GPU-Accelerated SVM Training Algorithm Based on PC and Mobile Device

Yi-Yan Nan, Quan-Zhe Li, Jin-Chun Piao, and Shin-Dug Kim

***Abstract*—This work is to design an accelerated SVM (Support Vector Machine) which is suitable for Android operating system. SVM is widely used in the health-related applications. The SVM provides a potential classification technology based on the pattern recognition method and statistical learning theory. This paper proposes a parallel SVM algorithm based on GPU accelerator. GPU can provide better performance on matrix multiplication through parallelization which is the main drawback of conventional SVM execution. The cross validation function in the personal computer is designed and improved, and SVM training function in the mobile devices in addition. Through the above approach, the influence of matrix calculation on the whole system can be reduced to a certain extent. In the experiment of image classification, compared to the serial SVM, the proposed approach can achieve 3.3x speed up in the PC, and 1.5x speed up in the mobile devices. But the accuracy rate is not greatly improved both. Since the experiment mainly focuses on improving the execution time, no optimization is considered on the prediction process.**

***Index Terms*—Support vector machine algorithm, parallel computing, GPU and OpenCL based SVM, image classification, matrix multiplication.**

I. Introduction

Recently, more and more wearable manufacturers focus on the health-related products, and provide their supporting software clients and applications. At present, the wearable devices are used as the attachment of mobile phones. Thus, data rendering and processing need to rely on mobile smart phones. In general, Android phones offer a variety of sensors, such as direction, gravity, distance, acceleration and so on. With these sensor data, it is possible to provide health care services for users.

SVM (Support Vector Machine) is a potential classification technology based on the pattern recognition method and statistical learning theory. In the field of health care applications, SVM algorithm is widely used to analyze human behavior. Through the analysis and modeling of the training data, the unknown data can be predicted. Such a process includes target detection, feature extraction, modeling, prediction and other processes. But the SVM consumes a lot of memory space, because of a large-scaled data. Usually, it takes a lot of time to train various test data. Thus, it can be a main problem when SVM is applied to the mobile devices.

There are many studies related to the optimization of SVM algorithm in OpenCL (Open Computing Language) framework. The characteristics of OpenCL, such as shared virtual memory, dynamic parallelism and general memory space, greatly improve programming flexibility to avoid redundant data transfer. Sparse linear algebra, causing a huge computational load, is the main field of SVM, because the SVM solves the support vector by means of quadratic programming.

In this paper, we utilize the GPU accelerator to improve the performance with the proposed optimization method. An accelerated SVM is suggested with the modification of original LIBSVM (A Library for Support Vector Machines). The paper implements parallelization of the raw dataset which is passed to the cross-validation function in order to reduce computational complexity. Furthermore, the proposed optimization method is applied to the RBF (Radial basis function) kernel function in the mobile device.

In the experiment, the CIFAR-10 dataset is used to implement image classification. The performance of the accelerated SVM is evaluated both in the PC and mobile devices. The proposed parallel approach becomes 3.3 times faster than the serial computing in the PC, and 1.5 times faster in the mobile device. As a result, the total image classification time has been improved significantly without reducing the accuracy rate.

This paper covers the related work in Section II. The proposed new approach is described in Section III. Experiment and results are followed in Section IV. Finally, Section V concludes this paper.

II. Related Work

Many researches have been proposed for accelerating SVM's computational speed since it performs expensive computation during training big-scale dataset. In this section, several approaches are introduced for speeding up performance of SVM′s computation.

Cagnini *et al.* [1] presented a technique that parallelized SVM method within a GPU together with OpenCL framework in order to improve efficiency of binary classification tasks and SVM computations. The authors first identified the most computationally expensive functions and then parallelized these functions. [2] The proposed approach achieved a significant speedup compared to sequential and CUDA-based approach.

Manuscript received November 5, 2016; revised December 30, 2016. This work was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the Global IT Talent support program (IITP-2016-H0905-15-1003) supervised by the IITP (Institute for Information and Communication Technology Promotion).

