Visual Knowledge Discovery using Deep Learning

Gunnar Atli Sigurdsson
Carnegie Mellon University
gsigurds@cs

Shi Hu
Carnegie Mellon University
shihu@cs

Abstract

Most recent efforts in computer vision harness the ever increasing availability of information on the Internet. In this work, we are interested in developing a framework that automatically mines images from the Internet and develop "knowledge" from those images. We adopt CNNs to discover the high level concepts in images, and use introspection with NLP techniques to describe those concepts in words. Finally, we show that the system can successfully discover some interesting relationships between concepts.

1 Introduction

There is consensus in the community that using the crowd for labelling, such as the ImageNet database [5] and Visipedia [14], has its limits. The problem can be best addressed using unsupervised knowledge discovery from the web-scale data. We are interested in developing a framework that automatically mines images from the Internet, and try to develop "knowledge" from those images. A simple type of knowledge lies in relationships between images, such as "hoof is a part of horse", or "lemon is yellow". Given a scalable framework that has millions of images and is continously gathering more, we hope it could provide an ever-increasing knowledge base and aid further efforts in computer vision.

Deep learning has emerged as one of the best approaches for many machine learning problems, specifically in computer vision. Since the release of the work by Krizhevsky et al. [11], the performance of most standard benchmarks in vision has improved. Deep learning frameworks have the ability to represent extremely complicated functions in an elegant framework. An important property of the deep network is that it can represent the functions that a shallower network may need exponentially more parameters to represent. [1] An interest feature of the deep networks is that they can learn all representations automatically. In vision, this is everything from simple edge and texture detectors to high-level visual concepts, such as a human face. This property is very appealing for a knowledge discovery framework, since it makes no modeling assumptions on the data. Furthermore, since the network combines those representations in order to make a prediction and learn new concepts, there is an inherent structure of the visual knowledge present in the network that this work will try to extract and improve. The representation the convolutional neural networks learn has been shown to take visual classes which are very convoluted at the image level, and mostly separated in the higher layers [18]. This props for the use of high-level similarity metrics in the hidden layers. Interestingly, unlike the mixture-of-linear-classifiers framework of NEIL [4] that has to explicitly cluster the visual classifiers to address the multimodel appearances of the concepts, deep networks automatically capture the "polysemy" of the word concepts, and will for example automatically learn the difference in appearance between the fruit "apple" and the corporate logo "Apple". Extracting and utilizing these automatic visual clusters has the potential to discover interesting visual clusters automatically.

While there still remains unanswered theoretical questions about deep learning, such as the optimal network structure, the convolutional neural nets (convnets) are currently state-of-the art in many vision problems. There exists approaches to visualize different hidden layers in the network by

"running the network backwards" (deconvolutional nets) [18] in order to hallucinate an highest scoring input for a given neuron. Other approaches may search over all training images in order to find the image, or image patch that gives the highest score for a particular neuron [11].

In order to autonomously absorb new knowledge to the knowledge base, we will let the system gather data from a list of proposed concepts on its own, and learn the concepts in those images. Specifically, we will train the system using a set of images where each image is associated with some textual description. When the system sees a new image, it will summarize this new image in words using high-level concepts based on the knowledge it has discovered so far. Furthermore, we use the existing corpus, and the high-level feature embedding, which is optimized to maximize separation between visual concepts, and learn new concepts from clusters in this space. The concepts can be new, in the sense that 1). it is a combination of old concepts, 2). it is something that is already in the knowledge base but just has not been noticed yet. Then, the system queries Google using either more keywords from the proposed list or from the discovered cluster, and acquires more images with similar ideas. The main goal of this project is a long term one, in this paper we present a proof-of-concept that demonstrates what kind of concepts can be learned and how the system works. Specifically, we determine the higher level meanings in images and describe them in words. For example, after seeing women and men, the system can recognize the general notion of "face".

2 Related Work

A popular approach to mine visually salient patches from a set of images, aims to find discriminative patches in the data [15]. This approach, for example used to find iconic architectural landmarks for various cities [6], is a clustering approach, but still only works directly on patches of pixels, and thus cannot represent the complex invariations that a deep network can. A deep network has the ability to find interesting representations that are important for the data in a discriminative manner.

