INTERLEAVED INCREMENTAL-DECREMENTAL SUPPORT VECTOR MACHINE FOR EMBEDDED APPLICATIONS

JEEVAN A/L SIRKUNAN

UNIVERSITI TEKNOLOGI MALAYSIA

INTERLEAVED INCREMENTAL-DECREMENTAL SUPPORT VECTOR MACHINE FOR EMBEDDED APPLICATIONS

JEEVAN A/L SIRKUNAN

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Doctor of Philosophy

School of Electrical Engineering
Faculty of Engineering
Universiti Teknologi Malaysia

JULY 2022

ACKNOWLEDGEMENT

First and foremost, I would like to thank God for giving me the strength to complete this thesis. I would also like to express my greatest appreciation to my supervisor, Associate Professor Ir. Dr. Muhammad Nadzir Bin Marsono, for allowing me to work in an amazing field of research. With his continuous encouragement, criticism, and guidance, I completed my research. Thanks to him, I realized my full potential in academics and other aspects of life. Besides that, I would like to thank my co-supervisor, Associate Professor Dr. Shaikh Nasir @ Nasir bin Shaikh Husin for commenting and guiding me on my work and involving me in external projects, which expanded my horizon in this field.

My sincerest appreciation goes to my fellow lecturers Dr. Jasmine Hau Yuan Wen, Prof Dr. Mohamed Khalil Hani, Dr. Usman Ullah Sheikh, for their support and technical advice. I would also like to thank my all fellow researchers Mohammed Sultan Mohammed, Dr. Mohd Shahrizal Rusli, Dr. Mosab Hamdan, Dr. Tang Jia Wei, Dr. Lee Yee Hui, Dr. Loo Hui Ru, Tan Tze Hon and Tei Yin Zhen for accompanying, improving and supporting me through hard times during my study here.

I would also like to thank the utmthesis LaTeXproject developers for making the thesis writing process a lot easier for me. Thanks to them, I could focus on the content of the thesis and not waste time on formatting issues.

Finally, I would like to thank my family for always being there for me through thick and thin. Especially my parents, who never gave up believing in me. Their role in my life is something I will always need and constantly appreciate.

ABSTRACT

Incremental Decremental Support Vector Machine (IDSVM) is one of the widely used incremental learning algorithms known for its high accuracy for data stream analytics and high computational complexity. One of the biggest problems of IDSVM is that the model scales with the input data set size that directly correlates with the computational and memory resources. In order to deploy IDSVM in an embedded system with limited memory, a moving window architecture is needed to limit the kernel sizes. However, this also increases the overall complexity of the algorithm since each data instance needs to be unlearned when exiting the window. This thesis proposes an Interleaved IDSVM (IIDSVM) algorithm that performs incremental and decremental learning concurrently. The interleaved method can reduce the overall kernel size and consume less memory. This thesis also proposes a reduced-division IIDSVM algorithm that replaces the more complex division operations with simpler inverse multiplications. Certain IIDSVM tasks can be simplified by replacing most of the complex divisions with inverse multiplication that can achieve a similar outcome since only a single sample variation value is used to update the weights.Finally, a Radial Basis Function (RBF) kernel, which is a widely used kernel in SVM, is proposed to be implemented as a hardware accelerator to speed up the computation time of the IIDSVM. Based on our experiments, the proposed IIDSVM achieved a speedup of 2.5 - 4.2$\times$ on computation time while producing similar accuracy as IDSVM and LIBSVM. Furthermore, the reduced-division IIDSVM can improve computation time up to 1.4$\times$ on a Nios II embedded platform for certain data sets. The RBF kernel's hardware implementation is analyzed on the Stratix V Field Programmable Gate Array (FPGA) platform. It can perform up to four orders of magnitude faster than the software implementation on the Nios II embedded processor for data sets with 8, 12, and 16 feature sizes. Besides that, the proposed architecture RBF kernel can maintain a maximum operating frequency of approximately 200Mhz for feature sizes 8, 12, and 16. Collectively the proposed works can improve the runtime of incremental SVM compute-intensive data stream analytics.

