Feature Space Transfer for Data Augmentation

Bo Liu
University of California, San Diego
boliu@ucsd.edu

Xudong Wang
University of California, San Diego
xuw080@ucsd.edu

Mandar Dixit
Microsoft
madixit@microsoft.com

Roland Kwitt
University of Salzburg, Austria
rkwitt@gmx.at

Nuno Vasconcelos
University of California, San Diego
nuno@ece.ucsd.edu

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. 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.

1. Introduction

Convolutional neural networks (CNNs) trained on large datasets, such as ImageNet [2], have enabled tremendous gains in 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 tasks, by fine tuning. Many recent works have shown that ImageNet trained CNNs, like

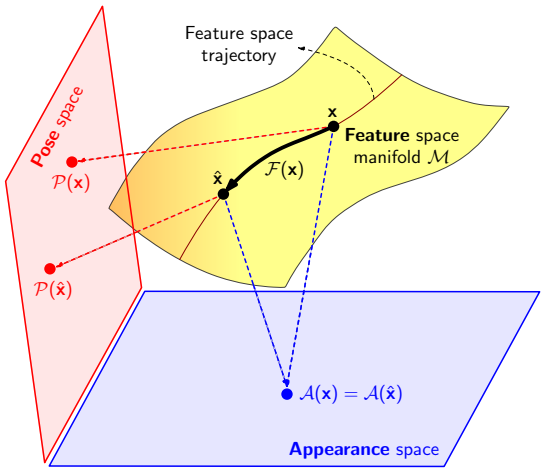

Figure 1: Schematic illustration of *feature space transfer* for variations in *pose*. The input feature $\mathbf{x}$ and transferred feature $\hat{\mathbf{x}}$ are projected to the same point in *appearance space*, but have different mapping points in *pose space*.

AlexNet [14], VGG [27], GoogLeNet [32], or ResNet [9] can be used as feature extractors for the solution of problems as diverse as object detection [6, 23] or generating image captions [12, 35]. 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 dense sampling of the viewing sphere. This, however, is not trivial to accomplish 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 [17], NORB [16], 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 thus 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 by resorting to computer generated images. This has indeed become an established practice to address problems that require multiple object views, such as shape recognition, where synthetic image datasets [19, 31] are routinely used. However, the application of networks trained on synthetic data to real images raises a problem of *transfer learning*. While there is a vast literature on this topic [28, 15, 24, 33, 36, 26, 22], these methods are usually not tailored for the transfer of object poses. In particular, they do not explicitly account for 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 has recently been addressed by [3], who have proposed an attribute guided augmentation (AGA) method to transfer object trajectories along the pose manifold.

Besides learning a classifier that generalizes on target data, the AGA transfer learning system also includes a module that predicts the responses of the model across views. More precisely, given a view of an unseen object, it predicts the model responses to a set of other views of this object. These can then be used to augment the training set of a one-shot classifier, *i.e.*, a classifier that requires a single image per object for training. While this was shown to improve on generic transfer learning methods, AGA has some limitations. For example, it discretizes the pose angle into several bins and learns an *independent* trajectory transfer function between each possible pair of them. While this simplifies learning, the trajectories are not guaranteed to be continuous. Hence, the modeling fails to capture some of the core properties of the pose manifold, such as continuity and smoothness. In fact, a 360° walk around the viewing sphere is not guaranteed to have identical start and finishing feature responses. In our experience, these choices compromise the effectiveness of the transfer.

Contribution. In this work, we propose an alternative, termed *FeATure TransfEr Network (FATTEN)*, that addresses these problems. Essentially, FATTEN is an encoder-decoder architecture, inspired by Fig. 1. We exploit a parametrization of pose trajectories in terms of an *appearance map,* which captures properties such as object color and texture and is constant for each object, and a *pose map,* which is pose dependent. The *encoder* maps the feature responses $\mathbf{x}$ of a CNN for an object image into a pair of appearance $\mathcal{A}(\mathbf{x})$ and pose $\mathcal{P}(\mathbf{x})$ parameters. The *decoder* then takes these parameters plus a *target* pose $\mathbf{t} = \mathcal{P}(\hat{\mathbf{x}})$ and produces the corresponding feature vector $\hat{\mathbf{x}}$. The network is trained end-to-end, using a multi-task loss that accounts for both classification errors and the accuracy of feature transfer across views.

