

AGA : Attribute-Guided Augmentation

Mandar Dixit
UC San Diego
mdixit@ucsd.edu

Roland Kwitt
University of Salzburg
rkwitt@gmx.at

Marc Niethammer
UNC Chapel Hill
mn@cs.unc.edu

Nuno Vasconcelos
UC San Diego
nvasconcelos@ucsd.edu

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 synthesis of 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 an external corpus of heavily annotated samples. While prior works primarily augment in the space of images, we propose to perform augmentation in feature space instead. 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 problems of (1) one-shot object recognition in a transfer-learning setting where we have no prior knowledge of the new classes, as well as (2) object-based one-shot scene recognition. As external data, we leverage 3D depth and pose information from the SUN RGB-D dataset. Our experiments show that attribute-guided augmentation of high-level CNN features considerably improves one-shot recognition performance on both problems.

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 [20, 30, 34] and object detection [14, 27]. Success of these networks is mainly due to their high selectivity for semantically meaningful visual concepts, *e.g.*, objects and object parts [29]. 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, 15, 6, 8]. The object recognition network of [20], *e.g.*, has been successfully used for object detection [14, 27], scene classification [15, 8], texture classification [6] and domain adaptation [9], using various transfer mechanisms.

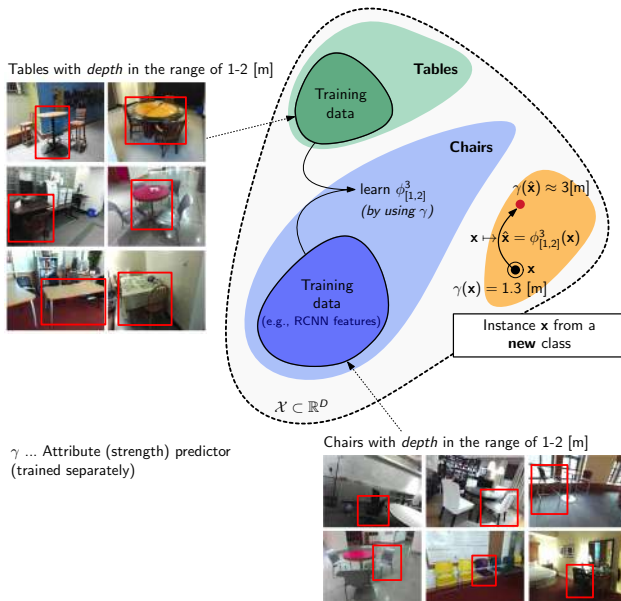


Figure 1: Given a predictor $\gamma : \mathcal{X} \rightarrow \mathbb{R}_+$ of some object attribute (*e.g.*, depth or pose), we propose to *learn* a mapping of object features $\mathbf{x} \in \mathcal{X}$, such that (1) the new synthetic feature $\hat{\mathbf{x}}$ is “close” to \mathbf{x} (to preserve object identity) and (2) the predicted attribute value $\gamma(\hat{\mathbf{x}}) = \hat{t}$ matches a desired object attribute value t , *i.e.*, $t - \hat{t}$ is small. In this illustration, we learn a mapping for features with associated *depth* values in the range of 1-2 [m] to $t = 3$ [m] and apply this mapping to an instance of a new object class. In our approach, this mapping is learned in an *object-agnostic* manner. With respect to our example, this means that *all* training data from ‘chairs’ and ‘tables’ is used to learn feature synthesis function ϕ .

CNN-based transfer is generally achieved either by *fine-tuning* a pre-trained network, such as in [20], 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, 15, 8, 6]. 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 such as in a *few-shot* or a *one-shot* learning scenario. In these problems, the number of examples available for learning may be as few as one per class. Fine-tuning 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 obtaining more examples for the given classes.

While augmentation techniques can be as simple as flipping, rotating, adding noise, or extracting random crops from images [20, 5, 37], *task-specific*, or *guided* augmentation strategies [4, 16, 28, 25] 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 unseen testing data. In scene recognition, we desire, for example, sufficient variability in the constellation and transient states of scene categories (*c.f.* [21]), whereas in object recognition, we desire variability in the specific incarnations of certain objects, 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 [35].

