

American University in Cairo

AUC Knowledge Fountain

Theses and Dissertations

2-1-2016

GAdaboost: Accelerating adaboost feature selection with genetic algorithms

Mai Tolba

Follow this and additional works at: <https://fount.aucegypt.edu/etds>

Recommended Citation

APA Citation

Tolba, M. (2016). *GAdaboost: Accelerating adaboost feature selection with genetic algorithms* [Master's thesis, the American University in Cairo]. AUC Knowledge Fountain.

<https://fount.aucegypt.edu/etds/553>

MLA Citation

Tolba, Mai. *GAdaboost: Accelerating adaboost feature selection with genetic algorithms*. 2016. American University in Cairo, Master's thesis. *AUC Knowledge Fountain*.

<https://fount.aucegypt.edu/etds/553>

This Thesis is brought to you for free and open access by AUC Knowledge Fountain. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AUC Knowledge Fountain. For more information, please contact mark.muehlhaeusler@aucegypt.edu.

The American University in Cairo

School of Science and Engineering

**GAdaboost: Accelerating Adaboost Feature
Selection with Genetic Algorithms**

A Thesis Submitted to the Department of Computer Science and Engineering
in Partial Fulfillment of the Requirements for the Degree of Master of Science

By

Mai Mohamed Tolba

Under The Supervision of

Prof. Mohamed Moustafa

Fall 2016

ACKNOWLEDGEMENTS

In the name of Allah the most gracious and most merciful

I would like to thank my supervisor Dr. Mohamed Moustafa, for his continuous support and guidance throughout this project. It was really an honor to be taught by such a great professor. I would also like to express my gratitude to my thesis committee members Dr. Amr Gonied and Dr. Hazem Abbas for their helpful feedback.

I would like to thank my father Prof. Mohamed Tolba for his constant guidance, support and encouragement. My mother Galila for her endless love and care. My brother Ahmed for his guidance. My sister Amal, my cousin Aeysha, my sister in law Ghada, my best friend Dalia and my husband Tarek for always being there for me and pushing me to do my best. Without you all this thesis would have not been complete.

ABSTRACT

Throughout recent years Machine Learning has acquired attention, due to the abundant data. Thus, devising techniques to reduce the dimensionality of data has been on going. Object detection is one of the Machine Learning techniques which suffer from this draw back. As an example, one of the most famous object detection frameworks is the Viola-Jones Rapid Object Detector, which suffers from a lengthy training process due to the vast search space, which can reach more than 160,000 features for a 24X24 image. The Viola-Jones Rapid Object Detector also uses Adaboost, which is a brute force method, and is required to pass by the set of all possible features in order to train the classifiers.

Consequently, ways for reducing the whole feature set into a smaller representative one, eliminating those features that have non relevant information, were devised. The most commonly used technique for this is Feature Selection with its three categories: Filters, Wrappers and Embedded. Feature Selection has proven its success in providing fast and accurate classifiers. Wrapper methods harvest the power of evolutionary computing, most commonly Genetic Algorithms, in finding the set of representative features. This is mostly due to the Advantage of Genetic Algorithms and their power in finding adequate solutions more efficiently.

In this thesis we propose GAdaboost: A Genetic Algorithm to accelerate the training procedure of the Viola-Jones Rapid Object Detector through Feature Selection. Specifically, we propose to limit the Adaboost search within a sub-set of the huge feature space, while evolving this subset following a Genetic Algorithm. Experiments demonstrate that our proposed GAdaboost is up to 3.7 times faster than Adaboost. We also demonstrate that the price of this speedup is a mere decrease (3%, 4%) in detection accuracy when tested on FDDB benchmark face detection set, and Caltech Web Faces respectively.

TABLE OF CONTENTS

Contents

ACKNOWLEDGEMENTS	II
ABSTRACT.....	III
TABLE OF CONTENTS	IV
LIST OF FIGURES.....	VII
LIST OF TABLES.....	IX
LIST OF ABBREVIATIONS.....	X
CHAPTER (1): INTRODUCTION.....	1
1.1 PROBLEM DEFINITION.....	2
1.2 MOTIVATION.....	2
1.2.1 Primary Experiments.....	4
1.3 ORGANIZATION OF THE THESIS	6
CHAPTER (2): BACKGROUND	7
2.1 OBJECT DETECTION BACKGROUND	7
2.1.1 Viola-Jones Rapid Object Detector.....	7
2.2 ENHANCEMENTS OVER VIOLA-JONES	12
2.2.1 The use of SVMs and new stopping criteria	12
2.2.2 Increasing Haar Features	13
2.3 FEATURE SELECTION.	15
2.3.1 Filters.....	15
2.3.2 Wrappers	16
2.3.3 Embedded.....	16
2.4 GENETIC ALGORITHMS	17
2.4.1 Overview	17
2.4.2 GA Details.....	19
2.4.3 Strength of GAs.....	24
2.5 FEATURE SELECTION WITH GAS	27
2.5.1 Work that utilizes GA with feature selection.	27
2.6 SUMMARY.....	29

CHAPTER (3): PROPOSED APPROACH30

3.1	OPENCV	30
3.2	GADABOOST OVERVIEW.....	31
3.3	GA DETAILS	35
3.3.1	Initial population	35
3.3.2	Chromosome Representation	35
3.3.3	Fitness Function	36
3.3.4	Selection Mechanism	37
3.3.5	Crossover.....	37
3.3.6	Mutation	38

CHAPTER (4) EXPERIMENTS.....39

4.1	TRAINING.....	39
4.1.1	Positive images.....	39
4.1.2	Negative images	40
4.1.3	Training Parameters.	40
4.1.4	Resultant trained classifier	42
4.2	TESTING.....	43
4.2.1	Datasets	43
4.2.2	Detection with OpenCV.	44
4.2.3	Evaluation Tools	45
4.3	INDIVIDUAL AND POPULATION FITNESS	47
4.3.1	Objective	47
4.3.2	Method	47
4.3.3	Results	47
4.3.4	Discussion	47
4.4	POPULATION SIZE VERSUS TRAINING TIME.....	48
4.4.1	Objective	48
4.4.2	Method	48
4.4.3	Results	48
4.4.4	Discussion	49
4.5	BASELINE.....	49
4.5.1	Objective	49
4.5.2	Method	49
4.5.3	Results	49
4.5.4	Discussion	51
4.6	GADABOOST 20 ITERATIONS.....	51
4.6.1	Objective	51
4.6.2	Method	52
4.6.3	Results	52
4.6.4	Discussion	54
4.7	GADABOOST 50 ITERATION	55
4.7.1	Objective	55
4.7.2	Method	55
4.7.3	Results	56
4.7.4	Discussion	58
4.8	TRAINING SPEED VERSUS ACCURACY	59
4.8.1	Objective	59

4.8.2	Results	59
4.8.3	Discussion	62
CHAPTER (5): CONCLUSIONS		64
5.1	CONTRIBUTIONS	64
5.2	FUTURE WORK	65
REFERENCES		66

LIST OF FIGURES

Figure1-1: The effect of varying the image size on the number of features.....	4
Figure 1-2: The effect of increasing Haar feature types on the total number of features per a 24x24 image	5
Figure 2-1: Haar features relative to the enclosing detection window (Viola & Jones, 2001)	8
Figure 2-2: Integral image illustration(Viola & Jones, 2001).....	9
Figure 2-3: Adaboost Algorithm(Viola & Jones, 2001)	10
Figure 2-4 Schematic description of a detection cascade(Viola & Jones, 2001).....	11
Figure 2-5: Roc Curve for detector on MIT+CMU dataset (Viola & Jones, 2001).....	12
Figure 2-6 Object detection average precision on selected objects (top 40 in 113 test image)(Q. Li et al., 2012).....	13
Figure 2-7 Extended set of Haar features black and white regions have negative and positive weights (Lienhart & Maydt, 2002).....	14
Figure 2-8: Basic versus extended features set. (Lienhart & Maydt, 2002)	15
Figure 2-9: How GAs Work (Lee & Lee, 2014).....	19
Figure 2-10: Single point Crossover (Hasançebi & Erbatur, 2000).....	19
Figure 2-11: Two-point crossover (Hasançebi & Erbatur, 2000)	20
Figure 2-12: Uniform crossover(Hasançebi & Erbatur, 2000)	20
Figure 2-13 Tournament selection(Noraini & Geraghty, 2011)	21
Figure 2-14 Roulette wheel selection (Noraini & Geraghty, 2011).....	23
Figure 2-15: Comparison of mean cutsizes (Manikas & Cain, 1996)	26
Figure 3-1 GAdaboost algorithm flowchart.....	32
Figure 3-2 Population illustration	35
Figure 3-3 Chromosome to feature mapping	36