The performance of FATTEN is investigated on two tasks. The first is a *multi-view retrieval* task, where synthesized feature vectors are used to retrieve images by object class and pose. These experiments are conducted on the popular ModelNet [37] shape dataset and show that FATTEN generates features of good quality for applications involving computer graphics imagery. This could be of use for a now large corpus of 3D shape classification works [37, 20, 30, 21], where such datasets are predominant. The *second* task is transfer learning. We compare the performance of the proposed architecture against both general purpose transfer learning algorithms and the AGA procedure. Our results show that there are significant benefits in developing methods explicitly for trajectory transfer, and in forcing these methods to learn continuous trajectories in the pose manifold. The FATTEN architecture is shown to achieve state-of-the-art performance for pose transfer.

Organization. In Sect. 2, we review related work; Sect. 3 introduces the proposed FATTEN architecture. Sect. 4 presents experimental results on ModelNet and SUN-RGBD and Sect. 5 concludes the paper with a discussion of the main points and an outlook on open issues.

2. Related Work

Since objects describe smooth trajectories in image space, as a function of viewing angle, it has long been known that such trajectories span a 3D manifold in image space, parameterized by the viewing angle. Hence, many of the manifold modeling methods proposed in the literature [25, 1, 34] 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.

One of the main goals of trajectory transfer is to "fatten" a feature space, by augmenting a dataset with feature responses of unseen object poses. In this sense, the problem is related to extensive recent literature on GANs [7], which have been successfully used to generate images, image-to-image translations [10], inpainting [18] or style-transfer [5]. While our work uses an encoder-decoder architecture, which is fairly common in the GAN-based image generation literature, we aim for a different goal of generating CNN feature responses. This prevents access to a dataset of "real" feature responses across the pose manifold, since these are generally unknown. While an ImageNet CNN could be used to produce some features, the problem that we are trying to solve is exactly the fact that ImageNet CNNs do not effectively model the pose manifold. Hence, the GAN formalism of learning to match a "real" distribution is not easily applicable to trajectory transfer.

Instead, trajectory transfer is more closely related to the topic of transfer learning, where, now, there is extensive work on problems such as zero-shot [28, 15, 24] or n-shot [33, 36, 26, 22] learning. However, these methods tend to be of general purpose. In some cases, they exploit generic semantic properties, such as attributes or affordances [15, 24], in others they simply rely on generic machine learning for domain adaptation [28], transfer learning [36] or, more recently, meta-learning [26, 4, 22]. None of these methods exploits specific properties of the pose manifold, such as the parametrization in Fig. 1. The introduction of networks that enforce such parameterizations is a form of regularization that improves on the transfer performance of generic procedures. This was shown in the AGA work [3] and is confirmed by our results, which show even larger gains over very recent generic methods, such as feature hallucination as proposed in [8].

Finally, trajectory transfer is of interest for problems involving multi-view recognition. Due to the increased cost of multi-view imaging, these problems frequently include some degree of learning from computer generated images. This is, for example, an established practice in the shape recognition literature, where synthetic image datasets [19, 31] are routinely used. The emergence of these artificial datasets has enabled a rich literature in shape recognition methods [13, 37, 30, 20, 21, 11] and already produced some interesting conclusions. For example, while many representations have been proposed, there is some evidence that the problem could be solved as one of multi-view recognition, using simple multi-view extensions of current CNNs [30]. It is not clear, however, how these methods or conclusions generalize to real world images. Our results show that feature trajectory transfer models, such as FATTEN, learned on synthetic datasets, such as ModelNet [37], can be successfully transferred to real image datasets, such as SUN-RGBD [29].

