AGA : Attribute-Guided Augmentation

Anonymous CVPR submission

Paper ID ****

Abstract

We consider the problem of data augmentation, i.e., generating artificial samples to extend a given corpus of training data. Specifically, we propose attributed-guided augmentation (AGA) which learns a mapping that allows to synthesize data such that an attribute of a synthesized sample is at a desired value or strength. This is particularly interesting in situations where little data with no attribute annotation is available for learning, but we have access to a large external corpus of heavily annotated samples. While prior works primarily augment in the space of images, we propose to perform augmentation in feature space directly. We implement our approach as a deep encoder-decoder architecture that learns the synthesis function in an end-to-end manner. We demonstrate the utility of our approach on the problem of one-shot object recognition in a transfer-learning setting where we have no prior knowledge of the new categories, except for a single training sample. As external data, we leverage 3D depth and pose information from the SUN RGB-D dataset and show that attribute-guided augmentation of high-level CNN features substantially improves one-shot object recognition performance.

1. Introduction

Convolutional Neural networks (CNNs), trained on large scale data, have significantly advanced the state-of-the-art on traditional vision problems such as object recognition [19, 27, 31] and object detection [13, 24]. Success of these networks is mainly due to their high selectivity for semantically meaningful visual concepts, *e.g.*, objects and object parts [26]. In addition to ensuring good performance on the problem of interest, this property of CNNs also allows for *transfer* of knowledge to several other vision tasks [9, 14, 5, 7]. The object recognition network of [19], *e.g.*, has been successfully used for object detection [13, 24], scene classification [14, 7], texture classification [5] and domain adaptation [9], using various transfer mechanisms.

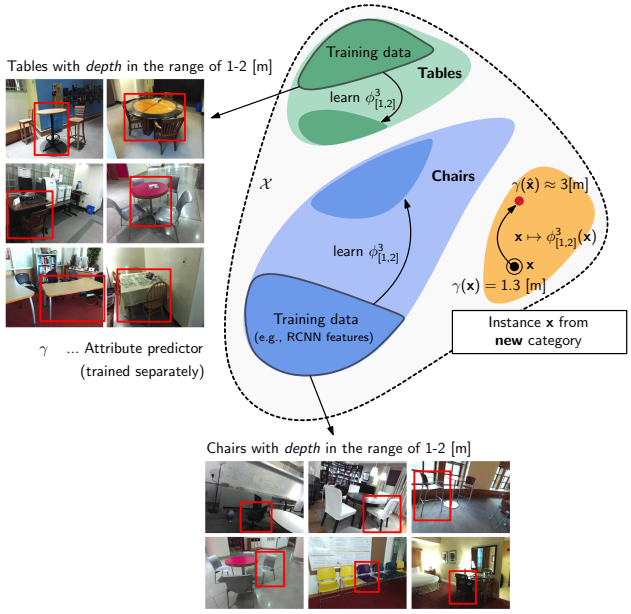

Figure 1: Given a predictor $\gamma : \mathcal{X} \rightarrow \mathbb{R}_+$ of attribute strength (*e.g.*, for depth or pose), we propose to *learn* a mapping of object features $\mathbf{x} \in \mathcal{X}$, such that (1) the mapped feature is "close" to $\mathbf{x}$ (to preserve object identity) and (2) the predicted attribute value $\gamma(\hat{\mathbf{x}}) = \hat{t}$ of mapped features $\phi(\mathbf{x}) = \hat{\mathbf{x}}$ matches a given target value t. In this example, we learn a mapping for features with associated *depth* values in the range of 1-2 [m] to $t = 3$ [m] and apply that mapping to an instance of a new object category. In our approach, this mapping is learned in an *object-agnostic* manner. In our example, this means that *all* training data from chairs and tables is used to learn ϕ.

CNN-based transfer is generally achieved either by *fine-tuning* a pre-trained network such as in [19] on a new image dataset or by designing a new image representation on such a dataset based on the activations of the pre-trained network layers [9, 14, 7, 5]. Recent proposals of transfer have shown highly competitive performance on different predictive tasks with a modest amount of new data (as few as 50 images per class). The effectiveness of transfer based methods, however, has not yet been tested under more severe constrains such as in a *few shot* or a *one shot* learn-

ing scenario. In these problems, the amount of examples available for learning may be as few as one per category. Finetuning a pre-trained CNN with millions of parameters to such inadequate datasets is clearly not a viable option. A one-shot classifier trained on CNN activations will also be prone to over-fitting due to the high dimensionality of the feature space. The only way to solve the problem of limited data is to *augment* the training corpus by generating more examples for the given classes.

While augmentation techniques can be as simple as flipping, rotating, adding noise, or extracting random crops from images [19, 4, 34], *task-specific*, or *guided* augmentation strategies [3, 15, 25, 23] have the potential to generate more realistic synthetic samples. This is a particularly important issue, since performance of CNNs heavily relies on sufficient coverage of the variability that we expect in previously unseen data. In scene recognition, *e.g.*, we desire sufficient variability in the constellation and transient states of scene categories, whereas in object recognition, we desire variability in the specific incarnations of certain objects, different lighting conditions, pose, or depth, just to name a few. Unfortunately, this variability is often dataset-specific and can cause substantial bias in recognition results [32].

