ENGN4528/6528: Computer Vision
Term Project Report

Libo Yin (u5483742), Manab Chetia (u5492350)
and Sakda Eamkulworapong(u5610617)*
The Australian National University

June 7, 2015

1 Introduction

The Household Object Recognition Challenge aims to train an image classifier that discriminates pictures of household objects, such as mugs, boxes, and toys. The given dataset contains 50 object classes, and four images are provided for each class. These four images are taken from different angles at the same well-illuminated object, with a clean, slightly reflective desktop background.

This challenge is characterised by its tiny dataset compared to the number of object classes. The PASCAL VOC 2007 dataset[2], in contrast, provides 9963 images in 20 classes. Such scarcity renders popular methods such as convolutional neural network unusable. However, the fact that the images are almost noise-free means lower-level approaches can be taken. Our group managed to create two distinctively different final solutions. The first one is based on SIFT and hue histogram, and is designed specifically for this problem. In this solution, we also propose a new classifier that borrows ideas from the bag of visual words model and the nearest neighbour classifier. The second solution is a more generalized system based on GIST and neural network. Both systems are implemented in Python.

2 SIFT + Hue Histogram Solution

2.1 SIFT Classification

A common practice in SIFT-based image classification is the bag of visual words model. Its assumptions are stated as follows:

> SIFT descriptors from all training images form an unknown number of clusters in the 128 dimensional vector space. These clusters have identical and diagonal covariance matrix.[1]

Under these two assumptions, SIFT keypoint descriptors from all training images are clustered with k-means, and centroids are extracted as codewords. With nearest neighbour classification, SIFT keypoint descriptors from a test image are converted to a bag of codewords. An image classification problem is thus converted to a document classification problem.

*Libo is enrolled in ENGN4528. Manab and Sakda are enrolled in ENGN6528.

[1] An assumption of k-means clustering algorithm[8]. This is a very weak assumption. Given the first assumption satisfied, we suggest modelling extracted SIFT keypoint descriptors as a mixture of Gaussian.

While such method has been proven effective for large dataset, our empirical test has shown that it is unreliable for this particular problem. First, the number of clusters K is hard to choose. Second, even with identical K, the variance of classification accuracy is still large. (For details please refer to section A.3.) We believe that such instability is a consequence of the small dataset. With OpenCV, the entire dataset produces a total number of 45,296 SIFT keypoints. In a vector space with dimensionality as high as 128, it is not very likely that the aforementioned two assumptions shall be satisfied.

Here, we propose Set Nearest Neighbour (SNN), a classification algorithm for tasks where a datapoint is a set of independent vectors in a vector space. Denote the number of classes by C, the number of training vectors per class by N (for notational convenience, assume N is a constant), the dimensionality of a vector by D, and the number of vectors in the testing dataset by M. Then the training dataset has shape $C \times N \times D$, and the testing datapoint has shape $M \times D$. We aim to create a C-dimensional vector denoting the probability of the test datapoint (recall that it is a set of independent vectors) belonging to each class. This vector is collapsed from a 3-dimensional array of shape $M \times C \times N$, where entry (i, j, k) denotes the distance, according to some metric, from the i-th vector of the test datapoint, to the k-th vector in the training dataset of class j. The third axis is collapsed by taking the minimum, followed by the first axis collapsed by taking the sum.

In other words, for each vector in the testing datapoint, we find, for each class, the minimum distance between that particular testing vector and any vector in the training dataset of this particular class. This step gives a C-dimensional distance vector. And since all vectors in the testing datapoint are independent, we take the summation of all their corresponding distance vectors, which gives a C-dimensional distance vector from the testing datapoint as a whole to each class. This vector is then turned into a discrete probability distribution by subtracting each entry from the largest value, and normalize such that all entries sum up to 1. Note that the result probability vector contains at least one zero, so ϵ has to be added to each entry before taking logarithm. This algorithm requires brute force calculation of distances, but for small, noise-free dataset, it provides exceptional accuracy and reliability.

