Hybrid Parallel Cascade Classifier Training for Object Detection

Eanes Torres-Pereira, Herman Martins-Gomes,
Andrey Elísio Monteiro-Brito, and João Marques de Carvalho

Universidade Federal de Campina Grande, Department of Systems and Computing
Av. Aprígio Veloso 882, 58429-900 Campina Grande-PB, Brazil
{eanes,hmg,andrey}@computacao.ufcg.edu.br,
carvalho@dee.ufcg.du.br

Abstract. A drawback of the Viola and Jones framework for object detection in digital images is the large amount of time needed to train the underlying cascade classifiers. In this paper, we propose a novel hybrid approach for parallelizing that framework. The approach employs message passing among computers and multi-threading in the processor cores, hence its hybrid nature. In contrast to related works, which dealt with only parts the original framework, in this paper we considered the complete framework. Besides, the set of weak classifiers obtained by our parallel approach is identical to the one of a serial version. An experimental evaluation on face detection focused on speedup and scalability measures and has shown the improvements of the proposed approach over a serial implementation of the original framework.

Keywords: Adaboost, parallelization, face detection.

1 Introduction

The framework proposed by Viola and Jones [1][2] has become a landmark in the area of face detection and the majority of subsequent works were inspired by that approach [3]. The framework's main techniques are: boosted cascade of weak classifiers, integral image representation, Haar features, and bootstrapping. Approaches for training a face detection classifier (e.g. [4], [5], and [6]) that were inspired by the original work by Viola and Jones [1][2] used some variation of those techniques.

This paper is concerned with the problem of reducing time complexity during cascade classifier training and, therefore, it does not deal with speeding up face detection after training as in the works of Cho et al [7], Hefenbrock et al. [8], and Harvey [9]. Within this context, we report a novel approach to parallelize the framework of cascade classifier training, including sample creation, features and classifier evaluation after training, and the boosting of classifiers. As it will be clear in the next section, the related work focused mostly on the parallelization of the boosting algorithm, whereas our approach employs hybrid techniques to parallelize the entire framework. The remainder of this paper is organized as

E. Bayro-Corrochano and E. Hancock (Eds.): CIARP 2014, LNCS 8827, pp. 810–817, 2014.

follows. After the review of related work, the proposed hybrid parallelization approach is presented in Section 3. An experimental evaluation is presented and discussed in Section 4. The final section contains the conclusions.

2 Related Work

One possibility for the parallelization of the boosting process is to perform trainings using disjoint sets, as it was proposed by Lazarevic and Obradovic [10]. Three boosting possibilities were presented: parallel learning, distributed learning on homogeneous bases, and distributed learning on heterogeneous bases. In all cases, weak classifiers are trained in each processor using subsets of all available data. However, there is no guarantee that the classifiers generated by the parallel approach will be equivalent to the ones generated by the original method, since the classifiers are trained using different image sets in each computer when a distributed approach is employed.

Merler et al. [11] designed a parallel version of Adaboost that did not use bootstrapping. Their focus was to improve the weight update dynamics. At each training step, Adaboost performs a weight update that relies on the results of the previous step. To parallelize the current weight update step, the training process is split in two phases: (1) the algorithm is sequentially executed until a given number of training steps is reached; and (2) the frequency of the asymptotic distribution of the weights is estimated, which makes the current step independent of the previous ones. Therefore, it is possible to train model instances in parallel and aggregate them in the traditional way at the end of the process. However, the experimental analysis of that work did not take into account the disk access time, an important factor that is ignored by all of the reviewed work.

Galtier et al. [12] considered the master-workers approach. At each training cycle, each worker trains its weak classifiers with all of the training set. The master then gets the best classifiers from each worker and selects the one with the lowest error. This strategy is focused on the division of the total set of features among slave processes. However, all the workers must read the entire non-face training samples. As it will be explained in the next section, the image loading stage may be very time consuming and may surpass the training stage itself. Galtier et al. [12] did not mention the impact of image loading tasks.

