HARDWARE IMPLEMENTATION OF NAIVE BAYES CLASSIFIER FOR MALWARE DETECTION

YAHYA KHALED AL HUSSEIN

DR. ISMAHANI BINTI ISMAIL
Pensyarah
Jabatan Kejuruteraan Elektronik & Kejuruteraan Komputer
Fakulti Kejuruteraan Elektrik
Universiti Teknologi Malaysia
81310 UTM Johor Bahru
Johor Darul Takzim

UNIVERSITI TEKNOLOGI MALAYSIA

HARDWARE IMPLEMENTATION OF NAIVE BAYES CLASSIFIER FOR MALWARE DETECTION

YAHYA KHALED AL HUSSEIN

A project report submitted in partial fulfilment of the
requirements for the award of the degree of
Master of Engineering (Computer and Microelectronic Systems)

School of Electrical Engineering
Faculty of Engineering
Universiti Teknologi Malaysia

FEBRUARY 2021

DEDICATION

This thesis is dedicated to my father. I would much prefer it if he were still alive and well. It is also dedicated to my dear mother, and that for her continuous encouragement, endless care, and unconditional support.

ACKNOWLEDGEMENT

In preparing this thesis, I was in direct contact with many people, academicians, researchers, and classmates. They all have contributed towards my basic understanding and toward the progress that I made. In particular, I want to express my sincere appreciation to my thesis supervisor Dr. Ismahani Binti Ismail, for her comments and recommendations on this thesis. I also want to express my thankful to the School of Electric Engineering in UTM and to all its member's staff for the guidance that they gave to me, and above all to my dear family and friends for supporting me all the time during my study.

ABSTRACT

Naïve bayes classifier is a probabilistic supervised machine learning algorithm, that can be launched on most general-purpose devices to solve wide range of classification problems. However, when it comes to real time applications, the general-purpose devices are limited in term of their computational throughput, thus this algorithm couldn't be used for that purpose. The aim of this project is to accelerate this algorithm in hardware environment to improve its performance by exploring its hidden concurrency and map it into parallel hardware as an optimized IP package, suitable for FPGA-SoC applications. Thus, it could be used as a middle box system for real time malware detection. In order for the proposed hardware to meet the requirements of this research, it should be able to handle both training, and inference part in hardware, and also should be able to receive a flow of 20 features, each of 32-bitsize, organized in 4-gram format. To meet these requirements, an enhanced version of the algorithm was developed and tested in C-programming. Then an equivalent design with a 5-stages pipelined architecture, and single instruction multiple data capabilities, was built in hardware to address the case. At the end, the proposed hardware found to be 65 times faster in term of its computational throughput compared to an existing design, and that with keeping the accuracy level as high as 94%, under the conditions of experiment carried.

ABSTRAK

Pengelas Naïve bayes adalah algoritma pembelajaran mesin berkebarangkalian yang boleh dilancarkan pada kebanyakan peranti serbaguna untuk menyelesaikan pelbagai masalah pengelasan. Walau bagaimanapun, untuk aplikasi waktu nyata, peranti serbaguna ini terhad dari segi hasil pengiraannya, oleh itu algoritma ini tidak dapat digunakan untuk tujuan tersebut. Tujuan projek ini adalah untuk mempercepatkan algoritma ini dalam keadaan perkakasan untuk meningkatkan prestasinya dengan meneroka keserentakan yang tersembunyi dan memetakannya kepada perkakasan selari sebagai IP pakej teroptimum, sesuai untuk aplikasi-aplikasi FPGA-SoC. Oleh itu, ia dapat digunakan sebagai sistem kotak tengah untuk pengesanan pengisian hasad masa nyata. Agar perkakasan yang dicadangkan memenuhi keperluan penyelidikan ini, ia seharusnya dapat menangani bahagian latihan dan inferens dalam perkakasan dan juga berupaya menerima aliran 20 ciri objek, masing-masing 32-bit, disusun dalam format 4-gram. Untuk memenuhi keperluan ini, versi algoritma tersebut ditambah baik dan diuji menggunakan pengaturcaraan C. Kemudian reka bentuk yang setara dengan seni bina talian paip 5-tahap, dan satu arahan pelbagai keupayaan data, dibina dalam perkakasan untuk menangani kes tersebut. Pada akhirnya, perkakasan yang dicadangkan menghasilkan 65 kali lebih cepat dari segi hasil pengiraannya berbanding dengan reka bentuk sedia ada dengan mengekalkan tahap ketepatan setinggi 94% di bawah keadaan eksperimen yang telah dijalankan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

