A Comparison of Methods for Near Duplicate Image Detection

William Beluch

Otto von Guericke University, 39104 Magdeburg, Germany
william.beluch@st.ovgu.de

Abstract. The problem of detecting near-duplicate images in a database has many practical applications, and thus numerous methods addressing the problem have been developed. However, many published studies tend to use different datasets for testing, report different metrics, and don't provide an easily usable implementation. Previous review papers comparing methods are informative, but limit their scope by their choice of test datasets. This study aims to test simple implementations of a variety of content-based near-duplicate image detection algorithms, and evaluate their accuracy and training times on two datasets reflecting the varying challenges of near duplicate detection. Results show that simple intensity based methods perform reasonably well on simple transformations and are suitable for many applications, but struggle with more complex near-duplicate detection.

1 Introduction

The near duplicate detection (NDD) problem has become increasingly relevant with the explosion of material available on the Internet. While finding exact duplicate images is a trivial task, locating near duplicates is significantly harder. These duplicates often arise from post-acquisition modifications to a source image (e.g. cropping, brightness), which over time, can result in multiple changes from the original picture. A separate category of modifications comprises changes in acquisition settings of the image; when taking a picture of the same object, one could modify lighting, viewpoint/angle, zoom, etc. As expected, methods vary in their ability to detect certain modifications.

Another aspect to consider in the context of duplicate detection is whether a given situation presents a retrieval or a discovery problem. Retrieval consists of querying a collection of images, usually stored as a database of some type of image signature, for near-duplicates given a specific input image [3, 5, 11, 16, 22]. This is conceptually and algorithmically simpler than the discovery problem, which entails finding all groups of duplicate images in a given image collection [6, 18]. This aspect of NDD problem is very context dependent and will not be investigated further - the rest of this paper will focus on methods and implementations geared towards the retrieval context, given one query image at a time.

NDD has many applications, the most common being copyright detection, curation of search results (to avoid returning duplicates), and reduction of required storage space (remove duplicates). NDD has also been applied to the analysis of video feeds in an attempt to link different coverage sources of the same event quickly [22]. Finally, with the increasing popularity of social media and image sharing websites (e.g. Twitter, Instagram, etc.), there has been interest in tracking user behavior via the spread of duplicate images [16]. With the advent of the Internet and the presence of millions of images in a given database, it is important to keep in mind that efficient retrieval methods are required for practical applications of NDD.

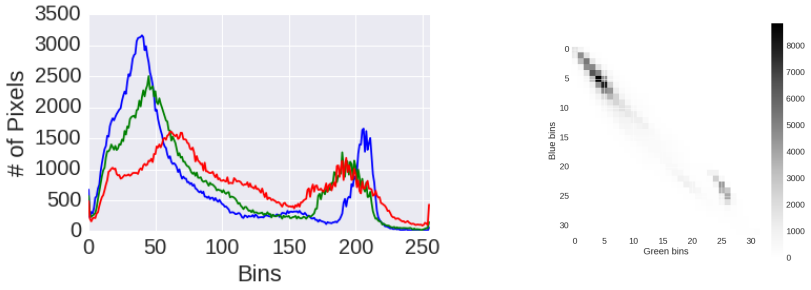
In the following section, a detailed overview of duplicate image detection methods will be presented. Section 3 will outline the experiments of this paper and their implementation. In Section 4 the results of the experiments will be presented, and finally the paper will conclude in Section 5.

2 Methods for Duplicate Image Retrieval

Generally, methods for duplicate image detection follow a similar pattern. One needs to find an accurate representation of an image that can be quickly compared to other images. An important aspect to consider is the speed of retrieval - less complex representations can likely be compared to each other more quickly and take up less storage space. Whether one chooses to lean more towards accuracy or retrieval speed depends on many factors, including size of the image database, how often queries are run, how diverse the image collection is, and of course context specific importance of speed or accuracy (in some situations we want all duplicates, in others getting 90% of duplicates is acceptable).