Huang and Shi [13] distributed the task of searching for the best feature for each weak classifier. Each machine in the distributed system returns the best feature for the root node and the best among them is used for training. The authors performed experiments to measure the processing speed. However, besides not presenting the classifier performance rates, the number of training samples employed was very low for proper classifier training within a face detection problem. They used $5,646$ face images and $13,030$ non-face images, while Jones and Viola [4], years before, used more than 100 million non-face images.

Zeng et al. [14] parallelized the Adaboost algorithm using MPI, OpenMP, and STM (Transactional Memory), following the master-only model. Training a face classifier cascade was modeled as a learning problem using the Adaboost method

in order to evaluate their approach. The number of non-face training samples was not aligned with traditional work in the face detection area (e.g. [1], [2], and [6]). Training was performed using 64,328 images of faces and only 43,712 images of non-faces. Their results indicated that using a hybrid master-only approach yields higher processing speed due to the reduction of inter-process communication.

Among all the reviewed papers, Zeng et al. [14] parallelized a complete cascade of classifiers trained using boosting. The remaining papers in this review addressed the parallelization of only parts of the cascade training, or only the boosting algorithms. Zeng et al. [14] performed a graphical analysis of speedup, but did not consider scalability and efficiency. The only reviewed paper that evaluated scalability and speedup is the one by Galtier et al. [12]. However, their time measures were computed regarding only the classifier combination loop. The authors presented two speedup graphs: one for execution times versus the number of used processors, and another one for speedup measures versus the number of processors. The time of the first graph decays exponentially, but the speedup in the second graph increases almost linearly. Huang and Shi [13] also presented an evaluation based on speedup. They proposed a distributed architecture in which the programmer itself is responsible for dividing the tasks into sequential or parallel parts. They presented the speedups for 4 situations resulting from the combination of the number of processors (2 or 4) and number of features employed (32 or 64). When 4 processors were used, the speedup was 2.66. There was no scalability evaluation.

3 Proposed Approach

The approach by Viola and Jones [1][2] is mainly based on the techniques of boosting weak classifiers, and bootstrapping. To create weak classifiers, decision trees with only a single node are trained. The Viola and Jones method, illustrated in Figure 1, is composed of four main steps that are amenable to parallelization: (1) data loading and classification procedures, (2) decision tree training, (3) classification after the addition of each new weak classifier, (4) the definition of the stage threshold.

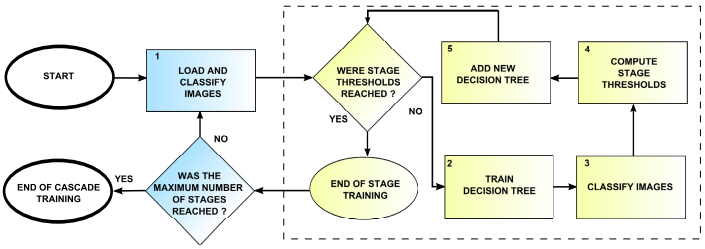

Fig. 1. Viola and Jones approach and indication of modules prone to parallelization

In Figure 1 there are two main classification steps. The first one, corresponding to module number 1, uses all classifiers from all trained stages. The second classification step, corresponding to module 3, uses only the classifiers for the stage that is currently being trained. Another part of the Viola and Jones method that may be easily parallelized is the stage threshold definition, marked as module 4. That definition is performed by classifying the face images, then choosing the decision three threshold that achieves the previously determined minimum hit rate. Finally, decision tree training (module 2) may also be parallelized. In this process, all features for all images must be evaluated, which requires the processing of a (*number of images*) $\times$ (*number of features*) matrix.

In this work, image samples are cropped only when they are needed for training. A total of $4,365$ images with varying resolutions were used for non-face sample cropping, which is much more feasible than having to save millions of samples onto disk. There are five variables that must be kept in memory: the coordinates of the upper left corner of the sample, the displacement step, the width, the height, and the crop scale. The available images are divided among the cluster computers. To perform an uniform distribution of non-face images, each computer must have stored in its local hard disk at least the set of images allocated to that computer. It was not necessary to use any type of distributed or network file system to share the negative sample images. The algorithm uses the identification number of the processor in order to divide the set of available images among the working processes.

