Parallelizing Support Vector Machines for Scalable Image Annotation

A Thesis submitted for the Degree of

Doctor of Philosophy

By

Nasullah Khalid Alham

Department of Electronic and Computer Engineering

School of Engineering and Design

Brunel University

April 2011

Abstract

Machine learning techniques have facilitated image retrieval by automatically classifying and annotating images with keywords. Among them Support Vector Machines (SVMs) are used extensively due to their generalization properties. However, SVM training is notably a computationally intensive process especially when the training dataset is large.

In this thesis distributed computing paradigms have been investigated to speed up SVM training, by partitioning a large training dataset into small data chunks and process each chunk in parallel utilizing the resources of a cluster of computers. A resource aware parallel SVM algorithm is introduced for large scale image annotation in parallel using a cluster of computers. A genetic algorithm based load balancing scheme is designed to optimize the performance of the algorithm in heterogeneous computing environments.

SVM was initially designed for binary classifications. However, most classification problems arising in domains such as image annotation usually involve more than two classes. A resource aware parallel multiclass SVM algorithm for large scale image annotation in parallel using a cluster of computers is introduced.

The combination of classifiers leads to substantial reduction of classification error in a wide range of applications. Among them SVM ensembles with bagging is shown to outperform a single SVM in terms of classification accuracy. However, SVM ensembles training are notably a computationally intensive process especially when the number replicated samples based on bootstrapping is large. A distributed SVM ensemble algorithm for image annotation is introduced which re-samples the training data based on bootstrapping and training SVM on each sample in parallel using a cluster of computers.

The above algorithms are evaluated in both experimental and simulation environments showing that the distributed SVM algorithm, distributed multiclass SVM algorithm, and distributed SVM ensemble algorithm, reduces the training time significantly while maintaining a high level of accuracy in classifications.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgments

I would like to thank all those who have given me academic and moral support for my research work over the last three years. I would like to thank the department of Electronic and Computing Engineering, in particular to my supervisor, Dr. Maozhen Li for his guidance and valuable advice.

I would like to thank all my family members, My mother, My wife, My brothers and sisters for their support and to whom I dedicate this work.

I would like to thank my friends and fellow PhD students Yang Liu and Suhel Hammoud for their help and advice over the last three years.

Author's Declaration

The work described in this thesis has not been previously submitted for a degree in this or any other university and unless otherwise referenced it is the author's own work.

Statement of Copyright

Publications

The following papers have been accepted for publication or under review as a direct or indirect result of the research discussed in this thesis.

Journal Papers

N. K. Alham, M. Li, Y. Liu and S. Hammoud, A MapReduce-based Distributed SVM Algorithm for Automatic Image Annotation, Computers & Mathematics with Applications, Elsevier (accepted for publication)

Y. Liu, M. Li, **N. K. Alham**, S. Hammoud, HSim: A MapReduce Simulator in Enabling Cloud Computing, Future Generation Computer Systems (FGCS), the International Journal of Grid Computing and e-Science, Elsevier Science (accepted for publication)

N. K. Alham, M. Li, Y. Liu and S. Hammoud, Parallelizing Multiclass Support Vector Machines for Scalable Image Annotation, Neurocomputing, Elsevier Science (under review)

N. K. Alham, M. Li, Y. Liu and S. Hammoud, A Resource Aware Parallel SVM for Scalable Image Annotation, Parallel Computing, Elsevier Science (under review)

G. Caruana, M. Li, **N.K. Alham**, and Y. Liu, A Parallel Support Vector Machine for Large Scale Spam Filtering, information Sciences, Elsevier Science (under review)

Y. Liu, M. Li, **N. K. Alham**, S. Hammoud, A Resource Aware Distributed LSI for Scalable Information Retrieval, Information Processing and Management, Elsevier Science (under review)

Conference Papers

N. K. Alham, M. Li, S. Hammoud, Y. Liu, M. Ponraj, MapReduce-based Distributed SMO for Support Vector Machines, Proc. of IEEE FSKD'10, pp. 2983-2987.

N. K. Alham, M. Li, S. Hammoud and H. Qi, Evaluating Machine Learning Techniques for Automatic Image Annotations, Proc. of IEEE ICNC09-FSKD09, pp.245-249, 2009 (invited paper).

N. K. Alham and M. Li, Content Based Image Retrieval: A Review, Proc. of ICAC'08, Sept. 2008, Brunel University, UK.

Y. Liu, M. Li, S. Hammoud, **N.K. Alham**, M. Ponraj, Distributed LSI for Information Retrieval, Proc. of IEEE FSKD'10, pp. 2978-2982.

S. Hammoud, M. Li, Y. Liu, **N. K. Alham**, Z. Liu, MRSim: A Discrete Event based MapReduce Simulator, Proc. of IEEE FSKD'10, pp. 2993-2997.

Chapter 1

Introduction

This chapter briefly describes the background to the problems investigated in this thesis, motivation of work, major contributions and the structure of the thesis.

1.1 Background

The increasing volume of images being generated by digitized devices has brought up a number of challenges in image retrieval. Content Based Image Retrieval (CBIR) was proposed to allow users retrieve relevant images based on their low-level features such as colour, texture and shape. The past decade has seen a rapid development in CBIR. In CBIR systems images are first segmented into regions or fixed size blocks, and then image features can be extracted. For example, by extracting colour histograms, the colour content of an image can be represented [21]. In a retrieval process, users feed the retrieval system with query images. The CBIR system then computes these images into its internal representation of feature vectors. The similarities or distances between the feature vectors of a query image and those of the images in the image database can be calculated and retrieval is performed with the aid of an indexing scheme such as HG-tree [23]. HG-tree is a multi dimensional Point Access Method (PAM) which is used to index multi-dimensional data to support queries such as "Find all images that are similar to a query image". Figure 1.1 shows the processes of CBIR.

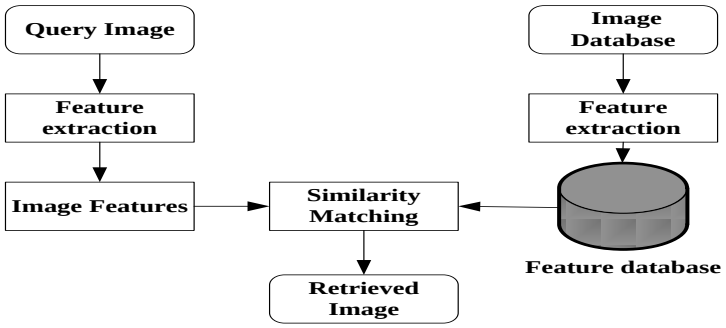

Figure 1.1: CBIR main processes

One of the primary components in CBIR is colour analysis [95]. Each image is analyzed to compute a colour histogram which shows the proportion of pixels of each colour within the image. The colour histogram for each image is then stored in the database. Colour moments are also measures that can be used to compute the similarity of images based on their colour features. Colour moments are based on the theory that the distribution of colour in an image can be interpreted as a probability distribution. Probability distributions are characterized by a number of unique moments; the moments of the distribution can be used as features to identify that image based on colour. Stricker and Orengo [125] use three moments of an image's colour distributions which are Mean, Standard deviation and Skewness. These values of similarities can then be compared with the values of images indexed in a database for tasks like image retrieval [95].

Another key component in CBIR is the analysis of the texture of an image which is the perception of smoothness or coarseness of an object. Similar to the colour histogram mentioned above, many of the current techniques for image texture analysis while quantified, lack the spatial information that allows one to compare the location of a coarse object with a smooth object within an image [100]. There is a notable use of Local Binary Pattern (LBP) in CBIR. Block based methods which divides a query image and database images (or database images only) into blocks and compare their LBP histograms are found to perform significantly better than the methods based on global LBP histograms [71]. Other texture features such as Gabor Filters are applied to images convert image texture components into graphs. A comparison of these images is performed based on the mathematical representation of these graphs. This makes it possible to compare the textures of two different images [136].

The ability to retrieve images based on shapes is perhaps the most obvious requirement at the primitive level [28]. Unlike texture, shape is a fairly well-defined concept and there is considerable evidence that natural objects are primarily recognized by their shapes. In contrast to colour and texture features, shape features are described after an image is segmented into objects. Since accurate image segmentation is difficult to achieve automatically. Using shapes in CBIR is limited to specific application where objects are readily available [93].

The accuracy of CBIR is not adequate due to the existence of a *Semantic Gap*, a gap between the low-level visual features such as textures and colours and the high-level concepts that are

normally used by the user in the search process [122]. Annotating images with labels is one of the solutions to narrow the semantics gap [132]. Automatic image annotation is a method of automatically generating one or more labels to describe the content of an image, a process which is commonly considered as a multi-class classification. Typically, images are annotated with labels based on the extracted low level features. Machine learning techniques have facilitated image annotation by learning the correlations between image features and annotated labels.

Support Vector Machine (SVM) techniques have been used extensively in automatic image annotation [12] [26] [34] [48] [49] [85] [147]. The qualities of SVM based classification have been proven remarkable [30] [40] [119] [143]. In its basic form SVM creates a *hyperplane* as the decision plane, which separates the positive and negative classes with the largest margin [119]. SVMs have shown a high level of accuracy in classifications due to their generalized properties. SVMs can correctly classify data which is not involved in the training process. This can be evidenced from our previous work in evaluating the performance of representative classifiers in image annotation [77]. The evaluation results showed that SVM performs better than other classifiers in term of accuracy, however the training time of the SVM classifier is notably longer than that of other classifiers.

SVM was initially designed for binary classifications. However classification problems in domains such as image annotation usually involve more than two classes. Extending binary SVM solutions effectively to solve multi-class classification is an ongoing research issue [59]. Due to various complexities, a direct solution to multiclass problems using a single step SVM training is usually avoided [44]. A superior approach is to combine a number of binary SVM classifiers to solve a multiclass problem. Various approaches have been proposed such as One Against Rest [45] (OAR), One Against One (OAO) [79] and decision trees based multiclass SVM techniques [114].

Due to various complexities in classification problems, it is difficult to systematically create classifiers with enhanced performance. The combination of classifiers leads to considerable reduction of misclassification error in a wide range of applications. Among them SVM ensembles is shown to outperform a single SVM in terms of classification accuracy [18].

Bagging [15] is the most commonly used combination method which combines multiple classifiers by introducing randomness in the training data. The bagging method is useful in

reducing the variance component of the expected misclassification error of a classifier. Bagging is effective particularly for classifiers with high variance and low bias, which is described in [15] as unstable classifiers. Unstable classifiers experience significant changes with small change of the training data or other parameters [54].

1.2 Motivation of Work

A number of machine learning techniques are available for image annotation. These techniques are usually evaluated under different environments using different low level features of images. To facilitate the selection of best machine learning techniques to be used in image annotation, there is a need for evaluating some representative techniques under the same environment using the same set of low level features.

It has been widely recognized that SVMs are computationally intensive when the size of a training dataset is large. A SVM kernel usually involves an algorithmic complexity of $O(m^2n)$, where n is the dimension of the input and m represents the training instances. The computation time in SVM training is quadratic in the number of training instances.

To speed up SVM training, distributed computing paradigms have been investigated to partition a large training dataset into small data chunks and process each chunk in parallel utilizing the resources of a cluster of computers [24] [41] [61] [153]. The approaches include those that are based on the Message Passing Interface (MPI) [8] [10] [20] [21] [148] [153].

A comparative study of the most popular multiclass SVM approaches indicates that OAO approaches usually perform better than others in terms of training efficiency and classification accuracy [27]. However OAO does not perform well when the datasets of the classes to be processed are different in size.

To speed up multiclass SVM training, distributed computing paradigms have been investigated to partition a large training dataset into small data chunks and process each chunk in parallel utilizing the resources of a cluster of computers [20] [21] [41] [63] [105]. The approaches include those that are based on the Message Passing Interface (MPI) [155]. However, MPI is primarily targeted at homogeneous computing environments and has limited support for fault tolerance. Although some progress has been made by these approaches, exiting distributed multiclass SVM algorithms employ naive and ineffective schedulers to address the problem of unbalanced multiclass datasets in homogeneous

computing environments in which the computers have similar computing capabilities. Currently heterogeneous computing environments are increasingly being used as platforms for resource intensive distributed applications. One major challenge is to balance the computation loads across a cluster of participating computer nodes.

SVM ensembles based on bagging show improvement in classification performance compare to a single SVM. Although some progress has been made by these approaches in classification accuracy, current method of builds replicates training data sample by randomly re-sampling with replacement, from the given training data set repeatedly. The number of samples required to create an effective ensemble SVM is debatable. Improving classification performances for fixed number replicates training data has not been studied. Ensemble learning is extremely computational intensive which limits their applications in real environments. Moreover SVM classifiers applied in ensemble learning require large computing resources due to the fact that computation time in SVM training is quadratic in terms of the number of training instances.

1.3 Major Contributions

Evaluation of seven representative machine learning classifiers for image annotation namely SVM, Bayesian Network, Naive Bayes, Boosting, Bagging, kNN and Decision tree from the aspect of accuracy and efficiency is presented. To facilitate performance evaluation, an image annotation prototype has been implemented which builds training models on low level features extracted from sample images. The evaluation results showed that SVM performs better than other classifiers in term of accuracy, however the training time of the SVM classifier is notably longer than that of other classifiers.

Resource Aware Sequential Minimal Optimization (RASMO), a distributed SVM algorithm for automatic image annotation has been implemented. RASMO builds on the Sequent Minimal Optimization (SMO) algorithm [113] for high efficiency in training and employs MapReduce [37] for parallel computation across a cluster of computers. MapReduce has become a major enabling technology in support of data intensive applications. RASMO is implemented using the Hadoop implementation [3] of MapReduce. The MapReduce framework facilitates a number of important functions such as partitioning the input data, scheduling MapReduce jobs across a cluster of participating nodes, handling node failures,

and managing the required network communications. A notable feature of the Hadoop implementation of MapReduce framework is the ability to support heterogeneous environments but without an effective load balancing scheme for utilizing resources with varied computing capabilities. For this purpose, a genetic algorithm based load balancing scheme is designed to optimize the performance of RASMO in heterogeneous computing environments.

The RASMO algorithm is designed based on a multi-layered cascade architecture which removes non-support vectors early in the training process and guarantees a convergence to the global optimum [58] [151]. The genetic algorithm based load balancing scheme is applied in the first layer computation in RASMO as this layer is the most intensive part in computation in optimizing the whole training dataset. The resulting support vectors from the first layer computation are used to create the input data for next layers which is usually much smaller in size in comparison with the original training data [104]. The size of each data chunk at the first layer is computed by the load balancing scheme based on the resources available in a cluster of computers such as the computing powers of processors, the storage capacities of hard drives and the network speeds of the participating nodes.

The performance of RASMO is first evaluated in a small scale experimental MapReduce environment. Subsequently, a MapReduce simulator is implemented to evaluate the effectiveness of the resource aware RASMO algorithm in large scale heterogeneous MapReduce environments. Both experimental and simulation results show that RASMO reduces the training time significantly while maintaining a high level of accuracy in classification. In addition, data chunks with varied sizes are crucial in speeding up SVM computation in the training process. It is worth pointing out that using different sizes for data chunks has no impact on accuracy in SVM classification due to the structure of the RASMO algorithm in which the training work in the first few layers is merely a filtering process of removing non-support vectors and the resulting support vectors of the last layer are evaluated for a global convergence by feeding the output of the last layer into the first layer.

Resource Aware Multiclass Sequential Minimal Optimization (RAMSMO), a resource aware distributed multiclass SVM algorithm for scalable image annotation has been designed and implemented. RAMSMO is built on MapReduce framework for parallel computation across a cluster of computers. A genetic algorithm based load balancing scheme is used to optimize

the performance of RAMSMO when processing binary data chunks with different sizes in heterogeneous environments in which the participating computers have varied resources in terms of the computing powers of processors, the storage capacities of hard drive and the network speeds of the participating nodes.

The performance of RAMSMO is evaluated in both small scale experimental and large scale MapReduce environments including the effectiveness of the load balancing scheme in large scale heterogeneous MapReduce environments. Both experimental and simulation results show that RAMSMO reduces the training time significantly while maintaining a high level of accuracy in classification.

MapReduce Ensemble Sequential Minimal Optimization (MRESVM), a distributed SVM ensemble algorithm for automatic image annotation has been implemented. MRESVM builds on the SMO algorithm for high efficiency in training and employs MapReducefor parallel computation across a cluster of computers.

The MRESVM algorithm is based on the bagging architecture which train multiple SVMs on bootstrap samples and combines the output in appropriate manners. Two types of combination methods are considered, firstly majority voting which is the commonly used combination method for bagging. Secondly combination of SVMs based double layer hierarchical combining that use second layer SVM to combine the first layer SVMs. Balanced sampling strategy for bootstrapping is introduced to increase classification accuracy for fixed number samples. The performance of the MRESVM algorithm is evaluated in both small scale experimental and large scale MapReduce environments. Both experimental and simulation results show that MRESVM reduces the training time significantly while increase the classification accuracy compare to a single SVM.

1.4 Structure of the Thesis

The rest of this thesis is organised as follows. Section 2.1 introduces image annotation techniques. Section 2.2 describes the basic concepts of SVM while Section 2.3 introduces distributed SVM. Section 2.4 introduces distributed Multiclass SVM Section 2.5 describes distributed SVM ensemble. Section 2.6 reviews and discuss the related work. Section 2.7 concludes the chapter.

Chapter 3 describes the implementation of an image annotation system which is essential for evaluating most commonly used machine learning classifiers in automatic image annotation. The evaluation results are presented in this chapter.

Chapter 4 is dedicated to the implementation of the RASMO and evaluation of the algorithm in experimental and simulation environment. A Genetic algorithm is introduced to enhance the performances in heterogonous computing environment.

Chapter 5 presents the implementation of the RAMSMO for training multiclass SVM and evaluation of the algorithm in experimental and simulation environment. A Genetic algorithm is introduced to enhance the performances in heterogonous computing environment.

Chapter 6 presents the implementation of the MRESVM for training SVM ensemble and evaluation of the algorithm in experimental and simulation environment.

Finally, chapter 7 summarises the contributions of the thesis and proposes directions for future work.

Chapter 2

Literature Review

This thesis is conducted from four different aspects, namely evaluated automatic image annotation techniques, distributed binary SVM, distributed multiclass SVM and distributed SVM ensemble. This chapter briefly describes the above techniques, reviewing the related literatures and summarising the weakness of the existing techniques.

2.1 Image Annotation Techniques

In recent years image annotation has become a major approach to bridging the semantic gap. This section describes some the main techniques used in image annotation.

Currently a great number of images are widely available on the World Wide Web. In order to organize and efficiently retrieve this vast number of images, contextual information of the images such as surrounding text and links is used for image annotation.

Semantic Web technologies such as ontologies have been used to annotate images with semantic descriptions. Ontology [123] is a specification of an abstract which defines a set of representational terms called concepts. Ontology based semantic image annotation focuses on describing the contents of an image, and tries to describe image contents as fully as possible.

Automatic image annotation is a method of automatically generating one or more labels to describe the content of an image. Typically, images are annotated with labels based on the extracted low level features. Machine learning techniques such SVM, Bayesian Networks, Artificial Neural Networks, Decision Tree and Composite Classifiers such bagging and boosting have facilitated image annotation by learning the correlations between image features and annotated labels.

2.2 Distributed SVM

It has been widely recognized that training SVMs is computationally intensive when the size of a training dataset is large. A SVM kernel usually involves an algorithmic complexity of $O(m^2n)$, where n is the dimension of the input and m represents the training instances. The computation time in SVM training is quadratic in terms of the number of training instances. To speed up SVM training, distributed computing paradigms have been investigated to partition a large training dataset into small data chunks and process each chunk in parallel utilizing the resources of a cluster of computers.

2.3 Distributed Multiclass SVM

Due to various complexities, a direct solution to multiclass problems using a single step SVM training is usually avoided [44]. A superior approach is to combine a number of binary SVM classifiers to solve a multiclass problem. Various approaches have been proposed such as One Against Rest [45] (OAR), One Against One (OAO) [79] and decision trees based multiclass SVM techniques [114]. To speed up SVM training, distributed computing paradigms have been investigated to partition a large training dataset into small data chunks and process each chunk in parallel utilizing the resources of a cluster of computers.

2.4 SVM Ensemble

The combination of classifiers leads to significant reduction of classification error in a wide range of applications. Among them SVM ensembles is shown to outperform a single SVM in terms of classification accuracy. However, SVM ensembles training are notably a computationally intensive process especially when the number replicated samples based on bootstrapping is large. Ensemble learning is extremely computational intensive which limits their applications in real environments.

2.5 Related Work to this Thesis

This section reviews the related literatures in automatic image annotation techniques, distributed binary SVM, distributed multiclass SVM and SVM ensemble.

2.5.1 Image Annotation Techniques

In order to organize and efficiently retrieve vast number of images on the Web, contextual information of the images such as surrounding text and links are used for image annotation. Hua et al. [66] introduce a system which automatically acquires semantic knowledge for Web images. A page layout analysis method is used to assign context to Web images. Joshi et al. [72] propose a scheme for automated story picturing using stop word elimination and identification of a set of proper nouns. The text of a story is processed based on the Wordnet [102] which forms a list of keywords.

Although image retrieval techniques based on textual information can retrieve many relevant images, the accuracy level of image retrieval is low [92]. The main reasons for low level of accuracy are; firstly the Web images are used freely in the Web pages and there is no standard exists for the relationships between the texts and embedded images in the same Web pages, secondly Web images are fairly comprehensive in meaning, and are created by different people for different purposes, thirdly the qualities of the Web images vary greatly [57]. The users need to go through the entire list of retrieved images to find the desired ones. To improve Web image retrieval performance, there is an on-going research to combine the textual information and visual image contents [92].

Marques and Barman [100] propose three layer architecture for image annotation. The bottom layer extracts low level features of images, which are mapped to semantically meaningful keywords in the middle layer, which are then connected to schemas and ontologies on the top layer. Petridis et al. [112] present a software environment called M-Onto Mat-Annotizer to bridge the gap between the low level visual descriptors and high level semantic concepts. M-Onto Mat-Annotizer allows linking low level MPEG-7 visual descriptions to the Visual Descriptor Ontology (VDO). Hollink et al. [65] argue that ontologies serve two purposes in image annotation. Firstly, user is immediately provided with the right context to find an adequate index term. This ensures quicker and more precise indexing. Secondly, the hierarchical presentation of concepts helps to disambiguate terms. They propose a scheme for semantic image annotation and retrieval in a collection of art images using multiple ontologies to support this process. Srikanth et al. [123] use a hierarchy of annotation words derived from text ontology for automatic image annotation.

Wang et al. [141] compare ontology-based image annotation with keyword-based image annotation. It has been found that keyword based approach is user friendly and easy to apply with acceptable retrieval accuracy, while semantically rich ontology addresses the need for complete descriptions of image retrieval and improves the accuracy of retrieval. Ontology works better with the combination of low level image features. However there is a trade-off between the complexity and performance. Ontology based annotation work better by combining low level features with high level textual information due to usefulness of visual information to filter a majority of inaccurate results. For instance, from an indoor background it can be inferred that a *wild fox* is not likely to exist in an image.

SVM is considered as a good candidate for image annotations due to its high generalisation performance without the need to add prior knowledge [25]. Zhang et al. [154] used a SVM classifier to separate two classes of relevant images and irrelevant images. A classifier is trained with training data of relevance images and irrelevance images marked by users. The trained model is used to find more relevance images in an image database. Tsai et al. [131] propose a system which is composed of three modules of SVMs for colour, texture, and high-level concept classification. Cusano et al. [34] present an image annotation tool for classifying image regions in one of seven classes- sky, skin, vegetation, snow, water, ground, and buildings using multi-class SVM. Wang et al [40] used SVM and point out the main drawback of the SVM models are too large to be used in a practical system with limited memory space. As a result, the speed of the classification is also slow when using SVM models with many support vectors.

Barrat and Tabbone [5] use a Bayesian network to classify images based on visual and textual features and to automatically annotate new images. Kane and Savakis [73] employ low-level classification based on colour and texture, semantic features such as sky and grass detections, along with indoor and outdoor ground truth information, to create a set of features for Bayesian network structure learning. It is reported that a Bayesian network provides classification rates which are 97% correct. Benitez and Chang [7] use a Bayesian network in combination of meta-classifiers. For a new image, the presence of concepts is first detected using the meta-classifiers and then is refined using Bayesian inferences. Niedermayer [35] claims weakness of Bayesian network lays on the quality and extent of the prior beliefs used

in Bayesian inference process. A Bayesian network is only is useful when this prior knowledge is reliable.

One of the widely used techniques is to predict the class of a new instance based on the most common class amongst the *k* Nearest Neighbours [9]. *k* Nearest Neighbours classifiers are known as non-parametric classifiers. Non-parametric classifiers can naturally handle a huge number of classes, and avoid over fitting of parameters which is a central issue in learning based approaches. In addition, non-parametric classifiers do not require learning/training phases. Makadia et al. [99] introduce a technique for image annotation that treats image annotation as a retrieval problem, using low-level image features and a simple combination of basic distances to find the nearest neighbours of a given image. The keywords are then assigned using a greedy label transfer mechanism. Pakkanen et al. [108] use MPEG-7 feature vectors to perform a kNN classification of the images. They report that the results are generally satisfactory especially the Colour Structure and Homogeneous Texture descriptors seem to perform well. Lepisto et al. [67] present a method for combining different visual descriptors in rock image classification. In their approach, the k-NN classification is first carried out for each descriptor separately. After that, a final decision is made by combining the nearest neighbours in each base classification. The total numbers of the neighbours representing each class are used as votes in the final classification.

In image annotation, low-level feature vectors are fed into the input layer of a multilayer perceptron (MLP) where each of the input neurons corresponds to each of the feature vectors and the output neurons of the MLP represent the class labels of images to be classified. Zhao et al [156] propose an annotation system based on a neural network for characterising the hidden association between the visual and the textual modalities. Latent semantic analysis (LSA) is employed to discover the latent contextual correlation among the keywords. Shah and Lim et al [89] use a three-layer feed-forward neural network with dynamic node creation capabilities to learn 26 visual keywords from 375 labelled image patches collected from home photos. Colour and texture features are computed for each training region as an input vector for the neural network. Breen et al. [14] propose an annotation system which uses ontologies and neural networks as object identifiers to provide a high level of accuracy in automatic classification of images.

ID3 and C4.5 are well known algorithms to construct a decision tree classifier; however ID3 has some disadvantages such as preference bias and the inability to deal with unknown attribute values [149]. Tseng and Su [133] use the decision tree algorithm to build a classifier with low-level features extracted from images. The classifier is then used for classifying images with one representative object. Huang et al [68] use decision tree to categorize new images. It has been suggested that this scheme performs better than standard k-nearest neighbour techniques, and also has both storage and computational advantages [68].

