BubbLeNet: Foveated imaging for visual discovery

Kevin Matzen   Noah Snavely
Cornell University
Ithaca, NY, USA
{kmatzen,snavely}@cs.cornell.edu

Abstract

We propose a new method for turning an Internet-scale corpus of categorized images into a small set of human-interpretable discriminative visual elements using powerful tools based on deep learning. A key challenge with deep learning methods is generating human interpretable models. To address this, we propose a new technique that uses bubble images—images where most of the content has been obscured—to identify spatially localized, discriminative content in each image. By modifying the model training procedure to use both the source imagery and these bubble images, we can arrive at final models which retain much of the original classification performance, but are much more amenable to identifying interpretable visual elements. We apply our algorithm to a wide variety of datasets, including two new Internet-scale datasets of people and places, and show applications to visual mining and discovery. Our method is simple, scalable, and produces visual elements that are highly representative compared to prior work.

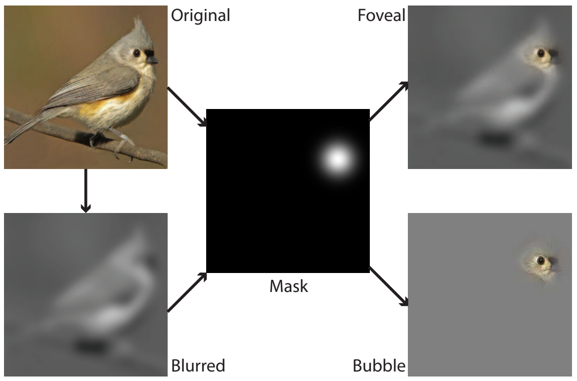


Figure 1. **Foveated imaging.** Our method analyzes the contribution of localized content in images by modifying the data shown to a CNN during training and at test time. Given an input image (top left), we apply an alpha mask (center) to produce two representations: a *foveal image* (top right) and a *bubble image* (bottom right). While a *bubble image* preserves a small amount of spatialized content, a *foveal image* includes a blurred, grayscale background to provide additional context.

1. Introduction

The rapid growth of online imagery and the emergence of powerful machine learning techniques have led to exciting developments in computer vision. While many of these developments have been in visual recognition, these same trends also enable new approaches to *visual discovery*. By visual discovery, we mean a form of visual data mining—identifying patterns or correlations in visual data that can be used to propose or validate hypotheses about the world.

Towards this goal, a few recent papers have explored visual discovery in particular domains. We are especially inspired by the work of Doersch *et al.* [6] where visual elements (i.e., patches) that give cities such as Paris their distinctive look are automatically mined from a corpus of Street View photos. This task is an application of a broader set of methods for mining imagery for distinctive visual elements [18, 5, 11, 1]. Our work also seeks to discover visual elements in the form of patches. However, while most prior work uses hand-crafted features such as HoG, we seek to harness the power of convolutional neural networks (CNNs) for visual discovery. We believe the time is ripe for studying this problem—just as text analysis can reveal trends on social networking sites like Twitter, visual discovery driven by powerful image understanding methods could enable the automatic identification of visual trends in fashion, design, art, and many other areas.

CNNs have shown astonishingly good performance for visual recognition [12, 7]. However, in order to be useful for visual discovery, we need to be able to extract insight about what a CNN has learned, and communicate these insights to humans. Extracting such insights at scale by analyzing learned networks has proved challenging, because CNNs tend to be very opaque. For instance, a given learned unit (neuron) deep in a network can depend on almost all of the input pixels in a highly non-linear way, making it very difficult to associate features in the network with visual elements that are meaningful to humans.

In contrast, rather than trying to interpret a network after the fact, our key contribution is a new type of training process that forces the network to learn much more localized, interpretable properties of the input imagery. Namely, we intentionally restrict how much of the image the network gets to see at training and test time. Inspired by the BubbleBank work of Deng *et al*. [4], the network is only allowed to look at a small portion of the image in high resolution, via a *foveal image*, where the rest of the image is blurred out, or a *bubble image*, where the rest of the image is blank, as illustrated in Figure 1. However, the network can optimize the spatial position of the bubble in order to maximize classification confidence. In this manner, our approach efficiently learns both network weights as well as the most salient parts of each training image. We then cluster these salient image regions to derive a set of visual elements for a given dataset.

Our new approach to finding discriminative visual elements has several key attributes: it is a simple modification of existing CNNs methods, and it scales to large amounts of data, which is critical for many applications. We show a wide range of results on existing datasets, as well as two new large-scale datasets of people and places. We also perform quantitative comparisons of the descriptive power of our visual elements at classification tasks, showing they outperform state-of-the-art algorithms. Finally, we show applications of our method to visual discovery in analyzing people over space and time.

2. Background and Related Work

Our work is in the vein of discriminative patch mining methods [18, 5, 11, 14]. Such methods typically take in a corpus of images and try to identify clusters of "mid-level patches" where each cluster represents an element that appears often in a single category, and rarely in other categories. Each cluster represents a representative, identifiable visual element that is **discriminative** for a class. These clusters can then be used for recognition tasks, or can reveal interesting aspects of the visual world [6]. By examining these clusters, we can get a sense of what visual elements distinguish one category from another. For human interpretability, it is also important that each cluster internally be visually **homogeneous**, i.e., that is has an identifiable theme.

