Filtering Image Spam with Near-Duplicate Detection

Zhe Wang, William Josephson, Qin Lv, Moses Charikar, Kai Li
Computer Science Department, Princeton University
35 Olden Street, Princeton, NJ 08540 USA
{*zhewang,wkj,qlv,moses,li*}*@cs.princeton.edu*

Abstract

A new trend in email spam is the emergence of image spam. Although current anti-spam technologies are quite successful in filtering text-based spam emails, the new image spams are substantially more difficult to detect, as they employ a variety of image creation and randomization algorithms. Spam image creation algorithms are designed to defeat well-known vision algorithms such as optical character recognition (OCR) algorithms whereas randomization techniques ensure the uniqueness of each image. We observe that image spam is often sent in batches that consist of visually similar images that differ only due to the application of randomization algorithms. Based on this observation, we propose an image spam detection system that uses near-duplicate detection to detect spam images. We rely on traditional anti-spam methods to detect a subset of spam images and then use multiple image spam filters to detect all the spam images that "look" like the spam caught by traditional methods. We have implemented a prototype system to achieve high detection rate while having a less than 0.001% false positive rate.

1. Introduction

Spam message volumes have doubled over the past year and now account for about 80% of the total messages on the Internet. A major reason for the increased prevalence of spam is the recent emergence of image spam (*i.e.* spam embedded in images). Image spam volumes nearly quadrupled in 2006, increasing from 10% to 35% of the overall volume of spam; worse, the volume of image spam continues to rise[3, 21]. The situation has significantly frustrated end-users as many image spam messages are able to defeat the commonly deployed anti-spam systems. In order to reduce the impact of spam, it is crucial to understand how to effectively and efficiently filter out image spam messages.

Spammers have recently begun developing image-based spam methods to circumvent current anti-spam technologies since existing anti-spam methods have proved quite successful at filtering text-based spam email messages. Early image-based spam simply embedded advertising text in images that linked to HTML formatted email so that its content could be automatically displayed to end-users while being shielded from text-based spam filters. As spam filters started using simple methods such as comparing the hashes of image data and performing optical character recognition (OCR) on images, spammers have quickly adapted their techniques. To combat computer vision techniques such as OCR, spammers have begun applying CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) techniques. These techniques distort the original image or add colorful or noisy backgrounds so that only humans can identify the intended message [2]. Once spammers have applied an image creation algorithm to make a message difficult to detect with computer vision algorithms, they apply further randomization to construct a batch of images for delivery. The additional randomization defeats hash-based detection mechanisms. The result is that current image spam methods present serious challenges for anti-spam systems.

Although some research has been done to distinguish spam images from non-spam images by using computer vision techniques including filtering noisy images for recognizing embedded text or monitoring color saturations in an image[1], such methods tend to have high false positive rates, labeling ham (non-spam) as spam. This is because computer vision techniques have not been able to defeat CAPTCHA. Furthermore, it is difficult to predict what spam images will look like as they are constantly evolving to evade detection. In addition, sophisticated computer vision techniques often require substantial CPU resources, making them less practical in high-volume environments.

We believe that an effective image spam detection system should satisfy several requirements. First, it should be *accurate*, detecting most image spams while maintaining a low false positive rate. Second, it should be *efficient*, parsing incoming emails with images at modern WAN speeds. Third, it should be *extensible*, allowing new image spam filtering methods to be added to deal with quickly evolving image spam techniques.

This paper proposes an image spam detection system to satisfy the requirements above. The basic idea is to use traditional anti-spam methods to detect some image-based spam messages and then use fast near-duplicate detection filters to detect the variations of known spam images. The system we propose is based on two observations. The first is that traditional spam detection methods such as honeypots, message header analysis or human reporting mechanisms can detect some image spam messages. The second is that image spam messages are typically sent in large

batches where the messages in each batch are visually similar, although the variations can be sophisticated. For example, spammers often design a template image and apply various randomized alterations or noise to the template before sending it out to each end user. Figure 3 and 4 show some spam image samples. We believe that this is because spammers still want to deliver clear information to end users and they want to use efficient methods to generate millions of unique spam images while not obscuring the template image too much.

Rather than studying the image itself to determine whether the particular image is a spam image or not, our system adopts an alternative approach. We use very efficient near-duplicate detection techniques to find spam images that are variations of other spam images caught by traditional anti-spam methods. Thus our system is complementary to the existing anti-spam system to help detect image spams missed by the traditional system.

We have designed and implemented a prototype of the proposed image spam detection system. The system supports the use of multiple image spam filters, allows users to plug in new filters and to specify different aggregation methods among the filters including AND (all filters agree), OR (one of the filters decides) and VOTE (certain number of filters agree). We have implemented three image spam filters using different near-duplicate detection techniques in our prototype system. Our experiments with a suite of image spam benchmark and 10 million non-spam images show that using VOTE method can effectively detect variations of most kinds of image spam messages while maintaining the false positive rate to less than 0.001%.