2.1 Histogram Comparison

One of the simplest and oldest methods of image comparison uses a histogram to represent the tonal distribution in an image. Such a histogram shows the frequency of pixel intensity values (thus the X axis will range from 0-255), and can be generated from either a gray scale image or color images. In the former case, separate histograms are generated for each of the red, blue, and green channels, and then flattened into one vector for a singe image representation (Fig. 1a). For an even more complex representation, 2D or 3D color histograms can be used to incorporate dependencies between the three color channels (Fig 1b). Image histograms can then be compared using a variety of distance metrics, including the L1 norm, the chi-squared distance, or the histogram intersection kernel [15]. While histograms are simple to obtain and compare, their performance is known to suffer when dealing with many common image transformations [16]. The method may provide good overview of similar images, but not necessarily duplicates.



(a) Color Histograms (b) 2D Histogram

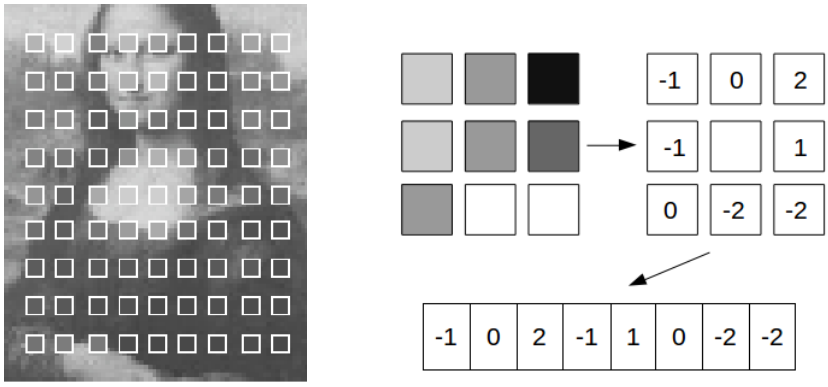
Fig. 1. a) 1D Color histograms of random image. These can be combined to form an image representation. **b)** Example 2D color histogram, where the X and Y axes are two separate channels (discretized into bins) and the color is the pixel corresponds to the pixel intensity.

2.2 Intensity Based Local-Image Signature

A more advanced intensity method incorporates local spatial information. An image is divided into a 9x9 grid, and for each of the grid point centers, the relative intensities of the 8 grid points surrounding the point are computed (Fig 2a, b). The resulting feature vector is only of length 648 (81 grid points x 8 neighbors), and since relative intensity is encoded simply as -2,-1,0,1,2, can be stored as a 648 byte array (Fig 2b). This method has the advantage of being rather fast to generate image signatures, and requires very little storage space, enabling scale ups to huge image databases. The method was shown to be robust to some compression and resizing, but does not perform well with extensive cropping or rotation [19].

2.3 Stochastic graph matching

The previous intensity based methods fail to incorporate scene and semantic information. In an attempt to account for this issue, a method was developed that represents an image as an attributed relational graph (ARG). The nodes of the graph represent component parts of a scene, while edges represent relations between various parts of scenes. To measure similarity of images, for example image A and image B, the authors look at the likelihood of obtaining the graph of image B by applying stochastic random processes to the graph of image A, the idea being that more similar images will have a higher likelihood [22]. This method was important for providing a more in depth analysis of scene and composition in the context of duplicate detection, however was never demonstrated to scale well beyond a few hundred images as it is computationally complex.



(a) Imagematch Grids (b) Example Array Calculation

Fig. 2. a) Example of image broken into a 9x9 grid of 5x5 squares [19]. **b)** Process of generating an image signature as presented in [19] - the relative intensities of the eight neighbor grid points are calculated, and flattened into an 8-dim vector. This repeated for all 81 grid points, resulting in an 81x8 = 648 dim feature vector.

2.4 SIFT Descriptors and the BoF Model

The development of SIFT descriptors, and the derivatives thereof provided a basis for successful image representation and classification for many years, until such methods were recently superseded by the use of convolutional neural networks. SIFT (scale invariant feature transform) is an algorithm used to locate useful local features of an image, called keypoints. These are designed to be only locally dependent, and are thus relatively robust to image scale, noise, and illumination. Each keypoint is then represented by a 128 dimensional SIFT vector, called a descriptor. The details of how the keypoints are located and how transformation invariance is ensured are beyond the scope of this paper, but are well explained in [12].

