

000
001
002
003
004
005
006
007
008
009
010
011054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107

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 aims to learn 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 facilitates to learn 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.

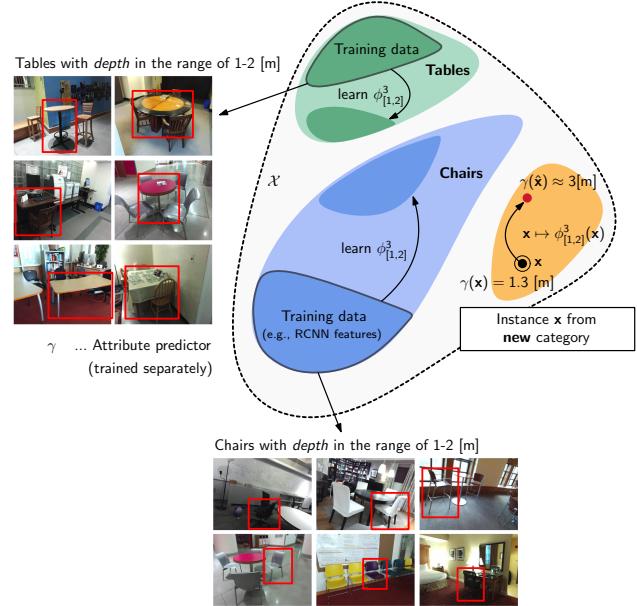


Figure 1: Given a predictor $\gamma : \mathcal{X} \rightarrow \mathbb{R}_+$ of attribute strength (e.g., for depth or pose), we propose to *learn* a mapping of object features $\mathbf{x} \in \mathcal{X}$, such that (1) the mapped feature is “close” to \mathbf{x} (to preserve object identity) and (2) 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 to learn ϕ .

CNN-based transfer is generally achieved either by *fine-tuning* a pre-trained network such as [19], on a new image dataset or by designing a new image representation on such a dataset with 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 constraints like in a *few shot* or a *one shot* learning sce-

108 nario. In these problems, the amount of examples available
109 for learning may be as few as one per category. Finetuning
110 a pre-trained CNN with millions of parameters to such
111 inadequate datasets is clearly not a viable option. A one-
112 shot classifier trained on CNN activations will also be prone
113 to over-fitting due to the high dimensionality of the feature
114 space. The only way to solve the problem of limited data
115 is to somehow *augment* the training corpus by generating
116 more examples for the given classes.
117

118 While augmentation techniques can be as simple as flipping,
119 rotating, adding noise, or extracting random crops
120 from images [19, 4, 34], *task-specific*, or *guided* augmentation
121 strategies [3, 15, 25, 23] have the potential to generate
122 more realistic synthetic samples. This is a particularly im-
123 portant issue, since performance of CNNs heavily relies on
124 sufficient coverage of the variability that we expect in pre-
125 viously unseen data. In scene recognition, *e.g.*, we desire
126 sufficient variability in the constellation and transient states
127 of scene categories, whereas in object recognition, we de-
128 sire variability in the specific incarnations of certain objects,
129 different lighting conditions, pose, or depth, just to name a
130 few. Unfortunately, this variability is often dataset-specific
131 and can cause substantial bias in recognition results [32].
132

133 An important observation in the context of our work is
134 that augmentation is typically performed on an image, or
135 video level. While this is not a particular problem with sim-
136 ple techniques, such as flipping or cropping, it can become
137 computationally expensive if more elaborate augmentation
138 techniques are used. We argue that, in specific problem
139 settings, augmentation might as well be performed in *feature*
140 *space*, especially in situations where features are input
141 to subsequent learning algorithms. This is common, *e.g.*,
142 in recognition tasks, where the softmax output of trained
143 CNNs is often not used directly, but activations at earlier
144 layers are input to an external discriminant classifier.
145

146 **Contribution.** In this work, we propose an approach to
147 augment the training set with *feature descriptors* instead of
148 images. Specifically, we introduce an augmentation tech-
149 nique that learns to synthesize features, guided by desired
150 values for a set of available object attributes, such as depth
151 or pose. An illustration of this concept is shown in Fig. 1.
152 We first train a fast RCNN [13] detector to identify objects
153 in 2D images. This is followed by training a neural network
154 regressor which predicts the 3D attributes of a detected ob-
155 ject, namely its depth from the camera plane and pose. An
156 encoder-decoder network is then trained which, for a de-
157 tected object at a certain depth and pose, will hallucinate
158 the changes in its RCNN features for different depths/poses.
159 Using this architecture, for a new image, we are able to
160 augment existing feature descriptors by an auxiliary set of
161 features that correspond to the object changing its 3D po-
sition. Since, our framework relies on object attributes to

