FPGA-BASED CASCADE SUPPORT VECTOR MACHINE WITH INTEGRATED TRAINING

A Thesis

by

YEN-JU LIN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Peng Li
Committee Members,	Gwan Choi
	Ricardo Gutierrez-Osuna
Head of Department,	Miroslav M. Begovic

August 2016

Major Subject: Computer Engineering

Copyright 2016 Yen-Ju Lin

ABSTRACT

Machine learning algorithms allow us to reason about and analyze large amounts of data. The support vector machine (SVM) is one popular learning algorithm, which has been applied to a broad range of applications. To this end, hardware-based SVM processors are very appealing due to their improved runtime and energy efficiency.

This research proposes an FPGA-based parallel support vector machine processor, which is capable of processing multi-dimensional data sets. The proposed FPGA SVM is based upon the cascade SVM algorithm, which is leveraged to allow efficient parallel processing of data on the FPGA platform, leading to significant processing efficiency.

DEDICATION

I would like to dedicate my thesis work to my family, for my parents and grandmother's selfless support and my brother's academic help since we were kids to now, which has made my graduate studies and accomplishment possible.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Li, my committee members, Dr. Choi, Dr. Gutierrez-Osuna, and my committee substitute Dr. Khatri, for their help and guidance through this research work.

I would also like to thank Qian Wang, Youjie Li and other peers from Dr. Li's research group, thanks to all for their help and time spent in discussions with me. Finally, I would like to thank all of my peers and friends, both present and past, for helping and supporting me throughout my graduate studies.

NOMENCLATURE

ANN	Artificial Neural Network
ASIC	Application-Specific Integrated Circuit
AXI	Advanced eXtensible Interface
BRAM	Block RAM
CF card	Compact Flash Card
FAT	File Allocation Table
FPGA	Field-Programmable Gate Array
GPU	Graphics Processing Units
KKT	Karush-Kuhn-Tucker
PCIe	Peripheral Component Interconnect Express
SMO	Sequential Minimal Optimization
SV	Support Vector
SVM	Support Vector Machine
System ACE	System Advanced Configuration Environment
UART	Universal Asynchronous Receiver/Transmitter

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

1. INTRODUCTION

1.1 Background and Motivation

In the world nowadays, human comes across with a lot of information every day. A biological research may have more than millions of data at a time. For example, in order to understand how computed tomography scans influence health, it requires the health records tracking for 11 millions of people [1]. With such amount of data, it shows its consequence and influence on human bodies. As another example, in meteorology studies, a lot of climate data are collected to understand the weather in recent years [2]. By long term climate changes and the current environment observation, future climate changes and possible dangers are predicted. These data show important information about the unknown in the world, while machine learning algorithms show a way to explore and make use of those data.

By machine learning algorithms, the relationship between those data and their outcomes are disclosed. Therefore, results of new samples can be predicted, while features and behaviors of original data can be understood. For example, by collecting common features of cancer patients can predict the tendency in cancer. With machine learning algorithms, those parts of gene, which play an important role on cancer development, can be distinguished from a huge amount of other irrelevant gene parts [3]. For another example, it is also applied in the radar image reading, while these radar images show where the oil spitted causing the environmental issues are [4]. Those radar images were observed and studied manually in the past, it took experience and heavy

work load to identify current situation in radar images. However, with machine learning technology, the oil spitted can be detected from radar images in an efficient way and at an early stage. Therefore people are able to take actions immediately to the situation and stop worsening the damage to the ocean.

Machine learning technology has these benefits in the research purpose; however, it requires a huge amount of time, from hours to weeks, and a lot of power to make use of these data in software implementations. To solve these issues and make machine learning algorithms more accessible, researchers have been dedicated to shorting the runtime and reducing power consumptions. For instance, it takes around 5 hours to train a model over a data set with 481,000 samples in a traditional machine learning algorithm. By improving the algorithm, the runtime may be reduced from 5 hours to less than 1 minute [5]. Apart from runtime speedup, some research has been done on power reduction by using different technology and designing power-aware algorithms [6].

On the other hand, a specific designed hardware is faster and consumes less power compared with same software logic performed on a general purpose CPU. Here are the reasons. Firstly, for the same logic, a hardware implementation is able to implement it in a parallel manner, where a parallel architecture is explored. Secondly, a hardware implementation does not need instruction memories to store computation instructions, which is necessary for the software. In the hardware, only those logic gates required by the design exist, but for software, it requires all the general logics implementation in a general purpose CPU.

To enjoy those benefits from machine learning algorithms and hardware implementations, a specific ASIC design of support vector machine algorithm (SVM) is developed in the previous research [7]. It is able to achieve five hundred times speedup and approximately twenty thousand times energy reduction compared with a software SVM algorithm. But due to the limited storage and the restricted design flow of the ASIC platform, only limited amount of data and those training models with two dimensions can easily be trained. If the structure needs to be modified for higher dimensions or improved, it requires a lot of time for modification and testing. Therefore, the main motivations and challenges for this thesis research are to maintain the benefits from the hardware design and a flexible version of hardware implementation.

1.2 Classification in Machine Learning

With machine learning algorithms, different kinds of training models can be trained and used, in order to get the specific feature in a model.

There are few kinds of machine learning algorithms in terms of the outputs relation with inputs in the training model, which are classification, regression, clustering, density estimation and dimension reduction [8]. With different kinds of training models, the corresponding learning algorithm is chosen. For example, a regression method predicts the output from a new sample in a continuous training model [9]. On the other hand, by clustering, it distinguishes different kinds unknown of data types for a given training model. The clustering method and regression can be applied in decease detection technology [10].

Apart from other methods, for the data set of a training model with known classifications, a new sample can be assigned to one of those classifications with classification algorithms. With the help of classification algorithms, classifications of unknown samples are predicted from the training model. There are different kinds of classification algorithms, and the linear classification is one of the commonly used classification algorithms. A linear classifier separates data according to the linear combination of samples from the training model.

From Figure 1.1, two linear classifications have been performed to separate different classifications of a training model. Because of the complexity of computation in linear combinations, it requires less time in the training process to reach the similar accuracy compared with a non-linear classifier. Though it is able to approximate a non-linear training model as a linear classification problem, it is not applicable for those training models which have complicated non-linear characteristic [11].

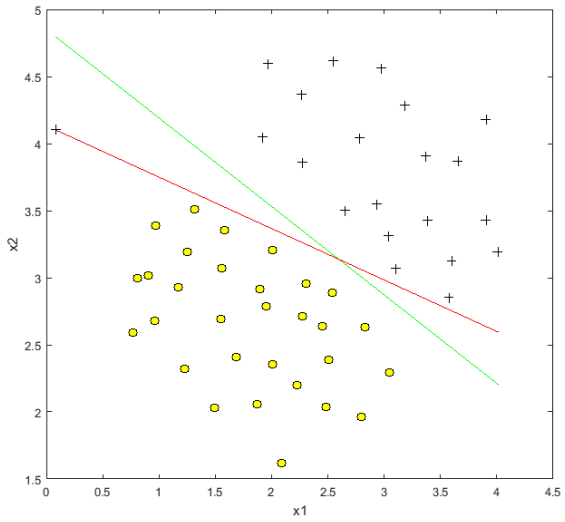

Figure 1.1 A demonstration of linear classification

For example, with linear classifiers, text categorization can be performed in a training model, which is able to separate text contents [12]. This is widely used in separating spam emails and regular emails for users in a huge amount of incoming emails.

On the other hand, artificial neural network (ANN) is also a widely used classification learning algorithm. It is inspired by the neural network of animals, in animals' brain, a series of connected neurons exchange messages by transmitting electrics signals. With those signals, animals are able to react to messages from outside world, including thinking, acting with emotions, and learning. By the concept of neurons, ANN constructs a large numbers of nodes, named neurons, connecting with one another. In each connection, it has a numerical number working as a weighed number. With those

nodes and weighed connections, it is able to adjust to the incoming unknown data, aiming to working as a biological neuron network [13]. For example, lung cancers are able to be identified via ANN [14]. By obtaining the image of lungs, it is possible to tell whether it is a healthy cell or a cancer cell by ANN.

With these classification algorithms in machine learning technology, it is possible to learn from those data, and explore the unknown of the world. Linear classifiers present a relative fast and easy way to distinguish a training model compared with non-linear classifiers. By the inspiration of biological neuron networks, ANN learns from training models. These algorithms contribute to all the aspects in the world, from separating spam emails in email boxes, to identifying cancer cells.

Finally, another widely used algorithm in classification is the support vector machine, which is the algorithm used in the research. More details would be addressed in the next section.

1.3 Overview of Support Vector Machine

Support vector machine, known as SVM, is one of the commonly used learning algorithms in classification and it predicts future samples from the training model [15]. SVM analyzes data for classification analysis; it distinguishes different features in a training model and predicts the result. That is, for instance, for the data set of a training model, it is able to assign a new sample into one of those different categories after the SVM training process. In this research, it is a parallel SVM algorithm implemented by the FPGA.

1.3.1 Applications

There are some typical uses for SVMs. For example, it separates a training model with a binary data set according to its classifications. If there is a new sample, it is able to predict the classification of that new sample. In Figure 1.2, it is a training model with two kinds of classifications, circle and plus sign. With SVM, whether circle or plus sign classification of a new sample can be predicted.

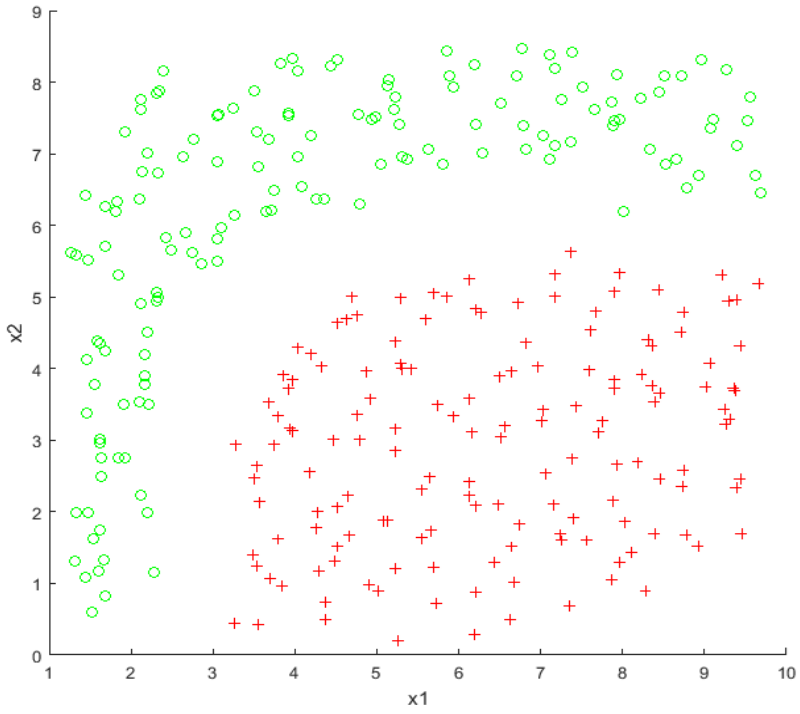

Figure 1.2 A training model with binary classification data set

On the other hand, for a training model which is not easily separable, SVM is still able to generate a suitable boundary for the training model according to the user's

need. By modifying the setup in the algorithm, the boundary is tuned in order to have a separating boundary that meets the requirement.

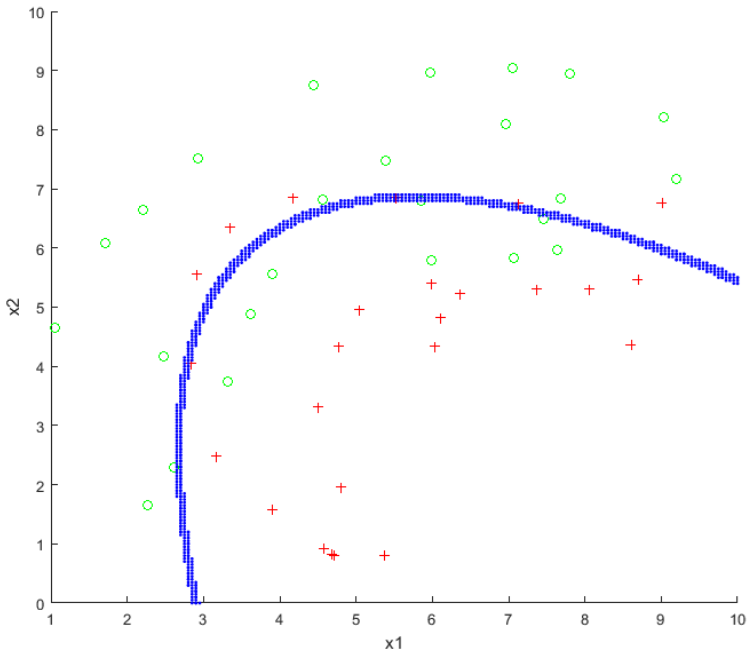

Figure 1.3 A difficult separating training model

In the Figure 1.3, these two categories are not as clearly divided in the Figure 1.2. However, it is still able to generate a boundary that separates most of the samples by SVM.