For this particular problem, we extract 30 strongest SIFT keypoint descriptors from each image. Since the dataset is split such that exactly one image in each class is left as the test datapoint, $N = 90$, $M = 30, C = 50$, and $D = 128$. We choose Euclidean distance as the metric, following Lowe's original paper [5]. An efficient implementation of Euclidean distance is provided in `scipy.spatial.distance.cdist`. On 1000 random train/test splits, the SIFT + SNN classifier by itself produces an average accuracy of 0.9181, with a standard deviation of 0.0295. The classification takes approximately 0.8 second per split[2]. Error analysis is shown in figure 1.

2.2 Hue Histogram Classification

Many frequently misclassified images of the SIFT classifier are distinctively different in color. This is within expectation, as color and geometry should be independent features of an image, and an ideal classifier should take both into consideration.

Thanks to the fact that the SIFT + SNN classifier is a soft probabilistic classifier, we can easily combine it with any other classifiers in the same framework. Here, we propose the hue histogram intersection classifier. First, the hue channel of each image is binned into a 181-slot histogram[3]. Since the three training images of each class are only different in the angle from which the picture is taken, we assume that their difference in hue histogram is insignificant.

[2]Tested on a 2013 MacBook Pro with 2.0GHz processor. Parallel processing is not used in this project.

[3]In OpenCV, the range of values in the hue channel of an 8-bit image is a closed interval [0, 180].

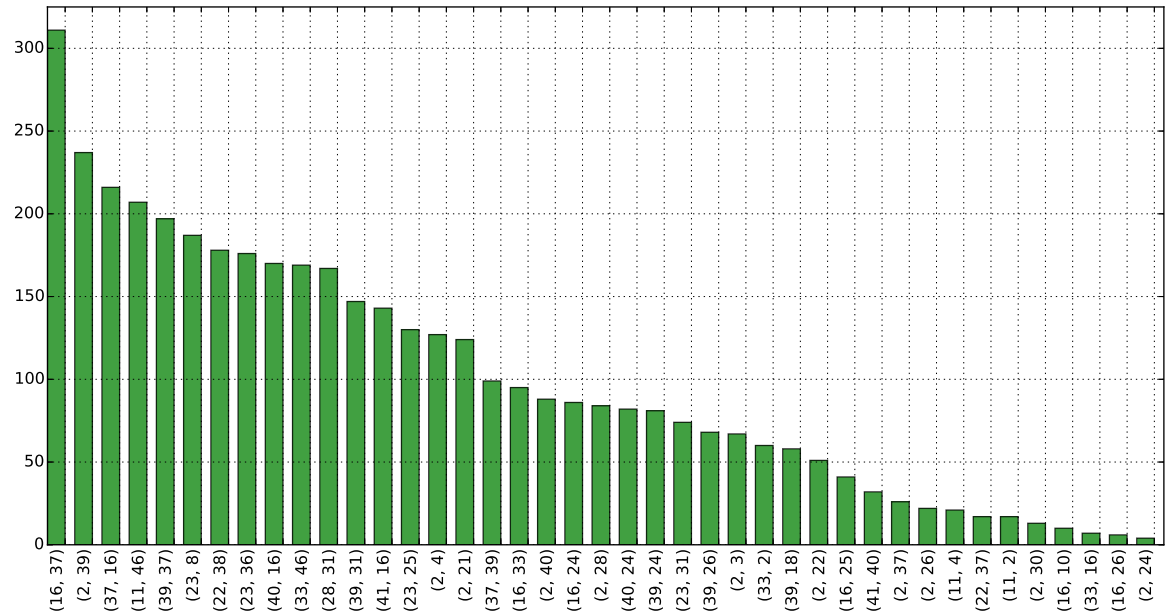


Figure 1: Misclassification statistics of the SIFT + SNN classifier on 1000 random train/test splits. Labels under bars are in (`ground truth`, `prediction`) format. For example, the most frequent error is misclassifying image of class 16 as of class 37. The classifier made 4,095 misclassifications in total.

