Discovering States and Transformations in Image Collections

Phillip Isola* Joseph J. Lim* Edward H. Adelson
Massachusetts Institute of Technology
phillipi@mit.edu, {lim,adelson}@csail.mit.edu

Abstract

Objects in visual scenes come in a rich variety of transformed states. A few classes of transformation have been heavily studied in computer vision: mostly simple, parametric changes in color and geometry. However, transformations in the physical world occur in many more flavors, and they come with semantic meaning: e.g., bending, folding, aging, etc. The transformations an object can undergo tell us about its physical and functional properties. In this paper, we introduce a dataset of objects, scenes, and materials, each of which is found in a variety of transformed states. Given a novel collection of images, we show how to explain the collection in terms of the states and transformations it depicts. Our system works by generalizing across object classes: states and transformations learned on one set of objects are used to interpret the image collection for an entirely new object class.

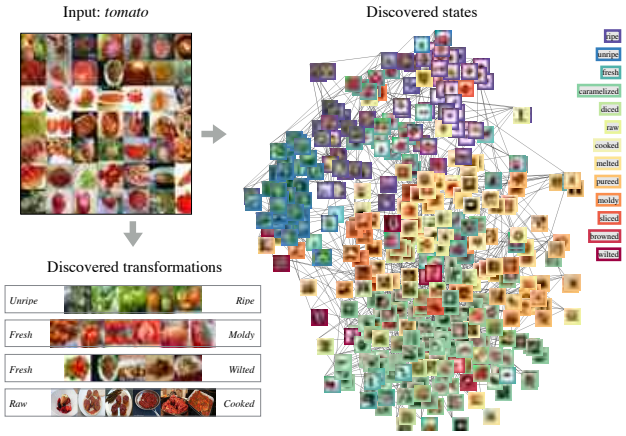

Figure 1. **Example input and automatic output of our system:** Given a collection of images from one category (top-left, subset of collection shown), we are able to parse the collection into a set of states (right). In addition, we discover how the images transform between antonymic pairs of states (bottom-left). Here we visualize the transformations using the technique described in Section 4.5.

1. Introduction

Much work in computer vision has focused on the problem of invariant object recognition [9, 7], scene recognition [32, 33], and material recognition [26]. The goal in each of these cases is to build a system that is invariant to all within-class variation. Nonetheless, the variation in a class is quite meaningful to a human observer. Consider Figure 1. The collection of photos on the left only shows tomatoes. An object recognition system should just see "tomato". However, we can see much more: we can see peeled, sliced, and cooked tomatoes. We can notice that some of the tomatoes are riper than others, and some are fresh while others are moldy.

Given a collection of images of an object, what can a computer infer? Given 1000 images of tomatoes, can we learn how tomatoes work? In this paper we take a step toward that goal. From a collection of photos, we infer the states and transformations depicted in that collection. For example, given a collection of photos like that on the left of Figure 1, we infer that tomatoes can be undergo the following transformations, among others: ripening, wilting, molding, cooking, slicing, and caramelizing. Our system does this without having ever seen a photo of a "tomato" during training (although overlapping classes, such as "fruit", may be included in the training set). Instead we transfer knowledge from other related object classes.

The problem of detecting image state has received some prior attention. For example, researchers have worked on recognizing image "attributes" (e.g., [10], [24], [23], [11]), which sometimes include object and scene states. However, most of this work has dealt with one image at a time and has not extensively catalogued the state variations that occur in an entire image class. Unlike this previous work, we focus on understanding variation in image *collections*.

In addition, we go beyond previous attributes work by linking up states into pairs that define a transformation: e.g., *raw↔cooked, rough↔smooth, defalted↔inflated*. We explain image collections both in terms of their states (unary states) and transformations (antonymic state pairs). In ad-

* - indicates equal contribution

978-1-4673-6964-0/15/$31.00 ©2015 IEEE

dition, we show how state pairs can be used to extract a continuum of images depicting the full range of the transformation (Figure 1 bottom-left).

Understanding image collections is a relatively unexplored task, although there is growing interest in this area. Several methods attempt to represent the continuous variation in an image class using subspaces [5], [22] or manifolds [13]. Unlike this work, we investigate discrete, nameable transformations, like *crinkling*, rather than working in a hard-to-interpret parameter space. Photo collections have also been mined for storylines [15] as well as spatial and temporal trends [18], and systems have been proposed for more general knowledge discovery from big visual data [21], [1], [3]. Our paper differs from all this work in that we focus on *physical state transformations*, and in addition to discovering states we also study state pairs that define a transformation.

To demonstrate our understanding of states and transformations, we test on three tasks. As input we take a set of images depicting a noun class our system has never seen before (e.g., *tomato*; Figure 1). We then parse the collection:

- Task 1 – Discovering relevant transformations: What are the transformations that the new noun can undergo in (e.g., a *tomato* can undergo *slicing*, *cooking*, *ripening*, etc).
- Task 2 – Parsing states: We assign a state to each image in the collection (e.g., *sliced*, *raw*, *ripe*).
- Task 3 – Finding smooth transitions: We recover a smooth chain of images linking each pair of antonymic states.

