Large Scale Visual Classification with Parallel, Imbalanced Bagging of Incremental LIBLINEAR SVM

Thanh-Nghi Doan[1], **Thanh-Nghi Do**[2], **and François Poulet**[1,3]
[1]IRISA, [3]Université de Rennes 1, Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France
[2]Institut Telecom, Telecom Bretagne, UMR CNRS 6285 Lab-STICC,
Université européenne de Bretagne, France, Can Tho University, Vietnam

Abstract—*ImageNet dataset with more than 14M images and 21K classes makes the problem of visual classification more difficult to deal with. One of the most difficult tasks is to train a fast and accurate classifier. In this paper, we address this challenge by extending the state-of-the-art large scale linear classifier LIBLINEAR-CDBLOCK proposed by Hsiang-Fu Yu in three ways: (1) improve LIBLINEAR-CDBLOCK for large number of classes with one-versus-all approach, (2) a balanced bagging algorithm for training binary classifiers, (3) parallelize the training process of classifiers with several multi-core computers. Our approach is evaluated on the 100 largest classes of ImageNet and ILSVRC 2010. The evaluation shows that our approach is 732 times faster than the original implementation and 1193 times faster than LIBLINEAR without (or very few) compromising classification accuracy.*

Keywords: Support Vector Machines, Incremental Learning Method, Balanced Bagging, High Performance Computing

1. Introduction

Visual classification is one of the important topics in computer vision and machine learning. The usual frameworks involve three steps: 1) extracting local image features, 2) building codebook and encoding features, and 3) training classifiers. These frameworks are evaluated on small datasets, e.g. Caltech 101 [1], Caltech 256 [2] and PASCAL VOC [3]. In step 3, most researchers choose either linear or nonlinear SVM classifiers that can be trained in a few minutes.

However, ImageNet dataset [4] with very large number of classes poses more challenges in training classifiers. ImageNet is much larger in scale and diversity than other benchmark 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. 1 - it has 21,841 classes with more than 1000 images for each class on average.

With millions of training examples or dimensions, training an accurate classifier may take weeks or even years [5], [6]. Recent works in large scale learning classifiers converge on building linear SVM classifiers, because it is possible to train linear classifiers (e.g. LIBLINEAR [7]) in order of seconds, even with millions training examples. However, when training data is larger and cannot fit into main memory, most existing linear classifiers encounter a problem. Yu [8] proposed a block minimization framework for linear classifier (LIBLINEAR-CDBLOCK), that can be applied to data beyond the memory capacity of computer. They show empirically that their method can handle data sets 20 times larger than the memory size. However, the current version of LIBLINEAR-CDBLOCK has two main drawbacks that prevent it scaleup to large scale dataset with many classes. Firstly, LIBLINEAR-CDBLOCK has not explored one-versus-all approach in the case of multi-class classification. Secondly, it does not take into account the benefits of high performance computing (HPC). On the dataset of ImageNet Challenge 2010 (ILSVRC 2010 [9]), it takes very long time to train classifiers. Therefore, it motivates us to study how to extend LIBLINEAR-CDBLOCK for large scale visual classification. Our key contributions include:

1. Improve LIBLINEAR-CDBLOCK for large number of classes by using one-versus-all approach.

2. Propose a balanced bagging algorithm for training the binary classifiers. Our algorithm avoids training on full data, and the training process of LIBLINEAR-CDBLOCK rapidly converges to the optimal solution.

3. Parallelize the training process of all binary classifiers based on HPC models. In the training step of classifiers, we apply our balanced bagging algorithm to achieve the best performance.

Our approach is evaluated on the 100 largest classes of ImageNet and ILSVRC 2010. The experiment shows that our approach is 732 times faster than the original implementation and 1193 times faster than LIBLINEAR without (or very few) compromising classification accuracy. Therefore, it can be easily applied to datasets with very large number of classes and the training data cannot fit into the memory of computer.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related work on large scale visual classification. The incremental LIBLINEAR support vector machines is described in section 3. Section 4 presents its improvement for large number of classes. We describe how to speedup the training process of incremental LIBLINEAR by using balanced bagging algorithm and take the benefits of HPC. Section 5 presents numerical results before the conclusion and future work.

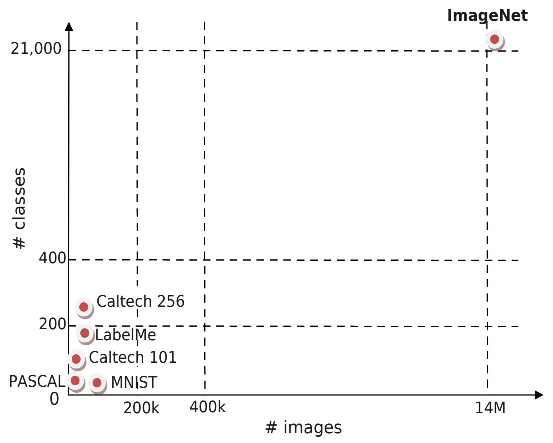

Fig. 1: A comparison of ImageNet with other benchmark datasets.