Therefore, we define the hue histogram of a class as the mean of the hue histograms of the three training images in that particular class. Note that this is different to the SIFT + SNN classifier, in which SIFT keypoint descriptors from the three testing images of each class are considered independent. A testing image is classified by measuring the intersection between its own hue histogram and the hue histogram of each class:

$$K_{\cap}(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} \min(\mathbf{a}_i, \mathbf{b}_i)$$

where $\mathbf{a}$ and $\mathbf{b}$ are the two histograms with n bins each. Like SNN, the hue histogram classifier is a soft classifier that produces a C-dimensional probability vector. However, since histogram intersection is a similarity function instead of a metric, the normalization is carried out by first subtracting each entry in the similarity vector by the minimum, and normalize such that all entries add up to 1.

The result of the SIFT + SNN classifier and the hue histogram intersection classifier are combined by addition, with the log probability vector from the hue histogram intersection classifier multiplied by $w = 0.35$. Weight factor w guarantees numerical stability, and its value is chosen by cross-validation. The combined classifier produces an average accuracy of 0.9896, with a standard deviation of 0.0130. The running time of the combined classifier remains at approximately 0.8 second per split, meaning that the running time of the hue histogram intersection classifier by itself is covered by noise. Discrimination analysis of the combined classifier is shown in figure 2.

Although it is possible to combine more classifiers, we choose not to do so since parameter tuning will be difficult, and the model will be less elegant.

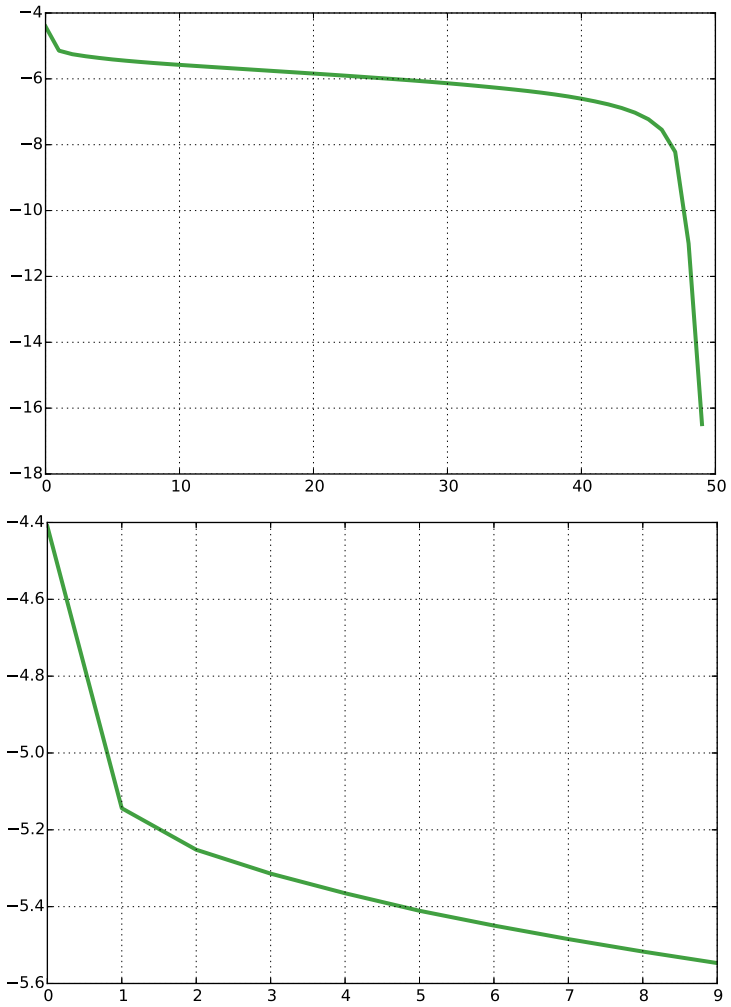


Figure 2: Discrimination analysis for the combined classifier. Top: Sorted C-dimensional probability vectors, averaged across all test images on 1000 random train/test splits. Horizontal axis is rank, and vertical axis is log probability. Most classes receive similar likelihood, hence the plateau in the middle. A few classes receive very low likelihood, due to the fact that normalization in both classifiers produce zeros. Bottom: Magnification of the top 10 classes in the top figure. It is clear that the leading class stands out from all others.