Similarly to previous works on transfer learning [6, 4, 19, 28] , our underlying assumption is the transferrability of knowledge between adjectives (state and transformation) (see Fig 2). To solve these problems, we train classifiers for each state using convolutional neural net (CNN) features [8]. By applying these classifiers to each image in a novel image set, we can discover the states and transformations in the collection. We globally parse the collection by integrating the per-image inferences with a conditional random field (CRF).

Figure 2. **Transferrability of adjective:** Each adjective can apply to multiple nouns. *Melted* describes a particular kind of state: a blobby, goopy state. We can classify images of *chocolate* as *melted* because we train on classes like *sugar* and *butter* that have similar appearance when *melted*.

Note that these tasks involve a hard generalization problem: we must transfer knowledge about how certain nouns work, like *apples* and *pears*, to an entirely novel noun, such as *banana*. Is it plausible that we can make progress on this problem? Consider Figure 2. *Melted chocolate*, *melted sugar*, and *melted butter* all look sort of the same. Although the material is different in each case, "meltedness" always produces a similar visual style: smooth, goopy, drips. By training our system to recognize this visual style on *chocolate* and *sugar*, we are able to detect the same kind of appearance in *butter*. This approach is reminiscent of Freeman and Tenenbaum's work on separating style and content [29]. However whereas they focused on classes with just a single visual style, our image collections contain many possible states.

Our contribution in this paper is threefold: (1) introducing the novel problem of parsing an image collection into a set of physical states and transformations it contains (2) showing that states and transformations can be learned with basic yet powerful techniques, and (3) building a dataset of objects, scenes, and materials in a variety of transformed states.

2. States and transformations dataset

The computer vision community has put a lot of effort into creating datasets. As a result, there are many great datasets that cover object [9, 7, 31, 25, 20], attribute [16, 1, 10], material [26], and scene categories [32, 33]. Here, our goal is to create an extensive dataset for characterizing state variation that occurs within image classes. How can we organize all this variation? It turns out language has come up with a solution: adjectives. Adjectives modify nouns by specifying the state in which they occur. Each noun can be in a variety of adjective states, e.g., *rope* can be *short*, *long*, *coiled*, etc. Surprisingly little previous work in computer vision has focused on adjectives [11, 17].

Language also has a mechanism for describing transformations: verbs. Often, a given verb will be related to one or more adjectives: e.g., *to cook* is related to *cooked* and *raw*. In order to effectively query images that span a full transformation, we organize our state adjectives into antonym pairs. Our working defining of a transformation is thus a pair {adjective, antonym}.

We collected our dataset by defining a wide variety of {adjective, antonym, noun} triplets. Certain adjectives, such as *mossy* have no clear antonym. In these cases, we instead define the transformation as simply an {adjective, noun} pair. For each transformation, we perform an image search for the string "adjective noun" and also "antonym noun" if the antonym exists. For example, search queries included *cooked fish*, *raw fish*, and *mossy branch*.

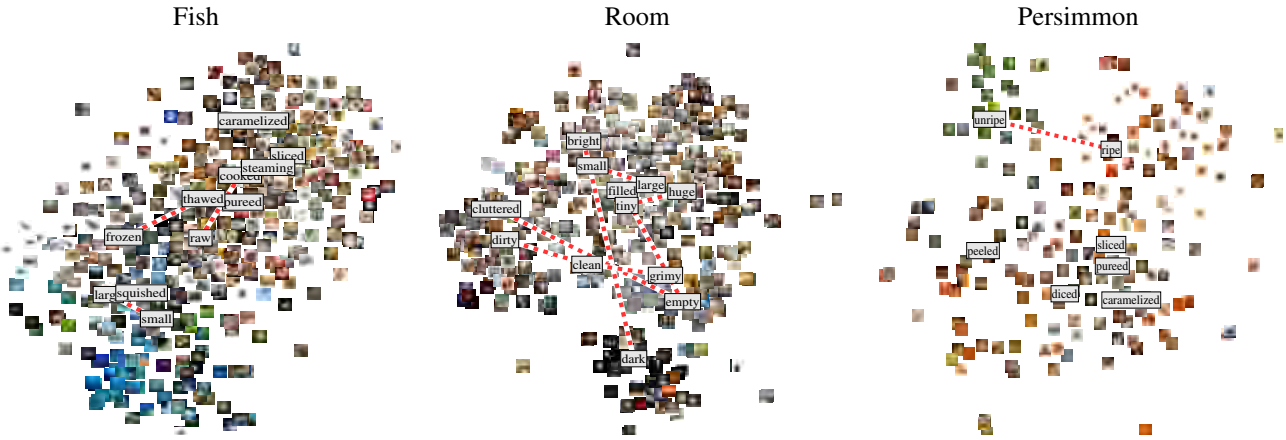

Figure 3. **Example categories in our dataset:** *fish*, *room*, and *persimmon*. Images are visualized using t-SNE [30] in CNN feature space. For visualization purposes, gray boxes containing ground truth relevant adjectives are placed at the median location of the images they apply to. Dotted red lines connect antonymic state pairs. Notice that this feature space organizes the states meaningfully.