While [12] also provides an efficient nearest neighbors approximation for comparing keypoints and descriptors between different images, the full representation is generally too complex for large-scale applications as it is very high dimensional: an image can have hundreds or even thousands of keypoints, each represented by a 128-dimensional descriptor vector. As a simplification, the Bag of Features (BoF) model was proposed, inspired by the Bag of Words model commonly used in natural language processing. In such a model, a vocabulary of visual words is generated by clustering a large amount of input descriptors and using the cluster centers as the visual words. In k-means clustering, the size of the visual vocabulary will thus be the number of user-defined cluster centers k. To generate the representation for a given image, the descriptors of the image are assigned to the closest visual word, and the counts of the number of descriptors assigned to each visual word are stored as a histogram. As a result, an image can then

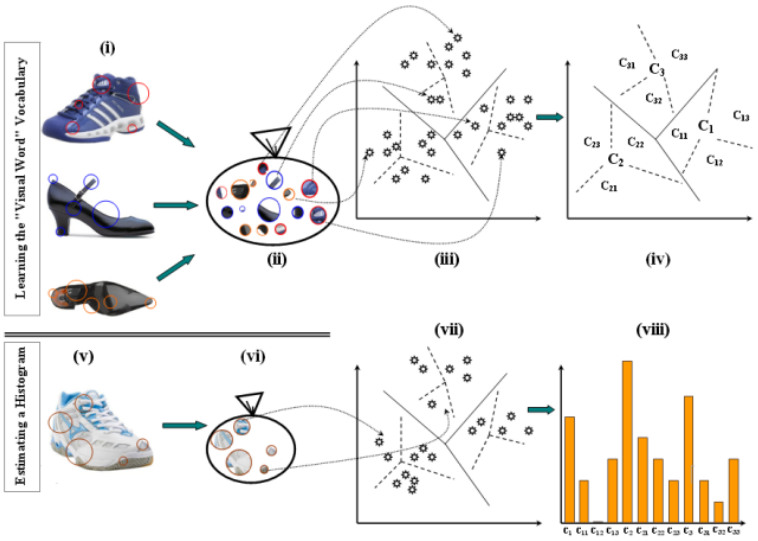


Fig. 3. Diagram of Bag of Features model i) Keypoints are located on various input images using an algorithm such as SIFT. **ii)** Descriptors are extracted from these keypoints. **iii)** Descriptors are clustered using k-means clustering to generate a visual vocabulary. **iv)** The visual vocabulary is represented by the k visual words. **v, vi, vii)** Descriptors are extracted from new input image keypoints, and assigned to closest visual words. **viii)** Images are represented as a histogram of visual words, the y axis being the counts of each visual word in the relevant image [17].

be represented by a single k-dimensional vector, and comparison methods for histograms can be used (Fig 3).

Various alternatives to SIFT descriptors have been developed, including SURF (speeded up robust features), and BRIEF (binary robust independent elementary features). The study of descriptors is a field of its own, but for the purpose of NDD, one can choose to use any of the relevant image descriptors.

2.5 MinHash

Even using 1D vectors of length k from the BoF model, the search to find similar images in a large database, even with optimized nearest neighbor algorithms, can be quite slow. As a result, different groups developed a series of improvements using hashing methods to greatly decrease retrieval times and storage costs. The idea of using locality sensitive hashing for near duplicate image detection was first introduced in [11]. Briefly, LSH hashes more similar items to the same bucket (as opposed to standard cryptographic hash methods) - thus the more similar the items, the greater the chances of a hash collision. By using multiple hash functions, the rate of chance collisions (false negatives) can be controlled, and by performing comparisons only with the images in a given bucket, retrieval time is greatly decreased.

An improvement on the above method utilizes MinHash [5]. MinHash is a form of LSH for estimating the similarity of two sets quickly. By appropriate choice of hash functions and proper parameter choosing, the method can very closely approximate the Jaccard similarity, defined as the ratio of the intersection over the union of two sets. For a given hash function h, the minimum value of the result of applying h to a given set is called hmin. If h is applied to two sets A and B, the same value of hmin is returned if the item returning it is in the intersection of the two sets. This equates to the equation of the Jaccard similarity. In practice, one chooses a number of hash functions k, and represents a signature as the set of hmins obtained by applying each hash function over the set. To estimate the similarity of two sets, one then uses y/k, where k is the number of hash functions, and y is the number of hash functions where the two sets return the same value of hmin.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \qquad Pr[hmin(A) = hmin(B)] = J(A,B) \tag{1}$$

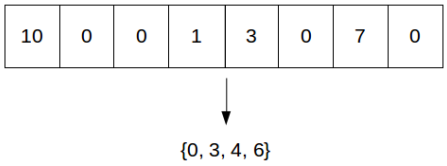


Fig. 4. An original Bag of Features histogram (above) is converted into a binary representation as described in [5], retaining the index of the visual word in the set only if the count is greater than zero. Count information is lost for future analysis.