ABSTRAK

Mesin Vektor Sokongan Tokokan dan Susutan (IDSVM) ialah salah satu algoritma pembelajaran tokokan yang digunakan secara meluas dengan ketepatan yang tinggi untuk analitik aliran data tetapi mempunyai kerumitan pengiraannya yang tinggi. Salah satu masalah terbesar IDSVM ialah model berskala dengan saiz set data input yang berkorelasi secara langsung dengan sumber pengiraan dan memori. Untuk menggunakan IDSVM dalam sistem terbenam dengan memori terhad, seni bina seakan tetingkap bergerak diperlukan untuk mengehadkan saiz kernel. Walau bagaimanapun, ini juga meningkatkan kerumitan keseluruhan algoritma kerana setiap tika perlu dilupakan apabila keluar dari tetingkap. Tesis ini mencadangkan algoritma IDSVM antara kembar (IIDSVM) yang melaksanakan pembelajaran tokokan dan susutan secara serentak. Kaedah antara kembar boleh mengurangkan saiz kernel keseluruhan dan menggunakan memori yang lebih kecil. Tesis ini juga mencadangkan algoritma IIDSVM dengan pengurangan pembahagian dan menggantikan operasi bahagi yang kompleks dengan pendaraban songsang yang lebih mudah. Tugas-tugas tertentu dalam IIDSVM boleh dipermudahkan dengan menggantikan sebahagian besar pembahagian kompleks dengan pendaraban songsang bagi mencapai hasil yang serupa kerana hanya satu nilai variasi sampel digunakan untuk mengemas kini pemberat. Akhir sekali, kernel Radial Basis Function (RBF), yang merupakan kernel yang digunakan secara meluas dalam SVM, dicadangkan untuk dilaksanakan sebagai pemecut perkakasan untuk mempercepatkan masa pengiraan daripada IIDSVM. Berdasarkan eksperimen kami, IIDSVM yang dicadangkan mencapai penambahbaikan 2.5 - 4.2$\times$ pada masa pengiraan sambil menghasilkan ketepatan yang serupa seperti IDSVM dan LIBSVM. IIDSVM pembahagian berkurang boleh meningkatkan masa pengiraan sehingga 1.4$\times$ pada platform terbenam Nios II untuk set data tertentu. Pelaksanaan perkakasan kernel RBF dianalisis pada platform Stratix V Tatasusunan Get Boleh Program Medan (FPGA). Pemecut perkakasan RBF boleh melakukan sehingga empat urutan magnitud lebih pantas daripada pelaksanaan perisian pada pemproses terbenam NiosII untuk set data dengan saiz ciri 8, 12 dan 16. Selain itu, kernel RBF seni bina yang dicadangkan boleh mengekalkan kekerapan operasi maksimum kira-kira 200Mhz untuk saiz ciri 8, 12, dan 16. Secara kolektif kerja-kerja yang dicadangkan boleh meningkatkan daya pemprosesan bagi pengiraan intensif analitik aliran data SVM.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AOSVR	–	Accurate On-line Support Vector Regression
FPGA	–	Field Programmable Gate Array
GUI	–	Graphic User Interface
HPO	–	Hyperparameter Optimization
IDSVM	–	Incremental Decremental Support Vector Machine
IELM	–	Incremental Extreme Learning Machine
IIDSVM	–	Interleave Incremental Decremental Support Vector Machine
ILVQ	–	Incremental Learning Vector Quantization
ISVM	–	Incremental Support Vector Machine
LIBSVM	–	Library Support Vector Machine
LPP	–	Learn++
MAE	–	Mean Absolute Error
NB	–	Naive Bayes
ORF	–	On-line Random Forest
RBF	–	Radial Basis Function
SGD	–	Stochastic Gradient Descent
SMO	–	Sequential Minimal Optimization
SoC	–	System on Chip
SVM	–	Support Vector Machine

LIST OF SYMBOLS

α	–	Lagrangian Multiplier
ϵ	–	Margin of tolerance
ζ	–	Slack variable
$\boldsymbol{\theta}$	–	Weights
λ	–	RBF kernel parameter
ϕ	–	Prequential error fading factor
$(\mathbf{x}_c, y_c)$	–	Candidate sample
b	–	Bias
C	–	Regularization parameter
Dir	–	Direction
Er_set	–	Error set
f	–	Frequency
H_c	–	Candidate sample error margin
No_clk	–	Number of clock cycles
$\mathbf{Q}$	–	Kernel matrix
$\mathbf{R}$	–	R matrix
Re_set	–	Remainder set
Sv_set	–	Support vector set
t	–	Elapsed time
W	–	Window size
$\mathbf{x}_c$	–	Candidate vector
y_c	–	Candidate target output

becomes apparent in embedded systems with limited computing power and memory. Storing and processing a very large volume of data will be deemed infeasible. Thus, incremental learning is needed for Big Data applications since it can process data in real-time and update its model as data arrives [7].

1.1 Embedded Incremental Learning

As a new data instance arrives, incremental learning in an embedded system would allow it to learn beyond its design and production phases. However, the environment and habits may differ depending on the user and in certain application traffic varies with time or season. Therefore dedicated machine learning model is required to adapt to the concept changes. The batch learning approach for an embedded system can overcome this problem by outsourcing its data to the cloud, giving it access to a shared pool of configurable computing resources. By outsourcing data, users of embedded system applications can be relieved from local data storage maintenance, computation burden and communication to the cloud. However, users no longer have physical possession of large outsourced data, which makes data integrity and protection a huge problem. Data encryption can be added, though this will introduce additional latency to the overall system.