In the AlexNet paper [11], similar images are found by extracting the last hidden layer of their network, and using Euclidean distance between these vectors for different images. These image similarities are able to capture various invariances of the data, and are much more interesting than a simple kernel that compares pixel values directly. In this work we hope to explore this property further in order to discover interesting relationships.

The work of Chen et al. on NEIL [4] was a seminal work in this direction, but the work makes significant modeling assumptions and has to manually build all visual knowledge from its linear classifers. The method requires every single concept to be manually fed into the model in terms of words, such as "horse", or "apple". Furthermore, the ability to recognize image attributes is based on a hard-coded list of such attribute-concepts. The hope is that a deep learning framework can learn the representations and relationships in a model free manner. This is an example of *extrospection*, the act of being given the visual knowledge in a supervised manner. We will aim to do this unsupervised.

Many previous works in computer vision have demonstrated effective ways to annotate an image. Socher and Fei-Fei [16] used a semi-supervised learning algorithm to perform this task using few labeled data. Specifically, they mapped both the textual words and image segments into a low dimensional latent meaning space using kernelized canonical correlation analysis. If the word and image segment are close in this space, then they are likely to share the same sematic concept. In testing, the learned mapping can be used to annotate new images. Blei and Jordan [2] developed the correspondence latent Dirichlet allocation model that finds conditional relationships between latent variable representations of set of image regions and sets of words. Under this model, the regions of the image can be conditional on any set of factors, but the caption words must be conditional on the factors that are present in the image. Thus, the model is better able to find correspondence between words and regions, compared to other related but simpler methods, such as Gaussian-multinomial mixture model and Gaussian-multinomial LDA. These techniques successfully combines the information in an image with a document to annotate the image. In contrast, we are interested in summarizing a group of similar images using their associated documents, under the assumption that these documents may just loosely relate to the image cluster.

3 Visual Knowledge

What does it mean to learn knowledge from a set of images? If we have a system that understands images, we can expect it to possess the following abilities:

- Generate keywords from an image, i.e., $\mathbb{R}^{N\times M\times 3} \ni I \mapsto \{w\}$
- Generate an image using keywords, i.e., $\{w\} \mapsto I \in \mathbb{R}^{N\times M\times 3}$
- Extract relationships, such as $C_1 \subset C_2$ or $C_1 \sim C_2$
- Able to compute image similarities, i.e., $I_1 \times I_2 \mapsto \mathbb{R}$

Here, I is an image, and $\{\text{w}\}$ is a set of words, and C is a concept. We are interested in creating such an automatic framework for understanding various aspects of images without making any explicit modeling assumptions. The goal is to continously feed the framework with random concepts, and allow the framework to learn these concepts, refine them, and discover new concepts from its knowledge. The framework should have a large collection of visual concepts, and be able to describe those concepts in words. Furthermore, the framework should discover relationships between concepts automatically.

There are two methodologies to knowledge discovery. One is through *extrospection*, where the learner is given some examples and told to learn their appearance. The other is through *introspection*, where the learner tries to make sense of the data and knowledge from what it already knows. In this work, we are interested in the second approach. We start with a deep convolutional neural network initialized using a large database of annotated images, the ImageNet [5]. To initiate the autonomous learning process, we need to explicitly show a few concepts to the learner at the beginning. Specifically, we pick a word or a combination of words that capture a concept, and let the system search the Internet to gather a few thousand weakly label images, which could be loosly related to the concept. To learn the visual appearance of the concept, an output neuron is added to the label layer, and the existing network is fine-tuned to learn the weights of the new data using a variant of the classic backpropagation algorithm [9]. By storing images for each concept, we follow the approach of NEIL [4] and create a co-occurrence matrix for all concepts we have learned so far. This co-occurrence matrix can be used to mine object-to-object, object-to-scene, object-to-attribute, and scene-to-attribute relationships from the data. Examples of such relationships are "eye is a part of dog", or "water is blue". Here, we assume the system can classify a concept as an object, a scene, or an attribute.