The method in [5] uses MinHash based off of vectors from the BoF model using SIFT descriptors. They describe three different similarity measures: set similarity, weighted set similarity, and histogram intersection. The latter two methods use a binary set representation of the BoF model - a given visual word from the vocabulary is either present in a given image, or it isn't - the count of said visual word is discarded (Fig. 4). The third method, histogram intersection, involves building an alternate expanded visual word vocabulary in order to incorporate this frequency information [5].

2.6 Other Improvements to the SIFT BoF model

Many improvements to the SIFT BoF model have been made in the field of image classification, and the aspects regarding the image representation could readily be applied to the NDD problem. Both spatial pyramidal matching combined with sparse coding and fisher vectors have been used with great success in image representation in a classification framework [13, 20, 21]. An improvement specifically designed for NDD involves an entropy based filtering of SIFT descriptors to reduce false positives in retrieval [3].

2.7 Convolutional Neural Networks

Convolutional neural networks have dominated image classification tasks in the past four years, superseding previous methods based on BoF models using local image descriptors and SVM classifiers [14]. While the deep learning performed by CNNs takes far too long to likely be practical in near duplicate image detection, one can use a pre-trained CNN on a large image dataset to output feature vectors as opposed to classification probabilities. This is obtained by simply taking the output of the layer before the final softmax classification layer, which in the case of the VGG network is the second to last layer (Fig 5). The output vector thus represents the weights of the various features learned by the convolutional layers in the network. Using these vectors as the image representation, euclidean distance or some other distance metric can be applied in a nearest neighbors framework to retrieve the most similar images. As the features learned by the CNNs have proven to be rather complex, the hope is that the network will be very tolerant to a wide range of transformations.

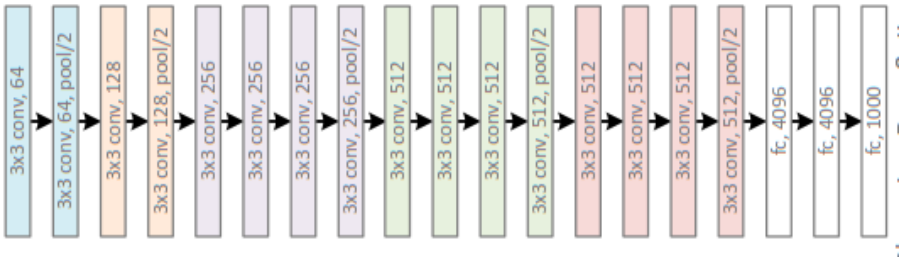


Fig. 5. Architecture of the VGG-19 convolutional neural network. For this study, a pre-trained slightly smaller network is used. Feature vectors are obtained from the output of the second to last layer, labeled fc, 4096. [7]

2.8 Previous comparison studies

There are two notable previous studies comparing NDD methods. Both of these studies are extremely thorough in their analysis, but tend to focus on different aspects of the problem and use datasets of a different nature. The work of Aly et al. compares seven different methods, comprising intensity approaches, full representation SIFT descriptor approaches, BoF model approaches, and hashing approaches [1]. They provide an extremely in depth and useful analysis of retrieval times for these methods, as well as a brief comparison of performance. However they don't provide details as to what aspects of NDD various methods over-perform or under-perform in. The more recent work of Thomee et al. compares a similar slew of methods, albeit using an alternative descriptor to SIFT, and focuses more on accuracy. Notably they provide an extremely useful

analysis for practical implementation that identifies the most common transformations seen on images on the web [16]. While they have the data to focus on which transformations affect which methods, the results are not presented in a clear, intuitive fashion. As an improvement on the Aly study, they use very large, disparate, and realistic datasets found on the web for their testing purposes. However, they do not provide a dataset that features the pre-acquisition transformations we are also interested in testing.

3 Implemented Methods

Five of the above algorithms were implemented in Python using available open source libraries to be tested on accuracy and retrieval time. The five algorithms chosen are briefly described below, and were chosen due to their relative practicality to implement, popularity in the prevailing literature, and potential to scale to very large datasets (with appropriate improvements). Note that while retrieval time is addressed in this paper, measurements should be taken skeptically due to potentially naive implementations, very limited hardware for testing, and varying runtime conditions.