Figure 3-4 GAdaboost crossover illustration	38
Figure 4-1 Positive training images	40
Figure 4-2 Negative images samples	40
Figure 4-3 Opencv_traincascade parameters	41
Figure 4-4 Resultant cascade classifier.....	42
Figure 4-5 Example of a stored stage of resultant classifier	43
Figure 4-6 Example of annotated FDDB dataset(Jain & Learned-Miller, 2010).	44
Figure 4-7 Best individual fitness and average population fitness over 50 iterations .	47
Figure 4-8 Population size vs training time.	48
Figure 4-9 Examples of detection of baseline on images	50
Figure 4-10: Baseline performance on FDDB dataset.....	50
Figure 4-11: Baseline performance on the Caltech dataset	51
Figure 4-12: Examples of detection of Gadaboost20 on images	53
Figure 4-13: ROC curve of multiple GAdaboost20 Classifiers on FDDB	54
Figure 4-14: ROC curve of multiple GAdaboost20 Classifiers on Caltech.....	54
Figure 4-15: Example of detections of GAdaboost50	57
Figure 4-16: ROC curve of multiple GAdaboost50 Classifiers on FDDB	57
Figure 4-17 ROC curve of multiple GAdaboost50 Classifiers on Caltech.....	58
Figure 4-18 Training time in minutes of each of the experiments.....	60
Figure 4-19 Y error bars for all the runs of the 20 iterations GAdaBoost on FDDB. .	60
Figure 4-20: Y error bars for all the runs of the 50 iterations GAdaBoost on FDDB .	61
Figure 4-21: Y error bars for all the runs of the 20 iterations GAdaBoost on Caltech	
Web Faces.....	61
Figure 4-22: Y error bars for all the runs of the 50 iterations GAdaBoost on Caltech	
Web Faces.....	62

LIST OF TABLES

Table 1-1 Comparison between Brute Force and GA in TSP.....	6
Table 2-1: Number of features inside a 24X24 image for each prototype (Lienhart & Maydt, 2002).....	14
Table 2-2: Summary of feature selection techniques.....	17
Table 2-3 Discrimination results on synthetic images (Ayala-ramirez et al., 2006) ...	24
Table 2-4 Discrimination results on natural images (Ayala-ramirez et al., 2006).....	24
Table 2-5: comparison on Caltech dataset to other methods	25
Table 3-1 Comparison of OpenCV Application used to train a cascade classifier.....	31
Table 4-1 Training time for each run of training GAdaboost 20	52
Table 4-2 Training time for each run of training GAdaboost 50	56
Table 4-3 Summary of performance of the baseline, GAdaboost50, and GAdaboost20	63

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimization
CV	Computer Vision
EC	Evolutionary Computing
EA	Evolutionary Algorithms
Fddb	Face Detection Database
GA	Genetic Algorithms
HOG	Histogram of Gradients
LBP	Local Binary Patterns
ML	Machine Learning
OpenCV	Open Source Computer Vision Library
PSO	Particle Swarm Optimization
ROC	Receiver Operator Curve
SFS	Sequential Forward Selection
SBS	Sequential Backward Selection
SVM	Support Vector Machine
TBB	Threading Building Blocks
TSP	Travelling Salesman Problem

CHAPTER (1): INTRODUCTION

Machine learning and training require large feature sets, which can be time consuming to explore. With the advancements in this field the need for algorithms to decrease the training time arises. Genetic Algorithms (GA) have proven their strength in solving problems like the aforementioned one, especially those concerned with exploring large search spaces and providing acceptable results in a significantly reduced amount of time than that of the brute force manner. Many researches have explored the use of GA in time consuming tasks like Feature Selection, which aims to choose a representative small sub-set of features from the whole set of features (B Xue, Zhang, Browne, & Yao, 2016).

Object detection lies in the set of machine learning techniques that require a huge search space for training, thus their training is time consuming. Object detection is concerned with detecting whether an object is present in a given image and where it lies in this image. It has many applications including but not limited to, face detectors in all modern state of the art cameras, automotive safety, video indexing, image classification, surveillance and content-based image retrieval (Lillywhite, Lee, Tippetts, & Archibald, 2013).

A lot of research has been applied to this area, due to its complex nature as detection is hard to achieve in different light conditions, occlusion and the angle in which the object appears in the image (Lienhart & Maydt, 2002; Lillywhite et al., 2013; Viola & Jones, 2001). Researchers have been trying to implement efficient high speed detectors that work in real time and have a high percentage of accuracy. Though the Viola-Jones detector has reached an impressive detection speed, it still consumes a lot of time in training. Viola-Jones uses Adaboost, a type of boosting algorithms, to select and combine weak classifiers to form a strong one. Adaboost is simple and adaptive (Dezhen & Kai, 2008), yet it operates in a brute force manner, passing by the set of all features multiple times. This can be very time consuming, as the search space consists of a set of more than 160,000 features for a 24X24 image.

This thesis is multi-disciplinary, as it deals with three sub-research areas in Computer Science. The three main areas are Computer Vision (CV), Machine Learning (ML) and Statistics, and Evolutionary Computing (EC). This thesis's main focus is on Object detection which lies under CV, Boosting and Feature Selection which is a sub-area of ML and Genetic Algorithms with is a famous algorithm in EC. In brief this work aims towards enhancing the training time taken by the Adaboost algorithm through Feature Selection using Genetic

Algorithms. Specifically it aims to speed up the training process of the Viola-Jones Rapid Object Detector by finding a small set of representative features to be provided to the Adaboost algorithm, instead of the original method of going through the set of all possible features in a brute force manner.

1.1 Problem Definition

Having Robust and efficient detectors has become the goal of many research over many years. An ideal detector can be described as one that is both efficient and provides plausible results. A lot of research has been done in order to enhance several machine learning techniques and try to reach the previously mentioned goal of ideal detectors.

Though Boosting algorithms like Adaboost are simple and effective, they suffer from lengthy training processes due to their brute force nature. With the advancement of Machine Learning and the abundance of data in recent years (Yusta, 2009), the drawback of these algorithms becomes more apparent, as the dimensionality and the volume of data directly affect the training time. For example, in the training of the Viola-Johns Rapid Object detector, the Adaboost algorithm goes through the set of all possible features in a brute force manner, for the training of each weak classifier. This can be very time consuming, as the search space consists of a set of more than 160,000 features for a 24X24 image (Viola & Jones, 2001). Some of the formerly mentioned features are non-representative as they have poor predictive power of the object's existence in this image. Selecting a representative set of features and discarding the non-useful ones can be achieved through Feature Selection. Feature selection, allows for the decrease of the search space with minimum loss of quality, as it focuses on eliminating those features that are not useful when solving the problem at hand. Applying this concept to the Adaboost algorithm will help in overcoming the drawback of its lengthy training process while benefiting from its simplicity and adaptively.