2. Related Work

Low-level local image features, bag-of-words model (BoW [10]) and support vector machines (SVM [11]) are the core of state-of-the-art visual classification systems. These may be enhanced by multi-scale spatial pyramids [12] on BoWs or histogram of oriented gradient [13] features. Some recent works consider exploiting the hierarchical structure of dataset for image recognition and achieve impressive improvements in accuracy and efficiency [14]. Related to classification is the problem of detection, often treated as repeated one-versus-all classification in sliding windows [15], [3]. In many cases, such localization of objects might be useful to improve classification accuracy performance. However, in the context of large scale visual classification with hundreds or thousands of classes, these common approaches become computationally intractable.

To address this problem, Fergus *et al.* [16] study semi-supervised learning on 126 hand labeled Tiny Images categories, Wang *et al.* [17] show classification experiments on a maximum of 315 categories. Li *et al.* [18] do research with landmark classification on a collection of 500 landmarks and 2 million images. On a small subset of 10 classes, they have improved BoWs classification by increasing the visual vocabulary up to 80K visual words.

The emergence of ImageNet makes the complexity of visual classification much larger and very difficult to deal with. Recently, many researchers are beginning to study strategies to improve the classification accuracy and avoid using high cost nonlinear kernel SVM classifiers. The prominent works are proposed in [5], [6], [19], [20] where the data are first transformed by a nonlinear mapping induced by a particular kernel and then linear classifier is trained in the resulting space. They argue that the classification accuracy of linear classifier with high-dimensional image signature is similar to low-dimensional BoW with nonlinear classifier.

In [6], the winner of ImageNet Challenge 2010, each local descriptor is encoded by using either Local Coordinate Coding [21] or Super-vector Coding [22]. Then, they perform spatial pyramid pooling and the resulting image signature is a vector in approximately 262K dimensions. To train classifiers, they propose a parallel averaging stochastic gradient descent (ASGD) algorithm. With 1K classes of ILSVRC 2010, it takes 4 days to train 1K binary SVM classifiers (one-versus-all) for one feature channel on three 8-core computers. However, their method involves training classifiers on a dataset in hundreds of giga-bytes. Therefore, it cannot be easily applied to the systems with limited memory resource. To tackle this challenge, Yu *et al.* [8] propose LIBLINEAR-CDBLOCK that can handle data larger than the memory size of computer. Nevertheless, LIBLINEAR-CDBLOCK has two main limitations: 1) multi-class classification with one-versus-all approach has not been explored, 2) it does not take into account the benefits of HPC. Therefore, the training time is very long on ILSVRC 2010 (at least 32 hours) due to learning 1K binary classifiers sequentially, independently.

3. Incremental LIBLINEAR Support Vector Machines

Let us consider a linear binary classification task with a training set $\mathbb{T} = \{(x_i, y_i)\}_{i=1}^n, x_i \in \mathbb{R}^d, y_i \in \{+1, -1\}$. SVM classification algorithm aims to find the best separating surface as being furthest from both classes. It can simultaneously maximize the margin between the supporting planes for each class and minimize the errors. This can be performed by solving the dual optimization problem (1).

$$\begin{aligned} &\min_{\alpha \in \mathbb{R}^n} f(\alpha) = \frac{1}{2}\alpha^T Q \alpha - e^T \alpha \\ &s.t. \left\{ \begin{array}{l} y^T \alpha = 0 \\ 0 \le \alpha_i \le C, \ \forall i = 1, 2, ..., n \end{array} \right. \end{aligned} \tag{1}$$

where $e = [1, \dots, 1]^T$, C is a positive constant used to tune the margin and the error, $\alpha = (\alpha_1, \dots, \alpha_n)$ are the Lagrange multipliers, Q is an $n \times n$ symmetric matrix with $Q_{ij} = y_i y_j K\langle x_i, x_j\rangle$, and $K\langle x_i, x_j\rangle$ is the kernel function.

The support vectors (for which $\alpha_i > 0$) are given by the optimal solution of (1), and then, the separating surface and the scalar b are determined by the support vectors. The classification of a new data point x is based on:

$$sign(\sum_{i=1}^{\#SV} y_i \alpha_i K\langle x, x_i\rangle - b) \tag{2}$$

Variations on SVM algorithms use different classification functions. No algorithmic changes are required from the usual kernel function K as a linear inner product other than the modification of the kernel evaluation, including a polynomial function of degree d, a RBF (Radial Basis Function) or a sigmoid function. We can get different support vector classification models.