Applications such as self-driving autonomous robots, robotic vacuum cleaners and lawnmowers can benefit from fast and reliable learning to perform decision-making when their environments are constantly changing [1, 6, 8]. Incremental learning can continually learn beyond the production phase. Moreover, for cloud computing to work, a permanent and reliable connection is needed; and such infrastructure may not be available in the location where the embedded systems are deployed. Besides that, users also may not want to part with their data due to privacy issues. Reliance on cloud systems to process all raw data would incur additional latency. Having an embedded device with continuous learning capability can provide earlier decision making [1]. Arduino Nano 33 [9], ARM Cortex-M processors [10], and Field Programmable Gate Arrays (FPGAs) [11] are some of the example of platform when embedded learning are implemented. The constrain on application is dependent on the platform in which the machine learning is implemented.

Multiple challenges need to be addressed in incorporating incremental learning in an embedded system. For instance, in stream-like architecture, data that have passed cannot be recovered. Therefore, incoming data need to be incorporated into the model as soon as possible. Besides that, limited embedded system memory limits the number of training samples that can be maintained. Therefore, the system memory needs to continuously remove old data and update the latest incoming data. On the whole, embedded incremental learning needs to be fast, accurate, and forget or unlearn outdated data.

There are several algorithms for incremental learning such as Incremental Decremental Support Vector Machine (IDSVM) [12], LASVM [13], On-line Random Forest (ORF) [14], Incremental Learning Vector Quantization (ILVQ) [15], Learn++ (LPP_{CART}) [16], Incremental Extreme Learning Machine (IELM) [17], Naive Bayes (NB_{Gauss}) [18], and Stochastic Gradient Descent $SGDB_{Lin}$) [19]. The aforementioned algorithms are able to perform instance incremental learning where learning takes place as data arrives. Each of these algorithms has various levels of complexity which result in different levels of performance in terms of accuracy and timing. Based on a comparison study on several incremental learning algorithm done in [1], IDSVM has the best accuracy but high computational complexity.

1.2 Problem Statement

IDSVM is one of the most accurate incremental learning algorithms for a variety of data set [1]. However, IDSVM has difficulty handling a very large data set due to its algorithmic limitation. When applied to a large data set, the IDSVM has to compute many support vectors, resulting in a large inverse kernel matrix which causes an exponential increase in memory size requirement. In order to cope with this limitation on IDSVM, Shao et al. [20], and Ma et al. [21] applied a moving window approach to limit the number of support vectors to a fixed value. With the moving window approach, IDSVM can be applied to larger data sets, and hardware implementation on an embedded platform and Field Programmable Gate Array (FPGA) devices becomes feasible [20].

The data samples within an IDSVM model cannot be removed. Instead, it has to be unlearned, where the weights of all remaining data samples are recalibrated as a single data sample is being unlearned. The complexity of unlearning a data sample is similar to learning it [12]. Therefore, the moving window approach adds complexity to the overall algorithm and directly reduces the system's overall throughput. The added computation of unlearning every time new data arrives becomes even more critical for embedded platform implementation, where processing capability is limited and operates at a lower clock frequency.

To the best of our knowledge, the first and only complete implementation of IDSVM in an FPGA device was done by Shao et al. [20]. In this work, IDSVM was implemented using a dataflow programming language, Maxeler J, targeted for regression problems. The implementation was done on a Stratix V FPGA device. The IDSVM algorithm was based on the work by Cauwenberghs and Poggio [12]. Shao et al. [20] tested their implementation for stock prediction. It also discussed the algorithm restructuring for faster memory access time. However, open problems on the IDSVM complexity on window based implementation are yet to be explored.

For embedded system platforms like ARM Cortex M33, M22 [10], and Nios II/f [22] embedded processor, the division instruction takes longer execution time compared to multiplication. Besides that, within the IDSVM algorithm, many division computations are repeated continuously during the learning and unlearning processes. The division operations that involve a constant denominator can be optimized into inverse multiplication during compile time [23–25]. However, the division operation in the IDSVM algorithm involves two variables, thus requiring the division instruction from the compiler, that reduces the overall performance of IDSVM on embedded systems. Reducing the number of division operations within the IDSVM algorithm is currently an open problem.

There are numerous works aimed at implementing SVM with Radial Basis Function (RBF) kernel for embedded systems, such as [20, 26–30]. The RBF kernel is the most popular choice in many applications since it can handle a wide range of data formats and has only a few tuning parameters [31]. Based on [32], the computation time for an IDSVM algorithm mainly revolves around RBF kernel computation. However, the RBF kernel hardware architecture for the IDSVM algorithm is yet to be explored.