1.2 Motivation

The Viola-Jones object detector uses a cascaded stage classifier in order to rapidly detect objects. However, the training of this classifier is time consuming, since the training algorithm utilized is Adaboost which works by going through the set of all possible features to

evaluate each feature in a brute force manner to choose one weak classifier. This process takes place multiple times as the essence of boosting is to combine multiple weak classifiers to get a strong one. The cascaded structure makes training even slower as the previously mentioned process is repeated for each stage of the cascaded classifier. The number of times the Adaboost algorithm passes through the set of all possible features to train a cascade classifier, can be obtained by summing up the number of weak classifiers in all the stages as shown in Equation 1.1, where WC is the number of weak classifiers per stage, and n is the number of stages in the cascade classifier.

$$iters = \sum_{i=0}^n WC_i \quad (1.1)$$

Examining the set of all possible features multiple times can be analogous to expanding the feature set. In order to have a deeper understanding of the effects of repeating the number of iterations a look at how much the feature set expands is necessary. The total number of features examined in training a cascade classifier is obtained by multiplying the number of iteration done by the Adaboost algorithm by the total number of features in the original feature set as shown in Equation 1.2, where TF is the total number of features examined, iters in the number of times the Adaboost passes by the original search space (which can be obtained from Equation 1.1), and osp is the number of features in the original search space

$$TF = iters * osp \quad (1.2)$$

As an example, if we built a simple 5 stage classifier and the number of features are 10, 15, 20, 25, 30 in stages 1, 2, 3, 4 and 5 respectively, then the total number of times the Adaboost passed by the set of all possible features (the original search space) in the training phase can be calculated by summing up the number of features, i.e. 10+15+20+25+30 which is equal to 100 in this final trained classifier, which is analogous to an increased search space by a 100 times, as the Adaboost would have passed by 160,00000 features if the original search space had 160,000 features (160,000 X 100 from Equation 1.2)

In conclusion, by eliminating the unnecessary features, the time taken to train a cascade classifier can be significantly reduced. This can be achieved by the means of Feature Selection, where the best features are chosen and the unnecessary ones are discarded. Feature Selection can be achieved by exploiting GAs, since GAs are widely used heuristics in Feature Selection (Tsai, Eberle, & Chu, 2013). Another motivation for using GA with Feature Selection is that

inducing GAs and Feature Selection mechanisms have been continuously studied for decades (Charaoui & Flórez-Revuelta, 2013; Yusta, 2009) and have proven to be successful.

1.2.1 Primary Experiments

This section provides 2 experiments to support the motivation of this work. It shows evidence of how vast the search space of features can be by examining the effect of increasing Haar feature types on the total number of features and the effect of image size of the number of features in the search space. Moreover, an experiment was done to compare the brute force technique versus the GAs in solving the Travelling Salesman Problem.

1.2.1.1 Number of Features Per image

The main problem to be dealt with in order to enhance the performance of the Viola-Jones detector, is the vast search space. To give an idea of how vast this search space can get; a simple experiment has been carried out. This experiment calculates the number of features (the search space) once when varying the image dimensions and another when increasing the types of Haar features. This experiment considers getting all possible sizes of each feature and all possible positions by shifting the window one pixel. Figure1-1 shows the exponential growth of the search space when increasing the image dimensions. Figure 1-2 also shows the growth of the search space by increasing the types of Haar features used.

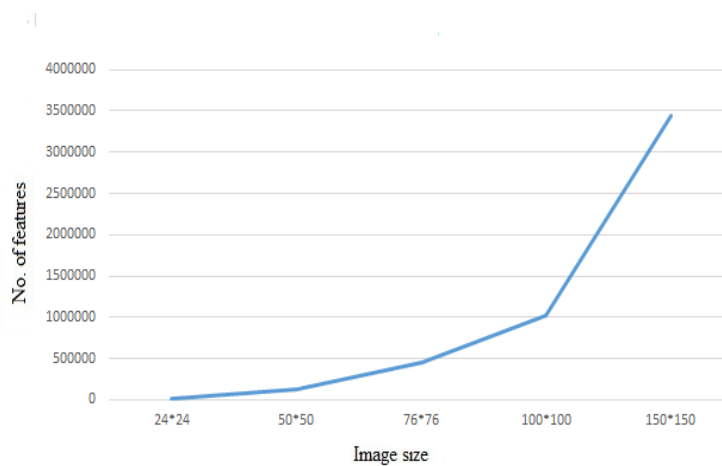


Figure1-1: The effect of varying the image size on the number of features

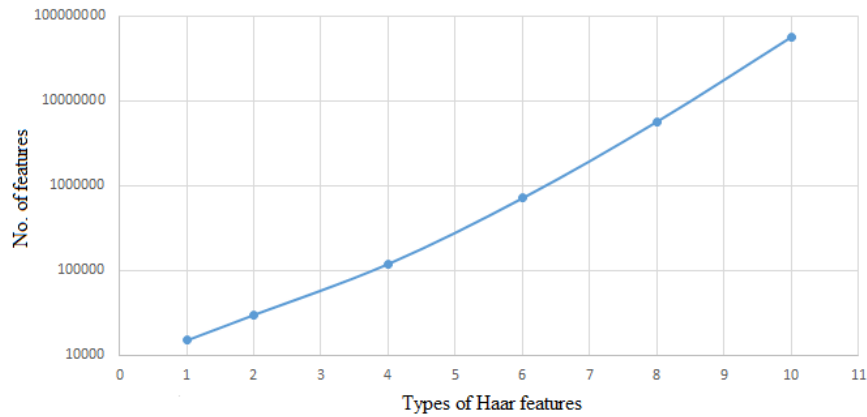


Figure 1-2: The effect of increasing Haar feature types on the total number of features per a 24x24 image

1.2.1.2 Performance of GA in Travelling Salesman Problem (TSP)

With a vast search space the main problem is time. It's a time consuming process to go through the search space one by one in a brute force manner (as done by the original Viola-Jones implementation). The former point is the motive for this work, since Genetic algorithms in general are efficient in searching large spaces (Lillywhite et al., 2013). To further show the effectiveness of GA on speed and accuracy an experiment was conducted. The famous Travelling Salesman Problem (TSP) has been examined once using brute force and once using Genetic Algorithms (implementation used was done by (Jacobson, 2012)). The Travelling Salesman Problem is concerned with finding the shortest route of a journey between given countries. For this experiment the same 9 countries have been used for both the GA and the brute force methods. The brute force method is done by exploring all the possible routes (which are $9!$ (362880) routes) in this case, then choosing the shortest one. The results of the experiment show that the GA achieved a comparable accuracy by evaluation a 100 generations in only 5.9% of the time taken by the brute force method. Table 1-1 shows the exact results of the timing and the shortest distance found by both the brute force and the GA.

Table 1-1 Comparison between Brute Force and GA in TSP

	Brute Force	GA
Time Taken in seconds	0.659	0.039
Shortest distance	332	452

1.3 Organization of the Thesis

The rest of the thesis is organized as follows: **Chapter 2** provides a comprehensive background on the main topics covered in this thesis, like the Viola-Jones Rapid Object Detector and the enhancements done over their work. Basic Genetic Algorithm concepts are discussed and previous work proving their strength is reviewed. Feature Selection concepts and terminology are provided. Finally previous work that utilizes Genetic Algorithms in Feature Selection is examined. **Chapter 3** explains the proposed method, while providing details on implementation and tools used. **Chapter 4** explains the experimental setup and details of the experiments provided. **Chapter 5** concludes the thesis and discusses future work.

CHAPTER (2): BACKGROUND

This chapter provides background on the three main concepts used in this work, by discussing the Viola-Jones object detector. Details on Viola-Jones Rapid Object Detector and some of the research that aims to enhance this detector are provided, since the enhancement of the training time of this detector is the main objective of this work. After that an overview on GAs and their main concepts are discussed, with some previous work that sheds light on the success and wide usage of these algorithms. Feature Selection is then mentioned, with their categorization and main concepts. Finally previous work that combined both Feature Selection and Genetic Algorithm is presented.

2.1 Object Detection background

As this research area is relatively new, as mentioned by Hjelmås et al. (Hjelmås & Low, 2001) that the face detection problem has attained little attention before 1998 (Amit, Geman, & Jedynak, 1998). This is apparently not the case now since this area has gained more attention by the time that Herman et al. conducted their survey (Hjelmås & Low, 2001). Since then many researchers have focused on this area. Scientists have been working and contributing to detectors over the past decade. Viola-Jones is an example of widely used detectors. This section will provide a brief introduction on this detectors; since it provides the basis for this research.

2.1.1 Viola-Jones Rapid Object Detector

Viola et al. devised a rapid object detector, with 3 major contributions. The first contribution is that they provided an image representation called the integral image that allows the features to be evaluated fast. Their second contribution is that they devised a method for construction of the classifier through the selection of important features using Adaboost. Their third contribution is successively combining complex classifiers in a cascade structure which allows for fast detection on the test images (Viola & Jones, 2001).