2.1. Adjective and noun selection

We generated 2550 "adjective noun" queries as follows. First we selected a diverse set of 115 adjectives, denoted $\mathcal{A}$ throughout the paper, and 249 nouns, denoted $\mathcal{N}$. For nouns, we selected words that refer to physical objects, materials, and scenes. For adjectives, we selected words that refer to specific physical transformations. Then, for each adjective, we paired it with another antonymic adjective in our list if a clear antonym existed.

Crossing all 115 adjectives with the 249 nouns would be prohibitively expensive, and most combinations would be meaningless. Each noun can only be modified by certain adjectives. The set of relevant adjectives that can modify a noun tell about the noun's properties and affordances. We built our dataset to capture this type of information: each noun is paired only with a subset of *relevant adjectives*.

N-gram probabilities allow us to decide which adjectives are relevant for each noun. We used Microsoft's Web N-gram Services[1] to measure the probability of each {adj noun} phrase that could be created from our lists of adjectives and nouns. For each noun, $N \in \mathcal{N}$, we selected adjectives, $A \in \mathcal{A}$, based on pointwise mutual information, PMI:

$$\text{PMI}(A, N) = \log \frac{P(A, N)}{P(A)P(N)}, \quad (1)$$

where we define $P(A, N)$ to be the probability of the phrase "*A N*". PMI is a measure of the degree of statistical association between A and N.

For each noun N, we selected the top 20 adjectives A with highest $\min(\text{PMI}(A, N), \text{PMI}(ant(A), N))$, where $ant(A)$ is the antonym of A if it exists (otherwise the score is just PMI(A,N)). We further removed all adjectives from this list whose PMI(A, N) was less than the mean value for that list. This gave us an average of 9 adjectives per noun.

[1] http://research.microsoft.com/en-us/collaboration/focus/cs/web-ngram.aspx

2.2. Image selection

We scraped up to 50 images from Bing by explicitly querying {adj, noun} pair, in addition to querying by only noun. While we scraped with an exact target query, the returned results are quite often noisy. The main causes of noise is {adj, noun} pairs being either a product name, a rare combination, or a hard concept to be visualized.

Hence, we cleaned up the data through an online crowd sourcing service, having human labelers remove any images in a noun category that did not depict that noun. Figure 3 shows our data for three noun classes, with relevant adjective classes overlaid.

2.3. Annotating transformations between antonyms

While scraped images come with a weak state label, we also collected human labeled annotations for a subset of our dataset (218 {adj, antonym adj, noun} pairs). For these annotations, we had labelers rank images according to how much they expressed an adjective state. This data gives us a way to evaluate our understanding of the full transformation from "fully in state A's antonym" to "fully in state A" (referred to as ranking $ant(A)$ to A henceforth). Annotators split each noun category into 4 sections as the followings. We give examples for $A =$ *open* and $N =$ *door*:

- "Fully A" – For example, fully open door images fall into this category.
- "Between-A and $ant(A)$" – Half-way open door images fall into this category.
- "Fully $ant(A)$" – Fully closed door images fall into this category.

- "Irrelevant image" – For example, an image of broken door lying on the ground.

We ask users to rank images accordingly by drag-and-drop.

3. Methods

Our goal in this paper is to discover state transformations in an image collection. Unlike the traditional recognition task, rather than recognizing an object (noun) or one attribute, we are interested in understanding an object (noun)'s states and the transformations to and from those states. There are various scenarios that we study for this: singe image state classification, relevant transformation retrieval from the image collection, and ordering by transformation.

The common theme is to learn states and transformations that can generalize over different nouns. The reason behind this generalization criterion is from the fact that it is impossible to collect all training examples that can cover the entire space of $\{noun\} \times \{adjective\}$. Hence, in the following problem formulations, we always assume that no image from the specific target noun has been shown to the algorithm. For example, no apple image is used during training if we want to order images for the transformation to *sliced* apple. In other words, we follow the concept of transfer learning.

3.1. Image state classification

First, a simple task is to classify what is the most relevant adjective that describes a single image. Figure 4 shows examples of images in our dataset. Can we tell that the dominant state of Figure 4b is *sliced*? Also, can we tell how *sliced* the apple image it is? As mentioned above, we put a hard constraint that we never saw any apple image (including *sliced* apple) during the training stage. Our goal is to learn what it means to be *sliced* apart from all other nouns and be able to transfer the knowledge to a new category (*e.g.* apple) and infer the state of the image.