Low-level vs. deep-learned features. Prior discriminative patch methods often build mid-level representations on top of low-level features (e.g., HoG) [18, 5, 11]. However, recent methods that learn features straight from pixels (e.g. CNNs) have enjoyed great success on recognition problems [12, 7]. Can we also use CNNs to learn distinctive visual elements? Raw CNN features (without fine-tuning) have been shown to have useful properties for this task [14]. On the one hand, the representative power of CNNs suggest that they can learn visual elements that are more subtle and with greater invariance to image appearance compared to low-level features. But on the other hand, standard CNN architectures, such as AlexNet [12], are opaque—their features and inner workings are such that can be very challenging for humans to understand and interpret.

Understanding deep networks. Most prior network visualization methods have sought ways to analyze and visualize existing networks. Given enough visual inputs, one can compute and examine distributions over neural activations (i.e., the scalar outputs of individual units of the network). For example, a particular neuron may learn a human face. Another may respond to text. By measuring which neurons are highly correlated with certain types of inputs, we can gain a low-level understanding of the organization taking place [7].

However, other recent results show that reasoning about the semantics of individual high-level units can be misleading, because linear combinations of units are just as meaningful as individual units [21]. Further, evidence suggests that neurons undergo co-adaptation in order to robustly and compactly encode concepts [26]. Indeed, our own experience has shown that it is challenging to reliably identify human-recognizable concepts from the activations of individual neurons. Units at the end of a large network often have very large receptive fields—i.e., regions of input pixels that pool together through the network to effect that unit—perhaps even the entire input image. Many useful visual elements are highly localized (e.g., the beak shape of a particular bird species), and it can be challenging to discern how these local features are represented in the network.

An alternative to direct analysis of network activations is to present controlled image stimuli to the network, and measure the difference in classification performance. For instance, one can slide an occluder (e.g., a gray box) over the input image and see where the performance decreases the most. These regions are identified as salient for a given category [27, 29]. However, these subtractive methods present CNNs with a new kind of image that they have never seen, and there are no guarantees on how the network will perform. We find that it is much better to fine-tune networks using these kinds of modified images as training data. Further, for many types of scenes (e.g., a bookstore full of bookshelves), any occluder location may yield similar classification performance; an additive method like ours is more suitable for identifying the most salient regions.

Our work is also related to the scenes and objects work of Zhou *et al*. [29], which uses a subtractive method to analyze saliency of scenes, and shows that objects emerge as key element of the learned representation. Our goal is a general technique that we apply to a range of datasets.

Unlike these prior methods, we change the way we train the network itself to make the results more interpretable. Our method is in some sense the inverse of the occlusion methods—we obscure or hide all of the image except for a limited region (a "bubble"), and train the network to recognize these obscured images resulting in a new network that we call a *BubbLeNet*. This idea is inspired by work in psychology on using bubbles for human studies, e.g., determining which parts of a human face are most important for gender and expression recognition in people [9]. This idea has been used in the vision community to use humans to identify discriminative elements in fine-grained categories [4]. In our case, we flip this around—rather than learn what humans see, we use bubbles to encourage the network to automatically learn localized, human-interpretable visual elements.

Visual attention. Our work is also related to work on saliency and visual attention. Recent work has used deep learning to attend to salient regions of an image [22, 15], and applied this to applications such as object recognition [2] and automatic caption generation [25]. Our bubble model also uses a form of visual attention to capture more local properties of scenes, but in a much simpler way that can be implemented by modifying the training data.

3. Method

Our method has two main parts: (1) **patch discovery**, where we train a CNN that can localize discriminative visual content and (2) **patch clustering**, where we aggregate the discovered visual elements in a way suited for visualization and for detecting visual elements in new images.

3.1. Patch Discovery

The main contribution of our method is a simple modification to the images shown to the classifier during both training and test time. This modification intentionally restricts the amount of information presented to the classifier in a spatially-localized way. In particular, the classifier is shown a blank background with a single "bubble" of exposed image, as shown in Figure 1 (bottom right). By making this modification, we can analyze the relative performance of the classifier as we move the focal location of this bubble. Effectively, this modification allows us to generate a saliency map, by forcing the network to fixate on regions of the image at a time. Once we have a saliency map, discovering discriminative elements is much easier since we can use that map as a distribution for sampling candidate visual elements.

Figure 1 illustrates this data modification. Given an input image (which we call a "normal" image), we generate an alpha mask with a single Gaussian blob (centered, for now, on a random location). Similar to [9], we use this mask to create a *bubble image* containing only a single colored patch. By modifying a normal image to create several bubble images with different focal locations, we can examine the CNN's prediction scores to determine which parts of the image provide the most evidence for the true class label.

We found that in practice if you take a CNN that has been trained on normal images for a particular task and then test it with bubble images, it will perform no better than random guessing. We found that the network can work much better if it is fine-tuned on **both** normal and bubble images.