An important observation in the context of our work is that augmentation is typically performed on an image, or video level. While this is not a problem with simple techniques, such as flipping or cropping, it can become computationally expensive if more elaborate augmentation techniques are used. We argue that, in specific problem settings, augmentation might as well be performed in *feature space*, especially in situations where features are input to subsequent learning algorithms. This is common, *e.g.*, in recognition tasks, where the softmax output of trained CNNs is often not used directly, but activations at earlier layers are input to an external discriminant classifer.

Contribution. In this work, we propose an approach to augment the training set with *feature descriptors* instead of images. Specifically, we introduce an augmentation technique that learns to synthesize features, guided by desired values for a set of available object attributes, such as depth or pose. An illustration of this concept is shown in Fig. 1. We first train a fast RCNN [13] detector to identify objects in 2D images. This is followed by training a neural network regressor which predicts the 3D attributes of a detected object, namely its depth from the camera plane and pose. An encoder-decoder network is then trained which, for a detected object at a certain depth and pose, will hallucinate(← Is hallucinate standard terminology here?) the changes in its RCNN features for different depths/poses. Using this architecture, for a new image, we are able to augment existing feature descriptors by an auxiliary set of features that correspond to the object changing its 3D position. Since our framework relies on object attributes to guide the augmentation process, we refer to it as *attribute-guided augmentation (AGA)*.

Organization. Sec. 2 reviews related work on data augmentation. Sec. 3 introduces the proposed encoder-decoder architecture for attribute-guided augmentation. Sec. 4 studies the building blocks of this approach in detail and demonstrates that AGA in feature space considerably improves one-shot object recognition performance on previously unseen objects. Sec. 5 concludes the paper with a discussion and an outlook on potential future directions.

2. Related work

Our review of related work primarily focuses on *data augmentation* strategies. While many techniques have been proposed in the context of training deep neural networks to avoid overfitting and to increase variability in the data, other (sometimes closely related) techniques have previously appeared in the context of one-shot and transfer learning. We can roughly group existing techniques into (1) *generic*, computationally cheap approaches and (2) task-specific, or guided approaches that are typically more computationally involved.

As a representative of the first group, Krizhevsky *et al.* [19] leverage a set of label-preserving transformations, such as patch extraction + reflections, and PCA-based intensity transformations, to increase training sample size. Similar techniques are used by Zeiler and Fergus [34]. In [4], Chatfield and Zisserman demonstrate that the augmentation techniques of [19] are not only beneficial for training deep architectures, but shallow learning approaches equally benefit from such *simple* and *generic* schemes.

In the second category of guided-augmentation techniques, many approaches have recently been proposed. In [3], *e.g.*, Charalambous and Bharath employ guided-augmentation in the context of gait recognition. The authors suggest to simulate synthetic gait video data (obtained from from avatars) with respect to various confounding factors (such as clothing, hair, etc.) to extend the training corpus. Similar in spirit, Rogez and Schmid [25] recently proposed an image-based synthesis engine for augmenting existing 2D human pose data by photorealistic images with greater pose variability. This is done by leveraging 3D motion capture (MoCap) data. 3D data, in the form of synthetic CAD models, is used by Peng *et al.* [23] to render synthetic images of objects (with varying pose, texture, background) that are then used to train CNNs for object detection. It is shown that synthetic data is beneficial, especially in situations where few (or no) training instances are available, but 3D CAD models are. Su *et al.* [30] follow a similar pipeline of rendering images from 3D models for 3D viewpoint estimation, however, with substantially more synthetic data ob-

tained, *e.g.*, by additionally deforming existing 3D models before rendering.

Another (data-driven) guided augmentation technique is introduced by Hauberg *et al*. [15]. The authors propose to *learn* class-specific transformations from external training data, instead of manually specifying transformations as in [19, 34, 4]. The learned transformations are then applied to the samples of each class. Specifically, diffeomorphisms are learned from data and encouraging results are demonstrated in the context of digit recognition on MNIST. Notably, this strategy is conceptually similar to earlier work by Miller *et al*. [21] on one-shot learning, where the authors synthesize additional data for digit images via an iterative process, called *congealing*. During that process, external images of a given category are aligned by optimizing over a class of geometric transforms (*e.g.*, affine transforms). These transformations are then applied to single instances of the new categories to increase data for one-shot learning.

Marginally related to our work, we remark that alternative approaches to implicitly learn spatial transformations have been proposed. For instance, Jaderberg *et al*. [17] introduce *spatial transformer* modules that can be injected into existing deep architectures to implicitly capture spatial transformations inherent in the data, thereby improving invariance to this class of transformations.

While *all* previously discussed methods essentially propose *image-level* augmentation to train CNNs, our approach is different in that we perform augmentation in *feature space*. Along these lines, the approach of Kwitt *et al*. [20] is conceptually similar to our work. In detail, the authors suggest to learn how features change as a function of the strength of certain transient attributes (such as sunny, cloudy, or foggy) in a scene-recognition context. These models are then transfered to previously unseen data for one-shot recognition. However, different to our approach, the learned models are simple linear regressors and learning is done in a *scene-class specific* manner. In contrast, we learn deep non-linear models in a *class-agnostic* manner which enables straightforward application to object recognition, without the requirement of a direct relation of new classes to classes in the external training data.