1.3 Objectives

The primary aim of this thesis is to propose an IDSVM algorithm that is suitable for embedded systems. The proposed algorithm is targeted to have faster runtime and low memory footprint. Besides that, the algorithm can be optimized better to suit the limited capability of an embedded system. Specifically, this thesis proposes the following objectives:

1. To propose an improved IDSVM algorithm that allows the learning and unlearning tasks to be done in parallel. The IDSVM algorithm is analyzed for tasks that can be performed in parallel. Next, the Interleaved Incremental Support Vector Machine (IIDSVM) algorithm is proposed using a dual window that enables simultaneous learning and unlearning. The IIDSVM is then implemented and analyzed for overall runtime, accuracy and memory utilization.
2. To propose an improved algorithm of IIDSVM by reducing the number of division operations. The reduced-division IIDSVM is targeted to speed up IIDSVM on embedded platform with an acceptable accuracy. The IIDSVM algorithm is first analyzed, and parts of the algorithm that utilizes division operations are replaced using inverse multiplication to achieve similar accuracy. The proposed algorithm is then validated against the unmodified IIDSVM on and the overall runtime is analyzed.
3. To design a fully pipelined hardware architecture of the RBF kernel for the IIDSVM. The proposed architecture is parameterizable and minimizes the data transfer needed between the hardware and software partitions. The proposed

hardware architecture is then analyzed for runtime, accuracy, resource utilization and maximum operating frequency.

1.4 Scope of Work

The IDSVM implementation in this thesis is based from Parrella's work [33] on MATLAB. Parrella's IDSVM [33] targets online learning for regression based on Cauwenberghs and Poggio work [12]. There are many variations of IDSVM, and Cauwenberghs and Poggio's work [12] has always been referred to as the true incremental support vector machine [1]. The main IDSVM benchmark is then modified with moving window architecture based on Shao et al. [20] architecture.

The IDSVM algorithm in this thesis only targets regression problems. Similar to [20], the proposed work also targets stock price applications. For classification data sets that consist of multiple classes, each class needs to be compared against other classes. SVM, by nature, is a binary classifier. In order to handle multiple classes, multiple incremental learning models are required. However, only a single model is required for regression applications and binary classification problems. The discussion on the effect of multi-class application towards the proposed algorithm is beyond the scope of this thesis.

Four regression data sets from the different applications are used for functional and performance validation. The data set consists of abalone [34], cadata [35], cpusmall [36], and stock_mid [37]. Abalone, cadata and cpu_small data sets are available in LIBSVM [31]. These data sets are taken from secondary sources since the data structure had been preprocessed to be compatible with LIBSVM. The stock-mid data is taken directly from reference [20] on incremental SVM implementation, and the data structure is already compatible with LIBSVM.

Multiple kernels can be applied on LIBSVM, such as Linear, Polynomial, RBF, or Sigmoid. In this thesis, the RBF kernel is used for all implementations. RBF kernel is the most common choice in many applications mainly due to its ability to handle a wide range of data types as well as having only a few parameters for tuning [31].

The FPGA device used in this thesis is Intel Stratix V. For embedded processor implementation of the incremental SVM, Nios II soft-core processor is used. The Nios II processor core is a 32-bit RISC processor optimized for use in Intel's mainstream FPGAs. Only the RBF kernel is implemented in hardware for the FPGA implementation of the proposed IIDSVM. The kernel is the most compute-intensive section in the incremental SVM algorithm [32]. In addition, this section of the algorithm is deterministically recursive and has the potential for parallelism. The proposed RBF kernel is designed without automation using Verilog code.

For objective 1, the IIDSVM experiment is done on a MATLAB environment. The computer has an Intel I5 4460 CPU, with 8 GB of memory. The proposed IIDSVM is evaluated for overall runtime, accuracy and memory utilization. For objective 2, a single Nios II SoC is used for the embedded platform, and the implementation is evaluated in terms of accuracy and runtime. The single Nios II implementation of IDSVM emulated the single window implementation. For the IIDSVM, the results are approximated from IDSVM result using the speed-up factor obtained from objective 1. Finally, for objective 3 the proposed RBF kernel hardware implementation for IIDSVM is implemented on Stratix V FPGA and analyzed for runtime, accuracy, resource utilization and maximum operating frequency.

1.5 Contribution

The following are the expected contribution of the thesis:

1. The proposed IIDSVM algorithm is an improved algorithm over the conventional IDSVM algorithm proposed in Shao et al. [20]. The IIDSVM algorithm enables learning and unlearning to be performed simultaneously, resulting in higher throughput and lesser memory utilization for the kernel implementation.
2. Specific compute-intensive division operations within the IIDSVM algorithm are substituted with inverse multiplication. As a result, the proposed algorithm reduces the overall computational complexity, which results in faster computational time while producing similar accuracy.
3. A hardware-accelerated RBF kernel module for IIDSVM is developed. The hardware module is parameterizable and can adapt to different application requirements. Furthermore, the proposed hardware computes the RBF kernel much faster than the equivalent software model implemented in the embedded processor core.

1.6 Thesis Structure

The remainder of the thesis is organized as follows.