An alternative to a bubble image is a *foveal image* which sits somewhere inbetween a normal image and a bubble image. A foveal image is created by first generating a Gaussian-blurred, gray-scale version of the normal image. This blurred image is then composited with the normal image using the same alpha mask used to generate the bubble image (Figure 1, top right). The result is an image which is very similar to the images used in the user study of [4]. Foveal images preserve some contextual information beyond the bubble which potentially makes training easier. However, when we do patch discovery using foveal images directly, the results tend to be misleading since the silhouettes of objects can still be discerned even if the focal point is placed far away. After this *FoveaNet* has converged, these foveal images can be replaced with bubble images and fine-tuned once again to produce a *BubbLeNet*.

To recap, starting from a CNN trained on ILSVRC12 [17], we use a three stage fine-tuning regimen to train a network better suited for spatially localized predictions:

AlexNet/GoogLeNet: Fine-tune the network on 100% normal images.

FoveaNet: Fine-tune the AlexNet/GoogLeNet on 50% normal images and 50% foveal images.

BubbLeNet: Fine-tune the FoveaNet on 50% normal images and 50% bubble images.

We found that in practice, it is sufficient to sample focal locations uniformly at random when generating the foveal and bubble images for training. Hence, generating this extra type of training data is very fast.

Now that we have a CNN trained on bubble images, we can use it to discover salient patches. A naive approach to exploring the saliency of a normal image is to generate a large number of bubble images and pass each through the CNN, recording the score for the true class label. However, this is prohibitively expensive. Fortunately, our choice of alpha mask makes local optimization of focal location quite easy. We can easily compute the gradient of the bubble image with respect to the Gaussian blob's center, and use stochastic gradient descent with the back-propagation algorithm to compute the gradient with respect to the bubble location, moving the bubble along that gradient until we reach a local optimum. In the setting where we wish to mine for discriminative patches, we know the class label for the image, so we can minimize cross entropy between the true label and the prediction distribution. In the setting where we wish to predict the class label for a test image given only one bubble to view at a time, we minimize the entropy of the prediction distribution. We present a more rigorous justification of our algorithm in Section 5.

This optimization has many local minima, so we sample 128 initial bubble locations per image at random and run

SGD for 10 iterations to optimize the position of each bubble according to the gradient of the loss function. The bubble with the minimum loss at the end is selected as the most discriminative bubble for that image.

3.2. Patch Clustering

Having discovered discriminative patches in each photo, we now cluster them to produce better visualizations and to build detectors for finding these patches in new photos.

We apply our BubbLeNet to a validation set of photos and find bubble locations that maximize the correct class label. We then take the top scoring bubble per photo. Each photo in our validation set now has a *normal* image and a *bubble* image as well as corresponding feature vectors extracted from the penultimate layer of our CNN (e.g., fc7 for AlexNet) for both normal and bubble images. We split this set of CNN features into two groups. We denote the these sets S_{normal} and T_{normal}, and the associated bubble image feature sets S_{bubble} and T_{bubble}. Our algorithm then (1) clusters the bubble images into groups of similar bubbles, and (2) builds a classifier for each such cluster that predicts whether a normal image contains that visual element.

This algorithm is based on [18]. We begin by running k-means clustering on the elements of S_{bubble} and use the cluster assignments as labels to train a linear multiclass SVM on S_{bubble}. The resulting SVM is applied to the holdout set T_{bubble}. The class prediction for each element in T_{bubble} is used as its new label and we train a linear multiclass SVM on T_{bubble}. This second SVM is applied to the new holdout set S_{bubble}. These predictions are propagated from the elements of S_{bubble} to their corresponding elements of S_{normal} and a third linear multiclass SVM is trained. This produces an SVM that can take a normal image and predict whether a patch class is contained within.

Unless we have prior knowledge of the number of clusters for a given dataset, we opt to over-provision the clusters, then rank them to determine which ones to keep. Like [18], we build a ranking function based on two measures of cluster quality. The first is entropy: we want a cluster to cover a small number of classes, i.e., have low entropy. The second is confidence: we want the scores of the top activations in the cluster to be high. Therefore, for each cluster we compute entropy minus λ times the sum of the top N SVM scores. We used $N = 5, \lambda = 0.01$ in our experiments. We apply this function to each cluster and rank them accordingly.

The results of this algorithm are a ranked set of bubble image clusters and a detector for each cluster. Because the bubbles in each image have a small footprint, we can also extract a patch from each bubble image as a square region containing the bubble. These can be used for visualization. Figure 4 shows examples of some of these clusters for a variety of datasets (see supplemental for more).

Discussion. Our patch discovery and clustering algorithm has several key features worth noting. It learns a network that can *jointly* classify both normal and bubble images. This property is very useful for building classifiers that predict the presence of a visual element from a normal image. In addition, because the network can model spatial location, the patch clusters we discover not only can encode appearance information, but also spatial distributions (e.g., if a feature usually appears towards the bottom of an image).

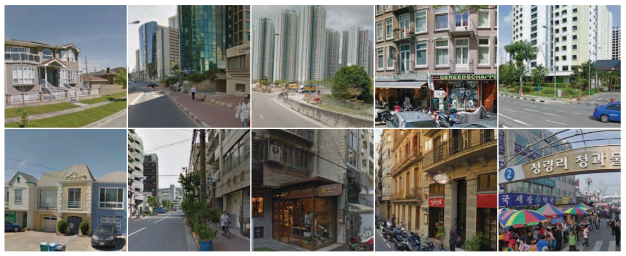


Figure 2. **StreetViewCities dataset.** Several tiles shown from various cities. The number of images per city varies from 102,192 (Venice) to 964,703 (Tokyo). Each 360° spherical panorama has been cropped to produce 7 tiles. Can you guess each city?[2]

Figure 3. **InstagramPeople dataset.** Several examples of cropped people from the InstagramPeople dataset. Networks were trained on crops shown resized to network input dimensions.

