

## Structural Image Search

Katie Kuksenok, Michael Brooks  
CSCI 364, Spring 2009

### Introduction:

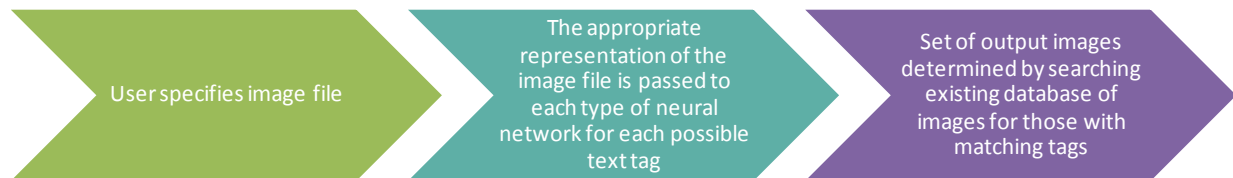
People have been accumulating more and more digital image data – photos of friends and family, cute kittens, and famous people, or just really, *really* hilarious clip art – both on their hard drives and, inevitable, within the vast expanses of the internet. But why did they do this? Why did that particular user save *another* picture of a baby in a funny hat? The least disturbing answer is that she wanted to see that picture again.

Here is where both local software and massive internet applications seem to be at a loss. Try as she might, the user cannot search for pictures of a baby or a kitten; she must search for the *word*. Yet if a picture is worth a thousand words, why did we ever think a single “tag” would be enough to specify one in a humongous, growing database?

Thus, we propose a method that would allow a comprehensive image-based search that relied on the image data. However, often, we *want* to search for images with words. Our proposed and prototyped system accommodates this human desire for language, and it attempts to use neural network learning and diverse image representation methods to allow a new kind of image search.

### Problem Definition and Algorithm

We constructed a system for implementing text/image structural search. The program reasons about images using textual labels, but takes as input an image and returns as output a set of images.



Our algorithm has several elements to it. First, we chose an image representation. We have implemented four different types of representations, all of which are too simple to be especially reliable. Second, we chose a lexicon of textual tags that the program could use to describe images. The dataset of images, which was pre-labelled, came with a pre-specified set of tags.

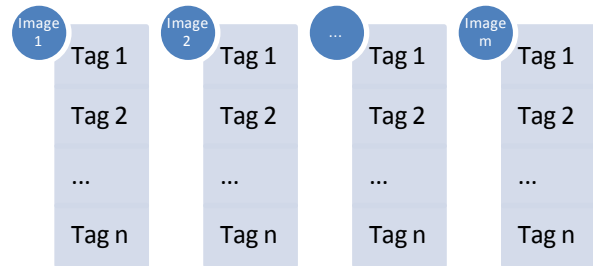
### System Design

We are using PHP to manage MySQL databases with both test/training data and neural network architecture data, and to run the Fast Artificial Neural Network (FANN) library (Nissen). The system design encompasses several layers of both processing and data storage. The design described here

reflects the framework of our program, but, in specified cases, not necessarily the current implementation. See *Appendix: System Design Components*.

### Data Storage

We use two libraries, one of images and one of neural networks, generated and formatted in FANN. A third database system introduces redundancy, but also efficiency, as it allows us to use PHP and MySQL to manipulate the data from both image and neural network libraries without doing expensive file operations.



An image in the library is associated with a text file listing  $n$  tags that are good descriptors of the image, which will aid in determining similarity. Since this system leaves different levels of users to make judgements about what makes a “good” tag or a “similar” image, we will assume that what we are given are, indeed, good images with good tags and a good notion of similarity.

### Neural Network Processing

The general approach is to use a neural network to learn what each particular tag in the database means. This is accomplished by training individual to each tag networks to output whether or not a tag applies. The neural networks, then, take as input one image and output a binary variable. We use the Cascade Correlation Training Algorithm in order to avoid topological limitations on the neural networks produced. Additionally, its batch approach to training addresses, at least partially, the limitations on efficiency posed by the iterative approach of iterative back-propagation training (Fahlman), (FANN Cascade Training).

We train four neural networks for each of 42 tags. Those tags were selected from the 29842 in the database because the number of images with those tags was greater than 5000, assuring an (arguably) non-negligible representation of each tag. A total of 168 networks were generated.