The authors are with the Yonsei University, Republic of Korea (e-mail: nanyiyan@yonsei.ac.kr, liquanzhe@yonsei.ac.kr, kumcun@yonsei.ac.kr, sdkim@yonsei.ac.kr).

doi: 10.18178/ijke.2016.2.4.076

Yan *et al*. [3] proposed a new accelerated GPU based support vector machine method handling kernel matrix calculation and cross validation by using GPU to speed up SVM processing. One-against-one, an algorithm that takes long time in large scale problems was chosen in this paper to parallelize. Ref. [4] the proposed method could achieve up to 41 times faster compared to LIBSVM according to this experiments.

Salleh *et al*. [5], [6] discussed on applying performance analysis during SVM training session which included SMP approach and vector processor approach. This research also benchmarked between CPU and GPU optimization. The experiment result showed that approach through vector processor achieves up to 3.11 times better performance comparing to LIBSVM CPU optimized program.

III. The Proposed Approach

A. GPU Optimized Support Vector Machine

In machine learning, the SVM is a supervised learning model [7]. The SVM is commonly used for pattern recognition, classification, and regression analysis. Main procedure of the SVM can be summarized as two points:

1) SVM generally analyzes linearly separable cases. For linear indivisibility cases, samples of low dimensional input space are transformed into high-dimensional feature spaces by using a non-linear mapping algorithm, so that it is possible to analyze the non-linear characteristics of the sample using a linear algorithm in the high-dimensional feature space.
2) Based on the structural risk minimization theory, the optimal partitioning hyperplane is constructed in the feature space to make the global optimization of the learner, and the expected risk in the whole sample space satisfies a certain upper bound with some probability.

Based on the above points, SVM algorithm is difficult to implement for large-scaled training samples. Since SVM solves the support vector by means of quadratic programming which will involve the calculation of *m*-order matrices, where *m* is the number of samples. When the number of *m* is large, the storage and computation of the matrix will consume a large amount of memory space and computing time. So in this paper, we utilize the OpenCL framework to speed up the training phase of the SVM algorithm.

OpenCL is the first open, free standard for parallel programming of heterogeneous systems for general purpose [8]. It is also a unified programming environment for software developers to write any efficient and lightweight code for high-performance computing servers. In this paper, the OpenCL2.0 framework is employed, which enhances the characteristics of shared virtual memory. As shown in Fig.1, for the shared virtual memory buffer, the host and the device can be directly accessed, and it will no longer have to worry about heterogeneous platform for data access. We choose AMD GPU which supports coarse-grained shared virtual memory.

Sharing occurs at the granularity of regions of OpenCL buffer memory objects. We force memory consistency at the synchronization point, and use the map / unmap command to update the data between the host and the device. It does not need to copy the data back and forth, the device and the host can directly access each other's data. It is useful to reduce the computational complexity of matrix operations.

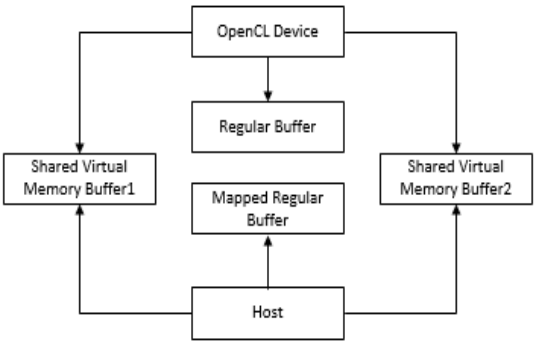


Fig. 1. Address space between host and device.

B. Optimization and Proposed Approach

In this paper, we propose a GPU-accelerated SVM with modification of the original LIBSVM. LIBSVM is an integrated software package that provides support vector classification, regression, and distribution estimation. The approach speeds up the training time through improving cross-validation process of SVM.

Cross-validation is a statistical analysis method which is used to verify the performance of SVM classifier. The basic idea is to group the raw data in a sense, some as training set, and the other as validation set. Firstly, the training set is used to train the classifier, and the validation set is used to test the model to evaluate the performance of the classifier. There are three common cross-validation methods. LIBSVM employs one-against-one approach, which is to design an SVM between any two classes of samples, so *k* samples need to design *k* (*k*-1)/2 SVMs. When an unknown sample is categorized, the category with the most votes is the category of the unknown sample.

We design a parallel version of cross-validation function as shown in Fig. 2. If the process does *n*-fold cross-validation, svm_cross_validation function will create an index array perm, then the array perm can access the prob which represents all dataset and their classes to disrupt the prob effect. Next, the prob is divided into *n* copies, *n*-1 for training, one for test.