4. Datasets

We evaluate on a range of existing datasets, plus two large, new datasets: StreetViewCities and InstagramPeople.

Existing datasets. Our method is designed to highlight discriminatively informative content between classes in a dataset, so large-scale, fine-grained datasets are ideal. **CUB-200-2011** [23] is a standard bird dataset for fine-grained classification. We crop the images with a padded bounding box and we do not use the part annotations. **Food-101** [3] is a fine-grained food classification set. **Indoor67** [16] and **Places205** [30] are datasets of scene images. Places205 is much larger, while Indoor67 has been used in much prior work. Finally, we apply our method to **ILSVRC12** [17].

StreetViewCities. To mine visual elements from cities, in the spirit of [6] (but on a much larger scale), we built a very large-scale database of Street View images of 44 cities. Google Street View photos were acquired by randomly sampling pre-defined regions of the world. First we randomly sampled a city from a short list of populous cities, and then sampled a geodetic coordinate uniformly at random from that city's bounding box. If Street View found a panorama within 50 meters of our query, we retained the preprocessed tiles. Figure 2 highlights tiles from several cities. Each tile was then resized to 256x256. We performed additional filtering for the experiments in this paper. The dataset was subsampled such that no two panoramas are within 30 meters of each other to produce 15.4 million photos. Next, the dataset was further subsampled to provide balanced classes. Finally, a 99-1 training-test split was made with the constraint that all tiles from a single panorama were placed entirely in the same part of the split. This final training set contains about 4.5 million photos. (See supplemental for a full list of cities.)

InstagramPeople. We are also interested in exploring elements, such as fashion, that distinguish *people* in different cities, or at different times, from one another. To that end, we built a dataset of Instagram photos taken in cities around the world, along with people detections in each photo. To build this dataset, we created a list of populous cities and for each city queried the Instagram API for public photos taken within 5km of the city center. We remove exact duplicates, and then each photo is then passed through a deformable part model detector [8] trained on images of people. This people detector has a low false positive rate, but we further cleaned it using MTurk (see supplemental for details).

To provide precise face localization and to filter out photos where a person is facing away from the camera, we ran all of our people detections through the faceplusplus.com face detection API. Each image additionally contains time, place, and weather annotations. Figure 3 shows several example detections used in this dataset. Each detection has been pose normalized such that that face appears in the same location and at the same scale.

InstagramPeople can be used in many different ways. We show results in Table 1 for classifying people in NYC during summer vs. winter, labeled `InstagramSeasons`. The full size of the dataset is 16.2 million people and each image contains on average 2.36 people.

5. Experiments

We now evaluate our mining algorithm from Section 3. In particular, we validate that our three stage fine-tuning protocol makes training a network to recognize entire images from isolated bubbles feasible, that we can mine for the best regions in a dataset automatically, and that those regions are in fact discriminative both qualitatively and quantitatively.

5.1. Raw Classification Performance

We first evaluate the performance of each stage of our classifier on different types of input for each dataset—for instance, testing how well a network trained only on normal images performs on bubble images. These classification numbers are in Table 1 (using two base network architectures, AlexNet and GoogLeNet [20]). All of the accuracies

[2]Top row: Vancouver, Tel-Aviv, Hong Kong, Amsterdam, Singapore. Bottom row: San Francisco, Tokyo, Taipei, Barcelona, Seoul.

		AlexNet/GoogLeNet Normal			FoveaNet Foveal+Normal			BubbLeNet Bubble+Normal
Dataset	Architecture	Normal	Foveal Random	Foveal Best	Normal	Foveal Random	Foveal Best	Bubble Best
CUB-200-2011	AlexNet	0.673*	0.018	0.053	0.561	0.263	0.319	0.481
CUB-200-2011	GoogLeNet	0.810†	0.035	0.147	0.737	0.403	0.459	0.655
Food-101	AlexNet	0.689*	0.034	0.058	0.646	0.341	0.392	0.391
Food-101	GoogLeNet	0.764†	0.042	0.081	0.782	0.451	0.594	0.561
ILSVRC2012	AlexNet	0.574‡	0.009	0.270	0.445	0.210	0.275	0.277
ILSVRC2012	GoogLeNet	0.688§	0.021	0.072	0.435	0.149	0.214	0.235
MIT Indoor67	AlexNet	0.714¶	0.033	0.033	0.688	0.382	0.444	0.450
MIT Indoor67	GoogLeNet	0.714†	0.043	0.067	0.702	0.398	0.533	0.584
MIT Places205	AlexNet	0.506$^{\|}$	0.017	0.025	0.472	0.272	0.312	0.256
MIT Places205	GoogLeNet	0.504†	0.035	0.025	0.495	0.275	0.346	0.311
StreetViewCities	AlexNet	0.692*	0.027	0.021	0.678	0.272	0.326	0.300
StreetViewCities	GoogLeNet	0.690†	0.037	0.036	0.708	0.328	0.397	0.400
InstagramSeasons	AlexNet	0.716*	0.500	0.530	0.721	0.596	0.585	0.618
InstagramSeasons	GoogLeNet	0.716†	0.510	0.495	0.703	0.634	0.634	0.659