2. Previous work

Nowadays, spam filters are widely deployed and various anti-spam techniques have been developed. At the network layer, systems such as Mail Avenger [12] track source, destination, network path, and software version information for analysis by spam filters. Many anti-spam systems also use a combination of whitelists, blacklists, and so-called greylists that force legitimate clients to re-send messages since spammers often do not bother doing so [10]. Other common techniques include block lists distributed via DNS that identify addresses assigned to dialup users or known open relays and challenge-response systems that automatically build whitelists. Most systems such as Mail Avenger, Spam Assassin, and SpamGuru [12, 20, 18] use multiple techniques, including multiple classifiers, to identify spam. Filters for text-based spam, including plain text and HTML e-mail, have employed a variety of statistical techniques, particularly Bayesian inference [17, 7]; these statistical filters appear to classify text-based e-mail well. Another popular approach is the use of so-called "fuzzy signatures" such as those employed by Vipul's Razor [23] and the Distributed Checksum Clearinghouse [4]. Fuzzy signatures are designed in such a way that the signature for substantially different messages are unlikely to collide but that the signature for very similar, although not necessarily identical, messages will collide with high probability. Systems such as DCC allow users at different sites to share fuzzy signatures for reported spam.

Although image-based spam has been around for some time, recent reports and anecdotal evidence suggest that there has been significant growth both in the volume of image-based spam and in the percentage of spam that uses image attachments to convey its message [14, 21, 8]. Unlike earlier image-based spam, current image spam is randomized to avoid signature based anti-spam techniques. Since the majority of spam is now delivered through botnets [15], spammers have the bandwidth and computational resources necessary to customize individual spam images extensively.

Although traditional spam filters that rely on analysis of sender, message header and various other information can detect some of the image spams without looking into the image itself, other image spams still pass through the spam filter since spammers try very hard to make everything except the spam image itself look innocent. Some recent research studied the image classification using computer vision techniques. One approach[1] is largely based on the extraction of text regions in the images of interest and SVM. This method can achieve about 85% detection rate with about 15% false positive rate. Wu *et al.* [24] have identified a number of useful visual features including banner images, computer generated graphics and embedded-text, and use SVM to train the classifier. Their approach can achieve about 81% detection rate with about 1% false positive rate. Although the results are encouraging, we believe that the misclassification rate of non-spam images needs to be dramatically lower in order to make image spam detection practical.

Our image spam detection system utilizes content-based image similarity search techniques. Much work has been done in this area [5, 16, 19, 22]. Recently, researchers have used content-based image search techniques for near-duplicate image detection [9, 25]. The image features they use are usually complex and slow to compute and compare. Given the large number of image spams that are recieved every day, our challenge is to achieve high effectiveness and high efficiency at the same time.

3. Main Idea

Our image spam detection system takes advantage of the nature of spam messages: they have to be sent in large quantity, and the machine generated spam images within the same batch will look similar to each other. Since spam messages are sent in large quantity, some of them can be detected using traditional spam detection methods such as honeypots (dummy email accounts set up to attract spams) or sender analysis, as well as those reported by end users. Once we have identified some of the spam images via traditional spam detection methods, we can then detect their near-duplicates as spams even if those emails evade traditional spam filters.

Figure 1 shows the architecture of our image spam detection system: the traditional spam filters not based on image content analysis; a set of image spam filters that detect near-duplicate images; and an evaluator to aggregate the results from multiple image spam filters. Each image spam filter is made up by its feature extraction unit, non-spam feature DB, spam feature DB and spam image detection

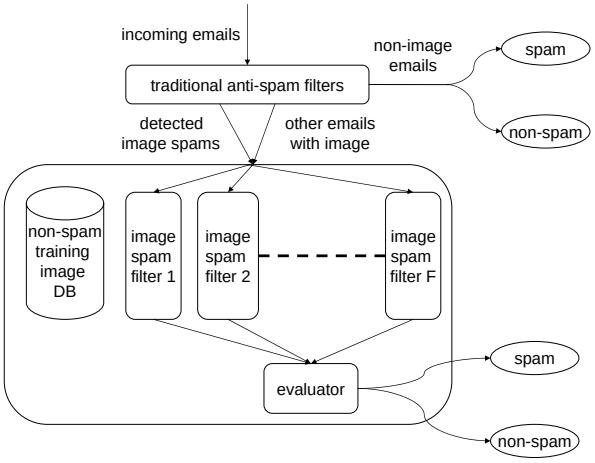

Figure 1: Image spam detection system architecture.