An important observation in the context of our work is that augmentation is typically performed on the 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 steps. 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 classifier.

Contribution. We propose an approach to augment the training set with *feature descriptors* instead of images. Specifically, we advocate an augmentation technique that learns to synthesize features, guided by desired values for a set of object attributes, such as depth or pose. An illustration of this concept is shown in Fig. 1. We first train a fast RCNN [14] 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” the changes in its RCNN features for a set of desired 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 augmentation, we refer to it as *attribute-guided augmentation (AGA)*.

Organization. Sec. 2 reviews prior work. 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 improves one-shot object recognition and object-based scene recognition performance on previously unseen classes. 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 over-fitting 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.* [20] 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 [37]. In [5], Chatfield and Zisserman demonstrate that the augmentation techniques of [20] 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 [4], *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 avatars) with respect to various confounding factors (such as clothing, hair, etc.) to extend the training corpus. Similar in spirit, Rogez and Schmid [28] propose 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. In [25], Peng *et al.* also use 3D data, in the form of CAD models, 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.* [33] follow a similar pipeline of rendering images from 3D models for viewpoint estimation, however, with substantially more synthetic data obtained, *e.g.*, by deforming existing 3D models before rendering.

Another (data-driven) guided augmentation technique is introduced by Hauberg *et al.* [16]. The authors propose to *learn* class-specific transformations from external training data, instead of manually specifying transformations as in [20, 37, 5]. 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.* [23] 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 classes 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.* [18] 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.* [21] 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 transferred to previously unseen data for one-shot recognition. There are, however, two key differences between their approach and ours. First, they require datasets labeled with *attribute trajectories*, *i.e.*, all variations of an attribute for every instance of a class. We, on the other hand, make use of conventional datasets that seldom carry such extensive labeling. Second, their augmenters are simple linear regressors trained in a *scene-class specific* manner. In contrast, we learn deep non-linear models in a *class-agnostic* manner which enables a straightforward application to recognition in transfer settings.

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 that are available for objects in the external training corpus. Further, we let $s \in \mathbb{R}_+$ denote the value 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 its range can be divided into I intervals $[l_i, h_i]$, where l_i, h_i denote the lower and upper bounds of the i -th interval. The set of desired object attribute values is $\{t_1, \dots, t_T\}$.

Objective. On a conceptual level, we aim for a synthesis function ϕ which, given a desired value t for some object attribute A , transforms the object features $\mathbf{x} \in \mathcal{X}$ such that

the attribute strength changes in a controlled manner to 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 $[l_i, h_i]$ within the range of attribute A and a selection of T desired object attribute values t_k . In our illustration of Fig. 1, *e.g.*, we have one interval $[l, h] = [1, 2]$ and one attribute (depth) with target value 3[m]. While learning separate synthesis functions simplifies the problem, it requires a good a-priori *attribute (strength) predictor*, since, otherwise, we could not decide which ϕ_i^k to use. During testing, we (1) predict the object’s attribute value from its original feature \mathbf{x} , *i.e.*, $\gamma(\mathbf{x}) = \hat{t}$, and then (2) synthesize additional features as $\hat{\mathbf{x}} = \phi_i^k(\mathbf{x})$ for $k = 1, \dots, T$. If $\hat{t} \in [l_i, h_i] \wedge t_k \notin [l_i, h_i]$, ϕ_i^k is used. Next, we discuss each component of this approach 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 [3], 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 reuse this building block in the learning stage of the synthesis 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 (BN) [17] and rectified linear units (ReLU) [24]. 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.

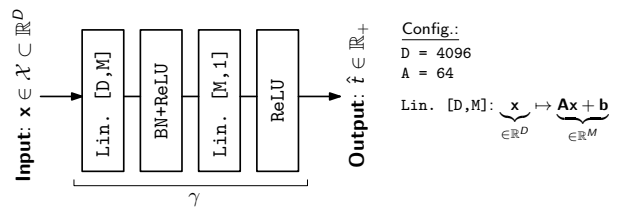


Figure 2: Architecture of the attribute regressor γ .