- **Histogram matching** Histogram matching was implemented using methods in the OpenCV image library [10]. Images are represented by a flattened 512 dimensional vector, obtained from a 3D RGB histogram with 8 bins for each channel. Increasing the number of bins would increase the resolution of the image representation, at the cost of larger storage and increased retrieval time. Histograms were compared using the histogram intersection kernel using a brute-force nearest neighbors algorithm.

- **Intensity based signature matching** The algorithm discussed in Wong et al. and introduced in the introduction has a very intuitive Python implementation called image-match [8]. This library was used to generate image signatures for each image, which were compared using Euclidean distance, and most similar images were returned using a brute-force nearest neighbors algorithm.

- **SIFT BoF matching** SIFT keypoints and descriptors were obtained from images using the OpenCV library. Visual vocabularies for the Bag of Features model were similarly generated using OpenCV and k-means clustering. Various sizes of the visual vocabulary and number of input images used as sources for the clustering were performed, and are discussed in further detail in the results. Comparisons between the resultant k-dimensional image representations were done using the histogram intersection kernel, and most similar images were returned as part of a nearest neighbors algorithm using a brute-force nearest neighbors algorithm.

- **MinHash on SIFT BoF vectors** The MinHash implementation uses the python library datasketch [23]. The k-dimensional vectors from the SIFT

BoF approach are first binarized (each entry in the vector of visual words is replaced with the index of the entry if the entry is greater than 0, otherwise it is removed) to switch to a set approach as described in [5]. Then, 128 different hash functions are applied to each member in the vector, and the resultant sketches (tuples of MinHash values) are stored. At query time, images with estimated Jaccard similarity above 0.5 are retrieved. Of the retrieved images, the similarity is then again approximated in order to rank and return the top results. This algorithm is difficult to optimize, as finding a suitable Jaccard similarity cutoff requires a not insignificant amount of work, and is dependent on the corpus size.

– **Pre-trained convolutional neural network** The VGG-16 convolutional neural network, pre-trained on the extensive ImageNet database, is used to generate feature vectors of input images [14]. Implementation is using the Python library Keras with a Theano backend [4]. The feature vectors are obtained by taking the output of the last dense layer before the final softmax layer in the neural network, and are thus 4096 dimensional vectors. These are then compared using euclidean distance, and most similar images were returned using a brute-force nearest neighbors algorithm. For efficient calculation of the feature vectors, a NVIDIA NVS 5400M GPU was used.

3.1 Datasets

Two primary datasets are used in this study, each designed to test different aspects of the duplicate image detection problem.

– Flickr 25k Dataset The Flickr dataset from Yahoo research features random images collected from Flickr, a photo sharing site. The advantages of this set are that the images are not some artificially designed dataset, and represent images one would find on the web in an real evaluation setting. Additionally, the content of the images varies greatly, including scenes, objects, people, and abstract images [9]. This set was used to test the robustness of the various NDD algorithms to common, simple transformations. Each image in the dataset had eight different transformations applied to it, summarized in Table 1, and illustrated in Fig 6a. These transformations were performed in Python using the OpenCV library.
– Pasadena Buildings Dataset While the above dataset is useful for testing common transformations, it doesn't contain duplicate images with varying acquisition settings. The Pasadena buildings dataset is exactly that, containing 50 sets of 5 images, each set consisting of 5 images of the same building featuring various angles, lighting conditions, and amounts of zoom (Fig. 6b) [2].

3.2 Evaluation

To evaluate accuracy in the Flickr dataset, the metric precision at k=9 was used. As there are 8 transformed images plus the original image, there are 9 potential

Transformation	Description
blur	Blur applied using normalized box filter with size 10x10 pixels.
bright	Pixel intensity doubled (contrast), and increased by 50 (brightness).
dark	Pixel intensity halved (contrast), and decreased by 50 (brightness).
rot	Image rotated 90°clockwise.
crop	Image cropped to be the center 25% (by total area).
noise	20% of randomly selected pixels are switched to be white or black.
resize	Image is resized with new aspect ratio(0.9 x 0.7).
text	Random color string of length 10 is overlayed at a random height, ten pixels from the left side.