3. The FATTEN architecture

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

3.1. Motivation

In this work, we assume the availability of a training set with pose annotations, *i.e.*, $\{(\mathbf{x}_n, \mathbf{p}_n, y_n)\}_n$, 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 problem is to learn the feature 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 FATTEN Architecture

The FATTEN architecture is inspired by Fig. 1, which depicts the manifold spanned by an object under pose variation. The manifold $\mathcal{M}$ is embedded in $\mathbb{R}^D$ and is parameterized by two variables. The first, is an *appearance descriptor* $\mathbf{a} \in \mathbb{R}^A$ that captures object properties such as color or texture. This parameter is pose invariant, *i.e.*, it has the same value for all points on the manifold. It can be thought of as an object identifier that distinguishes the manifold spanned by one object from those spanned by others. However, appearance is not the same as category. Some objects of different categories can have more similar appearance than objects of the same category). The second is a *pose descriptor* $\mathbf{p} \in \mathbb{R}^N$ that characterizes the point $\mathbf{x}$ on the manifold that corresponds to a particular pose $\mathbf{p}$. Conceptually, feature points $\mathbf{x}$ could be thought of as the realization of a mapping

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

The FATTEN architecture models the relationship between the feature vectors extracted from object images and the associated appearance and pose parameters. As shown

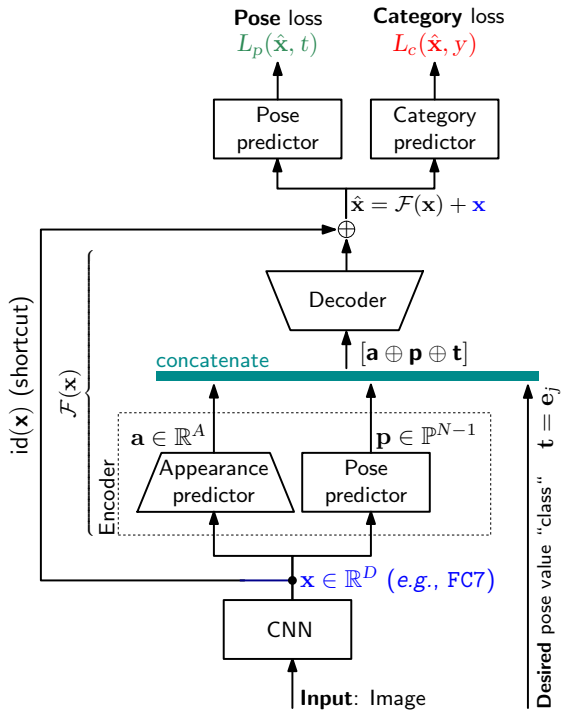

Figure 2: The FATTEN architecture. Here, id denotes the identity shortcut connection, D the dimensionality of the input feature space, A the dimensionality of the appearance space and $\mathbb{P}^{N-1}$ the $N-1$ probability simplex. Both pose predictors are pre-trained and share parameters.

in Fig. 2, it is an encoder/decoder architecture. The encoder essentially aims to invert the mapping of Eq. (1). Given a feature vector $\mathbf{x}$, it produces an estimate of the appearance $\mathbf{a}$ and pose $\mathbf{p}$ parameters. This is complemented with a *target* pose parameter $\mathbf{t}$, which specifies the pose associated with a desired feature vector $\hat{\mathbf{x}}$. This feature is then generated by a decoder that that operates on the concatenation of $\mathbf{a}$, $\mathbf{p}$ and $\mathbf{t}$. While, in principle, it would suffice to rely on $\hat{\mathbf{x}} = \phi(\mathbf{a}, \mathbf{t})$, *i.e.*, to use the inverse of the encoder as a decoder, we have obtained the best results with the following modifications.

