Content-Based Image Retrieval Using Deep Belief Networks

By

Jason Kroge

Submitted to the graduate degree program in the Department of Electrical Engineering and Computer Science of the University of Kansas in partial fulfillment of the requirements for the degree of Master's of Sciences.

_____
Dr. Brian Potetz (Committee Chair)

_____
Dr. Xue-Wen Chen (Committee Member)

_____
Dr. Bo Luo (Committee Member)

_____
Date Defended

The Thesis Committee for Jason Kroge certifies
that this is the approved version of the following thesis:

Content-Based Image Retrieval Using Deep Belief Networks

_____
Dr. Brian Potetz (Committee Chair)

_____
Dr. Xue-Wen Chen (Committee Member)

_____
Dr. Bo Luo (Committee Member)

_____
Date Defended

Acknowledgements

I would like to thank Dr. Brian Potetz for his advice and guidance during the past two years. This thesis would not have been possible without his help. I would also like to thank Dr. Xue-wen Chen and Dr. Bo Luo for serving as members of my committee.

Finally, I would like to thank Dr. Hinton for supplying source code for training deep belief networks.

Abstract

With the shear amount and variety of digital images available in the world today, people need an effective method to search for any particular image. The commonly used strategy of searching by keyword has several problems, especially when searching for aspects that are difficult to describe with words. In this paper, I will discuss an image retrieval system that can be used to search for visually-similar images based on image content rather than associated keywords. I will discuss the major components of this system including a pre-processing step using Haar wavelets and the steps for training a deep belief network to recognize higher-order features that may have a semantic or category specific meaning. The paper concludes with a comparison of performance between the newly proposed system and other published results.

Contents

1. Introduction

There are millions upon millions of digital images that exist on the World Wide Web and within the private domain. As each day passes, this number increases, and it will continue to increase as more and more people utilize computers and the Internet. As the number of images increases, it becomes increasing difficult to find a specific image or to locate images related to a specific class.

The most commonly used searching strategy is to index the images with keywords. However, this approach has many downsides. It requires a person to manually label all the images with tags or keywords which can be a slow and arduous task. Another problem with the keyword approach comes from the fact that some visual aspects of images are difficult to describe, while other aspects could be described in more than one way. It may also be difficult for the user to predict which visual aspects have actually been indexed for searching.

In this paper, I will describe a system that allows a user to search based on image content. The user will begin by presenting the system with a target image. This image could be digitally drawn, or it could be the result of an initial image search using keywords. First, the system will go through a pre-processing phase to extract information at varying scales of detail. Next, the system will feed that information into a multi-layered neural network which will respond with a group of images that is most similar to the target image.

One type of use for this system is *search by example* [6] which aims the search at a specific image. The search may be for an exact copy of the image in mind, for example searching within an art catalog. On the other hand, the search may be for another image of the same object or scene of which the user already has an image. The search is applied when the user has a specific image in mind and the resulting images are similar to the given example. This system is best suited to search for art, stamps, industrial components, and other catalogs in general.

Another type of use for this system is *category search* [6] which aims at retrieving images that are representative of a particular class. It may be the case that the user has an example and the search is for other elements that are within or related to the same class. For example, a business or some other entity may want to check their newly designed logo to ensure that it is not similar to an already existing image that has been trademarked. This system is also well-suited for images of natural scenes.

2. Background and Related Work

In this section, I will provide details about the major components used within the image retrieval system.

2.1 Haar Wavelet Transform

Haar wavelets offer a mathematical way of encoding image pixel data so that the data is layered according to level of detail. It can be thought of as a sampling process in which values of the transform matrix act as samples of finer and finer resolution.

To demonstrate how to transform a matrix, I will first describe a method for transforming an array of data. This method is often referred to as *averaging and differencing* [10]. Later, we will use this method to transform an entire matrix.

We start with our original data as seen in Figure 1. For each pair of numbers, we calculate the average and place those values in the first half of the next row. Next, we calculate the difference between each pair of numbers, and we place those values in the second half of the next row.

Original Data	56	74	132	150	156	130	147	147
After Round 1	65	141	143	147	-9	-9	11	0
After Round 2	103	145	-38	-4	-9	-9	11	0
After Round 3	124	-21	-38	-4	-9	-9	11	0

Figure 1. Example Haar wavelet transform of an array.