Chapter 2 covers literature-related works and discusses important aspects of incremental learning, mainly focusing on IDSVM. First, a comparison between batch and incremental learning is discussed. The application and justification of incremental learning are also included in this chapter. Next is the discussion on SVM and its algorithm developments, followed by a discussion on related works on IDSVM and the existing hardware implementation of IDSVM. Finally, this chapter ends with a discussion on the motivation for extended works on IDSVM based on the limitation of prior implementations.

Chapter 3 provides the methodology for the thesis work. This chapter also includes the general approach in IDSVM research presented in this thesis and the tools and platform used to model the proposed IIDSVM. The final section in this chapter describes the data sets used for verifications and the prequential analysis used to evaluate an incremental machine learning model's performance accurately.

Chapter 4 presents the proposed IIDSVM algorithm. This chapter first discusses the details on the development of the IDSVM. Then, the discussion continues to the moving window method for hardware implementation for the proposed IIDSVM. The proposed method is then compared with the conventional method and analyzed for overall runtime, accuracy and memory utilization.

Chapter 5 focuses on optimizing IIDSVM for embedded systems by reducing division operations. This chapter starts by comparing multiplication and division operations for embedded system applications. This is followed by a detailed analysis of the IIDSVM algorithm, where division operation can be replaced with multiplication and comparator operation. Finally, the proposed algorithm is compared with the unaltered IIDSVM for the overall runtime.

Chapter 6 discusses the fully pipelined RBF kernel implementation architecture in hardware. First, the RBF kernel equation is discussed, followed by the features and specifications needed to operate in an incremental learning environment. This is then followed by details on the hardware architecture of the RBF kernel. Finally, the performance of the proposed hardware is analyzed in terms of runtime, accuracy, resource utilization and maximum operating frequency.

Chapter 7 summarises the outcome of research objectives re-stating the contribution to knowledge and their significance and suggestions for future research directions.

REFERENCES

1. Losing, V., Hammer, B. and Wersing, H. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 2018. 275: 1261–1274.

2. Alippi, C. Learning in nonstationary and evolving environments. In: *Intelligence for Embedded Systems*. Springer. 211–247. 2014.

3. Hoi, S. C., Wang, J. and Zhao, P. Libol: A library for online learning algorithms. *Journal of Machine Learning Research*, 2014. 15(1): 495.

4. Yang, R. and Newman, M. W. Learning from a learning thermostat: lessons for intelligent systems for the home. *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. 2013. 93–102.

5. Pagani, S., Manoj, P. S., Jantsch, A. and Henkel, J. Machine learning for power, energy, and thermal management on multicore processors: A survey. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018. 39(1): 101–116.

6. Forlizzi, J. and DiSalvo, C. Service robots in the domestic environment: a study of the roomba vacuum in the home. *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*. 2006. 258–265.

7. Chen, M., Mao, S. and Liu, Y. Big data: A survey. *Mobile networks and applications*, 2014. 19(2): 171–209.

8. Losing, V., Hammer, B. and Wersing, H. Interactive online learning for obstacle classification on a mobile robot. *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2015. 1–8.

9. Warden, P. and Situnayake, D. *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media. 2019.

10. Arm Cortex-M series processors, 2022. URL `https://developer.arm.com/ip-products/processors/cortex-m`.

11. Jiao, B., Zhang, J., Xie, Y., Wang, S., Zhu, H., Kang, X., Dong, Z., Zhang, L. and Chen, C. A 0.57-GOPS/DSP Object Detection PIM Accelerator on

FPGA. *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE. 2021. 13–14.

12. Cauwenberghs, G. and Poggio, T. Incremental and decremental support vector machine learning. *Advances in neural information processing systems*, 2001: 409–415.

13. Bordes, A., Ertekin, S., Weston, J. and Bottou, L. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 2005. 6(Sep): 1579–1619.

14. Saffari, A., Leistner, C., Santner, J., Godec, M. and Bischof, H. On-line random forests. *2009 ieee 12th international conference on computer vision workshops, iccv workshops*. IEEE. 2009. 1393–1400.

15. Sato, A. and Yamada, K. Generalized learning vector quantization. *Advances in neural information processing systems*. 1996. 423–429.

16. Bunte, K., Schneider, P., Hammer, B., Schleif, F.-M., Villmann, T. and Biehl, M. Limited rank matrix learning, discriminative dimension reduction and visualization. *Neural Networks*, 2012. 26: 159–173.

17. Liang, N.-Y., Huang, G.-B., Saratchandran, P. and Sundararajan, N. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on neural networks*, 2006. 17(6): 1411–1423.

18. Zhang, H. The optimality of naive Bayes. *Aa*, 2004. 1(2): 3.

19. Richtárik, P. and Takáč, M. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 2016. 156(1-2): 433–484.

20. Shao, S., Mencer, O. and Luk, W. Dataflow design for optimal incremental svm training. *Field-Programmable Technology (FPT), 2016 International Conference on*. IEEE. 2016. 197–200.

21. Ma, J., Theiler, J. and Perkins, S. Accurate on-line support vector regression. *Neural computation*, 2003. 15(11): 2683–2703.