Using the output of a classifier $Pr(C_j^{(i)})$, where $C_j^{(i)}$ is the occurrence of concept j in image i, we can define the probability $Pr(C_j^{(i)}, C_k^{(i)})$, which is the probability that both concepts j and k occur in image i, and then marginalize over images to find probability $Pr(C_j, C_k)$. Similar to set overlap, we can determine if the concepts overlap, or if one concept almost exclusively occurs with another concept using similarity and part relationships. The entire pipeline is shown in Figure 1.

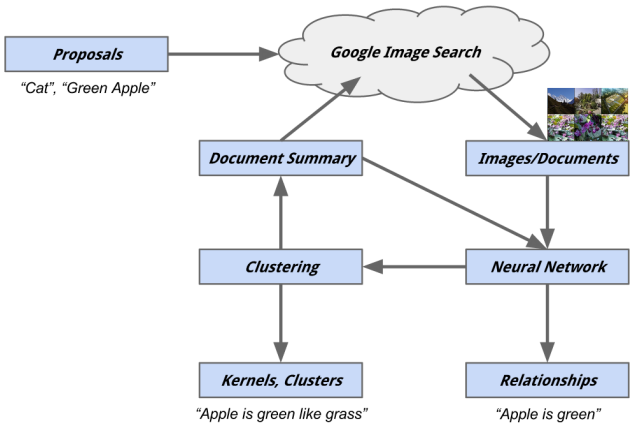

Figure 1: The computational pipeline.

Given this ever increasing framework, we can start to make use of the high-level representations that are learned by the network, which are the high-level layers that contain visually salient concepts, such as the face of a cat [12]. While in the standard image space, it might be challenging to compare two images, the task becomes easier in the high-level feature embedding. Figure 2 illustrates this point. To compare that both of them have an area of something blue (water), if we are in the standard image space, we would have to consider a large number of patches since these concepts are not localized. However, in the high-level feature space, these properties have proven to be localized, therefore we can use simpler similarity kernels to compare them [18].

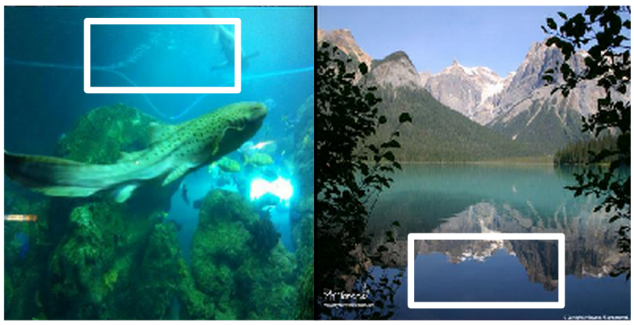

Figure 2: Image-level similarity. "Both images contain something blue"

In this framework, we use two approaches. The simpler approach hypothesizes that each neuron in the highest hidden layer corresponds to a visual concept. To introduce a simple notation, our network maps an image I_1 to a feature vector x_1, which is the vector of the neuron activations during inference.

$$\Phi(\cdot) : I_1 \mapsto x_1 \in \mathbb{R}^K \tag{1}$$

We follow some prior work in visualizing these layers [17, 8], and get a collection of images from the dataset that has the highest score for this neuron.

$$\max_{\mathcal{I} \subset \mathcal{I}_0, |\mathcal{I}|=100} \sum_{I \in \mathcal{I}} |\Phi(I)^{(j)}| \tag{2}$$

Here, $\mathcal{I}$ is a set of images, and $\Phi(I)^{(j)}$ is the output of the j-th neuron. Using the textual descriptions of these images, we try to extract common concepts from these images. This allows us to use words to identify a visual concept that has been extracted from the data. It might be a new concept, such as a forest, even though the concept of forest has not been introduced to the framework. Or this might be a visual cluster that is more specific to the concept than the one originally introduced to the network. For example, we might learn to associate keywords such as "steve jobs", "apple" and "mac" to the visual cluster corresponding to the "Apple" logo, even though we only introduced the word "apple" to the network.