FPGA	-	Field Programmable Gate Array
SoC	-	System on Chip
ASIC	-	Application Specific Integrated Circuit
IC	-	Integrated Circuit
APSoC	-	All-Programmable System on Chip
PL	-	Programmable Logic
PS	-	Processing System
AXI	-	Advance eXtensible Interface
HW/SW	-	Hardware/Software
CLB	-	Configurable Logic Block
LUT	-	Lookup Table
FFs	-	Flip-Flops
ISA	-	Instruction Set Architecture
HDL	-	Hardware Description Language
RTL	-	Register Transfer Level
DU	-	Data-path Unite
CU	-	Control Unite
FSM	-	Finite State Machine
FSMD	-	Finite State Machine Datapath
ALU	-	Arithmetic Logic Unit
SIMD	-	Single Instruction Multiple Data
PE	-	Processing Element
IP	-	Intellectual Property
OS	-	Central Processing Unite
CPU	-	Central Processing Unite
GPP	-	General Purpose Processor
GPU	-	Graphics Processing Unit

HDR	-	High Dynamic Range
DSP	-	Digital Signal Processing
ALA	-	Adaptable Logarithm Approximation
NBC	-	Naïve Bayes Classifier
bSOM	-	Binary Self Organization Map
CNN	-	Convolutional Neural Network
AI	-	Artificial Intelligent
ML	-	Machine Learning
NB	-	Naïve Bayes
TF	-	Term Frequency
IDF	-	Inverse Document Frequency
IOT	-	Internet of Things
TTM	-	Time to Market
FBI	-	Federal Bureau of Investigation

CHAPTER 1

INTRODUCTION

1.1 Background

By the current time, machine learning (ML) is one of the most important aspects that characterized the modern development in technology. It almost intercepts with every domain of our modern life [9]. From business to engineering, machine learning assists computers with well written algorithms to progressively improving its performance. The powerful of machine learning algorithms lays behind its ability to let the computers learn from there past experience without being explicitly programmed. This unique ability of ML to learn is developed by building mathematical model feed with sample data known as training data [10]. Image processing, speech recognition, product recommendation, medical diagnosis, traffic predication, spam mail detection, and malware filtering are some of the most current trending applications of machine leaning [11]. However, this noticeable leap in machine learning development was motivated by the expansion of cyberspace that provide these algorithms with the sample data that they need for training.

During the past years of our recent history, there were at least 12 definitions attempted to describe what we define as cyberspace [12]. This reflects who dramatically the cyberspace growth in the past decades. But unfortunately, this cyberspace is felt with harmful software that can harm the connected users as well. In general, malicious software, or malwares, has the ability to play very critical role in any intrusion attack that attempts to hack or harm the connected devices. Generally, we can consider any software that harms the network users as a malware. This can include all sorts of trojan horses, worms, viruses, rootkits, spyware and scareware [13]. With tons of malware roaming all the way around our cyberspace, installing malware detection system is critical for any network user. Based on internet crime complaint center (IC3), that created by the FBI recently in 2000 [14], up to 4,883,231 complaints reported have been received by the center since its creation. From those reported, up to 49,711 reports

have been received in its first year of operation. Furthers details about the numbers of complaints and the amount of money lost by such incidents are presented in Figure 1.1 [1]. These huge numbers that recorded by just one center around the world, are big enough to make one aware from being connected to a such a global network without having a robust security system running at the background.