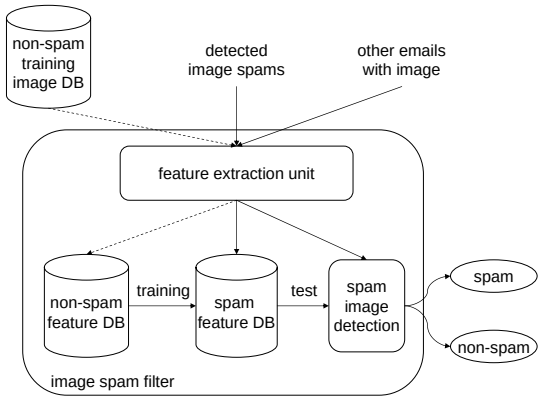

Figure 2: An image spam filter.

unit as shown in Figure 2.

An important issue in designing a near-duplicate detection system is to figure out how "near" when comparing two images with each other. In other words, we need to figure out a distance threshold in the filter's feature vector space of the images. If a threshold is too small, the filter tend to achieve low false positive rates, but low detection rates. Traditionally, threshold will be derived by training with a spam image dataset. We took a different approach, deriving thresholds by training with a non-spam image dataset. The idea is to learn what the "non-spam" images look like and use the information to bound the threshold to achieve high detection rates and low false positive rates. The intuition behind of this approach is that the look of non-spam images are relatively stable over time, whereas image spam methods are constantly evolving and new spams may have different looks. We collect a large collection of known non-spam images and save it in the non-spam training image DB. Each deployed image spam filter (or a new image spam filter plug-in) will extract features from all the non-spam images offline and store them in the non-spam feature DB for future use.

When deployed in real time, the image spam detection system will work together with the traditional anti-spam filters. All incoming emails will go through the traditional anti-spam filters first. The emails without any embedded images will be handled by traditional filter alone. The emails with embedded images will be filtered and labeled first by the traditional spam filter (such as analyzing their headers) and then passed to our image spam detection system.

All embedded images will be processed by all image spam filters. The feature extraction unit of each individual image spam filter will extract a feature vector from the image. If the image is labeled as spam by the traditional anti-spam filter, the feature vector will be inserted into the spam feature DB. During the insertion, we will calculate its distance from all the feature vectors stored in the non-spam feature DB and set the smallest distance as the detection threshold associated with this particular spam image. This step will essentially create a high dimensional sphere in the feature space such that any other feature vector that falls into the sphere will be considered as a spam.

An image spam filter then uses the spheres defined by the thresholds in the spam feature DB to detect near-duplicates of the known spams. Conceptually, the image spam filter will compare the feature vector of an image (not labeled by the traditional spam filter) with all the feature vectors in the spam feature DB. If the feature vector falls in any of the "sphere" of a known image spam, the email associated with this image will be considered as a spam by this specific image spam filter.

We allow multiple image spam filters to work together to increase the coverage of spam categories, improve the detection rate for each category while maintaining low false positive rate. Our idea is to use an evaluator to aggregate the results from multiple filters. We currently consider three methods:

- **"AND"**: an image is classified as spam if and only if all filters decide it is spam. This method will lead to relatively low detection rates and very low false positive rates.
- **"OR"**: an image is classified as spam if any filter decides it is spam. This method will lead to high detection rates and relatively high false positive rates.
- **"VOTE"**: an image is classified as spam if a specified number of filters decide it is spam. This method will provide balanced detection rates and false positive rates between AND and OR.

By supporting multiple aggregation methods, our system provides users with more flexibility.

4. Implementation

To evaluate our idea, we have implemented a prototype image spam detection system. This section describes the implementation details of our system components.

We use Mail Avenger [12], a customizable SMTP server, together with SpamAssassin mail filter [20] as the traditional spam filter in our system. Each incoming message was annotated with the output available from Mail

Avenger. Mail Avenger was configured to deliver mail unconditionally but to log the TCP fingerprint and results of a traceroute back to the mail relay as well as the results of DNSBL lookups for about twenty different popular black lists. Once messages to the mail server were spooled, they were run through the SpamAssassin mail filter. The real-time feedback from Mail Avenger and SpamAssassin were used to determine how existing tools would classify incoming spam *at delivery time*.

We store 100,000 legitimate images in our non-spam image DB. They are sampled from two online photo sharing sites photo.net and pbase.com and from a COREL stock photo collection. The images are stored in the database so that new image spam filter can be trained when introduced into the system.

Image spam filters are designed to detect variations of image spams using different image near-duplicate detection techniques. Our system supports multiple image spam filters and allow users to plug in new filters for emerging image spam techniques. Figure 2 shows the components of an image spam filter.

We have leveraged Ferret toolkit [11] to construct image spam filters, manage spam feature DB, and perform near-duplicate detections. It is convenient to use since it uses advanced indexing techniques to perform high-speed similarity searches. To construct an image spam filter, all one needs is to plug in a feature extraction unit and the definition of its distance function.

The feature extraction unit converts an image to a feature vector representation for near-duplicate detection purpose. If two images are near-duplicates, their feature vectors would be very close to each other in the feature vector space. The following properties are highly desirable for the feature extraction unit:

- *Efficient* : The unit should be able to process incoming images very efficiently in order to match the throughput of targeted mail servers.
- *Effective* : Spammers typically add random "noises" to each spam image. For effective detection, the unit should produce features that are relatively insensitive to those added noises.
- *Distinctive* : To minimize false positive rate, the unit should generate features that can distinguish spam images from non-spam images.