3. Architecture

Notation. To describe our architecture, we let $\mathcal{X}$ denote our feature space, $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^D$ denotes a feature descriptor (*e.g.*, a representation of an object) and $\mathcal{A}$ denotes a set of attributes available in the external training corpus. Further, we let $s \in \mathbb{R}_+$ denote the strength of an attribute $A \in \mathcal{A}$, associated with $\mathbf{x}$. We assume (1) that this attribute can be predicted by an attribute regressor $\gamma : \mathcal{X} \rightarrow \mathbb{R}_+$ and (2) that it is possible that its range can be divided into T intervals $[l_i, h_i]$, where l_i, h_i denote the lower and upper bounds of the i-th interval.

Objective. On a conceptual level, we construct a mapping function ϕ which, given a desired attribute strength t for some attribute A, transforms input features $\mathbf{x} \in \mathcal{X}$ such that the attribute strength changes in a controlled manner to a desired target value t. More formally, we aim to learn

$$\phi : \mathcal{X} \times \mathbb{R}_+ \rightarrow \mathcal{X}, (\mathbf{x}, t) \mapsto \hat{\mathbf{x}}, \quad \text{s.t.} \quad \gamma(\hat{\mathbf{x}}) \approx t \ . \quad (1)$$

Since, the formulation in Eq. (1) is overly generic, we constrain the problem to the case where we learn different ϕ_i^k for a selection of intervals $[l_i, h_i]$(← Should this be $t \in [l_i, h_i]$? What exactly are these intervals of?) and a selection of K given target attribute values t_k(← I am not following this. Are you saying that each interval has a fixed, constant target attribute value?). While this simplifies the problem, it requires a good a-priori *attribute predictor*, since, otherwise, we could not decide which ϕ_i to use(← What is the difference between ϕ_i and ϕ_i^k?). During testing, we (1) predict the attribute strength, *i.e.*, $\gamma(\mathbf{x}) = \hat{t}$, and then (2) synthesize features as $\hat{\mathbf{x}} = \phi_i^k(\mathbf{x})$ for $k = 1, \ldots, T$ if $\hat{t} \in [l_i, h_i]$(← I do not understand how the predicted value $\hat{t}$ is used in feature synthesis. Is the goal not to synethesize features for all attributes?). Next, we discuss each component of this architecture in detail.

3.1. Attribute regression

An essential part of our architecture is the attribute regressor $\gamma : \mathcal{X} \rightarrow \mathbb{R}_+$ for a given attribute A. This regressor takes as input a feature $\mathbf{x}$ and predicts its strength or value, *i.e.*, $\gamma(\mathbf{x}) = \hat{t}$. While γ could, in principle, be implemented by a variety of approaches, such as support vector regression [10] or Gaussian processes [2], we use a two-layer neural network instead, to accomplish this task. This is not an arbitrary choice, as it will later enable us to easily re-use this building block in the learning stage of the mapping function(s) ϕ_i^k. The architecture of the attribute regressor is shown in Fig. 2, consisting of two linear layers, interleaved by batch normalization [16] and rectified linear units (ReLU) [22]. While this architecture is admittedly simple, adding more layers did not lead to significantly better results in our experiments. Nevertheless, the design of this component is problem-specific and could easily be replaced by more complex variants, depending on the characteristics of the attributes that need to be predicted.

Learning. The attribute regressor can easily be trained from a collection of N training tuples $\{(\mathbf{x}_i, s_i)\}_{i=1}^N$. We remark that for this part of the architecture, we do not need any binning of the attribute range.(← I am not following why binning is needed at all.)

3.2. Feature regression

To implement ϕ, we design an encoder-decoder architecture, reminiscent of a conventional autoencoder [1]. Our

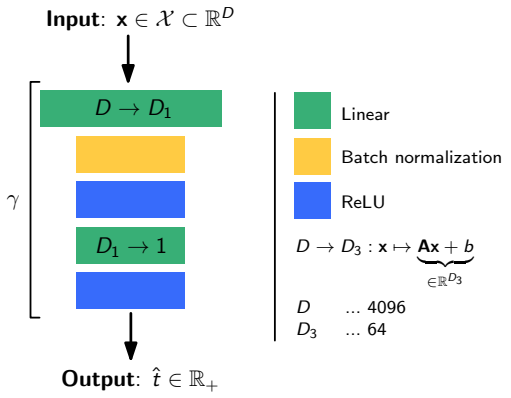

Figure 2: Architecture of the attribute regressor γ.(← I do not understand the symbols in this graph.)

objective, however, is not to encode and then reconstruct the input, but to produce an output that resembles a feature descriptor of an object at a desired attribute value.

In other words, the *encoder* essentially learns to extract the essence of features; the *decoder* then takes the encoding result and decodes it to the desired result. In general, we can formulate the optimization problem as

$$\min_{\phi \in \mathcal{C}} L(\mathbf{x}, t; \phi) = (\gamma(\phi(\mathbf{x})) - t)^2 \quad , \tag{2}$$

where the minimization is over a suitable class of functions $\mathcal{C}$. Notably, when implementing ϕ as an encoder-decoder network with an appended (pre-trained) attribute predictor (see Fig. 3) and loss $(\gamma(\phi(\mathbf{x})) - t)^2$, we have little control over the decoding results in the sense that we cannot guarantee that the *identity* of the input is preserved. This means that features from a particular class of objects might map to features that are no longer recognizable as this class, as the encoder-decoder will *only* learn to "fool" the attribute predictor γ. For that reason, we add a *regularizer* to the objective of Eq. (2), *i.e.*, we require the decoding result to be close, *e.g.*, in the L_2 norm, to the input. This changes the optimization problem to

$$\min_{\phi \in \mathcal{C}} L(\mathbf{x}, t; \phi) = \underbrace{(\gamma(\phi(\mathbf{x})) - t)^2}_{\text{Target mismatch}} + \lambda \underbrace{\|\phi(\mathbf{x}) - \mathbf{x}\|^2}_{\text{Regularizer}} \quad . \tag{3}$$

Interpreted differently, this resembles the loss of an autoencoder network with an added *target attribute mismatch* term. The encoder-decoder network that implements the function class $\mathcal{C}$ to learn ϕ is shown Fig. 3. The core building block is a combination of a linear layer, batch normalization, ELU [6], followed by dropout [29]. After the final linear layer, we add one ReLU layer to enforce $\hat{\mathbf{x}} \in \mathbb{R}^D_+$.