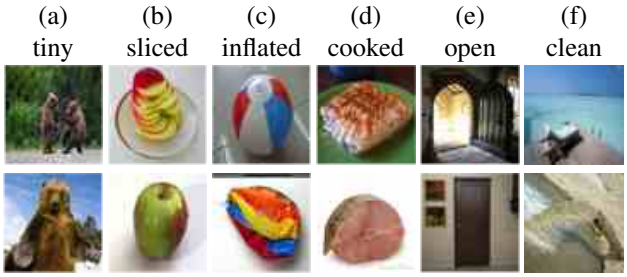

Figure 4. **Examples of objects in a variety of transformed states and their antonym states:** Notice that each state of an object has at least one antonym state.

Our approach for solving this problem is training a logistic regression model. Let $N \in \mathcal{N}$ be the query noun that will be excluded from our test set. Then, using all non-N images, we split them into the positive and negative sets. To train a classifier for adjective $A \in \mathcal{A}$, the positive set is all images of A, and the negative set is all images not related to A. Then, the score of A for image I, denoted $g(A|I)$, can be easily computed by:

$$g(A|I) = \sigma(-w_A^T f(I)), \tag{2}$$

where σ is the sigmoid function, $f(I)$ is a feature vector of I, and w_A is a weight vector trained using a logistic regression model.

It is worth noting that each image can be in the mix of different states (*e.g.* an image of fish can be *sliced* and *raw*). However, for the simplicity, we assume each image has one dominant state that we want to classify.

3.2. Which states are depicted in the image collection?

Figure 5. **Discovering transformations:** our goal is to find the set of relevant adjectives depicted in a collection of images representing one specific noun. In this figure, we want to predict the transformations that describe that describe a collection of apple images.

Our second problem is to discover the relevant set of transformations that are depicted in a *collection* of images. Figure 5 describes our task. We are given a set of *apple* images scraped from the web. While we assume our algorithm has never seen any of *apple* image, can we tell if this image collection contains the transformations between pairs of adjectives and their antonyms – (*sliced*, *whole*), (*chopped*, *whole*), and (*crisp*, *sofa*)?

We now formalize this task. We want to find the best adjective set, $\{\mathcal{A}_j\}_{j \in J}$, that can describe the collection of images, $\{I_i\}_{i \in \mathcal{I}}$, representing a single noun. We abbreviate this set as $\mathcal{A}_J$. Then, our goal is to predict what the most relevant set of adjectives and antonyms describing transformations, $\mathcal{A}_J$, for the given collection of images. In this problem, we constrain all J to have the same size. More formally, we find J by maximizing

$$J = \underset{J', |J'|=k}{\arg\max} \sum_{j \in J'} \sum_{i \in \mathcal{I}} \left[e^{\lambda g(\mathcal{A}_j|I_i)} + e^{\lambda g(ant(\mathcal{A}_j)|I_i)} \right]. \tag{3}$$

Rather than taking the sum over the raw $g(\cdot)$ scores, we take the exponential of this value, with λ being a free parameter that trades off between how much this function is like a

sum versus like a max. In our experiments, we set λ to 10. Thus, only large values of $g(\mathcal{A}_j|I_i)$ contribute significantly to making $\mathcal{A}_j$ appear relevant for the collection.

3.3. Collection parsing

Rather than classifying each image individually, we can do better by parsing the collection as a whole. This is analogous to the image parsing problem, in which each pixel in an image is assigned a label. Each image is to a collection as each pixel is to an image. Therefore, we call this problem *collection parsing*. We formulate it as a conditional random field (CRF) similar to what has been proposed for solving the pixel parsing problem (e.g., [27]). For a collection of images, **I**, to which we want to assign per-image states **A**, we optimize the following conditional probability:

$$\log p(\mathbf{A}|\mathbf{I}) = \sum_i g(A_i|I_i) + \lambda \sum_{i,j \in \mathcal{N}} \psi(A_i, A_j|I_i, I_j) + \log Z, \quad (4)$$

where Z normalizes, g serves as our data term, and the pairwise potential ψ is a similarity weighted Potts model:

$$\psi(A_i, A_j|I_i, I_j) = \mathbb{1}(A_i \neq A_j)\left(\frac{\xi + e^{-\gamma\|f(I_i)-f(I_j)\|2}}{\xi + 1}\right). \quad (5)$$

3.4. Discovering transformation ordering

Each image in our dataset depicts an object, scene, or material in a particular state. Unfortunately, since images are static, a single image does not explicitly show a transformation. Instead, we arrange multiple images in order to identify a transformation. At present, we only investigate a simple class of transformations: transitions from "anytonym of some state A" to "fully in state A" ($ant(A)$ to A).

Figure 6 shows our goal. Given a set of images and an adjective A, we sort images $\{I_i\}$ based on $g(A|I_i) - g(ant(A)|I_i)$ (Eqn. 2).