The next step is the decision tree training, which requires a matrix containing the features extracted from all training image samples. That matrix must be equally divided among the computers. Moreover, a further division is required to feed the multiple processor cores. Decision tree training is the central part of the proposed approach. Message passing is used for communication among the various computers, whereas multithreading is used for communication among the multiple processing cores of a given computer.

In the Viola and Jones [1] method for cascade classifier training there are two processes in which classification occurs. In the first situation, during loading, each training image is classified. Initially, in the first stage, all training images (positives and negatives) are classified as positive. From the second stage onwards there will be a trained classifier. At every stage, the matrix of extracted image features is updated. Only images incorrectly classified are used for feature extraction, and those features are assigned to the matrix. This way, the harder (more difficult for the classifier) images are used for feature extraction and are assigned to the matrix.

As the loading of images is parallelized, subsequent classification also occurs in the same computer where the image is being loaded. This implies that the weak classifiers must be sent to all computers at the end of each stage training. After training each weak classifier, the ensemble of weak classifiers must be evaluated, this situation is explained next. In the proposed approach, all computers must have copies of the weak classifiers.

The construction of decision trees starts by selecting an attribute to be the root node. Then the set of values are split up into subsets. The next problem is: how to select the attribute to be the root. A common approach is to measure the purity of the attributes and to choose the purest one [15]. When training decision trees with thousands of images and hundred of thousand of available features, the purity computation may be very slow. Therefore, in the proposed parallel approach we employed message passing and multiple threads to deal with that computations.

For image samples with resolution of 21 $\times$ 21 pixels there are more than hundred of thousands Haar-like features available. For each image sample, all the Haar-like features must be evaluated in terms of purity. The first step in the parallelization of the process of purity evaluation is to send to each computer in the cluster a set of images for which the purity of Haar-like features will be computed. At each computer, the set of images is further divided among the available processing cores. At the end of the purity computation, each computer sends to the master computer the evaluated purities. The master computer sorts the purities and chooses the one with higher value. The value of the feature with highest purity will be used as the core of a weak classifier. That weak classifier is added to the set of weak classifiers of the stage being trained.

After training, decision trees are used to classify the positive training samples. The decision trees composed of stump classifiers. The thresholds of each tree are sorted by classification rate in order to determine the best threshold that achieves the minimum hit rate set as a parameter. After choosing the threshold from the classification of positive samples, the negative samples are then classified. In this work, the two classification procedures are parallelized by message passing.

4 Experimental Evaluation

In this section two types of evaluation are presented: statistical, using boxplots analysis; and using the traditional speedup and scalability measures. Initially, we present the analysis of the processing times. Then, speedup and scalability are evaluated.

Training of the classifier cascades is affected by the type of image, i.e., samples cropped from images with cluttered background will be more difficult to train for weak classifiers. Therefore, two sets of experiments were performed with ten configurations each in order to evaluate speedup and scalability parallelization. Experimental configurations differ on how non-face images are cropped to generate the training samples. To obtain variability in the experiments, two parameters were systematically changed for cropping negative image samples: displacement step (number of pixels used as measure to determine the coordinates of next crop region), and resizing scale (factor used to determine the new crop size). The displacement step ranges from 2 up to 6 pixels, and the scales were 1.1 and 1.2. Speedup and scalability were computed using the average processing times from all configurations.

A total of 100 experiments were performed using the previously mentioned configurations: 10 repetitions using different images for each configuration. From

those experiments, 50 were performed to evaluate speedup and 50 to evaluate scalability. Each configuration was applied once for each number of used computers, ranging from 1 to 5. Due to the large amount of data collected from those experiments, boxplot diagrams are used to summarize those results, which are presented in Figures 2(a) and 2(b).