Feng and Chua [50] propose bootstrapping approach to deal with the problem of providing large labelled training data which is needed in the training stage of a classifier to annotate a large collection of images. The idea is to start from a small set of labelled training images, and consecutively annotate a larger set of unlabeled images by using the co-training approach, in which two statistically independent classifiers are used to co-train and co-annotate the unlabeled images. This process offers the advantage of requiring only a small initial set of training images. Huan [95] claim boosting method such as adaboost, boosts a weak learning algorithm by updating the sample weights iteratively. They propose to integrate feature reweighting into boosting scheme, which not only weights the samples but also weights the feature elements iteratively. Fan et al [48] propose a hierarchical boosting algorithm by integrating concept ontology and multi-task learning to achieve hierarchical image classifier training with automatic error recovery.

2.5.2 Distributed SVM

SVM training is a computationally intensive process especially when the size of the training dataset is large. Numerous avenues have been explored with an effort to increase efficiency and scalability, to reduce complexity as well as ensure that the required level of classification accuracy can be maintained. SVM decomposition is a widespread technique for performance improvement [4] [107] [127].

Decomposition approaches work on the basis of identifying a small number of optimization variables and tackling a set of problems with a fixed size. One approach is to split the training data set into a number of smaller data chunks and employs a number of SVMs to process the individual data chunks.

Various forms of summarizations and aggregations are then performed to identify the final set of global support vectors. Hazen et al. [61] introduced a parallel decomposition algorithm for training SVM where each computing node is responsible for a pre-determined subset of the training data. The results of the subset solutions are combined and sent back to the computing nodes iteratively. The algorithm is based on the principles of convex conjugate duality. The key feature of the algorithm is that each processing node uses independent memory and CPU resources with limited communication overhead. Zanghirati et al. [153] presented a parallel SVM algorithm using MPI which splits the problem into smaller quadratic programming problems. The output results of the sub-problems are combined. The performance of the parallel implementation is heavily depended on the caching strategy that is used to avoid re-computation of the previously used elements in kernel evaluation which is considered as computationally intensive. Similarly, MPI based approaches have been proposed for speeding up SVM in training [8] [10] [20] [21] [148]. Whilst good performance improvements can be achieved by MPI based parallelization, these approaches tend to suffer from poor scalability, high overhead in inter-node communication, and limited support for heterogeneous computing environments.

Collobert et al. [31] proposed a parallel SVM algorithm which trains multiple SVMs with a number of subsets of the data, and then combines the classifiers into a final single classifier. The training data is reallocated to the classifiers based on the classification accuracy and the process is iterated until a convergence is reached. However the frequent reallocation of training data during the optimization process may cause a reduction in the training speed. Huang et al. [67] proposed a modular network architecture which consists of several SVMs of which each is trained using a portion of the whole training dataset. It is worth noting that speeding up the training process can significantly reduce the generalization performance due to the increase in the number of partitions. Lu et al. [97] proposed a distributed SVM algorithm based on the idea of partitioning training data and exchanging support vectors over a strongly connected network. The algorithm converges to a global optimal classifier in finite steps. The performance of this solution is depended on the size and topology of network. The larger a network is, the higher communication overhead will incur. Kun et al. [83] implemented a parallel SMO using Cilk [130] and Java threads. The idea is to partition the training data into smaller parts, train these parts in parallel, and combines the resulting

support vectors. However Cilk's main disadvantage is that it requires a shared-memory computer [81].

An interesting alternative is considered and discussed in [21]. The work on updating optimality condition vectors is performed in a parallel way leading to a speedup in SVM training. However this approach can incur considerable network communication overhead due to the large number of iterations involved. Another approach utilizes Graphics Processing Units (GPU) for SVM speedup [27]. MapReduce was adopted in this work exploiting the multi-threading capabilities of graphics processors. The results show a considerable decrease in processing time. A key challenge with such an approach lies in the specialized environments and configuration requirements. The dependency of specific development tools and techniques as well as platforms introduces additional, non-trivial complexities.

SVM algorithms rely on the number of support vectors for classification. Removing non-support vectors in an early stage in the training process has proven to be useful in reducing the training time. Dong et al. [43] proposed a parallel algorithm in which multiple SVMs are solved with partitioned data sets. The support vectors generated by one SVM are collected to train another SVM. The main advantage of this parallel optimization step is to remove non-support vectors which can help reduce the training time. Graf et al. [58] proposed a similar parallel SVM algorithm using a homogenous Linux cluster. The training data is partitioned and an SVM is solved for each partition. The support vectors from each pair of classifiers are then combined into a new training dataset for which an SVM is solved. The process carries on until a final single classifier is left. Although the convergence to the global optimum can be guaranteed, partitioning a large dataset into smaller data chunks with the same size can only be effective in a homogeneous computing environment in which computers have similar computing capabilities. Another similar work is presented in [146].

Given the focus that most of the current approaches are primarily on the SVM solver, parallelization using a number of computers may introduce significant communication and synchronization overheads. Frameworks such as MapReduce are believed to provide an effective application scope in this context [56]. Chu et al. [29] capitalized natively on the multi-core capabilities of modern day processors and proposed a distributed linear SVM using the MapReduce framework; batch gradient descent is performed to optimize the

objective function. The mappers calculate the partial gradient and the reducer sums up the partial results to update weights vector. However the batch gradient descent algorithm is extremely slow *to* converge with some type of training data [119].

2.5.3 Distributed Multiclass SVM

Existing research efforts in multiclass SVM classifications generally fall into two approaches. One approach is to consider all the classes in a single optimization step and the other approach is a combination of several binary classifiers.

A multiclass SVM classification method based on a single optimization process was introduced in [33]. A major advantage of this method is that the training of all the classes occurs in a single optimization step. Keerthi et al. [75] presented a dual method based on direct multiclass formulations of linear SVM. The main idea is to sequentially pass through the training dataset and optimize the dual variables associated with one example at a time. However a single step multiclass optimization is not practical to many classification applications due to the creation of a large optimization problem [44]. While directly extending a binary SVM into a multiclass SVM is not practical, a commonly used approach is to create a multiclass classifier based on the combination of binary classifiers. The One Against Rest (OAR) method is one of the popular methods to solve multiclass problems in which a binary classifier is trained for each class, which separates a single class from the rest of the classes and then combines the classifiers for multiclass inference. OVR can achieve high accuracy in classification [75] but the training process is not efficient due the involvement of all training data for creating binary classifiers for each class.

OAO method trains a binary classifier for each pair of classes. To classify an unlabelled instance, all binary classifiers are used. One advantage of the OAO method lies in its efficiency in training process. However, OAO does not perform well when the binary classifiers have different dataset in size.

An interesting solution is the use of error correcting output codes (ECOC) together with binary classifiers for solving multiclass classification problems [39]. Li et al. [88] combined different feature selection methods using ECOC strategies for multiclass cancer

classifications. One of the main limitations of the ECOC framework is the requirement of considering all classes for each binary classifier, hence is slow in training process. Platt et al. [114] introduced Directed Acyclic Graph SVM (DAGSVM) in which each node represents a classifier trained with the dataset of a pair of classes. DAGSVM depends on a rooted binary directed acyclic graph to make a decision on classifying unlabelled instances. However DAGSVM does not work well on an unequally distributed training data where the number of samples of each class is not equal.

To speed up multiclass SVM training, distributed computing paradigms have been investigated to partition a large training dataset into small data chunks and process each chunk in parallel utilizing the resources of a cluster of computers. Zhang et al. [155] presented a parallel multiclass SVM based on OAO using Message Passing Interface. Although the heterogeneity of multiclass training datasets is considered in their implementation, the scheduling of the computation tasks among multiple processors is based on a naive cyclically approach which does not consider the processing power of participating computing nodes. Additionally MPI is primarily targeted at homogeneous computing environments and has limited support for fault tolerance. Herrero-Lopez et al. [63] utilized GPU which is a specialized processing hardware. Here the authors considered a parallel multiclass SVM approach based on OAR exploiting the multi-threading capabilities of graphics processors. The results show a considerable decrease in processing time. Although the accuracy level of the GPU based SVM is comparable to the original OAO method, the training process is considerably less efficient.

Munoz- Mari et al. [105] presented a parallel SVM algorithm for multiclass problems based on OAO method using Medusa cluster [103] in a homogenous environment. Although the different sizes of classes in multiclass training datasets is considered in their implementation, however the scheduling of the computation tasks among multiple processors is simply to keep all the processor busy without considering the resources available on the underlying computing nodes. Lu et al. [94] presented a part-versus-part method to decompose a large multiclass classification problem into a number of two class sub-problems. A significant difference of the part-versus-part method with existing popular OAO multiclass classification approaches is that a large-scale two-class sub-problem can be further divided into a number of relatively smaller and balanced two-class sub-problems to increase training efficiency.

However the classification of par-versus-part method is slow in computation compared with the OAO classification approach due to the large number of support vectors to be processed.

2.5.4 SVM Ensemble

Ensemble methods represent one of the main current research issues in machine learning for improving classification accuracy [115]. Mason et al [101] show that ensembles enlarge the margins, consequently improve the generalization performances of learning algorithms while Schapire et al [117] present analysis of ensemble learning methods based on bias variance decomposition of classification error which shows that ensemble classifiers reduce variance and bias, therefore reducing the overall classification error rate.

Bagging is the most commonly used method for constructing ensemble classifiers. Bagging introduces randomness in the training data. Recently a number of SVM ensemble based on bagging have been proposed. Kim et al. [78] proposed SVM ensembles based bagging to improve the classification accuracy. The experimental results show improvement of classification accuracy of SVM ensemble. However, the experiments were performed with small datasets. This approach of ensemble learning is extremely computational intensive for large data set and large number of samples which limits their applications in real environments. Yan et al. [150] presented a SVMs ensemble method based on bagging. The results show the ensemble method performs better than a single SVM. The ensemble method involves tuning each of the base SVMs. However, the algorithm is evaluated using a small number of bootstrap samples, evaluating the algorithm with large number bootstrap samples is extremely computational intensive. Tao et al. [129] presented a SVMs ensemble method based on bagging and random subspace to improve the user relevance feedback performance in content-based image retrieval. The results show improvement in classification accuracy. However the ensemble method cannot guarantee diversity within SVMs base classifiers due to use only negative user feedback in the training process of SVMs.

Theoretical analysis of the performance of bagging in classification show that expected misclassification probability of bagging has the same bias component as a single bootstrap sample while the variance component is reduced significantly [54]. Valentini et al. [134] present a low bias SVMs ensemble based on bagging. The aim is to reduce bias of base SVMs before applying bagging. They consider the bias variance tradeoffs to improve the

classification accuracy of SVM ensemble. The experiments show improvement in classification accuracy. However, the idea was only tested on small datasets and no efficiency analysis was given. This approach of ensemble learning is also extremely computational intensive for large data set and large number of samples. Silva et al. [121] proposed a distributed SVM based ensemble system. Processing times is reported to have shown notable improvements over sequential approaches. Furthermore, the deployment of ensemble techniques improves classification performance in terms of accuracy. The system is evaluated using evaluate Condor and Alchemi middleware platforms.

Re et al. [115] evaluate the performance of several SVM ensemble, in which each base classifier is trained on different data types, the output are aggregated based on different combination methods. Their results show that heterogeneous data integration through ensemble methods is highly accurate for gene function prediction. Derbeko et al. [38] propose a new technique for aggregating SVM classifiers based on bootstrapping. In this method a linear combination of the base classifiers using weights are optimized to reduce variance. However efficiency of the ensemble is not analysed.

Lei et al. [86] propose the ensemble of support vector machines based on the bagging and boosting for text-independent speaker recognition, the experimental results show improvement of classification accuracy of SVM ensemble compare to single SVM. However, this approach of ensemble learning is extremely computational intensive for large data set and large number of samples which limits their applications in real environments. Tang et al. [128] applies bootstrapping to create samples from the original training dataset. An SVM is trained on each sample. The SVMs output are aggregated by Bayesian Sum Rule for a final decision. The algorithm is efficient and scalable. However there is slight reduction in the accuracy level compare to standard SVM.

2.7 Summary

Research on distributed SVM algorithms has been carried out from various dimensions, but mainly focuses on specialized SVM formulations, solvers and architectures [22] [58] [61] [67]. Although some progress has been made in speeding up SVM computation in training, existing approaches on high performance SVMs are mainly targeted at homogenous

computing environments using an MPI based solution. Scalability still remains a challenging issue for parallel SVM algorithms. These challenges motivated the design of RASMO which targets at a scalable SVM in heterogeneous computing environments empowered with a load balancing scheme.

Although some progress has been made in speeding up multiclass SVM computation in training, existing approaches on multiclass SVMs are mainly targeted at the classifications in which the classifiers have equal sizes of datasets deployed in homogenous computing environments without effective load balancing scheme. Scalability still remains a challenging issue for multiclass SVM classifications. These challenges motivate the design of RAMSMO which targets at a scalable multiclass SVM in heterogeneous computing environments empowered with a load balancing scheme.

Research on SVM ensemble algorithms has been carried out from various dimensions, but mainly focuses on improving classification accuracy, however solving the training inefficiency of SVM ensemble remains a huge challenge. This challenge motivates the design of MRESVM which is an efficient distributed SVM ensemble algorithm building on a highly scalable MapReduce implementation for image annotation with higher level classification accuracy compare to a single SVM.

This chapter started with briefly description of automatic image annotation techniques, distributed binary SVM, distributed multiclass SVM, distributed SVM ensemble and reviewing the related literatures. The chapter concluded and summarising the weakness of the existing techniques.

Chapter 3

Evaluation of Machine Learning Classifiers for Image Annotation

This chapter review seven representative machine learning classifiers for automatically image annotation. To facilitate performance evaluation, an image annotation prototype has been implemented which builds training models on low level features extracted from sample images. This chapter concludes on presenting the evaluation results.

3.1 Support Vector Machine (SVM)

SVM is based on creating a *hyperplane* as the decision plane, which separates the positive (+1) and negative (-1) classes with the largest margin. An optimal *hyperplane* is the one with the maximum margin of separation between the two classes, where the margin is the sum of the distances from the *hyperplane* to the closest data points of each of the two classes. These closest data points are called Support Vectors (SVs) [119]. Given a set of training data D, a set of points of the type $D = \{(x_i, c_i) \mid x_i \in \Re^p, c_i \in \{-1,1\}\}^n_{i=1}$, where c_i is either 1 or -1 indicative of the class to which the point x_i belongs, the aim is to give a maximum margin *hyperplane* which divide points having $c_i = 1$ from those having $c_i = -1$. Any *hyperplane* can be constructed as a set of point x satisfying $w.x - b = 0$. The vector w is a normal vector. We want to choose w and b to maximize the margin. These *hyperplanes* can be described by the following equations:

$$w.x - b = 1 \tag{3.1}$$

$$w.x - b = -1 \tag{3.2}$$

The margin $m = 1/\|w\|_2$.

The dual of the SVM is shown to be the following optimization problem:

Maximize (in α_i)

$$\sum_{i=1}^{n}\alpha_i - 1/2\sum_{i,j}\alpha_i\alpha_j y_i y_j x_i x_j \qquad (3.3)$$

Subject to $\alpha_i > 0$ and $\sum_{i=1}^{n}\alpha_i y_i = 0$

y_i indicates the class of an instance, there is a one-to-one association between each Lagrange multiplier α_i and each training example x_i. Once the Lagrange multipliers are determined, the normal vector $\vec{w}$ and the threshold b can be derived from the Lagrange multipliers as follow:

$$\vec{w} = \sum_{i=1}^{n} y_i a_i \vec{x}_i \qquad (3.4)$$

$$b = \vec{w}.\vec{x}_k - y_k \qquad (3.5)$$

for some $a_k > 0$. Not all data sets are linearly separable. There may be no *hyperplane* exist that separate separates the positive (+1) and negative (-1) classes. SVMs can be further generalized to non-linear classifiers. The output of a non-linear SVM is computed from the Lagrange multipliers as follow:

$$u = \sum_{i=1}^{n} y_i a_i K(X_i, X) + b \qquad (3.6)$$

where K is a kernel function that measures the similarity or distance between the input vector X_i and the stored training vector X.

3.2 Bayesian Networks

Formally, a Bayesian network is directed acyclic graphs in which the nodes represent variables and the edges encode conditional dependencies between the variables [7]. Let $U = \{x_1..x_n\}, n \geq 1$ be a set of variables. A Bayesian network B over a set of variables U is a network structure B_S. The classification job is to classify a variable $y = x_0$ called the class variable given a set of variables $x = x_1....x_n$ called attribute variables. A classifier $h : x \rightarrow y$ is

a function that maps an instance of x to a value of y. The classifier is learned from a dataset D consisting of samples over (x, y).To use a Bayesian network as a classifier, one simply calculates $\arg\max_y P(y|x)$ using the distribution $P(U)$represented by the Bayesian network. The advantage of using Bayesian Networks is that they can be used to reason in the two different directions. Another advantage of a Bayesian Network is the usefulness of the graph itself; the graph is a compact representation of the knowledge surrounding the system [53].

3.3 k Nearest Neighbour

The k Nearest Neighbour (kNN) algorithm is a non-parametric classifier. The training examples are vectors in a multi dimensional feature space. The space is partitioned into regions by locations and labels of the training samples. A point in the space is assigned to the class *c* if it is the most frequent class label among the *k* nearest training samples. The training stage of the algorithm only stores the feature vectors and class labels of the training samples. In the classification stage, a test sample is represented as a vector in the feature space. Distances from the new vector to all stored vectors are computed and *k* closest samples are selected. There are a number of ways to classify a new vector to a particular class. One of the widely used techniques is to predict the new vector to the most common class amongst the *k* nearest neighbors [10]. Non-parametric classifiers can naturally handle a huge number of classes, and avoid over fitting of parameters which is a central issue in learning based approaches. In addition, non-parametric classifiers do not require learning/training phases.

3.4 Artificial Neural Networks (ANN)

ANN consists of an interconnected group of artificial neurons and processes. The input to neuron consists of a number of values *x1, x2, ...x_n*, while output is single value *y*. Both input and output have continuous values, usually in the range (0, 1). The neuron computes the weighted sum of its inputs, subtracts some threshold *T*, and passes the result to a non-linear function *f*. Each element in ANN computes the following:

$$y = f\left(\sum_{i=1}^{N} w_i x_i - T\right) \tag{3.7}$$

where w_i are the weights. The outputs of some neurons are connected to inputs of other neurons. A multi-layer perceptron is especially useful for approximating a classification that maps input vector (x_1,x_2, ... x_n) to one or more classes C_1,C_2,...C_m. By optimizing weights and

thresholds for all nodes, the network can represent a wide range of classification functions. Optimizing the weights can be done by supervised learning, where the network learns from the large number of examples [64]. Shah and Gandhi [118] claim ANNs are useful because they can handle non-convex decisions. One disadvantage of ANNs is that the output values do not come with any confidence measure, inspecting specific features is highly nontrivial. A gross sense of confidence in a neural network approach can be found by ("winner takes all approach") determining the difference between the two largest outputs [118].

3.5 Composite Classifiers

In machine learning, a number of classifiers can be used together for high accuracy in classifications. They are proposed to improve the classification performance of a single classifier [127]. The combination makes it possible to complement the errors made by the individual classifiers on different parts of the input space.

3.5.1 Bagging

In the bagging technique, a number networks are trained separately by different training sets using the bootstrap method [15]. Bootstrapping builds *n* replicated training data sets by random re-sampling the original training data sets with replacements. Each training instance may appear repeatedly or not at all in any particular replicated training data set of *n*. Then, the *n* classifiers are combined using an appropriate combination method, such as majority voting. The most commonly used base classifier with bagging is Decision Tree.

3.5.2 Boosting

The boosting algorithm consists of iteratively learning weak classifiers with respect to a distribution and adding them to a final strong classifier [51]. When they are added, they are typically weighted in a way that is usually related to the weak learner's accuracy. After a weak learner is added, the data is reweighed: examples that are misclassified gain weight and examples that are classified correctly lose weight. Thus future weak learners focus more on the examples that previous weak learners misclassified. One of the main drawbacks of boosting algorithm is in its initial assumptions; hence a large number of training examples are required [52].

3.6. Performance Evaluation

To evaluate the performances of the 7 representative classifiers in image annotations, we have implemented a prototype system using Java programming language and the WEKA package version 3.5[144]. Figure 3.1 shows the structure of the system.

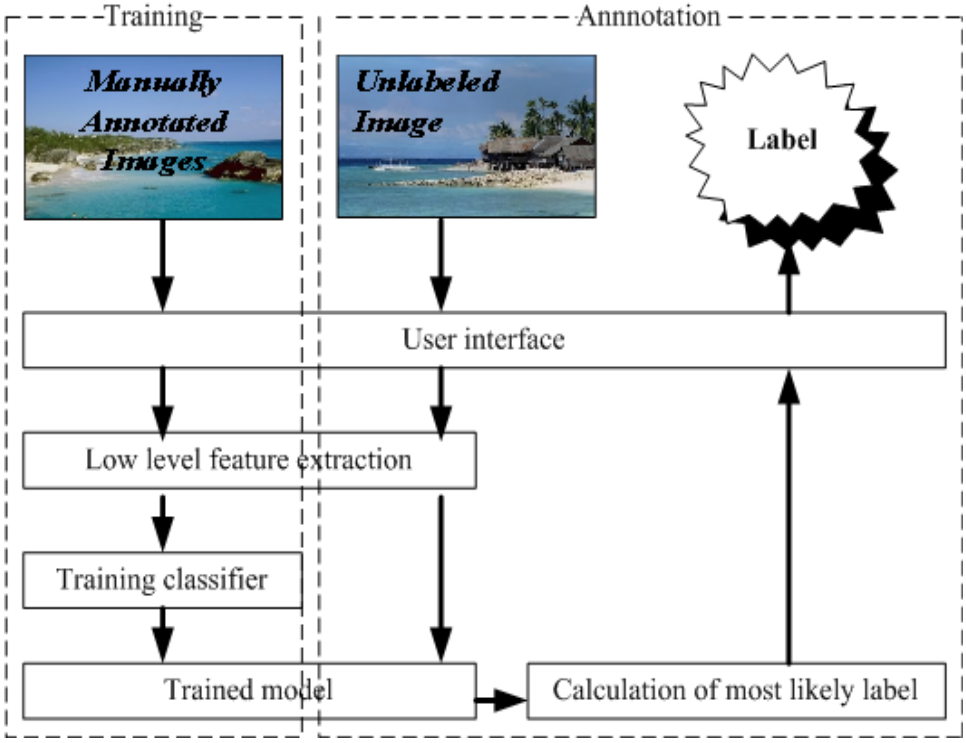

Figure 3.1: Image annotation system architecture.

The system learns the correspondence between low level visual features and image labels. Low-level MPEG-7 descriptors such as scalable colour [120] and edge histogram are used. The Edge Histogram Descriptor (EHD) proposed for MPEG-7 expresses the local edge distribution in an image. MPEG-7 edge histogram is designed to contain only 80 bins describing the local edge distribution [120]. The Scalable Colour Descriptor extracts a quantized HSV colour histogram from a given image. The probability values of each bin are calculated and indexed. The resulting histogram is transformed using a discrete Haar transformation, non-uniformly quantized and offset, and the resulting array of values is then sorted [120]. The image annotation systems can classify visual features into pre-defined classes. First images are segmented into blocks. Then, the low-level features are extracted from the segmented images. Each segmented block is represented by feature vectors. Next stage is to assign the low-level feature vectors to pre-defined categories. Training stage

requires choosing a classifier and create an empty training set, the classifier is fed with a set of training images in the form of attribute vectors with the associated labels After a model is trained, it is able to classify an unknown instance, into one of the learned class labels in the training set. Figure 3.2 shows the user interface of the prototype system which supports automatic annotation of images using 7 classifiers.

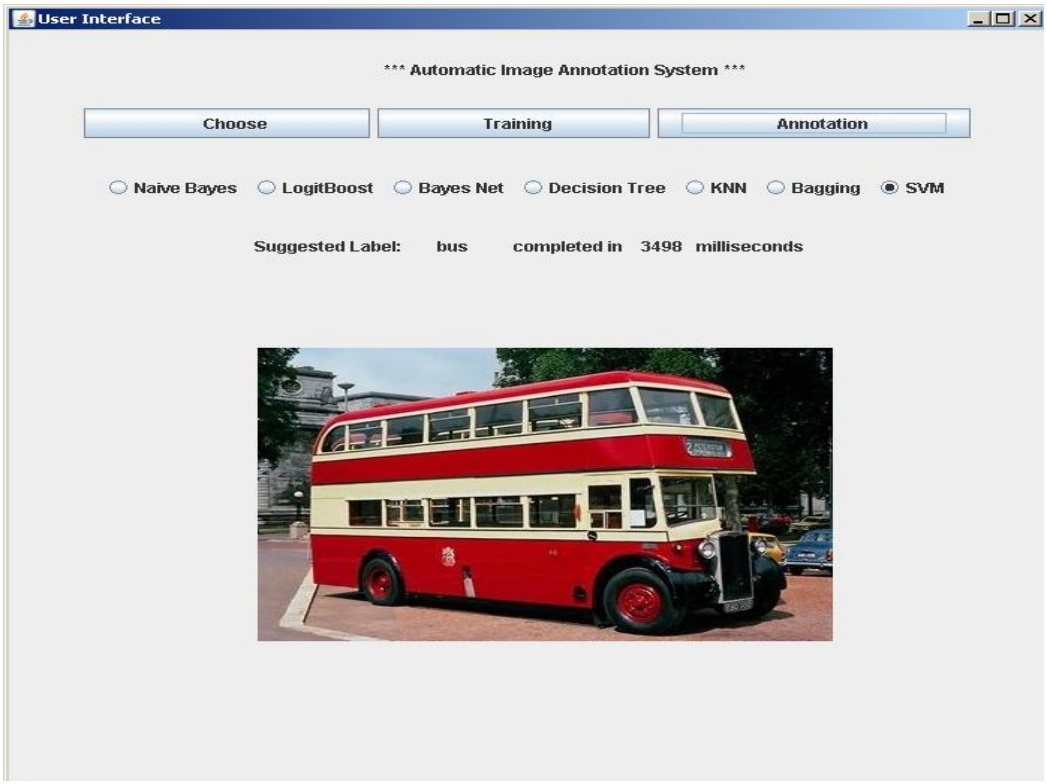

Figure 3.2: A snapshot of the system [32].

3.6.1 Preparing Training Images

The images are collected from the Corel database [32]. Images are classified into 10 classes, and each class of the images has one label associated with it. The 10 pre-defined labels are *people*, *beach*, *mountain*, *bus*, *food*, *dinosaur*, *elephant*, *horse*, *flower* and *historic item*. Typical images with 384x256 pixels are used in the training process. Low level features of the images are extracted using the LIRE (Lucene Image REtrieval) library [90]. After extracting low level features a typical image is represented in the following form:

0,256,12,1,-56,3,10,1,18,.....2,0,0,0,0,0,0,0,0,beach

Each image is represented by 483 attributes which include 58 attribute that represent edge histogram and 424 attributes represent Scalable Colour Descriptor and the last attribute indicates the class name which indicates the category to which the image belongs to. Figure 3.3 shows some of the sample images used in training classifiers.