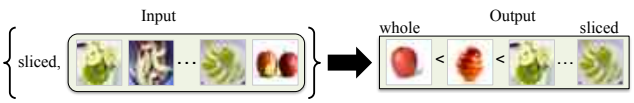

Figure 6. **Discovering transformation orders:** given a particular adjective A and a collection of images, our goal is to sort the images according to the transformation from $ant(A)$ to A. In this figure, we order images from *whole* to *sliced*. Note that we do not use any apple images while training.

4. Results

We evaluate three tasks: 1) Identification of relevant transformations for an image collection. 2) State classification per image. 3) Ranking images by transformation from $ant(A)$ to A.

Figure 7. **Example results on discovering states:** Subset of image collection for each noun is to the left of discovered states. Our system does well on foods, scenes, and clothing (left three collections), but performs more poorly on objects like *computer* (bottom-right).

4.1. Discovering relevant transformations

To implement $g(\cdot)$ (Equation 2), we used logistic regressions trained on CNN features [8] (Caffe Reference ImageNet Model[2], layer fc7 features). Figure 7 shows typical results for retrieval sets of size $|J| = 5$ (Equation 3). In order to ensure most effective examples can be used, we priortize negative examples from nouns that contain the particular adj we are interested in. This type of generalizaiton technique has been explored in [10] as well.

We evaluated transformation discovery in an image collection as a retrieval task. We defined the ground truth relevant set of transformations as those {adjective, antonym} pairs used to scrape the images (Section 2.1). Our retrieved set was given by Equation 3. We retrieved sets of size $|J| = 1..|\mathcal{A}|$. We quantify our retrieval performance by tracing precision-recall curves for each noun. mAP over all nouns reaches 0.39 (randomly ordered retrieval: mAP = 0.11). Although quantitatively there is room to improve, qualitatively our system is quite successful at transformation discovery (Figure 7).

In Figure 9(a), we show performance on several meta-classes of nouns, such as "metals" (e.g., *silver*, *copper*, *steel*) and "food" (e.g., *salmon*, *chicken*, *fish*). Our method does well on material and scene categories but struggles with many object categories. One possible explanation is that the easier nouns have many synonyms, or near synonyms, in our dataset. To test this hypothesis, we measured semantic similarity between all pairs of nouns, using the service provided by [12]. In Figure 9(b), we plot semantic similarity versus AP for all nouns. There is indeed a correlation between synonymy and performance (r = 0.28): the more synonyms a noun has, the easier our task. This makes

[2]http://caffe.berkeleyvision.org

since because it is easier to generalize to a novel noun when the training set contains many similar nouns. We investigate our ability to generalize across dissimilar nouns in Section 4.4.

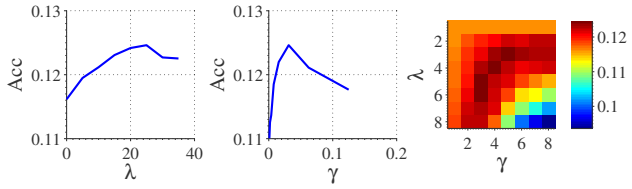

Figure 8. **Performance of CRF over various parameters:** We show the results of our CRF method on the collection parsing problem. The parameters λ and γ correspond to those in Equation 4. Note that the accuracy improves as we increase the weights on pairwise smoothing term.

4.2. State classification

To implement state classification we optimize our CRF model from Equation 4 using the method from [2]. This gives us a maximum a posteriori (MAP) configuration of states per image in the collection. We evaluated the MAP classifications by measuring mean accuracy at correctly classifying the state of each image across all collections. We used the states from our set of discovered transformations as the label space for the CRF. It is also possible to run the CRF with all adjectives as candidates classes. However, using all adjectives does worse: mean accuracy drops from 12.46% to 11.72%. Thus, the relevant transformation discovering acts as a holistic context that improves the classification of each individual image.

In Figure 8 we show how performance varies as a function not the CRF parameters λ and γ (Section 3.3). The rightmost subplot shows the mean accuracy as a function of a grid of settings of λ and γ. The left two subplots show the accuracy profile for λ setting γ to its best value and vice versa. We can consider the data term g alone by setting λ to zero. This performs worse than when we include the pairwise potential, demonstrating that parsing the collection holistically is better than treating each image individually.

Even though state classification per image is quite noisy, because each image collection contains many images, these noisy classifiers add up to give fairly accurate characterizations of entire image collections, as demonstrated by the success of discovering the relevant transformations in the collection (Section 4.1).

4.3. Ranking images by transformation

We also evaluated how well we perform at ranking images from $ant(A)$ to A. As ground truth, we use the transformation annotations provided by human labelers (Section 2.3). We only consider images that fall in the Mid-A section. Our method achieves $\bar{\rho} = 0.46$. We visualize the rankings in Section 4.5. There is ample room for improvement on this difficult task, which we hope will inspire future work.