Learning. Training the encoder-decoder network of Fig. 3 first requires a pre-trained attribute regressor γ for each given attribute $A \in \mathcal{A}$. During training, this attribute regressor is appended to the network and its weights are frozen.

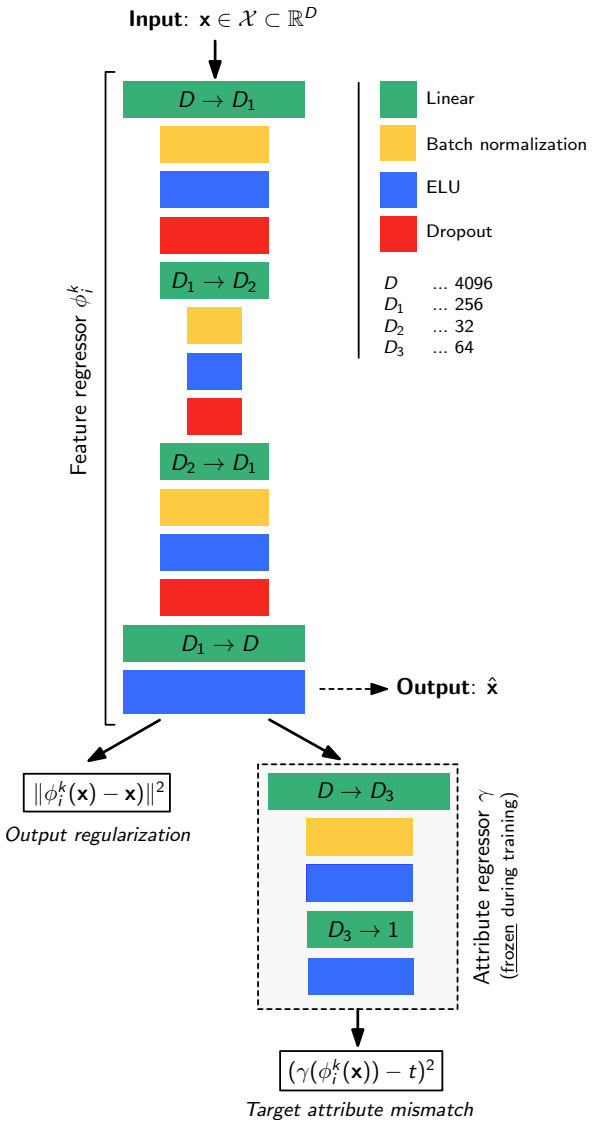

Figure 3: Illustration of the proposed encoder-decoder network for AGA. During *training*, the attribute regressor γ is appended to the network, whereas, for *testing* (*i.e.*, feature synthesis) this part is removed.

Hence, only the encoder-decoder weights are updated. To train one ϕ_i^k for each interval $[l_i, h_i]$ and target attribute value t_k(← Is this a fixed value in the middle of the interval? When using the network for feature synthesis do you still need the feature predictor or do you simply run the input through all K networks? I am a bit confused here.), we partition the training data from the external corpus into subsets $\mathcal{S}_i$, such that $\forall(\mathbf{x}_n, s_n) \in \mathcal{S}_i : s_n \in [l_i, h_i]$. Each ϕ_i^k is then learned by using only those data tuples. We note that, since training is done in feature space $\mathcal{X}$, we have no convolutional layers and consequently training is computationally cheap. For testing, the attribute regressor is removed and

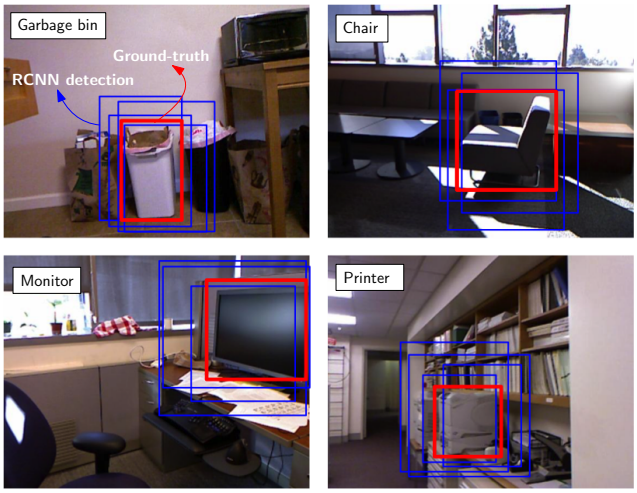

Figure 4: Illustration of *training data generation*. *First*, we keep Fast-RCNN [13] activations (AlexNet `FC7`) of Selective Search [33] bounding boxes that overlap with ground-truth bounding boxes (IoU$>$ 0.5) to generate a sufficient amount of training data. *Second*, attribute values (*i.e.*, depth and pose) of the corresponding 3D ground-truth bounding boxes are transferred to *each* sufficiently overlapping selective search box in 2D.

only the trained encoder-decoder network is used to synthesize features.