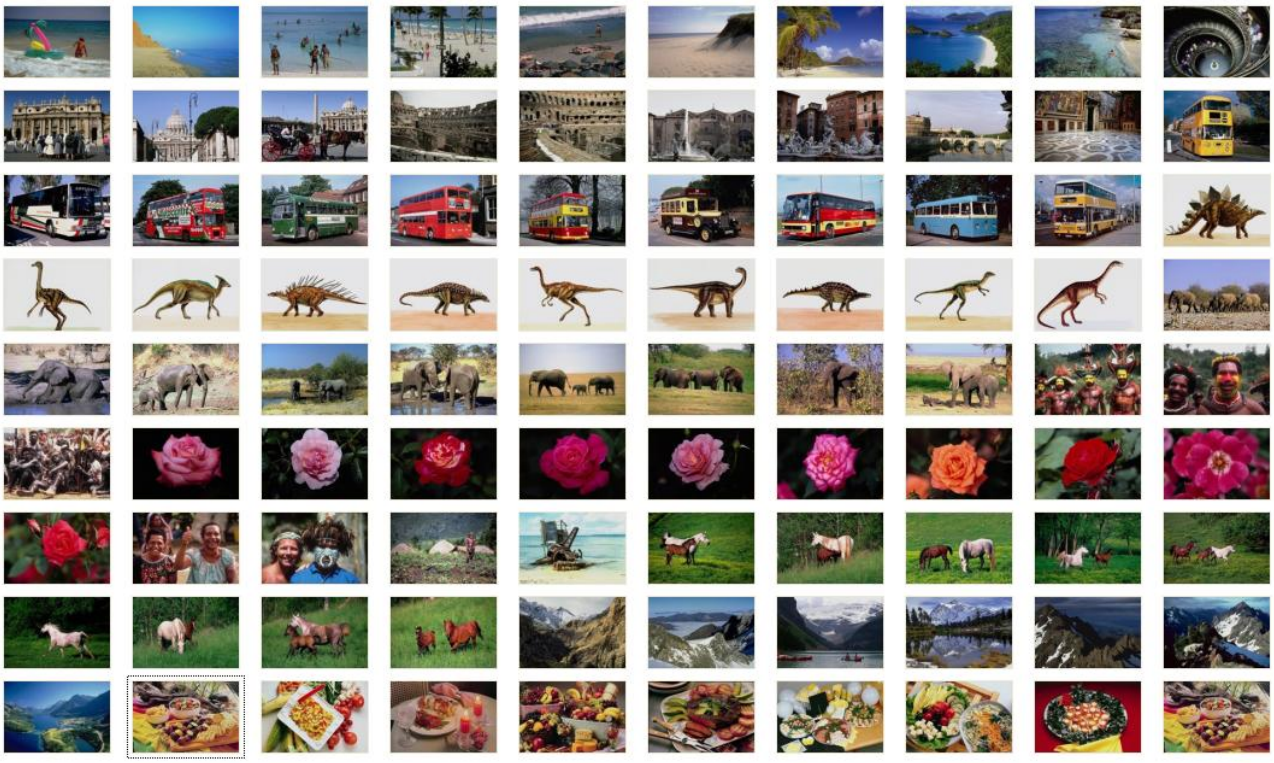

Figure 3.3: Sample images [32].

3.6.2 Experiment Results

A number of tests were carried out on a Dell computer, Microsoft Vista, RAM- 1.00 GB, Processor-520 @1.60Ghz. The 7 classifiers were evaluated from the aspects of accuracy in annotating images and efficiency in training the models. In total 50 unlabeled images were tested (10 images at a time), the average accuracy level was considered. Figure 3.4 shows the accuracy of the 7 classifiers increases when the numbers of sample images are increased in the training process.

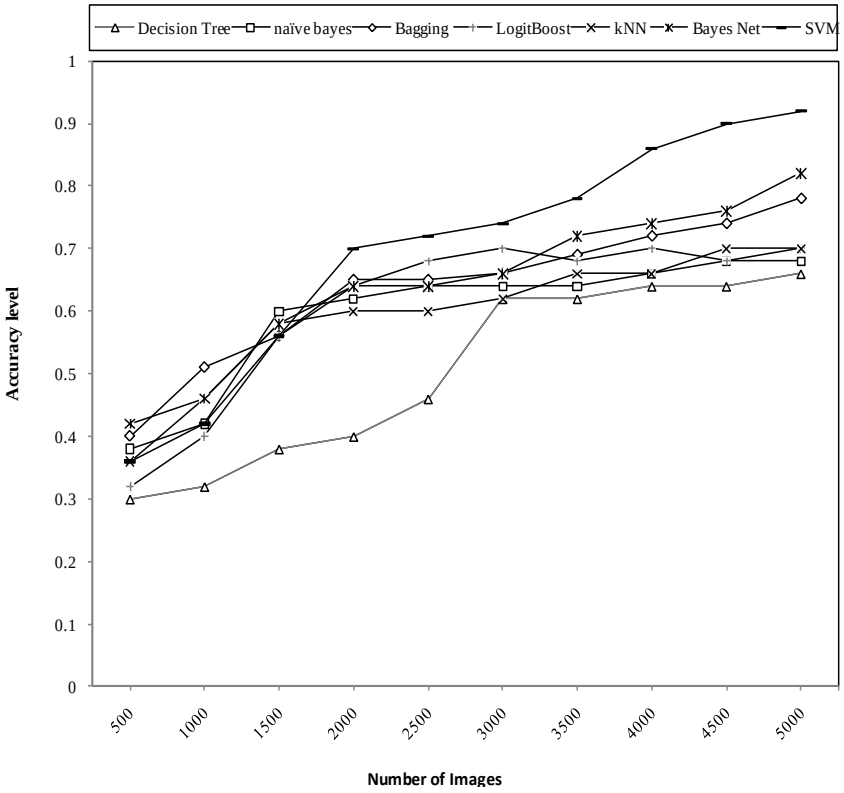

Figure 3.4: Accuracy in image annotations.

Among the 7 classifiers, SVM performs the best producing most accurate results in annotating images. SVM achieves a level of accuracy over 90% when 5000 images are used in the training. SVM accuracy level is due to its high generalization performance without the need to add a priori knowledge, even when the dimension of the input space is very high. The ability of a classifier to correctly classify data not in the training set is known as its generalization [119]. The decision tree C4.5 algorithm performs the worst with a level of accuracy of just 70%. The low level of accuracy is possibly due to the instability of the decision tree algorithm. Slight variations in the training data can result it different attribute selections at each choice point within the tree [157]. The effect can be significant since attribute choices affect all descendent sub trees.

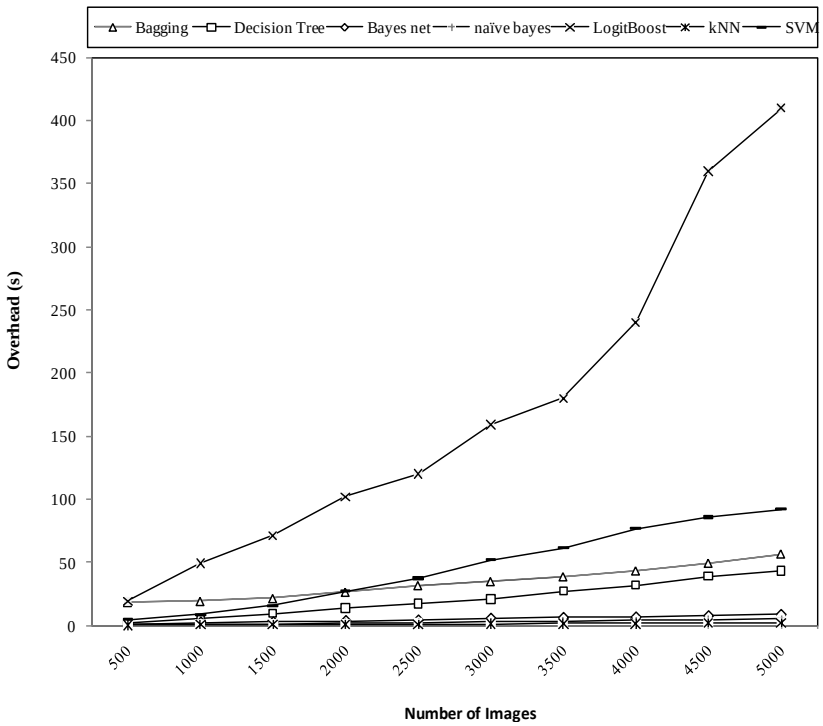

Figure 3.5: Overheads in training models.

However, from the results presented in Figure 3.5 we observe that SVM incurs one of the highest overhead in training the model. Training a SVM is equivalent to solve a quadratic programming problem with linear and constraints in a number of variables equal to the number of data points [110]. The training time of SVM can increase to almost 100 seconds even though the sequential minimal optimization (SMO) [113] is used, a fast algorithm for training SVM models.

3.7 Summary

This chapter started with the review of seven representative machine learning classifiers for automatically image annotation. An image annotation prototype was presented which builds training models on low level features extracted from sample images. This chapter concluded on presenting the evaluation results.

Chapter 4

Resource Aware Parallel SVM for Scalable Image Annotation

This chapter presents RASMO, a resource aware parallel SVM algorithm for large scale image annotation which partitions the training data set into smaller subsets and optimizes SVM training in parallel using a cluster of computers. A genetic algorithm based load balancing scheme is designed to optimize the performance of RASMO in heterogeneous computing environments.

4.1 The design of RASMO

This section starts with a brief description of the SMO algorithm followed by a detailed description of RASMO.

4.1.1 SMO Algorithm

The SMO algorithm was developed by Platt [113] and further enhanced by Keerthi et al. [74]. Platt takes the decomposition to the extreme by selecting a set of only two points as the working set which is the minimum due to the following condition:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{4.1}$$

where a_i is a Lagrange multiplier and *y* is a class name. This allows the sub-problems to have an analytical solution. Despite the need for a number of iteration to converge, each iteration only uses a few operations. Therefore the algorithm shows an overall speedup of some orders of magnitude [119]. The SMO has been recognized as one of the fastest SVM algorithms available. We define an index set *I* which denotes the following training data patterns:

$I_0 = \{i : y_i = 1, 0 < a_i < c\} \cup \{i : y_i = -1, 0 < a_i < c\}$

$I_1 = \{i : y_i = 1, a_i = 0\}$ (Positive Non-Support Vectors)

$I_2 = \{i : y_i = -1, a_i = c\}$ (Bound Negative Support Vectors)

$I_3 = \{i : y_i = 1, a_i = c\}$ (Bound Positive Support Vectors)

$I_4 = \{i : y_i = -1, a_i = 0\}$ (Negative Non-Support Vectors)

where *c* is the correction parameter. Bias b_{up} and b_{low} are defined with their associated indices as follows:

$b_{up} = \min\{f_i : i \in I_0 \cup I_1 \cup I_2\}$

$I_{up} = \arg\min_i f_i$

$b_{low} = \max\{f_i : i \in I_0 \cup I_3 \cup I_4\}$

$I_{low} = \arg\max_i f_i$

The optimality conditions are tracked through the vector f_i in equation (4.2).

$$f_i = \sum_{j=1}^{l} a_j y_j K(X_j, X_i) - y_i \tag{4.2}$$

where *K* is a kernel function and X_i is a training data point. SMO optimizes two a_i values related to b_{up} and b_{low} according to equation (4.3) and equation (4.4).

$$a_2^{new} = a_2^{old} - y_2\left(f_1^{old} - f_2^{old}\right)/\eta \tag{4.3}$$

$$a_1^{new} = a_1^{old} - s\left(a_2^{old} - a_2^{new}\right) \tag{4.4}$$

where $\eta = 2k(X_1, X_2) - k(X_1, X_1) - k(X_2, X_2)$. After optimizing a_1 and a_2, f_i which denotes the error of the i th training data can be updated according to equation (4.5).

$$f_i^{new} = f_i^{old} + (a_1^{new} - a_1^{old}) y_1 k(X_1, X_i) + (a_2^{new} - a_2^{old}) y_2 k(X_2, X_i) \quad (4.5)$$

To build a linear SVM, a single weight vector needs to be stored instead of all the training examples that correspond to non-zero Lagrange multipliers. If the joint optimization is successful, the stored weight vector needs to be updated to reflect the new Lagrange multiplier values. The weight vector is updated according to equation (4.6).

$$\vec{w}^{new} = \vec{w} + y_1 (a_1^{new} - a) \vec{x}_1 + y_2 (a_2^{new,clipped} - a_2) \vec{x} \quad (4.6)$$

We check the optimality of the solution by calculating the optimality gap between the b_{low} and b_{up}. The algorithm is terminated when $b_{low} \leq b_{up} + 2\tau$ as shown in Algorithm 4.1.

Algorithm 4.1: Sequential Minimal Optimization Algorithm

Input: training data x_i, labels y_i,
Output: sum of weight vector, α array, b and SV

1: Initialize: $\alpha_i = 0$, $f_i = -y_i$
2: Compute: b_{high}, I_{high}, b_{low}, I_{low}
3: Update α_{Ihigh} *and* α_{Ilow}
4: repeat
5: Update f_i
6: Compute: b_{high}, I_{high}, b_{low}, I_{low}
7: Update α_{Ihigh} *and* α_{Ilow}
8: until $b_{low} \leq b_{up} + 2\tau$
9: Update the threshold *b*
10: Store the new α1 and α2 values
11: Update weight vector *w* if SVM is linear

4.1.2 Cascade SVM

SVM training can be speeded up by splitting the training data set into a number of smaller data chunks and trained separately with multiple SVMs. When the training process is completed, the generated training vectors have support vectors and non-support vectors.

Removing the non-support vectors in an early stage in the training process is an effective strategy in speeding up SVM. The multilayered cascade architecture follows such an approach until a global optimum is reached. The SVM classifiers can be considered as the nodes in a binary tree. Figure 4.1 shows an example of a cascade SVM.

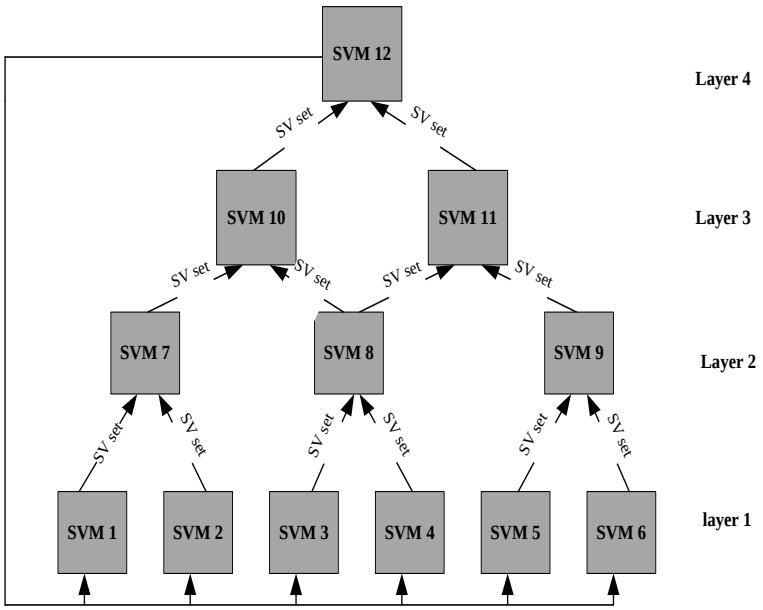

Figure 4.1: A cascade SVM example.

In this architecture a single SVM is trained with a smaller data chunk. The support vectors generated from one layer are combined as input for the next layer. The cascade architecture is guaranteed to converge to a global optimum as the support vectors of the last layer are fed back into the SVMs in the first layer to determine the level of convergence.

4.2 The RASMO Algorithm

RASMO builds on MapReduce for parallelization of SVM computation in training. This section starts by a brief description of the MapReduce programming model followed by a detailed description of the RASMO algorithm.

4.2.1 MapReduce Model

MapReduce provides an efficient programming model for processing large data sets in a parallel and distributed manner. The Google File System [137] that underlies MapReduce provides an efficient and reliable data management in a distributed computing environment.

The basic function of MapReduce model is to iterate over the input, compute key/value pairs from each part of input, group all intermediate values by key, then iterate over the resulting groups and finally reduce each group. The model efficiently supports parallelism. Figure 4.2 presents an abstraction of a typical MapReduce framework. *Map* is an initial transformation step, in which individual input records are processed in parallel. The system shuffle and sort the map outputs and transfer them to the reducers. *Reduce* is a summarization step, in which all associated records are processed together by a single entity.

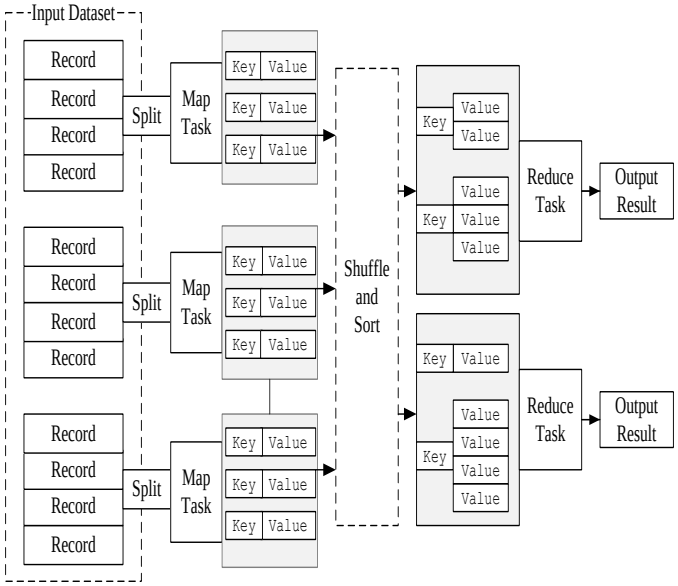

Figure 4.2: The MapReduce model.

4.2.2 RASMO Design

The RASMO algorithm partitions the entire training data set into smaller data chunks and assigns each data chunk to a single *map* task. The number of *map* tasks is equal to the number data chunks. Each *map* function optimizes a data chunk in parallel in each layer. The output of each *map* function is the *alpha* array (Lagrange multipliers) for a local partition and the training data X_i which corresponds Lagrange multipliers $\alpha_i > 0$ in order to create input for the next layer, the output of the last layer includes the *alpha* array, bias threshold *b* and the training data X_i which correspond $\alpha_i > 0$ in order to calculate the SVM output *u* using equation (4.7).

$$u = \sum_{i=1}^{n} y_i a_i K(X_i, X) + b \tag{4.7}$$

where X is an instance to be classified, y_i is class labels for X_i and K is the kernel function.

Each *map* task processes the associated data chunk and generates a set of support sectors. Each set of support sectors is then combined and forwarded to the *map* task in the next layer as input. The process continues until a single set of support sectors is computed. The set of support sectors of the last layer is then fed back into the first layer together with non-support vectors to determine the level of convergence. The entire process stops until a global optimum is reached indicating that no further optimization is needed in the first layer, and the generated SVM model will be used in the classification. Figure 4.3 presents a high level pictorial representation of this approach, in part similar to the approach adopted in [58].

Algorithm 4.2 shows the pseudo code of RASMO with a 3 layers structure. Lines 1-4 show the optimization process of SMO for each data chunk and combine support vectors of layer 1. Lines 5-8 show the assembling results from layer 1 which are used as input for layer 2. Lines 9-12 show the assembling results from layer 2 which are used as input for layer 1, and the training process in layer 3.

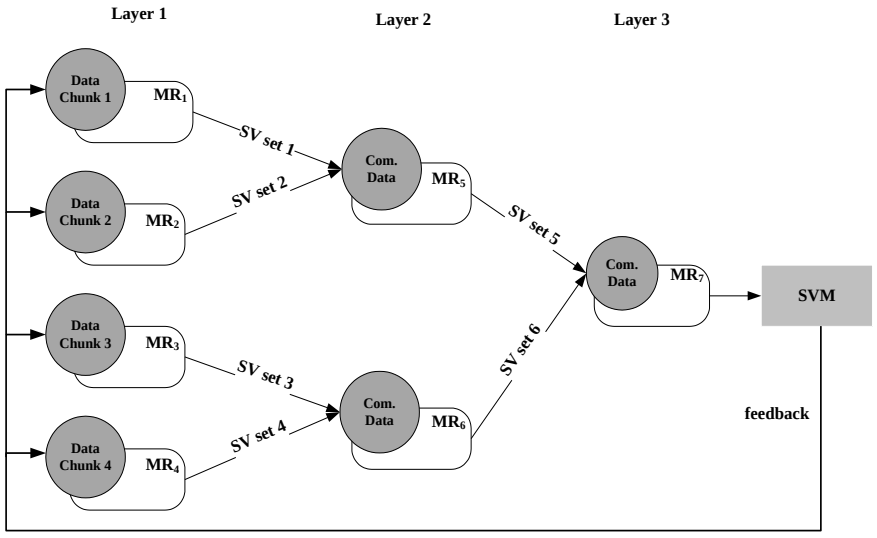

Figure 4.3: The architecture of RASMO.

Algorithm 4.2: RASMO Algorithm

Map tasks
Input: training data x_i
Output: support vectors sv_i , b and data x_l
1: train SMO on *m* chunks;
2: obtain sv_m set for *m* chunks; $sv_m = \{\alpha_m > 0\}$;
3: combine each two sv_m sets ;
4: store all x_m for sv_m to create *k* input chunks for next layer *Map* tasks;
5: train SMO on *k* chunks;
6: obtain sv_k set for *k* chunks; $sv_k = \{\alpha_k > 0\}$
7: combine two sv_k sets;
8: store all x_k for sv_k to create input chunk for next layer *Map* task;
9: train SMO on x_k
6: obtain sv_i set for x_k ; $sv_i = \{\alpha_i > 0\}$
10: evaluate sv_i for global convergence;
12: store the final set sv_i if further optimization is not required

4.3 Load Balancing

A remarkable characteristic of the MapReduce Hadoop framework is its support for heterogeneous computing environments. Therefore computing nodes with varied processing capabilities can be utilized to run MapReduce applications in parallel. However, current implementation of Hadoop only employs first-in-first-out (FIFO) and fair scheduling with no support for load balancing taking into consideration the varied resources of computers. A genetic algorithm based load balancing scheme is designed to optimize the performance of RASMO in heterogeneous computing environments.

To solve an optimization problem, genetic algorithm solutions need to be represented as chromosomes encoded as a set of strings which are normally binary strings. However, a binary representation is not feasible as the number of *map* instances (operations) in a Hadoop cluster environment is normally large which will result in long binary strings. A decimal string has been employed to represent a chromosome in which the data chunk assigned to a *map* instance (also called a *mapper*) is represented as a gene. The numbers of gene are defined based on the number of available *mappers*. The crossover rate of the genetic algorithm is 0.9 and the mutation rate is 0.01.

However simply crossing the chromosome can be problematic. As each gene is the value of the actual volume of data each Map instance takes, to change the members of genes may differentiate the original total volume of data $\sum_{i=1}^{k} D_i$. Assume the original total volume of data is $\sum_{i=1}^{k} D_i$ and the volume of data after crossover is $\sum_{i=1}^{k} d_i$, then the difference $\Delta D = \left| \sum_{i=1}^{k} D_i - \sum_{i=1}^{k} d_i \right|$ should be considered and processed, ΔD is divided into k parts. The size of each part is randomly assigned. And then these k parts will be randomly added to or removed from k genes in the chromosome.

In Hadoop, the total time (T) of a *mapper* in processing a data chunk consists of the following four parts:

- Data copying time (t_c) in copying a data chunk from Hadoop distributed file system to local hard disk. It depends on the available network bandwidth and the writing speed of hard disk.
- Processor running time (t_p) in processing a data chunk.
- Intermediate data merging time (t_m) in combining the output files of the *mapper* into one file for *reduce* operations.
- Buffer spilling time (t_b) in emptying a filled buffer.

$$T = t_c + t_p + t_m + t_b \tag{4.8}$$

Let

- D_m be the size of the data chunk.
- H_d be the writing speed of hard disk in MB/second.
- B_w be the network bandwidth in MB/second.
- P_r be the speed of the processor running the *mapper* process in MB/second.
- B_f be the size of the buffer of the *mapper*.
- R_a be the ratio of the size of the intermediate data to the size of the data chunk.

- N_f be the number of frequencies in processing intermediate data.
- N_b be the number of times that buffer is filled up.
- V_b be the volume of data processed by the processor when the buffer is filled up.
- S be the sort factor of Hadoop.

We have

$$t_c = \frac{D_m}{\min(H_d, B_w)} \tag{4.9}$$

Here t_c depends on the available resources of hard disk and network bandwidth. The slower one of the two factors will be the bottleneck in copying data chunks from Hadoop distributed file system to the local hard disk of the *mapper*.

$$t_p = \frac{D_m}{P_r} \tag{4.10}$$

When a buffer is filling, the processor keeps writing intermediate data into the buffer and in the mean time the spilling process keeps writing the sorted data from the buffer to hard disk. Therefore the filling speed of a buffer can be represented by $P_r \times R_a - H_d$. Thus the time to fill up a buffer can be represented by $\frac{B_f}{P_r \times R_a - H_d}$. As a result, for a buffer to be filled up, the processor will generate a volume of intermediate data with the size of V_b which can be computed using equation (4.11)

$$V_b = P_r \times R_a \times \frac{B_f}{P_r \times R_a - H_d} \tag{4.11}$$

The total amount of intermediate data generated from the original data chunk with a size of D_m is $D_m \times R_a$. Therefore the number of times for a buffer to be filled up can be computed using equation (4.12).

$$N_b = \frac{D_m \times R_a}{V_b} \tag{4.12}$$

The time for a buffer to be spilled once is $\frac{B_f}{H_d}$, therefore the time for a buffer to be spilled N_b times is $\frac{N_b \times B_f}{H_d}$. Then we have

$$t_b = \frac{N_b \times B_f}{H_d} \tag{4.13}$$

The frequencies in processing intermediate data N_f can be computed using equation (4.14).

$$N_f = \left\lfloor \frac{N_b}{s} \right\rfloor - 1 \tag{4.14}$$

When the merging occurs once, the whole volume of intermediate data will be written to the hard disk causing an overhead of $\frac{D_m \times R_a}{H_d}$. Thus if the merging occurs N_f times, the time consumed by hard disk IO operations can be computed by $\frac{D_m \times R_a \times N_f}{H_d}$. We have

$$t_m = \frac{D_m \times R_a \times N_f}{H_d} \tag{4.15}$$

The total time T_{total} to process data chunks in one processing wave in Hadoop is the maximum time consumed by k participating *mappers*:

$$T_{total} = \max(T_1, T_2, T_3, ..., T_k) \tag{4.16}$$

According to divisible load theory, to achieve a minimum T_{total}, it is expected that all the *mappers* to complete data processing at the same time:

$$T_1 = T_2 = T_3 = ..., T_k \tag{4.17}$$

Let

- T_i be the processing time for the i^{th} *mapper*.
- $\overline{T}$ be the average time of the k *mappers* in data processing, $\overline{T} = \frac{\sum_{i=1}^{k} T_i}{k}$.