First, to discourage the encoder/decoder pair from learning a mapping that simply "matches" feature pairs, FATTEN implements the residual learning paradigm of [9]. 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 mappings that explicitly recover the appearance $\mathbf{a}$ and pose $\mathbf{p}$ are used instead of a single monolithic encoder. This facilitates learning, since the pose predictor can be learned with full supervision. *Third*, a vector encoding is used for the source $\mathbf{p}$ and target $\mathbf{t}$ parameters, instead of continuous values. This makes the dimensionality of the pose parameters closer to that of the appearance parameter, enabling a more balanced learning problem. We have found that, otherwise, the learning algorithm can have a tendency to ignore the pose parameters and produce a smaller diversity of target feature vectors. *Finally*, rather than a function of $\mathbf{a}$ and $\mathbf{t}$ alone, the decoder is a function of $\mathbf{a}$, $\mathbf{p}$, and $\mathbf{t}$. This again guarantees that the *intermediate* representation is higher dimensional and facilitates the learning of the decoder. We next discuss the details of the various network modules.

3.3. Network 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 feature vectors $\mathbf{x}$ to pose parameters. The poses are first internally mapped 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 pose space is quantized into N cells of centroids $\mathbf{m}_i$. Each pose is then assigned to the cell of the nearest representative $\mathbf{m}^*$ and represented by a N-dimensional one-hot encoding that identifies $\mathbf{m}^*$. The pose mapping $\mathcal{P}$ is finally implemented with a 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 $\mathbb{P}^{N-1}$. This is implemented with 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 feature vectors $\mathbf{x}$ to appearance descriptors $\mathbf{a}$. This is realized 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. Assuming that this pose belongs to the cell of centroid $\mathbf{m}_j$, this is $\mathbf{t} = \mathbf{e}_j$, where $\mathbf{e}_j$ is a vector of all zeros with a 1 at position j.

Decoder. The decoder maps the vector of concatenated appearance and pose parameters[1], *i.e.*,

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

into the residual $\hat{\mathbf{x}} - \mathbf{x}$. 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 is then added to the input $\mathbf{x}$ to produce the target $\hat{\mathbf{x}}$.

Although conceptually similar, our architecture is different from AGA [3] and solves some key limitations of the latter. In particular, the feature synthesis function $f(x, p, t)$

[1] $\oplus$ denotes vector concatenation.

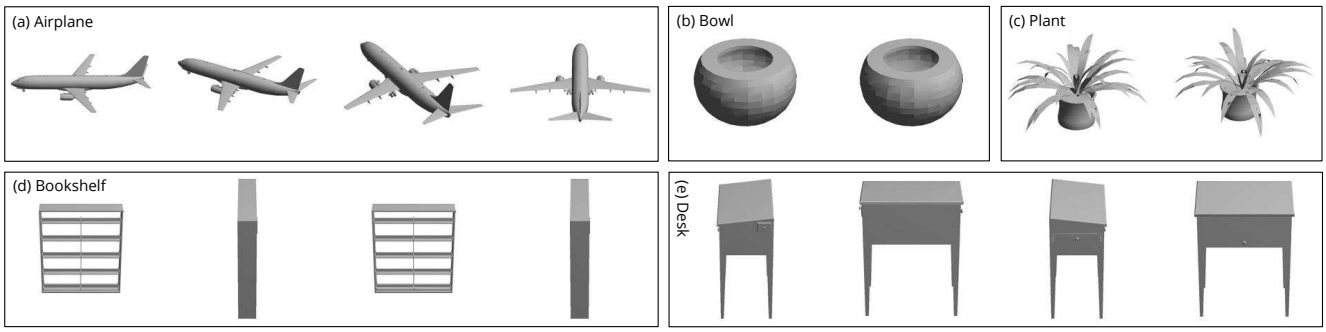

Figure 3: Exemplary ModelNet [37] views: (a) different views of one object (airplane); (b)-(c) different views of two *symmetric* objects (bowl, plant); (d)-(e) four views (bookshelf, desk) with 90 degrees difference.

of AGA is implemented as a series of encoder-decoder modules $f_{p,t}(x)$, one for each (p, t). The number of these functions grows exponentially and AGA needs to learn a different f for each $p \rightarrow t$ and $t \rightarrow p$; there is no provision to share information. As we train only a *single* network for 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; we use a category loss which is certainly a better choice.

