



**Computer Systems Department
Faculty of Computer & Information Sciences
Ain Shams University**

SPEEDING UP LARGE SCALE MACHINE LEARNING ALGORITHMS USING GPGPU

This thesis is submitted as a partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences.

BY

Islam Ahmed Hamed Elgarhy

B.Sc. in Computer and information Sciences,
Demonstrator at Computer Systems Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

Under Supervision of

Prof. Dr. Hossam El-Deen Mostafa Fahim

Professor in Computer Systems Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

Prof. Dr. Rania Abd El-Rahman El Gohary

Professor in Information Systems Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

Dr. Heba Ahmed Khaled

Assistant Professor in Computer Systems Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

Cairo 2019

قسم نظم الحاسبات
كلية الحاسبات و المعلومات
جامعة عين شمس



تسريع أداء خوارزميات تعلم الآلة باستخدام كروت الشاشة عالية الأداء متعددة الاستخدام

رسالة مقدمة لإستيفاء الحصول على درجة الماجستير فى الحاسبات و المعلومات

إعداد

إسلام احمد حامد الجارحي
بكالوريوس الحاسبات و المعلومات،
المعيد بقسم نظم الحاسبات،
كلية الحاسبات و المعلومات،
جامعة عين شمس.

إشراف

ا.د/ حسام الدين مصطفى فهمي
أستاذ بقسم نظم الحاسبات،
كلية الحاسبات و المعلومات،
جامعة عين شمس.

ا.د/ رانية عبد الرحمن الجوهري
أستاذ بقسم نظم المعلومات،
كلية الحاسبات و المعلومات،
جامعة عين شمس.

د/ هبة خالد احمد
مدرس بقسم نظم الحاسبات،
كلية الحاسبات و المعلومات،
جامعة عين شمس.

القاهرة 2019

*This thesis is dedicated to my parents and my brothers for
their love, continuous support and encouragement. Many
Thanks to my wife. Without you, I couldn't complete my work.
To my daughters, I'm fond of you.*

ACKNOWLEDGMENT

All gratitude is to ALLAH for the strengths and his blessing in completing this thesis. Profound appreciation goes to *prof. Dr. Hossam Faheem* for his supervision and continuous support and invaluable help by constructive comments and suggestions during the study and fulfillment of this thesis.

I would like to acknowledge and express my appreciation to *prof. Dr. Rania El Gohary* for her support, suggestions and kind instructions. No words to express my sincere thanks and gratitude to *Dr. Heba Khaled* for her guidance, supports at all phases of this research work. My appreciation goes to my professors, staff members of FCIS Ain Shams University.

ABSTRACT

As many of the machine learning algorithms, SVM requires a high computational cost (memory and time) to solve a complex quadratic programming (QP) optimization problem, so SVM necessitate a high computing hardware capabilities.

Due to the physical limitation in miniaturization process, the central processing unit (CPU) clock frequency can't be increased, therefore the huge improvements done in packaging multiple CPU cores onto the same silicon chip and also using graphical processing unit (GPU) for general purpose numerical computing.

With the advantages of parallel multi-architecture in both multi-core CPU and a high-scalable GPU, there is a promising candidate to enhance the performance of the algorithms that fits well to run in parallel multi-architecture, so there is a chance to enhance the SVM high computational time for solving the optimization problem.

Moreover, Tensorflow is an open-source machine learning framework library that allows to implement machine learning algorithms using Application program interfaces (APIs). Tensorflow has the ability to migrate to alternative hardware components, and it will reduce time for developing alternative algorithms, so there is a chance to use Tensorflow library for implementing a cross-platform SVM implementation with short development time.

This thesis, presents the design and implementation of a hybrid parallel implementation for SVM algorithm, also it show a comparative study between this hybrid parallel implementation and tensorflow implementation.

The benchmark shows a significant improvements in speed up for hybrid parallel implementation over sequential implementation, other parallel implementation and tensorflow implementation.

