and practical aspects of learning methods that allow to leverage and control structural characteristics of data.

Nuno Vasconcelos Nuno Vasconcelos received the licenciatura in electrical engineering and computer science from the Universidade do Porto, Portugal, and the MS and PhD degrees from the Massachusetts Institute of Technology. He is a Professor in the Electrical and Computer Engineering Department at the University of California, San Diego, where he heads the Statistical Visual Computing Laboratory. He has received a NSF CAREER award, a Hellman Fellowship, several best paper awards, and has authored more than 150 peer-reviewed publications. He has been Area Chair of multiple computer vision conferences, and is currently an Associate Editor of the IEEE Transactions on PAMI. He is a Fellow of the IEEE.