Large Scale Image Classification: Fast Feature Extraction, Multi-codebook Approach and Multi-core SVM Training

Thanh-Nghi Doan and François Poulet

Abstract. The usual frameworks for image classification involve three steps: extracting features, building codebook and encoding features, and training the classifier with a standard classification algorithm (e.g. SVMs). However, the task complexity becomes very large when applying these frameworks on a large scale dataset like ImageNet containing more than 14 million images and 21,000 classes. The complexity is both about the time needed to perform each task and the memory and disk usage (e.g. 11TB are needed to store SIFT descriptors computed on the full dataset). We have developed a parallel version of LIBSVM to deal with very large datasets in reasonable time. Furthermore, a lot of information is lost when performing the quantization step and the obtained bag-of-words (or bag-of-visual-words) are often not enough discriminative for large scale image classification. We present a novel approach using several local descriptors simultaneously to try to improve the classification accuracy on large scale image datasets. We show our first results on a dataset made of the ten largest classes (24,807 images) from ImageNet.

1 Introduction

Image classification is one of the important research topics in the areas of computer vision, object recognition, and machine learning. Low-level local image features and the bag-of-words model (BoW) are the core of state-of-the-art image classification systems. The usual frameworks for image classification involve three steps:

Thanh-Nghi Doan
IRISA, Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France
e-mail: `thanh-nghi.doan@irisa.fr`

François Poulet
Université de Rennes I, IRISA, Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France
e-mail: `francois.poulet@irisa.fr`

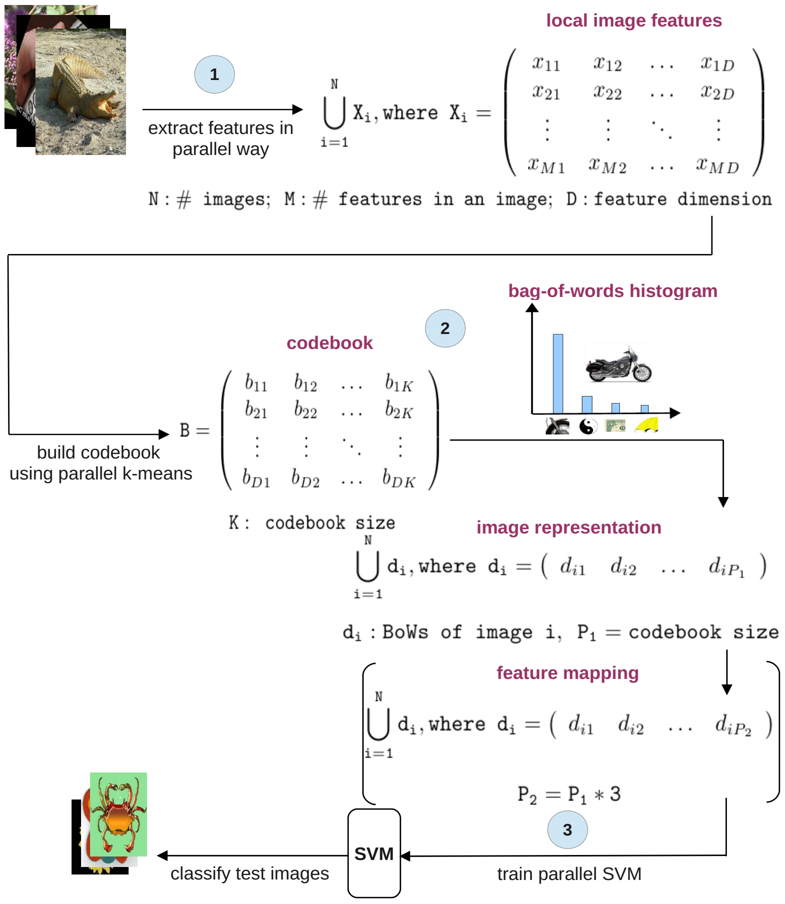
1) extracting features, 2) building codebook and encoding features, and 3) training classifiers. Step 1 is to extract low-level local invariant features from images: the usual choices are SIFT [Lowe, 2004], SURF [Bay et al., 2008], and dense SIFT (DSIFT) [Bosch et al., 2007]. Step 2 is to build codebook and encode features: k-means clustering algorithm is the usual choice for building codebook, BoW model is the state-of-the-art of feature encoding. The image representation is obtained by applying the clustering algorithm and then constructing the histogram of each image SIFT distribution in the previously obtained set of clusters. Step 3 is to train classifiers: many systems choose either linear or non-linear kernel SVM classifiers. All these frameworks are evaluated on small datasets, e.g. Caltech 101 [Li et al., 2007], Caltech 256 [Griffin et al., 2007], and PASCAL VOC [Everingham et al., 2010] that can fit into desktop memory. However, the emergence of ImageNet [Deng et al., 2009] with more than 14 million images and 21,000 classes makes the complexity of image classification very large and difficult to deal with. This challenge motivates us to study an efficient framework in both computation time and classification accuracy. In this paper, we show how to address the challenge and achieve promising results over the state-of-the-art classification algorithms on ImageNet. We propose a fast and efficient framework for large scale image classification, as shown in Fig. 1. Our key contributions include:

1. A parallel version of LIBSVM to deal with a large scale dataset in reasonable time.
2. A novel approach using several different local robust descriptors and how to combine them efficiently by using multi-feature and multi-codebook approach.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related work on large scale classification and image representation. The benchmark datasets in computer vision are introduced in section 3. In section 4, we present the efficient low-level local image features for many vision tasks. Our multi-feature and multi-codebook approach and parallel LIBSVM are described in section 5. Section 6 presents numerical test before the conclusion and future work.