The basic and main 3 components of the Viola-Jones classifiers are:

- The Haar features
- Integral Image
- Adaboost
- Cascaded structure

2.1.1.1 Haar Features

The use of features has proven to be better than using pixels, as features proved a set of comprehensive information that can be learned by machine learning algorithms. Features reduce the in-out class variability compared to that of the raw pixels (Lienhart & Maydt, 2002; Viola & Jones, 2001). This is in general, a clear incentive that provides more reasons to use features instead of raw data. For this particular system a critical issue is speed of calculation and the features operate much faster than raw pixels (Viola & Jones, 2001). The Haar features used are shown in Figure 2-1. The value of the feature is obtained by subtracting the sum of the pixels in the white region from the sum of the pixels in the black region. The four features used are those that are best for distinguishing upright front-facing faces. For example, feature (c) in Figure 2-1 can detect the nose area as its lighter than the eyes and feature (a) can detect eyes as the eyes region is darker than the region under it (Viola & Jones, 2001). For each image, each of the four Haar features is computed in all possible sizes and all possible locations which provide a huge number of features.

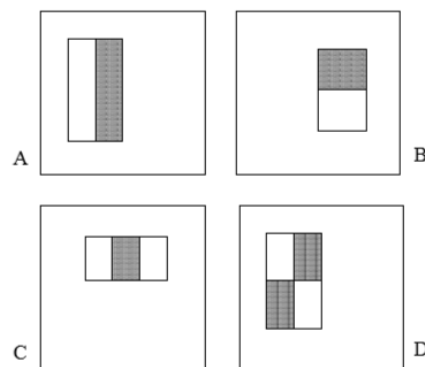


Figure 2-1: Haar features relative to the enclosing detection window (Viola & Jones, 2001)

2.1.1.2 Integral Image

Viola et al. introduced a new concept called the integral image in order to facilitate the computations of features since there are a lot of them. Any position in the integral image x, y is the sum of all the pixels above and to the left of x, y inclusive (Viola & Jones, 2001). Figure 2-2 shows the illustration of the integral image. For example, the value of location 1 in the

integral image is the sum of pixel values of rectangle A. Similarly the value of location 2 in the integral image is the sum of pixel values of rectangle A and B. The value at location 3 is A+ C. As for the sum of pixel values in rectangle D, it can be obtained by subtracting the value at location 2 and 3 from the value at location 4 then adding the value at location 1, as its going to be subtracted twice while subtracting both 2 and 3, since the value at 1 is contained in both 2 and 3. The equation of obtaining the pixel values at rectangle D is $4 + 1 - (2+3)$. The integral image reduces the calculation cost of pixels as it can calculate the sum of pixel values at any given rectangle by 4 array accesses at most.

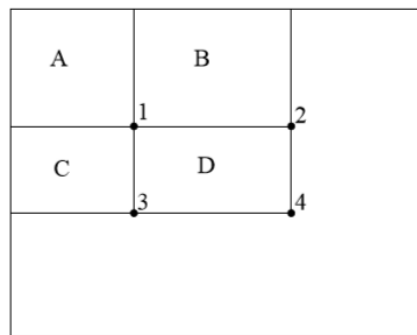


Figure 2-2: Integral image illustration(Viola & Jones, 2001)

2.1.1.3 Boosting

The authors chose Adaboost as a method to obtain their strong classifier. “Boosting is an approach to machine learning based on the idea of creating a highly accurate prediction rule by combining many relatively weak and inaccurate rules.” (Schölkopf, Luo, & Vovk, 2013). Adaboost, which was proposed by Freund and Schapire (Freund & Schapire, 1995), has been the first practical boosting algorithm and is still widely used in many applications (Schölkopf et al., 2013). Adaboost is simple and adaptive (Dezhen & Kai, 2008) yet it operates in a brute force manner, passing by all the set of features multiple times. Figure 2-3 explains the Adaboost algorithm, where each round of boosting selects one feature from the set of all possible features.

- Given example images $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \dots, T$:

1. Normalize the weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

so that w_t is a probability distribution.

2. For each feature, j , train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t , $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
3. Choose the classifier, h_t , with the lowest error ϵ_t .
4. Update the weights:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

- The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

Figure 2-3: Adaboost Algorithm (Viola & Jones, 2001)

The general idea of the algorithm works as follows:

For a number of iterations T :

- Pass through the set of all possible features and calculate the error of each one on the given images.
- Choose the best feature (the one with the lowest error) as the first weak classifier.
- Update the sample images and their corresponding weights, by putting more weights on the wrongly classified images.

- Go through the next iteration, until it finds the set of best features to be used in classification.

As shown from Figure 2-3 the weights are updated as a function of the error produced by the chosen classifier. In other words, the samples that has been misclassified by the chosen classifier are given more weight. These weights are used to inform the training of the weak classifiers i.e, the classifier that correctly classifies samples with higher weights are considered to be of better performance than the other classifiers.

2.1.1.4 Cascade Classifier

One of the important contributions of (Viola & Jones, 2001) is the cascaded classifier. This structure of the classifier allows for better accuracy while radically reducing the time consumed in detection (Viola & Jones, 2001). The cascaded classifier is a stage classifier where the thresholds vary. The first stages have a low threshold, thus detecting all the true positive while eliminating the strong negatives, before the more complex classifiers are called to achieve less false positives. Figure 2-4 provides a description of this classifier.

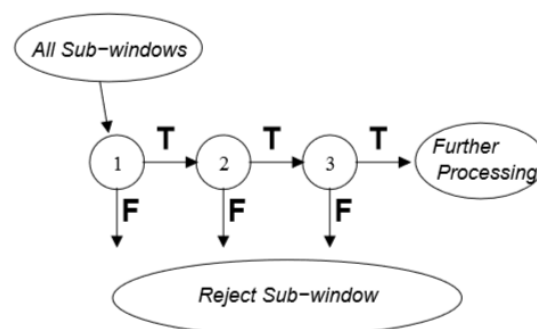


Figure 2-4 Schematic description of a detection cascade(Viola & Jones, 2001)

From Figure 2-4 it is clear that a series of classifiers are applied to every sub-window. The initial classifier is able to eliminate a huge number of negative examples with little processing. The following stages of classifiers then eliminate additional negatives, yet they apply more computations. After several stages of processing the number of sub-windows are drastically reduced (Viola & Jones, 2001).

2.1.1.5 Results

The resultant classifiers, on which the authors of (Viola & Jones, 2001) trained and based their experiments on is a cascaded one of 38 layers. The training set consisted of a set of 24X24 pixel images, of which 4916 faces and 9544 non faces. Within these non faces there are 350 million sub-windows and the total number of features is 6061. This detector was tested on the MIT+CMU frontal faces test. This set has a total of 130 images with 507 labeled frontal faces. The results are shown in the Receiver Operator Curve (ROC) in Figure 2-5.

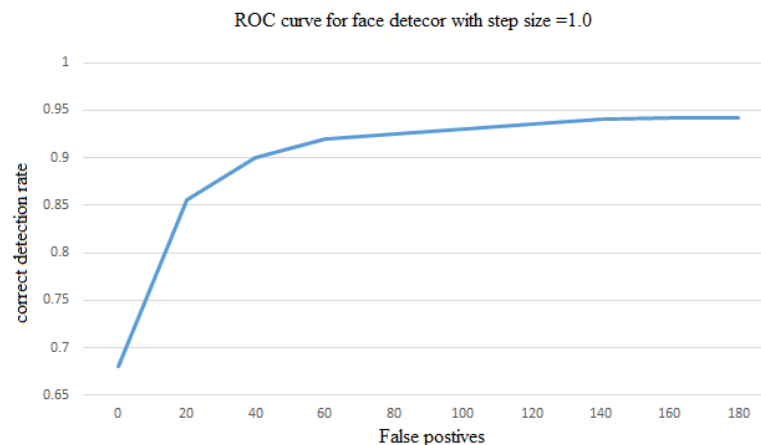


Figure 2-5: Roc Curve for detector on MIT+CMU dataset (Viola & Jones, 2001)

2.2 Enhancements over Viola-Jones

Some of the researchers used the Viola-Jones algorithm as a base for their research then proposed and implemented their concepts to provide even more powerful detectors. Li et al. proposed new enhancements that include SVMs and stopping criteria to detect more objects instead of just frontal-upright faces. Lienhart et al. proposed the increase of Haar features used. This section will give more details about both approaches.

2.2.1 The use of SVMs and new stopping criteria

Li et al (Q. Li, Niaz, & Merialdo, 2012) have achieved 3 major contributions. They used multiple feature images instead of just gray ones used by Viola-Jones, They devised a way to avoid the non-converging in training the classifier. They also outputted a weighted value as a confidence measure to whether the test image contains the desired objects or not.