With the help of SVM, it is not only possible to separate linear training models as linear classifiers, but it is also possible to separate those training models in a more complex form, like Figure 1.3.

For example, by SVM, tissues are separated accurately according to the difference in normal and cancer tissues. Furthermore, with the help of SVM, it is also able to distinguish mistakenly labels tissues. Finally, it separates not only normal tissues and cancer ovarian tissues, but also those tissues from other part of the human body [15]. With more accurate ways proposed to distinguish cancers and diseases, it provides more precise and correct treatments compared with the past. Therefore more lives can be saved in the future as the technology developed.

Apart from medical use as the example above, SVM is also used in image retrieval [16]. With a big database of images, it is time-consuming to label and make description for each image manually. And it is also difficult for image searchers to be specific on picture features, while colors or sizes of the picture may not play an important role in terms of searching. For example, if an image of a rabbit is needed, the image of a rabbit may have different colors in the background or the rabbit itself has more than one color according to its species; on the other hand, the image may be a hand drawing rabbit uploaded by a scanner with low quality, it may also be a high quality picture taken by a professional camera. With SVM, it is able to create a boundary between targeted images and other images in the database. Therefore, it saves time and the work of labeling images manually, and it has a better outcome if images are found with a different classification way from labels. This method helps in the picture preservation and search.

From these applications of SVM, SVM plays an important role in current technology and human lives. Therefore to make SVM more efficient in data training,

researchers have dedicated their time and efforts to transfer it from the software platform to hardware implementations in order to enjoy benefits of the hardware characteristics.

1.3.2 Hardware Implementation

Because of time reduction from hardware implementations and benefits of SVM achieved in research works, some research has been proposed on hardware implementations of SVM.

For example, an analog VLSI design of a SVM algorithm has been proposed [17]. By implementing the kernel of a SVM algorithm, it provides a faster solution for the SVM algorithm. On the other hand, with the trade-off with power, it is also a power efficient solution. With a SVM algorithm implemented on chip, it is portable and flexible for users. In order to have a great power efficiency improvement, another design in the analog VLSI of SVM has been proposed [18]. Instead of a traditional MOS transistor dependence method, by adopting the margin propagation in design gives the analog VLSI implementation a better performance in power efficiency.

Apart from analog VLSI, some works in digital architecture are proposed, by implementing a SVM algorithm in other ways, different benefits are achieved. By implementing the sequential minimal optimization SVM in a digital VLSI, it provides the retraining flexibility to the implementation, noted that it is only applicable for a linear kernel [19]. In the future, it may be further developed as a system on a chip. With a system on a chip implemented SVM, it gives more opportunities and convenience for those using SVM algorithms.

Finally, the FPGA is another popular platform to implement SVM in hardware. By implementing SVM in the FPGA, it achieves a better precision and resolution compared with SVM in analog VLSI implementations [20]. Moreover, some SVM implementations on the FPGA focus on the acceleration of the classification process [21]. On the other hand, the graphics processing unit, GPU, is also another popular implementation choice, which has also been used in the SVM implementation [22]. By comparing the FPGA and the GPU in SVM implementation, different implementations have a better performance in different kinds of training models [23]. The FPGA implementation gave a better performance in chunking technology of SVM, while GPU is less limited by the dimension increment.

There have been a lot of researchers dedicated to hardware implementations on SVM with different methods, the speedup have been achieved from previous works. A lot of works have been focus on the internal training process acceleration; on the other hand, by speeding up the algorithm itself is also an approach to improve the training efficiency. This research is based on a previous work that focused on algorithm speedup [7].

1.4 Research Objective

From previous discussions, SVM is known as an effective way for data classifications. With the hardware implementation, it is also able to have a better performance in power and runtime.

In the previous studies of SVM hardware implementations, it has the benefits of time reduction and power efficiency by implementing the cascade SVM algorithm in hardware; however, there are certain limitations of it. It only trains certain dimensions of a training model, and it has limited data storage, which led to a difficulty of real world data training.

The goal of this research is as followed, first, expanding the possible data dimensions for training in a faster manner, in order to train with real world data. Second, increase the internal data storage to work with large training models. By migration of the design from the ASIC platform to the FPGA, the above goals are able to be achieved.

2. ALGORITHMS OF SUPPORT VECTOR MACHINE

2.1 Basic Support Vector Machine

The hardware implementation is based on the SVM learning algorithm as discussed in the previous section. In order to separate samples and predict the classifications of unknown samples, the goal of SVM is to find a separating hyperplane between those close samples with the largest distance to the hyperplane [24].

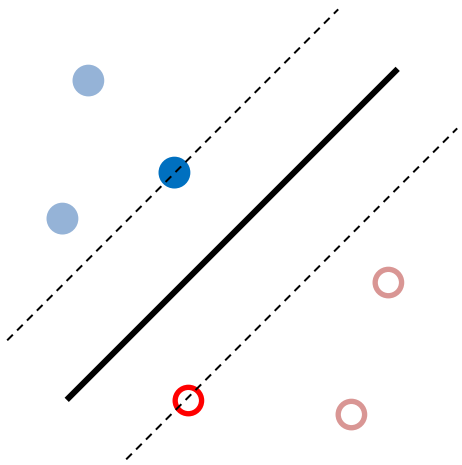

Figure 2.1 Hyperplane demonstration

In Figure 2.1, it is a demonstration of a hyperplane between two different classifications of samples. To have a precise separation from samples of different categories, the close samples to each other in different categories are chosen as the

starting points and form a boundary of their categories. The hyperplane is between those boundaries, in the Figure 2.1, the hyperplane lies between the boundaries of these two different categories, namely circle and ring. Therefore, only some of samples contribute to the hyperplane formulation, which are called support vectors, SVs.

Firstly, a sample and its classification are shown in below:

$$(\overrightarrow{x_i}, y_i), \qquad y_i = \{+1, -1\}, i = 1,2, \dots N$$

where $\overrightarrow{x_i}$ is the input vector, and y_i is the corresponding classification.

By assuming a mapping function $\phi(x)$ between samples and classifications, the relationship between an input sample and its classification can be written as these two equations:

$$\begin{cases} y = +1, w \cdot \phi(x) + b \geq 1 \\ y = -1, w \cdot \phi(x) + b \leq -1 \end{cases}$$

where *w* is the normal toward the separating plane.

To find the hyperplane, the relation between $\overrightarrow{x_i}$ and y_i is obtained as following:

$$f = y_i(w \cdot \phi(x_i) + b)$$

where b is the bias. And the distance from close samples to the hyperplane, named margin, is $2/\|w\|$.

As a result, the optimization problem is:

$$min_{w,b} \frac{\|w\|^2}{2}$$

which is subject to $y_i(w \cdot \phi(x_i) + b) = 1$.

However, not all training models have a strictly separated hyperplane. Therefore a soft-margin SVM is proposed in terms of the trade-off between the margin and errors. For the tolerance of training errors in this research, a soft-margin SVM is used with the optimization problem as below:

$$min_{w,\xi,b} \frac{\|w\|^2}{2} + C \sum_{i=1}^{N} \xi_i$$

The additional ξ_i in the new optimization problem is the slack variable, which gives the flexibility for samples to violate the hyperplane. C is a constant controlling the trade-off between margin accuracy and error tolerance. Figure 2.2 demonstrates the error tolerance of two different categories and its hyperplane. The SVs are all the samples inside the dash lines, which makes the new optimization problem is now subject to

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

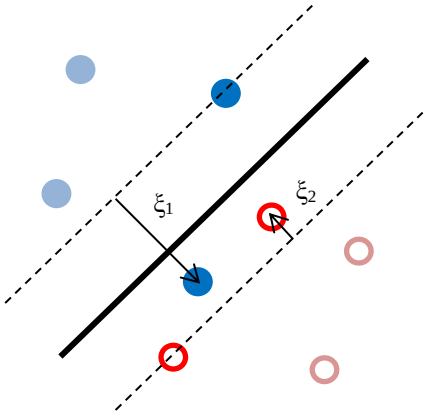

Figure 2.2 Soft-margin SVM with SVs in darker shades

After the optimization problem is obtained, its Lagrangian problem is shown as below:

$$Min\ L_p(w,b,\alpha_i) = \frac{\|w\|^2}{2} - \sum_{i=1}^{N} \alpha_i(y_i(w \cdot \phi(x_i) + b)\text{-}1))$$

$$s.t.\ \alpha_i \geq 0$$

and the dual problem:

$$\text{Max} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i\, \alpha_j y_i y_j k(x_i, x_j)$$

$$s.t. \quad C \geq \alpha_i \geq 0 \ \text{ and } \ \sum_{j=1}^{N} \alpha_i\, y_i = 0$$

From the dual problem, a kernel function $k(x_i, x_j)$ is introduced from the mapping function. In order to find the separating plane and solve the problem, the kernel method is introduced here.

2.1.1 Kernel Method

The kernel method provides a measurement between two similar vectors and ensures the optimization problem is convex and unique. It gives a better way to find the relationship in different samples of the training model, and it is mostly used in SVM training [25].

To decide the decision boundary of a classification training model, there are many different kinds of kernel methods. For example, a linear kernel is widely used in SVM. It is able to determine a less complicated model and generate a boundary. For more complex models or higher accuracy requirements, the Gaussian kernel is a popular kernel, which is also used in this research. Apart from these two kernels, there are other kernels also used in SVM, including Fisher kernel and Graph kernel.

A general kernel is defined as $\mathrm{k}(x_i, x_j) = \phi(\overrightarrow{x_i}) \cdot \phi(\overrightarrow{x_j})$, it gives the relation of samples between each other and their mapping function. The Gaussian kernel is used in this research as $\mathrm{k}(x_i, x_j) = \exp(-\gamma \left\| \overrightarrow{x_i} - \overrightarrow{x_j} \right\|^2)$. In the Gaussian kernel computation, an exponential computation is involved. Unlike linear kernels or polynomial kernels, the Gaussian kernel is not easy and straightforward to be implemented by adders and multipliers.

For hardware implementations, a hardware friendly kernel has been proposed, which remains good performance in many cases [26]. It is an analog circuit implementation proposed in substitute of the Gaussian kernel. However, the Gaussian

kernel is still used in this research; the detailed implementation is provided in section 3.1.1.

With the help of the kernel method, it is possible to solve the Lagrangian problem and its dual problem of SVM in section 2.1. By substituting the kernel function $\mathrm{k}(x_i, x_j)$ with a Gaussian kernel in the Lagrangian dual problem, the problem is rewritten as below:

$$\mathrm{Max} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \, \alpha_j y_i y_j \exp(-\gamma \left\| \overrightarrow{x_i} - \overrightarrow{x_j} \right\|^2)$$

$$s.t.\ C \geq \alpha_i \geq 0 \ \text{ and } \ \sum_{j=1}^{N} \alpha_i \, y_i = 0$$

By finding α_i, it would lead to SVs, which would be discussed in the next section.

2.1.2 Classification Checking

In SVM learning algorithms, the sequential minimal optimization (SMO) is one of the popular algorithms to solve the optimization problems in SVMs [27]. SMO algorithm is able to solve the above function in an iterative manner. By computing multipliers in pair, it is able to solve the Lagrangian problem efficiently. However, in this research, the gradient ascent method is chosen to solve the above optimization problem, which is more hardware friendly [26]. Firstly, the above problem can be checked by Karush-Kuhn-Tucker (KKT) conditions. With the help of KKT conditions, the α values are able to be determined.

The KKT conditions indicate that for a solution in the nonlinear problem to be optimal, its inequality constraints are no longer existed [28]. Consider a nonlinear problem:

$$\text{Max } f(x)$$

$$\text{s.t. } g_i(x) \leq 0, h_j(x) = 0$$

If it has an optimal solution, which must satisfy the following conditions:

$$\nabla f(x^*) = \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{l} \lambda_j \nabla h_j(x^*)$$

$$m=0$$

For minimum problems, the left side of the first condition is $-\nabla f(x^*)$, instead of $\nabla f(x^*)$.