We have constructed three filters (see Section 6).

The system has a spam feature DB for each filter. The DB stores all feature vectors extracted from all known spam images labeled by the traditional anti-spam filter, and an associated threshold value for each feature vector. The threshold value is the smallest distance between the feature vector of the spam image and the feature vectors of all the 100,000 non-spam images. If a new known spam image is within the threshold value distance away from an old known spam image, it will be treated as the spam from the same batch and no new entry will be inserted into the spam feature DB to save space. Older known spam feature could be retired if it had not been able to detect spams for a while.

We have implemented three aggregation methods AND, OR, and VOTE in the evaluator. See previous section for their aggregation functionalities.

(a) wave (b) animate (c) deform (d) rotate

Figure 3: Examples of spam construction techniques.

5. Known Image Spam Techniques

Spam images are typically generated in two steps. The first step is *template construction*, where a spam image template is constructed with the intended content for end user. The main goal is to use different methods such as CAPTCHA to defeat computer vision (such as OCR-based) anti-spam techniques. The second step is *randomization*, where a large number of spam images can be generated from the template image using various randomization techniques, in order to defeat signature-based anti-spam techniques. This section describes the template construction and randomization methods used in the image spam datasets we have collected.

Construction methods: among the spam images we have collected, we have identified four template construction methods:

- *wave* : See Figure 3 (a). This method uses wavy text to make it more difficult for OCR recognition.
- *animate* : See Figure 3 (b), the URL in the web browser is animated. By using animations in the GIF format, it is harder to detect the real spam content.
- *deform* : See Figure 3 (c). This method uses deformed text (such as irregular handwritten fonts, different font colors) in order to defeat OCR.
- *rotate* : See Figure 3 (d). This method rotates the text to a certain angle such that it is not horizontal and more difficult for OCR. Depending on the number of different angles can be used, this technique can also be used as a randomization technique (see below).

Randomization methods: The goal is to add random noises to a template spam image in order to generate a large number of spam images to defeat traditional signature-based anti-spam techniques. From the spam images we have collected, we have observed 17 randomization techniques:

- *shift* : template image is shifted horizontally or vertically on the canvas. See Figure 4 (1), (7).
- *crop* : template image is cropped differently (sometimes sacrificing part of content). See Figure 4 (2),

(3) and (13).

- *size* : slight variations of the size (height and width) of the template. For example, this can be achieved by writing the same template content on canvas of different sizes or resizing an image. See Figure 4 (4), (10).
- *dots* : adds random dots (or speckles). See Figure 4 (5), (11), and (13).
- *bar* : adds a randomized bar (of pixels with similar colors) to the top, middle, or end of a template image. See the blue bars in Figure 4 (4).
- *frame* : adds a frame (of randomized pixels and different thickness) to the template image. See the frames in Figure 4 (5).
- *fonttype* : uses different font types for the text. See Figure 4 (5), (7).
- *fontsize* : uses different font sizes for the text. See Figure 4 (11), (12).
- *fontcolor* : uses different font colors for the text. see Figure 4 (8). Further randomization is achieved by using different colors within each individual letter. See Figure 4 (7).
- *line* : uses randomized lines in the background. See Figure 4 (11).
- *linecolor* : uses randomized lines of different colors in the background. See Figure 4 (6), (7).
- *shape* : uses randomized shapes (such as polygons or ellipse) in the background. See the pink shapes in Figure 4 (3).
- *rotate* : rotates the text to a random angle. See Figure 4 (6).
- *bits* : uses a few randomized bits either in the metadata or at pixel level, resulting in different image files but no noticeable different images. See Figure 4 (14).
- *content* : uses different wording (but of similar theme) for each line in a multi-line message to achieve a combined high level of randomness. See Figure 4 (8).
- *fuzzy* : uses fuzzy text and lines. See the fuzzy text and discontinued lines in Figure 4 (10).
- *url* : uses different URLs (pointing to the same products). See Figure 4 (9), notice the only difference is the URL.

Figure 4 shows the samples of spam images, some of which use combinations of the methods above[1].

6. Image Spam Filters

Since spammers use different randomization methods to introduce noises to spam images, a single feature might not be able to effectively detect all variations. In our system, we have experimented with 3 filters based on color histogram, wavelet, and orientation histogram.

[1]We did not collect enough of the "slice-and-dice" image spams (where the spammer randomly slice the original spam image into several small pieces) mentioned in other report to form a batch.