Learning. The attribute regressor can easily be trained from a collection of N training tuples $\{(\mathbf{x}_i, s_i)\}_{i=1}^N$ for each attribute. As the task of the attribute regressor is to predict in which interval the original feature \mathbf{x} resides, we do not need to organize the training data into intervals in this step.

3.2. Feature regression

To implement¹ ϕ , we design an encoder-decoder architecture, reminiscent of a conventional autoencoder [1]. Our 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 and decodes it to the desired result. In general, we can formulate the optimization problem as

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

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 result in the sense that we cannot guarantee that the *identity* of the input is preserved. This means that features from a particular object class 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 2-norm, to the input. This changes the optimization problem of Eq. (2) to

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

Interpreted differently, this resembles the loss of an autoencoder network with an added *target attribute mismatch* penalty. The encoder-decoder network that implements the function class \mathcal{C} to learn ϕ is shown in Fig. 3. The core building block is a combination of a linear layer, batch normalization, ELU [7], followed by dropout [32]. 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 requires an a-priori 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*. Hence, only the encoder-decoder weights are updated. To train one ϕ_i^k for each interval $[l_i, h_i]$ of the object attribute range and a desired object attribute value t_k , we partition the training data from the external corpus into subsets \mathcal{S}_i , such that $\forall (\mathbf{x}_n, s_n) \in \mathcal{S}_i : s_n \in [l_i, h_i]$. One ϕ_i^k is learned from \mathcal{S}_i for each desired object attribute value t_k . As training is in feature space \mathcal{X} , we have no convolutional layers and consequently training is computationally cheap. For testing, the attribute regressor is removed and only the trained encoder-decoder network (implementing ϕ_i^k) is used to synthesize features. Consequently, given $|\mathcal{A}|$ attributes, I inter-

¹We omit the sub-superscripts for readability.

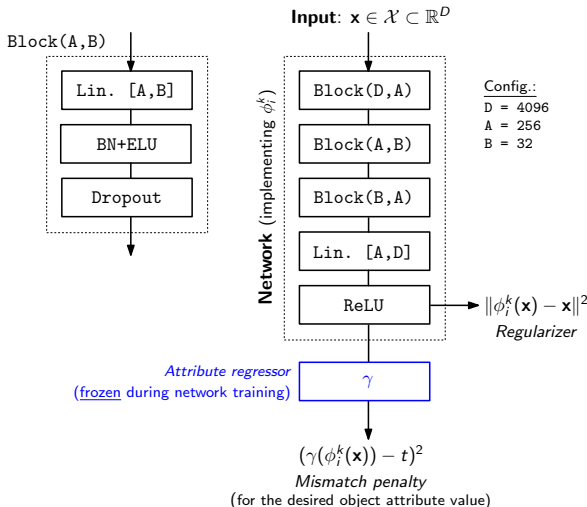


Figure 3: Illustration of the proposed encoder-decoder network for AGA. During *training*, the attribute regressor γ is appended to the network, whereas, for *testing* (*i.e.*, feature synthesis) this part is removed. When learning ϕ_i^k , the input \mathbf{x} is such that the associated attribute value s is within $[l_i, h_i]$ and one ϕ_i^k is learned per desired attribute value t_k .

vals per attribute and T target values for an object attribute, we obtain $|\mathcal{A}| \cdot I \cdot T$ synthesis functions.

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 (1) one-shot object recognition in a transfer learning setting and (2) one-shot scene recognition.

Dataset. We use the SUN RGB-D dataset from Song *et al.* [31]. 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 attributes, *i.e.*, $\mathcal{A} = \{\text{Depth}, \text{Pose}\}$. For each ground-truth 3D bounding box, we extract the depth value at its centroid and obtain pose information as the rotation of the 3D bounding box about the vertical y -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 ob-