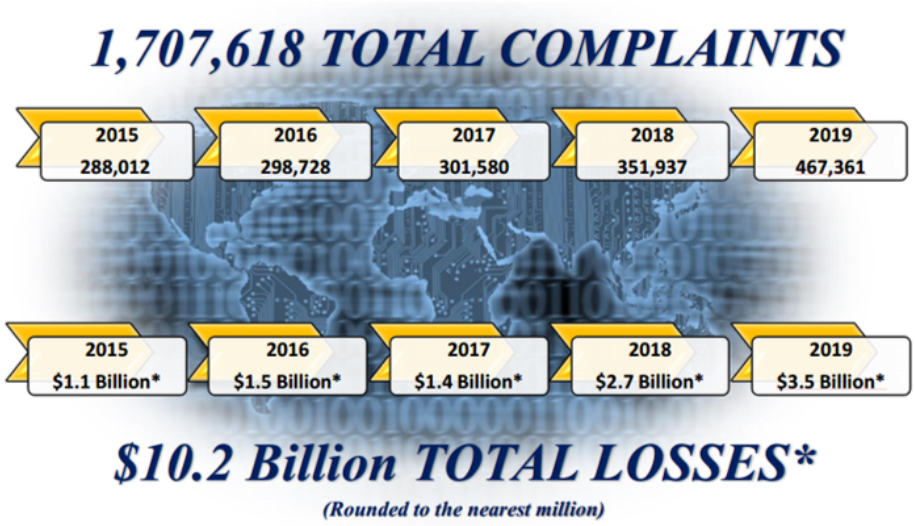

Figure 1.1: Complaints count and amount of money lost over the years between 2015 and 2019 recorded by IC3 [1]

In network security, malware static analysis methods are used to classify the software as either malware or not by extracting several features from its files without the actual need to execute its suspected codes [15], [16]. This means that if it is known what the malware will attempt to do in the system, it is possible to create a specific signature for it to detect its presence in the network at earlier time to prevent any future attack. Recently, this method is gaining more popularity because of the noticeable development in machine learning classification techniques.

Naïve bayes, is a probabilistic algorithm used to categorize elements into classes based on conditional probability, thus it could be used in malware detection as well. The aim of this project is to accelerate an existing version of this algorithm in hardware environment by exploring its hidden concurrency and map it into parallel hardware using pipeline and SIMD approach. This should allow it to work smoothly in the

background to protect the connected devices in real time as a standalone device that can work independently from the central processing unit (CPU) of the device used.

1.2 Problem Statement

Knowing whether a network packet is malware or not, has to be done in real time. This means that the time response is very critical. Naïve bayes classifier is an efficient algorithm in terms of computing reduction. But in most computing systems, the computing resources where shared among several applications. This means that the performance of the algorithm in terms of its overall speed will be affected as the CPU is shared with other applications. In network security, handling the real time processing of data to ensure safe connection is not the only challenge that presented, as improving the user experience by securing the connection with a minimum delay, is required as well. Due to the CPU limitation, even if the target algorithm is considered as a lite algorithm in terms of its computing time, it still not efficient enough to handle the real time processing of data. This is because of the sharing business that the CPU is doing all the time. At this point, it is clear that moving from general computing approach to domain specific computing approach is an essential step.

1.3 Research Gap

As our literature review is concerned, less information regards accelerating the mentioned algorithm using a single instruction multiple data (SIMD) nor using a pipelined design, are presented. Also, our literature review revealed that there were limited studies have focused on implementing naïve baye classifier (NBC) in hardware for malware detection; thus, this research is designed to cover this gap.

1.4 Research Objectives

The objectives of this research are:

- To find the System Verilog design that best improve the performance of the naive algorithm in hardware in terms of throughput and speed for both training and inference part.
- To validate the correctness functionality of the implemented design.

- To benchmark the design outcome by comparing its performance with hardware [17].