Table 1. **Raw classification performance.** A synopsis of classifier performance for different types of input (normal, foveal, and bubble) and each stage of fine-tuning. *Foveal random* indicates that at test time, a single, randomly-sampled focal location is used to process the input that the classifier sees. *Foveal best* and *bubble best* indicate that the network is allowed to search for the focal location that produces the lowest entropy prediction distribution. *Fine-tuned from [10] reference CaffeNet. †Fine-tuned from [10] reference GoogLeNet. ‡Original [10] reference CaffeNet. §Original [10] reference GoogLeNet. ¶Fine-tuned from [30] Hybrid-CNN. $^{\|}$Original [30] Places205-CNN.

for normal images in AlexNet/GoogLeNet are competitive with state-of-the-art results.[3] For all datasets, results were computed on the test set, with the exception of ILSVRC12 which was computed on the validation set. This table highlights several interesting properties of our bubble training method:

Normal networks don't do well on foveal images. If a AlexNet/GoogLeNet (normal-image-trained) classifier is given a foveal image, it will nearly randomly guess, even if the focal location is optimized to produce more confident predictions (column "foveal best"). This motivates the use of fine-tuning to adapt the network to a configuration that can predict both normal and foveal/bubble images.

Foveal images can be classified with a FoveaNet. The FoveaNet numbers show that it is indeed possible to produce a network that has non-trivial performance on images with such restricted content. Many of our datasets have FoveaNets that can perform almost half as well as a AlexNet/GoogLeNet applied to the full normal images (an exception being ILSVRC12).[4] In addition, by optimizing the focal location to minimize prediction entropy ("foveal best"), foveal performance for each FoveaNet improves significantly, indicating that this "saliency landscape" is something that we can explore using SGD and backprop to drive

[3] In terms of Indoor67 and Food-101, our networks exceed prior published methods [14] [3]. Baseline networks from which we fine-tuned were provided by [10] and [30]. Therefore, we replicate their reported performance on ILSVRC12 and Places205. We do fall slightly short for CUB-200-2011, but it does come very close to the best reported result of 68.29% [28] (no parts oracle-ft setting).

[4]Datasets such as Indoor67, Places205, and Food-101 are constructed in such a way that looking at any random location in the photo will often give some clue as to the correct class label. However, datasets such as StreetViewCities often have sections such as the sky that are common across all classes.

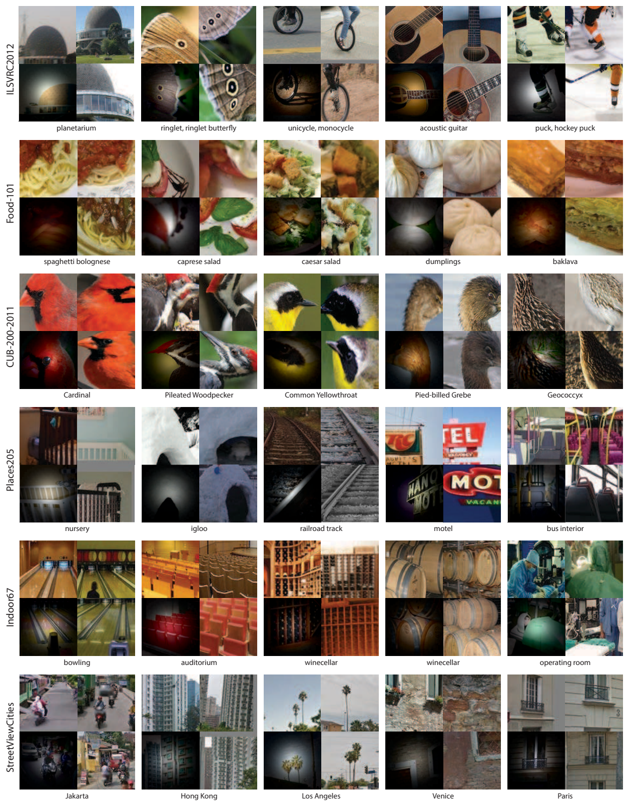


Figure 4. **Extracted patch clusters.** Patches extracted from clusters with low entropy. We show one example in the lower left of each cluster with a mask indicating the size of the Gaussian bubble used during discovery. The majority class label is shown below each cluster.

the focal location.

Bubble images can be classified with a BubbLeNet. BubbLeNet shows the performance of the classifier used to mine patches after the entire background has been removed. This is the classifier we use for our other evaluations. The performance of classifying optimized bubble images is significantly above chance. This shows that we can indeed classify images based on finding single discriminative bubbles. Interestingly, in some cases bubble image performance exceeds that of the foveal images (e.g., for CUB-200-2011). Adding foveal images also sometimes boosts classification performance for normal images (e.g., Food-101 with GoogLeNet).