Table 1. Descriptions of image transformations.

matches for each image. Thus a precision of 1.0 would imply that the first 9 images are the correct duplicates, a precision of 0.56 would imply that 5 of the 9 returned images are accurate duplicates, etc. Retrieval time was also measured in the Flickr dataset. Note that the experiments were set up assuming that a query image is not already in the database - thus obtaining its image signature involves applying the appropriate algorithm as opposed to simply performing a lookup.

To evaluate the Pasadena buildings dataset over the 50 queries, mean average precision was used.

There is no overlap between the images used for training the BoF model, and the images used in testing/querying. As the Flickr dataset is very large, this was not an inconvenience. For the Pasadena buildings dataset, 30 of the families of 5 were used for training, and 20 of the families of 5 were used as the test set (all 100 in the test set were used as a query image once).

For retrieval time measurements, the Flickr dataset was used, and measurements were made within the Python scripts using the datetime module.

$$Precision = \frac{tp}{tp + fp} \qquad Average\ Precision = \frac{\sum_{k=1}^{n}(P(k) * rel(k))}{\#\ of\ relevant\ documents} \tag{2}$$

$$MAP = \frac{\sum_{q=1}^{Q}}{AveP(q)} \tag{3}$$

(a) Flickr25k dataset

(b) Caltech buildings dataset

Fig. 6. a) Example image from the Flickr dataset, and the resulting images from the applied transformations. *1:* Original *2:* Aspect ratio *3:* Crop *4:* Rotate *5:* Blur *6:* Noise *7:* Text overlay *8:* Brightness increased *9:* Brightness decreased **b)** Example family from the Pasadena buildings dataset.

4 Results

4.1 Flickr Dataset - Artificial Transformations

Table 2 summarizes the average performance of each algorithm as the corpus size varies. One would expect that as the corpus size increases, precision will decrease, as more images means a higher chance of false positives. This is indeed the case for all of the algorithms, however tapers off from the 4500 size set to the 9000 size set - this implies the algorithms will scale well to larger sets. The CNN is the best performing, followed closely by imagematch. SIFT and histogram matching perform rather well at comparable levels. As expected, the minhash implementation is less accurate than the SIFT, and shows a strong decrease in performance as the corpus size is increased. This algorithm likely can be perform better with proper tuning of a similarity threshold cutoff.

Table 3 highlights the effect of changing both the number of descriptors and the number of input images used for training the bag of features model. Performance increases as more descriptors are used (a longer feature vector) as expected, however increasing the number of input images has a marginal effect. This is rather surprising, as one would expect 100 images is not sufficient to represent the diversity in the dataset. However, as Flickr images are all photographic in nature, perhaps the dataset is not as diverse as intended, despite the plethora of subject matters in the images.

	Histogram	ImageMatch	SIFT	MinHash	CNN
Corpus Size					
900	0.611	0.778	0.652	0.338	0.849
4500	0.566	0.767	0.622	0.103	0.806
9000	0.552	0.767	0.607	0.010	0.794

Table 2. Precision @9 performance of the various algorithms.

	Number of Visual Words		
Number of Images	**100**	**500**	**1000**
100	0.507	0.578	0.610
500	0.512	0.581	0.617
1000	0.511	0.581	0.607

Table 3. Precision @9 performance of the SIFT BoF approach.

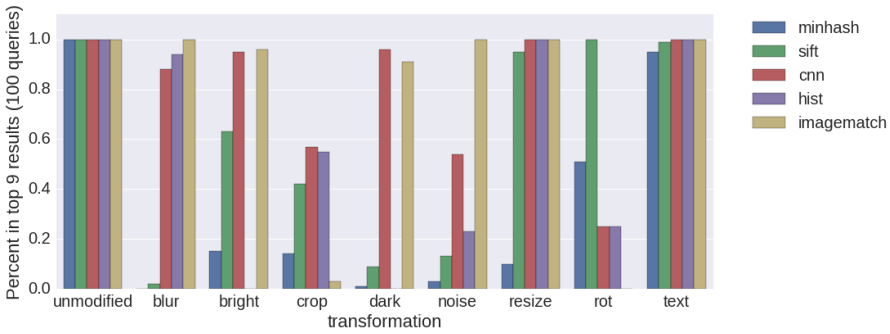
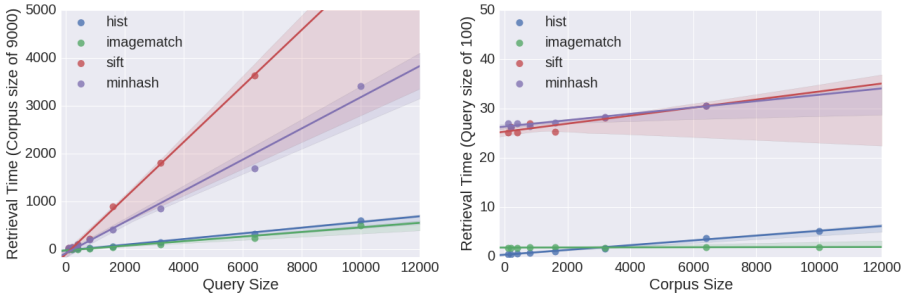