The more interesting approach, instead of picking just a single neuron, which corresponds to picking an indicator function over all the responses in a given hidden layer, we consider a random subset of the neurons, along with a similarity kernel. Furthermore, instead of doing this only once, we employ a mode-seeking algorithm by finding a random subspace, picking the best examples, and updating the projection as the mean of the examples. This is hypothesized to converge towards an interesting concept in the network. At each iteration we perform the following steps:

$$\mathcal{I} = \operatorname*{arg\,max}_{\mathcal{I} \subset \mathcal{I}_0, |\mathcal{I}|=100} \sum_{I \in \mathcal{I}} |\langle w, \Phi(I) \rangle| \tag{3}$$

$$w = \frac{1}{100} \sum_{I \in \mathcal{I}} \Phi(I) \tag{4}$$

where w is a weighting vector. Given we have associated each neuron, or combination of neurons with some keywords, for a new image, we can find the keywords that are close to it in the neuron activation space. In section 4 we elaborate on this idea and extend it to more interesting concepts.

We shall do the above procedure in a loop. It allows us to discover new concepts from the data and find names for them. These concepts can then be added to the co-occurrence matrix. Further, when a visual concept has been learned, we can revisit the concept learning process and refine our knowledge of that concept. An interesting advantage of this approach is that this combines visual knowledge with word knowledge and has the potential to learn a great number of classifiers through zero-shot or one-shot learning by making use of the sub-concepts [7].

4 Approach

4.1 Discovering Clusters

We take advantage of the fact that it is easier to separate concepts in the high-level feature space. To discover clusters in our visual knowledge base, we maximize a kernel density estimate with a mode-seeking algorithm, specifically mean-shift. This contrasts with other clustering algorithms in the sense that we simply want to discover a single cluster for computational reasons, in order to try to capture some part of the space. Since there is no single optimal answer when looking for some properties of the visual knowledge space, we actually prefer to simply find a local optimum. Given a fixed similarity kernel that compares the images in some way, the kernel density estimate is as follows:

$$f(\mathbf{x}) \propto \frac{1}{n} \sum_i K(\mathbf{x}, \mathbf{x}_i) \tag{5}$$

$\max_{\mathbf{x}} f(\mathbf{x})$ has a known solution using the mean-shift algorithm, which corresponds to finding a mode of the kernel density estimate.

However, it is challening to come up with a kernel that captures something interesting. We propose to pick a random set of similarity kernels, and maximize the objective function under certain conditions to find the best kernel in that region. Since the activations of the hidden layers of the network are sparse, we know that the activations of single neurons in the hidden layers can correspond to interesting concepts such as cat faces [12]. We constrain the kernel to only compare a small number of neurons. That is, the kernel is sparse. The optimization criterion is as follows:

$$\max_{\mathbf{x},\mathbf{k}} f(\mathbf{x}; \mathbf{k}) \propto \frac{1}{N} \sum_i K(\mathbf{x}, \mathbf{x}_i; \mathbf{k}) \text{ s.t. } \|k\|_0 \leq T \tag{6}$$

where $K(\mathbf{x}, \mathbf{x}_i; \mathbf{k})$ is a similarity kernel between $\mathbf{x}$ and $\mathbf{x}_i$ with hyperparameters $\mathbf{k}$.

For the scaled inner product kernel $K(\mathbf{x}, \mathbf{y}; \mathbf{k}) = \mathbf{x}^T D_{\mathbf{k}} \mathbf{y}$ (where $D_{\mathbf{k}}$ is a diagonal matrix with $\mathbf{k}$ on the diagonal) we have the following optimization problem

$$\max_{\mathbf{x},\mathbf{k}} f(\mathbf{x}; \mathbf{k}) \propto \max_{\mathbf{x},\mathbf{k}} \frac{1}{N} \sum_i \mathbf{x}^T D_{\mathbf{k}} \mathbf{x_i} \text{ s.t. } \|k\|_0 \leq T \tag{7}$$

$$= \max_{\mathbf{x},\mathbf{k}} \mathbf{w}^T \mathbf{k} \text{ s.t. } \|k\|_0 \leq T \tag{8}$$

$$\tag{9}$$

where $\mathbf{w} = \frac{1}{N} \sum_i \mathbf{x}_i D_{\mathbf{x}}$. This is optimized with an alternating scheme as follows:

$$\mathbf{k} = \arg\max_{\mathbf{k}} \mathbf{w}^T \mathbf{k} \text{ s.t. } \|k\|_0 \leq T \tag{10}$$

$$k_i = 1 \text{ if } |\{w_i < w_j, j \neq i\}| \leq T \tag{11}$$

and $\max_{\mathbf{x}} f(\mathbf{x}; \mathbf{k})$ is as before. While it is hard to compare the clusters qualitatively, we observed that these clusters capture more specific concepts than naively doing mode seeking in the feature space, and tended to be more variable than for maximizing single neurons.