The training data is preprocessed and a set of 6 image features are produced, the 6 types are: Gray image, Local Binary Patterns (LBP), EDGE, L-channel, A-channel and B-channel images. They tackled the problem of the non-converging training set in the cascaded classifier since the stopping criteria is a preset false alarm rate which sometimes is never reached. In order to fix this, they introduced a new stopping criterion, which is the maximum variance ratio (R) between the score of the positive and the negative training images. The main idea is to separate the positive and negative as much as possible and keep the inner variance of each class small. The score is defined as “the stage sum of the last stage classifier of a survived image patch. Stage sum is the cumulative sum of Haar like features convolved with the image patch (Q. Li et al., 2012). If R keeps increasing the training continues, the training classifier will converge since R will not be increasing all the time. As for the detection part, a key point based SVM is incorporated to get a confidence measure (to weigh the output score). The authors tested their algorithm on the TRECVID 2011 development dataset, they chose four objects which are: Computers, Scene_Text, Telephone and Hand. In all of these categories their algorithm performed much better than the Viola-Jones implementation in OpenCV. The results can be seen in Figure 2-6.

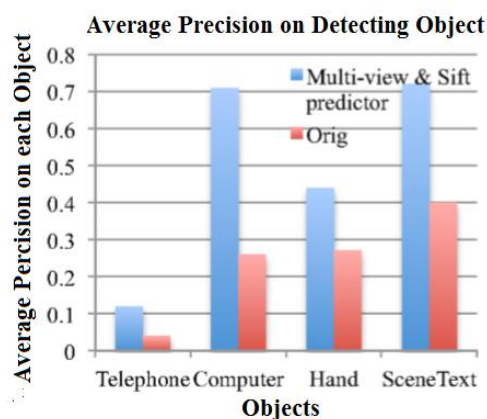


Figure 2-6 Object detection average precision on selected objects (top 40 in 113 test image)(Q. Li et al., 2012)

2.2.2 Increasing Haar Features

The authors of (Lienhart & Maydt, 2002) approach in enhancing the Viola-Jones Rapid Object Detector differs from the approach pursued by the authors of (Lienhart & Maydt, 2002). They wanted to enhance Viola-Jones by increasing the Haar features to more than the 4 used in the original work. They used 45 degrees rotation of feature that adds domain knowledge to the learning framework. These features can be seen in Figure 2-7.

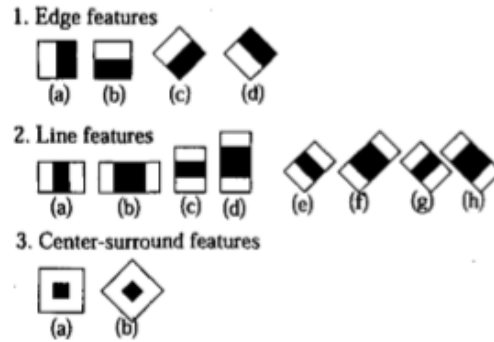


Figure 2-7 Extended set of Haar features black and white regions have negative and positive weights (Lienhart & Maydt, 2002)

Increasing the type of features from 4 to 14 substantially increased the number of generated features per image. Table 2-1 gives a summary of the number of features inside a 24x24 image window per feature prototype from Figure 2-7. The upright features can be computed fast by the integral image (Lienhart & Maydt, 2002). As for the rotated ones the authors created a rotated summed area table to enable them to calculate the value of the rotated features fast. The results shows that with these extended set of features the classifier performs better than the original one that had only 4 features, they also had comparable computation complexity. Figure 2-8 shows the ROC curve of the 2 classifiers with 12 stages.

Table 2-1: Number of features inside a 24X24 image for each prototype (Lienhart & Maydt, 2002)

Feature Type	Number of Features
1a; 1b	43,200
1c; 1d	8,464
2a; 2c	27,600
2b; 2d	20,736
2e; 2g	4,356
2f; 2h	3,600
3a	8,464
3b	1,521
Sum	117,941

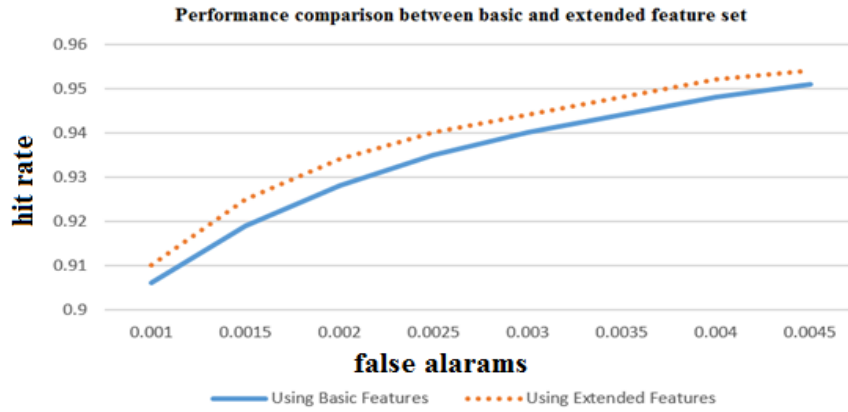


Figure 2-8: Basic versus extended features set. (Lienhart & Maydt, 2002)

2.3 Feature Selection.

A feature can be defined as measurable property of the data being observed (Chandrashekar & Sahin, 2014). Feature Selection is the process of reducing the whole search space into a sub-set of relevant features. This helps in removing noise and irrelevant features reducing time while providing good prediction results (Chaaroui & Flórez-Revuelta, 2013; Chandrashekar & Sahin, 2014; Jeong, Shin, & Jeong, 2014; Lee & Lee, 2014; Liang, Tsai, & Wu, 2014; Oreski & Oreski, 2014; Santana, Silva, Canuto, Pintro, & Vale, 2010; Vignolo, Milone, & Scharcanski, 2013; Xia, Zhuang, & Yu, 2014; B Xue et al., 2016; Bing Xue, Fu, & Zhang, 2014; Yusta, 2009). The need for feature selection methods arose due to the availability of high dimensional data with hundreds or thousands of attributes. In other words Feature Selection methods are ways to solve the curse of dimensionality (Powell, 2007).

Feature Selection techniques are divided into 3 main categories which are Wrappers, Filters and Hybrid (Embedded) methods. (Chaaroui & Flórez-Revuelta, 2013; Liang et al., 2014; Oreski & Oreski, 2014; Santana et al., 2010; Vignolo et al., 2013; Yusta, 2009). Table 2-2 provides a summary for these three categories.

2.3.1 Filters

Filter techniques rely on the intrinsic properties of the data without involving a classification technique (Oreski & Oreski, 2014). They use variable ordering techniques as criteria for selection by ordering. Variables that are below a certain threshold and excluded from the original variable set (Chandrashekar & Sahin, 2014). A basic criteria of the chosen

feature is to have useful information about the classes of the data. This property can be called feature relevance, which is the ability of this feature to discriminate between classes. Feature reference can be defined as “feature can be regarded as irrelevant if it is conditionally independent of the class labels.” (Chandrashekar & Sahin, 2014). Some examples used for filter techniques are: Correlation criteria, mutual information (Chandrashekar & Sahin, 2014).

The advantages of the filter methods are: That they are computationally efficient, avoids overfitting and has proven to work well on certain datasets (Chandrashekar & Sahin, 2014). They don't rely on learning algorithms which are biased and change the data to fit the learning algorithm. The disadvantages of some of these methods are that they don't consider the feature in relation with other features. In other words, features that are not informative on their own but give valuable information when combined with other features might be disregarded. (Chaaroui & Flórez-Revuelta, 2013; Chandrashekar & Sahin, 2014; Oreski & Oreski, 2014; Santana et al., 2010; B Xue et al., 2016).

2.3.2 Wrappers

Wrapper methods use classifier predictions as a fitness measure for the sub-set of features (Chaaroui & Flórez-Revuelta, 2013; Chandrashekar & Sahin, 2014; Jeong et al., 2014; Lee & Lee, 2014; Liang et al., 2014; Oreski & Oreski, 2014; Santana et al., 2010; Vignolo et al., 2013; Xia et al., 2014; B Xue et al., 2016; Bing Xue et al., 2014; Yusta, 2009). Since evaluating multiple subsets is an N-P hard problem, Wrappers become computationally expensive especially with large datasets. Wrappers often utilizes metaheuristics like GAs, Particle Swarm Optimization (PSO) and Ant Colony optimization (ACO). Though Wrappers are generally more accurate than Filters, their main drawback is computational complexity since each sub-set of features is passed to a classifier for training and testing to in order to calculate the accuracy (Chaaroui & Flórez-Revuelta, 2013; Chandrashekar & Sahin, 2014; Oreski & Oreski, 2014; Santana et al., 2010; B Xue et al., 2016). Another drawback of these methods which use classifier prediction as the objective function is that these classifiers are prone to overfitting. Overfitting happens when the classifier lacks the ability for generalization and only acts well on the data used for training. In this case the classifier will be biased and provide poor classification results (Chandrashekar & Sahin, 2014).