The proposed hybrid parallel implementation achieves a speed up of 40X over the sequential open-source library (LIBSVM), a speed up of 7.5X over the CUDA-OPENMP for training process with (44442 records, 102 features size, and 9 classes), a speed up of 13.7X over LIBSVM in classification process for 60300 records, and a speed up of 14.9X over the SVM Tensorflow implementation on pavia centre dataset.

CUDA-GPU achieves a speed up of (154.3X, 60.5X, and 119.7X) over Tensorflow-GPU for three different training datasets (pavia centre hyperspectral, breast cancer, and iris folower) respectively. Also, Experimental results show that the explicit control in CUDA API have a speed up over the implicit control in Tensorflow. However, Tensorflow is a cross-platform implementation where it can be migrated to alternative hardware components, which will reduces the development time.

TABLE OF CONTENTS

Acknowledgment.....	II
Abstract.....	III
Table of Contents.....	V
List of Figures.....	VII
List of Tables.....	X
List of Abbreviations.....	XI
List of Publications.....	XII

Chapter 1: Introduction

1.1 Overview.....	1
1.2 Motivation.....	4
1.3 Objectives.....	5
1.4 Contributions.....	5
1.5 Thesis Outlines.....	6
1.6 Summary.....	6

Chapter 2: Parallel Architecture Background And Related Work

2.1 Introduction.....	7
2.2 Parallel Architecture Models.....	7
2.2.1 Flynn’s Classifications.....	8
2.2.1.1 Single Instruction Stream Single Data Steam (SISD).....	8
2.2.1.2 Single Instruction Stream Multiple Data Steam (SIMD).....	9
2.2.1.3 Multiple Instruction Stream Single Data Steam (MISD).....	9
2.2.1.4 Multiple Instruction Stream Multiple Data Steam (MIMD).....	10
2.2.2 Memory Arrangement and Communication Type Classification	11
2.2.2.1 Shared Memory (Tightly Coupled) Architecture.....	11
2.2.2.2 Distributed Memory (Loosely	

Coupled) Architecture	12
2.2.3 Parallel Architecture Paradigms.....	12
2.2.3.1 Compute Unified Device Architecture (CUDA).....	14
2.2.3.2 Message Passing Interface (MPI).....	22
2.2.3.3 Hybrid Parallel Memory Architecture.....	24
2.3 Related Work	25
2.4 Summary.....	28

Chapter 3: Proposed Parallel Implementations for Solving SVM Optimization Problem

3.1 Introduction.....	29
3.2 Machine Learning Basics.....	29
3.2.1 Overview.....	29
3.2.2 Support Vector Machine (SVM).....	31
3.2.3 Sequential Minimum Optimization (SMO)...	35
3.3 Support Vector Machine using MPI-CUDA	38
3.3.1 Parallel Binary-Class SMO Implementation.....	38
3.3.2 Parallel Multi-Class SMO Implementation.....	41
3.3.3 Parallel SVM Classification	42
3.4 Support Vector Machine using Tensorflow	43
3.5 Summary.....	46

Chapter 4: Experimental Results and Discussion

4.1 Introduction.....	47
4.2 Evaluation Metrics	47
4.3 Datasets.....	48
4.3.1 Pavia Centre Hyperspectral Dataset	49
4.3.2 Iris Flower Dataset.....	50
4.3.3 Breast Cancer Dataset	50
4.4 Results	51
4.4.1 Results Using MPI-CUDA.....	51
4.4.1.1 Hardware and Software	

Specifications.....	51
4.4.1.2 Experimental Results and Discuss...	51
4.4.2 Results Using TensorFlow.....	55
4.4.2.1 Hardware and Software	
Specifications.....	55
4.4.2.2 Experimental Results and Discuss...	55
4.5 Summary.....	59
 Chapter 5: Conclusion and Future Work	
5.1 Conclusion.....	60
5.2 Future Work.....	62
 References.....	63