According to equations (4.16) and (4.17), the fitness function is to measure the distance between T_i and $\overline{T}$. Therefore, the fitness function can be defined using equation (4.18) which is used by the genetic algorithm in finding an optimal or a near optimal solution in determining the size for a data chunk.

$$f(T) = \sqrt{\sum_{i=1}^{k} (\overline{T} - T_i)^2} \qquad (4.18)$$

4.4 Experimental results

RASMO has been incorporated into our image annotation system which is developed using the Java programming language and the WEKA package. The image annotation system classifies visual features into pre-defined classes. Figure 4.4 shows a snapshot of the system.

Figure 4.4: A snapshot of the image annotation system [32].

4.4.1 Image Corpora

The images are collected from the Corel database. Images are classified into 10 classes, and each class of the images has one label associated with it. The 10 pre-defined labels are *people*, *beach*, *mountain*, *bus*, *food*, *dinosaur*, *elephant*, *horse*, *flower* and *historic item*. Typical images with 384x256 pixels are used in the training process. Low level features of the images are extracted using the LIRE (Lucene Image REtrieval) library. After extracting low level features a typical image is represented in the following form:

0,256,12,1,-56,3,10,1,18,.....2,0,0,0,0,0,0,0,0,beach

Each image is represented by 483 attributes which include 58 attribute that represent edge histogram and 424 attributes represent Scalable Colour Descriptor and the last attribute indicates the class name which indicates the category to which the image belongs to.

4.4.2 Performance Evaluation

RASMO is implemented using Weka's base machine learning libraries written in the Java programming language and tested in a Hadoop cluster. To evaluate RASMO, the SMO algorithm provided in the Weka package, has been extended, configured and packaged it as a basic MapReduce job. The Hadoop cluster for this set of experiments consist of a total of 12 physical cores across 3 computer nodes as shown in Table 4.1.

Table 4.1: Hadoop Configuration.

Hardware environment			
	CPU	Number of Cores	RAM
Node 1	Intel Quad Core	4	4GB
Node 2	Intel Quad Core	4	4GB
Node 3	Intel Quad Core	4	4GB
Software environment			
SVM	WEKA 3.6.0 (SMO)		
OS	Fedora10		
Hadoop	Hadoop 0.20		
Java	JDK 1.6		

The performance of RASMO has been evaluated from the aspects of efficiency and accuracy. Polynomial kernel function has been used in the experiments. Figure 4.5 shows the efficiency of the RASMO in SVM training which achieves close to 12 times in speedup.

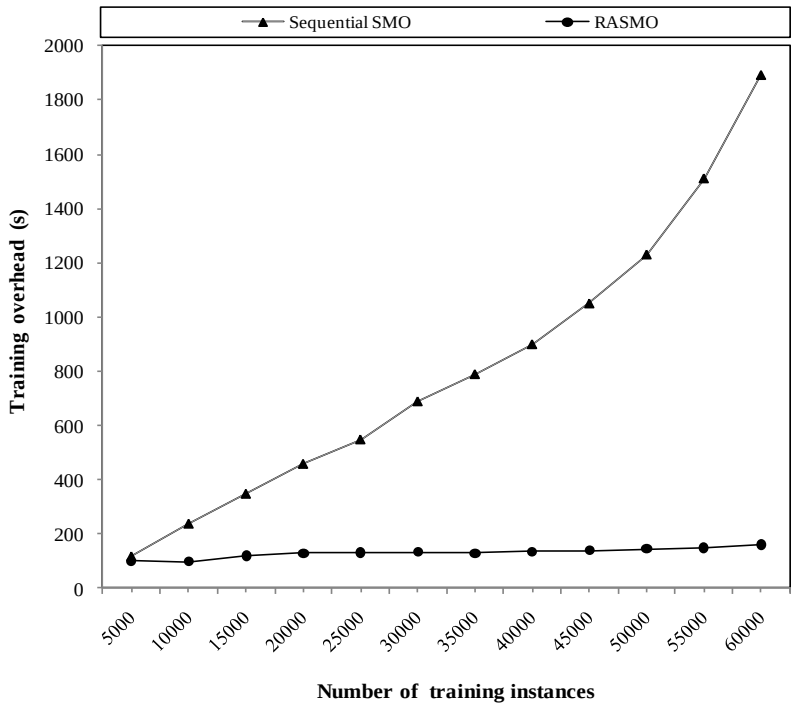

Figure 4.5: The efficiency of RASMO using 12 *mappers*.

Figure 4.6 shows the efficiency of the RASMO in SVM training in two iterations which converge to the global optimum.

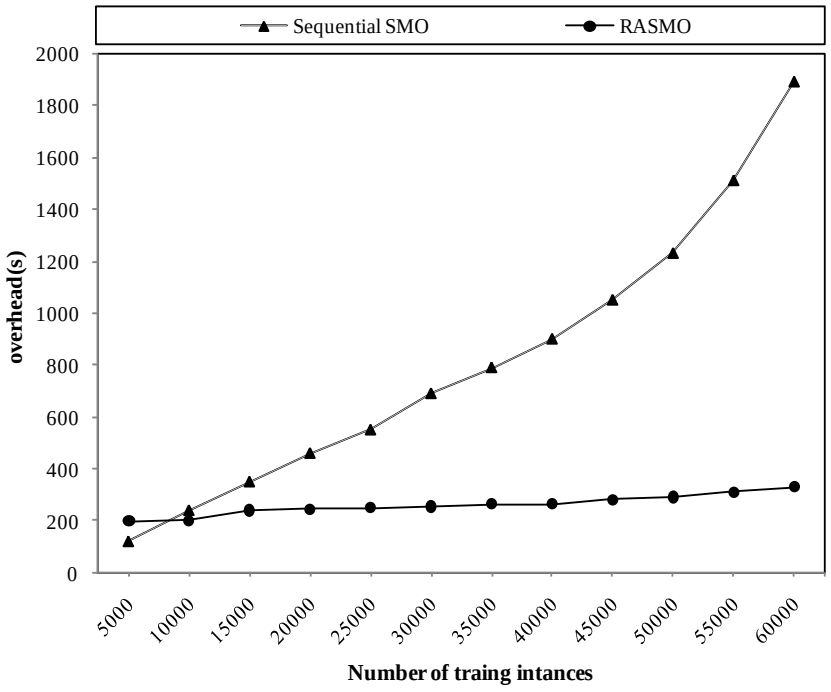

Figure 4.6: The efficiency of fully converge RASMO using 12 *mappers*.

The experiments demonstrated that Hadoop startup and the associated overhead introduce performance penalties for the cases with smaller numbers of training instances. However, RASMO starts to outperform the sequential SMO with an increasing number of instances in terms of training time required. Figure 4.7 shows the increasing efficiency with the number of participating MapReduce *mappers* varying from 4 to 12.

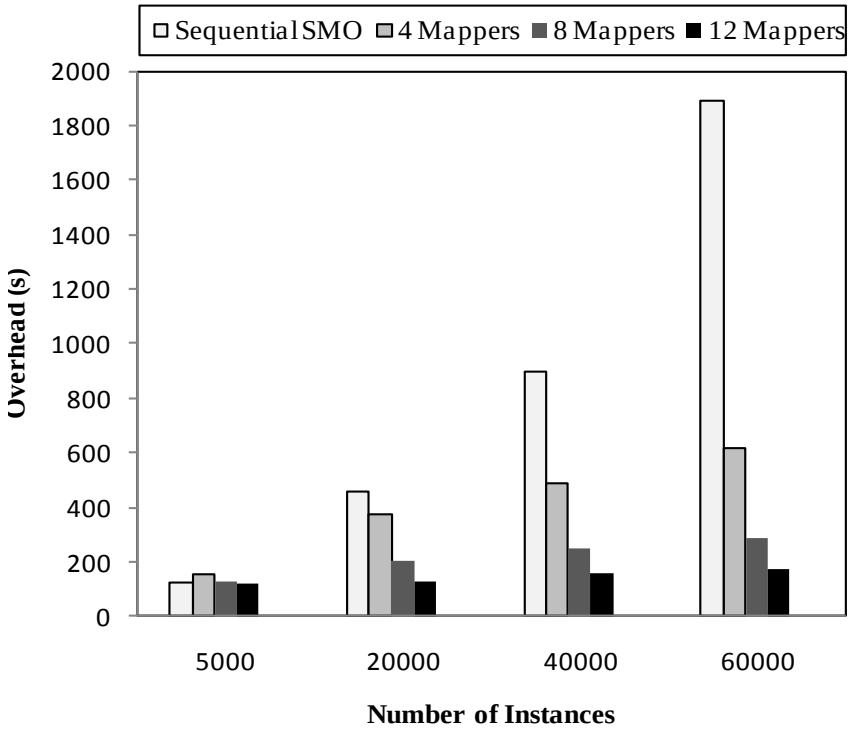

Figure 4.7: The overhead of RASMO.

Furthermore the accuracy of the sequential SMO and RASMO have been evaluated in classification and presented the results in Table 4.2 using 5000 instances. In total 50 unlabeled images were tested (10 images at a time), the average accuracy level was considered. It is clear that the parallelization of RASMO has no affect on the accuracy level even after the first iteration which is close to global optimum. The results show that RASMO achieves 94% which was the same as the sequential SMO.

Table 4.2: Summarized performance results.

	Sequential SMO	**RASMO 12 *Mappers***
Correctly Classified	≈ 94 %	≈ 94 %
Incorrectly Classified	≈ 6%	≈ 6%
Training time	240 (s)	34 (s)

4.5 Simulation results

To further evaluate the effectiveness of RASMO in large scale MapReduce environments, HSim, a MapReduce Hadoop simulator has been implemented using the Java programming language by a research group which I was a member. In this section, the design of HSim is briefly presented and the performance of the RASMO in simulation environments is assessed.

4.5.1 Simulator Design

HSim follows a master-slave mode in its design. Parameters related to a simulated cluster include the number of Hadoop nodes, the topologies of these nodes (currently only supporting simple racks), the number of *mappers* and *reducers*, the CPU speed, memory size, the average reading and writing speeds of hard disk and network bandwidth of each node. HSim supports one processor per node and each processor can have one or more processor cores. The processing speed of each core is defined as the volume of data processed per second. The values of some parameters such as CPU speed and the writing and reading speeds of hard disk can be assigned based on measurements from real-world experiments. Each job in HSim has a job ID which is used for job tracking. The size of a job is the total size of input data. The MapOutputRatio parameter represents the volume of intermediate data that will need to be generated by *map* instances. The NumberOfChunk parameter specifies the number of splits to be used in the *map* process which is related to the number of *mappers*. The Number of Reducers specifies the number of *reduce* instances. Figure 4.8 shows the architecture of HSim.

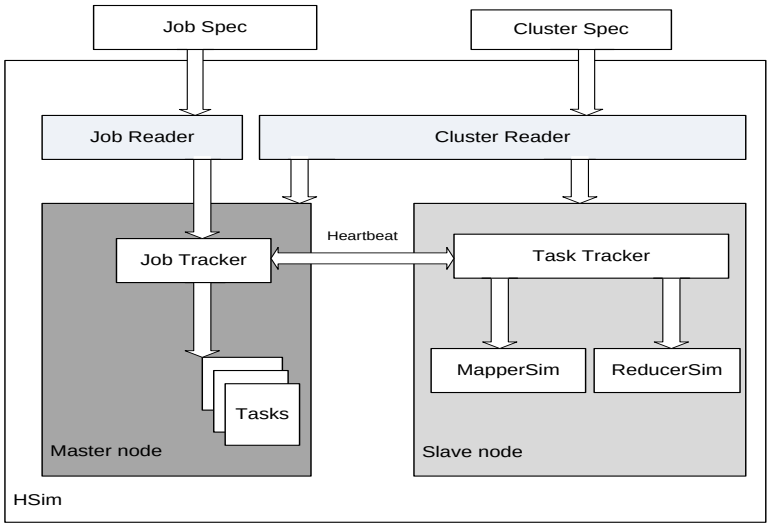

Figure 4.8: HSim Architecture.

When a job is submitted to the simulated Hadoop cluster, the JobTracker splits the job into several tasks. Each task will be assigned to a *map* instance. The TaskTrackers and JobTracker communicate with each other via heartbeat based messaging. When all the *map* tasks have finished, the *reduce* instances will be notified to be prepared for merging. Each *map* instance, called a *mapper,* is simulated by the MapperSim component. For a simulation job, MapperSim reads the input data in the form of chunks, processes the job, generates a number of output data splits and subsequently performs a sort and merge process based on the keys of the input data chunks. Finally, MapperSim splits the output dataset based on the number of *reducers* specified in the job configuration. The ReducerSim component collects output data splits from the MapperSim component and performs a merge process generating a single output result.

4.5.2 Validation of HSim with Benchmarks

For validation, HSim is evaluated against the benchmark results presented in [111] using 3 scenarios - Grep Task, Selection Task and UDF Aggregation Task. In HSim, the exact physical environments adopted in the benchmarking have been simulated.

A. Grep Task

A cluster with 1 node, 10 nodes, 25 nodes, 50 nodes and 100 nodes respectively are simulated. 2 scenarios are tested. In the first scenario 535MB of data is assigned to each node. In the second scenario, 1TB of data is submitted to the cluster. Each scenario was evaluated 5 times. The simulation results of the 2 scenarios are plotted in Figure 4.9 and Figure 4.10 respectively which are close to the benchmark results. The confidence intervals of the results are small in both scenarios (between 0 and 2.6 seconds in the first scenario and between 4.1 and 7.6 seconds in the second scenario) demonstrating a high stability of HSim in performance.

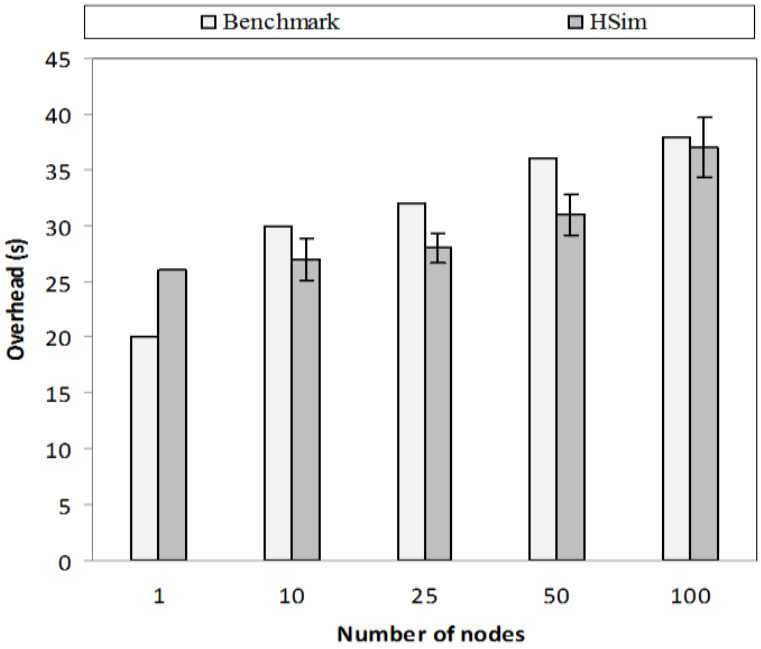

Figure 4.9: Grep Task evaluation (533MB/node).

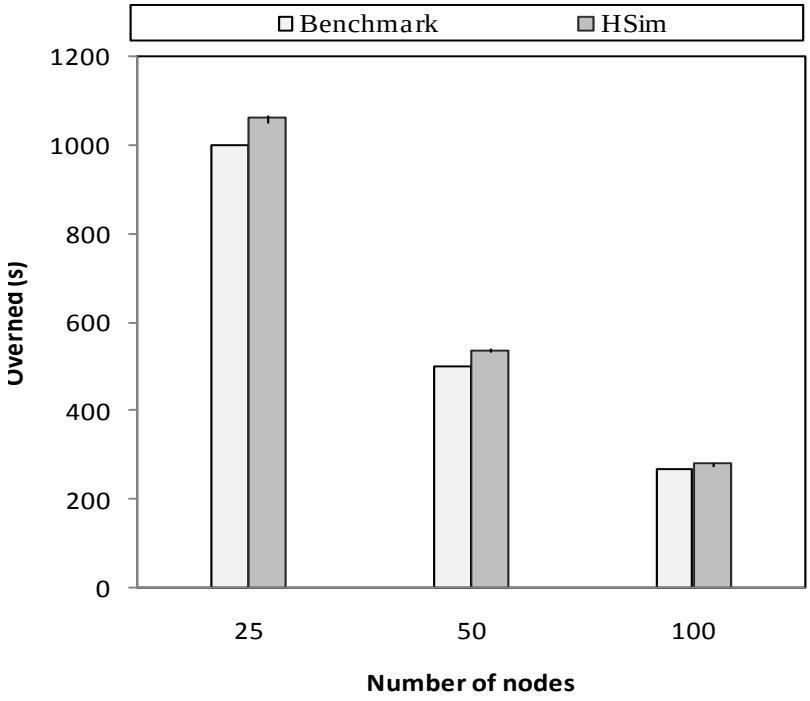

Figure 4.10: Grep Task evaluation (1TB/cluster).

B. Selection Task

The Selection Task scenario was designed to observe the performances of the Hadoop framework in dealing with complex tasks. Each node processes one 1GB ranking table to retrieve the target pageURLs using a user defined threshold. The simulation results shown in Figure 4.11 are again close to the benchmark results with small confidence intervals in the range between 2.6 and 6.6 seconds.

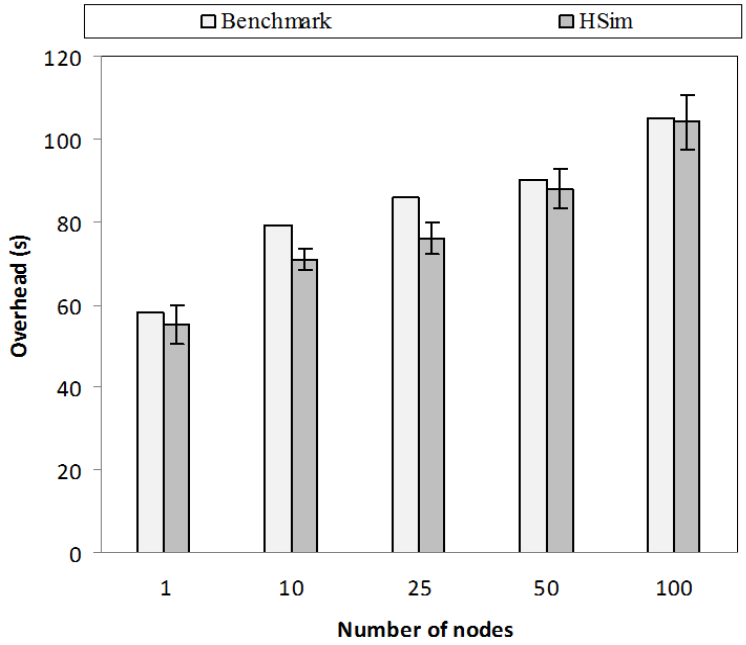

Figure 4.11: Selection Task evaluation.

C. UDF Aggregation Task

The UDF Aggregation Task reads the generated document files and searches for all the URLs appearing in the content. For each unique URL, the system counts the number of unique pages that refer to that particular URL across the entire set of files. Each node processes around 7GB documents. Figure 4.12 shows the simulation results in the respective scenario, which are also close to benchmark results with small confidence intervals in the range between 2.6 and 13.4 seconds.

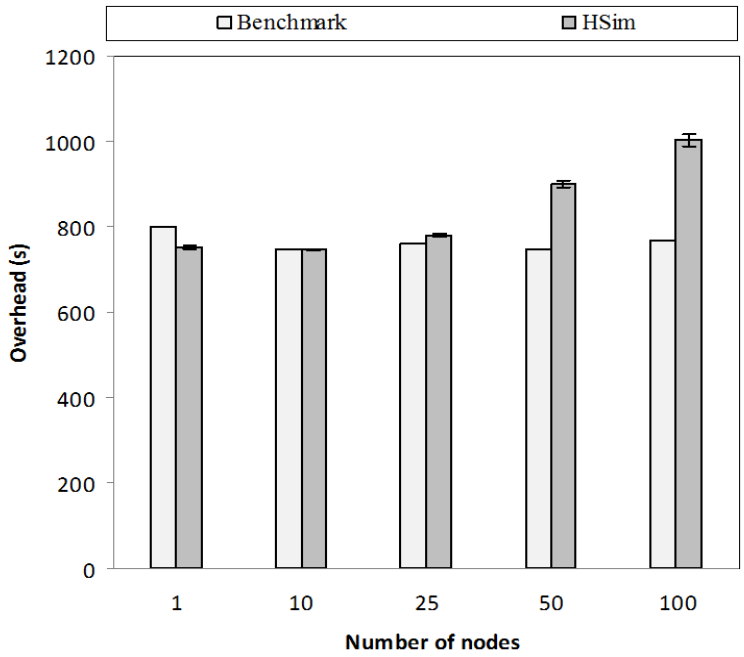

Figure 4.12: UDF Aggregation Task evaluation.

4.5.3 *Comparing HSim with MRPerf*

It should be pointed out that HSim has been designed because few exiting MapReduce simulators are available and *MRPerf* [142] is a representative one. *MRPerf* is evaluated and compared its performance with that of HSim using real Hadoop configurations. Figure 4.13 shows the comparison from which it can be observed that HSim significantly outperforms MRPerf when compared with real Hadoop cluster behavior.

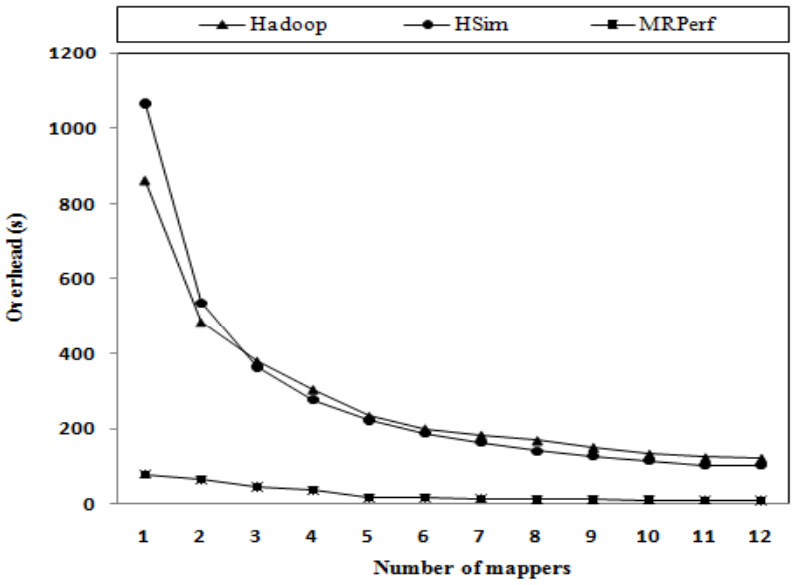

Figure 4.13: A comparison of HSim with MRPerf.

One reason for such performance mismatch is that *MRPerf* does not simulate exactly the behaviors of Hadoop. For example, in a *map* operation, the spilled data will be kept writing onto buffer space while the *map* task is running. When the occupied size of the buffer is less than a certain threshold, in-memory data will be kept spilling onto hard disk simultaneously. Due to the highly changing capacities of system resources, this mechanism can have an impact on the number of spilled files and further I/O behavior will be significantly affected. However, MRPerf simply ignores these events and writes a pre-defined value onto the hard disk.

4.5.4 Simulation Results

Using HSim, a number of Hadoop environments are simulated and evaluated the performance of RASMO from the aspects of scalability, the effectiveness in load balancing and the overhead of the load balancing scheme.

Scalability

To further evaluate the scalability of the RASMO algorithm, HSim has been employed and simulated a number of Hadoop environments using a varying number of nodes up to 250. Each Hadoop node was simulated with 4 *mappers*, and 4 input data sets were used in the simulation tests. Table 4.3 shows the configurations of the simulated Hadoop environments.

Table 4.3 Configurations for scalability evaluation.

Simulation environment	
Number of simulated nodes:	250
Data size:	100,000MB
CPU processing speed:	0.75MB/s
Hard drive reading speed:	80MB/s
Hard drive writing speed:	40MB/s
Memory reading speed:	6000MB/s
Memory writing speed:	5000MB/s
Network bandwidth:	1Gbps
Total number of Map instances:	4 *mappers* per node

From Figure 4.14 it can be observed that the processing time of RASMO decreases as the number of nodes increases. It is also worth noting that there is no significant reduction in

processing time of RASMO beyond certain number of nodes. This is primarily due to the fact that Hadoop incurs a higher communication overhead when dealing with a larger number of computing nodes.

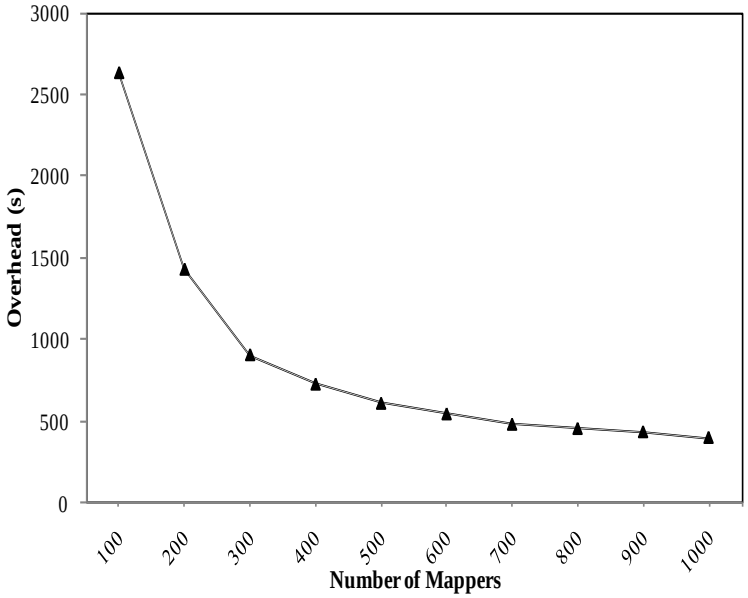

Figure 4.14: The scalability of RASMO in simulation environments.

4.5.5 Load Balancing

Table 4.4 shows the configurations of the simulated Hadoop environments in evaluating the effectiveness of the load balancing scheme of RASMO.

Table 4.4 Configurations for load balance evaluation.

Simulation environment	
Number of simulated nodes	20
Number of processors in each node	1
Number of cores in each processor	2
The processing speeds of processors	depending on heterogeneities
Heterogeneities	from 0 to 2.28
Number of hard disk in each node	1
Reading speed of Hard disk	80MB/s
Writing speed of Hard disk	40MB/s
Number of Mapper	each node employs 2 *map* instances
Sort factor:	100

To evaluate the load balancing algorithm a cluster with 20 computing nodes is simulated. Each node has a processor with two cores. The number of *mappers* is equals to the number of cores. Therefore two *mappers* on a single processor with two cores have been run.