2 Related Work

Large Scale Classification: Many previous works on image classification have relied on BoW models [Csurka et al., 2004], local feature quantization, and support vector machines. These models can be enhanced by multi-scale spatial pyramids (SPM) [Lazebnik et al., 2006] on BoW or histogram of oriented gradient (HoG) [Dalal and Triggs, 2005] features. Fergus *et al.* [Fergus et al., 2009] study semi-supervised learning on 126 hand labeled Tiny Images categories, Wang *et al.* [Wang et al., 2009] show classification on a maximum of 315 categories. Li *et al.* [Li et al., 2009] do research with landmark classification on a collection of 500 landmarks and 2 million images. On a small subset of 10 classes, they could improve BoW classification by increasing the visual vocabulary up to 80K visual words. To make large scale learning more practical, many researchers are beginning to study strategies where the original data in low-dimensional space is often



local image features
1
extract features in parallel way
$\bigcup_{i=1}^{N} X_i$, where $X_i = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{M2} & \dots & x_{MD} \end{pmatrix}$
N : # images; M : # features in an image; D : feature dimension
bag-of-words histogram
2
codebook
$B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1K} \\ b_{21} & b_{22} & \dots & b_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ b_{D1} & b_{D2} & \dots & b_{DK} \end{pmatrix}$
build codebook using parallel k-means
K : codebook size
image representation
$\bigcup_{i=1}^{N} d_i$, where $d_i = (d_{i1} \quad d_{i2} \quad \dots \quad d_{iP_1})$
d_i : BoWs of image i, P_1 = codebook size
feature mapping
$\bigcup_{i=1}^{N} d_i$, where $d_i = (d_{i1} \quad d_{i2} \quad \dots \quad d_{iP_2})$
$P_2 = P_1 * 3$
3
SVM
classify test images
train parallel SVM

Fig. 1 The overview of our framework for large scale image classification

transformed to high- dimensional space by a nonlinear mapping induced by a particular kernel and then efficient linear classifiers are trained on the resulting space [Deng et al., 2010], [Perronnin et al., 2010]. Some recent works consider exploiting the hierarchical structure of dataset for image recognition and achieve impressive improvements in accuracy and efficiency, but has not evaluated classification minimizing hierarchical cost. Related to classification is the problem of detection, often treated as repeated 1-vs-all classification in sliding windows. In many cases, such localization of objects might be useful to improve classification, but even the most efficient of state-of-the-art techniques [Vedaldi et al., 2009; Everingham et al., 2010] take a lot of computation time and thus it is very difficult to deal with large scale datasets. The difference between our work and previous studies is to take into account parallel algorithms to speedup two processes: extracting features and training classifiers. Our experiments show first promising results improving both classification time and accuracy and confirm that parallel algorithms are very essential for large scale image classification in terms of time efficiency.

Image Representation: Local image features and BoW model are the core of state-of-the-art image classification systems. Representing an image based on BoW model includes the three following steps: 1) feature detection, 2) feature description, and 3) codebook generation. Recent works have studied these steps and achieved impressive improvements. However, in each processing step there exists a significant amount of lost information, and the resulting visual-words are often not discriminative enough for large scale image classification applications. Many different approaches have been proposed to improve the discriminative power during these steps. At the feature detection step, multiple local features are grouped to obtain a more global and discriminative feature. At the feature description step, high-dimensional descriptors or descriptors enhanced by other information have been studied to get more image information [Winder and Brown, 2007]. At the codebook generation step, many previous works have proposed efficient quantizers or codebooks that reduce quantization errors and preserve more information of feature descriptors [Moosmann et al., 2006], [Philbin et al., 2008]. We have a more general view for all these three steps and propose a novel approach that combines both multi-feature and multi-codebook approach to construct the final image representation. Our approach aims to increase the discriminative power of image representation by embedding more useful information from the original image features. In multi-feature and multi-codebook approach, first BoW histograms of images for each feature channel is constructed based on their corresponding codebook. The result is a bag-of-BoW for all different feature types extracted in step 1 and we call it a bag-of-visual packets or a bag-of-packets (BoP). Finally, all BoW histograms in BoP are concatenated to form the final image representation, as shown in Fig. 3. In our novel approach the final image representation is constructed by using parallel multi-feature and multi-codebook computation, improving the discriminative power of image representation for large scale image classification. These are the major differences between our approach and previous studies.

3 Datasets

There are quite a few benchmark datasets for image classification, such as MNIST (`http://yann.lecun.com/exdb/mnist`), Caltech 101, Caltech 256, PASCAL VOC, etc. However, there are very few multi-class image datasets with many images for more than 300 categories. In recent years, there is an agreement that it is necessary to build a large scale dataset for studying object retrieval and recognition systems. One is Tiny Images [Torralba et al., 2008], 32×32 pixel versions of image collected by performing web queries for nouns in the WordNet hierarchy [Fellbaum, 1998], without verifying the content. The other one is ImageNet, a large-scale ontology of images built upon the backbone of the WordNet structure. The images are also collected from web searches for the nouns in WordNet, but the content of images are verified by human labelers. ImageNet is much larger in scale and diversity