5.2. Qualitative Results

In order to get a visual sense of what kinds of elements our algorithm discovers, we ran our patch discovery and clustering algorithm to generate visual elements for each dataset. Figure 4 shows a few patches from five clusters of each dataset. For CUB-200-2011, often the best place to look was the birds' heads. However, for some species such as the Geococcyx it is better to look at the body.

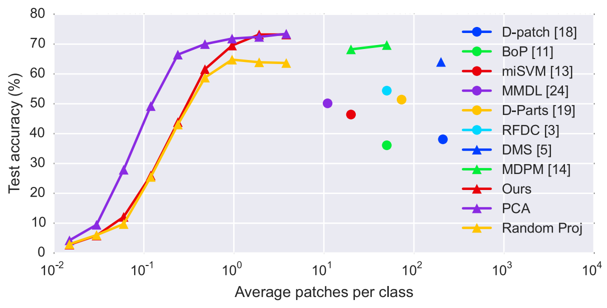
For some ILSVRC12 classes it is difficult to actually see the object anywhere in the image. For one such class (hockey puck), our patch selection method does highlight, however, that it is important to look near the ice by the players, rather than elsewhere, such as the players' helmets. Some classes, such as the winecellar in Indoor67, have multiple important components. Figure 4 shows one component highlighting racks of bottled wine, and another component highlighting wine barrels. Most of these high-scoring patch classes are easily identifiable as a particular element. Like Doersch *et al*. [6], we find metal railings in Paris as a distinctive element, as well as interesting types of features in other cities (e.g., motorbikes in Jakarta).

5.3. Discriminability

While our main goal is to extract a human interpretable representation of categories, we also wish to evaluate our method quantitatively—how good of a representation of the data does our method produce? To facilitate comparison with prior work [18, 5, 14, 13, 24, 19, 3] we evaluate using Indoor67. There are often two goals at hand. One is to show that the patch mining method itself produces a set of mid-level features that are well-suited for learning a classifier. However, our baseline CNN produced by fine-tuning outperforms all prior work. Perhaps more interesting is how well the mid-level representation acts as a method of compression. For instance, how well can we encode the 67 classes in Indoor67 with as few visual elements as possible?

Figure 5 shows the trade-off between representation size and discriminability for a range of methods, including our own (red line). To create an image encoding with our method, we use a method inspired by the BubbleBank algorithm [4]: we take the top K scoring bubble clusters produced by our algorithm, and, for a given input image run each of the classifiers for these clusters on the input image. These K classifier outputs are then stacked into a vector, and a new SVM is learned for categorizing based on this representation. Compared to bag-of-words methods, our approach detects each visual element once on the entire image, and also takes into account any spatial information encoded in the classifier.

Figure 5 (top) plots the performance of our method as a function of number of patches per class, and similarly plots the results of other published methods on this dataset. The results of running PCA or random projections on the penultimate CNN vectors are plotted for comparison (note that these representations are not sparse). As the figure shows, our method requires only a couple of patches per scene class to exceed the performance of the 20-50 patches per class used in prior work (the closest competitor is the MDPM algorithm, which also uses deep learning [14]). This suggests that (a) scenes in Indoor67 can often be classified accurately with only a couple canonical elements per scene, and (b) the invariances baked into the CNN are what make the good performance possible. Despite these invariances, we find



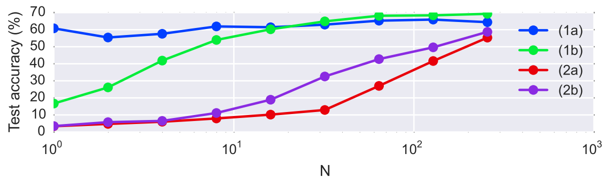


Figure 5. **Indoor67 evaluation.** *Top*: Our patch selection method allows us to select a very small set of semantic elements as a representation, yet still retain good classification performance. *Bottom*: We validate two alternate BubbleBank design choices. (1) We take the BubbleBank feature vector (a) or PCA applied to the penultimate layer of the CNN (b) and perserve the top N features of each descriptor while setting the remaining to zero. For example, $N = 1$ would produce a feature vector with exactly 1 non-zero located in the position of the component with largest magnitude. (2) We use only the top N ranked bubble features given by (a) a random ranking or (b) our ranking.

that the visual elements are still understandable. Please see the supplemental material for a more comprehensive set of patches shown in Figure 4.

Figure 5 (bottom) shows the effect of (1) sparsifying BubbleBank feature vectors rather than using PCA on the penultimate CNN layer and (2) using our ranking to select which clusters are to be included in the BubbleBank feature vector to build more concise representations. (1) shows that only a few non-zeros in this vector are needed for good classification suggesting the representations are much sparser compared to dimensionality-reduced vectors from PCA and (2) shows that incorporating clusters according to our ranking leads to better classifiers than selecting clusters at random.