After *n* times, the target saves the test results of all elements of prob. And the sequence of storage is the same as dataset arrangement of prob. Finally we compare target and prob.y which represents the classes of dataset, in order to obtain the cross-validation recognition accuracy. We estimate generalization error for *n* models, and choose optimal model to predict.

In the above process, the parameter prob is involved in a large number of sparse matrix multiplication operations. The memory consumption of the matrix is large, which is unfavorable for practical applications.

So we call cross_validation_with_KM_percalculated function that proposed before svm_cross_validation. In this function, OpenCL framework is employed which is introduced before. In the original LIBSVM version, prob is passed to the function svm_cross_validation as an argument.

Prob carries lots of raw data without any optimization, then the matrix multiplication operation is performed with these raw data.

```
Parallel version of cross-validation algorithm

Input : All dataset and their classes as parameter Prob,
        Classifier training parameters,
        The number of cross validation as parameter n
Output : Target

Function cross_validation_with_KM_precalculated
    Initialize svm_problem pecm;
    Procedure _kernel void matrix_product
        Calculate multiplication of relation matrix;
    End
    Return Prob_library;
End

Function svm_cross-validation
    Initialize array perm;
    Prob_library divided into n, n-1 for training,
    1 for test;
    For i=1 to n Do
        Train all data except i-th;
        Get training parameters;
    Endfor
End

Estimated generalization error for n model;
Choose optimal model;
```

Fig. 2. Parallel version of cross-validation.

The cross_validation_with_KM_percalculated function is designed to optimize this process and reduce the amount of training time.

The specific implementation method is shown in Fig. 3. For each row of prob matrix, we copy data into buffer space, then calculate the inverse matrix of pecm. The result will be stored in the shared virtual buffer memory. Next, prob and pecm perform multiplication parallelization. Finally, Prob_library which is the result of matrix multiplication will be stored in the shared virtual buffer memory.

```
For each row of prob
      Copy to buffer space;
      Calculate pecm^T ;
      Stored in shared virtual buffer memory;

      Calculate prob_i * pecm^T ;
      Stored in shared virtual buffer  memory;
Endfor
Return Prob_library;
```

Fig. 3. Cross_validation_with_KM_percalculated function.

IV. Experiment and Result

A. Experimental Setup

In this paper, we implement two experiments to evaluate the proposed approach of modification. First experiment is based on the PC. The configuration of PC is shown in Table I.

The PC device we used in the first experiment is Samsung laptop, which is composed with 7.7GB main memory, Intel Core TM i5-3230M 2.60GHz CPU, AMD Radeon R9 M200X Series GPU. And the OpenCL platform is from AMD Accelerated Parallel Processing OpenCL 2.0 AMD_APP (1800.11). The experiment is implemented on 64bit Ubuntu 14.04LTS operating system.

Second experiment is based on the mobile device. The configuration of mobile is shown in Table II.

The mobile device we used in the second experiment is XiaoMi MI2 series, which is composed of 2GB main memory, four-cores 1.5GHz QUALCOMM Krait CPU, QUALCOMM Adreno GPU. And the OpenCL platform is QUALCOMM Adreno. The experiment is implemented on Android operating system.

The difference between these two experiments is the type of GPU. So we need to modify the approach in a way. Compared to the PC experiment, we optimize the function which is called svm_train in the original LIBSVM, rather than svm_cross_validation. In the experiment, we choose BF kernel function which is mainly used for the linear indivisibility cases. This function has many parameters. The result of classification is dependent on these parameters. So it is a time-consuming process. The proposed optimization of the PC is applied to the different function in the mobile device. The parallel computing also can be applied to the implementation of feature matrix multiplication in the RBF kernel function.

The dataset that we use in two experiments is CIFAR-10. The CIFAR-10 dataset consists of 60,000 32*32 color images distributed in 10 classes, each with 6000 images. These 60,000 images include 50,000 training images and 10,000 test images, respectively [9].