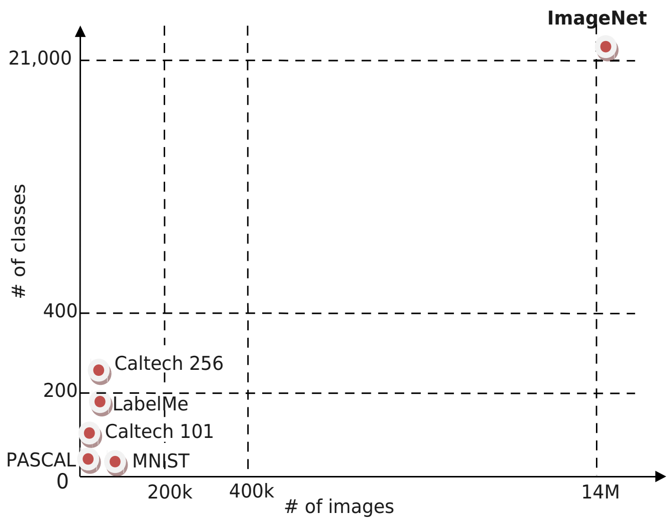


Fig. 2 A comparison of ImageNet with other benchmark datasets

and much more accurate than the current image datasets. The current released ImageNet has grown a big step in terms of the number of images and the number of classes, as shown in Fig. 2 - it has 21,841 classes with more than 1000 images for each class on average. Positively, it is necessary to have many images in the same class to cover visual variance, such as illumination, view point changes, and different appearance, even if in the dataset, some classes have only one or less than 10 images so machine learning algorithm cannot learn anything.

4 Low-Level Local Image Features

As shown in Fig. 1, given a set of input images, our system first extracts SIFT, SURF, and DSIFT features. These features have been proven to be efficient in various vision tasks such as object recognition, texture analysis, scene classification, etc.

4.1 SIFT

SIFT (Scale-invariant feature transform) is an algorithm proposed by [Lowe, 2004] to detect and describe local features in images. Extracting SIFTs consists of four key stages: scale-space extrema detection, keypoint localization, orientation assignment and keypoint descriptor. The first stage uses Difference-of-Gaussian function (DoG) to identify candidate interest points that are invariant to scale and orientation. DoG is used instead of Gaussian to speedup the computation.

In the keypoint localization stage, they reject the candidate points that have low contrast or are poorly localized along an edge. Hessian matrix is used to compute the principal curvatures and eliminate the keypoints that have a ratio between the principal curvatures greater than the threshold. An orientation histogram is formed from the gradient orientations of sample points within a region around the keypoint in order to get an orientation assignment. According to the paper's experiments, the best results are achieved with a 4×4 array of histograms with 8 orientation bins in each. So the SIFT descriptor used is $4 \times 4 \times 8 = 128$ dimensions.

4.2 SURF

SURF (Speeded Up Robust Feature) is a robust image detector and descriptor presented by [Bay et al., 2008]. The standard version of SURF is several times faster than SIFT and claimed by its authors to be more robust against different image transformations than SIFT. SURF is based on sums of 2D Haar wavelet responses and makes an efficient use of integral images.

SURF is partly inspired by the SIFT descriptor and has slightly different ways of detecting features. It uses an integer approximation to the determinant of Hessian blob detector, which can be computed extremely quickly with an integral image. For features, it uses the sum of the Haar wavelet response around the point of interest. Again, these can be computed with the aid of the integral image.

4.3 DSIFT

DSIFT is a variant of SIFT descriptors that is extracted at multiple scales. It is roughly equivalent to running SIFT on a dense grid of locations at a fixed scale and orientation. This type of feature descriptors is often used for object categorization.

- **Bin size vs. keypoint scale.** DSIFT specifies the descriptor size by a single parameter, size, which controls the size of a SIFT spatial bin in pixels. In the standard SIFT descriptor, the bin size is related to the SIFT keypoint scale by a multiplier, denoted magnif, which defaults to 3. As a consequence, a DSIFT descriptor with bin size equal to 5 corresponds to a SIFT keypoint of scale $5/3 = 1.66$.
- **Smoothing.** The SIFT descriptor smoothes the image according to the scale of the keypoints (Gaussian scale space). By default, the smoothing is equivalent to a convolution by a Gaussian of variance s^2 where s is the scale of the keypoint and 0.25 is a nominal adjustment that accounts for the smoothing induced by the camera CCD.

5 Classifiers

In various applications, kernel machines such as Support Vector Machines (SVM) have been used with impressive success often delivering state-of-the-art results. Using the kernel trick, they are applied in several domains and even enable heterogeneous data fusion by concatenating feature spaces or multiple kernel learning. Before performing image classification, we apply multi-feature and multi-codebook approach to construct the final image representation for all images in dataset. To stick to the efficient linear classifier, we use the explicit feature mapping approach from [Vedaldi and Zisserman, 2012] to improve the accuracy performance of image classification.

5.1 *Multi-feature and Multi-codebook*

As mentioned in section 3, the images in the same class of ImageNet usually have high intraclass variability. This variability poses more challenges for image classification systems. Many previous works want to design a robust image feature which is invariant to image transformation, illumination and scale change [Lowe, 2004; Bay et al., 2008; Bosch et al., 2007; Tola et al., 2010]. There are some improvements when using these robust features for image classification, but it is easy to realize that none of the feature descriptors have the same discriminative power for all classes. For instance, the features based on texture analysis and shape might be useful when classifying the photos with the same geometric direction. However, it will not be sufficient when the images are rotated or the objects are taken a shot in different camera angles. In this case, the appropriate choice should be the features based on interesting keypoints (e.g. SIFT). Obviously, instead of using a single feature type for all classes we can combine many different feature types to get higher improvement in classification accuracy. In this section, we present a novel multi-feature and multi-codebook approach and demonstrate how to combine these features.