22. Nios II Classic Processor Reference Guide, 2016. URL `https://www.intel.com/content/dam/support/us/en/programmable/support-resources/bulk-container/pdfs/literature/hb/nios2/n2cpu-nii5v1-01.pdf`.

23. Granlund, T. and Montgomery, P. L. Division by invariant integers using multiplication. *Proceedings of the ACM SIGPLAN 1994 conference on Programming language design and implementation*. 1994. 61–72.

24. Cavagnino, D. and Werbrouck, A. E. Efficient algorithms for integer division by constants using multiplication. *The Computer Journal*, 2008. 51(4): 470–480.

25. Godbolt, M. Optimizations in C++ compilers. *Communications of the ACM*, 2020. 63(2): 41–49.

26. Cutajar, M., Gatt, E., Grech, I., Casha, O. and Micallef, J. Hardware-based support vector machine for phoneme classification. *EUROCON, 2013 IEEE*. IEEE. 2013. 1701–1708.

27. Qasaimeh, M., Sagahyroon, A. and Shanableh, T. FPGA-based parallel hardware architecture for real-time image classification. *IEEE Transactions on Computational Imaging*, 2015. 1(1): 56–70.

28. Kyrkou, C., Bouganis, C.-S., Theocharides, T. and Polycarpou, M. M. Embedded hardware-efficient real-time classification with cascade support vector machines. *IEEE transactions on neural networks and learning systems*, 2015. 27(1): 99–112.

29. Ramadurgam, S. and Perera, D. G. An Efficient FPGA-Based Hardware Accelerator for Convex Optimization-Based SVM Classifier for Machine Learning on Embedded Platforms. *Electronics*, 2021. 10(11): 1323.

30. Cadambi, S., Durdanovic, I., Jakkula, V., Sankaradass, M., Cosatto, E., Chakradhar, S. and Graf, H. P. A massively parallel FPGA-based coprocessor for support vector machines. *Field Programmable Custom Computing Machines, 2009. FCCM'09. 17th IEEE Symposium on*. IEEE. 2009. 115–122.

31. Chang, C.-C. and Lin, C.-J. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2011. 2(3): 27.

32. Laskov, P., Gehl, C., Krüger, S. and Müller, K.-R. Incremental support vector learning: Analysis, implementation and applications. *Journal of machine learning research*, 2006. 7(Sep): 1909–1936.

33. Parrella, F. Online support vector regression. *Master's Thesis, Department of Information Science, University of Genoa, Italy*, 2007. 69.

34. abalone_data_set. `http://archive.ics.uci.edu/ml/datasets/Abalone`. Accessed: 2018-09-30.

35. cadata_data_set. `http://lib.stat.cmu.edu/datasets/`. Accessed: 2018-09-30.

36. cadata_data_set. `http://www.cs.toronto.edu/~delve/data/datasets.html`. Accessed: 2018-09-30.

37. stock_mid_data_set. `https://github.com/custom-computing-ic/SVM`. Accessed: 2010-09-30.

38. Norvig, P. R. and Intelligence, S. A. *A modern approach*. Prentice Hall. 2002.

39. Hinton, G. E., Sejnowski, T. J. *et al. Unsupervised learning: foundations of neural computation*. MIT press. 1999.

40. Abdi, H. and Williams, L. J. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2010. 2(4): 433–459.

41. Zhu, X. and Goldberg, A. B. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 2009. 3(1): 1–130.

42. Loo, H. R. and Marsono, M. N. Online network traffic classification with incremental learning. *Evolving Systems*, 2016. 7(2): 129–143.

43. Kaelbling, L. P., Littman, M. L. and Moore, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 1996. 4: 237–285.

44. Carbonara, L. and Borrowman, A. A comparison of batch and incremental supervised learning algorithms. *European Symposium on Principles of Data Mining and Knowledge Discovery*. Springer. 1998. 264–272.

45. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H. and He, Q. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 2020. 109(1): 43–76.

46. Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 2009. 22(10): 1345–1359.

47. Read, J., Bifet, A., Pfahringer, B. and Holmes, G. Batch-incremental versus instance-incremental learning in dynamic and evolving data. *International Symposium on Intelligent Data Analysis*. Springer. 2012. 313–323.

48. Costa, R. The Internet of moving things [industry view]. *IEEE Technology and Society Magazine*, 2018. 37(1): 13–14.

49. Quigley, M. and Burke, M. Low-cost Internet of Things digital technology adoption in SMEs. *International Journal of Management Practice*, 2013. 6(2): 153–164.

50. Sanchez-Iborra, R. and Skarmeta, A. F. Tinyml-enabled frugal smart objects: Challenges and opportunities. *IEEE Circuits and Systems Magazine*, 2020. 20(3): 4–18.