### Image Processing

The question arises, however, of how exactly one conveys image data to a neural network. Different approaches lead to different results. If tags are colors or combinations of colors, for example, we would want the neural network to be sensitive to colors. If tags are purely shape, then we would want the neural network to ignore color data completely. But in our generalized vision, we have no idea what the tags are! Thus, we must try to accommodate various kinds of image representation methods in order to avoid limitations on search functionality. In these beginning steps of this program, we implement a few very simple algorithms. To see how they fare on some simple examples, see *Appendix: Image Representation*.

1. BrightnessByAxis.30.30.x  
Reduce the image to a 30 by 30; Find average brightness for every column; Return size 30 array

2. BrightnessByAxis.30.30.y  
Reduce the image to a 30 by 30; Find average brightness for every row; Return size 30 array
3. BrightnessFullHist.30  
For every pixel, put it one of 30 bins based on brightness; Return size 30 array of counts
4. HueFullHist.30  
For every pixel, put it one of 30 bins based on hue; Return size 30 array of counts

### *User Involvement and Interface*

There are several levels of user involvement, which we elaborate on, starting with the lowest. Ideally, a user that can act on all the levels will be able to manipulate the system to do a vast amount of things. If used for more than one person, the system would be optimally suited for a hierarchical user structure of administrators, who can manipulate really low-level elements, moderators who can perform functions such as editing, or expanding the image library, and end-users, who would only be able to perform search functions. Placing as much power as possible into the hands of the user without burdening them, we believe, will address the fundamental problems faced by other major image searches (*see Related Work section for discussion*) (Bailey, 2009).

### *Redefining Image Metrics Relevant to Similarity*

The most low-level way for a user to manipulate the system is to provide alternatives to image processing methods. As this is a very important, and powerful, aspect of the system, we do not want to limit the user's capabilities. The system structure, which is modular and versatile, allows the user to add processing methods.

### *Redefining Image Similarity*

Having created a metric for measuring similarity, a user is able to manipulate the weighting a particular representation. This is a general format of specifying similarity that surpasses "similarity is similarity in color," or "similarity is similarity in the stuff that is happening in the lower left corner of the image," by allowing the user to decide on their own that similarity is, in fact, 20% color, 30% stuff in the lower left corner, and 50% presence of turtles facing to the left.

### *Reaping the Fruits: Search*

The user is able to, currently, populate a "library" folder with images, then search with those images as input. The user can also search by tags, but this does not involve any learning, simply looking up appropriate entries in the image database. Then, the user can see which tags the algorithm has identified for the image and choose the appropriate ones. This is mainly for testing and example purposes of this program. Then, the user can look through a result set of up to 5000 images, sorted by closeness to the identified set of tags. All result set tables are saved to later pleasure.

## **Experimental Evaluation**

### **Methodology**

For our test data, we used a set of 100,000 tagged images (Ahn) generated by Luis von Ahn's ESP Game. This game has served as the basis for the Google Image Labeler, which is used in the tagging process central to Google Image Search (Arrington, 2006).

Before any kind of qualitative testing could take place, we calculated the accuracy of each tag's and type's Neural Network, versus the accuracy of a random number generator (where probability of 1 is proportional to how many images there are in the database with the tag in question). As is clear from the data plotted in the *Appendix: Neural Network Learning, Figures 3a-d*, the accuracy of the neural networks was no worse than, but also not much better than that of the random number generator, despite what seemed to be successful training on representative data.

We also tested the performance of the program by recording how much time, in seconds, it took to assign tags to arbitrary images. On a sample of five images, it took an average of 62 seconds per image to assign tags, with a minimum of 35 and a maximum of 90 seconds. It should be noted that, due to the image processing involved, the processing time depends on the image size. In the first two representations, the images must be resized before processing, and in the second two (histogram) representations are directly connected to the processing.

### **Results & Discussion**

Overall, our results show that this kind of image representation (closely tied to the pixels of the image) does not produce particularly effective neural networks. One would do about as well by just guessing based on the occurrence rate in our training database. Still, it is very possible that by choosing different types of image representations, neural networks could perform well at this task.

### **Related Work**

On April 20, 2009, Google released a new feature in searching images called "Similar Images," which allows the user to refine their search based on "visual similarity" (Google, 2009). The tool, however, still seems tied to the textual keywords assigned to each image, using visual similarity to narrow preexisting text-information-based searches: "If an image is uploaded with a different title or no keywords at all, then it won't show up as a similar image in Google". Additionally, the visual similarity appears to be "very broad," and focused "mostly [on] the color of the images, not the actual content" (Bailey, 2009). The searching technique seems similar to that used by Like.com to search within a catalog of products (Seigler, 2009), so it is possible that the vastly broader scope of various kinds of images in Google is not easily addressed by an approach successful in something as specific as catalog search.