Let a set of all different feature descriptor types extracted from an image i be $F = \{f_i^j\}$, where f_i^j are the descriptors of feature type j extracted from image i, M is the number of feature types, and $j = 1,..,M$. Our approach is that BoW histograms of each feature type are constructed based on their corresponding codebook, as shown in Fig. 3. Instead of using a single codebook for constructing the final image presentation, we use multiple codebooks $\{C^1, C^2, \ldots, C^M\}$ that are built from different feature types. More specifically, the codebook C^j is used to construct BoW histogram h_i^j for feature descriptors $f_i^j \in F$. Then all BoW histograms h_i^j are concatenated to form the final image representation H_i. As a result, for each image i, we obtain H_i with M elements $H_i = \{h_i^1, h_i^2, \ldots, h_i^M\}$. For simplicity, we call H_i a ¨bag- of- packets¨ (BoP) that is the final image representation constructed based on different codebooks of the original image i. A BoP is more discriminative than an usual BoW because two BoPs H_i and H_j are considered identical if and only if their corresponding BoWs are identical. Formally, it takes the intersection of the BoWs elements from multiple features:

$$(H_i = H_j) \equiv (h_i^1 = h_j^1) \wedge (h_i^2 = h_j^2) \wedge \ldots \wedge (h_i^M = h_j^M) \tag{1}$$

Obviously, this approach improves the discriminative power of the final image representation more than the classical approach with a single codebook.

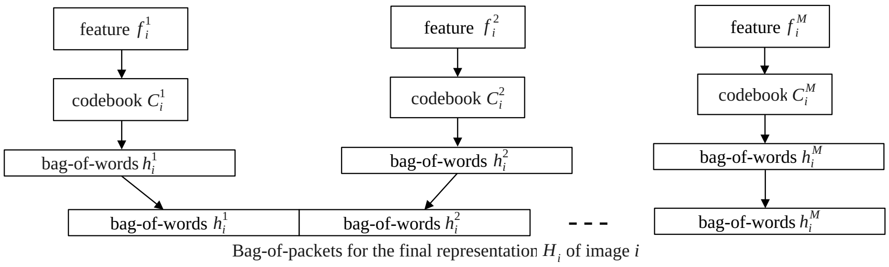


Fig. 3 Constructing bag-of-packets based on multi-feature and multi-codebook approach

5.2 Parallel LIBSVM (pLIBSVM)

LIBSVM is an integrated software for support vector classification, (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). It supports multi-class classification. Since version 2.8, it implements an SMO-type algorithm [Chang and Lin, 2001]. LIBSVM provides a simple interface where users can easily link it with their own programs. Main features of LIBSVM include: different SVM formulations, efficient multi-class classification, cross validation for model selection.

Keerthi *el al.* [Keerthi and Lin, 2003] present the theoretical proof that SVMs with RBF kernel and suitable parameters give at least as good accuracy as linear kernel. Yuan *et al.* [Yuan et al., 2012] show empirically that LIBSVM (RBF kernel) often offers better and more stable results than LIBLINEAR [Fan et al., 2008] on many benchmark datasets. However, the training cost of LIBSVM is too high in terms of computation time. It would take many days when performing on large scale datasets like ImageNet. Therefore, speedup the training process of LIBSVM become a very essential task in the context of large scale image classification.

In the multi-core era, computers with multi-cores or multiprocessors bring to us many advantages. Advanced technologies designed for the systems where several processing cores have access to a single memory space are becoming popular choice for high performance computing systems. OpenMP Application Program Interface (API) is a multi-platform shared-memory parallel programming model working on these systems [OpenMP Architecture Review Board, 2008]. It has been proven to work effectively on shared memory systems by the Board of OpenMP Architecture Review Board, 2008. Therefore, it motivates us to investigate parallel algorithms and demonstrate how LIBSVM can benefit from these modern platforms. In the original implementation of LIBSVM, computing kernel values in the matrices of various formulations is a very time-consuming step, especially when performing on datasets with a very large number of instances. Fortunately, the values in these matrices can be computed independently allowing to apply parallel algorithms. In this paper, we use OpenMP to parallelize this computation on a multi-core computer. With this modification, we significantly reduce the training time of LIBSVM.

To evaluate the performance of pLIBSVM, we compare it with LIBLINEAR and OCAS [Franc and Sonnenburg, 2008]. Franc *et al.* [Franc and Sonnenburg, 2008] have shown in their experiments that OCAS even in the early optimization steps shows often faster convergence than the so far in this domain prevailing approximative methods. So we chose OCAS to compare with our pLIBSVM instead of Pegasos (Primal Estimated sub-GrAdient SOlver for SVM) or SGD SVM (Stochastic Gradient Descent SVM).

6 Experiments and Results

6.1 Datasets

The full size of ImageNet dataset is 1TB. In the experiments, we evaluated our framework on 10 largest classes that contains 24,807 images with data size 2.4GB. The specific names of these classes are n00483313, n01882714, n02086240, n02087394, n02094433, n02100583, n02100735, n02138441, n02279972, n09428293. There are more than 2000 diversified images per class. In each class, we sample 90% of images for training and 10% of images for testing.