1.5 Scope

For the purpose of completing this project, proving whether naïve bayes algorithm is efficient in identifying malicious software or not is outside the scope. This is because it is already done by other researchers before [17]. This project is to improve the performance of the target algorithm in hardware. For simplicity, a simulated database with features similar to the ones used in [17] was created. In addition, it should be recorded that the version of Vivado tool involved in this experiment for design, synthesize, and simulate the project is 2019.1, and the computer device that carry out the whole experiment including the benchmarking business follows these specifications: Intel(R) Core (TM) i-5200, 2.20 GHz, 8 GB ram, 2 GB AMD graphic card, with 64-bit version of Windows 10.

1.6 Thesis Organization

This thesis is organized as follows. Chapter 2 reviews the hardware and software aspects that related to this project, as well as the recent similar works that addressed by other researchers before. Chapter 3 explains the methodology used in this project to implement the design. This includes the algorithm and its hardware implementation. Chapter 4 elaborates the research finding and its standing compared to other existing researches. Chapter 5 summarizes the project accomplishment and provides suggestions for future works.

REFERENCES

1. FBI's Internet and (IC3), C. C. C. 2019 Internet Crime Report. *2019 Internet Crime Report*, 2019: 1–28. URL https://pdf.ic3.gov/2019{_}IC3Report.pdf.

2. Crockett, L., Elliot, R., Enderwitz, M. and Stewart, B. The Zynq Book. 2014: 484. URL http://www.zynqbook.com/.

3. Ha, S. and Teich, J. *Handbook of hardware/software codesign*. Springer Publishing Company, Incorporated. 2017.

4. Liu, D. Application specific instruction set DSP processors. In: *Handbook of Signal Processing Systems*. Springer. 415–447. 2010.

5. Wilab - Wireless communication laboratory Bologna. URL http://wilab.org/teaching-activities/.

6. Introduction to FPGA with Verilog | by Roshanmaharana | Medium. URL https://medium.com/@roshanmaharana1510/introduction-to-fpga-with-verilog-ab6f02cbda34.

7. Nervous System - Definition, Function and Parts | Biology Dictionary. URL https://biologydictionary.net/nervous-system/.

8. Stone, J. V. *Bayes' rule: A tutorial introduction to Bayesian analysis*. Sebtel Press. 2013.

9. Chowdhary, K. *Fundamentals of Artificial Intelligence*. Springer. 2020.

10. James, G., Witten, D., Hastie, T. and Tibshirani, R. *An introduction to statistical learning*. vol. 112. Springer. 2013.

11. Applications of Machine Learning - Javatpoint. URL https://www.javatpoint.com/applications-of-machine-learning.

12. Singer, P. W. and Friedman, A. *Cybersecurity: What everyone needs to know*. oup usa. 2014.

13. Sikorski, M. and Honig, A. *Practical malware analysis: the hands-on guide to dissecting malicious software*. no starch press. 2012.

14. The FBI's Internet Crime Complaint Center (IC3) Marks Its 20th Year — FBI. URL https://www.fbi.gov/news/pressrel/press-releases/the-fbis-internet-crime-complaint-center-ic3-marks-its-20th-year.

15. Nath, H. V. and Mehtre, B. M. Static malware analysis using machine learning methods. *International Conference on Security in Computer Networks and Distributed Systems*. Springer. 2014. 440–450.

16. YusirwanS, S., Prayudi, Y. and Riadi, I. Implementation of malware analysis using static and dynamic analysis method. *IJCA*. 2015, vol. 117. 11–15.

17. Zuki, A. Z. B. M. FPGA implementation of naive bayes classifier for network security. 2018.

18. Theis, T. N. and Wong, H.-S. P. The end of moore's law: A new beginning for information technology. *Computing in Science & Engineering*, 2017. 19(2): 41–50.

19. Cong, J., Sarkar, V., Reinman, G. and Bui, A. Customizable domain-specific computing. *FPL*. 2009. 1.

20. Pang, X., Yu, D., Li, J. and Guo, Y. Design and application of Ip core in soc technology. *2010 Third International Symposium on Information Science and Engineering*. IEEE. 2010. 71–74.