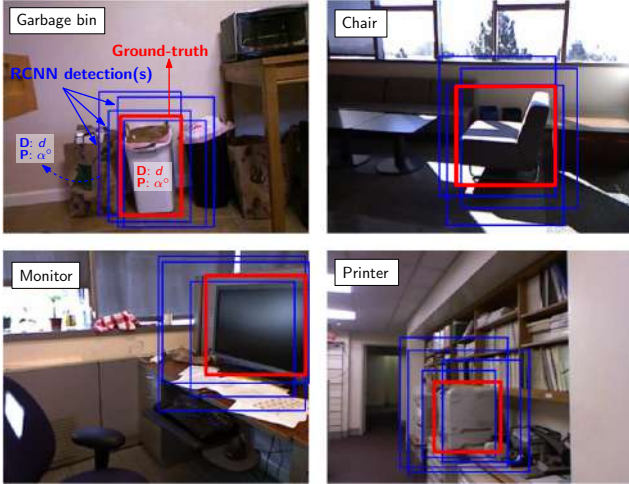


Figure 4: Illustration of *training data generation*. First, we obtain fast RCNN [14] activations (FC7 layer) of Selective Search [36] proposals that overlap with 2D ground-truth bounding boxes (IoU > 0.5) and scores > 0.7 (for a particular object class) to generate a sufficient amount of training data. Second, attribute values (i.e., depth **D** and pose **P**) of the corresponding 3D ground-truth bounding boxes are associated with the proposals (best-viewed in color).

ject class, if we would *only* use the ground-truth bounding boxes to extract training data. We circumvent this problem by leveraging the fast RCNN detector of [14] with object proposals generated by Selective Search [36]. In detail, we finetune the ImageNet model from [14] to SUN RGB-D, using the same 19 objects as in [31]. We then run the detector on all images from our training split and keep the proposals with detection scores > 0.7 and a sufficient overlap (measured by the IoU > 0.5) with the 2D ground-truth bounding boxes. This is a simple augmentation technique to increase the amount of available training data. The associated RCNN activations (at the FC7 layer) are then used as our features \mathbf{x} . Each 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 to train (1) the attribute regressor and (2) the encoder-decoder network. Training data generation is illustrated in Fig. 4 on four example images.

Implementation. The attribute regressor and the encoder-decoder network are implemented in `Torch`. All models are trained using Adam [19]. 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 classifica-

Object	D (MAE [m])		P (MAE [deg])	
	per-object	agnostic	per-object	agnostic
bathtub	0.23	0.94	37.97	46.85
bed	0.39	0.30	44.36	42.59
bookshelf	0.57	0.43	52.95	41.41
box	0.55	0.51	27.05	38.14
chair	0.37	0.31	37.90	32.86
counter	0.54	0.62	40.16	52.35
desk	0.41	0.36	48.63	41.71
door	0.49	1.91	52.73	102.23
dresser	0.32	0.41	67.88	70.92
garbage bin	0.36	0.32	47.51	45.26
lamp	0.42	0.69	25.93	23.91
monitor	0.24	0.22	34.04	25.85
night stand	0.56	0.65	23.80	20.21
pillow	0.38	0.43	32.56	35.64
sink	0.20	0.19	56.52	45.75
sofa	0.40	0.33	34.36	34.51
table	0.37	0.33	41.31	37.30
tv	0.35	0.48	35.29	24.23
toilet	0.26	0.20	25.32	19.59
\emptyset	0.39	0.51	40.33	41.12

Table 1: Median-Absolute-Error (MAE), for depth / pose, of the attribute regressor, evaluated on 19 objects from [31]. In our setup, the pose estimation error quantifies the error in predicting a rotation around the z -axis. **D** indicates Depth, **P** indicates Pose. For reference, the range of of the object attributes in the training data is [0.2m, 7.5m] for Depth and $[0^\circ, 180^\circ]$ for Pose. Results are averaged over 5 training / evaluation runs.

tion 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 AGA in feature space, as no convolutional layers need to be trained. All trained models+source code are publicly available online².

4.1. Attribute regression

While our strategy, AGA, to data augmentation is *agnostic* to the object classes, in both the training and testing dataset, it is interesting to compare attribute prediction performance to the case where we train *object-specific* regressors. In other words, we compare object-agnostic training to training one regressor γ_j , $j \in \{1, \dots, |\mathcal{S}|\}$ for each object class in \mathcal{S} . This allows us to quantify 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. We train on instances of 19 object classes (\mathcal{S}) in our training split of SUN RGB-D and test on instances of the same object classes, but extracted from the testing split. As we can see, training in an object-specific manner leads to a lower MAE overall, 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 (e.g., door).