Fig. 7. Performance of algorithms on the various induced image transformations.

Figure 8 shows the performance of the algorithms on various transformations (expressed as a percent retrieved in the top 9 results). Text overlays and resized images are handled well by all of the algorithms (except minhash), while all of the other transformations have various results. Histograms fail to identify images with brightness changes (dark and light) as expected, and imagematch fails to deal well with rotated and cropped images, as mentioned in the publication. SIFT retrieves all rotated images as the features are rotational invariant, but performs very poorly with blurred images. Overall, the CNN does very well in identifying most transformations, with the exception of rotated images. The most difficult transformations to detect are cropping, dark, and noise - however imagematch does very well with the former two.

All of the methods with the exception of minhash are expected to be linear in query time, as they have to compute a distance between the query and every image in the database. The overall complexity is expected to be O(n*(m+c)), where n is the number of query images, m is the number of images in the corpus



(a) (b)

Fig. 8. a) Retrieval time vs by size of image database. **b)** Retrieval time vs size of query set.

set, and c is some constant reflecting the time required to generate an image vector from the query image. Thus the differences between them arise in the image vector generation phase - as seen in Fig 8b, the SIFT vector generation takes significantly longer than histogram or imagematch. The CNN is excluded from this analysis simply due to difficulties in measurement. The values fluctuated greatly due to problems with the GPU, and were difficult to accurately measure. However, in general it performs around 2x slower than SIFT.

Surprisingly, MinHash performs only slightly better than SIFT - the point of minhash is to have sublinear query times, which are unfortunately barely achieved in these experiments. This likely has to due with details of the implementation and the choosing of a proper distance threshold - if too many potential matches are returned, not much retrieval time is saved.

Changing the corpus size was expected to show similar results to changing the query size, but resulted in virtually flat curves. This is likely due to errors in the timing process, but needs to be investigated further.

4.2 Pasadena Buildings Dataset - Viewpoint changes

On the Pasadena buildings dataset the CNN performs substantially better than the other algorithms at identifying the duplicate images, although the SIFT method does reasonably well, and the histogram is mediocre (Table 4, Fig 9). Imagematch performs much poorer than the other algorithms in this test. As the CNN is able to learn very complex features, it is not surprising that it performs well in this more difficult task - SIFT is the next most complicated image description, and also performs well on the more complex task.

5 Conclusion

Overall, the methods have varying performance dependent on the types of images expected. If on only expects simple modifications such as blurs, brightness,

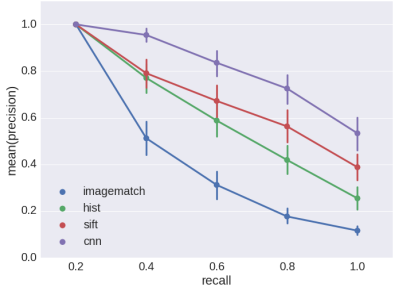


Fig. 9. Precision-recall curve on the Pasadena dataset. Precision is averaged over 100 queries.

	Histogram	**Imagematch**	**SIFT**	**CNN**
MAP	0.606	0.423	0.682	0.810

Table 4. MAP of algorithms on the 100 queries on the Pasadena buildings dataset

resizing, and overlays, the imagematch method is supreme and runs very fast. To check for rotations, one could always manually rotate the images and use them as additional queries. However for more complex detection such as in the Pasadena buildings dataset, more complex methods are required, shown by the CNN and SIFT outperforming the other algorithms.