A "reverse image search engine," TinEye, searches for copies or slightly modified versions of images (TinEye). Its matching algorithm relies on generating a visual "fingerprint" for images, and searching within those. The tool allows a user to upload an image to get a set of similar images, and has divorced itself entirely from the preoccupation with textual tags, which haunts Google Similar Images. It is much more useful for artists, who are interested in using much more specific metrics for image comparison than those utilized by Google Similar Images (Bailey, 2009).

This brief survey suggests that there is no middle-ground tool, one that uses textual tagging information, but it not harnessed by it. Additionally, tools such as TinEye appeal primarily to very specific niche audiences, such as artists, due to their narrow purpose of finding very, very similar images. Before the vast improvement of Google image search, there was research into using text and image properties

simultaneously (Frankel, Swain, & Athitsos, 1996). However, it was limited greatly by a focus on color-based heuristics (in addition to a few other arbitrary properties of images) and text surrounding images.

## **Future Work**

Our primary goal was to explore the success of particular image representations in providing a basis in image-based search. Future work is vast, but two general directions can be discerned at this point; specifically, improvements in performance and in image awareness.

### *Performance*

The actual search requires a great deal of computation, and, even if we achieve success in search accuracy and conceptual validity, we do not expect the program to perform well. Thus, future optimization work would have to be done, which would involve, perhaps, reducing the amount of computation by showing equivalent time-saving heuristics.

### *Image Awareness*

Another possibility for future work is to develop more image representations. As we have seen from our testing, raster-based approaches leave much to be desired. Thus, using foreground/background recognition, object detection, and other methods for which relatively successful algorithms have been developed, we may substantially increase the effectiveness of our general approach.

## Works Cited

- Ahn, L. v. (n.d.). *Useful Resources*. Retrieved May 8, 2009, from Luis von Ahn's Home Page: <http://www.cs.cmu.edu/~biglou/resources/>
- Arrington, M. (2006, September 1). *Google Image Labeler*. Retrieved May 8, 2009, from TechCrunch: <http://www.techcrunch.com/2006/09/01/google-image-labeler/>
- Bailey, J. (2009, April 24). *Google Similar Images: Poor Copy Detection*. Retrieved May 8, 2009, from Plagiarism Today: <http://www.plagiarismtoday.com/2009/04/24/google-similar-images-poor-copy-detection/>
- Fahlman, S. E. (n.d.). *Scott E. Fahlman*. Retrieved May 10, 2009, from <http://www.cs.cmu.edu/~sef/sefPubs.htm>
- FANN Cascade Training*. (n.d.). Retrieved May 10, 2009, from Fast Artificial Neural Network Library: [http://leenissen.dk/fann/html/files/fann\\_cascade-h.html](http://leenissen.dk/fann/html/files/fann_cascade-h.html)
- Frankel, C., Swain, M. J., & Athitsos, V. (1996). *WebSeer: An Image Search Engine for the World Wide Web*. The University of Chicago Computer Science Department.
- Google. (2009, April 20). *Hard at Play at Google Labs with Similar Images and Google News Timeline*. Retrieved May 8, 2009, from Official Google Blog: <http://googleblog.blogspot.com/2009/04/hard-at-play-in-google-labs-with.html>
- Nissen, S. e. (n.d.). *Fast Artificial Neural Network Library*. Retrieved from <http://leenissen.dk/fann/>
- Seigler, M. (2009, April 20). *Google Similar Images First Look*. Retrieved May 8, 2009, from Tech Crunch: <http://www.techcrunch.com/2009/04/20/google-similar-images-first-look/>
- TinEye. (n.d.). *FAQ*. Retrieved May 8, 2009, from TinEye: <http://tineye.com/faq>