3.4. Training

The network is trained *end-to-end* to optimize a multi-task loss that accounts for two goals. The *first* goal is that the generated feature vector $\hat{\mathbf{x}}$ indeed corresponds to the desired pose $\mathbf{t}$. This is measured by the pose loss, which is the cross-entropy loss commonly used for classification, *i.e.*,

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

where $\rho_j(v) = \frac{e^{v_j}}{\sum_k e^{v_k}}$ is the softmax function and j is the non-zero element of the one-hot vector $\mathbf{t} = \mathbf{e}_j$. Note that, as shown in Fig. 2, this requires passing the target feature vector $\hat{\mathbf{x}}$ through the pose predictor $\mathcal{P}$. It should be emphasized that this is only needed during training, albeit the loss in Eq. (6) can also be measured during inference, since the target pose $\mathbf{t}$ is known. This can serve as a diagnostic measure for the performance of FATTEN.

The *second* goal is that the generated feature vector $\hat{\mathbf{x}}$ is assigned the same class label y as the source vector $\mathbf{x}$. This encourages the generation of features with high recognition accuracy on the original object recognition problem. Recognition accuracy depends on the network used to extract the feature vectors, denoted as CNN in Fig. 2. Note that this network can be fine-tuned for operation with the FATTEN module in an end-to-end manner. While FATTEN can, in principle, be applied to any such network, our implementation is based on the VGG16 model of [27]. More specifically, we rely on the `fc7` activations of a fine-tuned VGG16 network as source and target features. The category predictor of Fig. 2 is then the `fc8` layer of this network. The accuracy of this predictor is measured by 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 this network. The *multi-task loss* is then defined as

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

In general, it is beneficial to pre-train the pose predictor $\mathcal{P}(\mathbf{x})$ and *embed* it into the encoder-decoder structure. This reduces the number of degrees of freedom of the network, and minimizes the ambiguity inherent to the fact that a given feature vector could be consistent with multiple pairs of pose and appearance parameters. For example, while all feature vectors $\mathbf{x}$ extracted from views of the same object should be constrained to map into the same appearance parameter value $\mathbf{a}$, we have, so far, felt no need to enforce such constraint. This endows the network with robustness to small variations of the appearance descriptor, due to occlusions, etc. Furthermore, when a pre-trained pose predictor is used, *only* the weights of the encoder/decoder need to be learned. 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. Experiments

We first train and evaluate the FATTEN model on the artificial ModelNet [37] dataset (Sec. 4.1), and then assess its feature augmentation performance on the one/few-shot object recognition task introduced in [3] (Sec. 4.2).

4.1. ModelNet

Dataset. ModelNet [37] is a 3D artificial data set with 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

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

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

rendering strategy of [30]. 12 virtual cameras are placed around the object, in increments of 30 degrees along the z-axis, and 30 degrees above the ground. Several rendered views are shown in Fig. 3. The training and testing division is the same as in the ModelNet benchmark, using 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)). For training, these objects are eliminated and the remaining 28 object categories are used.

Implementation. To verify the generality of our approach, both VGG16 [27] and ResNet-101 [9] are adopted as backbone architectures in feature transfer experiments, while only VGG16 is used in others. All feature vectors $\mathbf{x}$ are collected from activations of the last fully-connected layer of fine-tuned networks, 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. Any given angle is then converted to a classification label based on the interval it belongs to.

4.1.1 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% in VGG16 and 85.8% in ResNet-101.

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 96.2% and the category prediction accuracy of 83.65%. Further, pose prediction error as well as pose and category accuracy of generated features on ModelNet show similar performance w.r.t. both backbones.

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

Table 2: Retrieval performance in mAP [%] of real and generated features, on the testing portion of ModelNet, for distance functions d_1, d_2 and d_c, see Sec. 4.1.2.