By KKT conditions, the optimal solutions can be found for nonlinear problems. For example, convex optimization methods are very common in communications or signal processing. To find the optimal, KKT conditions are used for these methods [29].

For here, to satisfy KKT conditions, it requires the product of Lagrangian parameters and their constraints to be zero, which is

$$\alpha_i(y_i(w^T \cdot \phi(\overrightarrow{x_i}) + b) - 1 + \xi_i) = 0, \ \ \beta_i \xi_i = 0.$$

β_i is the Lagrangian variable for ξ_i. Therefore KKT conditions of the soft-margin SVM are shown in below:

$$\begin{cases} \alpha_i = 0 & y_i(w^T \cdot \phi(\overrightarrow{x_i}) + b) \geq 1 \\ 0 < \alpha_i < C & y_i(w^T \cdot \phi(\overrightarrow{x_i}) + b) = 1 \\ \alpha_i = C & y_i(w^T \cdot \phi(\overrightarrow{x_i}) + b) \leq 1 \end{cases}$$

With the relation between the hyperplane and samples, *w* value is written as $w = \sum_{i=1}^{N} \alpha_i y_i \, \phi(\overrightarrow{x_i})$ and b is the bias mentioned before. From KKT conditions discussed above, if $\alpha_i = 0$, it is not a SV, if $\alpha_i = C$ it is a SV with error tolerance, and finally if $0 < \alpha_i < C$, the sample is right on the hyperplane.

From the above process, if α_i is determined for every sample, the hyperplane can be found and classifications of future samples are able to be predicted. For a new sample $\vec{x}$, by computing its relation with SVs:

$$\mathrm{f}(\vec{x}) = \sum_{i=1}^{N_{SV}} \alpha_{SV} \, y_{SV} K(\vec{x}, \overrightarrow{x_{SV}})$$

if $\mathrm{f}(\vec{x}) > 0$, its classification is determined as $+1$, else it is -1.

Therefore, in the hardware implementation, α value is the final result after the training process finishes. The whole process of training is based on iterations of finding SVs and eliminates non-SVs; therefore it may take a long time of training if the training model has a huge amount of data.

2.2 Cascade Support Vector Machine

From the basic SVM discussed in the previous section, it shows that only SVs contribute to the hyperplane, most of the samples in the data set of a training model are non-SVs. Therefore, in the cascade SVM, it eliminates non-SVs at the early stage, which speeds up the whole training process [30].

The cascade SVM is implemented in the following manner. First, the data set of a training model is spit into smaller subsets, and those subsets are trained simultaneously. After each subset is trained, SVs from those subsets are fed to SVM units in the next layer for a combined training. This progress is repeated until all subsets are combined and trained. After it reaches the last training process, training results will be fed back for KKT conditions checking. To achieve the design, multiple SVM units are needed, which are placed as a hierarchical structure as Figure 2.3.

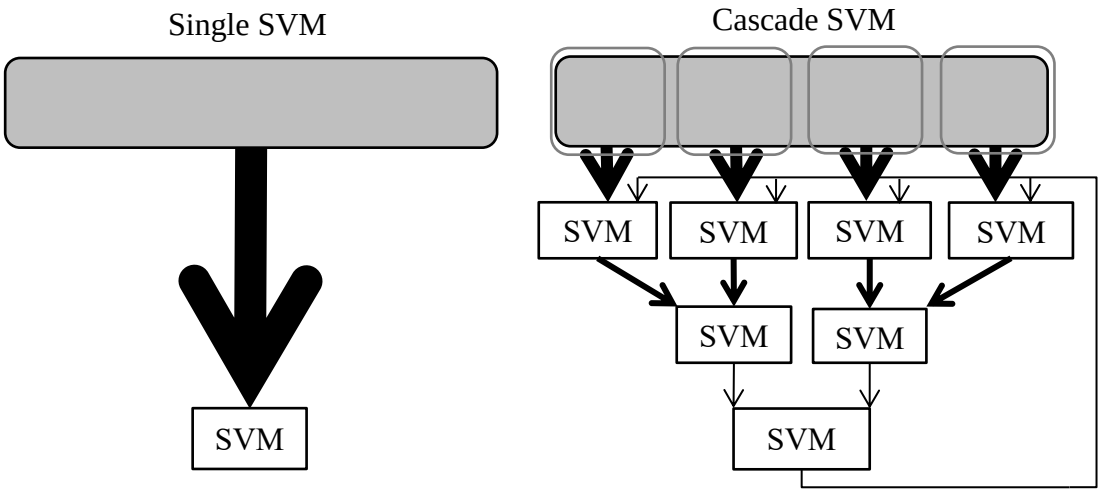

Figure 2.3 Cascade SVM versus single SVM

For example, by separating a large data set into four smaller subsets, which is shown in Figure 2.3, those four data sets are trained at the same time. After each training process, the next layer has less samples to train compared with the previous layer, due to

the elimination at the previous layer. Also, since the training process is in quadratic, even though it trains three times for three layers, it is still faster than training all samples at one time.

There are some implementations of the cascade SVM showed a great speedup compared with a basic SVM. For instance, with the implementation of the cascade SVM, the training time for a training model of a data set with 27,000 samples can be reduced from 70 hours to 3 minutes [31]. That is, it has a 1400 times speedup compared with the previous training method.

Though from the aspect of structure, more SVM units are required in order to achieve the parallel computation and multiple layers training compared with a basic SVM, the significance in speedup shows the great benefit of using the cascade SVM instead of a basic SVM.

3. HARDWARE IMPLEMENTATIONS

3.1 Support Vector Machine Structure

The main hardware architecture is based on the previous ASIC design [7]. In this section, with the original structure as a starting point, other versions implemented in the FPGA are proposed.

3.1.1 SVM Unit Structure

For a single and basic SVM discussed as before, apart from the internal computation module, it has an address generator for memory access and a lookup table for exponential computations.

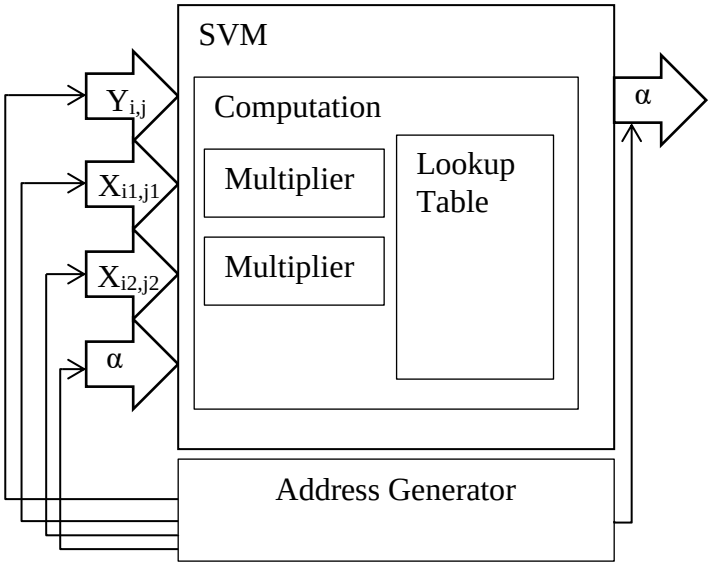

Figure 3.1 A SVM unit structure

The Figure 3.1 shows the overall structure of a SVM unit design. Firstly, a sample is stored in sequence as followed: its classification Y, its first dimension data X_1, its second dimension data X_2, and the corresponded α value which is initialized as 0. According to the address generator, the SVM unit has these four values of a sample. To compute the corresponding α value, other samples are also entered in a same manner. After samples are entered, the computation proceeds to the further computation.

A fixed multiplier is also used in the computation. The fixed multiplier is designed specifically according to the data storage in Figure 3.2. Data is stored as a 32 bits number in two's complement, which is defined as a fixed point number with 16 bits before the fixed point in Figure 3.2, and 16 bits after the fixed point. That is, the minimal number of a positive number is 2^{-16}. The storage demonstration is shown as Figure 3.2, which shows the precision and implementation of the fixed point data storage design.

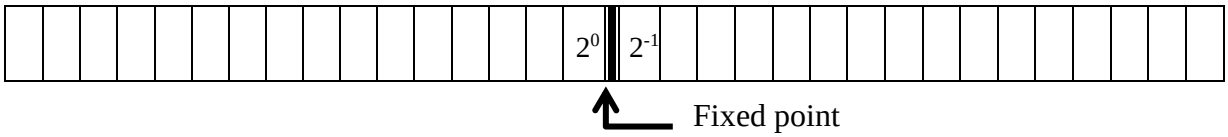

Figure 3.2 Data storage demonstration

The kernel method is an important part in SVMs as discussed in the previous section. Although as mentioned in 2.1.1, some specific kernels designed for hardware have been proposed, to ensure the performance in a SVM unit, the Gaussian kernel is used here. To implement the exponential function in Gaussian kernel computation, a lookup table is added. After all the dimensions of the sample and its corresponding

compared sample are computed, the value is fed into the lookup table module; the result of Gaussian kernel is further obtained.

A sample is compared with all other samples to get α value, α is stored in the data storage for future training process. In the end of each training process for one sample, α value is compared with the previous α value of the last training process of the same sample. Because α would eventually converge into a same value, by comparing the current value and the one from previous training process, the convergence of the training process is determined.

After the training process converged and the data set is fully trained, α values are verified in a Matlab function for the verification.

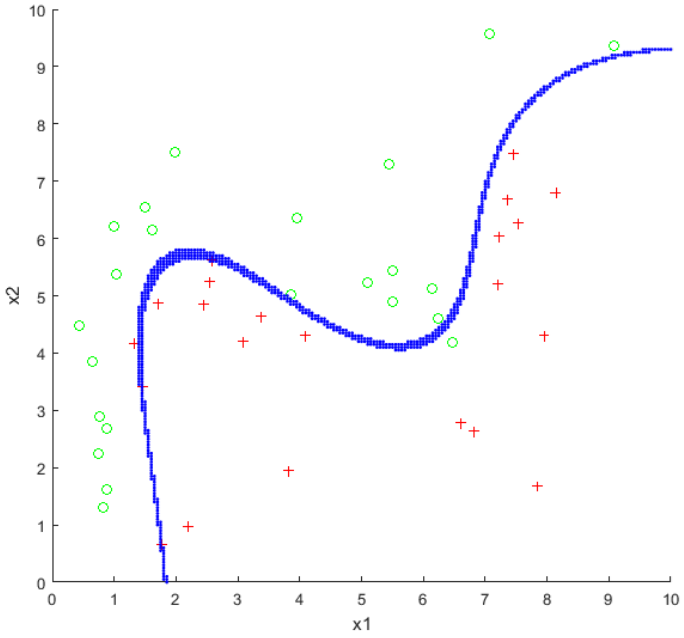

Figure 3.3 A two dimension training model and its result

With the data set of a two dimensions training model and its α values, the training result is shown in graphic as Figure 3.3. On the other hand, for the higher dimensions, a testing data set is required for the final training result.

3.1.2 Cascade SVM Unit Setup

Apart from the basic SVM, a cascade SVM is also implemented in the previous ASIC design, it has a significant speedup and energy reduction compared with a basic SVM. The cascade SVM module is developed based on the cascade SVM discussed in previous section; however, there are some changes in the structure thanks to the characteristic of hardware implementation.

Due to the benefit of hardware implementations and the training process of the cascade SVM, the number of all the SVM units is the same as the maximum number of SVM subsets in the first layer of a cascade SVM structure.

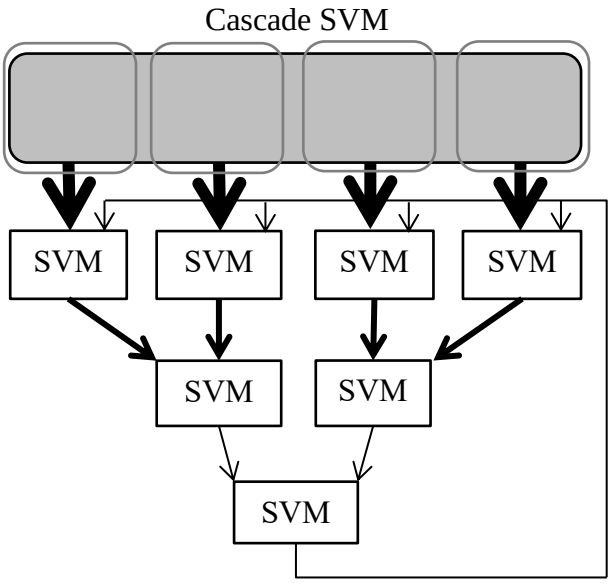

Figure 3.4 A cascade SVM structure

Figure 3.4 shows that every layer operates at a different time, and the next layer always has less SVM units compared with the previous layer. If a SVM unit is reusable, the numbers of SVM units required for a cascade SVM can be reduced, which saves the device usage in hardware.