## Appendix

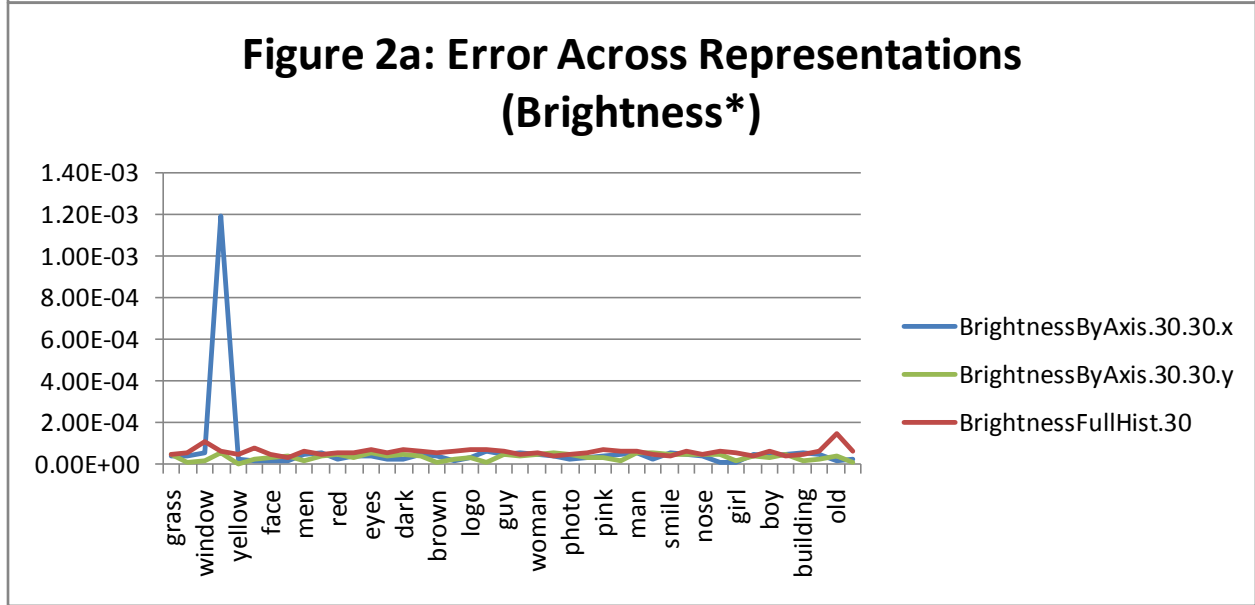
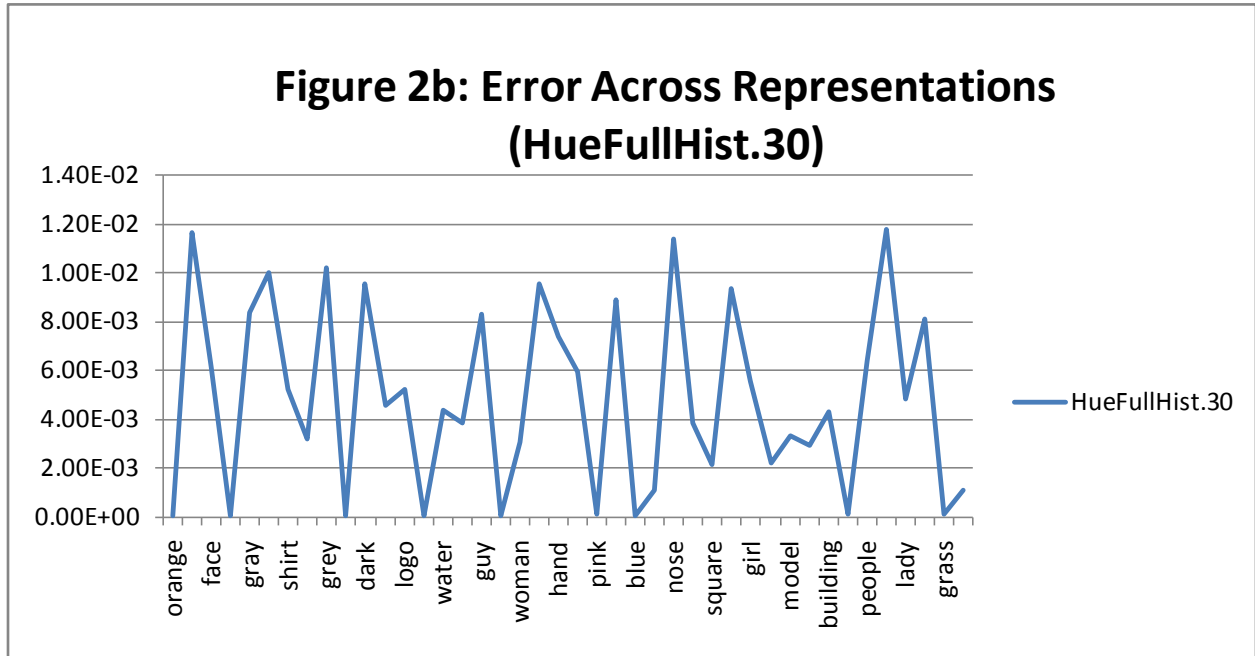
### Image Representations

You See These  
Pictures ...