After one round, we can see that the averages between the original pairs are 65, 141, 143, and 147. The differences between the original pairs are -9, -9, 11, and 0.

This process is repeated, using the averages generated from the previous round as data for the next round. The process is complete, when there is only one average value remaining. All other values in the array represent differences on various levels of detail. As you may have already guessed, the discrete wavelet transformation works best on arrays of length 2^k and requires k rounds of averaging and differencing.

Now that it is clear how to transform an array, it is very simple to apply the process to a matrix. You simply treat each row as an array, and perform the averaging and differencing on each one to obtain a new matrix, and then apply exactly the same steps on each column of this new matrix, finally obtaining a row and column transformed matrix.

Another way to picture this is to do each of the row transformations, transpose the matrix, do the row transformations on the result of that transposition, and then finally transpose the matrix back. The result is called the *Haar wavelet transform*.

2.2 Restricted Boltzmann Machine

A Boltzmann Machine is a recurrent neural network that consists of stochastic binary units. It is similar to a Hopfield Network, but differs with its stochastic nature. These random variations in the data are useful for optimization problems because they can help escape from a local minimum.

A Restricted Boltzmann Machine has a condition stating that the units in a layer can not be connected to each other. This creates visible units v ε {0, 1} and hidden units h ε {0, 1} that form an undirected bipartite graphical model as can be seen in Figure 2. The visible units will receive data from the training set. The hidden units will represent some higher order information. Each unit is represented by the energy function:

$$E(v,h) = -\sum_{i,j} w_{ij} v_i h_j - \sum_i b_i v_i - \sum_j c_j h_j$$

where w_{ij} is a weight representing the strength between unit i and unit j, v_i is the visible state of unit i, h_j is the hidden state of unit j, and b_i and c_j are bias parameters for v_i and h_j, respectively.

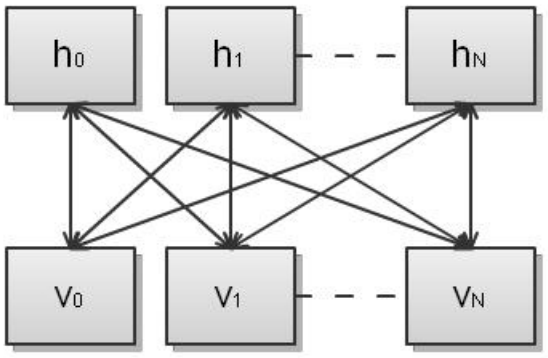

Figure 2. Restricted Boltzmann Machine

The stochastic nature of the system comes from the probability of turning on unit i, which is given by:

$$p_i = \frac{1}{1 + \exp(-b_i - \sum_j s_j w_{ij})}$$

where b_i is the bias parameter of the unit, s_i is the current state of the unit, and w_{ij} is a weight representing the strength between the unit and another unit connected to the one in question.

The network is run by repeatedly choosing each unit and setting its state according to the probability function. To train the network according to external data, we need to set the weights so that the global states with the highest probabilities will receive the lowest energies. Eventually, the network will reach an equilibrium state.

To approximate the external distribution, we can measure the difference between the external distribution and the model-generated distribution using the Kullback-Leibler divergence. This learning rule is the same as maximizing the log probability of the data [4]. At this point, the gradient of the log probability for the training data is given by:

$$\frac{\partial \log p(v^0)}{\partial w_{ij}} = \left\langle v_i^0 h_j^0 \right\rangle - \left\langle v_i^\infty h_j^\infty \right\rangle$$

where $< \cdot >$ denotes the average over the states.

2.3 Deep Belief Network

A Deep Belief Network simply consists of multiple layers of Restricted Boltzmann Machines. The first, or the lowest layer, is an input data vector. There can be any number of layers in the middle. Lastly, the top two layers have undirected, symmetric connections which form an associative memory.

The network is trained one layer at a time by treating the hidden units of the previous layer as the input for the next layer. This continues for as many layers as needed until the top layer is reached. After the training phase, another learning procedure is applied to adjust the weights in each layer in order to improve the performance of the entire network.

During the testing phase, the weights of the network are refined by calculating the mean-squared error between the modeled output and the target value. The goal is to further improve the network by minimizing the cost of the connections using gradient decent. This technique is commonly referred to as backpropagation. Figure 3 depicts a 768-500-500-1000-26 network architecture that was used in the experiment. The network is trained from the bottom up and refined from the top down.