4. Experiments

We first discuss the generation of adequate training data for the encoder-decoder network, then evaluate every component of our architecture separately and eventually demonstrate its utility on the problem of one-shot object recognition in a transfer learning setting.

Dataset. We use the SUN RGB-D dataset from Song *et al*. [28]. This dataset contains 10335 RGB images with depth maps, as well as detailed annotations for more than 1000 objects in the form of 2D and 3D bounding boxes. In our setup, we use object depth and pose as our object attributes, *i.e.*, $\mathcal{A} = \{\texttt{Depth}, \texttt{Pose}\}$. For each ground-truth 2D bounding box of an object, we use the depth value at its centroid. Pose information is computed from the 3D bounding box of each object and refers to the rotation of the 3D bounding box along the z-axis. In all experiments, we use the first 5335 images as our *external database*, *i.e.*, the database for which we assume availability of attribute annotations. The remaining 5000 images are used for testing; more details are given in the specific experiments.

Training data. Notably, in SUN RGB-D, the number of instances of each object class are not evenly distributed, simply because this dataset was not specifically designed for object recognition tasks. Consequently, images are also not object-centric, meaning that there is substantial variation in the location of objects, as well as the depth and pose at which they occur. This makes it difficult to extract a sufficient and balanced number of feature descriptors per object class, if we would use the ground-truth bounding boxes only. We circumvent this problem, by leveraging the Fast-RCNN detector of [13] with object proposals generated by Selective Search [33]. In detail, we fine-tune the ImageNet model used in [13] to SUN RGB-D, using the same 19 objects as in [28]. We then run the detector on all images from our training split and keep all bounding boxes with detection scores > 0.7 and a sufficient overlap (measured by the IoU $>$0.5) with the 2D ground-truth bounding boxes(← Not totally clear to me. Why are you using all of thes proposals? Just to crate more input data, even though some of these proposals do not exactly capture the objects of interest?). The associated RCNN activations (at the `FC7` layer) are then used as our features $\mathbf{x}$. Each bounding box proposal that remains after overlap and score thresholding is annotated by the attribute information of the corresponding ground-truth bounding box in 3D. As this strategy generates a larger number of descriptors (compared to the number of ground-truth bounding boxes), we can evenly balance the training data in the sense that we can select an equal number of detections per object class for training (1) the attribute regressor and (2) the encoder-decoder network. Training data generation is illustrated in Fig. 4 on four example images and four object classes.

Implementation. The attribute regressor and the encoder-decoder network are implemented in `Torch`. All models are trained using `Adam` [18]. For the attribute regressor, we train for 30 epochs with a batch size of 300 and a learning rate of 0.001. The encoder-decoder network is also trained for 30 epochs with the same learning rate, but with a batch size of 128. The dropout probability during training is set to 0.25. No dropout is used for testing. For our classification experiments, we use a linear C-SVM, as implemented in `liblinear` [11]. On a Linux system, running Ubuntu 16.04, with 128 GB of memory and one NVIDIA Titan X, training one model (*i.e.*, one ϕ_i^k) takes $\approx$ 30 seconds. The relatively low demand on computational resources highlights the advantage of augmentation in feature space, as no convolutional layers need to be trained. All trained models are available at `AnonymousURL`.

4.1. Attribute regression

In principle, we have the choice of two training strategies for the attribute regressor(s). *First*, we could train *object-specific* regressors $\gamma_c, c \in \{1, \ldots, |\mathcal{C}|\}$, *i.e.*, one regressor for each object class. This would, however, prevent us from using the regressor(s) in a transfer-learning setting, since we cannot assume knowledge about the object classes that appear in the target dataset. An alternative, *second* strategy is to train *object-agnostic* regressors using features from

Object	**D** (MAE [m])		**P** (MAE [deg])	
	per-object	agnostic	per-object	agnostic
bathtub	0.31	1.05	36.02	108.17
bed	0.40	0.30	44.51	70.51
bookshelf	0.64	0.45	50.61	95.44
box	0.49	0.59	29.69	59.37
chair	0.40	0.31	37.82	53.08
counter	0.51	1.45	43.81	13.47
desk	0.39	0.36	48.24	47.07
door	0.41	2.03	45.62	51.84
dresser	0.27	0.44	65.42	63.82
garbage bin	0.34	0.32	45.93	54.43
lamp	0.40	1.04	30.51	132.49
monitor	0.27	0.26	28.90	69.48
night stand	0.53	0.85	28.19	99.40
pillow	0.39	0.46	34.93	73.19
sink	0.17	0.20	60.04	59.43
sofa	0.41	0.33	32.25	51.51
table	0.39	0.30	41.52	50.60
tv	0.47	0.75	32.33	61.77
toilet	0.24	0.23	21.58	50.89

Table 1: Median-Absolute-Error (MAE), for depth / pose, of the attribute regressor, evaluated on *19 objects* from [28]. In our setup, the pose estimation error quantifies the error in predicting a rotation around the z-axis. **D** indicates `Depth`, **P** indicates `Pose`.(← Should there be an average over all these results? Also, what is the typical range for these values? I.e., is a depth error of 0.5 m good or not? Lastly, pose errors seem to be enormous. And what does an error ¿90 degrees mean for object that are approximately symmetric?)

all object classes together. While we will use the object-agnostic model in all subsequent experiments, we evaluate both strategies to assess the potential loss in prediction performance in the object-agnostic setting.