The speeds of the processors are generated based on the heterogeneities of the Hadoop cluster. In the simulation environments the total processing power of the cluster was $P = \sum_{i=1}^{n} p_i$ where n represents the number of the processors employed in the cluster and p_i represents the processing speed of i^{th} processor. For a Hadoop cluster with a total computing capacity denoted with P, the levels of heterogeneity of the Hadoop cluster can be defined using equation (4.19).

$$Heterogeneity = \sqrt{\sum_{i=1}^{n} (\bar{p} - p_i)^2} \tag{4.19}$$

In the simulation, the value of heterogeneity varied from 0 to 2.28. The reading and writing speeds of hard disk were measured from the experimental results. In the RASMO algorithm, *mappers* are the actual processing units. Therefore balancing the workloads of the *mappers* in the first layer in the cascade SVM model is the core part of the load balancing algorithm. 10GB data in the tests has been employed.

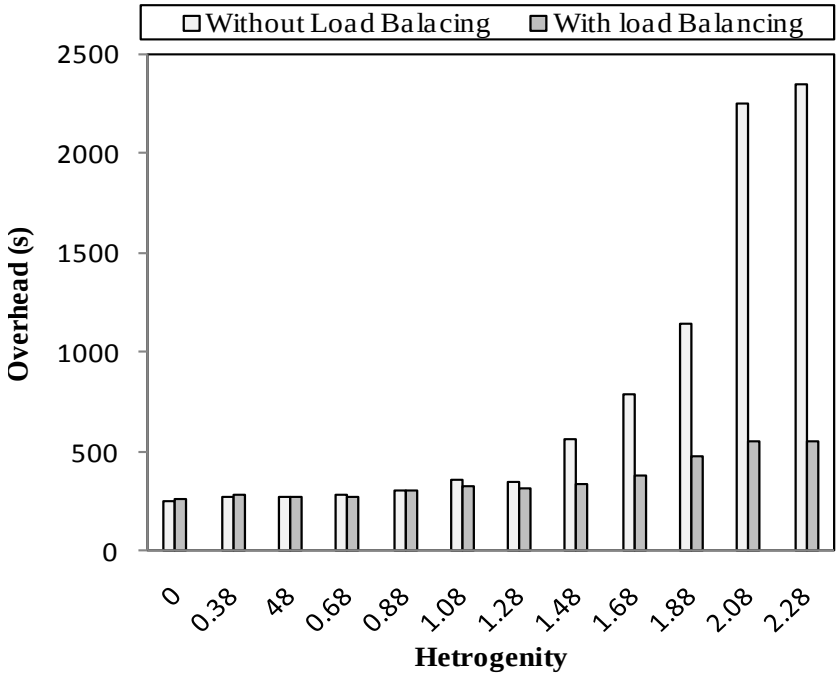

Figure 4.15: The performance of RASMO with load balancing.

Figure 4.15 shows the performance of RASMO with load balancing. It can be observed that when the level of heterogeneity is less than 1.08 indicating homogeneous environments, the load balancing scheme does not make any difference to the RASMO algorithm in performance. However the load balancing scheme reduces the overhead of RASMO significantly with an increasing levels of heterogeneity showing that the resource aware RASMO can optimize resource utilization in highly heterogeneous computing environments.

The degree of heterogeneity is kept the same in the simulated cluster but varied the size of data from 1GB to 10GB. This set of tests was used to evaluate how the load balancing scheme performs with different sizes of data sets. Figure 4.16 shows that the load balancing scheme always reduces the overhead of RASMO in SVM training using varied volumes of data.

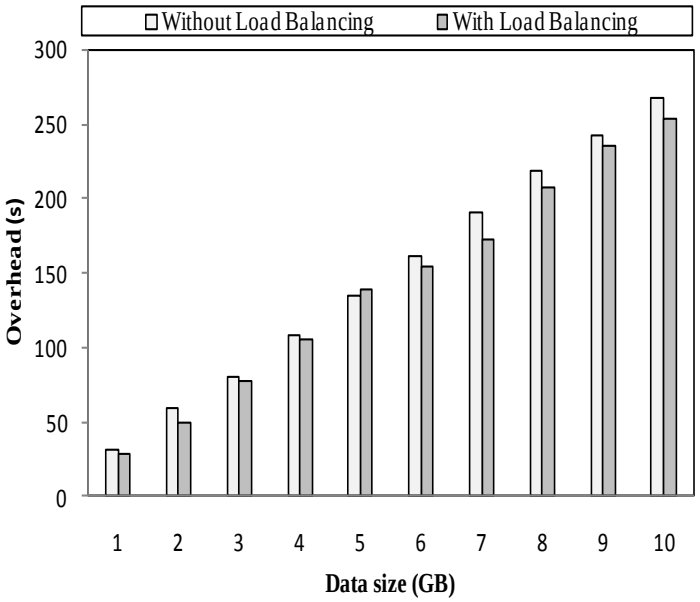

Figure 4.16: The performance of RASMO with varied sizes of data.

4.5.5.1 Overhead of the Load Balancing Scheme

The load balancing scheme builds on a genetic algorithm whose convergence speed affects the efficiency of RASMO in training. To analyze the convergence speed of the genetic algorithm, the numbers of generations are varied and the overhead of RASMO in processing a 10GB dataset in a simulated Hadoop environment are measured. Figure 4.17 shows that

RASMO has a quick convergence process in reaching a stable performance. After approximately 300 generations an optimal or near optimal solution is found.

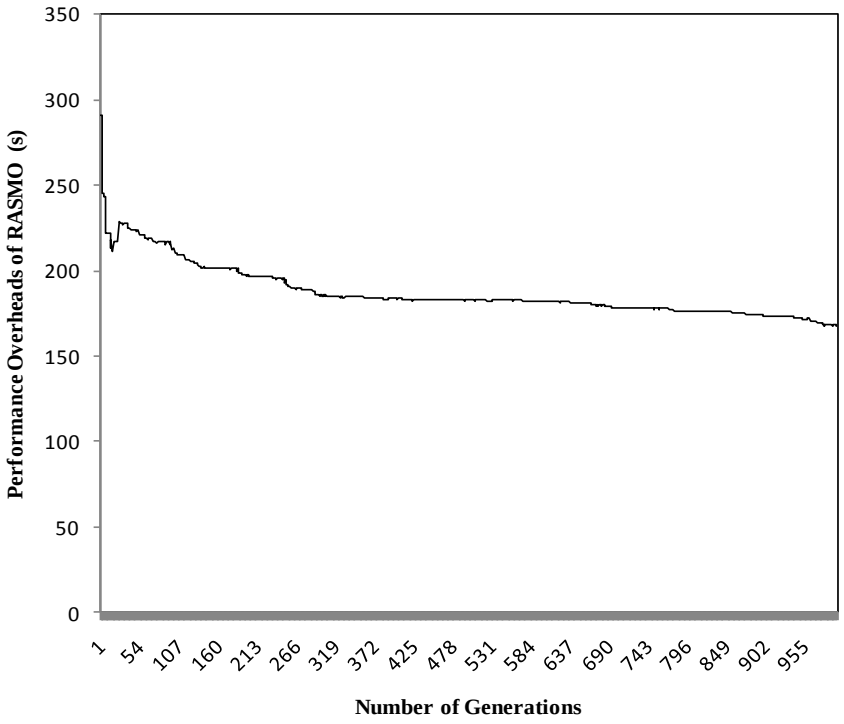

Figure 4.17: The convergence of the RASMO.

The load balancing algorithm incur overhead during execution. Figure 4.18 shows the overheads of the algorithm with the increasing of number of Map instances and job data size. However the overhead of the load balancing algorithm is insignificant in comparison to total overhead of RASMO.

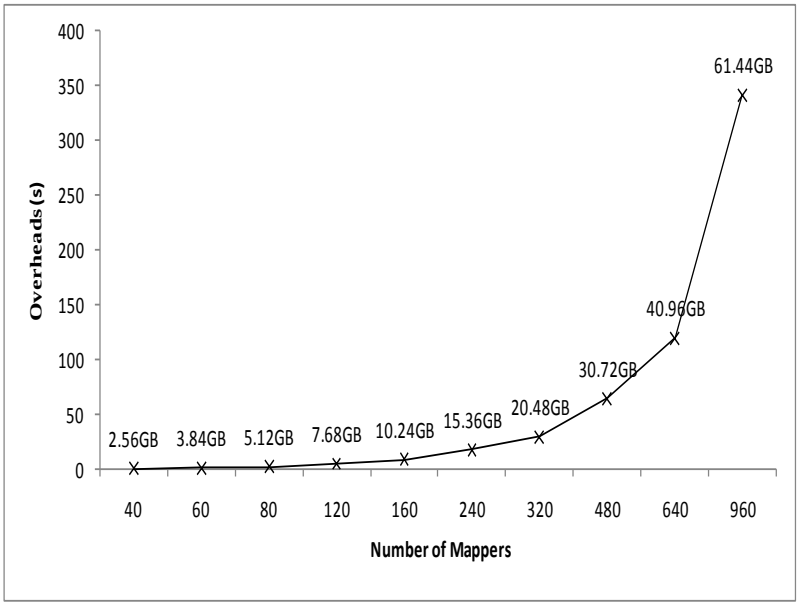

Figure 4.18: Overheads of the load balancing algorithm.

4.6 Summary

This chapter presented RASMO, a resource aware parallel SVM algorithm for large scale image annotation which partitions the training data set into smaller subsets and optimizes SVM training in parallel using a cluster of computers. RASMO was evaluated in both experimental and simulation environments showing that the distributed SVM algorithm reduces the training time significantly while maintaining a high level of accuracy in classifications.

Chapter 5

Parallelizing Multiclass SVM for Scalable Image Annotation

This chapter presents RAMSMO, a resource aware parallel multiclass SVM algorithm for large scale image annotation which partitions the training dataset into smaller binary chunks and optimizes SVM training in parallel using a cluster of computers. A genetic algorithm based load balancing scheme is designed to optimize the performance of RAMSMO in balancing the computation of multiclass data chunks in heterogeneous computing environments.

5.1 The Design of RAMSMO

This section starts with a brief description of the One Against One technique followed by a detailed description of RAMSMO.

5.*1.1 OAO Method*

Multiclass classification based on *OAO* method is the formation of a binary classifier for every pair of distinct classes. The decision function of the SVM classifier for classes such as class $(1,\ 2)$ and class $(2,\ 1)$ has reflectional balance; hence only one of these pairs of classifiers is required. Therefore a total of $k(k-1)/2$ binary classifiers are created where k is the number of classes. The training data for each classifier is a subset of the available training data which only contains the data for the two classes involved. A binary classifier C_{ij} is trained with the training samples from class i as positive and the training samples from class j as negative. The output of each binary classifier can be interpreted as the posterior probability of the positive class [44]. Hastie and Tibshirani [60] proposed a pairwise coupling strategy for combining the probabilistic outputs of all the OAO binary classifiers to estimate the posterior probabilities $p_i = \Pr ob(\omega i \mid x)$, $i = 1,.....k.$ Once posterior probabilities are estimated, based on pairwise coupling technique unlabeled instance is assigned to the class with the largest p_i. Based on a comparative study carried out in [44] the pairwise coupling

scheme is highly recommended as the best kernel discriminate method for solving multiclass problems.

5.1.2 Pairwise Coupling

Pairwise coupling is the learning of k $k(k-1)/2$ pairwise decision rules and couples the pair wise class probability estimates into a joint probability estimate for the entire classes [116]. In comparison to other commonly used multiclass classification techniques, pairwise coupling is more suitable in reducing the computational cost which is closely related to the size of the training data [116]. Pairwise coupling process is as follows. Let r_{ij} denote the probabilistic output of C_{ij} then $r_{ij} = \Pr ob(\omega i \mid \omega i \text{ or } \omega j)$. here the objective is to couple the sets r_{ij} into a general set of probabilities $p_i = \Pr ob(\omega i)$, this problem has no general solution due to the existence of $k-1$independent parameters and $k(k-1)/2$ equations. However Hastie and Tibshirani [60] proposed a new set of auxiliary variables μ_{ij} which are related to p_i.

$$\mu_{ij} = \frac{p_i}{p_i + p_j} \tag{5.1}$$

p_i need to be found such that the corresponding μ_{ij} are in some sense close to r_{ij}. The Kullback-Leibler [82] distance between μ_{ij} and r_{ij} is chosen as the suitable measurement of closeness.

$$l(p) = \sum_{i<j} n_{ij} \left(r_{ij} \log \frac{r_{ij}}{\mu_{ij}} + (1 - r_{ij}) \log \frac{1 - r_{ij}}{1 - \mu_{ij}} \right) \tag{5.2}$$

The associated gradient equations are as follow:

$$\sum_{i \neq j} n_{ij} r_{ij} = \sum_{i \neq j} n_{ij} \mu_{ij}, i = 1,2...k \tag{5.3}$$

$$\text{subject to } \sum_{i=1}^{k} p_i = 1$$

The p_i values are computed using the following iterative procedure:

- set p_i with some initial guess values and the correspoding μ_{ij} values.
- p_i is computed which minimizes $l(p)$ by iterating

$$p_i \leftarrow p_i = \frac{\sum_{i \neq j} n_{ij} \mu_{ij}}{\sum_{i \neq j} n_{ij} r_{ij}} \tag{5.4}$$

- Renormalize the p_i's , $p_i \leftarrow p_i = \frac{p_i}{\sum_i p_i}$
- Re-compute the μ_{ij} and check for convergence.

5.2 RAMSMO

RAMSMO builds on MapReduce for parallelization of SVM computation in training. This section starts with a detailed description of the RAMSMO algorithm.

5.2.1 *Algorithm Design*

The RAMSMO algorithm partitions the entire training dataset into binary subsets (data chunks) and assigns each subset to a single *mapper* in MapReduce. The number of *mappers* is equal to the number of binary chunks. Each *mapper* optimizes a data chunk in parallel. Figure 5.1 presents a high level pictorial representation of this approach.

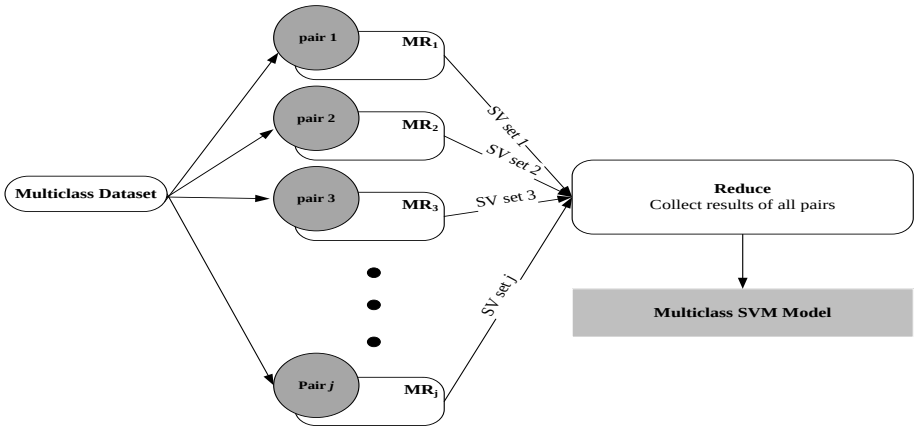

Figure 5.1: The architecture of RAMSMO.

The output of each *mapper* is the *alpha* array (Lagrange multipliers) for a binary subset, the training data X_i which corresponds to *alpha* $a_i > 0$ and a bias *b* in order to compute SVM output *u* using equation (5.5).

$$u = \sum_{i=1}^{n} y_i a_i K(X_i, X) + b. \tag{5.5}$$

where *X* is an instance to be classified and *K* is the kernel function.
In the case of a linear SVM the output of each *mapper* includes a *weight vector* and the value *b* in order to calculate the SVM output *u* using equation (5.6).

$$u = \vec{w}.\vec{x} - b. \tag{5.6}$$

The reduce task simply collects and stores generated binary classifiers which are used as the trained multiclass SVM model for classification. Algorithm 5.1 shows the pseudo code of RAMSMO. Line 1-2 show the construction of all the binary data chunks. Lines 3-6 show the optimization process of SMO for each binary chunk. Line 7 shows the assembling results from all the *mappers*.

Algorithm 5.1: RAMSMO Algorithm

Input: training data x_i
Output: support vectors sv_k, weight vectors w_i if SVM is linear
1: split training data x_i into single class chunks
2: combine single chunks to create all possible binary pairs x_b;
MAP$_j$ $\forall j \in \{1..n\}$, $n = k(k-1)/2$
Input: binary chunks x_b
Output: support vectors sv_k and data x_k
3: train SMO on each binary pair
4: obtain sv_k set for k pair; $sv_k = \{\alpha_k > 0\}$
5: store all weight vectors sv_k
6: weight vectors w_i if SVM is linear
REDUCE
7: collect and store all results.

5.3 Load Balancing

A genetic algorithm based load balancing scheme is designed to optimize the performance of RAMSMO in heterogeneous computing environments. The load balancing scheme computes

optimal number of binary chunks processed by available *Mappers*, a single *Mapper* may process a number of binary chunks based on the resources available in a cluster of computers such as the computing powers of processors, the storage capacities of hard drives and the network speeds of the participating nodes.

A genetic algorithm is similar to the algorithm describe in section 4.3 of chapter 4. However the major difference is that there is no crossover due to the uniqueness of the binary subsets which is regarded as a evolutionary algorithm. Assume there are a fixed number of binary subsets (genes) in a chromosome. A random number of genes are allocated to the available *Mappers*. The positions of two randomly selected genes belonged to the corresponding *Mappers* are changed to perform mutation, the mutation rate is 0.01. The fitness of newly generated chromosome is evaluated based on equation (5.7) which is used by the genetic algorithm in finding an optimal or a near optimal solution in determining the number binary data chunks processed by available *Mappers.*

$$f(T) = \sqrt{\sum_{i=1}^{k} (\bar{T} - T_i)^2} \tag{5.7}$$

5.4 Experimental Results

RAMSMO has been incorporated into our image annotation system which is developed using the Java programming language and the Weka package. The image annotation system classifies visual features into pre-defined classes. Figure 5.2 shows a snapshot of the system.

Figure 5.2: A snapshot of the image annotation system [32].

5.4.1 Image Corpora

The images are collected from the Corel database. Images are classified into 10 classes, and each class of the images has one label associated with it. The 10 pre-defined labels are *people*, *beach*, *mountain*, *bus*, *food*, *dinosaur*, *elephant*, *horse*, *flower* and *historic item*. Typical images with 384x256 pixels are used in the training process. Low level features of the images are extracted using the LIRE (Lucene Image REtrieval) library. After extracting low level features a typical image is represented in the following form:

0,256,12,1,-56,3,10,1,18,.....2,0,0,1,0,0,0,0,0,0,0,0,0,0,beach

Each image is represented by 483 attributes which include 58 attribute that represent edge histogram and 424 attributes represent Scalable Colour Descriptor and the last attribute indicates the class name which indicates the category to which the image belongs to.

5.4.2 *Performance Evaluation*

MRSMO is implemented using WEKA base machine learning libraries written in the Java programming language and tested in a Hadoop cluster. To evaluate RAMSMO, the SMO algorithm provided in the Weka package is extended, configured and packaged it as a basic MapReduce job. The Hadoop cluster for this set of experiments consist of a total of 12 physical processor cores across 3 computer nodes as shown in Table 5.1.

Table 5.1 Hadoop Configurations for RAMSMO.

Hardware environment			
	CPU	Number of Cores	RAM
Node 1	Intel Quad Core	4	4GB
Node 2	Intel Quad Core	4	4GB
Node 3	Intel Quad Core	4	4GB
Software environment			
SVM	WEKA 3.6.0 (SMO)		
OS	Fedora10		
Hadoop	Hadoop 0.20		
Java	JDK 1.6		

RAMSMO is evaluated the performance of from the aspects of efficiency and accuracy. Polynomial kernel function has been used in the experiments. Figure 5.3 shows the efficiency of the RAMSMO in SVM training which achieved close to 12 times in speedup.

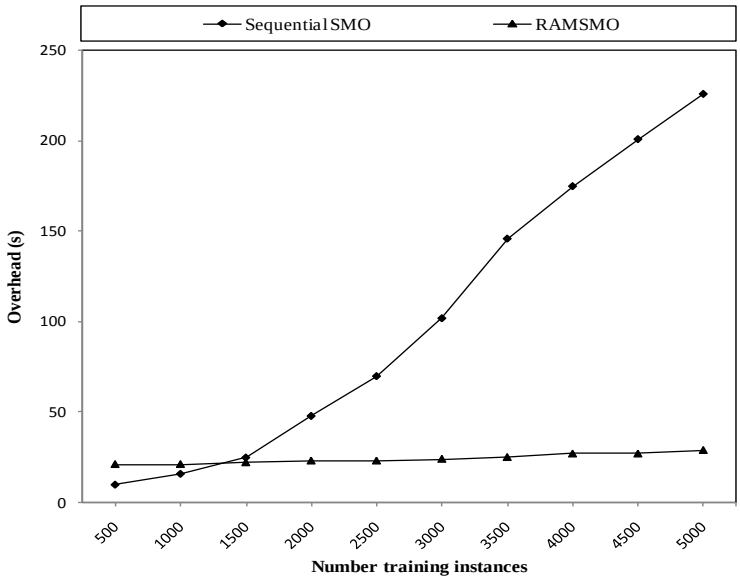

Figure 5.3: The efficiency of RAMSMO in SVM training using 12 *mappers*.

Figure 5.4 shows the efficiency of the RAMSMO in comparison with MRSMO [76] which is one against all based distributed multiclass SVM. RAMSMO is more efficient due to the fact that training data for each binary classifier is a subset of the available training data which only contains the data for the two classes involved. One against all based MRSMO incurs higher training overhead due to the involvement of all training data for creating binary classifiers for each class.

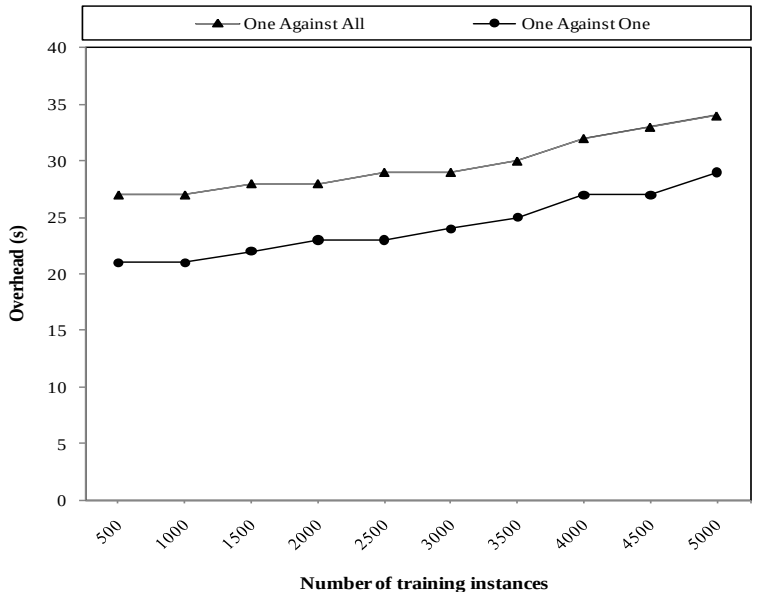

Figure 5.4: A comparison of RAMSMO and MRSMO.

RAMSMO is evaluated with an unequal number of instance for each class, resulting in the fact that the *mapper* that processes the largest data chunk is the last to finish before the *reduce* phase can start. Figure 5.5 shows the increase in the overhead of RAMSMO with unequal binary data size which highlights the need for an effective load balancing scheme for heterogeneous environments.

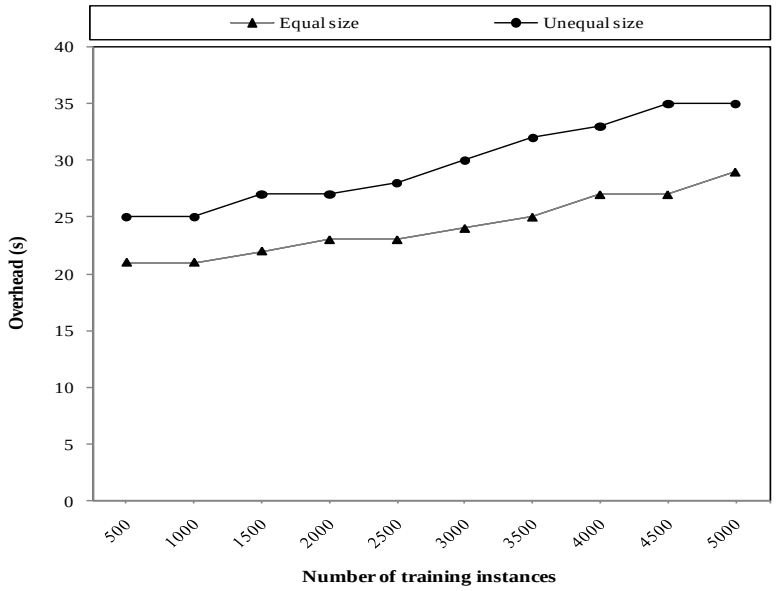

Figure 5.5: The overhead of RAMSMO using equal and unequal binary chunks.

Furthermore the accuracy of the sequential SMO and RAMSMO is evaluated in classification and presented the results in Table 5.2. In total 50 unlabeled images were tested (10 images at a time), the average accuracy level was considered. It is clear that the parallelization of

RAMSMO has no affect on the accuracy level due to the way of the algorithm is parallelized. The results show that RAMSMO achieves 94% which was the same as the sequential SMO.

Table 5.2 Summarising Performance Results.

	Sequential SMO	RAMSMO 3 computers (12 *Mappers*)
Correctly Classified	≈ 94 %	≈ 94 %
Incorrectly Classified	≈ 6%	≈ 6%
Training time for 5000 instances	241 (s)	35 (s)

5.5 Simulation results

To further evaluate the effectiveness of RAMSMO algorithm in MapReduce environments, a number of Hadoop environments are simulated and the performance of RAMSMO is evaluated using HSim from the aspects of scalability, the effectiveness in load balancing and the overhead of the load balancing scheme.

5.5.1 Scalability

To further evaluate the scalability of the RAMSMO algorithm, HSim is employed and a number of Hadoop environments are simulated using a varying number of nodes up to 250. Each Hadoop node was simulated with 4 *mappers*, and 4 input datasets were used in the simulation tests. Table 5.3 shows the configurations of the simulated Hadoop environments.

Table 5.3: Configurations for Scalability Evaluation.