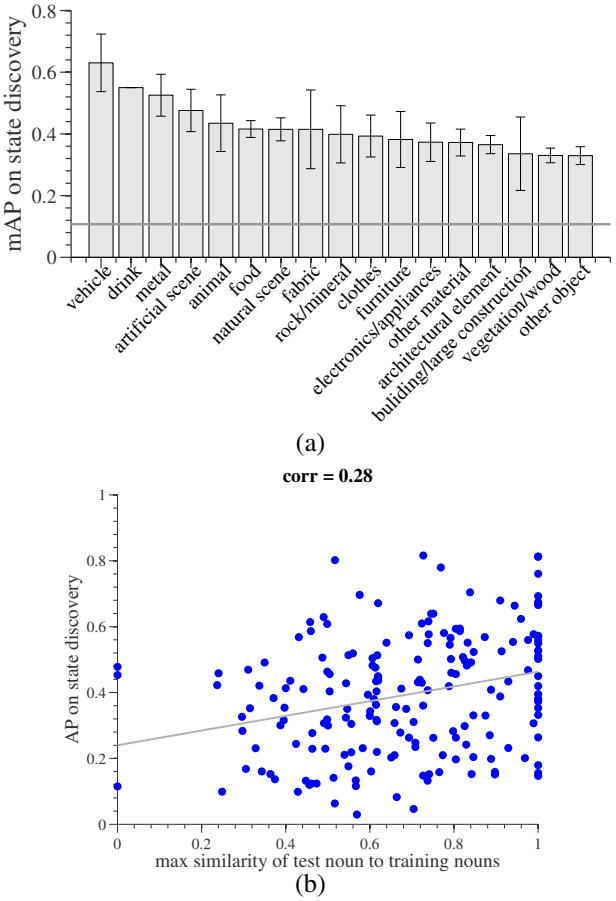

Figure 9. (a) Mean AP at discovering states for different classes of noun. (b) Performance correlates with the semantic similarity between the training set and the test noun, but this does not fully explain why some image collections are easier to understand than others.

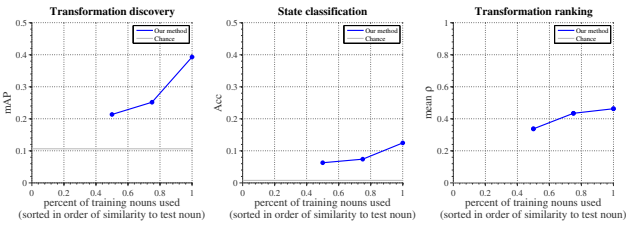

Figure 10. Performance versus percent of training nouns used, taking nouns in order of semantic similarity to test noun. Performance increases as more similar nouns are included in the training set, but is still well above chance even when the training set only includes dissimilar nouns (chance = gray line, estimated by assigning random scores to images).

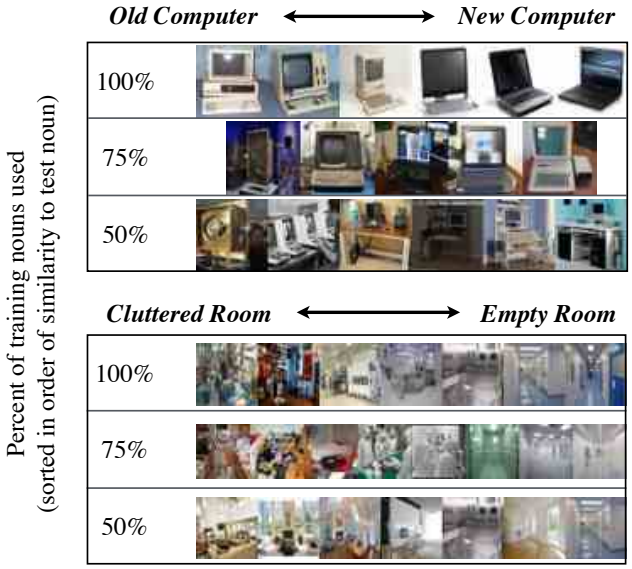

Figure 11. Some classes generalize poorly, others generalize better. In this example we visualize how performance degrades as semantically similar nouns are removed from the training sets for *computer* and *room* (visualized using the "transformation taxi" method from Section 4.5). Notice that *old↔new computer* degrades rapidly whereas *cluttered↔empty* may be more easily generalizable across dissimilar noun classes.

4.4. How well does our system generalize across dissimilar noun classes?

Our tasks are all about transferring knowledge from a training set of noun classes to novel nouns. Sometimes this task is very easy: transferring from *laptop* to *computer* might not require much generalization. To test how well our method generalizes, we restricted our training set, for each query noun, to only include nouns that a certain semantic distance from the query noun. As in Figure 9(b), we again use semantic similarity scores obtained from [12]. In Figure 10, we plot how performance increases, on each of our tasks, as the training set grows to include more and more nouns that are semantically similar to the query noun. Clearly, including synonyms helps, but performance is still well above chance even when the training set only contains nouns quite distinct from the query noun: our system can generalize state transformations over fairly dissimilar noun classes.