6.2 Parallel Feature Extraction

We perform our experiments on an Intel(R) Xeon(R) CPU E5345, 2.33GHz computer. Depending on parameters setting, the computation time of extracting feature (e.g. SIFT) of an image ranges from 0.46 to 1 second (single thread is used in computation). To process the 10 largest classes, it would take from 3 to 7 hours. Therefore, it is very difficult to scaleup to full ImageNet because if it takes 1 second per image for feature extraction then we need 14M $\times$ 1 second $\simeq$ 162 days. To deal with this challenge, we apply parallel solutions to reduce the computation time.

SIFT/DSIFT: VLFeat, a free version for extracting SIFTs, can be downloaded from the author's homepage (www.vlfeat.org). It has been developed by Andrea Vedaldi from the Vision Lab of the University of California. The original implementation of SIFT descriptors are integer vectors in 128 dimensions. There is no interdependence in feature extraction tasks, so we can extract features in parallel way. In this experiments, we use 8 CPU cores on our computer to extract features. As shown in Table 1, we need 56 minutes to extract more than 639M DSIFTs from the 10 largest classes. That means it takes 0.14 second to extract DSIFTs from an image on average. Therefore, with full ImageNet dataset, extracting DSIFTs would take 0.14s $\times$ 14M $\simeq$ 22 days.

Parallel SURF: Parallel SURF is a fast parallel version of SURF maintained by David Gossow [Gossow et al., 2010]. The local image descriptors extracted from original implementation of Parallel SURF are floating vectors in 64 dimensions. In this experiment, we also use 8 CPU cores for extracting feature. As shown in Table 1, we need 54 minutes to extract more than 47M SURFs from the 10 largest classes. That means it takes 0.13 second to extract SURFs from an image on average.

So, with full dataset, extracting SURFs will take $0.13s \times 14M \simeq 21$ days. Obviously, we can speedup the process of extracting features by using more resources (CPU cores, computer, etc.).

Table 1 Extract features from the 10 largest classes ImageNet using 8 CPU cores

Features	Time	# keypoints	Size
SIFT	17m46s	18,923,756	6GB
SURF	54m	47,308,685	24.4GB
DSIFT	56m	639,904,650	201.3GB

6.3 Fast Codebook Building

In BoW model, one of the steps that takes a long time is to build codebook. With a large scale dataset we need to get a large amount of datapoints to build a discriminative codebook, so this task becomes very large in terms of time complexity. One of the popular choices to build codebook is the k-means clustering algorithm. However, the original implementation of k-means takes many days to converge when performing on a large scale dataset like ImageNet. So reducing the execution time for this task is becoming an essential task when we study an efficient framework for large scale image classification. In this experiment, we have used the parallel version of k-means from Wei Dong [Dong,]. This program is a re-implementation of the k-means clustering algorithm. It has the following features:

1. An out-of-core k-means that allows clustering data larger than main memory,
2. Support parallel reading from multiple input files,
3. Accelerate L2 distance calculation with BLAS or KD-tree.

To perform the k-means algorithm, we use 8 CPU cores on the same computer as in section 6.2. We sample all visual descriptors of the images in training dataset to build codebooks with 5,000 codewords. We set the maximum iteration of the k-means to 40 and the convergence threshold to 0.001. By using parallel k-means, we build codebooks from very large datasets in reasonable time, as shown in Table 2.

Table 2 Parallel k-means on the 10 largest classes ImageNet using 8 CPU cores

Features	# datapoints	Dimension	Size	Time
SIFT	17,032,522	128	5.4GB	5h21m
SURF	42,610,816	64	22GB	4h03m
DSIFT	575,790,745	128	181.2GB	8 days

6.4 Parallel Bag-of-Packets Constructing

To speedup the construction of BoW histograms of images, we take into account the implementation of randomized kd-tree forests from VLFeat toolbox. It not only improves the effectiveness of the representation in high dimensions, but enables fast medium and large scale nearest neighbor queries among high dimensional data points. Once k-means is performed, we build a hierarchical structure for codebooks by using *vl_kdtreebuild*. By this way, we can use *vl_kdtreequery* to speedup the process of mapping visual descriptors to visual words (or codewords). The computation time for applying each codebook is similar to classical approach (single codebook). When we use n codebooks for constructing BoP of images, it means we need n more times to finish this process. To achieve the same computation time as single codebook approach, we perform the process of constructing BoP in a parallel way. Consequently, the whole computation time of this process is the same as the largest individual standard approach. As shown in Table 3 and 4, we can reduce the computation time for constructing BoP of DSIFT+SURF+SIFT with multi-codebooks to the same amount of time that the one of DSIFT with a single codebook.

Table 3 Parallelize bag-of-packets construction using 8 CPU cores. The image representation is normalized by L2-Norm.

Features	Dimension	Time	Size
SIFT	5,000	3m18s	179MB
SURF	5,000	7m48s	320MB
DSIFT	5,000	1h01s	934MB
DSIFT+SURF	10,000	1h02s	1.2GB
DSIFT+SURF+SIFT	15,000	1h05s	1.5GB

Table 4 Parallelize bag-of-packets construction using 8 CPU cores. The image representation is converted to high-dimensional space by using homogeneous kernel map.