LIBLINEAR proposed by [7] uses a dual coordinate descent method for dealing with linear SVM using L1- and L2-loss functions. And then, LIBLINEAR is simple and reaches an ϵ-accurate solution in O(log(1/ϵ)) iterations. The algorithm is much faster than state of the art solvers such as LibSVM [23] or SVM$^{\text{perf}}$ [24].

Most SVM algorithms are designed by assuming that data can be stored in the main memory. Therefore, in the context of large scale classification, these approaches become intractable. To solve this problem, [25] and [8] propose incremental learning methods for solving the memory usage problem of linear classifiers. They show that training SVM classifiers can be performed on the successive subsets of the training set.

Let $\{B_j\}_{j=1}^{m}$ be a fixed partition of $\mathbb{T}$ into m blocks of rows. These blocks of rows are disjoint sets stored in m separate files. At each iteration, we consider a block of rows B_j and solve the problem (1) only for the samples in B_j, so the algorithm does not need to keep in memory the samples from other blocks of rows. According to memory size, we choose block size such that the samples in B_j can fit into memory. LIBLINEAR is used to solve the sub-problems and the solution is updated in growing training data without loading the entire data into memory at once. The incremental learning for LIBLINEAR is summarized in Algorithm 1.

Algorithm 1: Incremental learning for LIBLINEAR

```
  input : A set of training samples $\mathbb{T} = \{(x_i, y_i)\}_{i=1}^{n}$
  output: The values $\alpha$ or $w$
1 Split $\mathbb{T}$ into $B_1, ..., B_m$ and store data in m files accordingly
2 $\alpha \leftarrow 0$ or $w \leftarrow 0$
3 for $j \leftarrow 1$ to $m$ do
4     Read $x_r \in B_j$ from disk
5     Solve the sub-problem (3) by using LIBLINEAR
6     Update $\alpha$ or $w$
7 end
```

Solving dual SVM by LIBLINEAR for each block. The optimal solution of (1) can be obtained by solving the sub-problems (3).

$$\min_{d \in \mathbb{R}^n} f(\alpha + d) = \frac{1}{2}(\alpha + d)^T Q(\alpha + d) - e^T(\alpha + d)$$

$$s.t. \begin{cases} d_i = 0, \forall i \notin B_j \\ 0 \leq \alpha_i + d_i \leq C \ \ , \ \ \forall i \in B_j. \end{cases} \tag{3}$$

Let d_{B_j} be a vector of $|B_j|$ non-zero coordinates of d that correspond to the indices in B_j. The objective (3) is equivalent to

$$\frac{1}{2} d_{B_j}^T Q_{B_j B_j} d_{B_j} + (Q_{B_j,\bullet}\alpha - e_{B_j})^T d_{B_j} \ \ , \tag{4}$$

where $Q_{B_j,\bullet}$ is a sub-matrix of Q including elements $Q_{ri}, r \in B_j, i = 1, \ldots, n$. Obviously, $Q_{B_j,\bullet}$ in Eq. 4 involves all training data. This violate the method presented in Algorithm 1. However, by maintaining $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ into memory, we can compute $Q_{B_j,\bullet}$ by using the Eq. 5.

$$Q_{B_j,\bullet} - 1 = y_r w^T x_r - 1, \forall r \in B_j \tag{5}$$

where $w \leftarrow w + \sum_{r \in B_j} d_r^* y_r x_r$
This operation involves only the samples in B_j.

4. Improving incremental LIBLINEAR for large number of classes

Most SVM algorithms are only able to deal with a two-class problem. There are several extensions of binary classification SVM solver to multi-class (k classes, $k \geq 3$) classification tasks. The state-of-the-art multi-class SVMs are categorized into two types of approaches. The first one is to consider the multi-class case in an optimization problem [26], [27]. The second one is to decompose multi-class into a series of binary SVMs, including one-versus-all [11], one-versus-one [28] and Decision Directed Acyclic Graph [29]. Recently, hierarchical methods for multi-class SVM [30], [31] start from the whole data set, hierarchically divide the data into two subsets until every subset consists of only one class.

In practice, one-versus-all, one-versus-one are the most popular methods due to their simplicity. Let us consider k classes ($k > 2$). The one-versus-all strategy builds k different classifiers where the i^{th} classifier separates the i^{th} class from the rest. The one-versus-one strategy constructs $k(k-1)/2$ classifiers, using all the binary pairwise combinations of the k classes. The class is then predicted with a majority vote.

When dealing with very large number of classes, e.g. hundreds of classes, the one-versus-one strategy is too expensive because it needs to train many thousands of classifiers. Therefore, the one-versus-all strategy becomes popular in this case.