6.1 Color Histogram Filter

The color histogram is a simple feature and can be calculated very efficiently by one simple pass of the whole image. We have used 64-dimensional color histogram based in the RGB color space. Values in each of the three color channels (R,G,B) are divided into 4 bins of equal size, resulting in $4 \times 4 \times 4 = 64$ bins in total. For each bin, the amount of color pixels that falls into that particular bin is counted. Finally it is normalized so that the sum equals to one. We use L_1 distance to calculate the distance between two color histogram features. For images represented by D-dimensional real-valued feature vectors, the $L1$ distance of the pair of points $X = (X_1, \ldots, X_D)$ and $Y = (Y_1, \ldots, Y_D)$ has the form:

$$d(X, Y) = \sum_{i=1}^{D} |X_i - Y_i|$$

We adopt color histogram in our system for its simplicity and efficiency.

The color histogram is effective against randomly added noises and simple translational shift of the images. For the spam randomization techniques described in section 5, the color histogram is designed to handle shift, size, dots, bar, frame, fonttype, fontsize, line, rotate, bits, content, fuzzy, url.

6.2 Haar Wavelet Filter

2-D Haar wavelet transformation is popular in image analysis and can be calculated efficiently in $O(n)$ time. We convert the color image into a 256×256 grayscale image and apply 2-D Haar wavelet transformation on it. We then take the first 4×4 wavelet coefficients at low resolution end of the matrix. This essentially provides the low resolution information about the original image. $L1$ distance measure is used to calculate the feature distance.

The Harr wavelet feature is mainly targeted for these randomization techniques: size, dots, bar, frame, fonttype, fontsize, line, shape, bits, content, fuzzy, url.

6.3 Orientation Histogram Feature

Orientation histogram feature provides the histogram of orientation of edges in the image. It was shown to be effective in hand gesture recognition in [6] and can be calculated by one simple pass of the image. We start by calculating the orientation of each pixel, then bin the orientation of each pixel into 36 groups, each of which is 10 degrees. After that, we use a 1 4 6 4 1 filter to blur the orientation histogram. The final histogram is normalized and $L1$ distance is used to calculate the feature distance.

Orientation histogram is designed to work better with these randomization techniques: shift, crop, size, fontsize, fontcolor, linecolor, bits, url.

7. Evaluation

We would like to answer the following questions:

- How effective is each image spam filter?
- How well do multiple image spam filters work together?
- What are the performance implications of these filters?

- How efficiently can we propagate spam image signatures for distributed spam detection?

To answer these questions, we have conducted experiments with our prototype system using a collection of spam images and non-spam images.

7.1 Evaluation Datasets

We use two different kinds of images in our evaluations: spam images and legitimate ("ham" or non-spam) images. Since many new kinds of image spams have emerged recently, the image spam techniques used a year ago are substantially different from the current ones. We have decided to create an image spam dataset using image spams collected during recent three months (Dec. 2006 to Feb. 2007) instead of using an old public spam corpus.

Our spam images are collected from seven different email accounts. These are personal email accounts including accounts from two popular online webmail service providers, one IT company account, and three education accounts. All images in the spam emails, including user identified ones, are collected. Duplicate spam images are removed by using SHA-1 hash; some malformed images are also removed. All the remaining images are manually classified into batches based on their content. We also extract the time stamp at the receiver for each image to verify the batches or further split the batches whenever necessary. We have removed batches that contain only a single image for our benchmark. The resulting spam image dataset contains 1071 images in 178 different batches. The min, max, average and standard deviation of the batch sizes are 2, 50, 6.02, 6.39 respectively. The full spam image benchmark is available online [13].

In order to evaluate false positive rates, we have constructed a non-spam image dataset. Since there are few publicly available non-spam email repositories (especially for emails containing images) due to privacy concerns, we have used samples of photos downloaded from popular photo sharing web sites as our non-spam images. We use two sets of non-spam images, one for training and one for testing:

- Training non-spam dataset: 100,000 image randomly selected from over 600,000 images downloaded from PBase and Photonet, and the COREL stock photo collection.
- Testing non-spam dataset: 10 million images downloaded randomly from the Flickr web site.

We believe they represent a good proportion of the kind of non-spam images sent via email.

7.2 Individual Image Spam Filter Results

We begin by evaluating the effectiveness of each individual image spam filter in isolation. To evaluate the effectiveness of a filter, we present the system with a single marked spam image, the remaining unmarked spam images, and 10 million non-spam images and see if our filter can detect the unmarked spam images in the same batch. Table 1 shows the false positive and detection rates for each spam category using different image spam filters. For each filter, the first group (shown in bold) contains the categories that the particular filter is designed to handle while the second group shows the remaining categories. The last row shows the overall false positive and detection rate for all the categories.

The results show that all three filters did well in "targeted" categories in terms of detection rates and false positive rates. The color histogram filter was able to achieve perfect detection rates for 13 out of 17 targeted categories, and more than 96.7% for the remaining 4 categories. The false positive rates of all categories except one (shift) are below 0.006%. The wavelet filter achieves perfect detection rates for all targeted categories while keeping the false positive rates below 0.0009%. The orientation histogram filter achieves perfect detection rates for 4 out of 7 targeted categories, while keeping the false positive rates below 0.0007%. The detection rates for the remaining 3 categories are 94.2%, 88.2% and 83.3% with false positive rates 0.00179%, 0.02089% and 0.00052% respectively.