With a reusable SVM unit developed, the previous structure of Figure 3.4 is modified as Figure 3.5. At first, four SVM units train the data sets from the corresponding data storage of each SVM unit. After all of the units are trained, only two SVM units are activated again, they not only read SVs from its own data storage, but they also read from the data storage from their combined partner. In the end, only one SVM unit is activated, and it trains all SVs from four data storage.

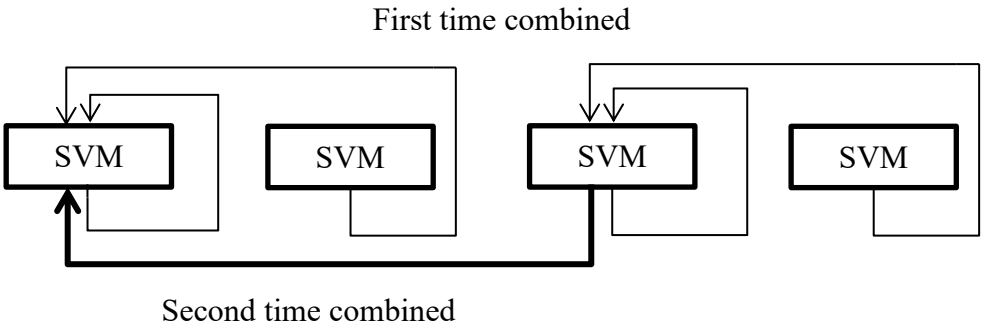

Figure 3.5 A reusable cascade SVM unit structure

As a result, a cascade SVM has the following structure: reusable SVM units, address mapping units, multilayer system bus and processing configuration.

In the cascade SVM implementation, after the address is generated from the address generator in a SVM unit in Figure 3.1, it is fed into a memory manage unit, which translates the address according to the current layer. Figure 3.6 shows how memory management units are used in the last layer.

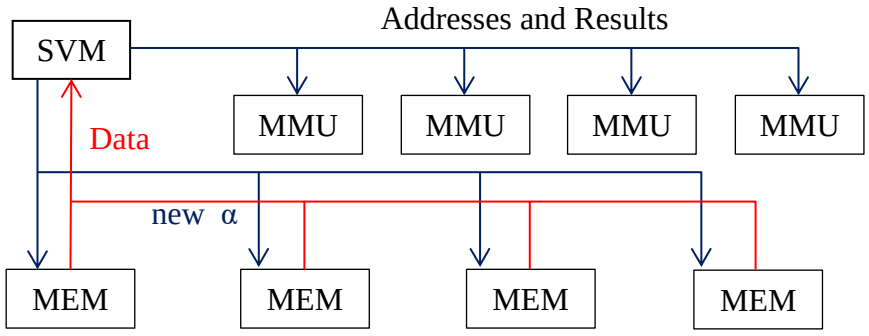

Figure 3.6 Memory management units implementation

The indexes of SVs are stored in the memory manage unit, which are produced from reusable SVM units after convergence. On the other hand, every reusable SVM unit in each layer has its own memory manage unit, it stores indexes after each convergence and provides the correct address for the next training.

To separate every layer, it also has a multilayer system bus. The multilayer system bus records the convergences from every reusable SVM unit. Aside from that, the processing configuration feeds the corresponding sample into activated SVM units.

With the cascade SVM design discussed as above, it provides a faster training process compared with the basic SVM in 3.1.1. These two structures serve as the base of this research and by implementing these two structures in the FPGA platform, the issues on the ASIC platform can be solved.

3.2 UART System Implementation

Due to limitations of the ASIC platform, it is difficult to expand the dimensions of a training model in a faster manner and the data storage limitation constraints the possibility of large models training.

In order to solve the limitations of the original implementation, the design of the SVM is migrated to the FPGA. As a result, a UART system implementation is proposed in the beginning.

3.2.1 Hardware Structure Overview

First, to implement the design in the FPGA, a UART communication is added between the host computer and the FPGA for communication purpose, and the main structure is modified for synthesizability difference in the FPGA. A host computer is the user interface to communicate with the FPGA board, it is able to load the data for training and receive the training result. On the other hand, the memory structure is changed from SRAM structure to Block RAM.

The universal asynchronous receiver/transmitter system, which is known as the UART system, is a common communication method between computer hardware devices for data transfer [32]. The UART system transfers data by bits in sequence according to the corresponding speed of bit transition for the receiver and the transmitter, which is called baud rate. With the same speed between the transmitter and the receiver, it is able to interpret the signal correctly.

Start bit	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8	Bit 9	Stop bit 1	Stop bit 2

Figure 3.7 UART data frame

From Figure 3.7, the data frame of the UART transition process is shown, noted that it only transfers 8 bits data at a time. Because the data storage in this design is a 32

bits fixed point format, it requires four transmission processes to transmit one sample from the FPGA board to the host computer, the user interface, with UART communication. On top of that, for this design, its baud rate is set as 115200 Bd, which indicates the transfer speed of each bit. On the other hand, peripheral component interconnect express, PCIe, is also a common transmission method with a high transmission speed, which is 2.5 gigabits per second. However, it requires more hardware device and setup requirement. The UART communication only requires 43 slices of registers, while the PCIe requires 673 slices of registers, which would increase 64.4% of the device utilization.

Aside from the UART implementation, the data storage of the FPGA implementation is the Block RAM (BRAM) storage [33]. The BRAM is a configurable memory module provided by Xilinx FPGA. It is memory storage with its data length and storage size modifiable by the user.

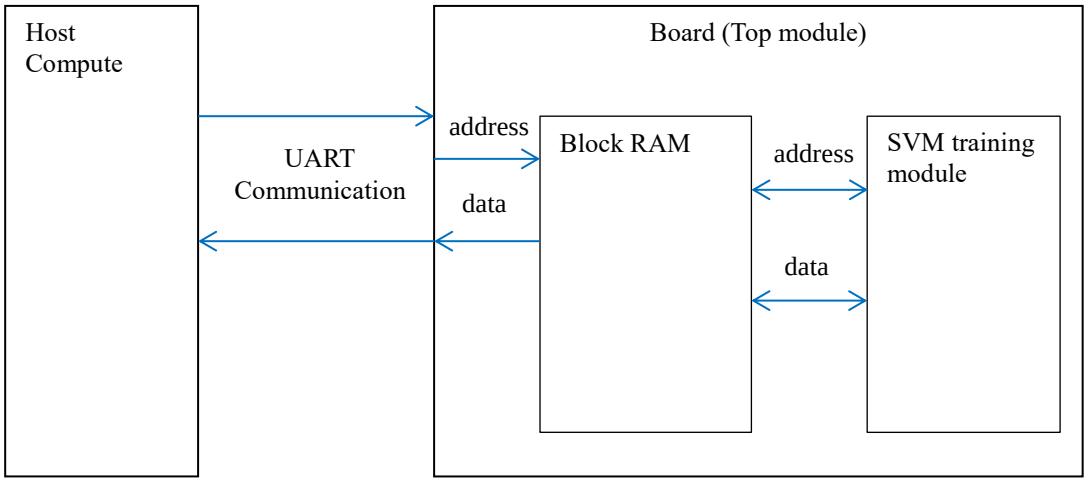

Figure 3.8 Hardware structure in the FPGA with UART

The hardware structure is shown as Figure 3.8. Firstly, a data set of the training model is loaded in the BRAM, which is connected to the SVM training module. The amount of BRAMs is same as the number of SVM units, because every SVM unit has its own memory storage. The SVM training module is discussed in 3.1, both basic and cascade SVM structures can be served as the SVM training module here.

After the SVM training module reports the convergence of the training process, the top module disconnects the BRAM from SVM training module, and connects it to the UART communication. That is, it reconnects the address input and data output from the SVM module to the UART communication. The receiver in the UART communication is connected with the address port of a BRAM, while the transmitter is connected to the data output port of a BRAM. Therefore the final training results can be read from the FPGA to the host computer.

After opening the serial port, it is able to send the address in four eight bits unsigned integers according to the setup of UART, which is shown in Figure 3.7, and saves the corresponding data. The receiving data is also interpreted from four eight bits unsigned integers to its correct value, which is saved for the further verification.

In this research, it is implemented in the Xilinx Virtex-6 FPGA ML605 evaluation board. Upon this point with the basic structure of SVM, the device utilization of a cascade SVM is around 1% of the FPGA, which gives flexibilities of expanding the original design. On the other hand, because of that a basic SVM only has one SVM unit; the device utilization is smaller than a cascade SVM. For a two cores cascade SVM design, it requires 2187 slices registers while the total device has 301,440 slices registers.

More dimensions can be introduced for training. As a result, the two dimensions design is expended to an eight dimensions design.

3.2.2 Multiple Dimensional Implementations

In order to expend dimensions, more registers are added as data storage and multipliers are parallelized for different dimensions data computation, higher dimensional data are able to be read in the training process, which is shown in Figure 3.9; additional parts are highlighted compared with the original design in Figure 3.1.By parallelizing the whole design makes training the training models with different dimensions in one design possible as long as the dimensions are less than the maximum dimensions. By setting the unused dimensions inputs as 0, it trains data sets without any difference with a design of the exact dimensions setup.

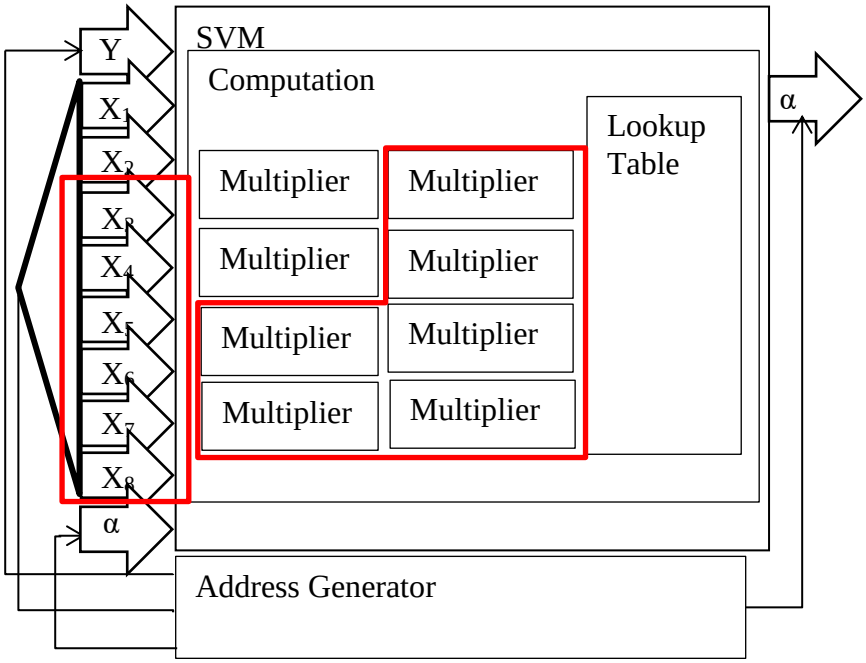

Figure 3.9 An 8-dimension SVM unit.

The shape marks for the additional dimensions registers and multipliers compared with a two dimensions SVM.

By replacing the SVM module in Figure 3.8 with a multiple dimensional SVM unit in Figure 3.9, data sets over two dimensions is able to be trained in the FPGA. Figure 3.9 shows that it still shares a similar structure with a two dimensions SVM unit. To understand the overall device utilization, the distribution of a two dimensions SVM unit implementation for the device utilization is shown as Figure 3.10. Most parts of the device are contributed to the internal computation, while in the computation most of the device is used in the multipliers, it requires 650 slices registers, the lookup table for the

exponential control requires 110 slices registers, and the address generator requires 108 registers.