However, for multi-class classification, LIBLINEAR-CDBLOCK solves a single optimization problem by using [32]. Therefore, the current version of LIBLINEAR-CDBLOCK needs very long time to classify very large number of classes.

Due to this problem, we propose three ways for speedup the learning task of LIBLINEAR-CDBLOCK. The first one is to implement one-versus-all approach for multi-class case. The second one is to build the balanced bagging classifiers with sampling strategy. Finally, we parallelize the training task of all classifiers with several multi-core computers.

Balanced bagging incremental LIBLINEAR

In the one-versus-all approach, the learning task of incremental LIBLINEAR SVM is to try to separate the i^{th} class (positive class) from the $k-1$ other classes (negative class). For very large number of classes, e.g. 1000 classes, this leads to the extreme imbalance between the positive class and the negative class. The problem is well-known as the class imbalance. As summarized by the review papers [33], [34] and the very comprehensive papers [35], [36], solutions to the class imbalance problems were proposed both at the data and algorithmic level. At the data level, these algorithms change the class distribution, including over-sampling the minority class or under-sampling the majority class. At the algorithmic level, the solution is to re-balance the error rate by weighting each type of error with the corresponding cost. Our balanced bagging incremental LIBLINEAR SVM belongs to the first approach (forms of re-sampling). Furthermore, the class prior probabilities in this context are highly unequal (e.g. the distribution of the positive class is 0.1% in the 1000 classes classification problem), and over-sampling the minority class is very expensive. We propose the balanced bagging incremental LIBLINEAR SVM using under-sampling the majority class (negative class).

For separating the i^{th} class (positive class) from the rest (negative class), the balanced bagging incremental LIBLINEAR SVM trains T models as shown in algorithm 2.

Algorithm 2: Balanced bagging incremental LIBLINEAR SVM

```
  input : $B_+$ the training data of positive class in $B_j$
          $B_-$ the training data of negative class in $B_j$
          $T$ the number of base learners
  output: LIBLINEAR SVM model
1 Learn:
2 for $k \leftarrow 1$ to $T$ do
3     1. $B'_-$ = sample($B_-$) (with $|B'_-|$ = $|B_+|$)
4     2. LIBLINEAR($B_+$, $B'_-$)
5 end
6 combine $T$ models into the aggregated
  LIBLINEAR SVM model
```

We remark that the margin can be seen as the minimum distance between two convex hulls, H_+ of the positive class and H_- of the negative class (the farthest distance between the two classes). Under-sampling the negative class (B'_-) done by balanced bagging provides the reduced convex hull of H_-, called H'_-. And then, the minimum distance between H_+ and H'_- is larger than between H_+ and H_- (full dataset). It is easier to achieve the largest margin than learning on the full dataset. Therefore, the training task of incremental LIBLINEAR SVM is fast to converge to the solution. According to our experiments, by setting $T = \sqrt{\frac{|B_-|}{|B_+|}}$, the balanced bagging incremental LIBLINEAR SVM achieves good results in very fast training speed.

Parallel incremental LIBLINEAR training

Although the incremental LIBLINEAR SVM and balanced bagging incremental LIBLINEAR SVM deal with very large dataset with high speed, they do not take into account the benefits of HPC, e.g. multi-core computers. Furthermore, both incremental LIBLINEAR SVM and balanced bagging incremental LIBLINEAR SVM train independently k binary classifiers for k classes problems. This is a nice property for parallel learning. Our investigation aims to speedup the training task of multi-class incremental LIBLINEAR SVM and balanced bagging incremental LIBLINEAR SVM with several multi-processor computers. The idea is to learn k binary classifiers in parallel way.

The parallel programming is currently based on two major models, Message Passing Interface (MPI) [37] and Open Multiprocessing (OpenMP) [38]. MPI is a standardized and portable message-passing mechanism for distributed memory systems. MPI remains the dominant model (high performance, scalability, and portability) used in high-performance computing today. However, MPI process loads the whole subset (block) into memory during learning tasks, making it wasteful. The simplest development of parallel incremental LIBLINEAR SVM algorithms is based on the shared memory multiprocessing programming model OpenMP. However, OpenMP is not guaranteed to make the most efficient computing. Finally, we present a hybrid approach that combines the benefits from both OpenMP and MPI models. The hybrid MPI/OpenMP parallel incremental LIBLINEAR SVM algorithm is described in algorithm 3. The number of MPI processes depends on the memory capacity of the HPC system used.