Given that the number of features in each layer does not decrease, it has been shown that a deep belief network increases the lower bound on the log probability of the data [4]. A deep hierarchical model is capable of learning complex relationships between the features in the layers below.

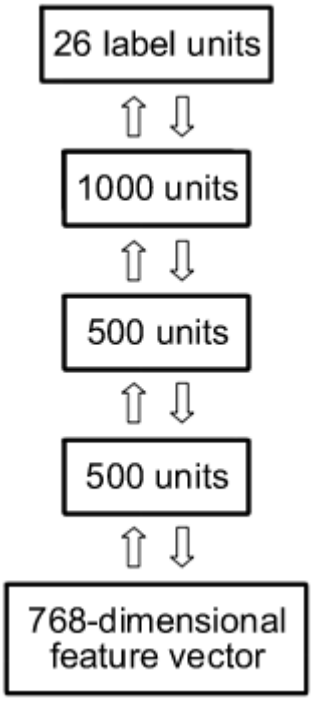

Figure 3. Deep Belief Network with 768-dimensional input vector and 26 output classes.

3. Methodology

3.1 Dataset

Images from the Caltech 101 Dataset were used to test the system. This dataset contains pictures of objects belonging to 101 different categories. For my experiment, I choose to use a subset of this dataset. I choose this because a number of the categories contained objects in overly busy environments, and some of the categories did not contain enough sample images for accurately training and testing a deep belief network.

For the experiment, I used 26 of the 101 categories. These 26 categories contained 1008 images. Figure 4 shows sample images from a few of the image classes. For each class, the images were randomly split into two equal sized groups. One group was used for training the network while the other group was used for testing the network. On average, there were about 20 training images for each category. It was important to use a separate set of images for testing in order to show that the network did not simply memorize the training set, but instead show that it is capable of correctly classifying new images.

Figure 4. Example images from the Caltech 101 dataset.

3.2 Pre-processing

The images in the dataset are of various sizes, but on average, each image is roughly 300×200 pixels. Initially, each image is represented by a w × h × 3 matrix, where w represents the width, h represents the height, and the 3 represents the number color channels in the RGB colorspace. Each value in the matrix ranges from 0 (no intensity) to 1 (full intensity). Since the Haar wavelet transform works best on images that are 2^k pixels in width and height, each image is scaled to $2^k \times 2^k$ pixels. This may create minor distortions, but the images in each category are roughly the same dimensions, so each image within a category will be distorted in the same manner.

After an image has been appropriately sized, each color channel is individually decomposed using the Haar wavelet transform. The result of this transform is a matrix containing the average intensity of the color channel for the image in the first position. All other elements will contain a number representing the difference in intensity across regions of various sizes within the original image. Finally, the transformation data from all three color channels are combined into a single vector. For example, if k is chosen to be 4, the length of the vector will be $2^4 \times 2^4 \times 3 = 768$.

3.3 Training the Networks

After the pre-processing, each 16 × 16 image is represented by a vector of length 768. These vectors are presented as input to the deep belief networks. During the experiment,

I trained the networks with varying parameters including several different image sizes and several different configurations of nodes in the hidden layers.

I used two different methods for training the deep belief networks. The first method used supervised learning. During the training phase, the class label for each of the training images is used for adjusting the weights. The second training method used unsupervised learning. During the training phase, the network has no knowledge of which class the image belongs to. The network will group the images into classes that are best fit.

4. Results

In order to achieve the best results, I trained several different network configurations on the same dataset of 1000 images within 26 image classes. After each network had been trained, I calculated the resulting class label by applying the learned weights. Figure 5 and Figure 6 show the accuracy of label assignment to images in the training set and images in the test set for each network configuration.

Figure 5 shows that the best performing network used an image size of 16 × 16 with a configuration of 500 first-level hidden units, 500 second-level hidden units, and 1000 third-level hidden units. This network achieved 72% accuracy on the test set. For images with a size of 16 × 16, the accuracy generally goes down as the number of hidden units increases. This observation seems reasonable because there are fewer details that can be represented with a small image size. With fewer details, it is harder to make

higher-level connections, and the extra hidden units begin to contribute to additional error.