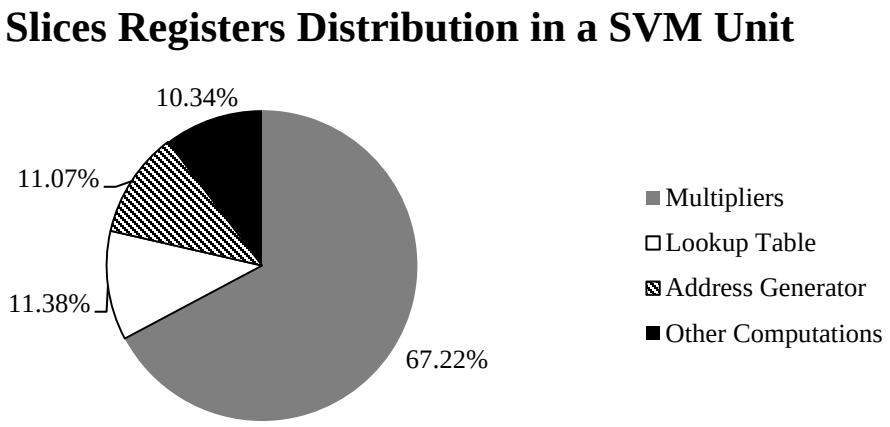

Figure 3.10 Slices registers of distribution for two dimensions SVM unit

The Figure 3.11 shows the area break down of the top module control, UART communication and the SVM module. Most parts of the device are used for the SVM module.

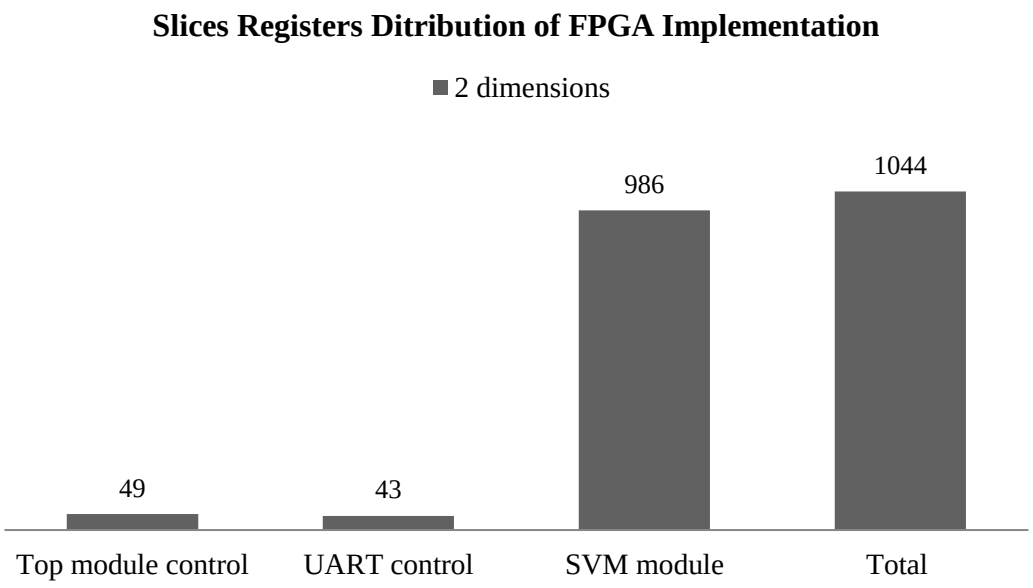

Figure 3.11 Slices registers distribution a two dimensions basic SVM FPGA implementation

From Figure 3.12, it shows the trend of device utilization growth as the dimensions increase. The higher dimensions it has, the more device it uses.

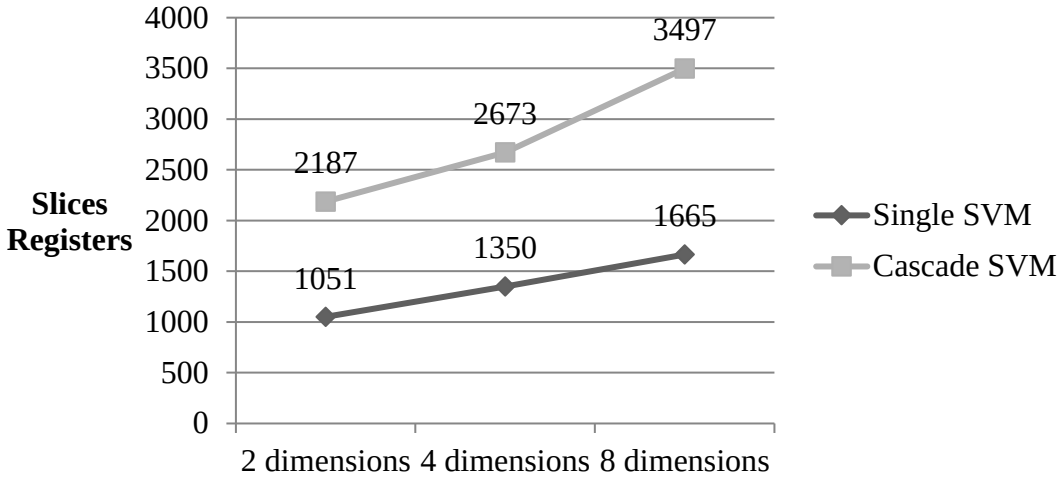

Figure 3.12 Device utilization in different dimensions

Though it is able to train multiple dimensional training models with the FPGA implementation, due to the UART communication between the host computer and the FPGA, the limitation of huge data sets training still exists. The process of data training itself has been speeded up, while the transfer speed of UART is slow for large data sets training.

3.3 Memory Card System Implementation

To solve the transfer speed issue, a new system design is introduced as followed. A 2G memory card (the compact flash card, CF card) is introduced into the design; therefore it is able to obtain the data in a faster manner and increase the size of training models that it is able to train.

3.3.1 Hardware Structure Overview

In order to manipulate a CF card in an easy manner, the Microblaze system is introduced. A Microblaze system is a soft-core microprocessor for Xilinx FPGA, it is able to control and interact with those features embedded in the FPGA according to the user's setup, including a CF card control [34]. The system bus between the Microblaze system and other functions is the Advanced eXtensible Interface, AXI [35]. It is an on-chip connection between functional blocks in the embedded system.

With the help of a Microblaze system, it is possible to interact with a CF card with straightforward instructions in C, compared with the implementation in Verilog of complex timing setup and signal exchange.

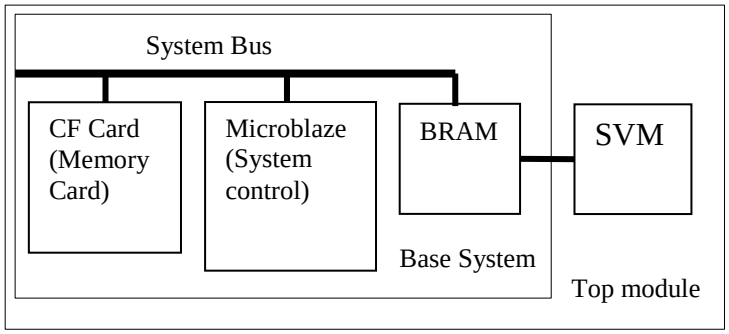

Figure 3.13 Hardware structure in the FPGA with the memory card

In Figure 3.13, a Microblaze system is used, which is connected with a CF card control and a BRAM. The BRAM here, compared with the previous design, is working

as a buffer rather than the data storage. In this design, the BRAM is a dual-port BRAM that one port connects to the internal system bus, which is accessed by the Microblaze system; the other port is set as an external port and connects the SVM module. The SVM module is separated from the base system in order to have a different clock from the Micorblaze; therefore its clock will not interact with other system structures in the base system.

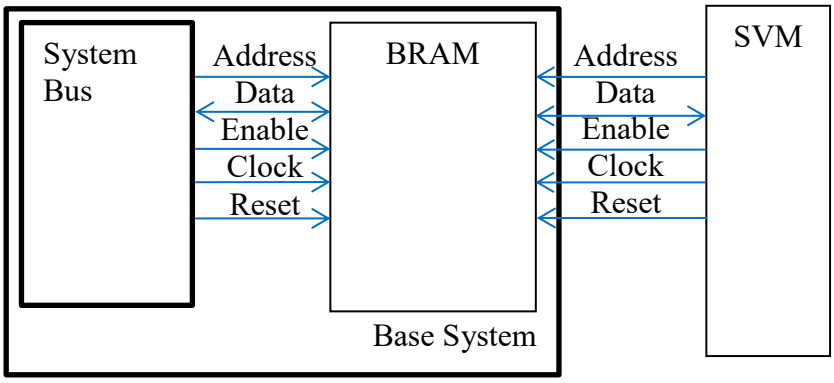

Figure 3.14 A BRAM buffer

The Figure 3.14 shows an insight of the BRAM buffer. Because the SVM unit is indirectly connected with the base system, the start command and the convergence signal are also stored in the BRAM buffer. Therefore the data storage is modified compared with the previous design.

Before

Address	Data
0	Y_1
1	$X_{1,1}$
:	:
8	$X_{1,8}$
9	Y_2
10	$X_{2,1}$
:	:
9×N	α_1
9×N +1	α_2

After

Address	Data
0	*
1	Y_1
:	$X_{1,1}$
8	:
9	$X_{1,8}$
10	Y_2
:	$X_{2,1}$
9×N	:
9×N +1	α_1

Figure 3.15 Data storage change

The * place stores the communication signal, including numbers of data, start and reset signal and converge signal.

From Figure 3.15, the left side represents the previous data storage order. The data of a sample is stored as the following way, its classification and rest of the dimensions data. After the whole data set is stored, α values are stored as the same order with its corresponding sample. For the right side, to communicate with the base system, the data set for training and its training results are moved for one storage space. With the spare space, the number of whole data set is sent first, and the convergence signal is read after the SVM module finishes training. On the other hand, if it is needed to start the

training process again with a new data set, the spare place is also used to give a restart signal to start a new training process.

On the other hand, if it is a cascade SVM structure, the BRAM buffer is multiplied according to the cores of the cascade SVM. That is, if it is a two cores cascade SVM, it would have two BRAMs in the system; each core has its own buffer. The data set is firstly stored in the CF card, and it is written into the BRAM buffer via the Microblaze.

After a data set is trained by the external SVM module, it is read from the buffer and written into the CF card again. If the intended data set is bigger than the size of the buffer, the Microblaze system divides the original data set, trains certain data and saves those outcomes first.

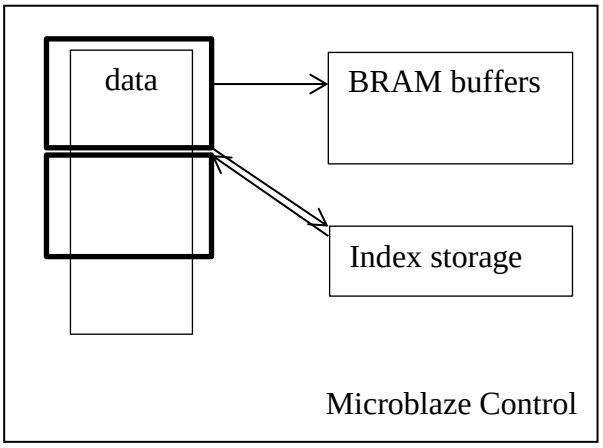

Figure 3.16 Data division in the Microblaze control

In Figure 3.16, the original data set is divided in the corresponding size of BRAMs, and the index of a dividing point is saved for the next division. After the data set is converged, the Microblaze control scans SVs from first round of training and write to the buffer again, working as the cascade SVM. As a result, it is capable of training data sets with the size constraint from a CF card instead of the size of BRAMs. However, the dividing point is chosen in order regardless of the classifications, the subset may become difficult to train compared with the original data set.

Though the memory card implementation provides the benefit of training a large data set with bigger size than the BRAM storage, in order to save not only indexes of separating points shown in Figure 3.16 but it also has to save all the indexes of SVs from the previous training results, it requires additional storage to record indexes. Apart from that, because this process is inside the Microblaze system and implemented in software, it would reduce the execution efficiency.

Therefore to save as more time as possible, one way is to speedup the reading process from a CF card to BRAMs. A subset is read firstly and the computation process is started. The control system loads the data to another BRAM buffer during the training process, which works as a dual buffers system. After those data are trained, SVM is able to start another training process immediately without data loading. For the cascade SVM, in order to have dual buffers, each core has two BRAM buffers. Therefore, for a two cores cascade SVM, it has four BRAMs in the design instead of two.

In Figure 3.17, it depicts the overall training process; the data set is stored in a CF card from the host computer, and it is read from a CF card after training. The data set

is stored in a text file and all of the samples are in a 32 bit binary form. However, for the values in the training result, they are stored as integer numbers.