Table 1 lists the median-absolute-error (MAE) of depth (in [m]) and pose (in [deg]) prediction per object, evaluated on instances of 19 object classes extracted from the testing portion of SUN RGB-D. As we can see, training in an object-specific manner leads to a lower MAE for most objects, both for depth and pose. This is not surprising, since the training data is more specialized to each particular object, which essentially amounts to solving simpler sub-problems. However, in many cases, especially for depth, the object-agnostic regressor performs on par, except for object classes with fewer training samples (*i.e.*, lamp, door, etc.). We also remark that, in general, pose estimation from 2D data is a substantially harder problem then depth estimation.

4.2. Feature regression

In this section, we assess the performance of the feature regressor(s) ϕ_i^k, *i.e.*, the part of our architecture from Fig. 3 that is used to generate synthetic features.

In all experiments, we use an overlapping sliding window to bin the range of each attribute $A \in \mathcal{A}$ into T intervals $[l_i, h_i]$. In case of `Depth`, we set $[l_0, h_0] = [0, 1]$ and shift each interval by 0.5 meter, in case of `Pose`, we set $[l_0, h_0] = [0°, 45°]$ and shift by $25°$. We generate as many intervals as needed to cover the full range of the attribute values in the training data. The bin-width and step size were set as to ensure a roughly equal number of associated features in each bin. For augmentation, we choose $0.5, 1, \ldots, \max(\texttt{Depth})$ as target attribute values for `Depth` and $45°, 70°, \ldots, 180°$ for `Pose`. This setup results in 11 target values for `Depth` and 7 for `Pose`. (← I really cannot follow what the Φ_i^k are for.)

We use two separate evaluation metrics to assess the performance of ϕ_i^k. *First*, we are interested in *how well* the feature regressor can generate features that correspond to the desired attribute target values(← Is there not exactly one target value for each Φ_i^k?). To accomplish this, we run each synthetic feature $\hat{\mathbf{x}}$ through the attribute predictor and assess the MAE, *i.e.*, $|\gamma(\hat{\mathbf{x}}) - t|$, over all attribute targets t; Table 2 lists the average MAE, per object, for (1) features from object classes that were *seen* in the training data and (2) features from objects that we have never seen before. As wee can see from Table 2, MAE's for seen and unseen objects are similar, indicating that the encoder-decoder has learned to synthesize features, such that $\gamma(\hat{\mathbf{x}}) \approx t$.

Second, we are interested in how much synthesized features differ from original features. (← I do not follow this? Can you not compute the true features for the test data and compare these features to the synthesized ones? What am I missing?)While we cannot guarantee an identity preserving mapping, "closeness" to the original features can serve as a surrogate measure of this desired property. While, in principle, we could simply evaluate $\|\phi_i^k(\mathbf{x}) - \mathbf{x}\|^2$, the L_2 norm is hard to interpret. Instead, we suggest to compute the Pearson correlation coefficient ρ between each original feature and its synthesized variants, *i.e.*, $\rho(\mathbf{x}, \phi_i^k(\mathbf{x}))$. As ρ ranges from $[-1, 1]$, high values indicate a strong linear relationship to the original features. Results are reported in Table 2. Contrary to the results for MAE, we observe that ρ, when averaged over all objects, is higher for objects that appeared in the training data. This decrease in correlation, however, is relatively small.

In summary, we conclude that ϕ_i^k can be used on feature descriptors from object classes that have *not* appeared in the training corpus. This enables us to test ϕ_i^k in transfer learning setups, as we will see in the one-shot experiments of Sec. 4.3.

4.3. One-shot object recognition

Finally, we demonstrate the utility of our approach on the problem of one-shot object recognition in a transfer learning setup. Specifically, we aim to learn attribute-guided augmenters ϕ_i^k from instances of object classes that are avail-

	Object	ρ	**D** (MAE)	ρ	**P** (MAE)
Seen objects, see Table 1	bathtub	0.76	0.13	0.72	6.51
	bed	0.81	0.10	0.81	4.45
	bookshelf	0.80	0.09	0.80	4.90
	box	0.73	0.11	0.75	5.32
	chair	0.71	0.10	0.73	4.10
	counter	0.75	0.10	0.77	5.28
	desk	0.74	0.10	0.75	4.30
	door	0.66	0.13	0.66	6.11
	dresser	0.78	0.10	0.77	5.29
	garbage bin	0.75	0.10	0.77	4.17
	lamp	0.80	0.09	0.80	4.72
	monitor	0.82	0.09	0.82	4.25
	night stand	0.79	0.10	0.79	5.17
	pillow	0.79	0.11	0.81	4.71
	sink	0.75	0.11	0.75	5.33
	sofa	0.77	0.10	0.79	4.81
	table	0.73	0.10	0.75	4.53
	tv	0.78	0.11	0.76	4.69
	toilet	0.79	0.10	0.79	4.79
	∅	**0.76**	**0.11**	**0.77**	**4.91**
Unseen objects	picture	0.66	0.13	0.67	5.87
	ottoman	0.69	0.13	0.71	5.16
	whiteboard	0.66	0.16	0.67	6.09
	fridge	0.68	0.13	0.69	5.44
	counter	0.75	0.10	0.77	5.30
	books	0.73	0.11	0.75	5.43
	stove	0.70	0.11	0.72	5.67
	cabinet	0.73	0.11	0.73	5.52
	printer	0.72	0.11	0.74	5.15
	computer	0.82	0.09	0.82	4.27
	∅	**0.71**	**0.12**	**0.73**	**5.38**