²<https://github.com/rkwitt/GuidedAugmentation>

We also remark that, in general, pose estimation from 2D data is a substantially harder problem than depth estimation (which works remarkably well, even on a per-pixel level, *c.f.* [22]). Nevertheless, our recognition experiments (in Secs. 4.3 and 4.4) show that even with mediocre performance of the pose predictor (due to symmetry issues, etc.), augmentation along this dimension is still beneficial.

4.2. Feature regression

We assess the performance of our regressor(s) ϕ_i^k , shown in Fig. 3, that are used for synthetic feature generation. In all experiments, we use an overlapping sliding window to bin the range of each attribute $A \in \mathcal{A}$ into I 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 / step-size were set to ensure a roughly equal number of 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 results in $T = 11$ target values for Depth and $T = 7$ 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 evaluate this directly (as we do not have data from one particular object instance at multiple depths and poses), we can assess how “close” synthesized features are to the original features. The intuition here is that closeness in feature space is indicative of an object-identity preserving synthesis. In principle, we could simply evaluate $\|\phi_i^k(\mathbf{x}) - \mathbf{x}\|^2$, however, the 2-norm is hard to interpret. Instead, we 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. Similar to our previous results for MAE, we observe that ρ , when averaged over all objects, is slightly lower for objects that did not appear in the training data. This decrease in correlation, however, is relatively small.

In summary, we conclude that these results warrant the

	Object	ρ	D (MAE [m])	ρ	P (MAE [deg])
Seen objects, see Table 1	bathtub	0.75	0.10	0.68	3.99
	bed	0.81	0.07	0.82	3.30
	bookshelf	0.80	0.06	0.79	3.36
	box	0.74	0.08	0.74	4.44
	chair	0.73	0.07	0.71	3.93
	counter	0.76	0.08	0.77	3.90
	desk	0.75	0.07	0.74	3.93
	door	0.67	0.10	0.63	4.71
	dresser	0.79	0.08	0.77	4.12
	garbage bin	0.76	0.07	0.76	5.30
	lamp	0.82	0.08	0.79	4.83
	monitor	0.82	0.06	0.80	3.34
	night stand	0.80	0.07	0.78	4.00
	pillow	0.80	0.08	0.81	3.87
	sink	0.75	0.11	0.76	4.00
	sofa	0.78	0.08	0.78	4.29
table	0.75	0.07	0.74	4.10	
tv	0.78	0.08	0.72	4.66	
toilet	0.80	0.10	0.81	3.70	
	\emptyset	0.77	0.08	0.76	4.10
Unseen objects (\mathcal{T})	picture	0.67	0.08	0.65	5.13
	ottoman	0.70	0.09	0.70	4.41
	whiteboard	0.67	0.12	0.65	4.43
	fridge	0.69	0.10	0.68	4.48
	counter	0.76	0.08	0.77	3.98
	books	0.74	0.08	0.73	4.26
	stove	0.71	0.10	0.71	4.50
	cabinet	0.74	0.09	0.72	3.99
	printer	0.73	0.08	0.72	4.59
	computer	0.81	0.06	0.80	3.73
		\emptyset	0.72	0.09	0.71

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

use of ϕ_i^k 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 following one-shot experiments of Secs. 4.3 and 4.4.

4.3. One-shot object recognition

First, we demonstrate the utility of our approach on the task 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 completely different object classes, denoted as the *target classes* \mathcal{T} , we aim to train a discriminant classifier C on \mathcal{T} , *i.e.*, $C : \mathcal{X} \rightarrow \{1, \dots, |\mathcal{T}|\}$. 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} .