162 guide the augmentation process, we refer to it as *attribute-*
163 *guided augmentation (AGA)*.
164

165 **Organization.** Sec. 2 reviews related work on data augmen-
166 tation. Sec. 3 then introduces the proposed encoder-decoder
167 architecture for attribute-guided augmentation. In Sec. 4,
168 we study the building blocks of this approach in detail and
169 then demonstrate that AGA in feature space considerably
170 improves one-shot object recognition performance on pre-
171 viously unseen objects. Sec. 5 concludes the paper with a
172 discussion and an outlook on potential future directions.
173

174 2. Related work

175 Our review of related work primarily focuses on *data*
176 *augmentation* strategies. While many techniques have been
177 proposed in the context of training deep neural networks to
178 avoid overfitting and increase variability in the data, other
179 (sometimes closely related) techniques have previously ap-
180 peared in the context of one-shot and transfer learning.
181 We can roughly group existing techniques into (1) *generic*,
182 computationally cheap approaches and (2) *task-specific*, or
183 *guided* approaches that are typically more computationally
184 involved.
185

186 As a representative of the first group, Krizhevsky *et al.*
187 [19] leverage a set of label-preserving transformations, such
188 as patch extraction + reflections, and PCA-based intensity
189 transformations, to increase training sample size. Simi-
190 lar techniques are used by Zeiler and Fergus [34]. In [4],
191 Chatfield and Zisserman demonstrate that the augmentation
192 techniques of [19] are not only beneficial for training deep
193 architectures, but shallow learning approaches equally ben-
194 efit from such *simple* and *generic* schemes.
195

196 In the second category of guided-augmentation tech-
197 niques, many approaches have recently been proposed.
198 In [3], *e.g.*, Charalambous and Bharath employ guided-
199 augmentation in the context of gait recognition. The authors
200 suggest to simulate synthetic gait video data (obtained from
201 from avatars) with respect to various confounding factors
202 (such as clothing, hair, etc.) to extend the training corpus.
203 Similar in spirit, Rogez and Schmid [25] recently proposed
204 an image-based synthesis engine for augmenting existing
205 2D human pose data by photorealistic images with greater
206 pose variability. This is done by leveraging 3D motion cap-
207 ture (MoCap) data. 3D data, in the form of synthetic CAD
208 models, is used by Peng *et al.* [23] to render synthetic im-
209 ages of objects (with varying pose, texture, background)
210 that are then used to train CNNs for object detection. It
211 is shown that synthetic data is beneficial, especially in situa-
212 tions where few (or no) training instances are available, but
213 3D CAD models are. Su *et al.* [30] follow a similar pipeline
214 of rendering images from 3D models for 3D viewpoint es-
215 timation, however, with substantially more synthetic data ob-
216 tained, *e.g.*, by additionally deforming existing 3D models
217

216 before rendering.
 217

218 Another (data-driven) guided augmentation technique is
 219 introduced by Hauberg *et al.* [15]. The authors propose to
 220 *learn* class-specific transformations from external training
 221 data, instead of manually specifying transformations as in
 222 [19, 34, 4]. The learned transformations are then applied to
 223 the samples of each class. Specifically, diffeomorphisms are
 224 learned from data and encouraging results are demonstrated
 225 in the context of digit recognition on MNIST. Notably, this
 226 strategy is conceptually similar to earlier work by Miller
 227 *et al.* [21] on one-shot learning, where the authors synthe-
 228 size additional data for digit images via an iterative process,
 229 called *congealing*. During that process, external images of
 230 a given category are aligned by optimizing over a class of
 231 geometric transforms (*e.g.*, affine transforms). These trans-
 232 formations are then applied to single instances of the new
 233 categories to increase data for one-shot learning.

234 Marginally related to our work, we remark that alterna-
 235 tive approaches to implicitly learn spatial transformations
 236 have been proposed. For instance, Jaderberg *et al.* [17] intro-
 237 duce *spatial transformer* modules that can be injected
 238 into existing deep architectures to implicitly capture spa-
 239 tial transformations inherent in the data, thereby improving
 240 invariance to this class of transformations.