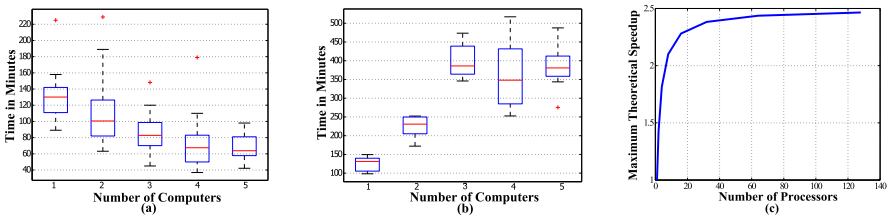

Fig. 2. Boxplot diagrams using data collected for evaluating (a) speedup and (b) scalability, and graph showing (c) maximum theoretical speedup estimates

Speedup evaluation experiments were performed using the same amount of images regardless the number of computers (1,000 face samples, and 2,000 non-face samples at each training stage). For scalability evaluation, the ratio between the amount of images and the number of computers was kept constant. For instance, the first experiment employed 1 computer and 3,000 images (1,000 face samples, and 2,000 non-face samples) and the fifth experiment employed 15,000 images (5,000 face samples, and 10,000 non-face samples). In comparison with the Viola and Jones [2] work (who reported training with 350 million non-face image samples), the results presented in Table 1 considered approximately 6 billion samples. The number of processors, and the corresponding average processing times, standard deviations and scalability are shown in Table 1. Scalability has been obtained with a similar equation to the one used for speedup (as discussed next), the difference being only the amount of images required for different number of processors.

It may be observed in Figure 2(a) that as more computers are added the median values of processing time decrease indicating an increase in processing speed. Regarding processing time of scalability distributions, it is possible to observe from Figure 2(b) that stabilization occurs from 3 computers onwards.

Table 1. A: speedup. B: scalability and efficiency. The measures are presented with corresponding mean time and standard deviation for the different numbers of computers used in the experiments.

A				B			
Number of Computers	**Mean Time (Min.)**	**Standard Deviation**	**Speedup**	**Mean Time (Min.)**	**Standard Deviation**	**Scalability**	**Efficiency**
1	128.67	28.75	1.00	128.31	18.63	1.00	1.00
2	109.39	38.43	1.18	237.35	37.32	0.54	0.27
3	84.00	25.17	1.53	402.12	49.07	0.32	0.11
4	71.56	31.98	1.80	388.82	107.30	0.33	0.08
5	67.01	14.77	1.92	398.21	87.80	0.32	0.06

Speedup is a measure commonly used in parallel computing to determine how much a parallel program is faster than a corresponding sequential one [16], which is defined according to the ratio between the execution time T_s of the sequential algorithm (running on a single processor, i.e., $s = 1$) and the execution time T_p of the parallel algorithm executed by p processors. When $S_p = p$, linear or ideal speedup is achieved, although higher speedups may be achieved which are called superlinear speedups [17]. For example, if 100 seconds are necessary for running an algorithm using 1 processor, ideally it should be necessary 50 seconds for running the same algorithm with the same processing load when using 2 computers.

The data presented in Table 1 shows an increase in processing speed as more processors are added. Amdahl's Law [18] states that if the fraction of a program that may be parallelized is P, then the maximum speedup using its parallelized version running on N computers is defined according to Equation (1). The value of P is estimated using Equation (2) [16], the term SU indicates the speedup measured for N computers. Therefore, if the value of P is estimated using the total speedup obtained by 5 processors, the result will be 0.60. From that value of P, the estimated speedup curve for our parallelized approach, as a function of the number of processors, is given in Figure 2(c). The efficiency of parallel processing is given by the ratio between speedup (SU) and the number of processors (N) used to obtain such speedup.

$$SU = \frac{1}{(1-P) + \frac{P}{N}}. \tag{1}$$

$$P_{estimated} = \frac{\frac{1}{SU} - 1}{\frac{1}{N} - 1}. \tag{2}$$

The weak classifiers obtained by the parallel approach were compared with weak classifiers obtained by the serial version for the same conditions of training and same images, and it was concluded that they are identical.