51. Niu, W., Ma, X., Lin, S., Wang, S., Qian, X., Lin, X., Wang, Y. and Ren, B. Patdnn: Achieving real-time dnn execution on mobile devices with pattern-based weight pruning. *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. 2020. 907–922.

52. Shafique, M., Theocharides, T., Reddy, V. J. and Murmann, B. TinyML: Current Progress, Research Challenges, and Future Roadmap. *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE. 2021. 1303–1306.

53. Cai, H., Gan, C., Wang, T., Zhang, Z. and Han, S. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

54. Wang, T., Wang, K., Cai, H., Lin, J., Liu, Z., Wang, H., Lin, Y. and Han, S. Apq: Joint search for network architecture, pruning and quantization policy. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020. 2078–2087.

55. Rusci, M., Capotondi, A. and Benini, L. Memory-driven mixed low precision quantization for enabling deep network inference on microcontrollers. *Proceedings of Machine Learning and Systems*, 2020. 2: 326–335.

56. Vapnik, V. N. and Vapnik, V. *Statistical learning theory*. vol. 1. Wiley New York. 1998.

57. Berwick, R. An Idiot's guide to Support vector machines (SVMs). *Retrieved on October*, 2003. 21: 2011.

58. Vapnik, V. *The nature of statistical learning theory*. Springer Science & Business Media. 2013.

59. Smola, A. J. and Schölkopf, B. A tutorial on support vector regression. *Statistics and computing*, 2004. 14(3): 199–222.

60. Bustio-Martínez, L., Cumplido, R., Hernández-Palancar, J. and Feregrino-Uribe, C. On the Design of a Hardware-Software Architecture for Acceleration of SVM's Training Phase. *Mexican Conference on Pattern Recognition*. Springer. 2010. 281–290.

61. Cao, K.-k., Shen, H.-b. and Chen, H.-f. A parallel and scalable digital architecture for training support vector machines. *Journal of Zhejiang University SCIENCE C*, 2010. 11(8): 620–628.

62. Mahmoodi, D., Soleimani, A., Khosravi, H., Taghizadeh, M. *et al.* FPGA simulation of linear and nonlinear support vector machine. *Journal of Software Engineering and Applications*, 2011. 4(05): 320.

63. Lazer, D., Kennedy, R., King, G. and Vespignani, A. The parable of Google Flu: traps in big data analysis. *Science*, 2014. 343(6176): 1203–1205.

64. Syed, N. A., Huan, S., Kah, L. and Sung, K. Incremental learning with support vector machines. 1999.

65. Rüping, S. *Incremental learning with support vector machines*. Technical report. Technical Report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund. 2002.

66. Xu, J., Xu, C., Zou, B., Tang, Y. Y., Peng, J. and You, X. New Incremental Learning Algorithm With Support Vector Machines. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018. (99): 1–12.

67. Hao, Y. and Zhang, H. A fast incremental learning algorithm based on twin support vector machine. *Computational Intelligence and Design (ISCID), 2014 Seventh International Symposium on*. IEEE. 2014, vol. 2. 92–95.

68. Zheng, J., Shen, F., Fan, H. and Zhao, J. An online incremental learning support vector machine for large-scale data. *Neural Computing and Applications*, 2013. 22(5): 1023–1035.

69. Diehl, C. P. and Cauwenberghs, G. SVM incremental learning, adaptation and optimization. *Neural Networks, 2003. Proceedings of the International Joint Conference on*. IEEE. 2003, vol. 4. 2685–2690.

70. Gâlmeanu, H., Sasu, L. M. and Andonie, R. Incremental and Decremental SVM for Regression. *International Journal of Computers Communications & Control*, 2016. 11(6): 755–775.

71. Karasuyama, M. and Takeuchi, I. Multiple incremental decremental learning of support vector machines. *IEEE Transactions on Neural Networks*, 2010. 21(7): 1048–1059.

72. Chen, B.-W. Recursion-Free Online Multiple Incremental/Decremental Analysis Based on Ridge Support Vector Learning. *arXiv preprint arXiv:1608.00619*, 2016.

73. Le, D. V.-K., Chen, Z., Wong, Y. W. and Isa, D. A complete online-SVM pipeline for case-based reasoning system: a study on pipe defect detection system. *Soft Computing*, 2020: 1–17.

74. Chen, Y., Xiong, J., Xu, W. and Zuo, J. A novel online incremental and decremental learning algorithm based on variable support vector machine. *Cluster Computing*, 2019. 22(3): 7435–7445.

75. Khemchandani, R., Chandra, S. *et al.* Twin support vector machines for pattern classification. *IEEETransactions on pattern analysis and machine intelligence*, 2007. 29(5): 905–910.

76. Tian, Y. and Qi, Z. Review on: twin support vector machines. *Annals of Data Science*, 2014. 1(2): 253–277.

77. Lichman, M. UCI Machine Learning Repository, 2013. URL `http://archive.ics.uci.edu/ml`.

78. Furao, S. and Hasegawa, O. An incremental network for on-line unsupervised classification and topology learning. *Neural networks*, 2006. 19(1): 90–106.