4.1.2 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

$$d_1(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_2 \tag{8}$$

is a sensible similarity measure between $\mathbf{x}$ and $\mathbf{y}$ 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

$$d_2(\mathbf{x}, \mathbf{y}) = ||\gamma(\mathbf{x}) - \gamma(\mathbf{y})||_2 \ . \tag{9}$$

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

$$d_c(\mathbf{x}, \mathbf{y}) = d_1(\mathbf{x}, \mathbf{y}) + \lambda d_2(\mathbf{x}, \mathbf{y}) \ . \tag{10}$$

All queries and instances to be retrieved are based on *generated* features from the *testing* corpus of ModelNet. For each generated feature, three queries are performed: (1) *Category*, (2) *Pose*, and (3) *Category & Pose*. This is compared to the performance, on the same experiment, of the real features extracted from the testing corpus.

Retrieval results are listed in Table 2 and some retrieval examples are shown in Fig. 4. The generated 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 generated features on *category retrieval* and the combination of both is comparatively low. However, the performance of 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 Eq. (10) 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.

4.2. Few-shot object recognition

The experiments above provide no insight on whether FATTEN generates meaningful features for tasks involving real world datasets. 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 [29], following the setup of [3].

Dataset. The whole SUN-RGBD dataset contains 10335 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. To allow a fair comparison with AGA, however, during testing, we restrict the desired pose t to take the values $45°$, $75°$, ..., $180°$, prescribed in [3]. This is mainly to ensure that our system generates 11 synthetic points along the *Depth* trajectory and 7 along the *Pose* trajectory. The first 5335 images of SUN-RGBD are used for training and the remaining 5000 images for testing. However, if only ground truth bounding boxes are used for object extraction, the instances are neither balanced w.r.t. categories, nor w.r.t. pose/depth values. To remedy this issue, a fast R-CNN [6] object detector is fine-tuned on the dataset and the selective search proposals with IoU > 0.5 (to ground truth boxes) and detection scores > 0.7 are used to extract object images for training. As this strategy produces a sufficient 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.

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

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

Evaluation is based on the source and target object classes in [3]. We denote $\mathcal{S}$ as the source dataset, and let $\mathcal{T}_1$ and $\mathcal{T}_2$ denote two different (disjoint) target datasets; further, $\mathcal{T}_3 = \mathcal{T}_1 \cup \mathcal{T}_2$ denotes a third dataset that is a union of the first two. 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 overlap with any $\mathcal{T}_i$ which ensurers that FATTEN has no access to shared knowledge between training/testing images or classes.

Implementation. The attribute predictors for *pose* and *depth* are trained with a learning rate of 0.01 for 1000 epochs. The feature transfer network is fine-tuned, starting from the weights obtained from the ModelNet experiment of Sec. 4.1, 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 transfer 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 now *augmented* ("fattened") feature set (source and target features).

4.2.1 Results

Table 4 lists the averaged one-shot (and five-shot) recognition accuracies (over 500 random runs) for all three evaluation sets $\mathcal{T}_i$. Additionally, recognition accuracies of two recently proposed data augmentation works, *i.e.*, feature hallucination [8] and AGA [3] are also reported. 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 effec-