TABLE I: The Configuration of PC

Samsung laptop	64bit
Operating system	Ubuntu 14.04LTS
CPU	Intel Core TM i5-3230M 2.60GHz
Main memory	7.7GB
GPU	AMD Radeon R9 M200X Series
	AMD Accelerated Parallel Processing
OpenCL platform	OpenCL 2.0 AMD_APP (1800.11)

TABLE II: The Configuration of Mobile Device

XiaoMi	MI2
Operating system	Android4.11
CPU	four-cores 1.5GHz QUALCOMM Krait
Main memory	2GB
GPU	QUALCOMM Adreno
OpenCL platform	1.1 Adreno(TM) 320

Note that, the type of data loaded from CIFAR-10 is integer, it needs to convert into the image type. Each file has a 10000*3027N array, each row of the array stores a 32*32 color image. The first 1024 entries represent red, the next 1024 represents green, and the last 1024 represents blue. The image is stored in a row-major order, so that the first 32 entries of the array are the red channel values of the first row of the image.

Based on such environment and optimization method, we implement the image classification to verify the performance of the proposed algorithm.

First of all, we extract image features. GIST feature extraction is a computational model that identifies the real world. The model bypasses the segmentation and processing

of individual objects or regions [10]. A five-dimensional perception dimension is used to represent the main elements of a scene, including nature, openness, roughness, extensibility, and robustness. These dimensions can reliably estimate the used spectral and coarse location information, although these dimensions can be used to represent a scene picture.

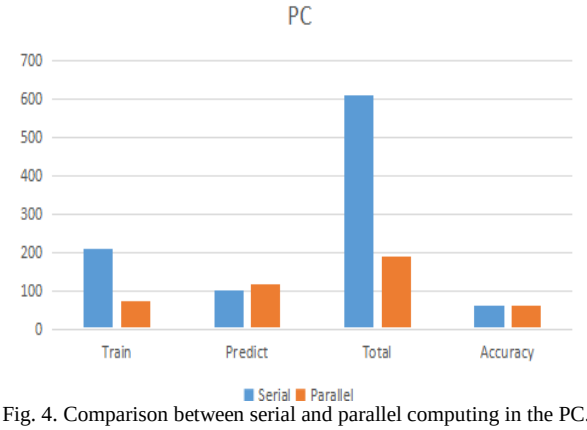


Fig. 4. Comparison between serial and parallel computing in the PC.

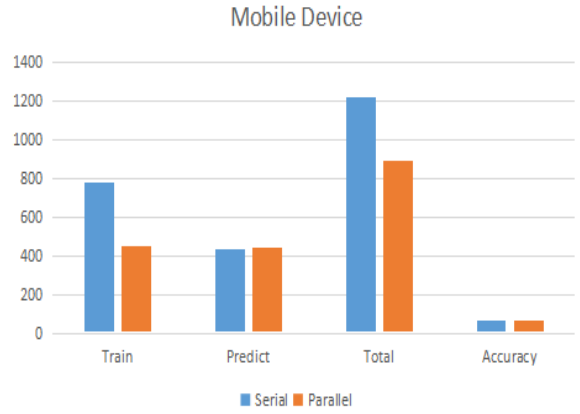


Fig. 5. Comparison between serial and parallel computing in the mobile device.

Here we use the GIST feature extraction, download the gistdescriptor package that already has a Gist function, and it can call it as follows : [Gist1, param] = LMgist (Image, '', param); We import each picture on the corresponding feature extraction, with a 1:50000 cycle. So the image data are converted into the format of training and testing. The format is defined as follows : <label> <index> : <value>.

B. *Result*

Fig. 4 shows the result of the first experiment, the comparison between serial and parallel computing in the PC. The serial computing represents the original LIBSVM without any modification. Because we only optimize the training process, only the training time of classification is affected. The training time of serial and parallel SVM is 208.74s, 72.66s respectively. As the Fig.4 shows, the proposed approach, the parallel computing, it gains 3.3x speed up in the total execution. The total time of serial and parallel SVM is 610.55s, 189.82s respectively. The classification accuracy has not been greatly improved.

Fig. 5 shows the result of the second experiment, the comparison between serial and parallel computing in the mobile device. The parallel computing becomes 1.5 times faster than the serial computing. The training time of serial and parallel SVM is 782s, 452s respectively. And it gains 1.5x speed up in the total execution. The classification accuracy also has not been greatly improved.

As a result, the proposed parallel version SVM shows the better performance in both PC and mobile device. Although we just optimize the training process, the total image classification time has been improved without reducing the accuracy rate.