The filters achieve good detection rates in more than half of the "un-targeted" categories. The main reason is that although a filter is not designed specifically to handle these categories, spammers tend to be conservative in randomizing images since they must preserve the readability of the spam messages. This makes makes some of the randomization techniques less effective. All individual filters achieve low false positive rates.

7.3 Combined Image Spam Detection Results

To study the effect of aggregating multiple spam filters, we have experimented with three simple aggregation methods: "AND", "OR", and "VOTE". Table 2 shows the results using all three methods to aggregate results from multiple spam filters. The results are presented in two groups: the first group is the union of all "targeted" categories from three spam filters, the second group shows the remainder of the categories. The last row shows the overall false positive and detection rates for all the categories.

We can see that the "AND" method consistently achieves a false positive rate of zero, but is effective in only about half of all spam image categories. Because individual filters are focusing on different features of the images, when all filters agree collectively, we expect a minimum false positive rate. The low false positive rate can be useful for certain use cases.

The "OR" method delivers the best spam detection rates at the cost of higher false positive rates. It detects most of the spam images, including most of "un-targeted" ones. For some use cases, a false positive rate of 0.17% is considered tolerable, but other use cases require lower false positive rates.

The "VOTE" method provides a compromise between false positive and detection rates; it holds the false positive rates below 0.0002% for all targeted categories, while achieving good detection rates. The only category that it didn't do well is shift.dots.line, which exhibits a 50% detection rate. For the un-targeted categories, VOTE keeps false positive rates below 0.0005% and has good detection rates (63.7%, 100%, 43%, 100%, and 33%, respectively). We can understand effect of VOTE better if we study one particular category closely, say "shift". We get false positive rates of about 0.024%, 0.011%, 0.021% and detection rates of about 99.1%, 92.3%, 88.2% from three filters. After VOTE, we

Category	fpos%	det%
dots	0	100.0
shift, fonttype, dots	0.00010	100.0
shift, bar	0.00012	100.0
bits	0.00016	100.0
shift, fonttype, dots, frame	0.00023	100.0
shift, dots, url	0.00044	100.0
fontsize, dots, line	0.00046	100.0
shift, dots, line	0.00077	100.0
shift, fuzzy	0.00118	100.0
size, bar	0.00126	100.0
size	0.00126	100.0
shift, url	0.00128	100.0
size, dots	0.00161	100.0
shift	0.02401	99.1
shift, dots, fontsize	0.00259	97.7
shift, dots	0.00617	97.4
size, fuzzy	0.00381	96.7
crop, dots, shape	0.00012	100.0
shift, linecolor, rotate	0.00014	100.0
shift, linecolor, fontcolor	0.00039	100.0
crop	0.00167	100.0
shift, linecolor	0.00390	92.8
shift, linecolor, fontcolor, fonttype	0.03452	42.8
shift, content, fontcolor	0.00009	0.0
crop, dots	0.00027	0.0
overall	0.08655	84.7

(a) Color Histogram

Category	fpos%	det%
bits	0.00002	100.0
size, bar	0.00003	100.0
fontsize, dots, line	0.00003	100.0
dots	0.00004	100.0
size	0.00012	100.0
size, fuzzy	0.00032	100.0
size, dots	0.00088	100.0
shift, dots, url	0	100.0
shift, fonttype, dots, frame	0.00001	100.0
shift, fuzzy	0.00003	100.0
crop	0.00004	100.0
crop, dots	0.00005	100.0
shift, fonttype, dots	0.00009	100.0
crop, dots, shape	0.00053	100.0
shift	0.01138	92.3
shift, dots, fontsize	0.00013	88.6
shift, dots	0.00061	85.9
shift, linecolor, rotate	0.00001	75.0
shift, linecolor, fontcolor, fonttype	0.00936	69.2
shift, url	0.00004	62.5
shift, content, fontcolor	0.00001	60.0
shift, linecolor	0.00012	56.5
shift, dots, line	0	50.0
shift, linecolor, fontcolor	0.00006	25.0
shift, bar	0.00002	0.0
overall	0.02393	82.3

(b) Haar Wavelet

Category	fpos%	det%
shift, linecolor, fontcolor	0.00029	100.0
bits	0.00052	100.0
crop	0.00061	100.0
shift, url	0.00065	100.0
shift, linecolor	0.00179	94.2
shift	0.02089	88.2
size	0.00052	83.3
shift, linecolor, rotate	0.00028	100.0
fontsize, dots, lint	0.00031	100.0
shift, dots, url	0.00060	100.0
shift, bar	0.00065	100.0
dots	0.00073	100.0
shift, fonttype, dots	0.00077	100.0
size, fuzzy	0.00106	100.0
shift, dots, fontsize	0.00081	89.8
shift, dots	0.00286	89.1
shift, content, fontcolor	0.00016	80.0
crop, dots, shape	0.00007	75.0
shift, linecolor, fontcolor, fonttype	0.02616	62.2
shift, dots, line	0.00011	50.0
shift, fuzzy	0.00075	50.0
crop, dots	0.00007	33.3
size, bar	0.00082	33.3
size, dots	0.00103	33.3
shift, fonttype, dots, frame	0.00109	28.6
overall	0.06360	81.6

(c) Orientation Histogram

Table 1: Results using different image spam filters. The categories shown in bold are the "targeted" group for each filter.

can achieve a false positive rate of 0.00018% and a detection rate of 96.4%.