Algorithm 3: Hybird MPI/OpenMP parallel incremental LIBLINEAR SVM

```
   input : A set of training samples $\mathbb{T} = \{(x_i, y_i)\}_{i=1}^n$
           $P$ the number of MPI processes
   output: The value $\alpha$ or $w$
 1 Split $\mathbb{T}$ into $B_1, ..., B_m$ and store data in m files
   accordingly
 2 $\alpha^t \leftarrow 0, w^t \leftarrow 0,\ 1 \le t \le k$
 3 for $j \leftarrow 1$ to $m$ do
 4     Read $x_r \in B_j$ from disk        /* block $j$ */
 5     Learn:
 6     $MPI-PROC_1$
 7     #pragma omp parallel for
 8     for $t_1 \leftarrow 1$ to $k_1$ do         /* class $t_1$ */
 9         LIBLINEAR ($B_j^{t_1}, B_j \setminus B_j^{t_1}$)
10         Update $\alpha^{t_1}$ and $w^{t_1}$
11     end
12     :
13     $MPI-PROC_P$
14     #pragma omp parallel for
15     for $t_P \leftarrow 1$ to $k_P$ do         /* class $t_P$ */
16         LIBLINEAR ($B_j^{t_P}, B_j \setminus B_j^{t_P}$)
17         Update $\alpha^{t_P}$ and $w^{t_P}$
18     end
19 end
```

5. Experiments and Results

In this section we compare our implementation with LIBLINEAR-CDBLOCK and LIBLINEAR in terms of training time, memory usage and classification accuracy. Our experiments were run on a cluster of ten computers with the same hardware architecture as shown in Table 1. The cores in the same processor share one L2 cache and the main

Table 1: The physical features of a multi-core computer.

# of CPUs	# of cores	Frequency	Memory	L2 cache
2	8	2.10GHz*16	47.26GB	256KB*2

memory is shared among all the cores. All the computers are running Linux 3.2.0-4-amd64 (x86_64).

The extended versions of LIBLINEAR are designed for large scale datasets, so we have evaluated our implementations on the two following datasets.

ImageNet 100. This dataset contains the 100 largest classes from ImageNet (183,116 images with data size 23.6GB). In each class, we sample 1K images for training and 150 images for testing. We construct BoW histogram of images by using libHIK [39] with SIFT descriptor [40], 1000 codewords and parameters "use both, grid step size 2 and split level 1". The image is encoded as a 12000 dimensional vector. We end up with 10.5GB of training data.

ILSVRC 2010. This dataset contains 1K classes from ImageNet with 1.2M images for training, 50K images for validation and 150K images for testing. Due to the memory restriction of computer, we take $\leq$ 900 images per class for training. We use the BoW feature set provided by [9] and encode every image as a vector in 21000 dimensions. Therefore, the total training images is 887,816 and the training data size is 12.5GB. All testing samples are used to test SVM models.

5.1 Memory usage

According to the memory size of the computer used, we have split data into small blocks of rows that can fit into memory in each incremental step of LIBLINEAR.

ImageNet 100. We have split this dataset into 3 and 6 blocks of rows. As shown in Table 2, our implementation can run on computer with the main memory less than 4GB (LIBLINEAR-B-3) and less than 2GB (LIBLINEAR-B-6).

ILSVRC 2010. Due to the large size of dataset, we have split this dataset into 8 and 24 blocks of rows, that allows training data to fit into 4GB RAM (LIBLINEAR-B-8) and 2GB RAM (LIBLINEAR-B-24) in each incremental step. As shown in Table 3, LIBLINEAR and LIBLINEAR-CDBLOCK-B-8 consume a large amount of main memory (16.70GB and 9.68GB), making it intractable on computers with limited memory. On the other hand, by splitting data into many small blocks of rows and using one-versus-all approach for multi-class case, our approach is found to be very suitable for this case. For instance, LIBLINEAR-B-8 uses only 3.23GB RAM to train 1K classifiers on ILSVRC 2010. That means our implementation can save from 66.63% to 80.66% memory usage, compared to LIBLINEAR-CDBLOCK and LIBLINEAR. Furthermore, by setting the block size appropriately, the program does not need to swap parts of the blocks of rows between main memory and secondary memory (on the hard disk), as shown in Fig. 2.

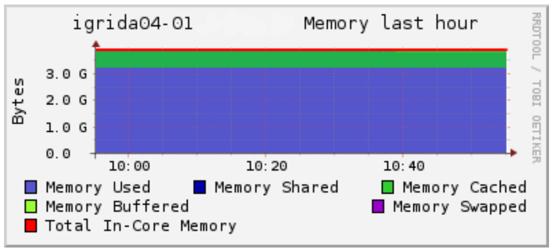

Fig. 2: Memory usage (GB) of the incremental LIBLINEAR (LIBLINEAR-B-8) on ILSVRC 2010.

Table 2: Memory usage (GB) of classifiers on ImageNet 100.