Simulation environment	
Number of simulated nodes:	250
Data size:	100,000MB
CPU processing speed:	0.75MB/s
Hard drive reading speed:	80MB/s
Hard drive writing speed:	40MB/s
Memory reading speed:	6000MB/s
Memory writing speed:	5000MB/s
Network bandwidth:	1Gbps
Total number of Map instances:	4 *Mappers* per node (1000 *Mappers)*

From Figure 5.6 it can be observed that the processing time of RAMSMO decreases as the number of nodes increases. It is also worth noting that there is no significant reduction in

processing time of RAMSMO beyond a certain number of nodes. This is primarily due to the fact that Hadoop incurs a high communication overhead when dealing with a large number of computing nodes. There is no significant difference between *mapper* overhead and total overhead (involving both *mapper* and *reducer*) which the *reducer* dose not incur significant overhead.

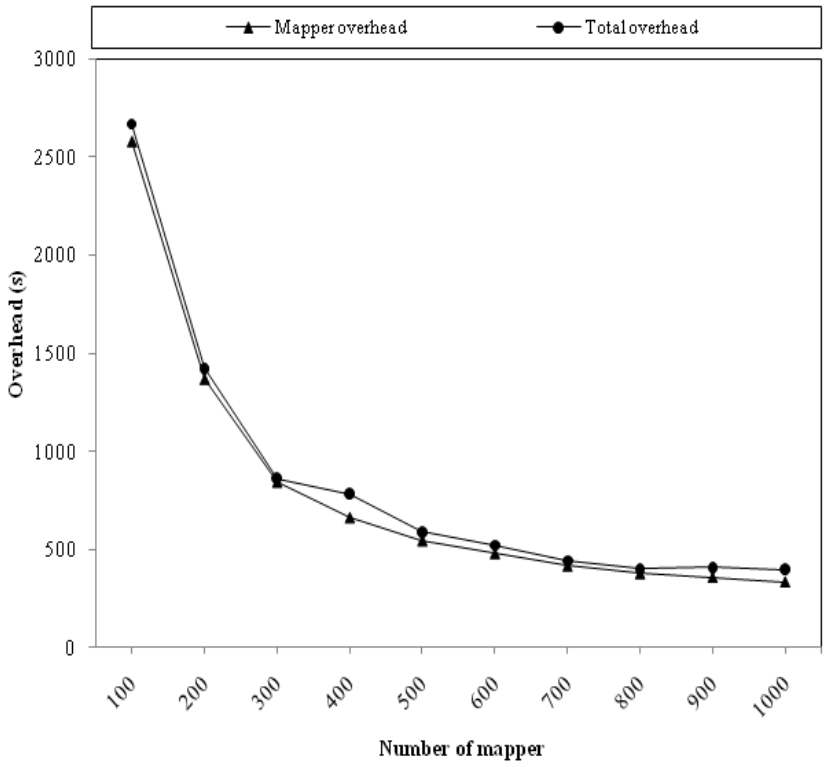

Figure 5.6: The scalability of RAMSMO in simulation environments.

5.5.2 Load Balancing

Table 5.4 shows the configurations of the simulated Hadoop environments in evaluating the effectiveness of the load balancing scheme of RAMSMO.

Table 5.4 Configurations for Load Balancing Evaluation.

Simulation environment	
Number of simulated nodes	20
Number of processors in each node	1
Number of cores in each processor	2
The processing speeds of processors	depending on heterogeneities
Heterogeneities	from 0 to 2.28
Number of hard disk in each node	1
Reading speed of Hard disk	80MB/s
Writing speed of Hard disk	40MB/s
Number of *Mappers* and *Reducers*	each node employs 2 *mappers* instances and 1 *reducers*
Sort factor:	100

To evaluate the load balancing algorithm a cluster with 20 computing nodes is simulated. Each node has a processor with two cores. The optimal number of *mappers* is equals to the number of cores. Therefore two *mappers* on a single processor with two cores are run. The number of *reducer* is set to one on each node.

The speeds of the processors are generated based on the heterogeneities of the Hadoop cluster. In the simulation environments the total processing power of the cluster was $p = \sum_{i=1}^{n} p_i$ where n represents the number of the processors employed in the cluster and p_i represents the processing speed of i^{th} processor. For a Hadoop cluster with a total computing capacity denoted with P, the levels of heterogeneity H of the Hadoop cluster can be defined using equation (4.19).

In the simulation, the value of heterogeneity varied from 0 to 2.28. The reading and writing speeds of hard disk were measured from the experimental results. Figure 5.7 shows the performance of RAMSMO with load balancing. It can be observed that when the level of heterogeneity is less than 1.08 indicating homogeneous environments, the load balancing scheme does not make any difference to the RAMSMO algorithm in performance. However the load balancing scheme reduces the overhead of RAMSMO significantly with an

increasing levels of heterogeneity showing that the resource aware RAMSMO can optimize resource utilization in highly heterogeneous computing environments.

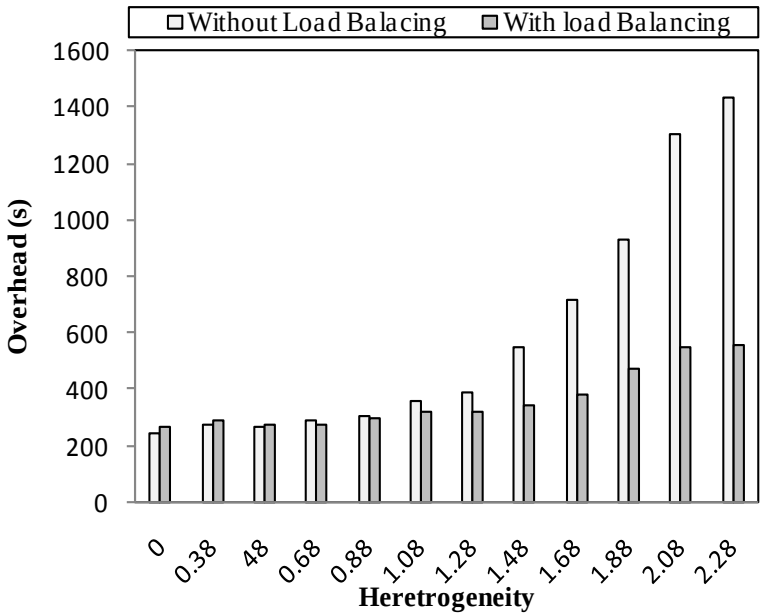

Figure 5.7: The performance of RAMSMO with load balancing.

The degree of heterogeneity is kept the same in the simulated cluster but varied the total size of data from 1GB to 10GB. This set of tests was used to evaluate how the load balancing scheme performs with different sizes of data sets. Figure 5.8 shows that the load balancing scheme always reduces the overhead of RAMSMO in SVM training using varied volumes of data.

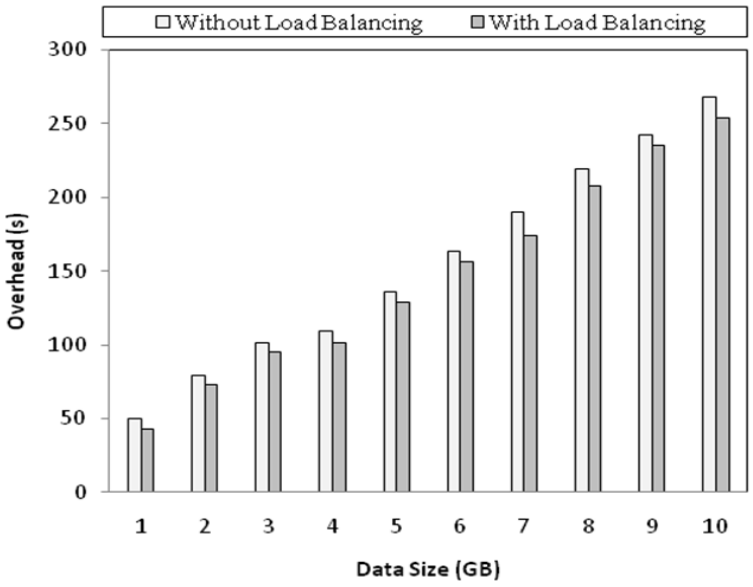

Figure 5.8: The performance of RAMSMO with different datasets.

Figure 5.9 compares the performance of RAMSMO with that of MinMin, MaxMin in load balancing. It can be observed that RAMSMO performs better than both MinMin and MaxMin, and the performance of MinMin is the worst due to the existence of a large number of tasks with short processing times and a small number task with long processing times.

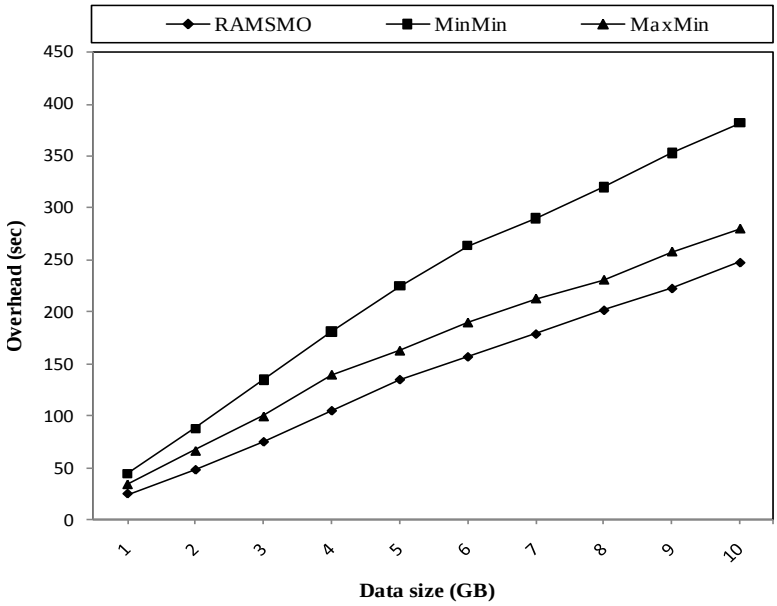

Figure 5.9: A comparison RAMSMO with MinMin and MaxMin.

5.5.3 Overhead of the Load Balancing Scheme

The load balancing scheme builds on a genetic algorithm whose convergence speed affects the efficiency of RAMSMO in training. To analyze the convergence speed of the genetic algorithm, the numbers of generations are varied and the overhead of RAMSMO in processing a 10GB dataset in a simulated Hadoop environment are measured. Figure 5.10 shows that RAMSMO has a quick convergence process in reaching a stable performance.

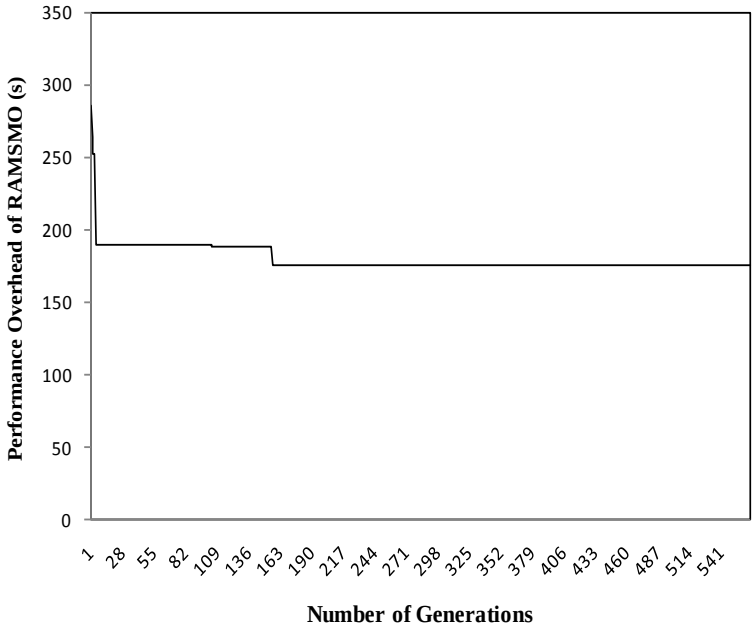

Figure 5.10: The convergence of the RAMSMO.

The load balancing scheme incurs overhead during execution. Figure 5.11 shows increased overhead of the scheme with the increasing number of *mappers* and job data sizes. The overhead is usually insignificant compare to the overall processing time of *map* operations.

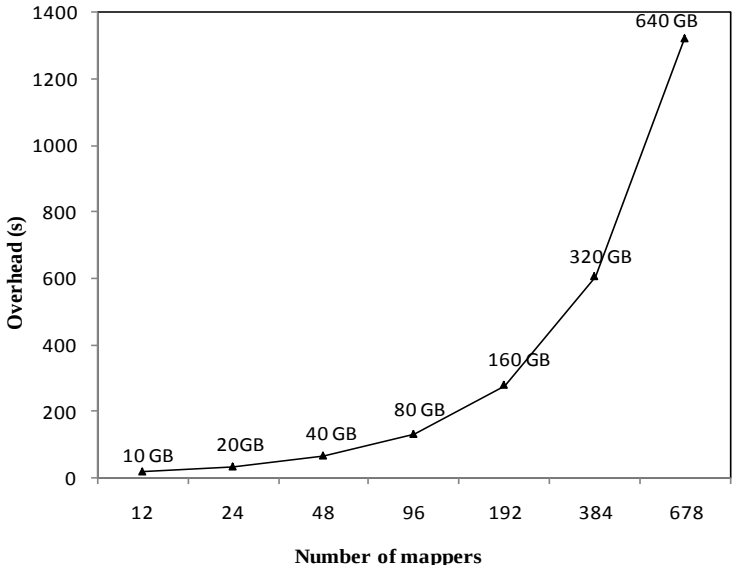

Figure 5.11: Overheads of the load balancing scheme.

5.6 Summary

This chapter presented RAMSMO, a resource aware parallel multiclass SVM algorithm for large scale image annotation which partitions the training data set into smaller subsets and optimizes SVM training in parallel using a cluster of computers. RASMO was evaluated in both experimental and simulation environments showing that the distributed SVM algorithm reduces the training time significantly while maintaining a high level of accuracy in classifications.

Chapter 6

Distributed SVM Ensemble for Scalable Image Annotation

This chapter presents MRESVM, a distributed SVM ensemble algorithm for image annotation which re-samples the training data based on bootstrapping and trains SVM on each sample in parallel using a cluster of computers. Balanced sampling strategy for bootstrapping is introduced to increase classification accuracy of SVM ensemble for fixed number samples.

6.1 SVM Ensemble

An ensemble of classifiers is a set of multiple classifiers based on the idea of combining a number of weak learners to create a strong learner. Training a diverse set of classifiers from a single training data set and to vote or average their predictions is simple and powerful [148]. There are a number of techniques for creating a diverse set of classifiers. The most common technique is to use re-sampling to diversify the training sets based on Bootstrap Aggregating (bagging). Breiman [136] showed bagging techniques reduces the variance component of misclassification error, therefore increase the reliability of the predictions. When the number of classifiers is large, the probability of error becomes small, bagging have been successfully applied to different classification problems [54] [78] [117] [129] [134] [150].

A single SVM may not always provide a good classification performance over all test data. To overcome this limitation, ensembles of SVMs have been proposed as a solution [78]. Figure 6.1 shows a general architecture of SVM ensemble.

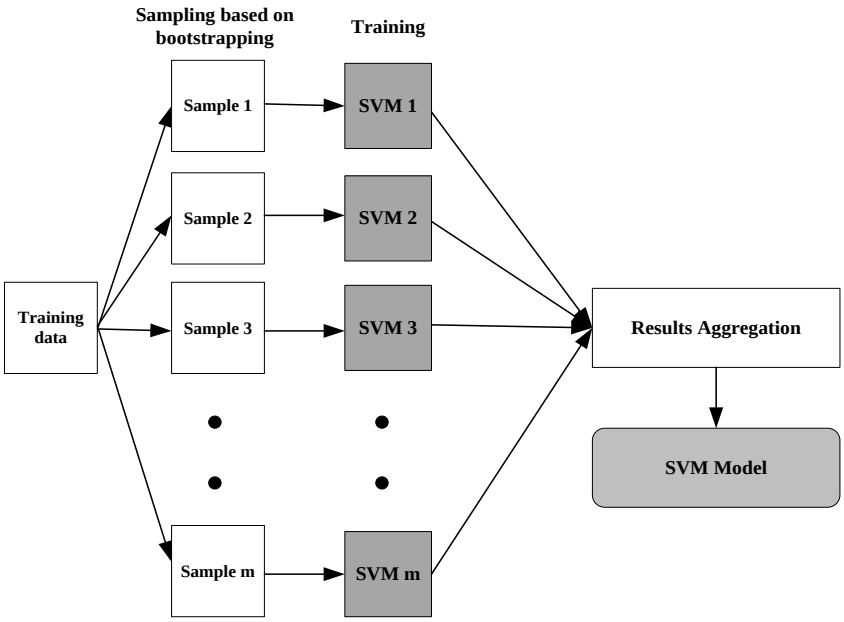

Figure 6.1: Architecture of SVM ensemble.

Each SVM is trained separately with a sample of training data created from the original data set based on bootstrapping technique. Bootstrap constructs m training data samples by random re-sampling with replacement, from the original training data set repeatedly. A particular training instance x may appear repeatedly or not appear in any particular sample. Once the training process is complete, trained SVMs are combined based on a suitable combination approach.

6.1.1 Aggregation Methods

Two types of combination methods are described in [78]. A linear combination approach that combines several SVMs linearly such as combining based on majority voting. A nonlinear combination approach is the nonlinear combination of several SVMs based double layer hierarchical combining that use second layer SVM to combine the first layer SVMs.

Majority voting is one of the commonly used and simplest combination techniques. The ensemble classifier predicts a class for a test instance which is predicted by the majority of the base classifiers [124]. Let us define the prediction of the i^{th} classifier P_i as $p_{i,j} \in \{1,0\}, i = 1,....,i$ and $j = 1,...,c$ where i the number of classifiers is and C is the number

of classes. If i^{th} classifier chooses class j, then $p_{i,j}=1$ otherwise $p_{i,j}=0$. The ensemble predict for class k if:

$$\sum_{i=1}^{I} p_{i,k} = \max_{j=1}^{c} \sum_{i=1}^{I} p_{i,j} \tag{6.1}$$

Double layer hierarchical is a combining method which uses a single SVM to aggregate the outputs of a number of SVMs. Therefore, this method of combination consists of two layers of SVMs hierarchy where the outputs SVMs in the first layer feed as input into a single SVM in the second layer [78]. Let $f_m(m=1,2,3,...M)$ be a decision function of the m^{th} SVM in the SVM ensemble and F be a decision function of SVM in the second layer. Then, the final decision of the SVM ensemble $f_{SVM}(x)$ for a given test vector x based on double-layer hierarchical combining is determined by $f_{SVM}(x)=F(f_1(x), f_2(x),.....f_m(x))$, m is the number of SVMs in the SVM ensemble.

6.1.2 Balanced Bootstrapping

In Monte Carlo algorithms [13], variance reduction is a technique used to increase the precision of the estimates that can be obtained for a fixed number of iterations in simulation. Balanced bootstrapping is a variance reduction technique for efficient bootstrap simulation proposed by Davison et al. [36]. Esposito and Saitta [47] have established the link between Bagging and Monte Carlo algorithms. Despite some differences, these two algorithms compute the very same function.

Balanced bootstrapping is based on the idea of controlling the number of times training instances appear in the bootstrap samples, so that in the *B* bootstrap samples, each instance appears the same number of times. For the bootstrap to work, some instances must be missing in certain bootstrap samples, while others may appear two or more times [36]. Balanced sampling dose not force each bootstrap sample to contain all training instances; the first

instance may appear twice in the first bootstrap sample and not appear at all in the second bootstrap sample, while the second instance may appear once in each sample.

A number of techniques introduced for creating balanced bootstrap samples. However a simple way of creating balanced bootstrap samples described in [36] is to construct a string of the instances $X_1, X_2, X_3, ..., X_n$ repeated B time, here we have the sequence $Y_1, Y_2, Y_3, ..., Y_{Bn}$. We take a random permutation p of the integers from 1 to B_n. We create the first bootstrap sample from $Y_p(1), Y_p(2), Y_p(3), ..., Y_p(n)$, the second bootstrap sample from $Y_p(n+1), Y_p(n+2), Y_p(n+3), ..., Y_p(2n)$ and so on, until $Y_p((B-1)n+1), Y_p((B-1)n+2), Y_p((B-1)n+3), ..., Y_p(Bn)$ is the B^{th} bootstrap sample. The balanced bootstrapping variance reduction technique can be in bagging to increase the classification accuracy.

6.2 Bias Variance Decomposition

Given a training set $\{x_1, x_2, ...x_n\}$ a trained classifier f is created, given a test instance x, the classifier predicts $y = f(x)$. Let a be the actual value of the predicted variable for the test instance x. A loss function $L(a, y)$ measures the cost of predicting y when the true value is a [42]. One of the commonly used loss functions is zero-one loss $L(a, y) = 0$ if $y = a$, otherwise $L(a, y) = 1$. The aim is to create a classifier with the smallest possible loss. For classification problems, several authors proposed bias–variance decompositions related to zero-one loss [16] [53] [62] [80].

Bias–variance decomposition of the classification error is useful tool for analyzing supervised learning algorithms and ensemble techniques to examine the relationships of learning algorithms and ensemble methods with respect to their bias– variance characteristics [135].Bias measures how closely a classifier's average predictions over all possible training sets of the given training set size matches the true value of class. Variance measure how much the classifiers prediction changes for the different training sets of the given size [135].Variance is large if different training sets D give rise to very different classifiers, bias is large in cases where a learning method produces classifiers that are consistently wrong .The bias and variance decomposition is crucial in understanding the bias/variance tradeoffs,

for example where the bias shrinks but the variance increases or bias increases but the variance decreases, the aim is to find the optimal point of the trade-off.

6.3 The MRESVM Algorithm

MESVM builds on MapReduce for parallelization of SVM ensemble computation in training. The MRESVM algorithm is based on the bagging architecture which trains multiple SVMs on bootstrap samples. Both random sampling with replacement and balanced sampling have been used.

As a first step data samples to train the base classifiers have to be generated. For random sampling with replacement, m samples of size s are generated according to the uniform probability distribution from a data set D_m, such m data samples are used to train base classifiers which create the SVM ensembles. In balanced sampling which is an alternative sampling method which forces each training instances to occur t times in the B bootstrap samples. Balanced bootstrap samples are generated by constructing a data set of m copies of the original data, after performing random permutation, the data set is partition into m samples.

In majority voting combinations, each *map* function optimizes a sample in parallel. The number of *map* tasks is equal to the number of sample. The reduce task simply collects and stores generated classifiers which are used in majority voting.

In double hierarchical combinations, each *map* function optimizes a sample in parallel in first layer. The number of *map* tasks is equal to the number of sample. The output of each *map* function is the *alpha* array (Lagrange multipliers) for a sample and the training data X_i which corresponds Lagrange multipliers $\alpha_i > 0$ in order to create input for the second layer, the output of the second layer includes the *alpha* array, bias threshold b and the training data X_i which correspond $\alpha_i > 0$ in order to calculate the SVM output u using equation (6.2).

$$u = \sum_{i=1}^{n} y_i a_i K(X_i, X) + b \tag{6.2}$$

where X is an instance to be classified, y_i is class labels for X_i and K is the kernel function. Each *map* task processes the associated sample and generates a set of support sectors. Each set of support sectors is then combined and forwarded to the *map* task in the second layer as input. In this layer single set of support sectors is computed and the generated SVM model will be used in the classification. Figures 6.2 and 6.3 present a high level pictorial representation of double hierarchical combination and majority voting.

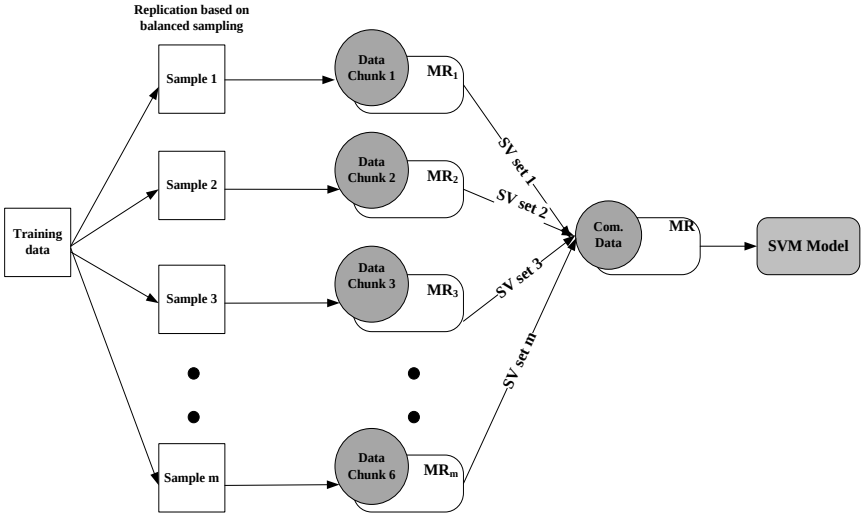

Figure 6.2: MRESVM architecture with double layer hierarchical is a combination.

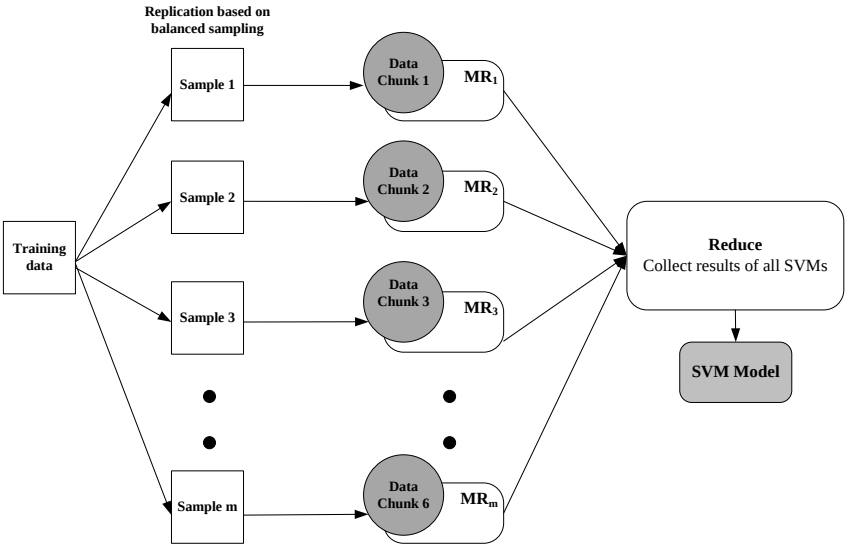

Figure 6.3: MRESVM architecture with majority voting combination.

Algorithm 6.1 shows the pseudo code of MRESVM with double layer hierarchical combination. Lines 1-3 show the bootstrapping process to create balance sample for training SVM. Lines 4-8 show the training process of SVM. Lines 9-12 show the assembling results of layer 1 which are used as input for layer 2, and the training process in layer 2.