Our results show that multiple filters can work better than an individual filter. For example, the VOTE method can deliver a better overall detection rate than each individual filter, while reducing the overall false positive rate by almost two orders of magnitude compared to each individual filter. This supports our design goal of making the system extensible.

7.4 Image Spam Filter Speed

	feature extraction time (ms)	training time (ms)	detection time (ms)
color histogram	20.9	19.8	2.0
Haar wavelet	5.4	9.4	1.1
orientation histogram	14.5	14.4	1.5

Table 3: Image Spam Filter Speed.

In order to understand the performance implications of the image spam filters, we have measured the processing time for the main components of our system on a P4 3GHz test machine. Table 3 shows the processing time for each image filter: image feature extraction time (assuming the image is rendered into memory ahead of time), training time where a new "known" spam image is inserted into the system and its threshold value is determined by comparing with 100,000 known non-spam images, and detection time where an incoming image's feature is compared with the features in the spam image feature database (we assume there are 10,000 spams in the database). Note that the training time is taken only for new kind of "known" spam image, thus it does not occur every time a known spam image is inserted. On average, a new image will take less than 50ms to be processed through all filters. The traditional computer vision based technique proposed by Wu [24] for image spam classification takes around 200-700ms to extract feature from an image, and roughly 2-3 seconds to classify an email with four images.

7.5 Image Spam Signature Size

Since the proposed image spam filters use feature vectors for near-duplicate detection, it is possible to distribute new feature vectors to end mail server systems over the Internet. Each participating email server can send its newly detected spam image "signatures" to the central server which aggre-

	Evaluator	AND		OR		VOTE	
Category	# spams	fpos%	det%	fpos%	det%	fpos%	det%
size,bar	23	0	33.3	0.00211	100.0	0	100.0
crop	15	0	100.0	0.00232	100.0	0	100.0
shift,dots,url	12	0	100.0	0.00104	100.0	0	100.0
size,dots	12	0	33.3	0.00352	100.0	0	100.0
dots	9	0	100.0	0.00077	100.0	0	100.0
size	9	0	83.3	0.00190	100.0	0	100.0
shift,linecolor,fontcolor	6	0	25.0	0.00074	100.0	0	100.0
shift,fonttype,dots	4	0	100.0	0.00096	100.0	0	100.0
shift,fuzzy	3	0	50.0	0.00196	100.0	0	100.0
bits	2	0	100.0	0.00070	100.0	0	100.0
fontsize,dots,line	2	0	100.0	0.00080	100.0	0	100.0
shift,url	26	0	62.5	0.00196	100.0	0.00001	100.0
shift,fonttype,dots,frame	8	0	28.6	0.00132	100.0	0.00001	100.0
shift,bar	2	0	0.0	0.00078	100.0	0.00001	100.0
size,fuzzy	36	0	96.7	0.00517	100.0	0.00002	100.0
shift,dots,fontsize	100	0	78.4	0.00353	100.0	0	97.7
shift,dots	185	0	75.0	0.00963	100.0	0.00001	97.4
shift	276	0	83.3	0.05610	100.0	0.00018	96.4
shift,linecolor	76	0	55.1	0.00579	97.1	0.00002	91.3
shift,dots,line	3	0	50.0	0.00088	100.0	0	50.0
shift,linecolor,rotate	7	0	75.0	0.00043	100.0	0	100.0
crop,dots,shape	5	0	75.0	0.00072	100.0	0	100.0
shift,linecolor,fontcolor,fonttype	240	0	26.9	0.06958	83.6	0.00046	63.7
shift,content,fontcolor	6	0	0.0	0.00026	100.0	0	40.0
crop,dots	4	0	0.0	0.00039	100.0	0	33.3
overall	1071	0	63.6	0.17336	96.1	0.00072	88.9

Table 2: Results using different evaluators to aggregate results from multiple image spam filters.

gate the spam image signatures and periodically broadcast them back to email servers.

To see how practical our method is for supporting collaboration between peers, we have calculated the network overhead for exchanging information about new spams. In our approach, only the image feature and the associated threshold value generated by the spam filter need to be exchanged over the network. The three spam filters require $(64+36+16+3)\times 4 = 476$ bytes per known image spam.