Setup. We evaluate one-shot object recognition performance on three collections of previously unseen object classes in the following setup: First, we randomly select two sets of 10 object classes and ensure that each object class has at least 100 samples in the testing split of SUN

	Baseline	AGA+D	AGA+P	AGA+D+P
<i>One-shot</i>				
\mathcal{T}_1 (10)	33.74	38.32 ✓	37.25 ✓	39.10 ✓
\mathcal{T}_2 (10)	23.76	28.49 ✓	27.15 ✓	30.12 ✓
\mathcal{T}_3 (20)	22.84	25.52 ✓	24.34 ✓	26.67 ✓
<i>Five-shot</i>				
\mathcal{T}_1 (10)	50.03	55.04 ✓	53.83 ✓	56.92 ✓
\mathcal{T}_2 (10)	36.76	44.57 ✓	42.68 ✓	47.04 ✓
\mathcal{T}_3 (20)	37.37	40.46 ✓	39.36 ✓	42.87 ✓

Table 3: Recognition accuracy (over 500 trials) for three object recognition tasks; *top*: one-shot, *bottom*: five-shot. Numbers in parentheses indicate the #classes. A ‘✓’ indicates that the result is statistically different (at 5% sig.) from the *Baseline*. +D indicates adding Depth-aug. features to the one-shot instances; +P indicates addition of Pose-aug. features and +D, P denotes adding a combination of Depth/Pose-aug. features.

RGB-D. We further ensure that no object class is in \mathcal{S} . This guarantees (1) that we have never seen the image, nor (2) the object class 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}_1^3 and \mathcal{T}_2^4 . 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*, we “train” a linear C-SVM (on 1-norm normalized features) using only the single instances of each object class in \mathcal{T}_i (SVM cost fixed to 10). Exactly the same parameter settings of the SVM are then used to train on the single instances + features synthesized by AGA. We repeat the selection of one-shot instances 500 times and report the average recognition accuracy. For comparison, we additionally list 5-shot recognition results in the same setup.

Remark. The design of this experiment is similar to [25, Section 4.3.], with the exceptions 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 [25] assumes the existence of 3D CAD models for objects in \mathcal{T}_i from which synthetic images can be rendered. In our case, augmentation *does not* require any a-priori information about the objects classes.

Results. Table 3 lists the classification accuracy for the 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 (+P) and *third*, a combination of both (+D, P); In general, we observe that adding AGA-synthesized features improves recognition accuracy over the *Baseline* in all cases.

³ $\mathcal{T}_1 = \{\text{picture, whiteboard, fridge, counter, books, stove, cabinet, printer, computer, ottoman}\}$

⁴ $\mathcal{T}_2 = \{\text{mug, telephone, bowl, bottle, scanner, microwave, coffee table, recycle bin, cart, bench}\}$

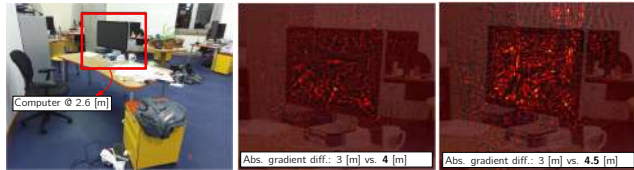


Figure 5: Illustration of the difference in *gradient magnitude* when backpropagating (through RCNN) the 2-norm of the difference between an original and a synthesized feature vector for an increasing desired change in depth, i.e., 3[m] vs. 4[m] (*middle*) and 3[m] vs. 4.5[m] (*right*).

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 [13] at 5% significance (indicated by ‘✓’ in Table 3). Adding both Depth- and Pose-augmented features to the original one-shot features achieves the greatest improvement in recognition accuracy, ranging from 4-6 percentage points. This indicates that information from depth and pose is complementary and allows for better coverage of the feature space. Notably, we also experimented with the metric-learning approach of Fink [12] which only led to negligible gains over the *Baseline* (e.g., 33.85% on \mathcal{T}_1).

Feature analysis/visualization. To assess the nature of feature synthesis, we backpropagate through RCNN layers the gradient w.r.t. the 2-norm between an original and a synthesized feature vector. The strength of the input gradient indicates *how much each pixel of the object must change* to produce a proportional change in depth/pose of the sample. As can be seen in the example of Fig. 5, a *greater* desired change in depth invokes a *stronger* gradient on the monitor. *Second*, we ran a *retrieval experiment*: we sampled 1300 instances of 10 (unseen) object classes (\mathcal{T}_1) and synthesized features for each instance w.r.t. depth. Synthesized features were then used for retrieval on the original 1300 features. This allows to assess if synthesized features (1) allow to retrieve instances of the *same class* (Top-1 acc.) and (2) of the desired attribute value. The latter is measured by the coefficient of determination (R^2). As seen in Table 4, the R^2 scores indicate that we can actually retrieve instances with the desired attribute values. Notably, even in cases where $R^2 \approx 0$ (i.e., the linear model does not explain the variability), the results still show decent Top-1 acc., revealing that synthesis *does not alter class membership*.