3 GIST + Neural Network Solution

Currently, convolutional neural network is the state-of-the-art tool for image classification and object recognition. However, as mentioned in section 1, training a convolutional neural network directly on the given dataset is infeasible due to its size. In this solution, we aim to approximate its performance by training a fully connected, feed-forward neural network on global features from an augmented dataset.

3.1 Dataset Augmentation

The original dataset contains 200 images in 50 object classes. In order to increase the size of dataset, we apply the following transformations to each image:

1. Rotation by $0°$, $30°$, $60°$, $90°$, $120°$, $150°$, $180°$, $210°$, $240°$, $270°$, $300°$, and $330°$.
2. Gaussian blur with $\mu = 0,\ \sigma = 1.1$.
3. Gaussian noise with $\mu = 0,\ \sigma = 0.001$.
4. Darkening with $\gamma = 1.5$.
5. Brightening with $\gamma = 0.5$.

The augmented dataset then contains 240 images per class[4].

3.2 Feature Extraction

In section 2.1, we demonstrated using a set of local SIFT keypoint descriptors as the global feature of an image. Here we use GIST[7], a global feature by itself, to discriminate between classes. This choice is backed by the fact that all images in the given dataset have similar background, and there is only one object in each image. GIST summarises the gradient information (scales and orientation) for different parts of an image, and provides a low dimensional representation of the image's naturalness, openness, roughness, expansion, and ruggedness. The GIST descriptor of an image is calculated as follows:

1. Convolve an image with 20 Gabor filters at 8 orientations for two finer scales and 4 orientations for the coarser scales, giving rise to 20 feature maps.
2. Divide each feature map into 16 regions (4×4 grid), then average the feature values within each region for 3 colour channels.
3. Concatenate the 3×16 averaged values of all 20 feature maps, resulting in a $3 \times 16 \times (8 + 8 + 4) = 960$ dimensional vector.

3.3 Network Architecture

The neural network has a fully connected, feed-forward structure, with one input layer, one hidden layer, and one output layer. The input layer contains 960 neurons, same as the dimensionality of a GIST descriptor. The output layer contains 50 neurons, corresponding to the number of object classes. The number of neurons in the hidden layer is chosen to be the average of the input layer and the output layer.

For the input layer and the hidden layer, the activation unit is rectified linear unit (ReLU), which uses rectifier $f(x) = \max(0, x)$ as the activation function. ReLU is chosen since it induces sparsity in the hidden units, does not face gradient vanishing problem as with sigmoid and tanh, and is computationally efficient. The activation function for the output layer is softmax, a common practice for multi-class neural network classifier.

[4] 12 rotations $\times$ (4 perturbation + 1 identity) $\times$ 4 original images per class

The loss function in our neural network is the cross entropy error, defined as:

$$E(\mathbf{o}, \mathbf{t}) = -\left(\mathbf{t} \cdot \log(\mathbf{o}) + (\mathbf{1} - \mathbf{t}) \cdot \log(\mathbf{1} - \mathbf{o})\right)$$

where $\mathbf{o}$ denotes the output distribution[5], and $\mathbf{t}$ denotes the ground truth, a vector with exactly one 1 entry and zero elsewhere. We choose the cross entropy error rather than the conventional squared error loss as it leads to faster training and better generalisation. Moreover, it is extremely sensitive to overconfident predictions[6].

3.4 Training and Testing

The augmented dataset of $240 \times 50 = 12000$ images is randomly split into a training dataset of 9000 images and a testing dataset of 3000 images. Note that this is different from the testing scheme in the first solution, where the testing dataset is formed by randomly choosing exactly one of the four images from each class.

Both networks are trained with stochastic gradient descent (SGD) with Nesterov momentum and RMSprop[3]. The parameters for SGD are: learning rate $= 0.01$, decay $= 10^{-6}$, and momentum $= 0.9$. For RMSprop, the parameters are: initial learning rate $= 0.001$, decay rate $= 0.9$. On this problem, empirical experiments shows that RMSprop has a faster speed of convergence. This is because it modulates the learning rate of each weight based on the magnitudes of its gradients, which has an equalising effect. The comparison is shown in figure 3.