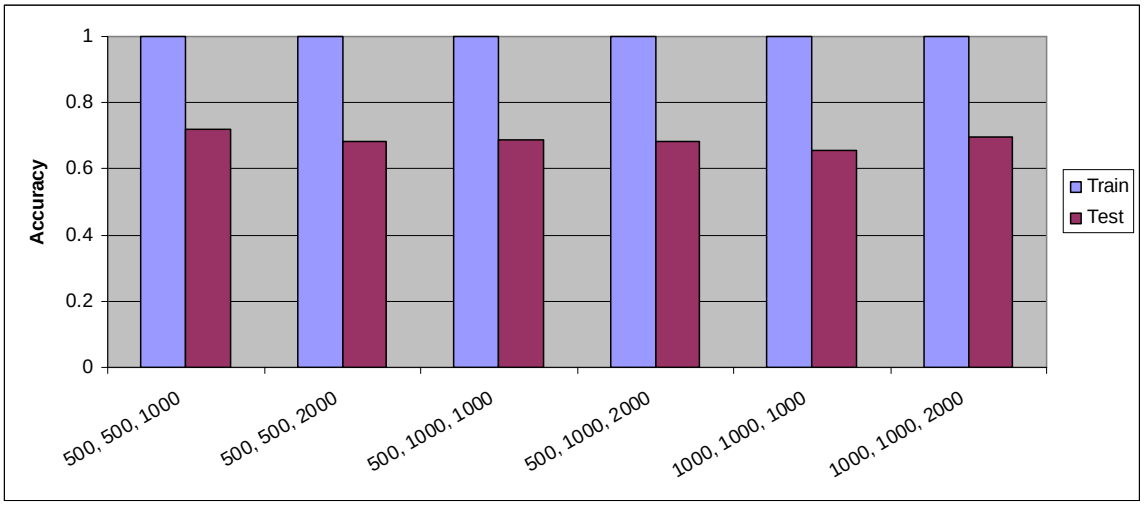

Figure 5. Accuracy of network configurations for 16 × 16 images.

Figure 6 shows the same network configurations as Figure 5, but uses images with a size of 32 × 32 pixels. Overall the images of size 32 × 32 underperformed when compared to images of size 16 × 16. It is interesting to note that this time the network configurations with more hidden units performed better than networks with fewer hidden units. It seems reasonable to believe that performance could be further increased by adding more hidden units, but due to computational constraints with the number of network nodes, I was not able to test this hypothesis.

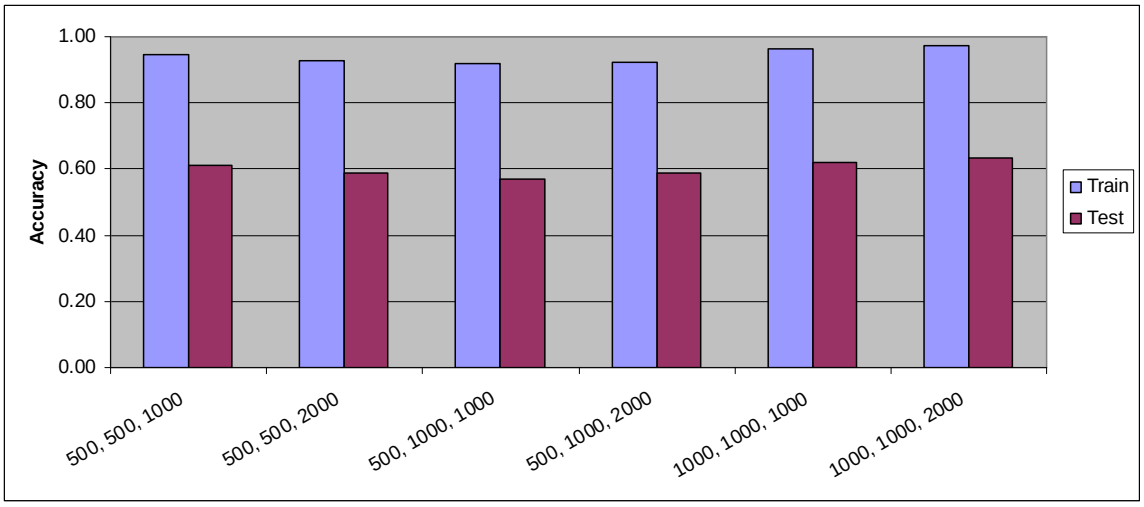

Figure 6. Accuracy of network configurations for 32 × 32 images.