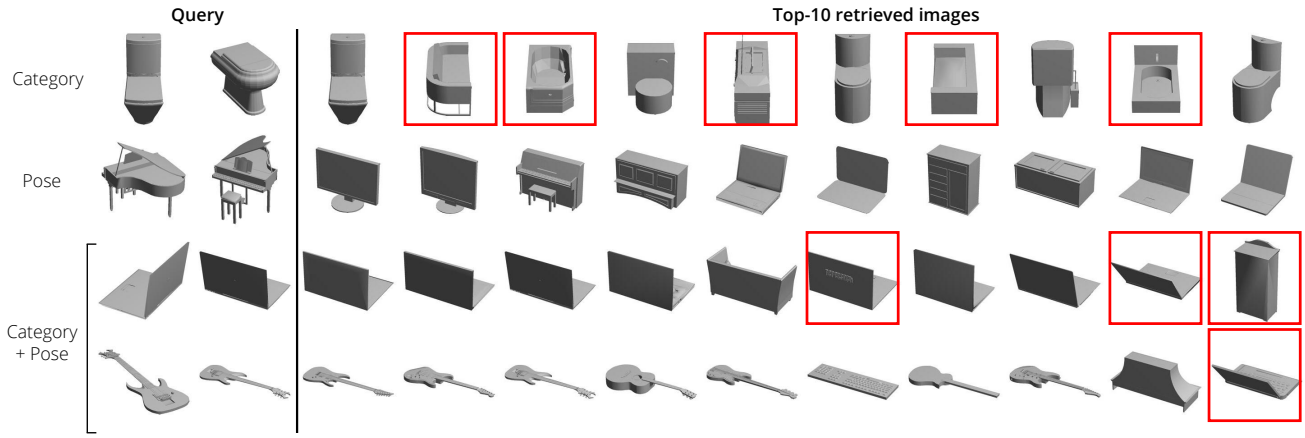

Figure 4: Exemplary retrieval results for the experiments of Sec. 4.1.2. Lines are annotated by the retrieval *type* and errors are highlighted in red. In the **query** part, *(left)* shows the original image, *(right)* shows the original image corresponding to the generated feature.

tively "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 [8]. 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 remarkably well in the ImageNet1k low-shot setup of [8], 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 [8]) 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 [8]. *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 [8] *vs*. SUN-RGBD here). Without clear guidelines of how to set this parameter, it seems challenging to adjust it appropriately. *Third*, all results of [8] 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.

		Baseline	Hal. [8]	AGA [3]	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

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.

5. Discussion

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.

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003. 2

[2] 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. 1

[3] M. Dixit, R. Kwitt, M. .Niethammer, and N. Vasconcelos. AGA: Attribute-guided augmentation. In *CVPR*, 2017. 2, 3, 4, 5, 7, 8

[4] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, `arXiv:1703.03400v3`, 2017. 3

[5] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 3

[6] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 7

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 3

[8] B. Hariharan and R. Girshick. Low-shot visual recognition by shrinking and hallucinating features. *CoRR*, `arXiv:1606.02819v4`, 2016. 3, 7, 8

[9] K. He, X. Zhang, S.Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 4, 6

[10] P. Isola, J. Zhu, T. Zhou, and A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 3

[11] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri. 3D shape segmentation with projective convolutional networks. *CoRR*, `arXiv:1612.02808v3`, 2016. 3

[12] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015. 1

[13] 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. 3

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[15] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *TPAMI*, 36(3):453–465, 2014. 2, 3

[16] Y. LeCun, F. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*, 2004. 2

[17] S. Nene, S. Nayar, and H. Murase. Columbia object image library. Technical Report CUCS-006-96, Columbia University, 1996. 2

[18] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 3

[19] X. Peng, B. Sun, K. Ali, and K. Saenko. Learning deep object detectors from 3D models. In *ICCV*, 2015. 2, 3

[20] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. *CoRR*, `arXiv:1612.00593v2`, 2016. 2, 3

[21] 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*, abs/1604.03265, 2016. 2, 3

[22] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017. 2, 3

[23] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection. In *NIPS*, 2015. 1

[24] B. Romera-Paredes and P. Torr. An embarrassingly simple approach to zero-shot learning. In *ICML*, 2015. 2, 3

[25] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000. 2

[26] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016. 2, 3

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, `arXiv:1409.1556v6`, 2014. 1, 5, 6

[28] R. Socher, M. Ganjoo, C. D. Manning, and A. Y. Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, 2013. 2, 3

[29] S. Song, S. Lichtenberg, and J. Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *CVPR*, 2015. 3, 7

[30] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *ICCV*, 2015. 2, 3, 6

[31] 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. 2, 3

[32] 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. 1

[33] K. Tang, M. Tappen, R. Sukthankar, and C. Lampert. Optimizing one-shot recognition with micro-set learning. In *CVPR*, 2010. 2, 3

[34] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-SNE. *JMLR*, 9:2579–2605, 2008. 2

[35] O. Vinjays, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015. 1

[36] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *NIPS*, 2016. 2, 3

[37] 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. 2, 3, 5