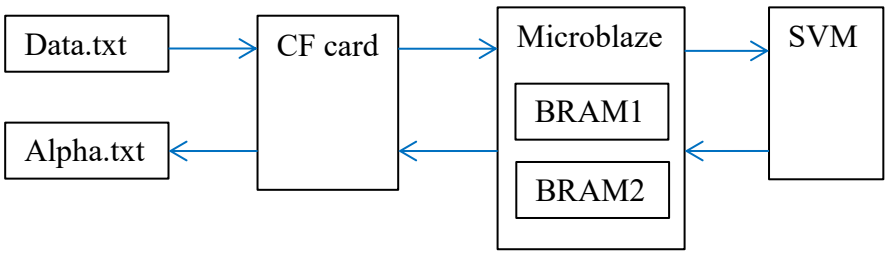

Figure 3.17 Data path

In this system the data transfer speed is no longer limited to UART system speed; it only needs to read a CF card from the computer to get training results. Furthermore, the data storage is no longer limited to the FPGA itself, it depends on the CF card memory size that user chooses.

3.3.2 Experimental Setup of the FPGA

In this section, a detailed FPGA setup is addressed for further results studies in section 4.3, which has experimental results for real world data sets.

In the setup of Microblaze system, apart from the Microblaze core and its cache, the system advanced configuration environment, system ACE, is attached to the system bus for a CF card control. The local memory size is set according to the Microblaze

system control implementation. And BRAMs are added, the number of BRAMs is determined by the SVM module used.

After the Microblaze system is set up and generated, the SVM module is included in the top module of the base system. The SVM module connects to the BRAM buffers, which is connected to the external ports from BRAMs. While everything is connected, the design is exported to the software develop kit.

For the control system setup inside the Microblaze, a xilkernel is chosen. It provides an easy and straightforward manner to control a CF card. By choosing the xilfatfs configuration, the data control is implemented inside the Microblaze system.

Xilkernel is a small and robust kernel in the Xilinx FPGA, it is highly customized and used for the higher level implementations [36]. Xilfatfs is one of the features provided by xilkernel. It provides read and write functions to files stored on a CF card. It supports the file systems of file allocation tables, FAT, from FAT12, FAT16, to FAT32. With the help of xilkernel and xilfatfs, the design is implemented in the Microblaze system.

Before the training process, the maximum training size has to be assigned by the user according to the size of BRAMs set up in the previous process. And in the base system, a stopping point is included to provide a checking method to avoid the data set is impossible to converge caused by the data separation of Microblaze controller, it stops the current training process and move to the next data set. Because the Microblaze controller divides the data set according to the order stored in the memory card without checking classifications, not all subsets would be as easy to train as the original data set.

3.4 Memory Improvement and Comparison

In the previous ASIC implementation, the total cache size of whole design is 8kB, and it is divided to smaller private caches for the cascade SVM units. On the other hand, with a large number of built-in BRAMs in the FPGA, the storage of data grows to 1872 kB. As a result, for the FPGA version with UART communication, it is able to train a training model with a relative larger data set compared with the ASIC version. And with the CF card version, the data storage grows to 2 GB, which has a significant growth in the data amount of training.

However, if more dimensions data sets need to be handled, the storage will be smaller. For an N dimensions sample, it requires N+2 32 bits storage. Because not only does it have to store its N dimensions data, it also needs to store its classifications and α values. The detail storage method is explained in Figure 3.14. As a result, the relation between data size and dimensions is provided in Figure 3.18.

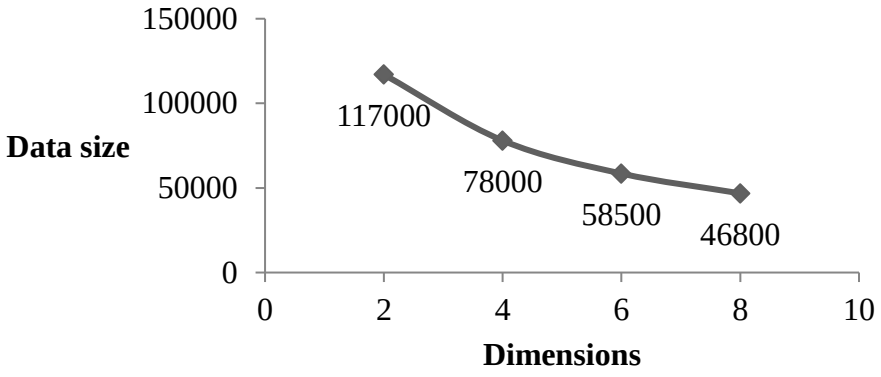

Figure 3.18 Possible data size in the built-in BRAMs

Though this is inevitable, with a 2 GB CF card involved, it is still able to train a large amount of data because the overall storage is about a thousand times larger than the size of internal BRAMs with the help of Microblaze. In the chosen 2GB CF card in this design, it requires 56.7 MB of storage for system files storage in order to become accessible for the Microblaze system; however, it is relatively small compared with 2 GB, which only occupied about 2.84% of the storage.

In this section, two hardware implementations on the FPGA are illustrated, and the implementation of both basic and cascade SVM units are explained. With the implementation of these structures, the following experiments in the next sections are conducted, which provides the insight of these implementations.

4. EXPERIMENTAL RESULTS

The SVM structure is designed in Verilog, and then it is generated by Xilinx Platform Studio. Afterward, the design is implemented through translating, mapping, placing and routing, and generating bitstream. In the end, it is configured to the Xilinx Virtex-6 FPGA ML605 evaluation board.

4.1 Runtime Comparison

The runtime comparison is based on a software SVM solution implemented in Matlab on an Intel core i5-3210M CPU in 2.5GHz. The FPGA implementation with the UART system has the maximum frequency of 112.752 MHz for a single SVM and 80.369 MHz for the cascade SVM, while the system with memory card has the maximum frequency in 104.976MHz for a single SVM and 72.844 MHz for the cascade SVM. The runtime starts from the beginning of the training process, after data are all loaded, and ends when the training process completes.

Apart from the timing analysis setup, to test the runtime with both FPGA implementations, three two dimensions artificial data sets are chosen and trained in the eight dimensions hardware structures. In order to verify the training results, these three artificial data sets are designed with a specific hyperplane, and each of which has 50, 100 and 200 samples. The training results are obtained by Matlab through UART communication in the UART version, and for the memory card version, results are saved

in the CF card. After training, results are verified in Matlab with the original data set and the separating plane is generated in graph; Figure 4.1 is one of the training results.

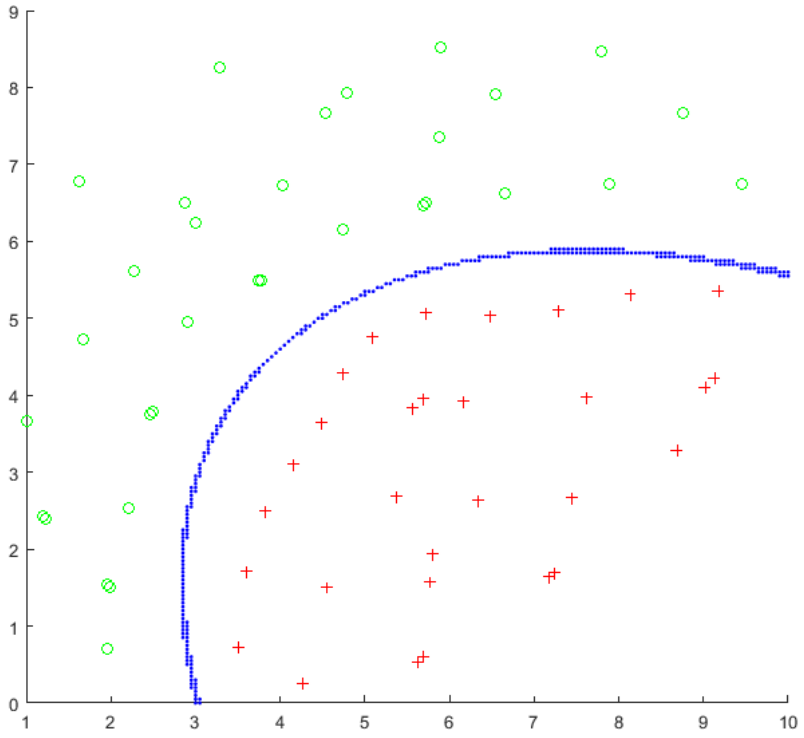

Figure 4.1 A training result for a training model using a data set with 100 samples in UART version of cascade SVM training

4.1.1 Hardware Speedup

From Figure 4.2, there is a big improvement in runtime between software and hardware implementations. The software solution has a significant higher runtime compared with the hardware version.

On the other hand, for the single and cascade SVM, the runtime difference is more and more significant as the samples grow. The more detailed runtime is provided in Table 4.2.

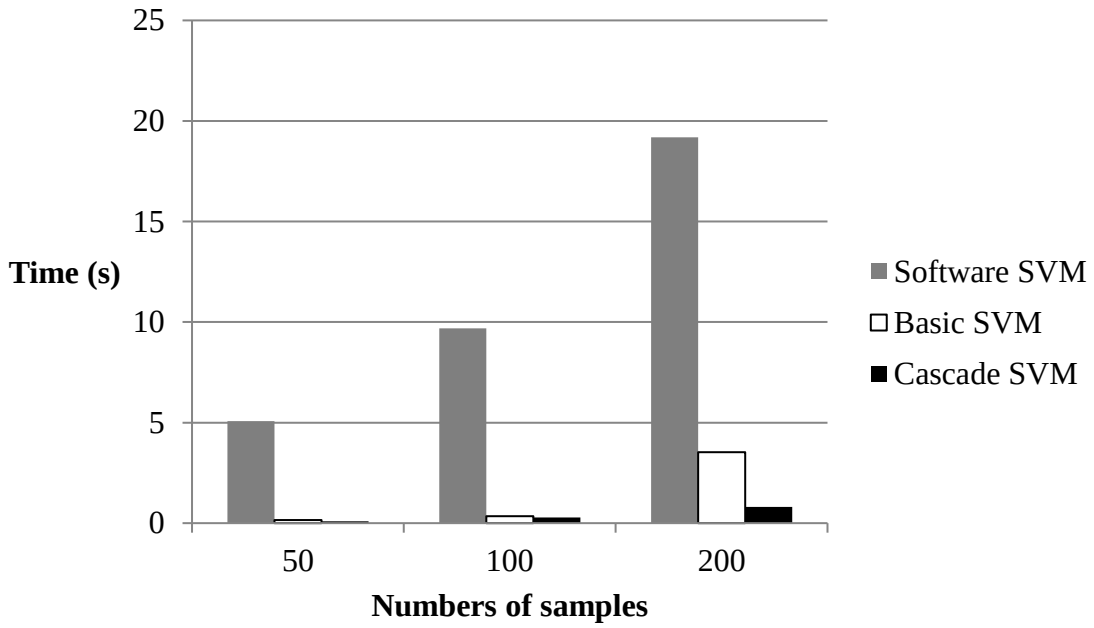

Figure 4.2 Runtime of 50, 100, 200 samples

On the other hand, due to the similarity between training and classification process, the hardware can be also used for the classification process. Both of basic and cascade SVM use SVs from the original data set, therefore they have the same classification process. With a testing data set with 50 samples and the basic SVM classification, the classification time is shown in Table 4.1.

Training Data Set	Number of SVs	Time
50 samples	10	4.76 ms
100 samples	16	7.62 ms
200 samples	25	11.91 ms

Table 4.1 Comparison of classification time

From Table 4.1, it has shown the classification time is less than fifty microseconds, on the other hand, the training process requires hundreds of microseconds to train a data set. The classification process is faster than the training process unless the testing data set is significantly large than the data set.

4.1.2 Basic and Cascade SVM Comparison

Table 4.2 gives a detailed runtime of all the testing data sets from Figure 4.2. As discussed before, a big difference between the basic and cascade SVM has shown. From section 2.2, a great speedup is introduced by using the cascade SVM compared with a basic SVM. And take the data set with 200 samples for example; around 4.3x speedup of

a cascade SVM compared with a basic SVM, and it also enjoys a 23.5x speedup compared with software SVM.

numbers of samples	50	100	200
Software	5.078 s	9.694 s	19.184 s
Hardware			
Basic SVM	0.148 s	0.353 s	3.527 s
Cascade SVM	0.094 s	0.289 s	0.815 s

Table 4.2 Training runtime comparison

Apart from that, if more cores are involved in the cascade SVM, the training data size of each layer will be reduced more. As a result, it is able to have a more significant speedup if more cores are implemented.