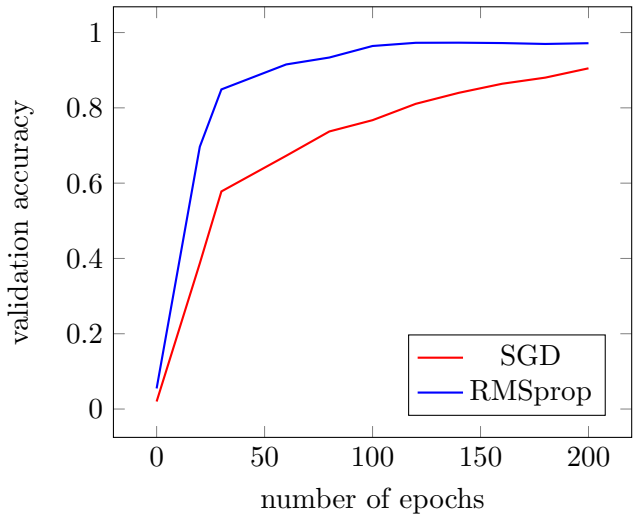


Figure 3: Comparison of convergence speed of SGD and RMSprop

To prevent overfitting, we introduce a dropout rate of 0.2 in the hidden layer. That is, with probability of 0.2, a hidden unit is set to 0. A dropout rate of 0.2 has also been tested in the input layer, but it lead to slower convergence and was hence removed.

With a batch size of 20 and 120 epochs of training, the classifier produced an accuracy of 97.32%. With other aspects unchanged, we also tried reducing the dimensionality of input vectors from 960 to 400 with PCA (in which case the input layer has 400 neurons), and the classifier produced an accuracy of 96.6%.

[5]The output of a softmax layer is a discrete probability distribution.

4 Project Timeline

During the entire lifetime of project, we used the GitHub Issues page[6] to track unsolved problems, enhancements in progress, as well as personal timelines. Important events are listed in the following table. Note that different solutions may have different testing schemes.

Date	Member	Event
Mar 27	All	Submission of proposal
Apr 13	Libo	20 SIFT + nearest neighbour: 80% accuracy
Apr 15	Manab	Root-SIFT + nearest neighbour: 75% accuracy
Apr 24	Libo	20 SIFT + SNN: 91% accuracy
Apr 25	Sakda	Tested Shackenberg's Minimal bag of visual words classifier: 60% accuracy
Apr 29	Manab	GIST + nearest neigbour: 84% accuracy
Apr 30	Manab	GIST + SVM on polynomial kernel: 80% accuracy
May 2	Libo	Soft probabilistic classifiers
May 2	Libo	25 SIFT + SNN + hue histogram: 95% accuracy
May 3	Sakda	SIFT match without filter or threshold: 80% accuracy
May 3	Sakda	SIFT match with fileter and threshold: 90% accuracy
May 10	Sakda	Custom bag of visual words classifier: 85% accuracy
May 12	Manab	GIST + neural network: 84% accuracy
May 12	Manab	Dataset augmentation
May 12	Manab	GIST + neural network + SGD on augmented dataset: 94.4% accuracy
May 20	Manab	GIST + PCA + neural network + RMSprop on augmented dataset: 96.6% accuracy
May 25	Libo	Map-reduce paradigm
May 25	Libo	30 SIFT + SNN + hue histogram, with tuned parameter: 98.9% accuracy
May 26	Manab	GIST + neural network + DropOut + RMSprop on augmented dataset: 97.32% accuracy
May 27	All	Project demo

5 Conclusion

With the given dataset, we managed to create two distinctively different solutions. Both of them exploit the fact that images are noise-free, and both of them have achieved an accuracy higher than 95%. Listed in the appendix are a few of our other attempts. They are unable to produce satisfactory classification accuracy on the given dataset, but have nonetheless inspired us on our final solutions.