5.4. Localization

We perform two experiments to evaluate the localization power of our method. In the first experiment, we measure how closely the most discriminative part per photo in the CUB-200-2011 dataset matches to a part annotation. If a element is close to a part, that suggests we are correctly localizing near an object of interest. We measure this by normalizing image coordinates to $[0, 1] \times [0, 1]$ and computing the Euclidean distance between the centroid of the optimal bubble and its nearest visible part. We compare this to random guessing. The histogram in Figure 7 shows

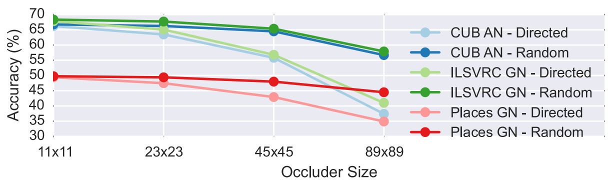


Figure 6. **Occlusion analysis.** Given a AlexNet/GoogLeNet classifier, we analyze how much the classification score is impacted by occluding parts at random and as guided by our discriminative bubbles.

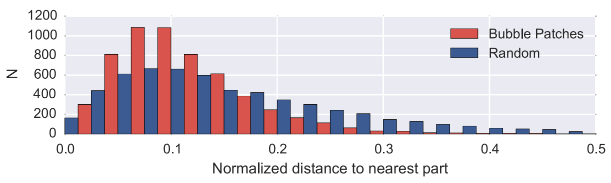


Figure 7. **Part localization.** Normalized distance to closed part for the bubble location that maximizes correct class label on CUB-200-2011. Our bubble method chooses locations closer to part annotations more often than random guessing.

that the distributions of element-to-part distances clearly differ where random guessing has a median of 0.140 and our method has a median of 0.091.

The second experiment measures the performance of the original AlexNet/GoogLeNet classifiers applied to unaltered images and to images that have had a grey occluder box inserted as in [27]. Bubble images can be seen as a additive method; the occluder box is a subtractive method. In this experiment, we select one location for this occluder per image based on the most discriminative part detected by our bubble method. Figure 6 compares the performance of several AlexNet/GoogLeNet classifiers on bubble-guided occluders and random occluders. The performance drops significantly, and much more if the occluder is guided by our bubble method than by random guessing, suggesting we are locating salient features.

5.5. Exploring Visual Data

The ultimate goal of our method is as a tool for visual discovery. To conclude, we use our technique to explore visual elements of our datasets across time and space.

How do people change over a year? There are many interesting questions about the appearance of people over time, relating to fashion, weather, and so on. As a simple experiment, we took photos of people from NYC from our InstagramPeople dataset and learned a bubble image classifier that predicts the month when the photo was taken. Rather than train an n-way classifier, we opted to train a shared CNN to produce an intermediate representation on which a large set of one-versus-one logistic regressors could be trained; the entire network was fine-tuned end-to-end. The results are shown in Figure 8 as a matrix of one-to-one test accuracies. Some observations: adjacent months are hard to differentiate, and there are clusters of months corresponding to similar seasons (summer and winter). Furthermore, by examining the off-diagonal, we find that the classifier can differentiate between summer and winter.

However, two adjacent months—March and April—are much more distinguishable than other pairs. What has the network learned? We ran our algorithm on the task of NYC March vs. April. It was clear from the extracted patches (shown in the supplemental material) that clothing had become lighter, but the top-ranked cluster was sunglasses. By running our detector for this cluster on the 2014 March+April portion of our dataset, we recorded a sunglasses score (percent photos with thresholded score) for each photo. We also collected weather condition reports from wunderground.com and categorized them: clear, cloudy, and precipitation. We found a strong correlation between the occurrence of sunglasses and the weather condition. Namely, the Pearson's r of the "sunglasses" signal to the "clear weather" signal is 0.487 and of the "sunglasses" signal to the "cloudy weather" signal is -0.593 (see Figure 8). This illustrates an end-to-end use of our algorithm to explore trends in a dataset.

How do people change over space? Style varies across space as well as time. Can we discover variations in personal fashion between different locales? To explore this question, we ran our method to classify the NYC and London subsets of `InstagramPeople`. Figure 9 shows several top-ranked clusters of patches. Accuracy is 60.8%. Sports caps, and stripes and certain text on shirts are more discriminative for NYC, while collared and textured shirts and dark, shiny jackets are more discriminative for London. More visualizations can be found on our project website.[5] Some clusters are specialized for color, some hair, and others clothing. However, despite high purity, some clusters remain difficult to interpret suggesting more is required to guarantee interpretability than enforcing spatial compactness alone.

Limitations. These initial experiments required that we use some intuition to figure out what signals might be correlated. An automated discovery system might be able to search through a diverse body of data to find correlations. We also present additional qualitative results in the supplemental material highlighting spatial trends in StreetView photos for which we do not have a way to automatically validate. Finally, the high invariance of the CNN features can still cause some clusters to be **too** invariant and appear noisy.

6. Conclusion

We presented a simple, scalable, new method for discovering visual elements in large datasets that directly builds on CNNs via a new form of modified training data. We believe that such techniques can start to open the door to new kinds of visual discovery. In the future, it would be intriguing to build on our preliminary work to explore fashion, design, art, and other visual arenas to identify differences across populations, as well as trends and influences. There are also a number of other interesting areas for future work. One key problem is that humans still have to interpret the clusters—this one represents hats, another represents a certain pattern, etc. Doing this kind of labeling automatically—perhaps using text or other cues found on the web, would further enable automatic interpretation of discovered elements.