Algorithm 6.1: MRESVM Algorithm

Input: training data x_i
Output: support vectors sv_m, weight vectors w_i if SVM is linear
1: replicate training data x_i based on balanced sampling;
2: perform random permutation;
3: create data chunks m to train SVM;
MAP$_j$ $\forall j \in \{1..m\}$,
Input: data chunks m
Output: support vectors sv_m and data x_m
4: train SVM on data chunks m
5: obtain sv_m set for m chunks; $sv_m = \{\alpha_m > 0\}$
6: store all support vectors sv_m
7: store weight vectors w_i if SVM is linear
8: combine sv_m sets;
9: store all x_m for sv_m to create input chunk for next layer *Map* task;
10: train SVM on x_m
11: obtain sv_i set for x_k; $sv_i = \{\alpha_i > 0\}$

6.4 Experimental results

MRESVM has been incorporated into our image annotation system which is developed using the Java programming language and the WEKA package. The image annotation system classifies visual features into pre-defined classes. Figure 6.4 shows a snapshot of the system.

Figure 6.4: A snapshot of the image annotation system [32].

6.4.1 Image Corpora

The images are collected from the Corel database. Images are classified into 2 classes, and each class of the images has one label associated with it. The 2 pre-defined labels are *people* and *beach*. Typical images with 384x256 pixels are used in the training process. Low level features of the images are extracted using the LIRE library. After extracting low level features a typical image is represented in the following form:

0,256,12,1,-56,3,10,1,18,.....2,0,0,0,0,0,0,0,0,beach

Each image is represented by 483 attributes which include 58 attribute that represent edge histogram and 424 attributes represent Scalable Colour Descriptor and the last attribute indicates the class name which indicates the category to which the image belongs to.

6.4.2 Performance Evaluation

MRESVM is implemented using WEKA's base machine learning libraries written in the Java programming language and tested in a Hadoop cluster. To evaluate MRESVM, the SMO algorithm provided in the Weka package is extended, configured and packaged it as a basic MapReduce job. The Hadoop cluster for this set of experiments consist of a total of 12 physical cores across 3 computer nodes as shown in Table 6.1.

Table 6.1: HADOOP Configuration.

Hardware environment			
	CPU	Number of Cores	RAM
Node 1	Intel Quad Core	4	4GB
Node 2	Intel Quad Core	4	4GB
Node 3	Intel Quad Core	4	4GB
Software environment			
SVM	WEKA 3.6.0 (SMO)		
OS	Fedora10		
Hadoop	Hadoop 0.20		
Java	JDK 1.6		

The performance of MRESVM is evaluated from the aspects of efficiency and accuracy. Polynomial kernel function has been used in the experiments. Figure 6.5 shows the efficiency of the MRESVM in SVM training which achieves close to 12 times in speedup.

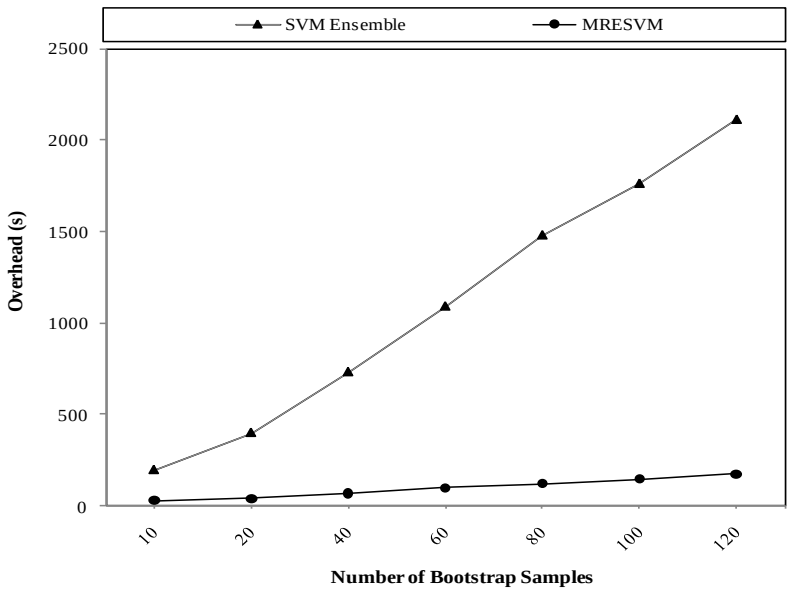

Figure 6.5: The efficiency of MRESVM using 12 *mappers*.

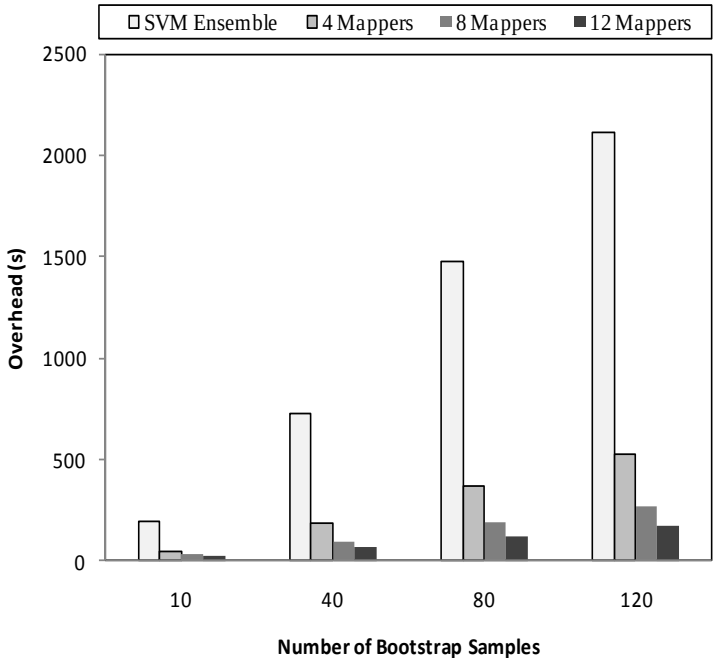

Figure 6.6: The overhead of MRESVM.

MRESVM outperform the single SVM with an increasing number of samples in terms of training time required. Figure 6.6 shows the increasing efficiency with the number of participating MapReduce *mappers* varying from 4 to 12.

6.4.3 Measuring Bias and variance

Figure 6.7 shows a simple Bias–variance decomposition process. The training samples are used to train SVMs. The leaned SVM models are applied on a test set. The bias and variance are then estimated for each instance in the test set and for 0-1 loss the bias and variance are calculated from the number of incorrect classifiers for the instance [11].

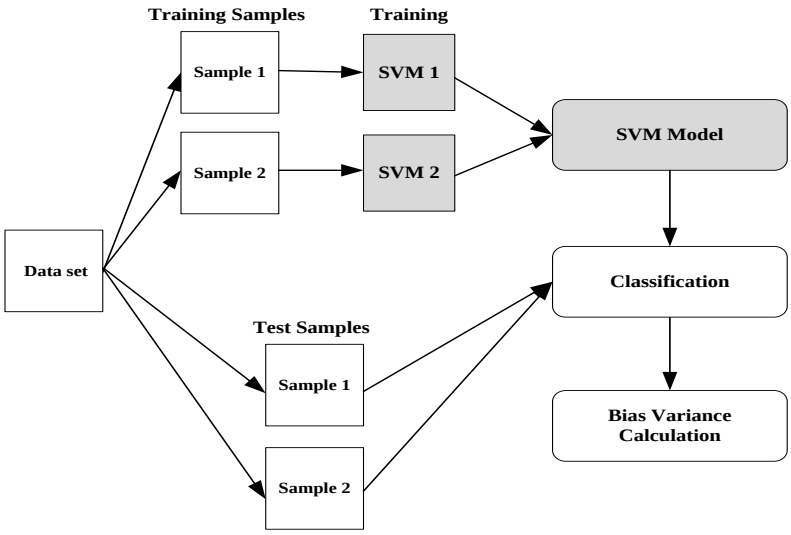

Figure 6.7: Bias–variance decomposition.

The bias for an instance is calculated as;

$$\sum_i (x_i - p_i)^2 - p_i(1-p_i)/c - 1 \tag{6.3}$$

where i sums over the class values, c is the number of classifiers, x_i is a variable that is indicates whether the instance class value equals the i^{th} value, and p_i the part of classifiers that correctly predicted x_i. The variance for an instance is calculated as; $1 - \sum_i p_i^2$.

To analyse the behaviour of MRESVM, the bias variance decomposition method described in [80] is used because it can be applied to any classifier and decomposition does not require the training sets to be sampled in any specific manner. The idea of tuning SVMs to minimize the bias presented in [134] is adopted before apply bagging to reduce variance; resulting MRESVM has lower classification error than a single SVM. As stated in [134] the bias of SVMs is controlled by two parameters. First, the parameter *C* which controls the tradeoffs between fitting the data and maximizing the margin, setting *C* with a large value tend to minimize bias. Second in polynomial kernel, the parameter is the degree *d* of the polynomial. In MRESVM, base SVM algorithm was tune with the value of $d = 4$ and $C = 100$.

Bagging based on random sampling with replacement and balanced sampling have been applied to the base SVMs. Bias, variance and error rate of MRESVM were measured. Figure 6.8 show comparison of classification error of MRESVM with random and balanced

sampling indicating lower classification error rate for MRESVM with balanced sampling. Balanced sampling based bagging has lower variance than random re-sampling with replacement, although the bias is almost the same in both cases.

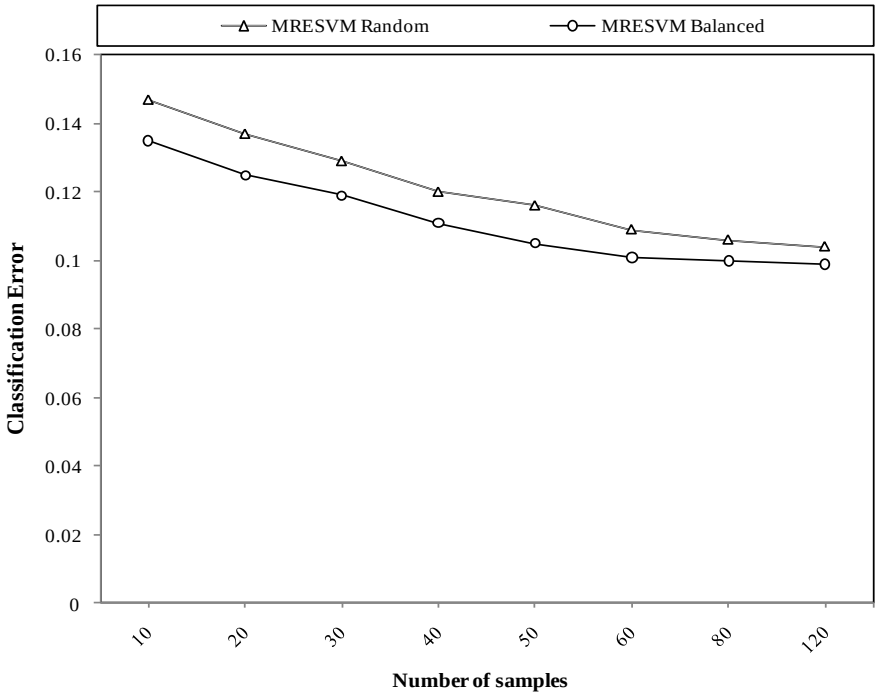

Figure 6.8: Classification error of MRESVM with random and balanced sampling.

Furthermore the accuracy of MRESVM with different sampling strategies and combination methods are evaluated in classification and presented the results in Table 2 using 2000 instances. In total 250 unlabeled images were tested (10 images at a time), the average accuracy level was considered. The results show that MRESVM achieves up to 98% which is higher than single SVM.

Table 6.2 Summarized Performances Results.

	SVM	MRESVM random Majority Vote	MRESVM random Two Layer	MRESVM Balanced Majority Vote	MRESVM Balanced Two Layer
Correctly Classified	**≈ 94 %**	**≈ 95%**	**≈ 96 %**	**≈ 96 %**	**≈ 98 %**
Incorrectly Classified	**≈ 6%**	**≈ 5 %**	**≈ 4 %**	**≈ 3 %**	**≈ 2 %**

6.5 Simulation results

To further evaluate the effectiveness of MRESVM algorithm in MapReduce environments, a number of Hadoop environments are simulated and the performance of MRESVM is evaluated using HSim from the aspects of scalability.

6.5.1 Scalability

To further evaluate the scalability of the MRESVM algorithm, HSim is employed and a number of Hadoop environments are simulated using a varying number of nodes up to 250. Each Hadoop node was simulated with 4 *mappers*, and 4 input datasets were used in the simulation tests. Table 6.3 shows the configurations of the simulated Hadoop environments.

Table 6.3: Configuration for Scalability Evaluation.

Simulation environment	
Number of simulated nodes:	250
Data size:	100,000MB
CPU processing speed:	0.75MB/s
Hard drive reading speed:	80MB/s
Hard drive writing speed:	40MB/s
Memory reading speed:	6000MB/s
Memory writing speed:	5000MB/s
Network bandwidth:	1Gbps
Total number of Map instances:	4 *Mappers* per node (1000 *Mappers)*

From Figure 6.9 it can be observed that the processing time of MRESVM decreases as the number of nodes increases. It is also worth noting that there is no significant reduction in processing time of MRESVM beyond a certain number of nodes. This is primarily due to the fact that Hadoop incurs a high communication overhead when dealing with a large number of computing nodes. There is no significant difference between *mapper* overhead and total overhead (involving both *mapper* and *reducer*) which the *reducer* dose not incur significant overhead.

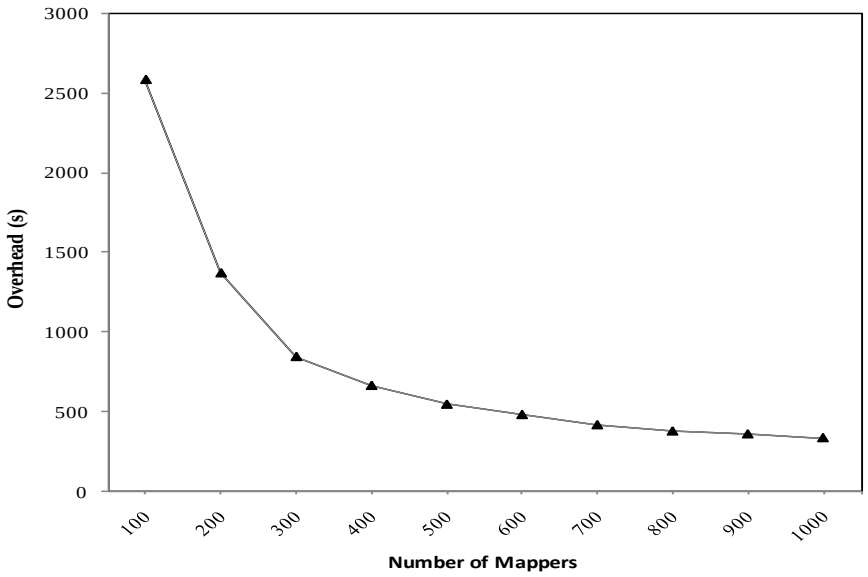

Figure 6.9: The scalability of MRESVM in simulation environments.

The performance of MRESVM can be observed by increasing the data chunk size that is processed by each *mapper*. A comparison of simulating results of *mapper* overhead between chunk size 11.4 MB and 100MB is presented in Figure 6.10 which indicates higher performances with chunk size of 100 MB due to the involvement of smaller number *mapper* waves.

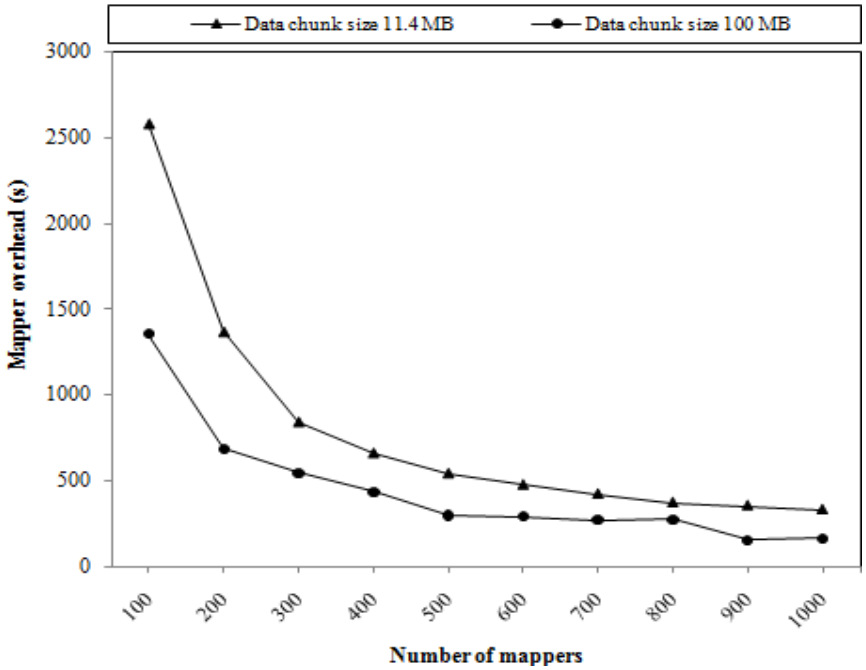

Figure 6.10: Comparison of simulation results between chunk sizes 11.4 MB and 100MB.

The performance of MRESVM can be further observed by changing the CPU power. The CPU power is measured by the size of data processed per seconds. A comparison of simulating results between CPU power of 0.1 and 0.9 MB/sec is made as shown in Figure 6.11. The results indicate significant decrease in level of performances for low CPU power.

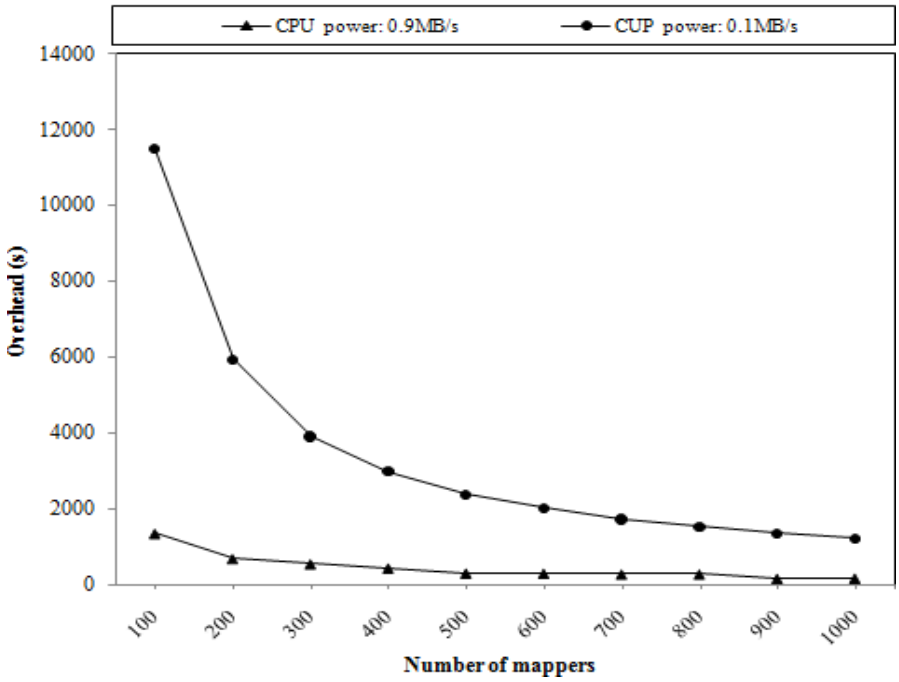

Figure 6.11: Comparison of simulating results with CPU power of 0.1 MB/s and 0.9 MB.

6.6 Summary

This chapter presented MRESVM, a distributed SVM ensemble algorithm for image annotation which re-samples the training data based on bootstrapping and training SVM on each sample in parallel using a cluster of computers. Balanced sampling strategy was used for bootstrapping is introduced to increase classification accuracy for fixed number samples. The chapter concludes with evaluating MRESVM in both experimental and simulation environments showing that the distributed SVM ensemble algorithm reduces the training time significantly and achieves high level of accuracy in classifications.

Chapter 7

Conclusions and Future Work

This chapter presents the main conclusions of the thesis and highlights future research work in the related areas.

7.1 Conclusions

The main solutions proposed in recent years to reduce the semantic gap is the automatic annotation of images which takes into account the low level features of images in the annotation process. Automatic annotation is usually presented as a classification problem. The evaluation of the representative classifiers for image annotation shows that in order to achieve high level of accuracy in image annotation; Support Vector Machine performs better than other classifiers in term of accuracy. However the training time of the classifier is longer than other classifier especially with larger dataset. The evaluation results confirm that SVM models are too large to be used in a practical hence the speed of annotation is lower.

The thesis have presented and evaluated RASMO, a resource aware distributed SVM algorithm that capitalizes on the scalability, parallelism and resiliency of MapReduce for large scale image annotations. By partitioning the training dataset into smaller subsets and optimizing the partitioned subsets across a cluster of computing nodes in multiple stages, the RASMO algorithm reduces the training time significantly while maintaining high level of accuracy in classification. A genetic algorithm based load balancing scheme is introduced to optimize the performance of RASMO in heterogeneous environment. Both the experimental and simulation results have shown the effectiveness of RASMO in training. The load balancing scheme reduces the overhead of RASMO significantly with an increasing levels of heterogeneity showing that the resource aware RASMO can optimize resource utilization in highly heterogeneous computing environments. In addition, data chunks with varied sizes are crucial in speeding up SVM computation in the training process. It is worth pointing out that using different sizes for data chunks has no impact on accuracy in SVM classification due to

the structure of the RASMO algorithm in which the training work in the first few layers is merely a filtering process of removing non-support vectors and the resulting support vectors of the last layer are evaluated for a global convergence by feeding the output of the last layer into the first layer.

The thesis have presented and evaluated RAMSMO, a resource aware distributed Multiclass SVM algorithm that capitalizes on the scalability, parallelism and resiliency of MapReduce for large scale image annotations. RAMSMO is based on OAO multiclass method by partitioning the training dataset into smaller binary data chunks and optimizing the computation of the binary data chunks across a cluster of computing nodes. Experimental results have shown that RAMSMO reduces the training time significantly while maintaining a high level of accuracy in classification. A genetic algorithm based load balancing scheme is introduced to optimize the performance of RAMSMO in heterogeneous environment which addresses the problem of unbalanced multiclass datasets. The processing times of all binary data chunks which have different sizes are equalized, hence reducing training overhead significantly. RAMSMO performances were compared with that of MinMin, MaxMin in load balancing. RAMSMO performed better than both MinMin and MaxMin, and the performance of MinMin is the worst due to the existence of a large number of tasks with short processing times and small number tasks with long processing times.

The thesis have presented and evaluated MRESVM, a scalable distributed SVM ensemble algorithm that capitalizes on the scalability, parallelism and resiliency of MapReduce for large scale image annotations. By re-samples the training data based on bootstrapping and training SVM on each sample in parallel using a cluster of computers. Balanced sampling strategy used for bootstrapping is introduced to increase classification accuracy for fixed number samples. MRESVM is evaluated in both experimental and simulation environments showing that the distributed SVM algorithm reduces the training time significantly and achieves high level of accuracy in classifications. The efficiency of the MRESVM was evaluated with 12 *mappers* which achieves close to 12 times in speedup SVM ensemble. The classification accuracy of accuracy of MRESVM is significantly higher than a single SVM.

7.2 Future work

As part of the future works more than one label can be used to describe an image in order to evaluate classification accuracy of different classifiers. Additionally more image can be used in training process. In this thesis only two low level features represent an image however a number of different low level features with various combinations methods can be used to analyse the behaviour of image annotation system.

In this thesis SMO was used to train SVM classifiers, however there are a number crucial parameters which significantly effects the performances of SMO, therefore an optimization study can be carried out by tuning different parameters to enhance the performances of the SVM based applications. Additionally different data set can be used to analyse the behaviour of the algorithm.

Different methods of distributing SVM can be explored using a cluster environment to further improve the efficiency of training SVM with large data sets while maintaining high level of classification accuracy.

A number of different static load balancing algorithms can be evaluated with the MapReduce framework. Dynamic load balancing scheme can be applied to the MapReduce framework to further enhance the performances MapReduce based application by dynamically allocating work load during the execution time.

The load balancing strategies are implemented based on the simulator HSim. The load balancing schemes can be added to the Hadoop code to achieve better performance in a real Hadoop cluster.

The effect of different sizes of classes on classification accuracy in OAO based multiclass SVM techniques should be analyzed, solving the undesirable bias towards the classes with a smaller training dataset.

The distributed SVM Ensemble algorithm MRESVM was evaluated in small scale homogenous environment however the genetic algorithm described in chapter 4 can be used to optimize the performance of MRESVSM in heterogeneous environment.

A distributed SVM Ensemble algorithm based on Boosting for high accuracy should be considered. Boosting based SVM ensembles have shown high performance in term of accuracy. However the training process is highly computationally expensive. The computation task has to be distributed among a cluster of computers.

In this research work a small scale cluster of participating nodes were employed to evaluate the performance of MapReduce based algorithms, in future work algorithms can be evaluated with a much larger cluster such as Amazon Elastic Compute Cloud (EC2).

Larger number MapReduce parameters which are considered to be crucial for MapReduce based application performances can be evaluated using the HSim simulator to improve performances of MapReduce based applications.

References

[1] S. Abe, Support Vector Machines for Pattern Classification (Advances in Pattern Recognition, Springer Verlag, New York, Inc., pp. 83-127, (2005).

[2] E. Akbas and F. Vural. "Automatic Image Annotation by Ensemble of Visual Descriptors", Intl. Conf. on Computer Vision (CVPR), Workshop on Semantic Learning Applications in Multimedia, pp. 1-8, (2007)

[3] Apache Hadoop! [Online]: http://hadoop.apache.org (Last accessed: 1 April 2011).

[4] F. Bach and M. Jordan, "Predictive low-rank decomposition for kernel methods", In Proceedings of the 22nd International Conference on Machine Learning (ICML), pp. 33-40, (2005).

[5] S. Barrat, S. Tabbone, "Classification and Automatic Annotation Extension of Images Using Bayesian Network", SSPR/SPR, pp. 937-946, (2008).

[6] I. Bartolini, P. Ciaccia ,M. Patella. "Accurate Retrieval of Shapes Using Phase of Fourier Descriptors and Time Warping Distance". IEEE Transaction on Pattern Analysis and Machine Intelligence, 27(1) pp.142–147, (2005).

[7] A. Benitez, S. Chang, "Image classification using multimedia knowledge networks", ICIP (3), pp. 613-616, (2003).

[8] D. Bickson, E. Yom-Tov, D. Dolev, "A Gaussian belief propagation solver for large scale support vector machines", In Proceedings of the 5th European Conference on Complex Systems, pp. 1640-1648, (2008).

[9] O. Boiman, E. Shechtman, M. Irani "In Defense of Nearest-Neighbor Based Image Classification", Computer Vision and Pattern Recognition (CVPR), pp. 1-8, (2008).

[10]L. Bottou, A. Bordes, S. Ertekin, LASVM, [Online]: http://leon.bottou.org/projects/lasvm (Last accessed: 24 March 2011).

[11] R. Bouckaert. "Practical Bias Variance Decomposition", Australasian Conference on Artificial Intelligence, pp. 247-257, (2008).