[6] https://github.com/liboyin/horc/issues

A Other Attempts

A.1 SIFT on Hue Channel

As mentioned in section 2.2, color and geometry are independent features of an image. This is not entirely true. Theoretically, extracting SIFT keypoints from the hue channel should allow a feature that combines color and geometry information. Unfortunately, with OpenCV, some images in the given dataset are only able to produce as few as 1 SIFT keypoint from its hue channel, making this feature unusable.

A.2 Classification with kNN and SVM

In early stage of this project, we tried classifying image features directly with distance-based classifiers. Assuming SIFT keypoint descriptors are representative to their classes, we extracted 25 strongest SIFT keypoint descriptors from each image. For a testing image, each SIFT keypoint descriptor is classified with 4-nearest-neighbour classifier, and the overall class of that testing image is decided by voting. Despite the fact that the aforementioned model assumption is clearly violated, this classifier produced an accuracy of 0.82. Replacing set of local features with a global feature GIST, the accuracy improved slightly to 0.84.

We also tested SVM as classifier on the same features. Unlike kNN, which uses Euclidean distance as metric, SVM uses polynomial kernel with degree = 3 instead of RBF. This choice is made by cross-validation. The SVM classifier produced a similar accuracy, but runs considerably faster than kNN since it avoids brute force calculation of pairwise distance.

A.3 Bag of Visual Words

Our bag of visual words classifier is based on [9]. In the training stage, we extract 10 strongest SIFT keypoint descriptors from each training image[7]. We did not use all SIFT keypoint descriptors for speed reasons. These descriptors are clustered with k-means, and the centroids are extracted as codewords. Each class is then converted into a histogram of frequency, where each bin is a codeword.

Likewise, in the testing stage, 10 strongest SIFT keypoint descriptors are extracted from each image. These descriptors are classified to codewords with nearest neighbour on Euclidean distance. With the occurrence frequency of codewords obtained, the testing image is represented as a histogram, which is further classified by nearest neighbour on Euclidean distance.

The impact of number of clusters K on prediction accuracy is shown in figure 4. Note that the accuracy has a standard deviation of at least 0.025. The reason of such a high standard deviation has been discussed in section 2.1. By setting cluster factor = 1.5 and choosing top 80 SIFT feature descriptors from each training image, this classifier can reach an accuracy of 80-85%.

[7] The train/test split is the same as in the first solution.

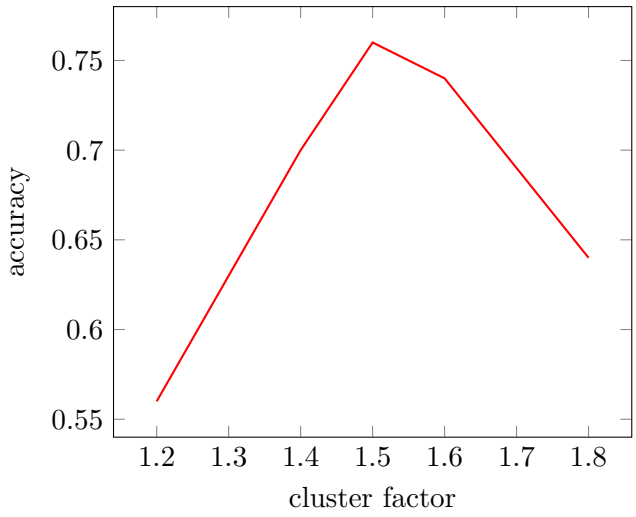


Figure 4: Impact of K on classification accuracy. K = cluster factor $\times$ number of classes.

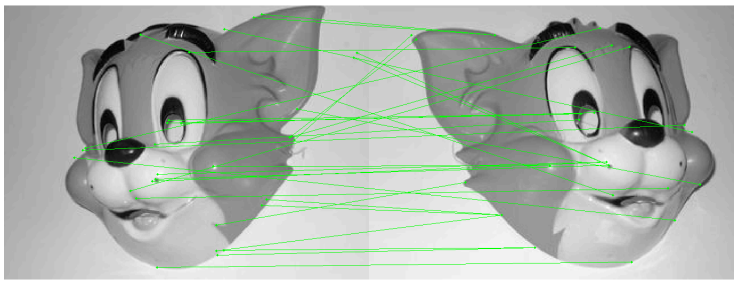
A.4 SIFT Match

We assume that images of the same class have more matching SIFT keypoints than images of different classes. With `siftmatch`, each testing image is compared to all training images, and is classified to the class which provides the highest number of matches. This approach allows an accuracy of 80%, with some misclassifications shown in Figure 5.