4.4. Object-based one-shot scene recognition

Motivation. We can also use AGA for a different type of transfer, namely the transfer from object detection networks to one-shot scene recognition. Although, object detection is

Object	Top-1	R^2	Object	Top-1	R^2
picture	0.33	0.36	whiteboard	0.12	0.30
fridge	0.26	0.08	counter	0.64	0.18
books	0.52	0.07	stove	0.20	0.13
cabinet	0.57	0.27	printer	0.31	0.02
computer	0.94	0.26	ottoman	0.60	0.12

Table 4: Retrieval results for unseen objects (\mathcal{T}_1) when querying with synthesized features of varying depth. Larger R^2 values indicate a stronger linear relationship ($R^2 \in [0, 1]$) to the depth values of retrieved instances.

Method	Accuracy [%]
max. pool (<i>Baseline</i>)	13.97
AGA FV (+D)	15.13
AGA FV (+P)	14.63
AGA CL-1 (+D, max.)	16.04
AGA CL-2 (+P, max.)	15.52
AGA CL-3 (+D, +P, max.)	16.32
Sem-FV [8]	32.75
AGA Sem-FV	34.36
Places [38]	51.28
AGA Places	52.11

Table 5: One-shot classification on 25 indoor scene classes [26]: {auditorium, bakery, bedroom, bookstore, children room, classroom, computer room, concert hall, corridor, dental office, dining room, hospital room, laboratory, library, living room, lobby, meeting room, movie theater, nursery, office, operating room, pantry, restaurant}. For Sem-FV [8], we use ImageNet CNN features extracted at one image scale.

a challenging task in itself, significant progress is made, every year, in competitions such as the ImageNet challenge. Extending the gains in object detection to other related problems, such as scene recognition, is therefore quite appealing. A system that uses an accurate object detector such as an RCNN [14] to perform scene recognition, could generate comprehensive annotations for an image in one forward pass. An object detector that supports one-shot scene recognition could do so with the least amount of additional data. It must be noted that such systems are different from object recognition based methods such as [15, 8, 6], where explicit detection of objects is not necessary. They apply filters from object recognition CNNs to several regions of images and extract features from all of them, whether or not an object is found. The data available to them is therefore enough to learn complex descriptors such as Fisher vectors (FVs). A detector, on the other hand, may produce very few features from an image, based on the number of objects found. AGA is tailor-made for such scenarios where features from an RCNN-detected object can be augmented.

Setup. To evaluate AGA in this setting, we select a 25-class subset of MIT Indoor [26], which may contain objects that the RCNN is trained for. The reason for this choice is our reliance on a detection CNN, which has a vocabulary of 19 objects from SUN RGB-D. At present, this is the largest such dataset that provides objects and their 3D attributes. The system can be extended easily to accommodate more scene classes if a larger RGB-D object dataset becomes available. As the RCNN produces very few detections per scene image, the best approach, without augmentation, is

to perform pooling of RCNN features from proposals into a fixed-size representation. We used max-pooling as our *baseline*. Upon augmentation, using predicted depth/ pose, an image has enough RCNN features to compute a GMM-based FV. For this, we use the experimental settings in [8]. The FVs are denoted as AGA FV (+D) and AGA FV (+P), based on the attribute used to guide the augmentation. As classifier, we use a linear C-SVM with fixed parameter (C).

Results. Table 5 lists the average one-shot recognition accuracy over multiple iterations. The benefits of AGA are clear, as both aug. FVs perform better than the max-pooling baseline by 0.5-1% points. Training on a combination (concatenated vector) of the augmented FVs and max-pooling, denoted as AGA CL-1, AGA CL-2 and AGA CL-3 further improves by about 1-2% points. Finally, we combined our aug. FVs with the state-of-the-art semantic FV of [8] and Places CNN features [38] for one-shot classification. Both combinations, denoted AGA Sem-FV and AGA Places, improved by a non-trivial margin ($\sim 1\%$ points).