4.2 Non-artificial Data Sets Training and Execution Time Comparison

Because both of systems are using the same SVM module, there is no difference in accuracy of training results. From section 4.1, it shows the frequency difference between the UART implementation and memory card implementation, which causes the runtime difference during the training process. Apart from artificial data sets, the following data sets are also tested. In order to transfer the relatively large amount of data through UART in the Matlab interface, the transfer speed becomes longer than the training process. For example, it takes 13.8 minutes to transfer 200 samples in the

Matlab interface, while the training process takes less than five seconds. As data sets grow, the transmission time would become longer.

On the other hand, to verify the dual buffers system and avoid the overheat issue caused by dual-ports BRAMs setup, the maximum samples for a BRAM to hold is set as 100 samples. Every implementation has a different constraint on BRAM samples training due to different data paths and BRAM connections. If more BRAMs are required in the design, the maximum BRAM size is reduced because more long data paths are introduced. For a basic SVM without dual buffers, the maximum training data size with lowest overheat issue is 64kB, while for a cascade SVM with dual buffers, the size is reduced to 4kB. The results are obtained according to the following data sets. On the other hand, the different accuracy between the basic and cascade SVM are not only determined by the original data set division, but also effected by the Microblaze system division. These data sets will be discussed in each sub section, and this section is mainly focused on the execution time discussion between two kinds of hardware implementations.

4.2.1 Pima Indians Diabetes Data Set

The Pima Indians Diabetes data set is tested, which has 796 samples in 8 dimensions [37]. These data are collected from the Pima Indians heritage, with all the patients are female over 21 years old. Each sample has 8 dimensions and 1 classification; those dimensions are medical facts of the patient, including numbers of pregnancy, plasma glucose concentration, diastolic blood pressure, triceps skin fold thickness,

insulin, body mass index, diabetes pedigree function and age. With this data set, more facts of the diabetes are shown.

On the other hand, it is also a widely used data set in machine learning algorithms training. For example, a generalized discriminant analysis and a least square SVM are used in order to find a more accurate way to identify the diabetes by using this data set [38]. In this study, it also gives an overview of other training results of the Pima Indians Diabetes data set. With other 60 training results of other algorithms, most of the accuracies lie in a similar range, from 70% to 80%. And in that research, it is able to achieve 82.05% of accuracy. For the basic SVM, it has 74.74% of accuracy, and for cascade SVM, it has 73.70% of accuracy. From the training results, both of the training methods reach similar training results as shown.

4.2.2 Vertebral Column Data Set

The Vertebral Column data set is also tested, which has 248 samples in 6 dimensions [39]. This data set is a medical data set to classify patients into two categories of their health conditions: normal and abnormal, and its 6 dimensions have the following information: pelvic tilt, lumbar lordosis angle, sacral slope, pelvic radius and grade of spondylolisthesis. These data are based on the patients' body movements to know the health status of joints and bones. With the information, the patient's body structure is classified as normal or abnormal.

With this data set, a research of computer aided diagnosis systems is developed [40]. In that research, it used the following learning algorithms to train the data sets,

including linear SVM, SVM of a kernel with moderate decreasing and a general regression neural network. The conventional training process reaches around 85% of accuracy; however, with the method proposed, it is able to reach around 96% of accuracy, noted that the high accuracy is only achieved for the vertebral column testing. The higher accuracy in the research is achieved by including the rejection techniques into the diagnoses process, which gave a higher accuracy for the diagnosis of the diseases with vertebral column.

By testing the data set in the FPGA, the following results are obtained. For the single SVM, it has 87.10% of accuracy, and for cascade SVM, it has 88.71% of accuracy.

4.2.3 Mammographic Data Set

Finally, the mammographic mass data set is tested, which has 961 samples in 5 dimensions [41]. This data set is about a mammography method that is used in the breast cancer detection. The mammography is one of the most effective ways to diagnose with breast cancer. Although by using mammography and creating an accurate result, it may result in 70% of unnecessary biopsies. To solve this issue, computer aided diagnoses methods are needed for physicians to test for the patients. This data set has the medical data from patients and their mammography results. There are 5 dimensions, including testing results, patients' age, mass' shape, mass' margin, mass' density. The classification indicates whether the patient has cancer or not.

With this data set, it helped researchers to develop the computer aided methods [41]. Two approaches are proposed from the previous research. One is based on the decision-tree learning method, the other one is based on the case-based reasoning with entropic distance measure, both of which reached around 80% in accuracy. Though further research and clinical tests are needed, one of the advantages for these two methods is their potential of the unnecessary breast biopsies reduction.

On the other hand, by testing the data set in the FPGA, we have the following result. For the single SVM, it has 79.29% of accuracy, and for cascade SVM, it has 81.84% of accuracy.

4.2.4 Execution Time Discussion

From section 4.1, the frequency difference between the UART implementation and the memory card implementation has shown. To understand the execution time difference, the first benchmark, the Pima Indians diabetes data set is used for the analyses, the following discussion is based on its training process.

Firstly, in the UART implementation, to transfer a sample from the FPGA board to the host computer, it takes 4.142 second to read one sample through Matlab function. The reading process is implemented through Matlab software for further storage and computation. However, most of the execution time is spent in the serial port connection with the software, which takes 4.131 seconds. The data transfer itself only takes 11.85 milliseconds for writing the address and receiving the sample. With the Matlab

communications, it takes 3297.43 seconds to transfer 796 samples, which is the amount of the first benchmark.

By considering only the UART communication process of the data set transmission, it requires writing 7960 samples into the FPGA board and reading 796 samples to the host computer. The execution time distribution is shown in Figure 4.3., where the transmission time for data transfer takes 21.52% of the whole process.

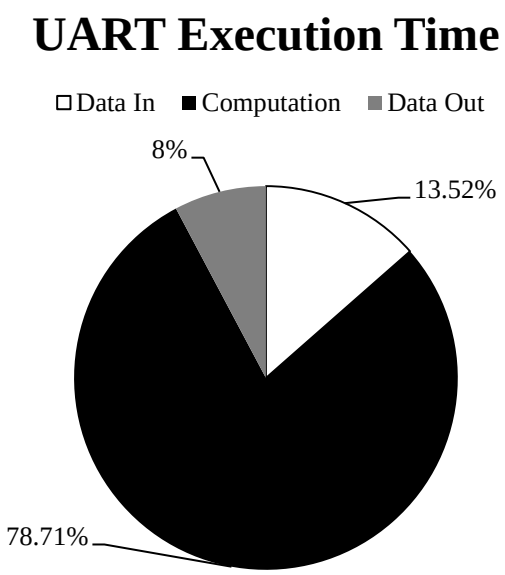

Figure 4.3 Execution time distributions in UART implementation for benchmark 1

On the other hand, by using the memory card implementation, the transmission time can be reduced, which is no longer constrained to the UART transmission. It takes 105.03 seconds of overall execution for the memory card implementation to train

benchmark 1. Most of the time is spent on the computation itself, which is inevitable for the overall training process. And it only takes 0.14 seconds to read the data set from the CF card to BRAMs, and 2.28 seconds to write it back.

From Figure 4.4, the time distributions of the memory card implementation is shown. Compared with Figure 4.3, there is a great difference in the execution time distribution. Also from section 4.1, it indicates the training time for the UART implementation is slightly faster than the memory card implementation. However, the UART transmission speed puts a limit on it.

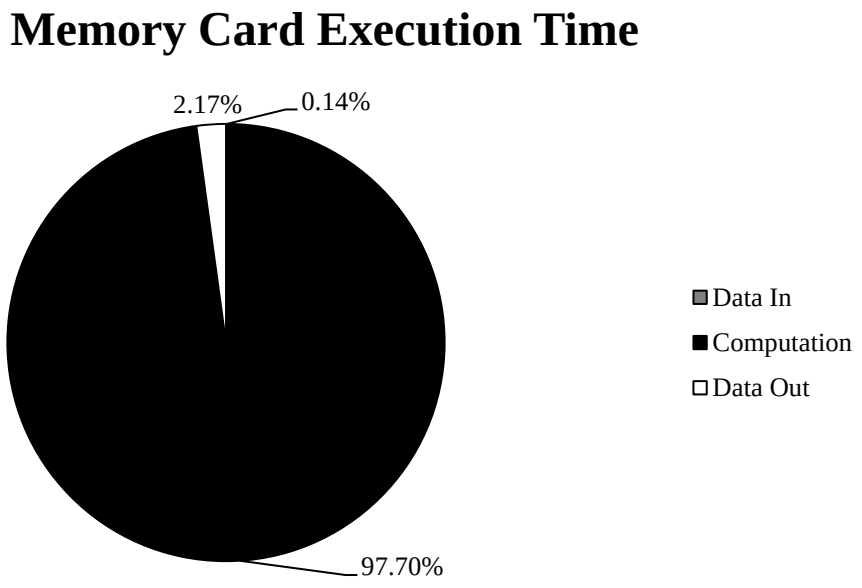

Figure 4.4 Execution time distributions in memory card implementation for benchmark 1

From the previous discussion, the memory card implementation enjoys a better execution time compared with the UART implementation. On the other hand, in order to see the difference, Figure 4.5 shows a side-by-side comparison between each other. In the data out column, it determines the greatest difference of the execution time. For the overall execution time, the memory card implementation has a 1.5 times speedup compared with the UART implementation, in which the data transfer time contributes 10.68 times speedup.

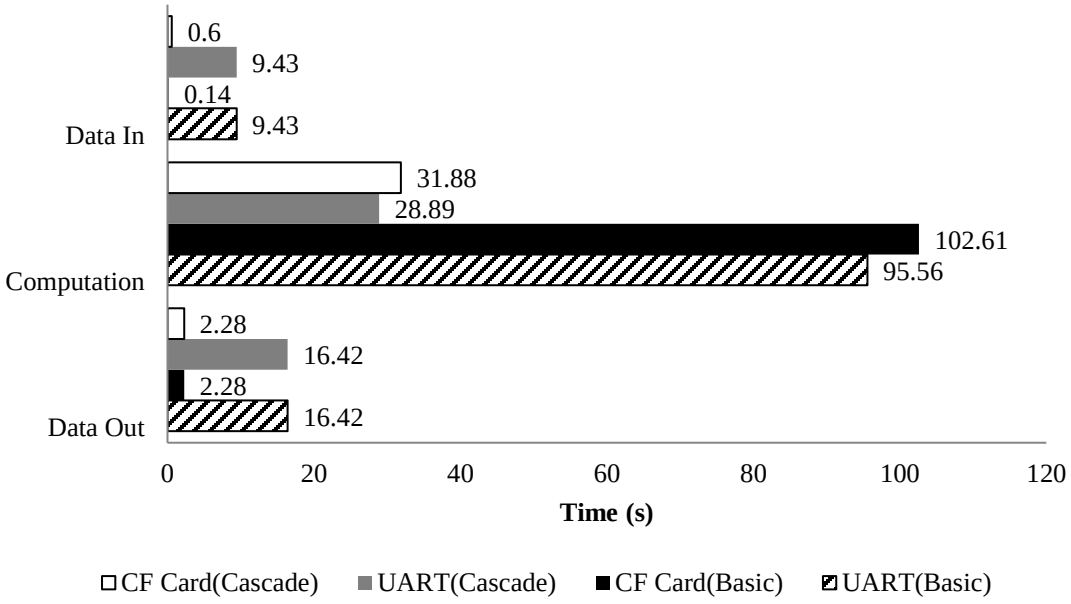

Figure 4.5 Execution time comparisons for benchmark 1

On the other hand, to take advantage of most of the execution time is in computation of the memory card implementation, the dual buffers system is introduced.

By loading in the next data set during the computation process, it is able to save more time because the computation and the data loading process are overlapped.

From Table 4.3, the speedup of different benchmarks has presented, the average speedup is different due to that every benchmark has different number of the samples, where benchmark 1 has 796, benchmark 2 has 248, and benchmark 3 has 961. From Table 4.3, it indicates that the more samples a data set has, the faster the speedup can achieve.

	Without Dual Buffers			With Dual Buffers		
	Basic	Cascade		Basic	Cascade	
		2-cores	4-cores		2-cores	4-cores
Benchmark1	110.34 s	52.32 s	25.91 s	105.03 s	49.06 s	21.73 s
Benchmark2	43.25 s	37.85 s	12.46 s	41.27 s	33.72 s	12.46 s
Benchmark3	144.68 s	66.80 s	38.61 s	131.25 s	57.90 s	30.72 s

Table 4.3 The memory card implementation execution time

With the dual buffer system implementation, the execution time is reduced. By observing the speedup from Figure 4.3, it has shown that if a data set has fewer samples during one training process, it would benefit more from the dual buffer speedup in the whole training process. From benchmark 3, the dual buffers speedup for a basic SVM training is 9.28%, while the 4-core cascade SVM training is 20.44%.