241 While *all* previously discussed methods essentially pro-
 242 pose *image-level* augmentation for training CNNs, our
 243 approach is different in that we aim for augmentation in
 244 *feature space*. Along these lines, the approach of Kwitt *et al.*
 245 [20] is conceptually similar to our work. In detail, the
 246 authors suggest to learn how features change as a function of
 247 the strength of certain transient attributes (such as sunny,
 248 cloudy, or foggy) in a scene-recognition context. These
 249 models are then transferred to previously unseen data for
 250 one-shot recognition. However, different to our approach,
 251 the learned models are simple linear regressors and learn-
 252 ing is done in a *scene-class specific* manner. Contrary to
 253 that, we learn deep non-linear models in a *class-agnostic*
 254 manner which enables straightforward application to object
 255 recognition, without the requirement of a direct relation of
 256 new classes to classes in the external training data.

258 3. Architecture

259 **Notation.** To describe our architecture, we let \mathcal{X} denote our
 260 feature space, $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^D$ denotes a feature descriptor
 261 (*e.g.*, a representation of an object) and \mathcal{A} denotes a set of
 262 attributes available in the external training corpus. Further,
 263 we let $s \in \mathbb{R}_+$ denote the strength of an attribute $A \in \mathcal{A}$,
 264 associated with \mathbf{x} . We assume (1) that this attribute can be
 265 predicted by an attribute regressor $\gamma : \mathcal{X} \rightarrow \mathbb{R}_+$ and (2) that
 266 it is possible that its range can be divided into T intervals
 267 $[l_i, h_i]$, where l_i, h_i denote the lower and upper bounds of
 268 the i -th interval.

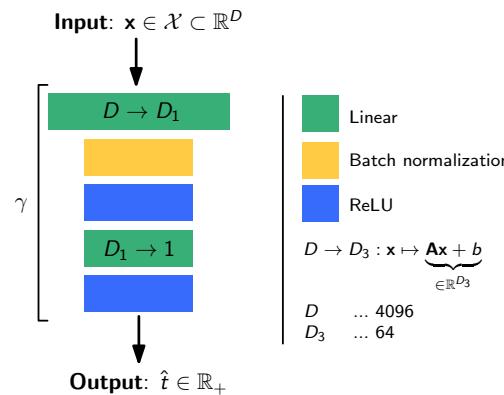
269 **Objective.** On a conceptual level, we aim for a mapping
 270 function ϕ which, given a desired attribute strength t for
 271 some attribute A , transforms input features $\mathbf{x} \in \mathcal{X}$ such that
 272 the attribute strength changes in a controlled manner to a
 273 desired target value t . More formally, we aim to learn
 274

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

275 Since, the formulation in Eq. (1) is overly generic, we con-
 276 strain the problem to the case where we aim to learn differ-
 277 ent ϕ_i^k for a selection of intervals $[l_i, h_i]$ and a selection
 278 of K given target attribute values t_k . While this simplifies
 279 the problem, it requires good *attribute predictor* a-priori,
 280 since, otherwise, we could not decide which ϕ_i to use.
 281 During testing, we (1) predict the attribute strength, *i.e.*,
 282 $\gamma(\mathbf{x}) = \hat{t}$, and then (2) synthesize features as $\hat{\mathbf{x}} = \phi_i^k(\mathbf{x})$
 283 for $k = 1, \dots, T$ if $\hat{t} \in [l_i, h_i]$. Next, we discuss each com-
 284 ponent of this architecture in detail.

285 3.1. Attribute regression

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



299 **Figure 2:** Architecture of the attribute regressor γ .

324 **Learning.** The attribute regressor can easily be trained from
 325 a collection of N training tuples $\{(\mathbf{x}_i, s_i)\}_{i=1}^N$. We remark
 326 that for this part of the architecture, we do not need any
 327 binning of the attribute range.
 328

329 3.2. Feature regression

330 To implement ϕ , we design an encoder-decoder archi-
 331 tecture, reminiscent of a conventional autoencoder [1]. Our
 332 objective, however, is not encode and then reconstruct the
 333 input, but to produce an output that resembles a feature de-
 334 scriptor of an object at a desired attribute value.
 335