Features	Dimension	Time	Size
SIFT	15,000	3m06s	560MB
SURF	15,000	7m02s	1GB
DSIFT	15,000	58m03s	2.9GB
DSIFT+SURF	30,000	59m01s	4GB
DSIFT+SURF+SIFT	45,000	1h05s	4.5GB

6.5 Classification Accuracy

The linear kernel on the classical histogram based feature gives very poor accuracy on image classification. Therefore, once BoW histogram is constructed, some recent image classification systems use feature map to convert BoW histogram from linear space to non-linear space. This step is useful when one want to stick to the efficient linear classifiers [Chatfield et al., 2011]. The result is the image representation in high-dimensional space, that ensures linear separability of the classes. Notice that before training classifiers, we should normalize BoW histograms, so that the image size does not influence histogram counts. The popular normalization methods used in recent image classification systems are L1-Norm and L2-Norm:

$$L1-norm: f(x) = \frac{x}{||x||_1} = \frac{x}{\sum\limits_{i=1}^{N} |x_i|} \quad (2)$$

$$L2-Norm: f(x) = \frac{x}{||x||_2} = \frac{x}{\sqrt{\sum\limits_{i=1}^{N} |x_i|^2}} \quad (3)$$

In the experiments, we want to evaluate our approach in two different cases. In the first case, we use L2-Norm to normalize BoW histograms of all images in dataset, as shown in Table 5 and 6. In the second one, we use L1-Norm to normalize BoW histograms and then the final image representation is converted to high-dimensional space by using homogeneous kernel map from Vedaldi, as shown in Table 7 and 8. By using this feature map, we obtain a significant improvement in image classification accuracy (from +6.24% to +16.93% with different feature types).

Multi-feature and multi-codebook. To evaluate the performance of multi-feature and multi-codebook approach on the ten largest classes from ImageNet, we perform the experiments for each single feature SIFT, SURF and DSIFT. Then we perform classification by using simultaneously different feature types DSIFT+SURF and DSIFT+SURF+SIFT. As shown in Fig. 4, in the case of training LIBSVM (RBF kernel) on the combination of three different feature types, we significantly improve the performance of overall classification accuracy up to 1.82 times, compared to single feature SIFT (Table 5). The picture of the improvements is the same when we use homogeneous kernel map (Table 7 and 8).

Parallel LIBSVM. To evaluate the performance of our pLIBSVM, we compare it with LIBLINEAR, OCAS, and the original implementation of LIBSVM (a non-parallel version) in terms of both classification accuracy and training time. In the experiments, we use 8 CPU cores on the same computer as in section 6.2. We also evaluate our implementation with different SVM (linear kernel and RBF kernel). In the case of training pLIBSVM with RBF kernel, we use cross validation on training data to find the best parameters C and gamma of SVM classifiers.

As aforementioned, the major challenge of large scale image classification is on training classifiers. This paper proposed a parallel version of LIBSVM that was efficient on the ten largest classes of ImageNet. As shown in Fig. 5, in the case of

Table 5 Overall classification accuracy and training time. The image representation is normalized by L2-Norm. Training LIBSVM with linear kernel.

Features	LIBLINEAR	OCAS	LIBSVM	pLIBSVM
SIFT	34.91% (3m05s)	38.94% (28m39s)	**46.31%** (45m55s)	**46.31%** (8m57s)
SURF	35.79% (3m34s)	44.7% (43m40s)	**50.83%** (1h13m)	**50.83%** (13m36s)
DSIFT	52.64% (11m52s)	59.09% (1h23m)	**64.57%** (1h50m)	**64.57%** (17m20s)
DSIFT+SURF	60.42% (15m06s)	64.05% (1h52m)	**67.63%** (3h26m)	**67.63%** (31m46s)
DSIFT+SURF+SIFT	63.80% (23m03s)	65.70% (4h45m)	**70.09%** (4h58m)	**70.09%** (43m27s)

Table 6 Overall classification accuracy and training time. The image representation is normalized by L2-Norm. Training LIBSVM with RBF kernel.

Features	LIBSVM	pLIBSVM	Accuracy
SIFT	1h01m	7m33s	50.42%
SURF	2h00m	15m31s	56.47%
DSIFT	3h01m	19m02s	68.36%
DSIFT+SURF	5h48m	37m10s	70.25%
DSIFT+SURF+SIFT	6h03m	47m44s	**71.46%**

Table 7 Overall classification accuracy and training time. The image representation is converted to high-dimensional space by using homogeneous kernel map. Training LIBSVM with linear kernel.

Features	LIBLINEAR	OCAS	LIBSVM	pLIBSVM
SIFT	41.15% (3m58s)	43.62% (55m31s)	**47.52%** (1h21m)	**47.52%** (16m41s)
SURF	47.84% (6m30s)	50.63% (1h23m)	**55.62%** (2h17m)	**55.62%** (25m19s)
DSIFT	69.57% (15m05s)	72.07% (3h07m)	**74.32%** (3h42m)	**74.32%** (43m44s)
DSIFT+SURF	71.22% (49m59s)	72.72% (6h53m)	**75.17%** (5h15m)	**75.17%** (1h08m)
DSIFT+SURF+SIFT	72.95% (1h30m)	73.97% (20h39m)	**75.98%** (5h50m)	**75.98%** (1h17m)