We visualize the effect of removing semantically similar nouns in Figure 11. Here we show ranked transformations for *old↔new computer* and *cluttered↔empty room*, using the visualization method described in Section 4.5. As we restrict the training set to include fewer and fewer similar nouns, the qualitative results degrade, as did the quantitative results. However, some classes generalize better than others. For example, *old↔new computer* may rely on having very similar examples in the training set (in particular, *old↔new laptop*) in order perform well; removing *laptop* from the training set has a big effect. On the other hand, many classes can undergo the transformation *cluttered↔empty* and this transformation tends to look alike between classes: a busy textural scene becomes flat and homogeneous in appearance. Correspondingly *cluttered↔empty room* is less rapidly affected by removing similar nouns from the the training set.

4.5. Visualizing transformations

A transformation can be visualized by finding a smooth sequence of images from a starting state (A) to a transformed ending state ($ant(A)$). We use a method similar to "image taxis" [14].

First, we convert the input image collection to a graph. Each image is connected to its k-nearest neighbors in feature space ($k = 5$ in our implementation). For adjective A, we find a path $\mathcal{P}$ through the graph that optimizes the following cost function:

$$\arg\min_{\mathcal{P}} g(A|I_s) + g(ant(A)|I_t) + \frac{1}{|\mathcal{P}|}\sum_{i=1}^{|\mathcal{P}|}\|\mathbf{f}_{\mathcal{P}_i} - \mathbf{f}_{\mathcal{P}_{i+1}}\|_1, \quad (6)$$

where $g(\cdot)$ is given by Equation 2, s is the starting node of $\mathcal{P}$, and t is the ending node. For features we use our adjective classifier scores: $\mathbf{f}_i = \{f_j(I_i)\}_{j=1}^{|\mathcal{A}|}$ and $f_j(I_i) = g(\mathcal{A}_j|I_i)$. In addition, we multiply the feature channel for A and $ant(A)$ by a constant (20 in our implementation) in order to encourage smoothness most of all in that channel. This cost function says: 1) starting image should be A, 2) ending image should be highly $ant(A)$, and 3) path should be smooth in feature space. This cost can be optimized efficiently using Djistra's algorithm. As an additional constraint we only consider values of s among the top 5 images according to $g(A|I_s)$, and t among the top 5 images according to $g(ant(A)|I_t)$.

Example transformation visualizations are shown in Figure 12. Here we show several transformations each for several noun classes. For simple color and transformations, such as *dark↔bright* and *cluttered↔empty*, the method is reasonably effective. For geometric transformations, such as *deflated↔inflated*, the results are much worse. Future work should focus on capturing these difficult types of transformations.

5. Conclusion

In this paper, we have introduced the novel problem of discovering and characterizing states and transformations in image collections. We have shown that simple yet powerful techniques are sufficient to make progress on this problem. We will publicly release our dataset and code to promote further work on this difficult problem.