**...while the Neural Networks See These Graphs**

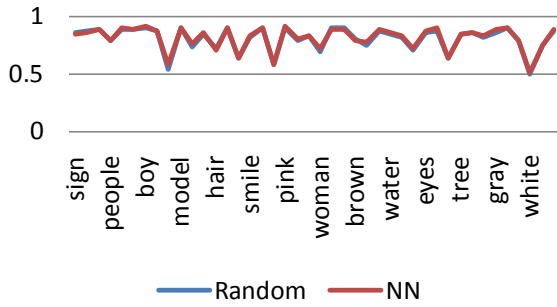


Neural Networks



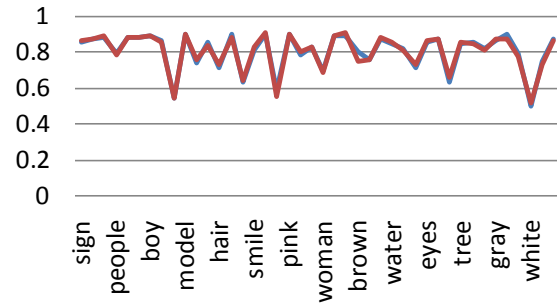
**Figure 3a: Accuracy on BrightnessByAxis.x.30.**

**30**

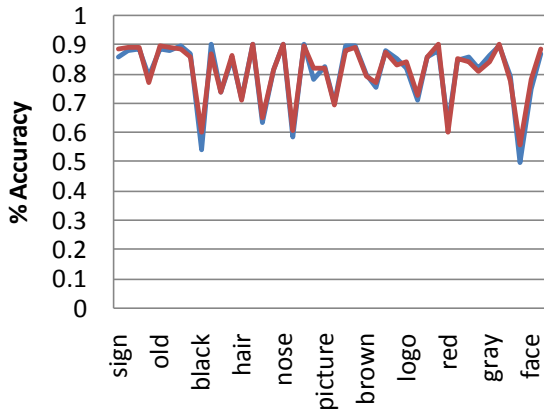


**Figure 3b: Accuracy on BrightnessByAxis.y.30.**

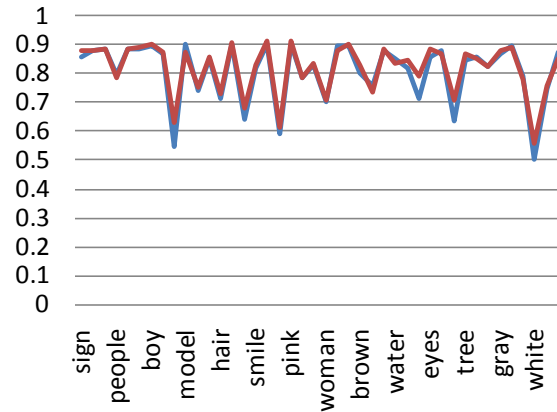
**30**



**Figure 3c: Accuracy on BrightnessFullHist.30**



**Figure 3d: Accuracy on HueFullHist.30**



## Website Use

[cs364final.dyndns.org/ai\\_final](http://cs364final.dyndns.org/ai_final)



Persistent components: can always click on logo to return to this front page, ran always chose a recent run to click on

First: enter tags, eg “hat, ball” into the top-right gray input box OR click on a radio button<sup>1</sup>. Note: in order to successfully select an image form the pre-existing sample library (about while there is nothing special), the textbox for tags must be empty. Click “Submit” at the bottom. If you’ve entered the text, skip one step in this explanation.

Second: this lets you see what the neural networks, jointly, came up with. You can use the probabilities window to see numerical properties of what will result from the search. There are further details on this page. It is called the probability feature because it allows you to say, oh, this result set is small. It’s very improbable it will be of any use. Additionally, it (kind of) justifies the sheep.

Third: peruse the result set at your leisure using the radio buttons to go +1 page, -1 page, first page, last page, or any page you specify where the “x” is (you must click Submit to enact the change). If you had multiple tags, the little squares beneath the pictures indicate which tags are (orange) and are not (blue) used to describe that picture. The results are sorted, with best-match at the top; they are limited to 5000 results. You can see the tags, in proper order, that yielded the result at the top, along with milliseconds taken to do the search, and the “cue” – the image that you originally specified, if any. Additionally, you can access that search anytime using the link provided at the top, or if it was one of the 100 most recent, the list to the right.

---

<sup>1</sup> Search for “idiot” when you get the chance.

## System Design Components