For the Gaussian similarity kernel $K(\mathbf{x}, \mathbf{y}; \mathbf{k}) = \exp\left[-(\mathbf{y} - \mathbf{x})^T D_{\mathbf{k}} (\mathbf{y} - \mathbf{x})\right]$ we have the following optimization problem:

$$\max_{\mathbf{k}} \mathcal{L}(\mathbf{x}, \mathbf{k}) = \max_{\mathbf{k}} \exp\left[-(\mathbf{x}_i - \mathbf{x})^T D_{\mathbf{k}} (\mathbf{x}_i - \mathbf{x})\right] \text{ s.t. } \|\mathbf{k}\|_0 \leq T \tag{12}$$

$$= \max_{\mathbf{k}} \exp\left[-(\mathbf{x}_i - \mathbf{x})^T D_{\mathbf{k}} (\mathbf{x}_i - \mathbf{x})\right] + \lambda \|\mathbf{k}\|_1 \tag{13}$$

$$= \max_{\mathbf{k}} \exp\left[-\mathbf{w}_i^T \mathbf{k}\right] + \lambda \|\mathbf{k}\|_1 \tag{14}$$

where $\mathbf{w}_i^{(j)} = (\mathbf{x}_i^{(j)} - \mathbf{x}^{(j)})^2$. We find the subgradient:

$$\frac{\partial}{\partial k}\mathcal{L}(\mathbf{x},\mathbf{k})^{(j)} = \frac{1}{N}\sum_i e^{\mathbf{w}_i^T\mathbf{k}}(-\mathbf{w}_i^{(j)}) \text{ if } \operatorname{sign}(k^{(j)}) \neq 0 \tag{15}$$

$$\in \left[\frac{1}{N}\sum_i e^{\mathbf{w}_i^T\mathbf{k}}(-\mathbf{w}_i^{(j)}) - \lambda, \frac{1}{N}\sum_i e^{\mathbf{w}_i^T\mathbf{k}}(-\mathbf{w}_i^(j)) + \lambda\right] \text{ if } \operatorname{sign}(k^{(j)}) = 0 \tag{16}$$

Here we use subgradient ascent to optimize for $\mathbf{k}$, and $\max_{\mathbf{x}} f(\mathbf{x};\mathbf{k})$ as before.

To guide the subgradient ascent we can use an upper bound on $f(\cdot)$ using Jensen's inequality and only accept a subgradient step in each dimension if $w_i^{(j)}k'^{(j)} - \lambda|k'^{(j)}| \geq w_i^{(j)}k^{(j)} - \lambda|k^{(j)}|$, where $\mathbf{k}'$ is the new $\mathbf{k}$. This helps the subgradient ascent step to ascend.

4.2 Naming Clusters

We've previously mentioned that we can use the image descriptions to extract concepts from a cluster. In this section, we will describe a procedure to accomplish this.

First, all images are obtained from some web pages online. We assume the documents on most web pages can relate to their images to some degree; hence it is natural to use them as the image descriptons. To extract the useful content from raw html files, we use "boilerpipe" [10]. Once we obtain a document from a html file, we only retain adjectives and nouns using WordNet [13] without the stopwords.

We then concatenate all documents together to form a single document, and run latent Dirichlet allocation (LDA) [3] on this document. We understand this approach has its drawbacks. However, if we use the alternative approach, which is to run LDA on every single document, then it can be difficult to weigh the keywords extracted from each individual document and identify the most relevant ones. We calculate the importance of each word as the product of its topic weight and its weight under this topic. We then sort the scores and return the top scoring words as the keywords.

Because the web pages are obtained randomly, we might encounter a cluster where most images are not associated with the their documents. For example, the background images of blogs. Using the words in the blogs cannot help much understand the images. Thus, we will need to rely on Google reverse image search to find a best guess of an image, search Wikipedia for the most relevant page about this guess, and then use this Wikipedia page as the image description. Though this approach can work in practice, all experiments shown in section 5 use the original documents.