Table 2: Assessment of ϕ_i^k with respect to (1) Pearson correlation ρ of *synthesized* and *original* features and (2) mean MAE of predicted attribute strengths of synthesized features, $\gamma(\phi_i^k(\mathbf{x}))$, with respect to the target attribute values t. **D** indicates `Depth`-augmentated features (MAE in [m]); **P** indicates `Pose`-augmented features (MAE in [deg]).

able in an external, annotated database (in our case, SUN RGB-D). We denote this collection of object classes as our *source classes* $\mathcal{S}$. Given one instance from a collection of completely different object classes, denoted as the *target classes* $\mathcal{T}$, we aim to train a discriminant classifier C on $\mathcal{T}$, *i.e.*, $C : \mathcal{X} \rightarrow \{1, \ldots, |\mathcal{T}|\}$. Hence, in this setting, $\mathcal{S} \cap \mathcal{T} = \emptyset$. Note that no attribute annotations for instances of object classes in $\mathcal{T}$ are available. This can be considered a variant of transfer learning, since we transfer knowledge from object classes in $\mathcal{S}$ to instances of object classes in $\mathcal{T}$, *without* any prior knowledge about $\mathcal{T}$, except for the single training instances.

Setup. To evaluate performance on one-shot object recognition for *unseen* object classes, we adhere to the following setup: First, we randomly select two collections(← Why do you randomly select two collections?) of 10 object classes to assess the quality of AGA on different sets of unseen objects. We ensure that each object class has at least 100 samples in the testing split of SUN RGB-D and that no object class is(← correct?) in $\mathcal{S}$. This guarantees (1) that we have never seen the image, nor (2) the object category during training. Since, SUN RGB-D does not have object-centric images, we use the ground-truth bounding boxes to obtain the actual object crops. This allows us to tease out the benefit of augmentation without having to deal with confounding factors such as background noise. The two sets of object classes are denoted $\mathcal{T}_1$[1] and $\mathcal{T}_2$[2]. We additionally compile a third set of target classes $\mathcal{T}_3 = \mathcal{T}_1 \cup \mathcal{T}_2$ and remark that $\mathcal{T}_1 \cap \mathcal{T}_2 = \emptyset$. Consequently, we have two 10-class problems and one 20-class problem. For each object image in $\mathcal{T}_i$, we then collect RCNN `FC7` features.

As a *Baseline* comparison(← Is this a standard thing to do? Do I understand this correctly that this means that for each class you only have one training instance?), we train a linear C-SVM (with L_1-normalized features) using only the single instances of each object class in $\mathcal{T}_i$. Exactly the same parameter settings of the linear SVM, but then used train on the single instances + features synthesized by our approach(← Grammar? What do you mean?). We repeat the selection of one-shot instances 500 times and report the average recognition accuray.

Remark. The design of this experiment is similar to [23, Section 4.3.], with the exception that we (1) *do not* detect objects, (2) augmentation is performed in feature space and (3) no object-specific information is available. The latter is important, since [23] assumes the existence of 3D CAD models for objects in $\mathcal{T}$ from which synthetic images are generated. In our case, augmentation does not require any a-priori information about the objects in the set of target classes.

Results. Table 3 lists the classification accuracy for different sets of one-shot training data. *First*, using original one-shot instances augmented by `Depth`-guided features (+**D**); *second*, using original features + `Pose`-guided features (+**D**) and *third*, a combination of the former (+**D**, **P**); In general, we observe that adding AGA synthesized features improves recognition accuracy over the *Baseline* in all cases. For `Depth`-augmented features, gains range from 3-5 percentage points, for `Pose`-augmented features, gains range from 2-4 percentage points on average. We attribute this effect to the difficuly in predicting object pose from 2D data, as can be seen from Table 1. Nevertheless, in both augmentation settings, the gains are statistically signif-

[1] $\mathcal{T}_1$ = {picture, whiteboard, fridge, counter, books, stove, cabinet, printer, computer, ottoman}

[2] $\mathcal{T}_2$ = {mug, telephone, bowl, bottle, scanner, microwave, coffee table, recycle bin, cart, bench}

	Baseline	AGA (Ours) +D	AGA (Ours) +P	AGA (Ours) +D, P
$\mathcal{T}_1$ (10)	33.74	38.84 ✓	36.01 ✓	39.12 ✓
$\mathcal{T}_2$ (10)	23.76	28.95 ✓	27.01 ✓	30.13 ✓
$\mathcal{T}_3$ (20)	22.84	25.84 ✓	24.35 ✓	26.91 ✓

AGA. . . Attribute-Guided Augmentation

Table 3: *Recognition accuracies* (averaged over 500 runs of randomly selecting one-shot samples) for three one-shot object recognition problems. The number in parentheses next to $\mathcal{T}_i$ indicates the number of classes. A '✓' indicates that the result is statistically different (at 5% significance) from the *Baseline* result. **+D** indicates adding `Depth`-augmented features to the one-shot instances; **+P** indicates addition of `Pose`-augmented features and **+D, P** denotes adding a combination of `Depth`-/`Pose`-augmented features.