Method	ImageNet 100
LIBLINEAR	11.00
LIBLINEAR-CDBLOCK-B-3	3.78
LIBLINEAR-CDBLOCK-B-6	1.92
LIBLINEAR-B-3	3.71
LIBLINEAR-B-6	1.86

Note that the training time increases if we split the data into blocks of rows with smaller size. It is because the classifiers need to load and train more blocks (Table 4, 5).

5.2 Training time

We have implemented two extended versions of LIBLINEAR-CDBLOCK: 1) OpenMP balanced bagging incremental LIBLINEAR (omp-iLIBLINEAR-B), 2) Hybrid MPI/OpenMP balanced bagging incremental LIBLINEAR (mpi-omp-iLIBLINEAR-B). Incremental LIBLINEAR is designed to handle data beyond the memory size, so the training time is considered at disk-level:

training time = user time to run data into memory + time to access data from disk.

ImageNet 100. As shown in Table 4, on medium dataset ImageNet 100 our implementation shows a very good speedup in training process, compared to the original implementation. For instance, by splitting the dataset into 3 blocks of rows and use 10 MPI process and 16 OpenMP threads per MPI process, our implementation (10mpi-omp-iLIBLINEAR-B-3) is 494 times faster than LIBLINEAR-CDBLOCK-B-3.

ILSVRC 2010. Our implementations achieve a significant speedup in training process on this large dataset.

Balanced bagging incremental LIBLINEAR

As shown in Table 5, by splitting ILSVRC 2010 into 8 blocks, the balanced bagging incremental LIBLINEAR

Table 3: Memory usage (GB) of classifiers on ILSVRC 2010.

Method	ILSVRC 2010
LIBLINEAR	16.70
LIBLINEAR-CDBLOCK-B-8	9.68
LIBLINEAR-CDBLOCK-B-24	7.74
LIBLINEAR-B-8	3.23
LIBLINEAR-B-24	1.29

(omp-iLIBLINEAR-B-8 running with 1 thread) has a very fast convergence speed in training process, it is 11 times faster than LIBLINEAR-CDBLOCK-B-8.

OpenMP balanced bagging incremental LIBLINEAR

By applying balanced bagging algorithm to OpenMP version of incremental LIBLINEAR, we significantly speedup the training process of 1K binary classifiers. With the number of OpenMP threads set to 16, our implementation (omp-iLIBLINEAR-B-8) is 127 times faster than LIBLINEAR-CDBLOCK-B-8 (Table 5).

Hybrid MPI/OpenMP balanced bagging incremental LIBLINEAR

Although OpenMP balanced bagging incremental LIBLINEAR shows a significant speedup in training process, it does not ensure that the program achieves the most efficient high-performance computing on multi-core computers. Therefore, we explore this challenge by using a combination of MPI and OpenMP models. With this approach, our implementation achieves an impressive parallelization performance on a cluster of ten SMP (symmetric multiprocessor) nodes. For shorter, we use the technical term node instead of SMP node. The program first loads the whole block of data into nodes and each MPI process runs on one node. Therefore, each MPI process can work with their local data independently. However, we cannot increase the number of MPI processes exceed the memory capacity of a node. It is because each MPI process occupy the main memory during their computation process, resulting in an increase in the overall memory requirement. Unfortunately, OpenMP has been proven to work effectively on shared memory systems. It is used for fine-grained parallelization within a node. Consequently, in each node we can increase the number of OpenMP threads without demanding more extra memory. In this experiment, we have set the maximum number of OpenMP threads equal to the number of cores available on a node. As shown in Table 5, our implementation (10mpi-omp-iLIBLINEAR-B-8) achieves a significant speedup in training process by using 160 cores from ten nodes (10 MPI processes $\times$ 16 OpenMP threads). It is 732 times faster than LINEAR-CDBLOCK-B-8 and 1193 times faster than LIBLINEAR. We need only 2.62 minutes to train 1K binary classifiers, compared to LIBLINEAR-CDBLOCK-B-8 ($\sim$ 32 hours) and LIBLINEAR ($\sim$ 52 hours). This result confirms that our approach has a great ability to scaleup to full ImageNet dataset with more than 21K classes.

5.3 Classification accuracy

We have compared our implementations with LIBLINEAR-CDBLOCK and LIBLINEAR in terms of classification accuracy.

LIBLINEAR. The linear SVM from [7] with default parameter value $C = 1$.

LIBLINEAR-CDBLOCK-B. The block minimization framework for LIBLINEAR [8] with parameter $C = 1$, s = 4 (multi-class SVM by Crammer and Singer).