From the previous discussion, a speedup has shown from migration of the design from UART implementation to memory card implementation. On the other hand, by the dual buffer method, it is able to reach a faster training process.

4.3 Device Utilization and Power/Energy Consumption

For device utilization, the memory card system implementation occupied more device compared with the UART system implementation. Table 4.4 shows the device utilization for hardware implementations of two designs. From Table 4.4, it shows that the memory card implementation requires more device usage, because the memory card implementation has the Microblaze system for the memory card control, which occupies most of the device.

UART	Slices Registers	LUTs	Occupied Slices	LUT Flip Flop
Basic SVM	1655	2647	918	3120
2-core SVM	3497	6025	2125	7119
Memory card	Slices Registers	LUTs	Occupied Slices	LUT Flip Flop
Basic SVM	7257	9189	3647	10884
2-core SVM	9540	13136	5301	15602

Table 4.4 Device utilization

In Table 4.5, it has shown the device utilization without a Microblaze system. Regardless the device occupied by the Microblaze system which has 5622 slices of registers, the top module only occupied 12 slices registers, the SVM unit dominants the

size of device utilization, which has 1623 slices registers, while it shares the same structure as the SVM implementation with the UART communication, which is shown in Figure 3.8. Although it has a significant growth in device utilization from UART implementation to the memory card implementation, it is still a relative small design compared with the whole FPGA.

Memory card	Slices Registers	LUTs	Occupied Slices	LUT Flip Flop
Basic SVM	1635	2469	832	3129
2-core SVM	3595	6531	2247	7529

Table 4.5 Device utilization without the Microblaze system

On the other hand, the cascade SVM requires more power compared with a single SVM implementation. Therefore, with more device used, more power are required to perform the implementation. The power of these two implementations is reported in Table 4.6.

The power report is generated from Xilinx XPower Analyzer, not only does it provide the power consumption of the design, but it also provides the maximum working temperature. With the experiment results from benchmark 1, the following comparison between basic and cascade SVM is shown in Table 4.6.

	Power	Execution Time	Energy
Basic SVM	4095.45 mW	110.34 s	451.892 J
Cascade SVM	4152.44 mW	52.32 s	217.2557 J

Table 4.6 Power and energy comparisons

From Table 4.6, it has shown that by reducing the execution time through different algorithms, the energy is able to be reduced significantly.

From the above device utilization and power comparison, though the cascade SVM requires more resource compared with a basic SVM, the cascade SVM has a better energy reduction compared with a basic SVM.

5. SUMMARY AND CONCLUSIONS

5.1 Summary and Conclusions

This thesis proposed a hardware implementation of the cascade SVM algorithm. From the previous discussion, the hardware implementation has shown contributions in technology and science research of a basic SVM algorithm in classifications of machine learning algorithms. By eliminating non-SVs at an early stage, the cascade SVM is a faster method compared with a single SVM.

On the other hand, by comparing hardware and software implementations, it shows the benefits of implementing the design in hardware. However, even an ASIC implementation of the cascade SVM has been proposed, there are still some limitations of it.

To break the limit, the design is implemented in the FPGA. In the FPGA version, it is a more flexible structure that can easily increase the dimensions of training models. The dimensions of training models for training are increased from two dimensions to eight dimensions. Moreover, by introducing the Microblaze system and a large storage memory card, the maximum data storage size is increased from 8 kB to 2 GB, which gives the potential for larger data sets training. Apart from that, it has the benefit of hardware implementations in terms of runtime comparison with software implementations. That is, it has approximately 23x speedup compared with software implementations for a two dimensions data set in a training model with 200 samples.

In sum, the hardware implementation has a better performance in runtime compared with the software implementations. With the FPGA version, it is also able to train multiple dimensional models and work with more samples in a faster manner. By the fast configuration characteristic of the FPGA, it is possible to modify the design to fit in higher dimensional training models compared with the ASIC design.

5.2 Future Work

Though in this thesis, it has the contributions discussed in previous section, there are still some limitations need to be improved.

In the memory card implementation, to train a large data set over the size of BRAMs, the base system divides the data set according to the order without checking the classification distribution in the subsets may reduce the accuracy of the training process. On the other hand, due to the long data path from SVM units to the Microblaze system control, which may cause an overheat issue if the BRAM size is relatively big.

The FPGA implementation of the SVM provides a faster way for training and more flexibility to modify the structure in the future. With characteristics of the FPGA, the FPGA is able to reconfigure the hardware in a faster way. Therefore, it is easier to reach a high dimensional models training in the future. More dimensions are able to be added in the current structure by adding the input storage registers and the corresponding multipliers. Aside from the dimensions growth, the number of classifications may also be increased in the future. With higher numbers of classifications, it is possible to deal with more training models in hardware implementations.

If the above improvements have been done, the hardware implementation of the cascade SVM is able to be more practical and useful for the research works or other needs of SVMs.

REFERENCES

[1] Mathews, J.D., et al., *Cancer risk in 680,000 people exposed to computed tomography scans in childhood or adolescence: data linkage study of 11 million Australians.* BMJ, 2013. **346**: p. f2360.

[2] Hales, S., et al., *Potential effect of population and climate changes on global distribution of dengue fever: an empirical model.* The Lancet, 2002. **360**(9336): p. 830-834.

[3] Wang, Y., et al., *Gene selection from microarray data for cancer classification--a machine learning approach.* Comput Biol Chem, 2005. **29**(1): p. 37-46.

[4] Kubat, M., R.C. Holte, and S. Matwin, *Machine learning for the detection of oil spills in satellite radar images.* Machine Learning, 1998. **30**(2-3): p. 195-215.

[5] Huang, G.-B., Q.-Y. Zhu, and C.-K. Siew. *Extreme learning machine: a new learning scheme of feedforward neural networks*. in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*. 2004. IEEE.

[6] Berral, J.L., et al. *Towards energy-aware scheduling in data centers using machine learning*. in *Proceedings of the 1st International Conference on energy-Efficient Computing and Networking*. 2010. ACM.

[7] Wang, Q., P. Li, and Y. Kim, *A Parallel Digital VLSI Architecture for Integrated Support Vector Machine Training and Classification.* IEEE Transactions on Very Large

Scale Integration (VLSI) Systems, 2015. **23**(8): p. 1471-1484.

[8] Guyon, I. and A. Elisseeff, *An introduction to variable and feature selection.* Journal of Machine Learning Research, 2003. **3**(Mar): p. 1157-1182.

[9] Segal, M.R., *Machine learning benchmarks and random forest regression.* Center for Bioinformatics & Molecular Biostatistics, 2004.

[10] Mantel, N., *The detection of disease clustering and a generalized regression approach.* Cancer Research, 1967. **27**(2 Part 1): p. 209-220.

[11] Yuan, G.-X., C.-H. Ho, and C.-J. Lin, *Recent advances of large-scale linear classification.* Proceedings of the IEEE, 2012. **100**(9): p. 2584-2603.

[12] Zhang, T. and F.J. Oles, *Text categorization based on regularized linear classification methods.* Information Retrieval, 2001. **4**(1): p. 5-31.

[13] Wang, S.-C., *Artificial neural network*, in *Interdisciplinary Computing in Java Programming*. 2003, Springer, New York. p. 81-100.

[14] Zhou, Z.-H., et al., *Lung cancer cell identification based on artificial neural network ensembles.* Artificial Intelligence in Medicine, 2002. **24**(1): p. 25-36.

[15] Furey, T.S., et al., *Support vector machine classification and validation of cancer tissue samples using microarray expression data.* Bioinformatics, 2000. **16**(10): p. 906-914.

[16] Tong, S. and E. Chang. *Support vector machine active learning for image retrieval*. in *Proceedings of the ninth ACM international conference on Multimedia*. 2001. ACM.

[17] Genov, R. and G. Cauwenberghs, *Kerneltron: support vector" machine" in silicon.* IEEE Transactions on Neural Networks, 2003. **14**(5): p. 1426-1434.

[18] Kucher, P. and S. Chakrabartty. *An energy-scalable margin propagation-based analog VLSI support vector machine*. in *2007 IEEE International Symposium on Circuits and Systems*. 2007. IEEE.

[19] Kuan, T.-W., et al., *VLSI design of an SVM learning core on sequential minimal optimization algorithm.* IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2012. **20**(4): p. 673-683.

[20] Anguita, D., A. Boni, and S. Ridella, *A digital architecture for support vector machines: theory, algorithm, and FPGA implementation.* IEEE Transactions on Neural Networks, 2003. **14**(5): p. 993-1009.

[21] Papadonikolakis, M. and C.-S. Bouganis, *Novel cascade FPGA accelerator for support vector machines classification.* IEEE Transactions on Neural Networks and Learning Systems, 2012. **23**(7): p. 1040-1052.

[22] Catanzaro, B., N. Sundaram, and K. Keutzer. *Fast support vector machine training and classification on graphics processors*. in *Proceedings of the 25th international conference on Machine learning*. 2008. ACM.

[23] Bauer, S., et al. *FPGA-GPU architecture for kernel SVM pedestrian detection*. in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*. 2010. IEEE.

[24] Cortes, C. and V. Vapnik, *Support-vector networks.* Machine Learning, 1995. **20**(3): p. 273-297.

[25] Shawe-Taylor, J. and N. Cristianini, *Kernel methods for pattern analysis*. 2004: Cambridge University Press, Cambridge, United Kingdom.

[26] Kang, K. and T. Shibata, *An on-chip-trainable Gaussian-kernel analog support vector machine.* IEEE Transactions on Circuits and Systems I: Regular Papers, 2010. **57**(7): p. 1513-1524.

[27] Platt, J.C., *12 fast training of support vector machines using sequential minimal optimization.* Advances in kernel methods, 1999: p. 185-208.

[28] Kuhn, H.W., *Nonlinear programming: a historical view*, in *Traces and Emergence of Nonlinear Programming*. 2014, Springer, Basel. p. 393-414.

[29] Luo, Z.-Q. and W. Yu, *An introduction to convex optimization for communications and signal processing.* IEEE Journal on selected areas in communications, 2006. **24**(8): p. 1426-1438.

[30] Graf, H.P., et al. *Parallel support vector machines: The cascade svm*. in *Advances in neural information processing systems*. 2004.

[31] Ramírez, J., et al., *Parallelization of automatic classification systems based on support vector machines: Comparison and application to JET database.* Fusion Engineering and Design, 2010. **85**(3): p. 425-427.

[32] Bell, C.G., J.C. Mudge, and J.E. McNamara, *Computer Engineering: A DEC View of Hardware Systems Design*. 2014: Digital Press: p. 73.

[33] Xilinx, *IP Processor Block RAM (BRAM) Block (v1.00a)*, accessed on 06/2016 http://www.xilinx.com/support/documentation/ip_documentation/bram_block.pdf

[34] Xilinx, *Using the MicroBlaze Processor to Accelerate Cost-Sensitive Embedded System Developmen*t, accessed on 06/2016 http://www.xilinx.com/support/documentation/white_papers/wp469-microblaze-for-cost-sensitive-apps.pdf

[35] Xilinx, *AXI Reference Guide*, accessed on 06/2016 http://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf

[36] Xilinx, *Xilkernel*, accessed on 06/2016 http://www.xilinx.com/ise/embedded/edk91i_docs/xilkernel_v3_00_a.pdf

[37] Smith, J.W., et al. *Using the ADAP learning algorithm to forecast the onset of diabetes mellitus*. in *Proceedings of the Annual Symposium on Computer Application in Medical Care*. 1988. American Medical Informatics Association: p. 261.

[38] Polat, K., S. Güneş, and A. Arslan, *A cascade learning system for classification of diabetes disease: Generalized discriminant analysis and least square support vector machine.* Expert Systems with Applications, 2008. **34**(1): p. 482-487.

[39] Sousa, R., B. Mora, and J.S. Cardoso. *An ordinal data method for the classification with reject option*. in *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*. 2009. IEEE.

[40] Neto, A.R. and G. Barreto, *On the application of ensembles of classifiers to the diagnosis of pathologies of the vertebral column: A comparative analysis.* IEEE Transactions on Latin America, 2009. **7**(4): p. 487-496.

[41] Elter, M., R. Schulz-Wendtland, and T. Wittenberg, *The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process.* Medical Physics, 2007. **34**(11): p. 4164-4172.