Next, I compared the performance between the deep belief network and the well-known k-nearest neighbors algorithm. To make this comparison, the 10 nearest neighbors for each image were found and the accuracy of those images belonging to the same class as the initial image was calculated. For images of size 16 × 16 the DBN achieved 72% accuracy whereas the nearest neighbors algorithm achieved only 44% accuracy. The deep belief network improves drastically over the performance of k-nearest neighbors algorithm.

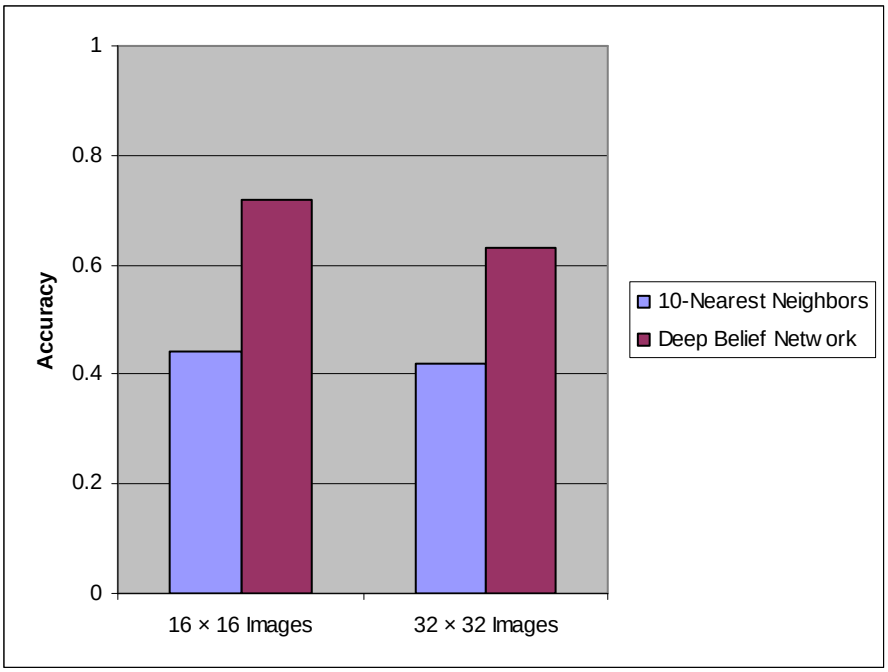

Figure 7. Comparison between the accuracy of 10-nearest neighbors and the best performing deep belief network.

It is clear that the 500, 500, 1000 network using 16 × 16 images has the best performance within the computational limits. Using this network, I executed search queries by comparing the distance between the output vector of the query image and the output vector associated with every other image. Figure 8 shows six search queries and the top three matches for each query. The first three queries are successful examples and the last three queries are unsuccessful examples. The percentage under each image shows how similar the result was to the query using the output vector produced by the network. In general, images of natural environments were harder to match than images that were computer generated.

Figure 8. Top three results for each query. The first three queries are successful examples. The last three queries are unsuccessful examples. When the query fails, the resulting images are likely to have a similar structure to the query image.

Lastly, I compared the performance between the deep belief network and other published results for object recognition on the Caltech 101 dataset. To make this comparison, I used the best performing deep belief network on the entire Caltech 101 dataset. Figure 9 shows the mean rate of recognition per class is 46% for the deep belief network.