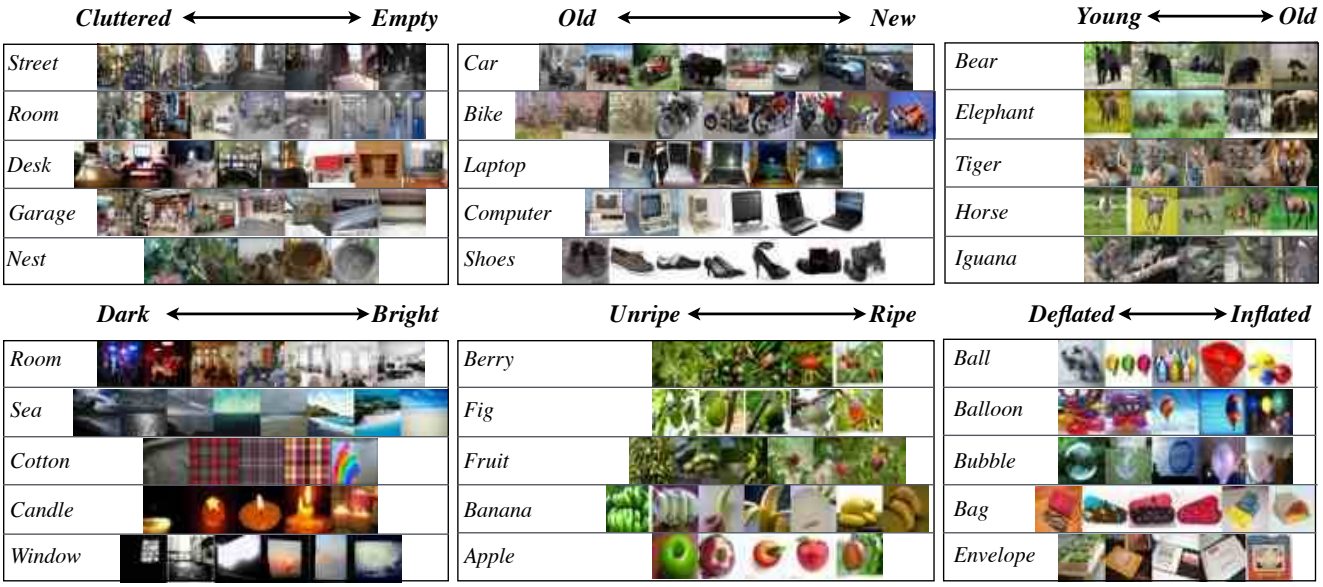

Figure 12. **Example transformation visualizations:** each transformation was discovered from the image collection of a noun class that was not included in the algorithm's training.

References

[1] T. L. Berg, A. C. Berg, and J. Shih. Automatic attribute discovery and characterization from noisy web data. In *Computer Vision–ECCV 2010*, pages 663–676. Springer, 2010. 2

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001. 6

[3] X. Chen, A. Shrivastava, and A. Gupta. NEIL: Extracting Visual Knowledge from Web Data. In *IEEE International Conference on Computer Vision*, 2013. 2

[4] J. Choi, M. Rastegari, A. Farhadi, and L. Davis. Adding unlabeled samples to categories by learned attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 875–882, June 2013. 2

[5] T. F. Cootes, G. J. Edwards, C. J. Taylor, et al. Active appearance models. *IEEE Transactions on pattern analysis and machine intelligence*, 23(6):681–685, 2001. 2

[6] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-scale object classification using label relation graphs. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *European Conference on Computer Vision*, volume 8689 of *Lecture Notes in Computer Science*, pages 48–64. Springer International Publishing, 2014. 2

[7] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 1, 2

[8] 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. *arXiv preprint arXiv:1310.1531*, 2013. 2, 5

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. 1, 2

[10] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 1, 2, 5

[11] A. Gupta. Beyond nouns and verbs, 2009. 1, 2

[12] L. Han, A. Kashyap, T. Finin, J. Mayfield, and J. Weese. Umbc ebiquity-core: Semantic textual similarity systems. *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, 2013. 5, 7

[13] M. J. Jones and T. Poggio. Multidimensional morphable models: A framework for representing and matching object classes. *International Journal of Computer Vision*, 29(2):107–131, 1998. 2

[14] B. Kaneva, J. Sivic, A. Torralba, S. Avidan, and W. T. Freeman. Infinite images: Creating and exploring a large photorealistic virtual space. *Proceedings of the IEEE*, 98(8):1391–1407, 2010. 7

[15] G. Kim and E. P. Xing. Reconstructing Storyline Graphs for Image Recommendation from Web Community Photos. In *27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014)*, 2014. 2

[16] A. Kovashka, D. Parikh, and K. Grauman. Whittlesearch: Image Search with Relative Attribute Feedback. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2012. 2

[17] P.-Y. Laffont, Z. Ren, X. Tao, C. Qian, and J. Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on Graphics (proceedings of SIGGRAPH)*, 33(4), 2014. 2

[18] Y. J. Lee, A. A. Efros, and M. Hebert. Style-aware mid-level representation for discovering visual connections in space and time. In *IEEE International Conference on Computer Vision*, December 2013. 2

[19] J. J. Lim, R. Salakhutdinov, and A. Torralba. Transfer learning by borrowing examples for multiclass object detection. In *Advances in Neural Information Processing Systems*, 2011. 2

[20] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 2

[21] T. Malisiewicz and A. A. Efros. Beyond categories: The visual memex model for reasoning about object relationships. In *Advances in Neural Information Processing Systems*, December 2009. 2

[22] H. Mobahi, C. Liu, and W. T. Freeman. A compositional model for low-dimensional image set representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 2

[23] D. Parikh and K. Grauman. Relative attributes. In *IEEE International Conference on Computer Vision*, 2011. 1

[24] G. Patterson and J. Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 1

[25] B. Russell, A. Torralba, K. Murphy, and W. Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1), 2008. 2

[26] L. Sharan, C. Liu, R. Rosenholtz, and E. H. Adelson. Recognizing materials using perceptually inspired features. *International journal of computer vision*, 103(3):348–371, 2013. 1, 2

[27] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23, 2009. 5

[28] K. Tang, V. Ramanathan, L. Fei-fei, and D. Koller. Shifting weights: Adapting object detectors from image to video. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, pages 638–646. Curran Associates, Inc., 2012. 2

[29] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283, 2000. 2

[30] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 2009. 3

[31] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-ucsd birds 200. Technical Report CNS-TR-201, Caltech, 2010. 2

[32] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 1, 2

[33] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning Deep Features for Scene Recognition using Places Database. *Advances in Neural Information Processing Systems*, 2014. 1, 2