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

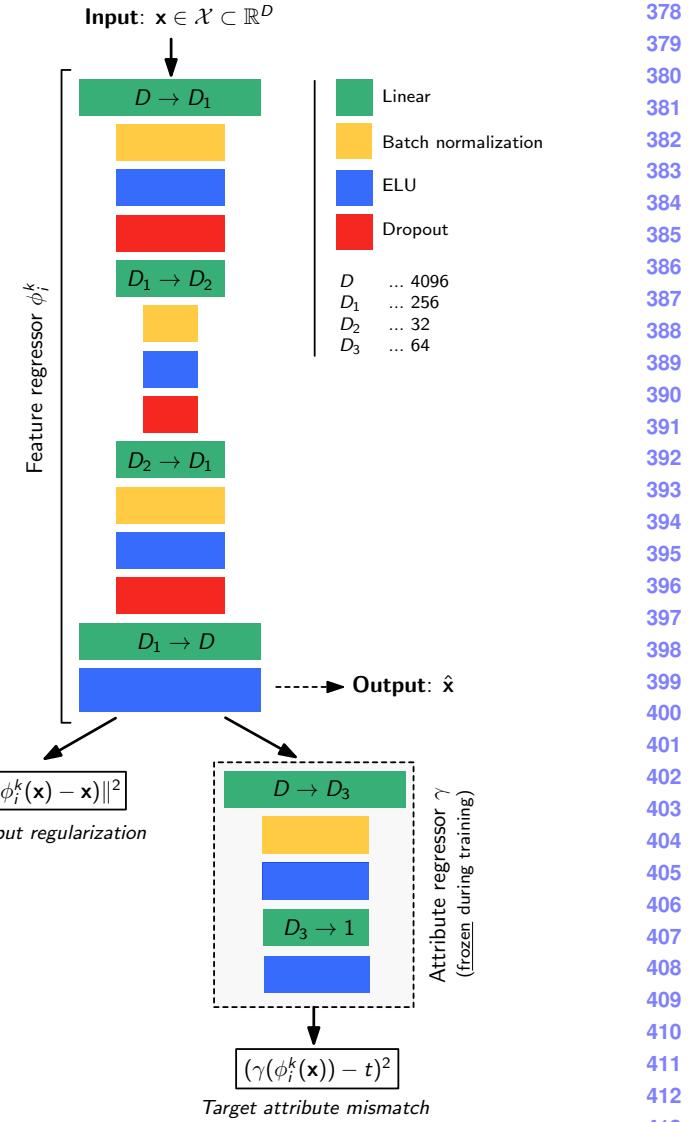
$$340 \quad \min_{\phi \in \mathcal{C}} L(\mathbf{x}, t; \phi) = (\gamma(\phi(\mathbf{x})) - t)^2, \quad (2)$$

342 where the minimization is over a suitable class of functions
 343 \mathcal{C} . Notably, when implementing ϕ as an encoder-decoder
 344 network with an appended (pre-trained) attribute predictor
 345 (see Fig. 3) and loss $(\gamma(\phi(\mathbf{x})) - t)^2$, we have little control
 346 over the decoding results in the sense that we cannot guar-
 347 antee that the *identity* of the input is preserved. This means
 348 that features from a particular class of objects might map
 349 to features that are no longer recognizable as this class, as
 350 the encoder-decoder will *only* learn to “fool” the attribute
 351 predictor γ . For that reason, we add a *regularizer* to the
 352 objective of Eq. (2), *i.e.*, we require that the decoding result
 353 needs to be close, *e.g.*, in the L_2 norm, to the input. This
 354 changes the optimization problem to
 355

$$356 \quad \min_{\phi \in \mathcal{C}} L(\mathbf{x}, t; \phi) = \underbrace{(\gamma(\phi(\mathbf{x})) - t)^2}_{\text{Target mismatch}} + \lambda \underbrace{\|\phi(\mathbf{x}) - \mathbf{x}\|^2}_{\text{Regularizer}}. \quad (3)$$

359 Interpreted differently, this resembles the loss of an au-
 360 toencoder network with an added *target attribute mismatch*
 361 term. The encoder-decoder network that implements the
 362 function class \mathcal{C} to learn ϕ is shown Fig. 3. The core build-
 363 ing block is a combination of a linear layer, batch normal-
 364 ization, ELU [6], followed by dropout [29]. After the final
 365 linear layer, we add one ReLU layer to enforce $\hat{\mathbf{x}} \in \mathbb{R}_+^D$.
 366