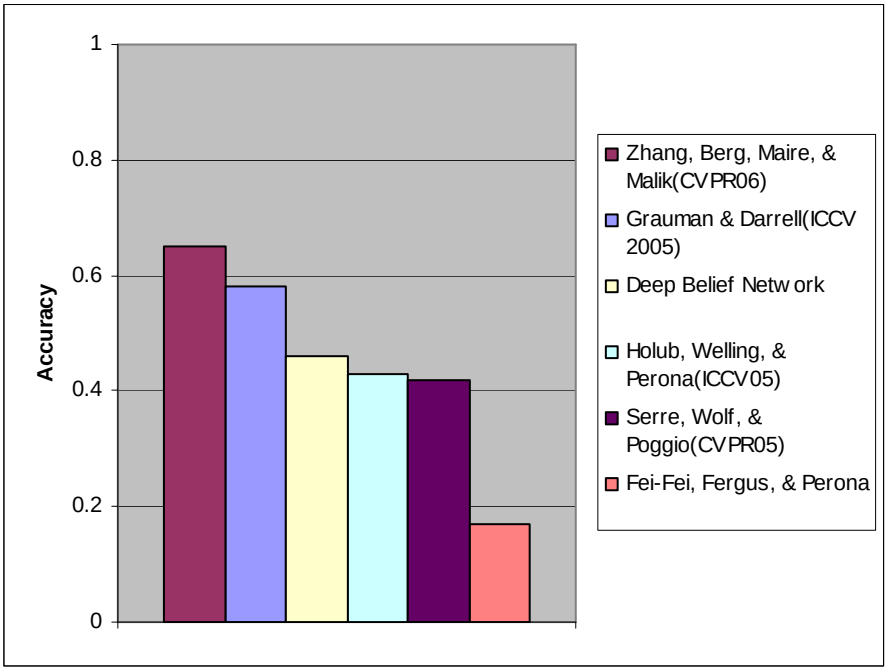

Figure 9. Comparison between the accuracy of the best performing deep belief network and other published results for object recognition on the Caltech 101 dataset.

The deep belief network falls behind the best performing recognition algorithm by Zhang *et al.* [16], which uses a hybrid of KNN and SVM classifiers and achieves a 65% mean rate of recognition per class. Also, Grauman and Darrell [3] achieve better performance using local image features that are matched to a pyramid of histograms. The resulting score forms a kernel which is then used in an SVM. The method proposed by Zhang *et*

al. uses a deformation model to capture information around a control point, and Grauman and Darrell similarly capture information about the object region by using sets of two points. These algorithms are tuned towards object recognition whereas the deep belief network approach is tuned more towards the general composition of the image.

It is more difficult to compare the performance of the deep belief network in the field of image retrieval because a standard benchmarking system has not been defined. Figure 10 shows a comparison of precision-recall curves for two different image categories. Wang *et al.* used a SVM as a classifier for their image retrieval system. The results are similar, but the deep belief network underperformed while the recall was low, but then performed better as the recall reached 50%. In terms of content-based image retrieval, the deep belief network performed well, and it would provide a good alternative to the standard method of searching for images by keyword.

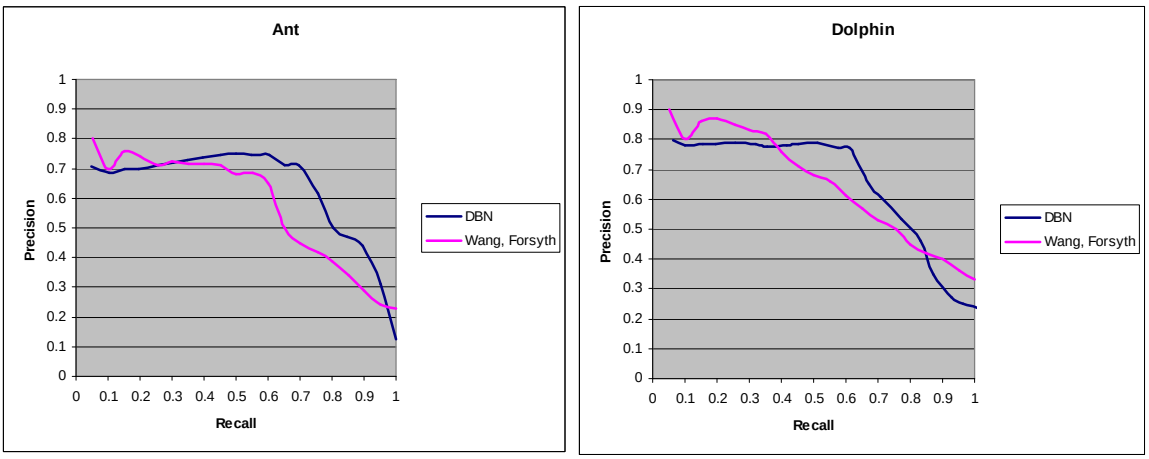

Figure 10. Precision-recall curves for images in the ant and dolphin classes.