LIBLINEAR-B. The incremental LIBLINEAR with the same SVM parameters as LIBLINEAR (multi-class classification is implemented by using one-versus-all approach).

iLIBLINEAR-B. The balanced bagging incremental LIBLINEAR.

As shown in Table 6, on medium dataset ImageNet 100, iLIBLINEAR-B (4GB) is 1.74% worse than LIBLINEAR-CDBLOCK-B (4GB) in terms of classification accuracy. However, on large dataset ILSVRC 2010, the classification accuracy obtained by iLIBLINEAR-B (4GB) is nearly the same as LIBLINEAR-CDBLOCK-B (4GB) (it is 0.89% worse than the original implementation). This result shows that our balanced bagging algorithm is very useful when one wants to speedup the training process of classifiers on large scale datasets without (or very few) compromising classification accuracy.

Table 4: SVMs training time (minute) on ImageNet 100.

Method	# OpenMP threads		
	1	8	16
LIBLINEAR	188.97		
LIBLINEAR-CDBLOCK-B-3	202.75		
LIBLINEAR-CDBLOCK-B-6	243.87		
omp-LIBLINEAR-B-3	56.45	9.12	7.50
omp-LIBLINEAR-B-6	72.55	11.28	8.82
omp-iLIBLINEAR-B-3	27.57	4.52	3.28
omp-iLIBLINEAR-B-6	30.07	4.88	3.43
5mpi-omp-iLIBLINEAR-B-3	6.08	0.98	0.70
10mpi-omp-iLIBLINEAR-B-3	3.32	0.55	**0.41**

Table 5: SMVs training time (minute) on ILSVRC 2010.

Method	# OpenMP threads		
	1	8	16
LIBLINEAR	3126.78		
LIBLINEAR-CDBLOCK-B-8	1917.37		
LIBLINEAR-CDBLOCK-B-24	2533.33		
omp-LIBLINEAR-B-8	1287.27	164.32	134.22
omp-LIBLINEAR-B-24	1716.00	238.42	202.17
omp-iLIBLINEAR-B-8	174.12	22.85	15.12
omp-iLIBLINEAR-B-24	210.22	29.82	23.42
5mpi-omp-iLIBLINEAR-B-8	38.14	5.10	3.96
10mpi-omp-iLIBLINEAR-B-8	23.89	3.22	**2.62**

6. Conclusion and future work

In this paper, we have developed the extended versions of LIBLINEAR-CDBLOCK in three ways: (1) develop multi-class classification for LIBLINEAR-CDBLOCK by using the one-versus-all approach, (2) a balanced bagging

Table 6: Overall classification accuracy (%).

Method	ImageNet 100	ILSVRC 2010
LIBLINEAR	43.17	21.11
LIBLINEAR-CDBLOCK-B (4GB)	44.19	19.99
LIBLINEAR-CDBLOCK-B (2GB)	44.10	18.12
LIBLINEAR-B (4GB)	42.78	19.15
LIBLINEAR-B (2GB)	41.80	18.17
iLIBLINEAR-B (4GB)	42.45	19.10
iLIBLINEAR-B (2GB)	41.46	18.12

algorithm for training binary classifiers, (3) parallelize the training process of these classifiers with several multi-core computers. Our approach has been evaluated on the 100 largest classses of ImageNet and ILSVRC 2010. The experiment shows that our implementation is 732 times faster than the original implementation and 1193 times faster than LIBLINEAR with 160 cores. We need only 2.62 minutes to train 1K binary classifiers. Furthermore, our approach can be easily applied to dataset larger than the memory capacity of computer. Obviously, this is a roadmap towards large scale visual classification for systems with limited individual resource. The next step is to perform incremental LIBLINEAR on 10K classes of ImageNet. With this large dataset, the training data would be much larger than the capacity of many existing HPC systems.

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

References

[1] F.-F. Li, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer Vision and Image Understanding*, vol. 106, no. 1, pp. 59–70, 2007.

[2] G. Griffin, A. Holub, and P. Perona, "Caltech-256 Object Category Dataset," California Institute of Technology, Tech. Rep. CNS-TR-2007-001, 2007. [Online]. Available: http://authors.library.caltech.edu/7694

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, jun 2010.

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

[5] J. Deng, A. C. Berg, K. Li, and F.-F. Li, "What does classifying more than 10, 000 image categories tell us?" in *European Conference on Computer Vision*, 2010, pp. 71–84.

[6] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. S. Huang, "Large-scale image classification: Fast feature extraction and svm training," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1689–1696.

[7] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear svm," in *International Conference on Machine Learning*, 2008, pp. 408–415.

[8] H.-F. Yu, C.-J. Hsieh, K.-W. Chang, and C.-J. Lin, "Large linear classification when data cannot fit in memory," *ACM Transactions on Knowledge Discovery from Data*, vol. 5, no. 4, p. 23, 2012.