In summary, since we can associate for each visual concept with some keywords, we could both map an image to words, and synthesize an image from given words (by maximizing the image input given a node). Furthermore, the natural structure of the network enables us to extract relationships between the concepts, and by mining the co-occurrence matrix we can discover other concept relationships. Finally, given our neuron responses we can construct similarity kernel. This address all of the "knowledge" requirements described at the begining of section 3.

5 Experiments

To implement this deep learning framework, we use the CAFFE [9] toolbox with the AlexNet architecture [11] pre-trained on the ImageNet database. Pre-training has proven to be an important way to initialize the network which avoid local minima in the non-convex training phase. To add concepts to the network, we iteratively add a batch of concepts to the network as new labels, and fine-tune the network to learn the new labels. The network is fine-tuned using stochastic gradient descent with momentum. Using the added data, we run the network and build a co-occurrence matrix of the detections.

To test the framework, we learn the following additional 10 concepts shown in figure 3 in the initialized net, and observe what relationships we can extract if we have around 700 images obtained from Google image search for each concept:

- beach, beach resort, cat, forest, man, pet, resort, swimming, woman, swimming hall

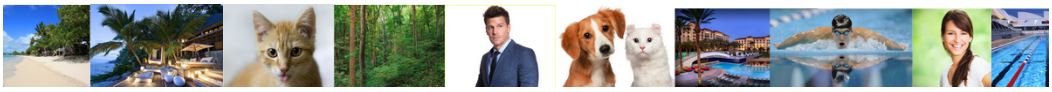

Figure 3: Examples of the test classes

We split the data equality into training and test. Using the AlexNet architecture this takes around 8 hours to train on a Tesla K40 GPU (by fine-tuning on top of the network pre-trained on ImageNet but discarding all the previous label nodes), and yields a test set accuracy of 0.60. Furthermore, we gathered another dataset of 10k images from Google image search with 1000 random classes. To see if our framework allows us to refine already learned concepts, we find a cluster of images in both the training dataset and the random dataset closest to the average of the training class in the feature space. Using the clusters we train another CNN classifier. Training took around 9 hours on a Tesla K40 GPU. We observe the following result

- Accuracy on test split before refining: 0.60
- Accuracy on test split after refining: 0.56

we see that finding a single cluster seems to be insufficient to refine the detector. We suspect that this is due to the loss of variability in training data, and hope to be able to remedy this by using multiple clusters.

We use the learned feature representation by extracting the "fc7" hidden layer activations from the network, which forms a 4096 dimensional feature space for each image. We use the clustering approaches described above to get the clusters in figure 4. By varying the hyperparameters we get different types of clusters, and using random restarts we get various clusters. Namely, by increasing the sparseness of the kernel, we get more concrete concepts, at the cost of occasionally finding concepts that is difficult to understand from visualizing the images. For example, using the inner product kernel with no sparsity, we exclusively converge to faces, nature or cars in our training data. By increasing the sparsity, we converge towards more interesting results. To avoid the issue of setting the width parameter of the kernels, we set the 100 nearest neighbors to evaluate as non-zero.

We attempt to extract a name for a cluster using the documents it contains. To obtain a cluster, we sample 100 representative images from an image category, and learn their appearance using a CNN classifier (fine-tuned on the previous one). The test accuracy on the new classes was 0.58. Using those new concepts, we attempt to discover relationships by running the detectors on the test dataset. The top 10 relationships obtained from co-occurrence matrix are the following:

- beach resort like resort
- resort like beach resort
- *face* part of woman
- *mountains* part of resort
- *shiny* part of *circular*
- *blue* part of swimming
- beach part of beach resort
- *face* part of man

where the concepts in italics are discovered by the network, and not a part of the 10 concepts introduced into the network in the extrospection phase.