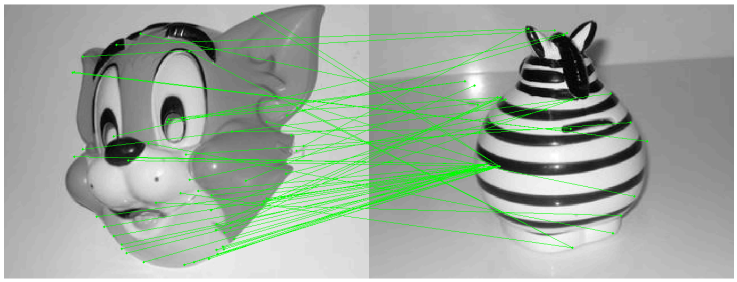
By experiments, we found that specifying edge threshold = 12 and applying a negative Gaussian filter[1, 4]

$$-\begin{bmatrix} 0.0113 & 0.0838 & 0.0113 \\ 0.0838 & 0.6193 & 0.0838 \\ 0.0113 & 0.0838 & 0.0113 \end{bmatrix}$$

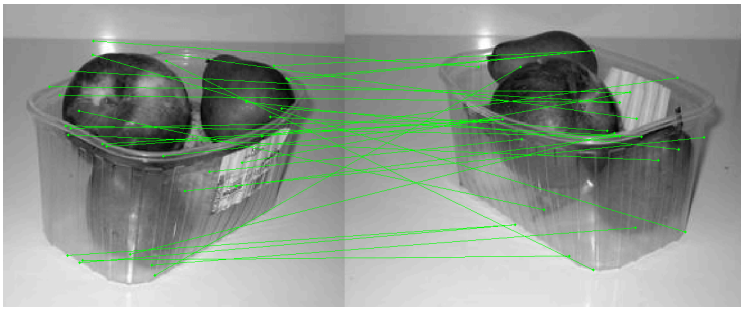
before extracting SIFT keypoints improves accuracy to 90%. We believe that this filter has reached a balance between reducing noise and suppressing image details. A few examples are shown in figure 6 and figure 7.



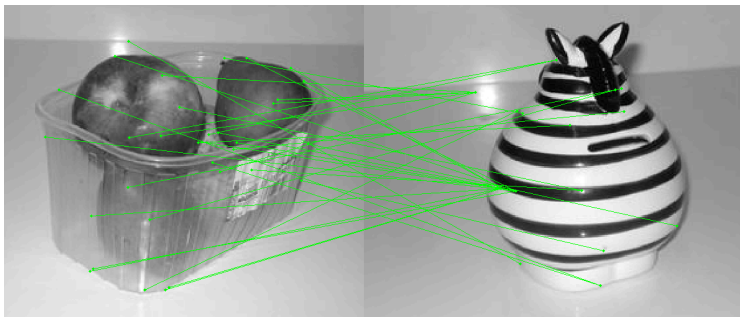
(a) 33 matches on images of the same class.



(b) 58 matches on images of different classes.



(c) 31 matches on images of the same class.



(d) 32 matches on images of different classes.

Figure 5: Most misclassifications are on toys and fruits.