8. Conclusion and Future Work

In this paper, we present an image spam detection system. By examining the content of new images contained in incoming emails and detecting images that are near-duplicates of known spam images, our system can effectively detect image spams while maintaining a low false positive rate. Rather than using computationally expensive algorithms to detect new types of image spams designed to thwart conventional computer vision algorithms, our system uses efficient algorithms to target randomization methods used to generate large number of unique but visually similar image spams from template images. Our system is designed to be integrated with existing anti-spam technologies to boost the detection rate of image spams. Our prototype system has demonstrated high detection rates in most spam categories while achieving a less than 0.001% false positive rate using the "VOTE" aggregation method.

We are planning to work on new feature extraction units for image spam filters that can improve the performance of the categories in which our current system does not perform well. Furthermore, since image spam is constantly evolving, we believe it is a constant battle to find new features that can effectively defeat new image spam techniques.

9. Acknowledgements

This work was supported in part by research grants from Google, Intel, Microsoft and Yahoo!. The authors would also like to thank Prof. Fei-Fei Li for helpful discussions.

10. References

[1] H. B. Aradhye, G. K. Myers, and J. A. Herson. Image analysis for efficient categorization of image-based spam e-mail. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, 2005.

[2] The CAPTCHA Project, 2000. `http://captcha.net`.

[3] Commtouch Inc. *2006 spam trends report: Year of the zombies.* `http://www.commtouch.com/documents/Commtouch_2006_Spam_Trends_Year_of_the_Zombies.pdf`.

[4] Distributed checksum clearinghouse, march 2007. `http://www.rhyolyte.com/dcc`.

[5] J. P. Eakins and M. E. Graham. Content-based image retrieval: A report to the jisc technology applications programme. Technical report, University of Northumbria at newcastle, Institute for Image Data Research, 1999.

[6] W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *Intl. Workshop on Automatic Face- and Gesture-Recognition, IEEE Computer Society*, pages 296–301, 1995.

[7] P. Graham. A plan for spam, august 2002. `http://www.paulgraham.com/spam.html`.

[8] Iron Port Inc. *Image Spam: The E-Mail Epidemic of 2006*. http://ironport.com/pdf/ironport_image_spam_datasheet.pdf.

[9] Y. Ke, R. Sukthankar, and L. Huston. An efficient parts-based near-duplicate and sub-image retrieval system. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 869–876, 2004.

[10] J. Levine. Experiences with greylisting. In *Second Conference on E-mail and Anti-Spam*, 2005.

[11] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Ferret: A Toolkit for Content-Based Similarity Search of Feature-Rich Data. In *Proceedings of the ACM SIGOPS EuroSys Conf.*, 2006.

[12] Mail Avenger, 2006. http://www.mailavenger.org.

[13] Princeton Spam Image Benchmark. http://www.cs.princeton.edu/cass/spam.

[14] C. Pu and S. Webb. Observed trends in spam construction techniques: A case study of spam evolution. In *Proc. of the 3rd Conf. on E-Mail and Anti-Spam*, 2006.

[15] A. Ramachandran and N. Feamster. Unerstanding the nework-level behavior of spammers. *ACM SIGCOMM Computer Communication Review*, 36(4), Oct. 2006.

[16] Y. Rui, T. S. Huang, and S.-F. Chang. Image retrieval: Current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10(4):39–62, 1999.

[17] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*. AAAI Technical Report WS-98-05, 1998.

[18] R. Segal, J. Crawford, J. Kephart, and B. Leiba. Spamguru: An enterprise anti-spam filtering system. In *First Conference on Email and Anti-Spam (CEAS)"*, 2004.

[19] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[20] SpamAssassin, 2007. http://spamassassin.apache.org.

[21] Symantec Inc. *The State of Spam, A Monthly Report - January 2007*. http://www.symantec.com/avcenter/reference/Symantec_Spam_Report_-_January_2007.pdf.

[22] R. C. Veltkamp and M. Tanase. Content-based image retrieval systems: A survey. Technical Report UU-CS-2000-34, Utrecht University, Information and Computer Sciences, 2000.

[23] Vipul's Razor, 2007. http://razor.sourceforge.net.

[24] C.-T. Wu, K.-T. Cheng, Q. Zhu, and Y.-L. Wu. Using visual features for anti-spam filtering. In *IEEE International Conference on Image Processing*, volume 3, pages 509–512, 2005.

[25] D.-Q. Zhang and S.-F. Chang. Detecting image near-duplicate by stochastic attributed relational graph matching with learning. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 877–884, 2004.

APPENDIX

(1) shift (2) crop

(3) crop,dots,shape (4) size,bar

(5) shift,fonttype,dots,frame (6) shift,linecolor,rotate

(7) shift,linecolor,fontcolor,fonttype (8) shift,content,fontcolor

(9) shift,url (10) size,fuzzy

(11) fontsize,dots,line (12) shift,dots,fontsize

(13) crop,dots (14) bits

Figure 4: Example spams belongs to different categories of spamming techniques.