[9] A. Berg, J. Deng, and F.-F. Li, "Large scale visual recognition challenge 2010," Tech. Rep., 2010. [Online]. Available: http://www.image-net.org/challenges/LSVRC/2010/index

[10] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *In Workshop on Statistical Learning in Computer Vision, ECCV*, 2004, pp. 1–22.

[11] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

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

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2005, pp. 886–893.

[14] G. Griffin and D. Perona, "Learning and using taxonomies for fast visual categorization," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2008.

[15] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 606–613.

[16] R. Fergus, Y. Weiss, and A. Torralba, "Semi-supervised learning in gigantic image collections," in *Advances in Neural Information Processing Systems*, 2009, pp. 522–530.

[17] C. Wang, S. Yan, and H.-J. Zhang, "Large scale natural image classification by sparsity exploration," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2009, pp. 3709–3712.

[18] Y. Li, D. J. Crandall, and D. P. Huttenlocher, "Landmark classification in large-scale image collections," in *IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 1957–1964.

[19] F. Perronnin, J. Sánchez, and Y. Liu, "Large-scale image categorization with explicit data embedding," in *CVPR*, 2010, pp. 2297–2304.

[20] J. Sánchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1665–1672.

[21] K. Yu, T. Zhang, and Y. Gong, "Nonlinear learning using local coordinate coding," in *Advances in Neural Information Processing Systems*, 2009, pp. 2223–2231.

[22] X. Zhou, K. Yu, T. Zhang, and T. S. Huang, "Image classification using super-vector coding of local image descriptors," in *European Conference on Computer Vision*, 2010, pp. 141–154.

[23] C. C. Chang and C. J. Lin, "LIBSVM – a library for support vector machines," 2001, http://www.csie.ntu.edu.tw/$\sim$cjlin/libsvm.

[24] T. Joachims, "Training linear svms in linear time," in *proc. of the ACM SIGKDD Intl. Conf. on KDD*. ACM, 2006, pp. 217–226.

[25] T.-N. Do, V.-H. Nguyen, and F. Poulet, "Speed up svm algorithm for massive classification tasks," in *ADMA*, 2008, pp. 147–157.

[26] J. Weston and C. Watkins, "Support vector machines for multi-class pattern recognition," in *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, 1999, pp. 219–224.

[27] Y. Guermeur, "Svm multiclasses, théorie et applications," 2007.

[28] U. Krebel, "Pairwise classification and support vector machines," *Advances in Kernel Methods: Support Vector Learning*, pp. 255–268, 1999.

[29] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multiclass classification," *Advances in Neural Information Processing Systems*, vol. 12, pp. 547–553, 2000.

[30] V. Vural and J. Dy, "A hierarchical method for multi-class support vector machines," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, pp. 831–838.

[31] K. Benabdeslem and Y. Bennani, "Dendogram-based svm for multi-class classification," *Journal of Computing and Information Technology*, vol. 14, no. 4, pp. 283–289, 2006.

[32] K. Crammer and Y. Singer, "On the learnability and design of output codes for multiclass problems," *Machine Learning*, vol. 47, no. 2-3, pp. 201–233, 2002.

[33] N. Japkowicz, Ed., *AAAI'Workshop on Learning from Imbalanced Data Sets*, ser. AAAI Tech Report, no. WS-00-05, 2000.

[34] S. Visa and A. Ralescu, "Issues in mining imbalanced data sets - A review paper," in *Midwest Artificial Intelligence and Cognitive Science Conf.*, Dayton, USA, 2005, pp. 67–73.

[35] P. Lenca, S. Lallich, T. N. Do, and N. K. Pham, "A comparison of different off-centered entropies to deal with class imbalance for decision trees," in *The Pacific-Asia Conference on Knowledge Discovery and Data Mining, LNAI 5012*. Springer-Verlag, 2008, pp. 634–643.

[36] N. K. Pham, T. N. Do, P. Lenca, and S. Lallich, "Using local node information in decision trees: coupling a local decision rule with an off-centered entropy," in *International Conference on Data Mining*. Las Vegas, Nevada, USA: CSREA Press, 2008, pp. 117–123.

[37] MPI-Forum, "Mpi: A message-passing interface standard," 1995. [Online]. Available: http://www.mpi-forum.org

[38] OpenMP Architecture Review Board, "OpenMP application program interface version 3.0," 2008. [Online]. Available: \url{http://www.openmp.org/mp-documents/spec30.pdf}

[39] J. Wu, W.-C. Tan, and J. M. Rehg, "Efficient and effective visual codebook generation using additive kernels," *Journal of Machine Learning Research*, vol. 12, pp. 3097–3118, 2011.

[40] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.