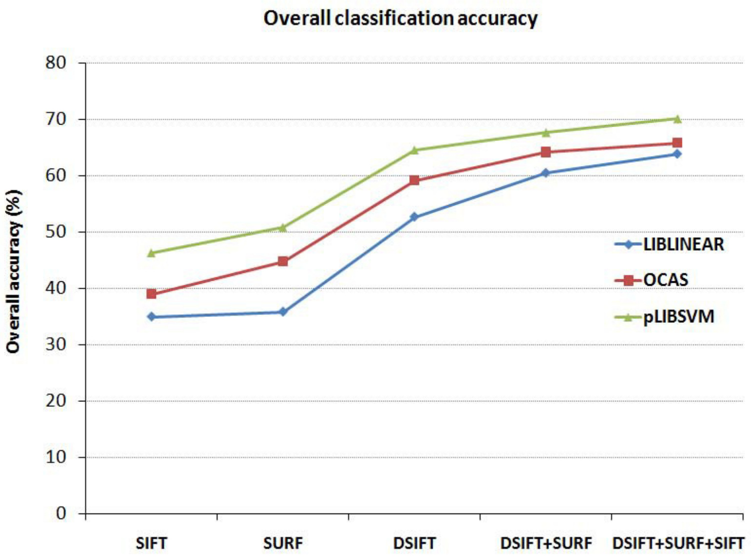


Fig. 4 Overall accuracy of SVM classifiers with different feature types

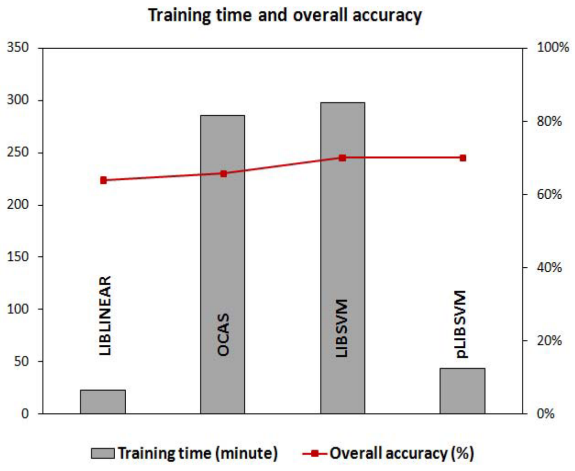


Fig. 5 Overall accuracy and training time of SVM classifiers

the combination of three different feature types (DSIFT+SURF+SIFT), the accuracy performance of pLIBSVM and LIBSVM are higher than the other classifiers from +4.39% to +6.29% (Table 5). Table 6 to 8 also show a high performance of pLIBSVM and LIBSVM on all different feature types. Furthermore, in the case of training SVM classifiers with RBF kernel, pLIBSVM achieves better results than those of linear kernel classifiers, as shown in Table 6 and 8.

About the training time, all the results presented in Table 5 to 8 are user time. We have chosen this instead of cpu time because cpu time is the sum of all threads cpu time when we use multi-threading, so we could not show the improvement of our approach this way. We know the drawback of user time: we cannot be sure we are the only user during the whole execution process. As shown in Fig. 5, we can see we significantly speedup the training time of classifiers with our approach. Particularly, our pLIBSVM use 43 minutes with 8 CPU cores to train classifiers, compared to 4 hours 58 minutes of LIBSVM and 4 hours 45 minutes of OCAS (Table 5). The accuracy of all the linear kernel classifiers is increased when we transform the data in a high-dimensional space by using homogeneous kernel map (Table 7), but the training time is increased too, due to the higher dimension of the input space. Finally the best result is obtained by using the same transformation and RBF kernel with LIBSVM/pLIBSVM as shown in Table 8. The time reported here is with only 8 CPU cores and the 10 largest classes from ImageNet, of course the training time can easily be reduced by using more resources (CPUs, cores, computers). This is what we plan to do for the 1000 largest classes of the same dataset.

Table 8 Overall classification accuracy and training time. The image representation is converted to high-dimensional space by using homogeneous kernel map. Training LIBSVM with RBF kernel.

Features	LIBSVM	pLIBSVM	Accuracy
SIFT	3h17m	19m22s	51.47%
SURF	4h54m	29m51s	58.57%
DSIFT	7h31m	53m11s	76.86%
DSIFT+SURF	8h51m	1h19m	77.03%
DSIFT+SURF+SIFT	9h50m	1h33m	**78.15%**

Among the different GPU-based approaches, the only one that could be used for very large datasets is the incremental SVM with CUDA [Poulet and Pham, 2010]. This method is 3 to 4 orders of magnitude faster than usual classification algorithms like libSVM, but it is only a linear kernel so we know the accuracy will be less than the one with RBF kernel. To the best of our knowledge, the other GPU-based SVMs with non linear kernel require to load the whole dataset into main memory. Most of the GPU architectures today have up to 6 GB memory size. Here with only 5k vocabulary size and the 10 largest classes from ImageNet, we already need 4.5GB for the data and the 1000 largest classes from the same dataset require 25GB memory so no GPU-based SVM can be used in this context.

7 Conclusion and Future Work

We have proposed a fast and efficient framework for large scale image classification and show how to address this challenge by using ImageNet dataset as an example. In this framework, we have developed a parallel version of LIBSVM to efficiently deal with very large datasets in reasonable time. To speedup the process of extracting features, we have presented how to use a multi-core computer to reduce the computation time of feature extraction. We have also presented a novel approach using several different local features simultaneously to improve the classification accuracy on a large scale image dataset (the relative increase is up to 82%). In the near future, we plan to study how to combine effectively the global features (e.g. contour, texture, shape, etc.) with the local features to get more discriminative power of image representation. About the computation time our approach allows us to get the classification results with a RBF kernel in almost the same time as with usual algorithms and linear kernel by using only 8 cores. The next step is the classification of the 1000 largest classes of ImageNet (more than 1.4 million images). Furthermore, the current version of libOCAS only offer parallel version of the binary solver, so we intend to parallelize it for multi-class problem. That will be a promising research for large scale image classification. Encoding spatial information of the interesting keypoints of image will be also studied.