367 **Learning.** Training the encoder-decoder network of Fig. 3
 368 first requires a pre-trained attribute regressor γ for each
 369 given attribute $A \in \mathcal{A}$. During training, this attribute regres-
 370 sor is appended to the network and its weights are frozen.
 371 Hence, only the encoder-decoder weights are updated. To
 372 train one ϕ_i^k for each interval $[l_i, h_i]$ and target attribute
 373 value t_k , we partition the training data from the external cor-
 374 pus into subsets \mathcal{S}_i , such that $\forall (\mathbf{x}_n, s_n) \in \mathcal{S}_i : s_n \in [l_i, h_i]$.
 375 Each ϕ_i^k is then learned by using only those data tuples.
 376 We note that, since training is done in feature space \mathcal{X} , we
 377 have no convolutional layers and consequently training is



414 **Figure 3:** Illustration of the proposed encoder-decoder
 415 network for AGA. During *training*, the attribute regressor γ is ap-
 416 pended to the network, whereas, for *testing* (*i.e.*, feature synthe-
 417 sis) this part is removed.
 418

419 computationally cheap. For testing, the attribute regressor
 420 is removed and only the trained encoder-decoder network is
 421 used to synthesize features.
 422

4. Experiments

423 We first discuss the generation of adequate training data
 424 for the encoder-decoder network, then evaluate every com-
 425 ponent of our architecture separately and eventually demon-
 426 strate its utility on the problem of one-shot object recogni-
 427 tion in a transfer learning setting.
 428

429 **Dataset.** We use the SUN RGB-D dataset from Song *et al.*
 430

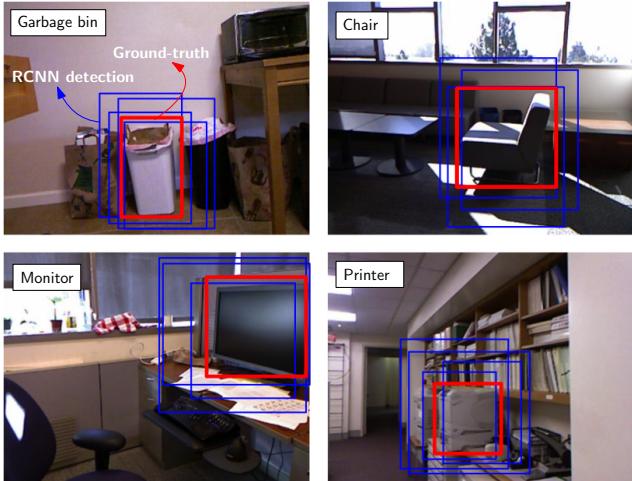


Figure 4: Illustration of *training data generation*. First, we keep Fast-RCNN [13] activations (AlexNet FC7) of Selective Search [33] bounding boxes that overlap with ground-truth bounding boxes ($\text{IoU} > 0.5$) to generate a sufficient amount of training data. Second, attribute values (*i.e.*,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

[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 will use object depth and pose as our object attributes, *i.e.*, $\mathcal{A} = \{\text{Depth}, \text{Pose}\}$. 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

$\text{IoU} > 0.5$) with the 2D ground-truth bounding boxes. 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 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 ≈ 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 γ_c , $c \in \{1, \dots, |\mathcal{C}|\}$, *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 all objects classes together. While we will use the object-agnostic model in all subsequent experiments, we evaluate both strategies to order 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

486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539

540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559	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.

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 than 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 $[l_0, h_0] = [0^\circ, 45^\circ]$ 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 $0.5, 1, \dots, \max(\text{Depth})$ as target attribute values for Depth and $45^\circ, 70^\circ, \dots, 180^\circ$ for Pose. This setup results in 11 target values for Depth and 7 for Pose.

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. 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 we 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. 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 available 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 a completely different object classes, denoted as the *target classes* \mathcal{T} , we aim to train a discriminant classifier C on \mathcal{T} , *i.e.*, $C : \mathcal{X} \rightarrow \{1, \dots, |\mathcal{T}|\}$. 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 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 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 actual benefit

648	Object	ρ	D (MAE)	ρ	P (MAE)
649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666	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
	\emptyset	0.76	0.11	0.77	4.91
667 668 669 670 671 672 673 674 675 676 677	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
	\emptyset	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-augmented features (MAE in [m]); **P** indicates Pose-augmented features (MAE in [deg]).

of augmentation without having to deal 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, 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

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

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

Baseline	AGA (Ours)		
	+D	+P	+D, P
\mathcal{T}_1 (10)	33.74	38.84 ✓	36.01 ✓
\mathcal{T}_2 (10)	23.76	28.95 ✓	27.01 ✓
\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.

the single instances + features synthesized by our approach. We repeat the selection of one-shot instances 500 times and report the average recognition accuracy.