6 Conclusion

In this work, we explored the idea of using deep learning to discover visual knowledge through introspection. We demonstrate the ability to find new concepts through learning similarity kernels and clustering the data. The concepts that are being learned are only limited by the choice of kernel and amount of data. We attempted to refine the existing classifers, but the resulting classifier did

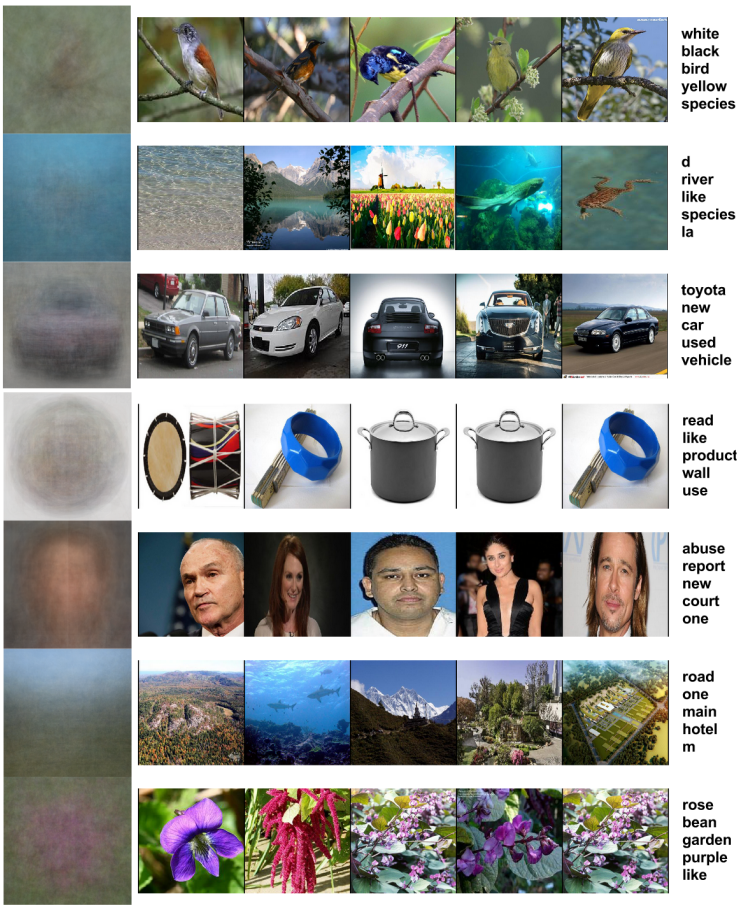

Figure 4: Example clusters found in the data, where the leftmost image of each row is the average image of the cluster, followed by five random samples from the cluster. Finally there are the top five keywords extracted from the documents using LDA.

not manage to improve on the baseline. We demonstrated how the discovered concepts could be fed back to the pipeline and used to discover novel relationships. Finally, since this was done with convnets, by iteratively adding concepts in an online manner with fine-tuning, and noting that the convnet efficiently shares parameters between concepts, the framework is very scalable.

References

[1] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[2] David Blei and Michael Jordan. Modeling annotated data. In *ACM SIGIR Conference on Research and Development in Informaion Retrieval, 2003. SIGIR 2003*, pages 127–134. ACM, 2003.

[3] David Blei, Andrew Ng, and Michael Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Neil: Extracting visual knowledge from web data. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1409–1416. IEEE, 2013.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[6] Carl Doersch, Saurabh Singh, A Gupta, J Sivic, and AA Efros. What makes Paris look like Paris? *ACM Trans. Graph.*, 2012.

[7] Li Fei-Fei, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):594–611, 2006.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2013.

[9] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[10] Christian Kohlschtter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow text features. In *ACM International Conference on Web Search and Data Mining, 2010. WSDM 2010.* ACM, 2010.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[12] Quoc V Le. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8595–8598. IEEE, 2013.

[13] George Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[14] Pietro Perona. Vision of a visipedia. *Proceedings of the IEEE*, 98(8):1526–1534, 2010.

[15] Saurabh Singh, Abhinav Gupta, and AA Efros. Unsupervised discovery of mid-level discriminative patches. *Computer VisionECCV 2012*, 2012.

[16] Richard Socher and Li Fei-Fei. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *Computer Vision and Pattern Recognition, 2010. CVPR 2010. IEEE Conference on*. IEEE, 2010.

[17] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.

[18] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2018–2025. IEEE, 2011.