Acknowledgements. This work was partially funded by Region Bretagne (France).

References

[Bay et al., 2008] Bay, H., Ess, A., Tuytelaars, T., Gool, L.J.V.: Speeded-up robust features (surf). Computer Vision and Image Understanding 110(3), 346–359 (2008)

[Bosch et al., 2007] Bosch, A., Zisserman, A., Muñoz, X.: Image classification using random forests and ferns. In: International Conference on Computer Vision, pp. 1–8 (2007)

[Chang and Lin, 2001] Chang, C.C., Lin, C.J.: LIBSVM – a library for support vector machines (2001), `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

[Chatfield et al., 2011] Chatfield, K., Lempitsky, V., Vedaldi, A., Zisserman, A.: The devil is in the details: an evaluation of recent feature encoding methods. In: British Machine Vision Conference, pp. 76.1–76.12 (2011)

[Csurka et al., 2004] Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: Workshop on Statistical Learning in Computer Vision, ECCV, pp. 1–22 (2004)

[Dalal and Triggs, 2005] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 886–893. IEEE Computer Society (2005)

[Deng et al., 2010] Deng, J., Berg, A.C., Li, K., Fei-Fei, L.: What does classifying more than 10, 000 image categories tell us? In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 71–84. Springer, Heidelberg (2010)

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Li, F.-F.: Imagenet: A large-scale hierarchical image database. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009)

[Dong,] Dong, W.: A parallel out-of-core k-means clusterer, `http://www.cs.princeton.edu/~wdong/kmeans`

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision 88(2), 303–338 (2010)

[Fan et al., 2008] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., Lin, C.-J.: Liblinear: A library for large linear classification. Journal of Machine Learning Research 9, 1871–1874 (2008)

[Fellbaum, 1998] Fellbaum, C.: WordNet: An Electronic Lexical Database. Bradford Books (1998)

[Fergus et al., 2009] Fergus, R., Weiss, Y., Torralba, A.: Semi-supervised learning in gigantic image collections. In: Advances in Neural Information Processing Systems, pp. 522–530 (2009)

[Franc and Sonnenburg, 2008] Franc, V., Sonnenburg, S.: Optimized cutting plane algorithm for support vector machines. In: International Conference on Machine Learning, pp. 320–327 (2008)

[Gossow et al., 2010] Gossow, D., Decker, P., Paulus, D.: An evaluation of open source surf implementations. In: Ruiz-del-Solar, J. (ed.) RoboCup 2010. LNCS (LNAI), vol. 6556, pp. 169–179. Springer, Heidelberg (2010)

[Griffin et al., 2007] Griffin, G., Holub, A., Perona, P.: Caltech-256 Object Category Dataset. Technical Report CNS-TR-2007-001, California Institute of Technology (2007)

[Keerthi and Lin, 2003] Keerthi, S.S., Lin, C.-J.: Asymptotic behaviors of support vector machines with gaussian kernel. Neural Computation 15(7), 1667–1689 (2003)

[Lazebnik et al., 2006] Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 2169–2178 (2006)

[Li et al., 2007] Li, F.-F., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. Computer Vision and Image Understanding 106(1), 59–70 (2007)

[Li et al., 2009] Li, Y., Crandall, D.J., Huttenlocher, D.P.: Landmark classification in large-scale image collections. In: IEEE 12th International Conference on Computer Vision, pp. 1957–1964. IEEE (2009)

[Lowe, 2004] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)

[Moosmann et al., 2006] Moosmann, F., Triggs, B., Jurie, F.: Fast discriminative visual codebooks using randomized clustering forests. In: Advances in Neural Information Processing Systems, pp. 985–992 (2006)

[OpenMP Architecture Review Board, 2008] OpenMP Architecture Review Board. OpenMP application program interface version 3.0 (2008)

[Perronnin et al., 2010] Perronnin, F., Sánchez, J., Liu, Y.: Large-scale image categorization with explicit data embedding. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 2297–2304 (2010)

[Philbin et al., 2008] Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Lost in quantization: Improving particular object retrieval in large scale image databases. In: IEEE Conference on Computer Vision and Pattern Recognition (2008)

[Poulet and Pham, 2010] Poulet, F., Pham, N.-K.: High dimensional image categorization. In: Cao, L., Feng, Y., Zhong, J. (eds.) ADMA 2010, Part I. LNCS, vol. 6440, pp. 465–476. Springer, Heidelberg (2010)

[Tola et al., 2010] Tola, E., Lepetit, V., Fua, P.: Daisy: An efficient dense descriptor applied to wide-baseline stereo. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(5), 815–830 (2010)

[Torralba et al., 2008] Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: A large data set for nonparametric object and scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(11), 1958–1970 (2008)

[Vedaldi et al., 2009] Vedaldi, A., Gulshan, V., Varma, M., Zisserman, A.: Multiple kernels for object detection. In: IEEE 12th International Conference on Computer Vision, pp. 606–613. IEEE (2009)

[Vedaldi and Zisserman, 2012] Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(3), 480–492 (2012)

[Wang et al., 2009] Wang, C., Yan, S., Zhang, H.-J.: Large scale natural image classification by sparsity exploration. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 3709–3712. IEEE (2009)

[Winder and Brown, 2007] Winder, S.A.J., Brown, M.: Learning local image descriptors. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2007)

[Yuan et al., 2012] Yuan, G.-X., Ho, C.-H., Lin, C.-J.: Recent advances of large-scale linear classification. Proceedings of the IEEE 100(9), 2584–2603 (2012)