79. Schölkopf, B., Smola, A. J., Williamson, R. C. and Bartlett, P. L. New support vector algorithms. *Neural computation*, 2000. 12(5): 1207–1245.

80. Gu, B., Wang, J.-D., Yu, Y.-C., Zheng, G.-S., Huang, Y.-F. and Xu, T. Accurate on-line ν-support vector learning. *Neural Networks*, 2012. 27: 51–59.

81. libsvm_data_set. `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`. Accessed: 2022-05-13.

82. Gu, B. and Sheng, V. S. Feasibility and Finite Convergence Analysis for Accurate On-Line \ ν-Support Vector Machine. *IEEE Transactions on Neural Networks and Learning Systems*, 2013. 24(8): 1304–1315.

83. Gu, B., Sheng, V. S., Wang, Z., Ho, D., Osman, S. and Li, S. Incremental learning for ν-support vector regression. *Neural Networks*, 2015. 67: 140–150.

84. Asuncion, A. and Newman, D. UCI machine learning repository, 2007.

85. Pistikopoulos, E. N. *Multi-parametric programming*. Wiley-vch. 2007.

86. Maxeler Technologies. https://www.maxeler.com/technology/dataflow-computing/. Accessed: 2022-05-13.

87. Afifi, S. M., GholamHosseini, H. and Poopak, S. Hardware implementations of SVM on FPGA: A state-of-the-art review of current practice. 2015.

88. Price, D. Pentium FDIV flaw-lessons learned. *IEEE Micro*, 1995. 15(2): 86–88.

89. Gama, J., Sebastiao, R. and Rodrigues, P. P. On evaluating stream learning algorithms. *Machine learning*, 2013. 90(3): 317–346.

90. Altera. Quartus II Handbook Version 12.1. 2012. URL `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/qts/archives/quartusii_handbook_121.pdf`.

91. Altera. User Guide Getting Started with Quartus II Simulation Using the ModelSim-Altera Software. 2014. URL `http://www.altera.com.my/literature/ug/ug_gs_msa_qii.pdf`.

92. Matlab. Matlab Primer. 2014. URL `http://in.mathworks.com/help/pdf_doc/matlab/getstart.pdf`.

93. Fan, R.-E., Chen, P.-H. and Lin, C.-J. Working set selection using second order information for training support vector machines. *Journal of machine learning research*, 2005. 6(Dec): 1889–1918.

94. Cortes, C. and Vapnik, V. Support-vector networks. *Machine learning*, 1995. 20(3): 273–297.

95. Boser, B. E., Guyon, I. M. and Vapnik, V. N. A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*. 1992. 144–152.

96. Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J. and Williamson, R. C. Estimating the support of a high-dimensional distribution. *Neural computation*, 2001. 13(7): 1443–1471.

97. Chang, C.-C. and Lin, C.-J. Training v-support vector classifiers: theory and algorithms. *Neural computation*, 2001. 13(9): 2119–2147.

98. Chang, C.-C. and Lin, C.-J. Training v-support vector regression: theory and algorithms. *Neural computation*, 2002. 14(8): 1959–1977.

99. Nash, W. J., Sellers, T. L., Talbot, S. R., Cawthorn, A. J. and Ford, W. B. The population biology of abalone (haliotis species) in Tasmania. I. Blacklip Abalone (h. rubra) from the north coast and islands of Bass Strait. *Sea Fisheries Division, Technical Report*, 1994. 48: p411.

100. libsvm_regression_data_set. `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html`. Accessed: 2018-09-30.

101. Freedman, D. A. Adjusting the 1990 census. *Science*, 1991. 252(5010): 1233–1236.

102. Delve, C.-A. D. Data for Evaluating Learning in Valid Experiments, 2005.

103. Sigillito, V. UCI Machine Learning Repository, 1989. URL `https://archive.ics.uci.edu/ml/datasets/ionosphere`.

104. Wolberg, W. H., Street, W. N. and Mangasarian, O. L. UCI Machine Learning Repository, 1995. URL `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)`.

LIST OF PUBLICATIONS

Indexed Journal (SCOPUS)

1. Sirkunan, J., Tang, J.W., Shaikh-Husin, N. and Marsono, M.N., 2019. A streaming multi-class support vector machine classification architecture for embedded systems. Indonesian Journal of Electrical Engineering and Computer Science, 16(3), pp.1286-1296.

Indexed conference proceedings

1. Sirkunan, J., Shaikh-Husin, N. and Marsono, M.N., 2019, May. Interleaved Incremental/Decremental Support Vector Machine for Embedded System. In 2019 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1-5). IEEE.
2. Sirkunan, J., Shaikh-Husin, N., Andromeda, T. and Marsono, M.N., 2017, September. Reconfigurable logic embedded architecture of support vector machine linear kernel. In 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI) (pp. 1-5). IEEE.