2.3.3 Embedded

Embedded methods are hybrid methods that try to combine the advantages of both Wrappers and Filters. It aims to reduce the time taken by wrappers in re-classifying the sub-

sets by incorporating the subset selection while training. In (Chandrashekar & Sahin, 2014) some of the embedded methods techniques are provided and discussed.

Table 2-2: Summary of feature selection techniques

	Filters	Wrappers	Embedded
Definition	Relies on general properties of data.	Uses machine learning approaches as black boxes to score features.	Combines both the filter and wrapper approach.
Advantages	Computationally more efficient in comparison to wrapper approach.	Provides more accurate subsets than filters.	Tries to reduce the time taken by wrappers by including filters in the learning process
Disadvantages	Provides worse subsets.	Involves computational overhead to score features.	

2.4 Genetic Algorithms

This section provides background on Genetic Algorithms (GAs), their techniques and the processes involved such as mutation, crossover and selection methods.

2.4.1 Overview

Genetic Algorithms are heuristic mechanisms that are successful in solving many difficult problems. They can be considered the best solution for high complexity problems such as the combinatorial optimization (Tabassum & Mathew, 2014). GAs are most likely the first Evolutionary Computing (EC) technique to be widely applied to Feature Selection problems (B Xue et al., 2016). Genetic Algorithms (GAs) were first proposed by John Holland (Holland, 1975). They are optimizing procedures that are devised from the biological mechanisms of reproduction and evolutionary science (survival of the fittest) (Andrade & Errico, 2008; Harb & Desuky, 2011; Sun, Bebis, & Miller, 2004). In natural, individuals compete for scarce resources like food and shelter. The best individuals that are suited for this competition survive. Adaptation to the surrounding environment is essential for the survival of a species. The traits that uniquely characterizes the individual determines its chances for survival (Srinivas & Patnaik, 1994). These traits are encoded in each individual as genes. The best genes survive through generations by means of reproduction. In other words, fit genes enable individuals to survive, reproduce, consequently passing on their fit genes to their offspring, which in turn will pass through competition and those who survive will reproduce passing on their genes. This

will ensure that over the course of generations, the genes in the offspring are to be refined, providing fitter generations that are more capable of adapting to the environment.

GAs resemble survival of the fittest mechanism as they start with an initial random population that propose solutions to the problem at hand (Mitchell, 1998; Sun et al., 2004). Each individual in the population is encoded (usually as a string of bits) in order to mimic a chromosome. This denotes that the parameters of the problems are joined to form one possible solution chromosome. In order to evaluate the fitness of this individual, it's associated with a fitness score that governs its ability to survive through generations and breed. This score is provided by an objective that is set and is referred to as a fitness function. The main Idea of GAs is to get those individuals which prove to be promising, pass them on to the reproduction phase where their genes are combined and slightly modified to provide offspring. The fitness score controls the probability of an individual to be chosen; as the selection process usually favors fitter individuals i.e. individuals of a higher fitness score. This means that fitter individuals have the chance to be selected more than once and poorly performing individuals might not be selected at all. This is done several times and finally the fitness of the population should converge to an optimal or a near optimal solution.

The formation of new offspring in the reproduction phase is attained by means of crossover and mutation. Crossover is the process where genes of 2 individuals are combined to form a new individual. Mutation occurs by changing one gene of the produced children from the crossover phase. (Lillywhite et al., 2013). Crossover allows for fast exploration of the search space, while mutation increases the probability of the exploration of all of the search space. In other words, it decreases the probability of having an unexplored solution in the search space.

In brief, the basic operations that guide the GAs search are: Encoding, evaluating, selecting and recombining individuals. These operation are preformed iteratively (Sun et al., 2004). They stop at a predefined stopping criteria or when the given maximum number of iterations is reached. Figure 2-9 explains how GA works.

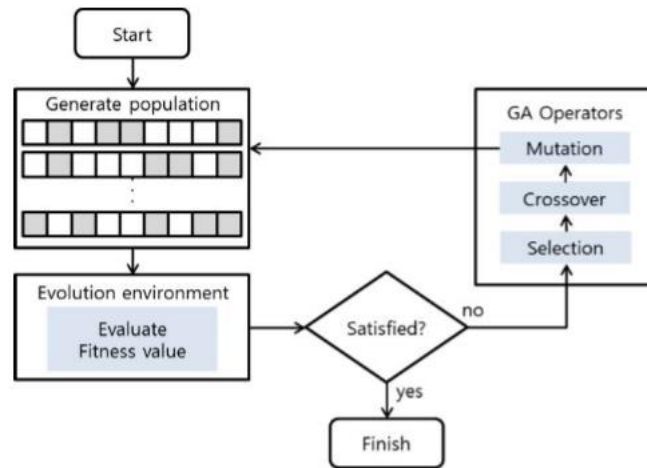


Figure 2-9: How GAs Work (Lee & Lee, 2014)

2.4.2 GA Details

2.4.2.1 Crossover Types

Crossover is the process where fit individuals are combined to form new individuals that will be a part of the next generation. This process helps in the exploration of the search space. Crossover has many forms; the most important ones are discussed in the following subsections.

2.4.2.1.1 One-Point Crossover

One-point crossover is the simplest form of crossover. In this type, a point is chosen randomly and the 2 parent chromosomes are cut at this point. Then the sections after this cut, are exchanged to form the 2 children (Hasaengebi & Erbatur, 2000; Magalhães-Mendes, 2013). Figure 2-10 visually illustrates the one-point crossover technique.

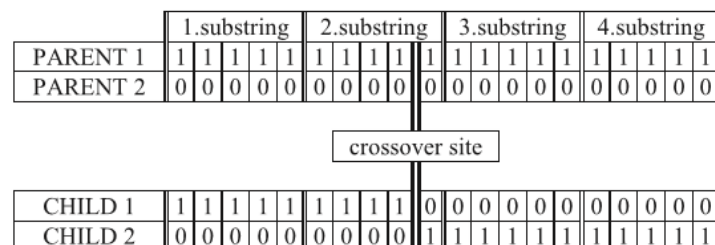


Figure 2-10: Single point Crossover (Hasaengebi & Erbatur, 2000)

2.4.2.1.2 Two-Point Crossover

Two-point crossover is when the 2 parents are cut at 2 different points. It is done by either swapping the inner portions (genes between the 2 points) or the outer portion, since both options provide the same results (Hasançebi & Erbatur, 2000; Magalhães-Mendes, 2013). Figure 2-11 illustrates the two point crossover.

	1.substring	2.substring	3.substring	4.substring
PARENT 1	1 1 1 1 1	1 1 1 1 1	1 1 1	1 1 1 1 1 1
PARENT 2	0 0 0 0 0	0 0 0 0 0	0 0 0	0 0 0 0 0 0
2. crossover site		1. crossover site		
CHILD 1	0 0 0 0 0	1 1 1 1 1	1 1 1	0 0 0 0 0 0
CHILD 2	1 1 1 1 1	0 0 0 0 0	0 0 0	1 1 1 1 1 1

Figure 2-11: Two-point crossover (Hasançebi & Erbatur, 2000)

2.4.2.1.3 Multi-point crossover

Multi-point crossover is an extension to the two point crossover where the two parents are cut at 3 or more points and the portions between these points are exchanged. This type of crossover helps the exploration of more parts of the search space (Hasançebi & Erbatur, 2000).

2.4.2.1.4 Uniform Crossover

In this type of crossover a bit mask of the same length of the individual (chromosome) length is randomly created. Each bit of the mask determines the gene would be copied from which parent into the child. 1 means the gene will be transferred from parent number one, 0 indicates that the gene will be copied from parent number two (Hasançebi & Erbatur, 2000). Figure 2-12 illustrates the uniform crossover.