5. Discussion

We presented an approach toward attribute-guided augmentation in feature space. Experiments show that object attributes, such as pose / depth, are beneficial in the context of one-shot recognition, *i.e.*, an extreme case of limited training data. Notably, even in case of mediocre performance of the attribute regressor (*e.g.*, on pose), results indicate that synthesized features can still supply useful information to the classification process. While we do use bounding boxes to extract object crops from SUN RGB-D in our object-recognition experiments, this is only done to clearly tease out the effect of augmentation. In principle, as our encoder-decoder is trained in an *object-agnostic* manner, no external knowledge about classes is required.

As SUN RGB-D exhibits high variability in the range of both attributes, augmentation along these dimensions can indeed help classifier training. However, when variability is limited, *e.g.*, under controlled acquisition settings, the gains may be less apparent. In that case, augmentation with respect to other object attributes might be required.

Two aspects are specifically interesting for future work. *First*, replacing the attribute regressor for pose with a specifically tailored component will potentially improve learning of the synthesis function(s) ϕ_i^k and lead to more realistic synthetic samples. *Second*, we conjecture that, as additional data with more annotated object classes and attributes becomes available (*e.g.*, [2]), the encoder-decoder can leverage more diverse samples and thus model feature changes with respect to the attribute values more accurately.

Acknowledgments. This work is supported by NSF awards IIS-1208522, CCF-0830535, ECCS-1148870 and a generous donation of GPUs from Nvidia.

References

- [1] Y. Bengio. Learning deep architectures for AI. *Found. Trends Mach. Learn.*, 2(1):1–127, 2009. 4
- [2] A. Borji, S. Izadi, and L. Itti. iLab-20M: A large-scale controlled object dataset to investigate deep learning. In *CVPR*, 2016. 8
- [3] C. W. C.E. Rasmussen. *Gaussian Processes for Machine Learning*. The MIT Press, 2005. 3
- [4] C. Charalambous and A. Bharath. A data augmentation methodology for training machine/deep learning gait recognition algorithms. In *BMVC*, 2016. 2
- [5] 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
- [6] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *CVPR*, 2015. 1, 8
- [7] D.-A. Clevert, T. Unterhiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *ICLR*, 2016. 4
- [8] M. Dixit, S. Chen, D. Gao, N. Rasiwasia, and N. Vasconcelos. Scene classification with semantic Fisher vectors. In *CVPR*, 2015. 1, 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] M. Fink. Object classification from a single example utilizing relevance metrics. In *NIPS*, 2004. 7
- [13] J. Gibbons and S. Chakraborti. *Nonparametric Statistical Inference*. Chapman & Hall/CRC Press, 5th edition, 2011. 7
- [14] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 2, 5, 8
- [15] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 2014. 1, 8
- [16] 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
- [17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 3
- [18] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015. 3
- [19] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5
- [20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2
- [21] R. Kwitt, S. Hegenbart, and M. Niethammer. One-shot learning of scene locations via feature trajectory transfer. In *CVPR*, 2016. 2, 3
- [22] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *CVPR*, 2015. 6
- [23] E. Miller, N. Matsakis, and P. Viola. Learning from one-example through shared density transforms. In *CVPR*, 2000. 3
- [24] V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 3
- [25] X. Peng, B. Sun, K. Ali, and K. Saenko. Learning deep object detectors from 3d models. In *ICCV*, 2015. 2, 7
- [26] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *CVPR*, 2009. 8
- [27] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection. In *NIPS*, 2015. 1
- [28] G. Rogez and C. Schmid. MoCap-guided data augmentation for 3D pose estimation in the wild. *CoRR*, abs/1607.02046, 2016. 2
- [29] 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
- [30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1
- [31] S. Song, S. Lichtenberg, and J. Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *CVPR*, 2015. 4, 5
- [32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15:1929–1958, 2014. 4
- [33] 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
- [34] 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
- [35] A. Torralba and A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011. 2
- [36] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013. 5
- [37] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 2
- [38] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using Places database. In *NIPS*, 2014. 8