Further investigation is needed to optimize the MinHash method and bring its performance to a level of the SIFT BoF implementation. There are also many possible improvements to the SIFT approach that can be implemented, discussed in the introduction, including trying the use of different descriptor algorithms.

Choosing the appropriate near duplicate detection algorithm is a trade-off between performance and speed. For more complex tasks, more complex algorithms are needed, however they perform significantly slower. For a practical application, a good knowledge of the types of near duplicates expected is extremely valuable, as it allows one to choose a simple algorithm for simpler duplication problems.

References

1. Mohamed Aly, Mario Munich, and Pietro Perona. Indexing in Large Scale Image Collections: Scaling Properties and Benchmark. *IEEE Workshop on Applications of Computer Vision (WACV)*, 2011.
2. Mohamed Aly, Peter Welinder, Mario Munich, and Pietro Perona. Towards automated large scale discovery of image families. *2009 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009*, pages 9–16, 2009.
3. Eagle Ave, Ann Arbor Mi, and Zhe Wang. High-Confidence Near-Duplicate Image Detection. *Proceedings of the 2Nd ACM International Conference on Multimedia Retrieval*, pages 1–8, 2012.
4. Francois Chollet. keras. https://github.com/fchollet/keras, 2015.
5. O Chum, James Philbin, and A Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. *Bmvc*, 3:4, 2008.

6. Jun Jie Foo, Justin Zobel, and Munch Madonna. Clustering Near-Duplicate Images in Large Collections returned by Google Images using the query Edvard. *Proceedings of the 9th ACM SIGMM International Workshop on Multimedia Information Retrieval*, pages 21–30, 2007.
7. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *Arxiv.Org*, 7(3):171–180, 2015.
8. Ryan Henderson. Image-match. https://github.com/ascribe/image-match, 2016.
9. Mark J Huiskes and Michael S Lew. The MIR flickr retrieval evaluation. *Proceeding of the 1st ACM international conference on Multimedia information retrieval MIR 08*, 4(November):39, 2008.
10. Itseez. Open source computer vision library. https://github.com/itseez/opencv, 2015.
11. Yan Ke, Rahul Sukthankar, and Larry Huston. Efficient Near-duplicate Detection and Sub-image Retrieva. *ACM International Conference on Multimedia*, pages 869–876, 2004.
12. David G Lowe. Distinctive image features from scale invariant keypoints. *Int'l Journal of Computer Vision*, 60:91–11020042, 2004.
13. Jorge Sanchez, Florent Perronnin, Thomas Mensink, Jakob Verbeek, Image Classification, S Jorge, Perronnin Thomas, and Mensink Jakob. Image Classification with the Fisher Vector : Theory and Practice. *International Journal of Computer Vision*, 105(3):22–245, 2013.
14. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Iclr*, pages 1–14, 2015.
15. Michael J. Swain and Dana H. Ballard. Swain1.pdf. *International Journal of Computer Vision*, 7(1):11–32, 1991.
16. Bart Thomee, Mark J. Huiskes, Erwin M. Bakker, and Michael S. Lew. An evaluation of content-based duplicate image detection methods for web search. *Proceedings - IEEE International Conference on Multimedia and Expo*, 2013.
17. Brian Tomasik, Phyo Thiha, and Douglas Turnbull. Tagging products using image classification. *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 792–793, 2009.
18. Xin-jing Wang and Lei Zhang. Duplicate Discovery on 2 Billion Internet Images. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 429–436, 2013.
19. Chi H. Wong, Marshall Bern, and David Goldberg. An Image Signature For Any Kind of Image. *Proc. of International Conference on Image Processing*, 2002.
20. J Yang, K Yu, Y Gong, and T Huang. Linear spatial pyramid matching using sparse coding for image classification. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1794 – 1801, 2009.
21. Chunjie Zhang, Shuhui Wang, Qingming Huang, Jing Liu, Chao Liang, and Qi Tian. Image classification using spatial pyramid robust sparse coding. *Pattern Recognition Letters*, 34(9):1046–1052, 2013.
22. Dong-Qing Zhang and Shih-Fu Chang. Detecting image near-duplicate by stochastic attributed relational graph matching with learning. *Proceedings of the 12th annual ACM international conference on Multimedia - MULTIMEDIA '04*, page 877, 2004.
23. Eric Zhu. Datasketch. https://github.com/ekzhu/datasketch, 2015.