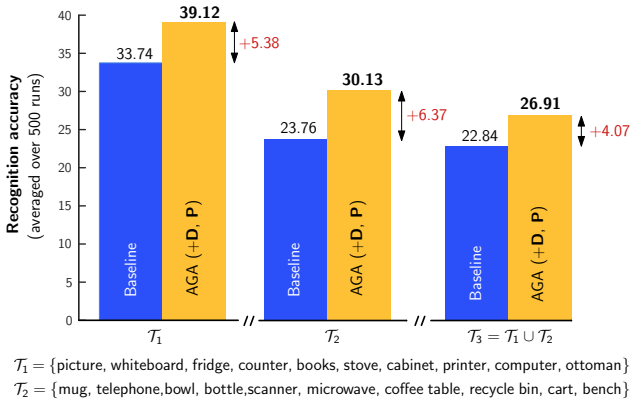

Figure 5: Comparison of the *Baseline* classifier *vs*. the *best* (*c.f*. Table 3) one-shot recognition results obtained by AGA, over three different sets $\mathcal{T}_i$ of (unseen) object classes.(← What is the point of this figure if you have Table 3¿ Does it not show the same information?)

icant (w.r.t. the *Baseline*), as evaluated by a Wilcoxn rank sum test for equal medians [12] at 5% significance (indicated by '✓' in Table 1). Notably, adding `Depth`- and `Pose`-augmented features to the original one-shot features achieves the greatest improvement in recognition accuracy, ranging from 4-6 percentage points, as illustrated in Fig. 5. This indicates that information from depth and pose is complementary and allows for better coverage of the feature space.

5. Discussion

We presented an approach toward attribute-guided augmentation of data in feature space. Our experiments show that object attributes, such as pose and depth, are beneficial in the context of one-shot recognition, *i.e*., an extreme case of limited training data (apart from zero-shot scenarios). While we do use bounding boxes to extract object crops from SUN RGB-D, this is only done to clearly tease out the effect of augmentation. In principle, as our encoder-decoder network is trained in an *object-agnostic* manner, no external knowledge about the target classes is required.

As SUN RGB-D data exhibits high variability in the range of both attributes, augmentation along these dimensions can indeed help for classifier training. However, when variability is constrained, *e.g*., under controlled acquisition settings, the gains may be less apparent. As our approach is not limited to depth or pose, augmentation with respect to other object attributes would be required.

While training the encoder-decoder network requires a pretrained attribute regressor, our results show that even in case of mediocre performance of this component (*e.g*., pose prediction), synthesized features can still supply useful information to the learning process.

Several aspects are interesting for future work. *First*, replacing the attribute regressor for pose with a specifically tailored component, will potentially improve learning of the augmentation function(s) ϕ_i^k and consequently lead to more realistic synthetic examples. *Second*, we conjecture that as additional data with object attributes becomes available, the encoder-decoder can leverage more diverse samples and thus better model changes in the attribute values. *Third*, other recognition tasks, *e.g*., scene recognition, might equally benefit from AGA, particularly when representations of scenes are built on top of detected objects, as in [8] for instance.

References

[1] Y. Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, 2009. 3

[2] C. W. C.E. Rasmussen. *Gaussian Processes for Machine Learning*. The MIT Press, 2005. 3

[3] C. Charalambous and A. Bharath. A data augmentation methodology for training machine/deep learning gait recognition algorithms. In *BMVC*, 2016. 2

[4] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 2, 3

[5] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *CVPR*, 2015. 1

[6] D.-A. Clevert, T. Unterhiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *ICLR*, 2016. 4

[7] M. Dixit, S. Chen, D. Gao, N. Rasiwasia, and N. Vasconcelos. Scene classification with semantic fisher vectors. In *CVPR*, 2015. 1

[8] M. Dixit and N. Vasoncelos. Object based scene representations using fisher scores of local subspace projections. In *NIPS*, 2016. 8

[9] J. Donahue, Y. Jia, O. Vinyals, J. Huffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 1

[10] H. Drucker, C. Burges, L. Kaufman, and A. Smola. Support vector regression machines. In *NIPS*, 1997. 3

[11] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C. J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 9(8):1871–1874, 2008. 5

[12] J. Gibbons and S. Chakraborti. *Nonparametric Statistical Inference*. Chapman & Hall/CRC Press, 5th edition, 2011. 7

[13] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 2, 5

[14] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 2014. 1

[15] S. Hauberg, O. Freifeld, A. Boensen, L. Larsen, J. F. III, and L. Hansen. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In *AISTATS*, 2016. 2, 3

[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 3

[17] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015. 3

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2, 3

[20] R. Kwitt, S. Hegenbart, and M. Niethammer. One-shot learning of scene locations via feature trajectory transfer. In *CVPR*, 2016. 3

[21] E. Miller, N. Matsakis, and P. Viola. Learning from one-example through shared density transforms. In *CVPR*, 2000. 3

[22] V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 3

[23] X. Peng, B. Sun, K. Ali, and K. Saenko. Learning deep object detectors from 3d models. In *ICCV*, 2015. 2, 7

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

[25] G. Rogez and C. Schmid. MoCap-guided data augmentation for 3D pose estimation in the wild. *CoRR*, abs/1607.02046, 2016. 2

[26] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 1

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1

[28] S. Song, S. Lichtenberg, and J. Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *CVPR*, 2015. 5, 6

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15:19291958, 2014. 4

[30] 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

[31] 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

[32] A. Torralba and A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011. 2

[33] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013. 5

[34] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 2, 3