[5] www.cs.cornell.edu/projects/bubblenet/

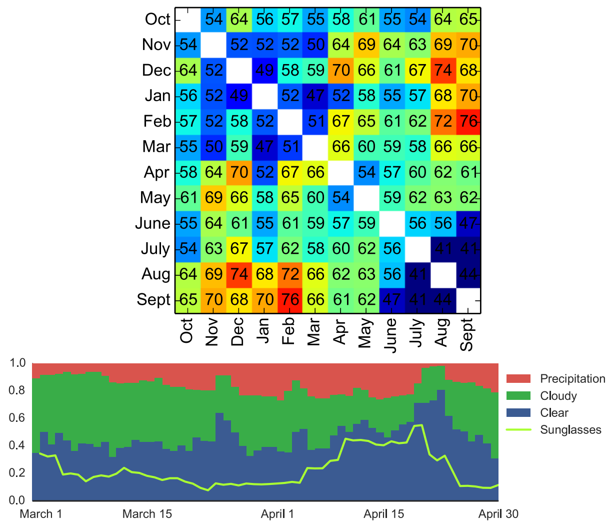


Figure 8. **Finding temporal correlations.** *Top*: NYC one-versus-one month classification accuracies. *Bottom*: We found sunglasses to be a discriminative element that makes April in NYC visually distinct from March and correlated them to weather reports.

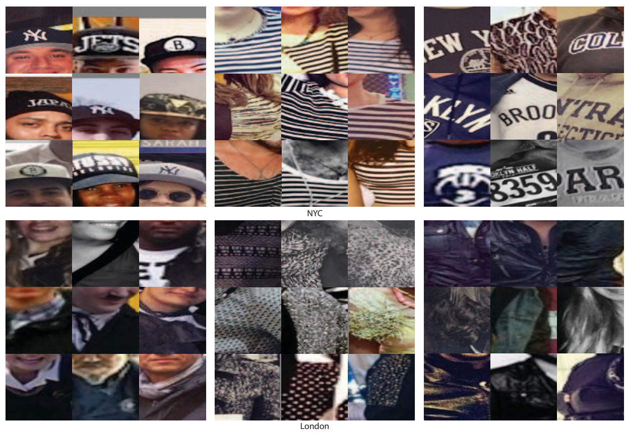


Figure 9. **NYC vs. London patches.** We used our method to find what makes people in NYC visually distinct from London. We show clusters correlated with NYC (top) and London (bottom).

7. Acknowledgements

This work is partly funded by NSF grant IIS-1111534, Google, and Intel ISTC-VC.

References

[1] S. Arietta, A. Efros, R. Ramamoorthi, and M. Agrawala. City forensics: Using visual elements to predict non-visual city attributes. *IEEE Trans. Visualization and Computer Graphics*, 20(12):2624–2633, Dec 2014. 1

[2] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. *CoRR*, abs/1412.7755, 2014. 3

[3] L. Bossard, M. Guillaumin, and L. Van Gool. Food-101 – mining discriminative components with random forests. In *ECCV*, 2014. 4, 5, 6

[4] J. Deng, J. Krause, and L. Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *CVPR*, pages 580–587, June 2013. 2, 3, 6

[5] C. Doersch, A. Gupta, and A. A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*, 2013. 1, 2, 6

[6] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros. What makes paris look like paris? *SIGGRAPH.*, 31(4):101:1–101:9, 2012. 1, 2, 4, 6

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2

[8] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/~rbg/latent-release5/. 5

[9] F. Gosselin and P. G. Schyns. Bubbles: a technique to reveal the use of information in recognition tasks. *Vision Research*, 41(17):2261 – 2271, 2001. 3

[10] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 5

[11] M. Juneja, A. Vedaldi, C. V. Jawahar, and A. Zisserman. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, 2013. 1, 2

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 2

[13] Q. Li, J. Wu, and Z. Tu. Harvesting mid-level visual concepts from large-scale internet images. In *CVPR*, 2013. 6

[14] Y. Li, L. Liu, C. Shen, and A. van den Hengel. Mid-level deep pattern mining. In *CVPR*, 2015. 2, 5, 6

[15] V. Mnih, N. Heess, A. Graves, and k. kavukcuoglu. Recurrent models of visual attention. In *NIPS*. 2014. 3

[16] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *CVPR*, pages 413–420, June 2009. 4

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *CoRR*, abs/1409.0575, 2014. 3, 4

[18] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*, 2012. 1, 2, 4, 6

[19] J. Sun and J. Ponce. Learning discriminative part detectors for image classification and cosegmentation. In *ICCV*, pages 3400–3407, Dec 2013. 6

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. 5

[21] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. 2

[22] Y. Tang, N. Srivastava, and R. R. Salakhutdinov. Learning generative models with visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1808–1816. Curran Associates, Inc., 2014. 3

[23] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 4

[24] X. Wang, B. Wang, X. Bai, W. Liu, and Z. Tu. Max-margin multiple-instance dictionary learning. In *ICML*. ICML, June 2013. 6

[25] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015. 3

[26] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014. 2

[27] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 2, 7

[28] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. In *ECCV*, volume 8689, pages 834–849. 2014. 5

[29] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *CoRR*, abs/1412.6856, 2014. 2

[30] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning Deep Features for Scene Recognition using Places Database. *NIPS*, 2014. 4, 5