V. Conclusion

In this paper, we optimize the SVM performance with the combination of the GPU device. We suggest an accelerated SVM with modification of the original LIBSVM.

We implement parallelization of the cross-validation function in the PC. In addition, the proposed optimization method is applied to the RBF kernel function in the mobile devices.

In the experiment, we use CIFAR-10 image dataset to evaluate the proposed SVM classifier both in the PC and mobile devices. The proposed approach, the parallel computing becomes 3.3 times faster than the serial computing in the PC, and 1.5 times faster in the mobile device. In general, the total image classification time has been greatly improved without reducing the accuracy rate.

For future work, we will focus on improving the predict process, since the accuracy rate and predict time are not greatly improved.

References

[1] S. N. Shakirah and M. F. Baharim, "Performance comparison of parallel execution using GPU and CPU in SVM training session," presented at 2015 4th International Conference on Advanced Computer Science Applications and Technologies (ACSAT), IEEE, 2015.

[2] C. Hsu, C. Chang, and C. Lin, "A practical guide to support vector classification," Department of Computer Science, National Taiwan University, Taipei, 2010, pp. 1-16.

[3] C. Henry *et al*., "A portable OpenCL-based approach for SVMs in GPU," presented at 2015 Brazilian Conference on Intelligent Systems (BRACIS), IEEE, 2015.

[4] Pc gpu market bounces back, with nvidia up and amd down. (2014). [Online]. Available: http://www.forbes.com/sites/jasonevangelho/2014/02/19/pc-gpu-market-bounces-back-with-nvidia-up-and-amd-down/

[5] Y. Bo *et al*., "A GPU based SVM method with accelerated kernel matrix calculation," presented at 2015 IEEE International Congress on Big Data, IEEE, 2015.

[6] Q. Liao, J. Wang, Y. Webster, and I. Watson, "GPU accelerated support vector machines for mining high-throughput screening data," *Journal of chemical information and modeling*, vol. 49, no. 12, pp. 2718-2725, 2009.

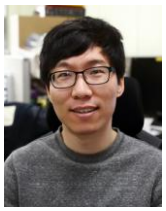
[7] WR. Fan. LIBSVM data: Classification, regression, and multi-label. [Online]. Available: https://www.csie.ntu.edu.tw/

[8] OpenCLTM 2.0 - Shared Virtual Memory. [Online]. Available: http://developer.amd.com/community/blog/2014/10/24/opencl-2-shared-virtual-memory/

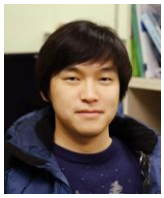
[9] E. F. Carvalho and P. M. Engel, "Convolutional sparse feature descriptor for object recognition in CIFAR-10," presented at 2013 Brazilian Conference on the Intelligent Systems (BRACIS).

[10] G. W. Juette and L. E. Zeffanella, "Indoor/outdoor image classification using GIST image features and neural network classifiers," presented at 2015 12th International Conference on the High-Capacity Optical Networks and Enabling/Emerging Technologies (HONET), , 21-23 Dec. 2015.

Yi-Yan Nan received the B.S. degree in software engineering from Jilin University, Changchun, China, in 2014. She is currently a M.S. student in Department of Computer Science at Yonsei University, Korea. Her research interest is activity recognition.



Quan-Zhe Li received the B.S. degree in computer science from Anhui University, Anhui, China, in 2009. He is currently a M.S. student in Department of Computer Science at Yonsei University, Korea. His research interests include activity recognition and parallel computing.



Jin-Chun Piao received the B.S. degree in computer science and technology from Beijing Forestry University, Beijing, China, in 2010. He is currently a combined M.S. and Ph.D. student in Department of Computer Science at Yonsei University, Korea. His research interests include AR based ubiquitous computing, mobile computing and graphics.

Korea Shin-Dug Kim received his B.S. degree in electronic engineering from Yonsei University, Seoul, Korea in 1982, and M.S. degree in electrical & computer engineering from University of Oklahoma, Norman, in 1987. In 1991, he received his Ph.D. degree from the School of Electrical & Computer Engineering at Purdue University, West Lafayette. He is currently a professor of computer science at Yonsei University. His research interests include advanced computer systems, intelligent memory system design, and ubiquitous computing platforms. He is a member of IEEE.