5. Conclusions and Future Work

It has been shown that deep belief networks can be used as an alternative to labeling images with keywords in order to provide effective image retrieval. After the network has been trained on a sample set of possible images, new images can be automatically labeled and associated with other similar images using only the content within the new image (the pixel data). This technique is more efficient than having human subjects assign labels to the countless number of new images created everyday. Not only can new images be labeled, but labeling could be skipped completely. The system could simply be used to group similar images together. An image of a car would not have to be labeled as a car. The system would be trained such that an image of a car would point to a group containing images of cars.

The major shortcoming of this system is that it will perform poorly on images where the object is not the focus of the image. Too much background noise, occlusion, and translational and rotational invariance will affect performance. These problems could be relieved by performing segmentation and aligning the object before it is processed by the deep belief network.

Although, it is important to keep in mind that these problems may affect object recognition, but the system will still return images with similar visual features. The output vector for each image will represent the gist of the scene [11] which may be enough to match user's intended query. If the query object appears in a grassy field with

a small amount of sky, the results are likely to contain similar images of this natural scene.

There are other techniques worth trying to help cope with variances on the objects. Initially, I tried using SIFT descriptors instead of the Haar wavelet transform. SIFT has the advantage of being invariant to small amounts of scale and orientation distortions along with partial occlusion of the object [9]. Although, for this case, the SIFT descriptors varied too much from one image of an object to another and did not work well with the deep belief network.

Another way to achieve translation invariance is to use a bag of words approach. Feature descriptors could be calculated at salient points within each region of the image. The, a histogram of the Haar wavelet responses at each of these points could be used as input to the deep belief network. Using this approach, the location and orientation of the object would be less of a concern, and the system could be more effective on a wider variety of images.

References

[1] T. Deselaers, S. Hasan, O. Bender, H. Ney, “A Deep Learning Approach to Machine Transliteration”, Proceedings of the Fourth Workshop on Statistical Machine Translation, pages 233-241, 2009.

[2] L. Fei-Fei, R. Fergus, P. Perona, “Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories”, CVPR 2004, Workshop on Generative-Model Based Vision, 2004.

[3] K. Grauman, T. Darrell, “The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features”, International Conference on Computer Vision, 2005.

[4] G. E. Hinton, S. Osindero, Y.Teh, “A fast learning algorithm for deep belief nets”, Science, 2006.

[5] A. D. Holub, M. Welling, P. Perona, “Combining Generative Models and Fisher Kernels for Object Class Recognition”, International Conference on Computer Vision, 2005.

[6] C. E. Jacobs, A. Finkelstein, D.H. Salesin, “Fast Multiresolution Image Querying”, Proceedings of the 22nd annual conference on Computer Graphics and Interactive Techniques, pages 277-286, 1995.

[7] F. Jurie, B. Triggs, “Creating Efficient Codebooks for Visual Recognition”, International Conference on Computer Vision, 2005.

[8] H. Lee, R. Grosse, R. Ranganath, A. Y. Ng, “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations”,

Proceedings of the 26th Annual International Conference on Machine Learning, pages 609-616, 2009.

[9] D. G. Lowe, “Object Recognition from Local Scale-Invariant Features”, Seventh International Conference on Computer Vision, 1999.

[10] C. Mulcahy, “Image compression using the Haar wavelet transform”, Spelman Science and Math Journal, 1996.

[11] A. Oliva, A. Torralba, “Building the gist of a scene: the role of global image features in recognition”, Progress in Brain Research, 2006.

[12] M. Ranzato, Y. Boureau, Y. LeCun, “Sparse Feature Learning for Deep Belief Networks”, Advances in Neural Information Processing Systems 20, 2008.

[13] R. Salakhutdinov, G. E. Hinton, “Deep Belief Networks”, 2007.

[14] R. Salakhutdinov, G. E. Hinton, “Deep Boltzmann Machines”, Proceedings of the International Conference on Artificial Intelligence and Statistics, pages 448-455, 2009.

[15] T. Serre, L. Wolf, T. Poggio, “Object Recognition with Features Inspired by Visual Cortex”, Conference on Computer Vision and Pattern Recognition, 2005.

[16] G. Wang, D. Forsyth, “Object Image Retrieval by Exploiting Online Knowledge Resources”, International Conference on Computer Vision, 2008.

[17] H. Zhang, A. Berg, M. Maire, J. Malik, “SVM-KNN: Discriminative Nearest Neighbor Classififcation for Visual Category Recognition”, International Conference on Computer Vision, 2006.