21. Sloss, A., Symes, D. and Wright, C. *ARM system developer's guide: designing and optimizing system software*. Elsevier. 2004.

22. What is Hardware Acceleration? Definition and FAQs | OmniSci. URL https://www.omnisci.com/technical-glossary/hardware-acceleration.

23. Teich, J. Hardware/software codesign: The past, the present, and predicting the future. *Proceedings of the IEEE*, 2012. 100(Special Centennial Issue): 1411–1430.

24. Hemsoth, N. and Morgan, T. P. *FPGA frontiers: new applications in reconfigurable computing*. Next Platform Press. 2017.

25. Santarin, M. Xilinx Redefines State of the Art With New 7 Series FPGAs. *Xcell Journal, Third Quarter*, 2010.

26. Xilinx, A. Reference Guide, UG761 (v13. 1). *URL http://www. xilinx. com/support/documentation/ip documentation/ug761 axi reference guide. pdf*, 2011.

27. The History of Artificial Intelligence | Data Science For The Earth. URL https://www.datasciencearth.com/en/the-history-of-artificial-intelligence/.

28. Artificial intelligence | Definition, Examples, and Applications | Britannica. URL https://www.britannica.com/technology/artificial-intelligence.

29. Rumelhart, D. E., Hinton, G. E. and Williams, R. J. Learning representations by back-propagating errors. *nature*, 1986. 323(6088): 533–536.

30. Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A. and Chanona-Hernández, L. Syntactic n-grams as machine learning features for natural language processing. *Expert Systems with Applications*, 2014. 41(3): 853–860.

31. Granik, M. and Mesyura, V. Fake news detection using naive Bayes classifier. *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*. IEEE. 2017. 900–903.

32. Müller, A. C., Guido, S. *et al. Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.". 2016.

33. Elkan, C. Boosting and naive Bayesian learning. *Proceedings of the international conference on knowledge discovery and data mining*. 1997.

34. Meng, H., Appiah, K., Hunter, A. and Dickinson, P. Fpga implementation of naive bayes classifier for visual object recognition. *CVPR 2011 WORKSHOPS*. IEEE. 2011. 123–128.

35. Appiah, K., Hunter, A., Dickinson, P. and Meng, H. Binary object recognition system on FPGA with bSOM. *23rd IEEE international SOC conference*. IEEE. 2010. 254–259.

36. Xue, Z., Wei, J. and Guo, W. A Real-Time Naive Bayes Classifier Accelerator on FPGA. *IEEE Access*, 2020. 8: 40755–40766.

37. Paul, S., Jayakumar, N. and Khatri, S. P. A fast hardware approach for approximate, efficient logarithm and antilogarithm computations. *IEEE transactions on very large scale integration (vlsi) systems*, 2008. 17(2): 269–277.

38. Bariamis, D., Maroulis, D. and Iakovidis, D. K. Adaptable, fast, area-efficient architecture for logarithm approximation with arbitrary accuracy on FPGA. *Journal of Signal Processing Systems*, 2010. 58(3): 301–310.

39. Klinefelter, A., Ryan, J., Tschanz, J. and Calhoun, B. H. Error-energy analysis of hardware logarithmic approximation methods for low power applications. *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2015. 2361–2364.

40. Shrestha, R. and Paily, R. P. VLSI design and hardware implementation of high-speed energy-efficient logarithmic-MAP decoder. *Journal of Low Power Electronics*, 2015. 11(3): 406–412.

41. Ha, M. and Lee, S. Accurate hardware-efficient logarithm circuit. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2016. 64(8): 967–971.

42. Chen, J. and Liu, X. A fast and accurate logarithm accelerator for scientific applications. *2017 IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE. 2017. 208–208.

43. Ansari, M. S., Cockburn, B. F. and Han, J. A hardware-efficient logarithmic multiplier with improved accuracy. *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2019. 928–931.

44. Patterson, D. A. and Hennessy, J. L. *Computer Organization and Design, Enhanced: The Hardware/Software Interface*. Morgan Kaufmann. 2014.