[12] M, Boutell , J. Luo , X. Shen, "Brown learning multi-label scene classification, Pattern Recognition", 37(9), pp. 1757-1771, (2004).

[13] G. Brassard and P. Bratley, "Algorithmic: theory and practice", Prentice-Hall, (1988).

[14] C. Breen, L. Khan, A. Ponnusamy, "Image Classification Using Neural Networks and Ontologies", DEXA Workshops pp. 98-102, (2002).

[15] L. Breiman, "Bagging predictors, Machine Learning" 24(2), pp.123-140 (1996).

[16] L. Breiman, "Bias, variance and arcing classifiers", Technical Report TR 460, Statistics Department, University of California, Berkeley, CA, (1996).

[17] D. Brickley and R. Guha, "Resource description framework (RDF) schema specification 1.0", W3C Candidate Recommendation, pp. 2000-2013, (2000).

[18] G. Brown, J. Wyatt, R. Harris, X. Yao, "Diversity creation methods: a survey and categorisation", Information Fusion 6(1), pp. 5-20, (2005).

[19] H. Byun, S. Lee, "Applications of Support Vector Machines for Pattern Recognition: A Survey", SVM, pp. 213-236, ((2002).

[20] L. Cao, S. Keerthi, C. Ong, P. Uvaraj, X. Fu, H. Lee, "Developing parallel sequential minimal optimization for fast training support vector machine", Neurocomputing, 70 (1-3), pp. 93-104, (2006).

[21] L. Cao, S. Keerthi, C. Ong, J. Zhang, U. Periyathamby, X. Fu, H. Lee, "Parallel sequential minimal optimization for the training of support vector machines", IEEE Transactions on Neural Networks, 17 (4), pp. 1039-1049, (2006).

[22] B. Catanzaro, N. Sundaram, K. Keutzer, "Fast support vector machine training and classification on graphics processors" In Proceedings of the 25th International Conference on Machine learning (ICML), pp. 104-111, (2008).

[23-] G. Cha, C. Chung, "A New Indexing Scheme for Content-Based Image Retrieval", Multimedia Tools Applications, 6(2), pp. 263-288, (1998).

[24] E. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, "PSVM: parallelizing support vector machines on distributed computers", In Proceedings of Advances in Neural Information Processing Systems, pp. 257–264, (2007).

[25] O. Chapelle, P. Haffner, V. Vapnik, "Support vector machines for histogram-based image classification", IEEE Transactions on Neural Networks 10(5), pp. 1055-106, (1999).

[26] Y. Chen, J. Wang, "Image categorization by learning and reasoning with regions", Journal of Machine Learning Research, 5 pp. 913-939, (2004).

[27] H. Chih-Wei and L. Chih-Jen, "A comparison of methods for multiclass support vector machines", IEEE transactions on neural networks, 13(2), pp. 415-425, (2002).

[28] V. Chitkara, M. Nascimento,C. Mastaller, "Content based image retrieval using binary signatures." In Technical Report TR0018, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada (2000).

[29] C. Chu, S. Kim, Y. Lin, Y. Yu, G. Bradski, K. Olukotun, "Map-Reduce for machine learning on multicore", B. Schölkopf, J.C. Platt, T. Hoffman (Eds.), NIPS, MIT Press, pp. 281-288, (2006).

[30] F. Colas and P. Brazdil, "Comparison of SVM and some older classification algorithms in text classification tasks", In Proceedings of IFIP-AI World Computer Congress, pp. 169-178, (2006).

[31] R. Collobert, S. Bengio, and Y. Bengio, "A parallel mixture of SVMs for very large scale problems", Neural Computation, 14(5), pp. 1105-1114, (2002).

[32] Corel Image Databases, [Online]: http://www.corel.com (Last accessed: 24 March 2011).

[33] K. Crammer, Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines", Journal of Machine Learning Research, 2, pp. 265–292, (2001).

[34] C. Cusano, G. Ciocca, R. Schettini, "Image annotation using SVM" Proc. SPIE, Vol. 5, pp. 304-330, (2003).

[35] I. Daryle-Niedermayer, "An Introduction to Bayesian Networks and Their Contemporary Applications", Innovations in Bayesian Networks, pp. 117-130, (2008).

[36] A. Davison, D. Hinkley, E. Schechtman, "Efficient bootstrap simulation", Biometrika **7**3 , pp. 555 – 566, (1986).

[37] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters", Communications of the ACM, 51(1), pp. 107-113, (2008).

[38] P. Derbeko, R. El-Yaniv, R. Meir, "Variance Optimized Bagging", ECML, pp. 60-71, (2002).

[39] T. Dietterich, G. Bakiri, "Solving multiclass learning problems via error-correcting output codes", Artificial Intelligence Research, 2, pp. 263-286, (1995).

[40] T. Do, V. Nguyen, F. Poulet, "Speed up SVM algorithm for massive classification tasks", In Proceedings of the 4th International Conference on Advanced Data Mining and Applications (ADMA), pp. 147-157, (2008).

[41] T. Do and F. Poulet, "Classifying one billion data with a new distributed SVM algorithm", In Proceedings of the international Conference on Research, Innovation and Vision for the Future (RIVF), pp.59-66, (2006).

[42] P. Domingos, "A Unified Bias-Variance Decomposition for Zero-One and Squared Loss", In Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pp. 564-569, (2000).

[43] J. Dong, A. Krzyżak, C. Suen, "A fast parallel optimization for training support vector machine", In Proceedings of the 3rd International Conference on Machine learning and Data Mining in Pattern Recognition (MLDM), pp. 96-105, (2003).

[44] K. Duan, S. Keerthi, "Which is the best multiclass SVM method? an empirical study", In Proceedings of the 6th International Workshop on Multiple Classifier Systems, pp. 278-285, (2005).

[45] R. Duda and P. Hart, "Pattern Classification and Scene Analysis", Wiley, New York, (1973).

[46] P. Duygulu, K. Barnard, N. de Fretias, D. Forsyth, "Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary". In Proceedings of the European Conference on Computer Vision, pages 97-112, (2002).

[47] R. Esposito and L. Saitta, "Monte Carlo Theory as an Explanation of Bagging and Boosting" In Proceeding of the Eighteenth International Joint Conference on Artificial Intelligence, pp. 499–504, (2003).

[48] J. Fan, Y. Gao, H. Luo, "Hierarchical Classification for Automatic Image Annotation", ACM SIGIR, Amsterdam, pp. 111-118, (2007).

[49] J. Fan, Y. Gao , H. Luo , G, Xu, "Automatic image annotation by using concept-sensitive salient objects for image content representation", In Proceedings of the 27th Annual International Conference on Research and Development in Information Retrieval (SIGIR), pp. 361-368, (2004).

[50] H. Feng, T. Chua, "A bootstrapping approach to annotating large image collection", Multimedia Information Retrieval, pp. 55-62, (2003).

[51] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm", In *Machine Learning: Proceedings of the Thirteenth International Conference*, pp. 148–156, (1996).

[52] Y. Freund and R. Schapire, **"A short introduction to boosting"** *Journal of Japanese Society for Artificial Intelligence*, 14(5), pp. 771—780, (1999).

[53] H. Friedman, "On bias, variance, 0/1 loss and the curse of dimensionality", *Data Mining and Knowledge Discovery*, pp. 55–77, (1997).

[54] G. Fumera, F. Roli, A. Serrau, "Dynamics of Variance Reduction in Bagging and Other Techniques Based on Randomisation", Multiple Classifier Systems, pp. 316-325, (2005).

[55] S. Ghemawat, H. Gobioff, S. Leung, "The Google file system" In Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP), pp. 29-43, (2003).

[56] D. Gillick, A. Faria, J. DeNero, "MapReduce: Distributed Computing for Machine Learning", [Online]http://www.icsi.berkeley.edu/~arlo/publications/gillick_cs262a_proj.pdf (Last accessed: 22 March 2011).

[57] Z. Gong, Q. Liu, J. Zhang, "Web Image Retrieval Refinement by Visual Contents", WAIM, pp. 134-145, (2006).

[58] H. Graf, E. Cosatto, L. Bottou, I. Durdanovic, V. Vapnik, "Parallel support vector machines: The Cascade SVM", In Proceedings of Advances in Neural Information Processing Systems (NIPS), pp. 521-528, (2004).

[59] J. Guo, N. Takahashi, T. Nishi, "An efficient method for simplifying decision functions of support vector machines", IEICE Transactions, 89-A (10), pp. 2795-2802, (2006).

[60] T. Hastie, R. Tibshirani, "Classification by pairwise coupling", In Proceedings of the Conference on Advances in Neural Information Processing Systems, pp. 451-471, (1997).

[61] T. Hazan, A. Man, A. Shashua, "A parallel decomposition solver for SVM: distributed dual ascend using fenchel duality", In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-8 (2008).

[62] T. Heskes, "Bias/Variance Decompostion for Likelihood-Based Estimators", *Neural Computation*, 10:1, pp. 425–1433, (1998).

[63] S. Herrero-Lopez, J. Williams, A. Sanchez, "Parallel multiclass classification using SVMs on GPUs", In Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU), pp. 2-11, (2010).

[64] M. Hofsheimer, A. Siebes, "Data Mining: The Search for Knowledge in Databases", Report CS-R9406, CWI, pp. 127-189, (1994).

[65] L. Hollink, A. Schreiber, J. Wielemaker, B. Wielinga, "Semantic Annotation of Image Collections", In Proceedings of the KCAP'03 Workshop on Knowledge Capture and Semantic Annotation, pp. 1-8, (2003).

[66] Z. Hua X. Wang Q. Liu H. Lu, "Semantic knowledge extraction and annotation for web images" In Proceedings of the 13th annual ACM international conference on Multimedia, pp. 467 – 470, (2005).

[67] G. Huang, K. Mao, C. Siew, D. Huang, "Fast modular network implementation for support vector machines", IEEE Transactions on Neural Networks, 16(6) pp. 1651-1663, (2005).

[68] J. Huang, S. Kumar, R. Zabih, "Automatic Hierarchical Color Image Classification," EURASIP Journal on Applied Signal Processing, 2, pp. 151-159, (2003).

[69] J. Jeon, V. Lavrenko, R. Manmatha, "Automatic Image Annotation and Retrieval Using Cross-Media Relevance Models", In Proceedings of the ACM Special Interest Group on Information Retrieval, pp. 119-126, (2003).

[70] R. Jin, J. Chai, L. Si "Effective Automatic Image Annotation via a Coherent Language Model and Active Learning" In Proceedings of MM, pp. 892-899, (2004).

[71] F. Jing, M. Li, L. Zhang, H.-J. Zhang, B. Zhang, " Learning in region based image retrieval", Proceedings of the International Conference on Image and Video Retrieval, pp. 206–215 (2003).

[72] D. Joshi, J. Ze Wang, J. Li, "The Story Picturing Engine - a system for automatic text illustration.", IEEE Publication, pp. 68-89, (2006).

[73] M. Kane, A. Savakis, "Bayesian Network Structure Learning and Inference in Indoor vs. Outdoor Image Classification", IEEE publications, pp. 479-482, (2004).

[74] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy, Improvements to Platt's SMO algorithm for SVM classifier design," Neural Computation, 13(3), pp. 637-649, (2001).

[75] S. Keerthi, S. Sundararajan, K. Chang, Ch.Hsieh, Ch. Lin, "A sequential dual method for large scale multi-class linear SVMs", In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 408-416, (2008).

[76] N. Khalid Alham, M. Li, S. Hammoud, Y. Liu, M. Ponraj, "A distributed SVM for image annotation", In Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp. 2983-2987, (2010).

[77] N. Khalid Alham, M. Li, S. Hammoud and H. Qi, Evaluating machine learning techniques for automatic image annotations, in: Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp. 245-249 (2009).

[78] H. Kim, S. Pang, H. Je, D. Kim, S. Bang, "Support Vector Machine Ensemble with Bagging", SVM, pp. 397-407, (2002).

[79] S. Knerr, L. Personnaz, G. Dreyfus, "Single-layer learning revisited: A stepwise procedure for building and training a neural network, in: F. Fogelman Soulié and J. Hérault" (Eds.) Neurocomputing: Algorithms, Architectures and Applications, volume F68 of NATO ASI Series, pp. 41-50, (1990).

[80] R. Kohavi and D. H.Wolpert, "Bias plus variance decomposition for zero-one loss functions", In *Proceedings of the Thirteenth International Conference on Machine Learning, The Seventeenth International Conference on Machine Learning*, pp. 275–283, (1996).

[81] B. Kuszmaul, "Cilk provides the "best overall productivity" for high performance computing (and won the HPC challenge award to prove it)", In Proceedings of the 9th ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp. 299-300 (2007).

[82] S, Kullback, "Letter to the Editor: The Kullback–Leibler distance", *The American Statistician*, 41 (4), pp. 340–341, (1987).

[83] D. Kun, L. Yih, A. Perera, "Parallel SMO for Training Support Vector Machines", SMA 5505 Project Final Report, (2003).

[84] R. Lämmel, "Google's MapReduce programming model — revisited", Science of Computer Programming, 70 (1), pp. 1-30, (2008).

[85] B. Le Saux, G. Amato, "Image recognition for digital libraries", In Proceedings of the ACM Multimedia Workshop on Multimedia Information Retrieval (MIR), pp. 91-98, (2004).

[86] Z. Lei, Y. Yang, Z. Wu, "Ensemble of Support Vector Machine for Text-Independent Speaker Recognition", International Journal Computer Science and Network Security 6 (5), pp. 163-167, (2006).

[87] L. Lepisto, I. Kunttu, A. Visa, "Rock image classification based on k-nearest neighbour voting" Vision, Image and Signal Processing, IEE Proceedings, 153, I pp. 475- 482, (2006).

[88] T. Li, C.Zhang, and M.Ogihara, "A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression", Bioinformatics, 20 (15), pp. 2429-2437, (2004).

[89] J. Lim, Q. Tian, P. Mulhem, "Home Photo Content Modeling for Personalized Event-Based Retrieval", IEEE MultiMedia, 10 (4), pp. 28-37, (2003).

[90] Lire, An open source Java content based image retrieval library, [Online]: http://www.semanticmetadata.net/lire/ (Last accessed: 24 March 2011).

[91] X. Liu, L. Zhang, M. Li, H. Zhang, D. Wang, "Boosting image classification with LDA-based feature combination for digital photograph management", Pattern Recognition 38(6), pp. 887-901, (2005).

[92] Y. Liu, D. Zhang, G. Lu, W. Ma, "A survey of content-based image retrieval with high-level semantics", Pattern Recognition 40(1), pp. 262-282, (2007).

[93] F. Long, H. Zhang, D. Feng, "Fundamentals of Content-Based Image Retrieval", Multimedia Information Retrieval and Management Technological Fundamentals and Applications, D. Feng and W. C. Siu, eds., Springer Publishers, pp. 1-26, (2003).

[94] B. Lu, K. A. Wang, M. Utiyama, H. Isahara, "A part-versus-part method for massively parallel training of support vector machines", In Proceedings of International Joint Conference on Neural Networks (IJCNN), pp. 735-740, (2004).

[95] Y. Lu, Q. Tian, T. Huang, "Interactive Boosting for Image Classification", MCAM, pp. 315-324, (2007).

[96] Y. Lu, C. Hu, X. Zhu, H.J. Zhang, Q. Yang, "A unified framework for semantics and feature based relevance feedback in image retrieval systems", In Proceedings of the eighth ACM international conference on Multimedia, pp. 31–37 (2000).

[97]Y. Lu, V. Roychowdhury, L. Vandenberghe, "Distributed parallel support vector machines in strongly connected networks", IEEE Transactions on Neural Networks, 19 (7), pp. 1167-1178, (2008).

[98] J. Luo, A. Savakis, "Indoor vs outdoor classification of consumer photographs using low-level and semantic features", ICIP (2), pp. 745-748, (2001).

[99] A. Makadia, V. Pavlovic, S. Kumar, "A New Baseline for Image Annotation", ECCV (3), pp. 316-329, (2008).

[100] O. Marques, N. Barman, "Semi-automatic Semantic Annotation of Images Using Machine Learning Techniques", International Semantic Web Conference, pp. 550-565, (2003).

[101] L. Mason, P. Bartlett, J. Baxter, "Improved Generalization Through Explicit Optimization of Margins". Machine Learning 38(3), pp. 243-255, (2000).

[102] B. McBride, A. Seaborne, J. Carroll, "Jena tutorial for release 1.4.0" *Technical report*, Hewlett-Packard Laboratories, Bristol, UK, (2002).

[103] Medusa, [Online]: http://www.lsc-group.phys.uwm.edu/beowulf/medusa/index.html (Last accessed: 3 April 2011).

[104] Y, Ming-Hsuan , B. Moghaddam, "Support vector machines for visual gender classification", In Proceedings of the International Conference on Pattern Recognition (ICOR), pp. 5115-5118, (2000).

[105] J. Munoz-Mari, A. Plaza, J. A. Gualtieri, G. Camps-Valls, "Parallel implementations of SVM for earth observation", In Parallel Programming, Models and Applications in Grid and P2P Systems, F. Xhafa (Ed.), pp. 292–312, (2009).

[106] T. Ojala, M. Pietikäinen, T. Mäenpää, "Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 971 – 987, (2002).

[107] E. Osuna, R. Freund, F. Girosit, "Training support vector machines: an application to face detection", In Proceedings of Computer Vision and Pattern Recognition (CVPR), pp.130-136, (1997).

[108] J. Pakkanen, A. Ilvesmäki, J. Iivarinen, "Defect Image Classification and Retrieval with MPEG-7 Descriptors", SCIA, pp. 349-355, (2003).

[109] D. Park, Y. Jeon, C. Won, "Efficient use of local edge histogram descriptor" ACM Multimedia Workshops pp. 51-54, (2000).

[110] D. Pavlov, J. Mao, B. Dom "Scaling-Up Support Vector Machines Using Boosting Algorithm", ICPR, pp. 2219-2222, (2000).

[111] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, M. Stonebraker, "A comparison of approaches to large-scale data analysis", In Proceedings of the 35th ACM SIGMOD International Conference on Management of Data, pp. 165-178, (2009).

[112] K. Petridis, D. Anastasopoulos, C. Saathoff, N. Timmermann, I. Kompatsiaris, S. Staab

"M-OntoMat-Annotizer: Image Annotation. Linking Ontologies and Multimedia Low-Level Features", Engineered Applications of Semantic Web Session (SWEA) at the 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems, Bournemouth, U.K, (2006).

[113] J. Platt, "Sequential minimal optimization: a fast algorithm for training support vector machines", Technical Report, MSR-TR-98-14, Microsoft Research, (1998).

[114] J. Platt, N. Cristanini, J. Shawe-Taylor, "Large margin DAGs for multiclass classification", In Proceedings of Neural Information Processing Systems (NIP), pp. 547-553, (1999).

[115]M. Re, G. Valentini, "Prediction of Gene Function Using Ensembles of SVMs and Heterogeneous Data Sources", Applications of Supervised and Unsupervised Ensemble Methods, pp. 79-91, (2009).

[116] V. Roth, K. Tsuda, "Pairwise coupling for machine recognition of hand-printed Japanese characters", In Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR), pp.1120-1125, (2001).

[117] R. Schapire, Y. Freund, P. Bartlett, W. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods", *The Annals of Statistics*, 26(5), pp. 1651–1686, (1998).

[118] S. Shah, V. Gandhi, "Image Classification Based on Textural Features using Artificial Neural Network (ANN)" Journal-Institution of Engineers India Part ET, pp. 1039-1053, (2004).

[119] J. Shawe-Taylor & N. Cristianini"An Introduction to Support Vector Machines and other kernel-based learning methods", Cambridge University Press, p. 52-67, (2000).

[120] T. Sikora, "The MPEG-7 visual Standards for content description-an overview", IEEE Transaction Circuits System, Video Techniques, 11(6), pp. 696-702, (2001).

[121] C. Silva, U. Lotric, B. Ribeiro, A. Dobnikar, "Distributed Text Classification With an Ensemble Kernel-Based Learning Approach," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40, pp. 287-297, (2010).

[122] A. Smeulders, M.Worring, S. Santini, A. Gupta, R. Jain, "Content based image retrieval at the end of the early years". IEEE Transaction Pattern Analysis Machine Intelegence, 22(12), pp. 1349-1380, (2000).

[123] M**.** Srikanth**,** J. Varner, M**.** Bowden, and D. Moldovan, "Exploiting Ontologies for Automatic Image Annotation" In Proceedings of 28th Annual International ACM SIGIR Conference, pp. 552-558, (2005).

[124] N. Stepenosky, D. Green, J. Kounios, C. Clark R. Polikar, "Majority vote and decision template based ensemble classifiers trained on event related potentials for early diagnosis of Alzheimer's disease," IEEE International Conference Acoustic, Speech and Signal Processing, pp. 901-904, (2006).

[125] M. Stricker and M. Orengo, "Similarity of color images", In SPIE Conference on Storage and Retrieval for Image and Video Databases III, pp. 381-392, (1995).

[126] B. Suh, B. Bederson, "Semi-automatic photo annotation strategies using event based clustering and clothing based person recognition", Interacting with Computers 19(4), pp. 524-544, (2007).

[127] J. Suykens and J. Vandewalle, "Least squares support vector machine classifiers", Neural Processing Letters, 9(3), pp. 293–300, (1999).

[128] Y. Tang, Y. He, S. Krasser, "Highly Scalable SVM Modelling with Random Granulation for Spam Sender Detection", ICMLA, pp. 659-664, (2008).

[129] D. Tao, X. Tang, X. Li, X. Wu, "Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval" IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(7), pp. 1088– 1099, (2006).

[130] The Cilk Project, [Online]: http://supertech.csail.mit.edu/cilk/ (Last accessed: 24 March 2011).

[131] C. Tsai, K. McGarry, J. Tait, "CLAIRE: A modular support vector image indexing and classification system", ACM Transactions on Information and System Security 24(3), pp. 353-379, (2006).

[132] C. *Tsai*, C. Hung, "Automatically annotating Images with Keywords*:* A Review of Image Annotation Systems", Recent Patents on Computer Science, 1, pp. 55-68, (2008).

[133] T. Tseng**,** C. Lee, J. Su, "Classify by representative **or.** Associations **(**CBROA): A hybrid approach for image classification**",** In Proceedings of the 6th international workshop on Multimedia data mining: mining integrated media and complex data, pp. 22-28, (2005.

[134] G. Valentini and T. Dietterich, "Low Bias Bagged Support Vector Machines", ICML pp. 752-759, (2003).

[135] G. Valentini and T. Dietterich, "Bias-variance analysis of Support Vector Machines for the development of SVM-based ensemble methods", Journal of Machine Learning Research, 5, pp. 725-775, (2004).

[136] N. Vasconcelos, "From Pixels to Semantic Spaces: Advances in Content-Based Image Retrieval" Published by the IEEE Computer Society, pp. 20-26, (2007).

[137] J. Venner, Programming Hadoop, Springer, pp 1-407, (2009).

[138] J. Vompras, and S. Conrad, "A semi-automated Framework for Supporting Semantic Image Annotation" , In Proceedings of 5th International Workshop on Knowledge Markup and Semantic Annotation at the 4rd International Semantic Web Conference (ISWC) pp. 105-

109, (2005).

[139] C. Wang, F. Jing, L. Zhang, H. Zhang, "Content-Based Image Annotation Refinement", IEEE Conference on Computer Vision and Pattern Recognition, pp.1-8, (2007).

[140] Y. Wang and H.J Zhang "Content-Based Image Orientation Detection with Support Vector Machines", in Proc. IEEE Workshop on Content-based Access of Image and Video Libraries, December, pp. 17-23, (2001).

[141] H. Wang, S. Liu, L. Chia, "Does ontology help in image retrieval?: a comparison between keyword, text ontology and multi-modality ontology approaches", ACM Multimedia, pp. 109-112, (2006).

[142] G. Wang, A. Butt, P. Pandey, K. Gupta, "Using realistic simulation for performance analysis of MapReduce setups", In Proceedings of the 1st ACM workshop on Large-Scale System and Application Performance (LSAP), pp. 19-26, (2009).

[143] C. Waring and X. Liu, "Face detection using spectral histograms and SVMs", IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 35(3), pp. 467-476, (2005).

[144] Weka 3, [Online]: http://www.cs.waikato.ac.nz/ml/weka (Last accessed: 1 April 2011).

[145] L. Wenyin, S. Dumais, Y. Sun, H. Zhang, M. Czerwinski B. Field, "Semi-Automatic Image Annotation", In Proceedings of INTERACT2001—8th IFIP TC.13 Conference on Human-Computer Interaction, Hirose, M. (Ed.), IOS Press, pp.326-333, (2001).

[146] S. Winters-Hilt and K. Armond, "Distributed SVM learning and support vector reduction", [Online]: http://cs.uno.edu/~winters/SVM_SWH_preprint.pdf (Last accessed: 24 March 2011).

[147]W. Wong and S. Hsu, "Application of SVM and ANN for image retrieval", European Journal of Operational Research, 173 (3), pp. 938-950, (2006).

[148] K. Woodsend and J. Gondzio, "Hybrid MPI/OpenMP parallel linear support vector machine training", Journal of Machine Learning Research, 10, pp. 1937-1953, (2009).

[149] M. Xu, J. Wang, T. Chen, "Improved Decision Tree Algorithm: ID3", Intelligent Computing in Signal Processing and Pattern Recognition, pp. 141-149, (2006).

[150] G. Yan, G. Ma, L. Zhu, "Support vector machines ensemble based on fuzzy integral for classification", ISNN, pp. 974–980, (2006).

[151] J. Yang, "An improved cascade SVM training algorithm with crossed feedbacks", In Proceedings of 1st International Multi-Symposium of Computer and Computational Sciences (IMSCCS), pp. 735-738, (2006).

[152] A. Yavlinsky, E. Schofield, S. Riuger, "Automated image annotation using global features and robust nonparametric density estimation", In Proceedings of the Computer Image and Video Retrieval conference (CIVR), pp. 507-517, (2005).

[153] G. Zanghirati and L. Zanni, "A parallel solver for large quadratic programs in training support vector machines", Parallel Computing, 29 (4), pp. 535-551, (2003).

[154] L. Zhang, F. Lin, B. Zhang, "Support vector machine learning for image retrieval", In Proceedings of International Conference on Image Processing, pp. 721-724, (2001).

[155] C. Zhang, P. Li, A. Rajendran, Y. Deng, "Parallel multicategory support vector machines (PMC-SVM) for classifying microcarray data", In Proceedings of the 1st International Multi-Symposiums on Computer and Computational Sciences (IMSCCS), pp. 110-115, (2006).

[156] Y. Zhao, Y. Zhao, Z. Zhu, J. Pan, "A Novel Image Annotation Scheme Based on Neural Network", ISDA (3), pp. 644-647, (2008).

[157] Y. Zhao and Y. Zhang "Comparison of decision tree methods for finding active objects" Advances in Space Research, 41(12), pp. 1955-1959, (2008).