5 Conclusion

In this paper, a novel hybrid approach to parallelize the Viola and Jones [1][2] framework was presented. The approach explores both message passing and multi-threading programming and may be used in different parallel computing environments (small clusters of large machines or large clusters of small machines, as it is more common in cloud environments). In addition, the proposed approach also implements a parallel technique for the generation of non-face images that addresses some practical issues (e.g., lengthy disk read phases, large number of files). The proposed approach was evaluated by means of two metrics: scalability, and speedup. For both metrics billions of non-face images were used in the training phase. The parallel proposed approach achieved true positive rates higher than 0.95 when trained with more than 2 billion non-face samples. Moreover, boxplot analysis (Figure 2) showed that speedup and scalability started to

be stable for number of used computers higher than 3. Finally, in contrast to related works, which dealt with only parts of the Viola and Jones framework, in this paper we addressed the problem of parallelizing their complete framework, which included the cropping of new image samples and the evaluation of previously trained classifier stages, and a realistic view of the expected results.

References

1. Viola, P., Jones, M.: Robust real-time object detection. In: Workshop on Statistical and Computational Theories of Vision, pp. 1–25 (2001)
2. Viola, P., Jones, M.J.: Robust real-time face detection. International Journal of Computer Vision 57(2), 137–154 (2004)
3. Zhang, C., Zhang, Z.: A survey of recent advances in face detection. Technical report, Microsoft Research, MSR-TR-2010-66 (2010)
4. Jones, M., Viola, P.: Fast multi-view face detection. Technical report, Mitsubishi Electric Research Laboratories (2003)
5. Wu, B., Ai, H., Huang, C., Lao, S.: Fast rotation invariant multi-view face detection based on real adaboost. In: International Conference on Automatic Face and Gesture Recognition, pp. 79–84 (2004)
6. Huang, C., Ai, H., Li, Y., Lao, S.: High-performance rotation invariant multiview face detection. Transactions on Pattern Analysis and Machine Intelligence 29(4), 671–686 (2007)
7. Cho, J., Benson, B., Mirzaei, S., Kastner, R.: Parallelized architecture of multiple classifiers for face detection. In: International Conference on Application-specific Systems, Architectures and Processors, pp. 75–82 (2009)
8. Hefenbrock, D., Oberg, J., Thanh, N.T.N., Kastner, R., Baden, S.B.: Accelerating viola-jones face detection to fpga-level using GPUs. In: International Symposium on Field-Programmable Custom Computing Machines, pp. 11–18 (2010)
9. Harvey, J.P.: Gpu accelaration of object classification algorithms using nvidia cuda. Master's thesis, Department of Computer Engineering, Kate Gleason College of Engineering, Rochester Institute of Technology (2009)
10. Lazarevic, A., Obradovic, Z.: Boosting algorithms for parallel and distributed learning. Distributed and Parallel Databases (11), 203–229 (2002)
11. Merler, S., Caprile, B., Furlanello, C.: Parallelizing adaboost by weights dynamics. Computational Statistics & Data Analysis 51, 2487–2498 (2007)
12. Galtier, V., Pietquin, O., Vialle, S.: Adaboost parallelization on PC clusters with virtual shared memory for fast feature selection. In: Signal Processing and Communications, pp. 165–168 (2007)
13. Huang, Z., Shi, X.: A distributed parallel adaboost algorithm for face detection. In: Intelligent Computing and Intelligent Systems, pp. 147–150 (2010)
14. Zeng, K., Tang, Y., Liu, F.: Parallization of adaboost algorithm through hybrid mpi/openmp and transactional memory. In: International Eumicro Conference on Parallel, Distributed and Network-Based Processing, pp. 94–100 (2011)
15. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Elsevier Inc. (2005)
16. Shi, Y.: Reevaluating amdahls law and gustfsons law. Technical report, Temple University (1996)
17. Akl, S.G.: Superlinear performance in real-time parallel computation. The Journal of Supercomputing 29(1), 89–111 (2004)
18. Hill, M.D., Marty, M.R.: Amdahls law in the multicore era. Computer 41(7), 33–38 (2008)