	1.substring	2.substring	3.substring	4.substring
PARENT 1	1 1 1 1 1	1 1 1 1 1	1 1 1 1 1	1 1 1 1 1
PARENT 2	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
CROS. MASK	1 1 1 1 1	0 0 0 0 0	1 0 1 1	0 0 1 0 0 1
CHILD 1	1 1 1 1 1	0 0 0 0 0	1 0 1 1	0 0 1 0 0 1

Figure 2-12: Uniform crossover (Hasançebi & Erbatur, 2000)

2.4.2.2 Selection Mechanisms

Selection mechanisms are crucial as they choose the individuals that will participate in the next generation. If the best individual is always chosen, premature convergence will occur (Andrade & Errico, 2008). Premature Convergence is when a highly fit gene (but not optimal) dominates generations, causing the population fitness to converge to a local maxima. As a form of avoiding this problems many selection techniques where devised.

2.4.2.2.1 Tournament Selection

Tournament selection is the most commonly used selection mechanism, due its simplicity and straight forward implementation (Goldberg & Deb, 1991; Noraini & Geraghty, 2011). It's achieved by randomly selecting a number of individuals from the population. These individuals compete and the fitter one is chosen to participate in the next generation. The number of competing individuals is called tournament size and is usually set to two (Noraini & Geraghty, 2011). Tournament section gives each individual the chance to participate, thus preserving diversity though this might lead to slower convergence. Tournament selection has several advantages which include efficient time complexity, especially if implemented in parallel, low susceptibility to takeover by dominant individuals and no requirement for fitness scaling or sorting. (Baker, 1985; Goldberg & Deb, 1991; Noraini & Geraghty, 2011). Figure 2-13 shows an illustration of the tournament selection mechanism.

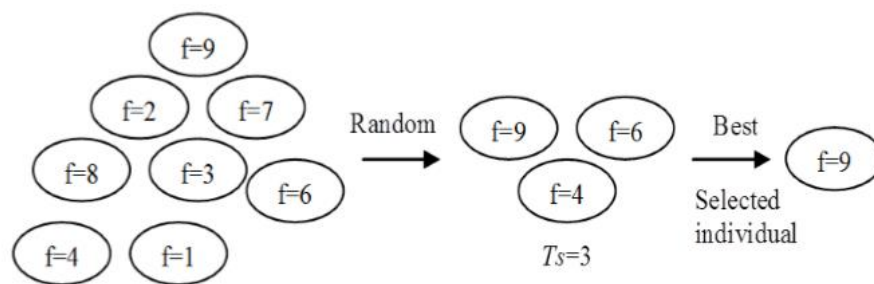


Figure 2-13 Tournament selection(Noraini & Geraghty, 2011)

2.4.2.2.2 Roulette Wheel Selection

- Proportional Roulette Wheel Selection

In proportional Roulette Wheel Selection, the probability of an individual being chosen is directly proportional to its fitness value, i.e the fitter individual has a higher probability of being selected. The probability of choosing a parent is analogous to a roulette wheel and the size of its segments are proportional to each parent's fitness. Thus parents with higher fitness have larger segments on the roulette wheel, consequently more chance of being chosen. The probability of choosing an individual is calculated by equation (2.1) (Noraini & Geraghty, 2011). Where p is the probability of choosing individual, f is the fitness value of individual. N is the total number of individuals the population.

$$p_i = \frac{f_i}{\sum_{i=0}^n f_i} \quad (2.1)$$

This type of selection mechanisms gives chance to all of the individuals in the population, preserving the diversity. Yet, it gives higher probability to fittest individuals, which may cause these individuals to dominate populations fast which eventually leads to premature convergence, and loss of genetic diversity. For example if the population contains two fit individual and the rest of the population has poor fitness, these two fit individuals will dominate the population quickly. On the other hand, if the whole population is of similar fitness, the population will face difficulty in evolving to a better solution since both probabilities of fit and unfit individuals are similar. (Noraini & Geraghty, 2011)

- Rank-Based Roulette Wheel Selection

At the beginning the individuals are sorted according to their fitness values, and the probability of one being chosen is based on its rank in the sorted array. Rank based selection is not influenced by “super-individuals” or the spread of fitness values. Rank-based selection depends on a mapping function that maps the indices of the individual in the sorted list according to their fitness values. Thus, the performance of this technique depends heavily on the mapping function

chosen. (Noraini & Geraghty, 2011). Figure 2-14 shows the Roulette Wheel selection mechanism.

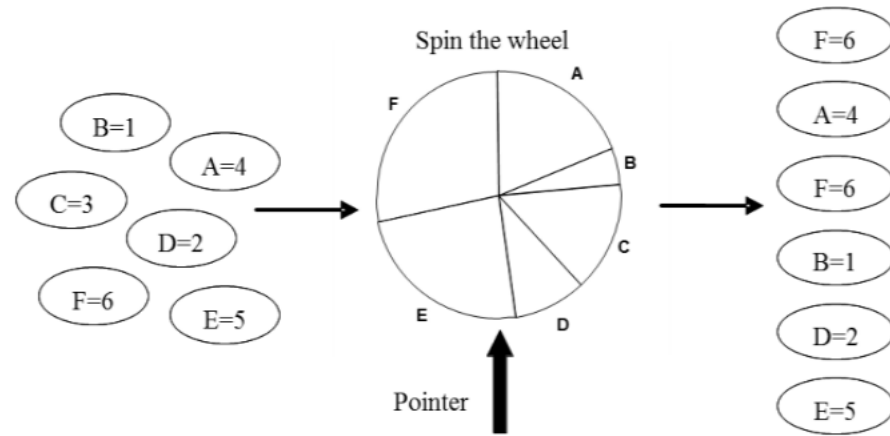


Figure 2-14 Roulette wheel selection (Noraini & Geraghty, 2011)

2.4.2.2.3 Deterministic Sampling

In deterministic sampling the average fitness of the population is calculated. After that, the fitness value of each individual is divided by the average fitness of the population and the integer part is stored. If the integer is greater than 1, the individual is chosen, else the individual will not be selected to participate in the next generation. The rest of the population size is then filled by choosing individuals with greater fractions. (Andrade & Errico, 2008)

2.4.2.2.4 Stochastic Remainder Sampling

Stochastic random sampling is identical to deterministic random sampling where the individual is chosen based on the integer part resulting from the operation of dividing the individual population by the average population. The rest of the population size is filled by the means of a roulette wheel selection.

2.4.2.3 Mutation

Mutation is another form of exploring the search space, it reduces the probability of having an unexplored solution. Mutation is mainly concerned with changing a one gene of the child produced by the crossover process, according to a preset probability.

2.4.3 Strength of GAs

GAs are powerful optimization algorithms that have proven their success in many fields. Tabassum et al. Mentioned that “It was proved that genetic algorithms are the most powerful unbiased optimization techniques for sampling a large solution space” (Tabassum & Mathew, 2014). In their work (Ferri & Pudil, 1994) highlighted the point of strength of the GAs which is the ability to perform the search in a near optimal region due to the inherit randomizations used in the search. In this sub-section general works on Genetic Algorithms is reviewed.

2.4.3.1 Circle Detection Using GAs

Ayala-ramirez et al. proposed a method to detect circles in an image using GAs. They preprocessed the image by a Sobel filter and got all the edge points in an image, then they took 3 points at a time to test if they formed a circle (Ayala-ramirez, Garcia-capulin, Perez-garcia, & Sanchez-yanez, 2006). They generated a circle with these 3 points and found virtual points that lie on this circle. After that, they examined how many of these virtual points actually exist in the edge points they got after applying the Sobel filter, considering this as the fitness function of the GA. This method has been tested on both synthesized images where the authors put random circles in an image, and on natural images taken by a digital camera; in both cases this method achieved good accuracy with a worst case scenario of 92%, in a short amount of time as shown in Table 2-3 and Table 2-4.