LIST OF FIGURES

Figure	Name	Page
1.1	The Representation of a Machine Learning Algorithm	2
2.1	Single Instruction Stream Single Data Steam (SISD)	9
2.2	Single Instruction Stream Multiple Data Steam (SIMD)	9
2.3	Multiple Instruction Stream Single Data Steam (MISD)	10
2.4	Multiple Instruction Stream Multiple Data Steam (MIMD)	10
2.5	Shared Memory (Tightly Coupled) Architecture	11
2.6	Distributed Memory (Loosely Coupled) Architecture	12
2.7	Shared and Distributed Memory architectures programming models	13
2.8	GPU performance over years	14
2.9	CPU Architecture vs GPU Architecture	16
2.10	GPU Architecture and GPU memory types	18
2.11	GPU Grid-Block-Thread Dimensions	19
2.12	The execution flow for a program between CPU and GPU	20
2.13	CPU vs GPU for incrementing array elements by scalar variable	21
2.14	MPI Communicator Architecture	23
2.15	Hybrid Memory Architecture	25
3.1	Binary Class Boundary line	30
3.2	SVM Maximum-Margin Classifier	32

3.3	SMO Algorithm two main steps	36
3.4	SMO Algorithm	37
3.5	Parallel Binary-Class SMO Algorithm	41
3.6	Parallel Multi-Class SMO Algorithm	42
3.7	Tensorflow Graph	44
3.8	Gradient Descent Optimizer Algorithm from Tensorboard	45
4.1-a	Ground truth of Pavia city center dataset	49
4.1-b	Sample band of Pavia city center dataset	49
4.2-a	Iris Flower setosa	50
4.2-b	Iris Flower versicolor	50
4.2-c	Iris Flower virginica	50
4.3	Training Time MPICUDA vs. LIBSVM	52
4.4	Training Time MPICUDA vs. OPENMPCUDA	53
4.5	Binary training time CUDA vs. Tensorflow	56
4.6	Multi-Class training time MPI-CUDA vs. Multi-Tensorflow	57

LIST OF TABLES

Table	Name	Page
4.1	Datasets	48
4.2	Training Time (MPI-CUDA VS. LIBSVM)	51
4.3	Hardware Specs (MPI-CUDA VS. OPENMP-CUDA)	52
4.4	Training Time (MPI-CUDA VS. OPENMP-CUDA)	53
4.5	Classification Time (MPI-CUDA VS. LIBSVM)	54
4.6	Machine Hardware Specs (CPU-GPU)	55
4.7	Training Time (CUDA-GPU VS. Tensorflow-GPU)	55
4.8	Training Time (Tensorflow-CPU VS. Tensorflow-GPU) for pavia centre dataset	56
4.9	Training Time (Tensorflow-CPU VS. Tensorflow-GPU) for Iris Folower and Breast Cancer dataset	57
4.10	Training Time (Tensorflow-CPU VS. Tensorflow-GPU)	58

LIST OF Abbreviations

ALU	A rithmetic L ogic U nit
API	A pplication P rogram I nterface
CPU	C entral P rocessing U nit
CU	C ontrol U nit
CUDA	C ompute U nited D evice A rchitecture
GDDR	G raphic D ouble D ata R ate
GPGPU	G eneral P urpose G raphics P rocessing U nit
GPU	G raphics P rocessing U nit
IC	I ntegrated C ircuit
KKT	K arush K uhn T ucker
MIMD	M ultiple I nstruction M ultiple D ata
MISD	M ultiple I nstruction S ingle D ata
ML	M achine L earning
MPI	M essage P assing I nterface
OpenCL	O pen C omputing L anguage
OpenMP	O pen M ulti P rocessing
QP	Q uadratic P rogramming
RAM	R andom A ccess M emory
SIMD	S ingle I nstruction M ultiple D ata
SISD	S ingle I nstruction S ingle D ata
SMO	S equential M inimal O ptimization
SMs	S treaming M ultiprocessors
SPs	S treaming P rocessors
SVM	S upport V ector M achine
UPC	U nified P arallel C