Remark. Notably, the design of this experiment is somewhat 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 difficulty in predicting object pose from 2D data, as can be seen from Table 1. Nevertheless, in both augmentation settings, the gains are statistically significant (w.r.t. the Baseline), as evaluated by a Wilcoxon 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.

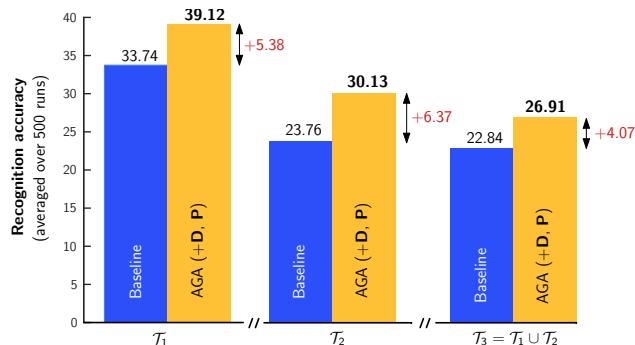


Figure 5: Comparison of the *Baseline* classifier vs. the *best* (c.f. Table 3) one-shot recognition results obtained by AGA, over three different sets \mathcal{T}_i of (unseen) object classes.

5. Discussion

We presented an approach towards 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. 4
- [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. Unterthiner, 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. Vasconcelos. Object based scene representations using fisher scores of local subspace projections. In *NIPS*, 2016. 8
- [9] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, 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

- 864 [21] E. Miller, N. Matsakis, and P. Viola. Learning from one- 918
865 example through shared density transforms. In *CVPR*, 2000. 919
866 3 920
867 [22] V. Nair and G. Hinton. Rectified linear units improve re- 921
868 stricted boltzmann machines. In *ICML*, 2010. 3 922
869 [23] X. Peng, B. Sun, K. Ali, and K. Saenko. Learning deep 923
870 object detectors from 3d models. In *ICCV*, 2015. 2, 7 924
871 [24] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: 925
872 Towards real-time object detection. In *NIPS*, 2015. 1 926
873 [25] G. Rogez and C. Schmid. MoCap-guided data augmentation 927
874 for 3D pose estimation in the wild. *CoRR*, abs/1607.02046, 928
875 2016. 2 929
876 [26] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and 930
877 Y. LeCun. OverFeat: Integrated recognition, localization and 931
878 detection using convolutional networks. In *ICLR*, 2014. 1 932
879 [27] K. Simonyan and A. Zisserman. Very deep convolutional 933
880 networks for large-scale image recognition. *CoRR*, 934
881 abs/1409.1556, 2014. 1 935
882 [28] S. Song, S. Lichtenberg, and J. Xiao. SUN RGB-D: A RGB- 936
883 D scene understanding benchmark suite. In *CVPR*, 2015. 5, 937
884 6 938
885 [29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and 939
886 R. Salakhutdinov. Dropout: A simple way to prevent neural 940
887 networks from overfitting. *JMLR*, 15:19291958, 2014. 4 941
888 [30] H. Su, C. Qi, Y. Li, and L. Guibas. Render for cnn: View- 942
889 point estimation in images using cnns trained with rendered 943
890 3d model views. In *ICCV*, 2015. 2 944
891 [31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, 945
892 D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. 946
893 Going deeper with convolutions. In *CVPR*, 2015. 1 947
894 [32] A. Torralba and A. Efros. Unbiased look at dataset bias. In 948
895 *CVPR*, 2011. 2 949
896 [33] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. 950
897 Selective search for object recognition. *IJCV*, 104(2):154– 951
898 171, 2013. 5 952
899 [34] M. Zeiler and R. Fergus. Visualizing and understanding con- 953
900 volutional networks. In *ECCV*, 2014. 2, 3 954
901 955
902 956
903 957
904 958
905 959
906 960
907 961
908 962
909 963
910 964
911 965
912 966
913 967
914 968
915 969
916 970
917 971