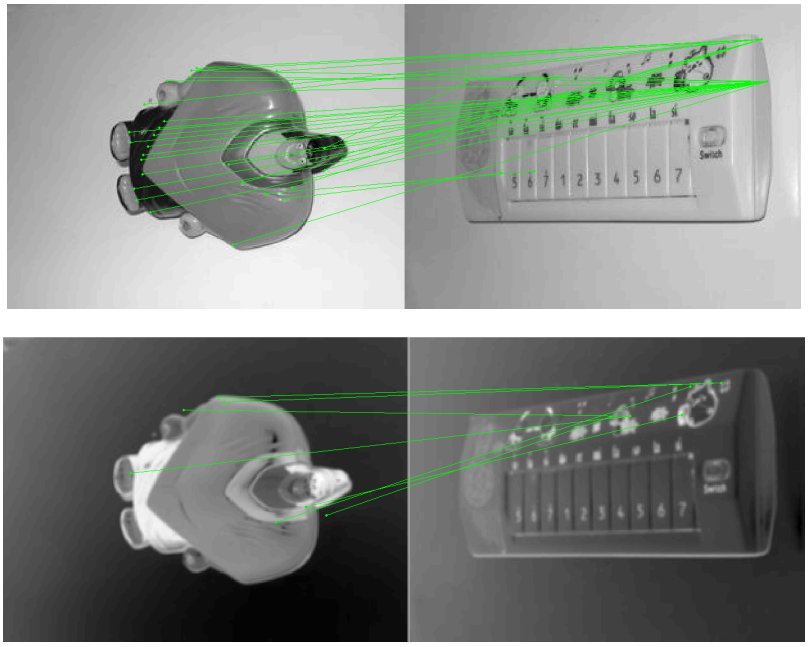


Figure 6: An example of SIFT match before and after filtering. Top: original images. Bottom: filtered images.

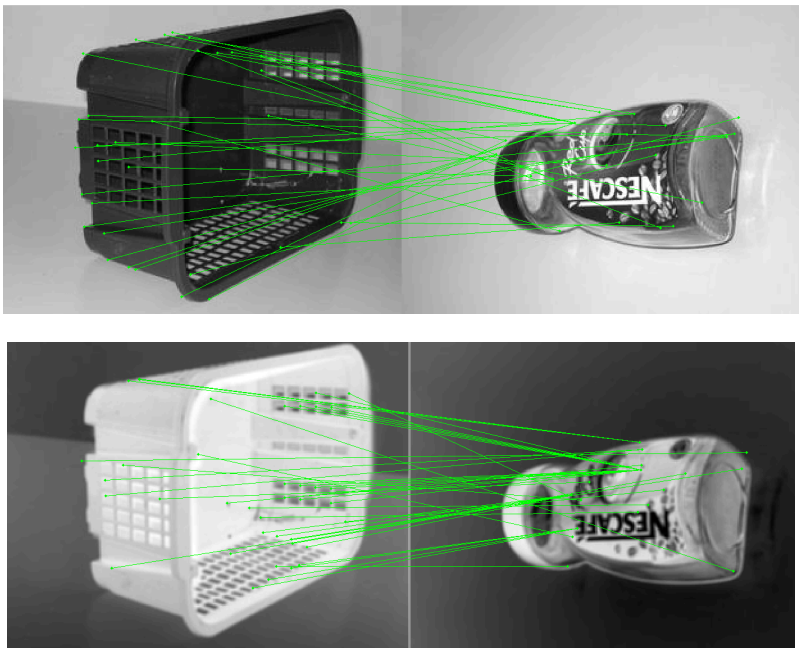


Figure 7: On certain images, filtering leads to more matching SIFT keypoints. Top: original images. Bottom: filtered images.

References

[1] Samarth Brahmbhatt. Computer vision: Why is gaussian filter used in sift algorithm? http://www.quora.com/Computer-Vision/Why-is-Gaussian-Filter-used-in-SIFT-algorithm, 2013. Accessed: 4 June 2015.

[2] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[3] Geoffrey Hinton. Overview of mini-batch gradient descent. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, 2012. Accessed: 1 June 2015.

[4] Tony Lindeberg. Scale-space: A framework for handling image structures at multiple scales. *CERN European Organization for Nuclear Research-Reports-CERN*, pages 27–38, 1996.

[5] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[6] James McCaffrey. Why you should use cross-entropy error instead of classification error or mean squared error for neural network classifier training. https://goo.gl/MrGP1I.

[7] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[8] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition, 2007.

[9] shackenberg. Minimal bag of visual words image classifier. https://github.com/shackenberg/Minimal-Bag-of-Visual-Words-Image-Classifier, April 2014. Accessed: 1 June 2015.