Table 2-3 Discrimination results on synthetic images (Ayala-ramirez et al., 2006)

Image	Shapes in image	Time (s)	Accuracy rate (%)
1	2	1.64	100
2	3	1.56	100
3	4	0.28	100
4	12	3.81	92
5	11	2.39	100

Table 2-4 Discrimination results on natural images (Ayala-ramirez et al., 2006)

Image	Shapes in image	Time (s)	Accuracy rate (%)
1	2	0.93	92
2	3	0.34	100
3	4	0.38	100
4	9	0.43	100
5	9	5.05	98

2.4.3.2 Feature Construction Using GAs

Lillywhite et al. devised a system that uses genetic algorithms to construct features, as little research is concerned with the point of feature construction (Lillywhite et al., 2013). They used Adaboost to build a strong classifier from a series of weak classifiers. Their features; which they called ECO features, are generated using a Genetic Algorithm, that creates an ordering of basic transformations like Sobel operator, Canny edge, Pixel statistics, Histogram, Gaussian blur...etc. the initial population is some vectors that are produced after the application of a series of these transformations on a sub-image $I(x_1, y_1, x_2, y_2)$.

After having the initial population, a genetic algorithm is applied with mutation and crossover processes. The genes are the elements of an ECO feature which includes the transformation type and the transformation parameters. They associated a weak classifier with each ECO feature in order as a means for calculating a fitness score. This fitness score is associated with how well the feature identifies an object in a small training set. The weak classifier is a single perceptron that maps the feature vector to a binary classification through a weight and bias. The weights are updated through the error rate, which is subtracting the perceptron output from the original image classification. The fitness score equation depends on the number of true positives, false negatives, true negative and false positives.

The following step that takes place after the Genetic algorithm has found good ECO features is to build a strong classifier based on the weak classifiers (the perceptrons in this case) using Adaboost algorithm.

This method has been tested against previously published papers using same dataset (Caltech dataset) for comparison and proved to be significantly more accurate overall. These results are shown in Table 2-5.

Table 2-5: comparison on Caltech dataset to other methods

Database	Method			
	Fergus [51]	Serre [52]	Schwartz [53]	ECO
Motorbikes	95.0	98.0	100.0	100.0
Faces	96.4	98.2	100.0	100.0
Airplanes	94.0	96.7	100.0	100.0
Cars	95.0	98.0	100.0	100.0

2.4.3.3 GAs versus Simulated Annealing in Mean Cut sizes

There exists other optimization methods that serve well, yet for some experiments GAs have proven to perform better. This might be due to the advantages of GAs, which are probabilistic and not deterministic, they work well with stochastic systems and have the ability to be better at avoiding to be stuck at a local maxima due to their parallelizable nature.

Manikas et al. provided a comparison in their paper between GAs and simulated annealing in the problem of optimizing the placement of the circuit's physical components on a chip (Manikas & Cain, 1996). The problem of circuit partitioning can be represented as a graph with a set of vertices V and a set of edges E the partitioning process splits the circuit into groups of equal sizes and tries to find the group with minimal interconnections called a cutsize. They used 3 circuits and applied both the GA and simulated annealing, to find a proper solution.

From their experiment they concluded that GA preforms as good as, or even better than simulated annealing. Figure 2-15 shows the result of the carried out experiment, it shows that in 2 circuits GA was able to find a smaller (better) cutsize than the commonly used simulated annealing.

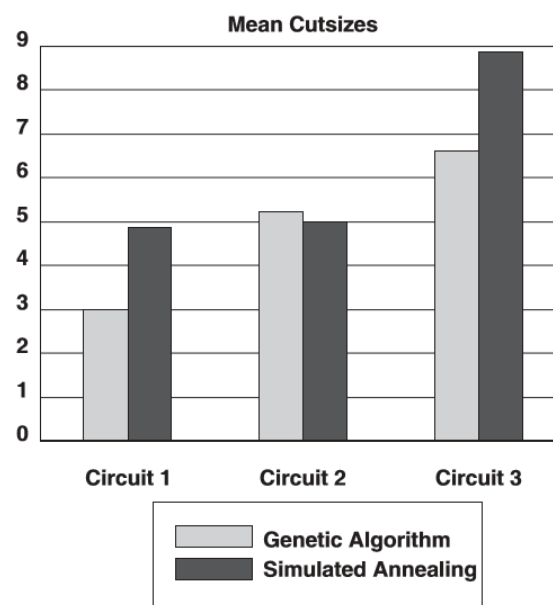


Figure 2-15: Comparison of mean cutsizes (Manikas & Cain, 1996)

2.5 Feature selection with GAs

“Genetic Algorithms (GAs), have been developed for solving feature selection problems due to their efficiency for searching feature sub-set spaces in feature selection problems”(Jeong et al., 2014). GAs are widely used in Feature Selection (Tsai et al., 2013). A lot of research has been done on the combination of GA with feature selection techniques and has been proven successful. In this section we discuss some of these works.

2.5.1 Work that utilizes GA with feature selection.

(Santana et al., 2010), (Oreski & Oreski, 2014) (Liang et al., 2014) experimented with filters, while (Sun et al., 2004) (Dezhen & Kai, 2008) (Chouaib et al., 2008) (R. Li, Lu, Zhang, & Zhao, 2010) (Harb & Desuky, 2011) (Jeong et al., 2014) (Oreski & Oreski, 2014) (Lee & Lee, 2014) (Liang et al., 2014) (Vignolo et al., 2013) used wrapper methods with GAs. (Vignolo et al., 2013) and (R. Li et al., 2010) used K-Nearest Neighbor as the black box classifier in the wrapper method. While (Chouaib et al., 2008), (Dezhen & Kai, 2008) , (R. Li et al., 2010), and (Harb & Desuky, 2011) used Adaboost as their classifier. SVMs have been used as classifiers in (Sun et al., 2004), (Lee & Lee, 2014), and (Liang et al., 2014). (Oreski & Oreski, 2014) and (Jeong et al., 2014) used Neural Networks as their classifier. The following discusses some research that use both GA and feature selection to solve different types of problems.

2.5.1.1.1 Comparing GA with other metaheuristic method in Feature selection

Sun et al. (Sun et al., 2004) used the powerful methods of GAs to select the best eigenvectors. They compared the use of GA with SBFS in Feature Selection. The SBFS is based on the 2 heuristic methods, which are the sequential forward selection (SFS) and sequential backward selection (SBS) methods. The GA results have been proven to improve detection results.

(Yusta, 2009) compared metaheuristic techniques including GA and SFBS along with other popular algorithms such as GRASP and Tabu search.

(Santana et al., 2010) Compared the use of GA with ACO in feature selection for building an ensemble of classifiers. They concluded that when using small ensembles (small number of individual classifiers), the best option is ACO, while for larger ones GA performed better.

2.5.1.1.2 Genetic Algorithm in feature selection with Adaboost

Chouaib et al (Chouaib et al., 2008) aimed to find the set of the most representative features using GAs, in order to decrease the detection time in hand-written digit recognition. Their results showed that for the majority of descriptors their feature set was significantly reduced up to 35% of the original set in multi-class problems.

Dezhen et al. (Dezhen & Kai, 2008) provided a post optimization technique to avoid the redundancy of classifiers. By doing so, they managed to increase the speed of classification by 110% due to reducing the number of features to 55% of the original set.

(R. Li et al., 2010) proposed the use of dynamic Adaboost with feature selection based on parallel GA, in image annotation, yet the Adaboost ensemble had better accuracy than the algorithm that included feature selection with GA.

(Harb & Desuky, 2011) used Adaboost ensemble with a post optimization process for feature selection using GA and applied it to intrusion detection. They concluded that their method effectively improved the results of the boosted classifier providing, better accuracy with fewer weak classifiers

2.5.1.2 Use of GA in feature selection in miscellaneous applications

(Charaoui & Flórez-Revuelta, 2013) proposed a human action recognition optimization using evolutionary feature sub-set selection and claimed to have achieved promising results, as they achieved perfect detection on their test dataset with a reduced feature set by approximately 47% on average.

(Oreski & Oreski, 2014) used Genetic Algorithm in feature in credit risk assessment, and proved that their technique provided promising results and that their classifier is a promising addition to existing data mining techniques.

(Lee & Lee, 2014) experimented with the same techniques in the problem of predicting heavy rain fall from big weather data, their experiment proved that their proposed approach had a similar accuracy when compared to original algorithm. Yet computation time was reduced 8 times due to the dimensionality reduction of the data.

(Liang et al., 2014) used several wrapper methods and included Particle Swarm Optimization PSO and GA and Filter methods like linear discriminant analysis (LDA), t-test, logistic regression (LR). They concluded that although it's hard to choose the best feature selection method for financial distress, the better wrapper method is the GA.

(Vignolo et al., 2013) investigated the use of feature selection with GA in face recognition and proved that their proposed approach enhanced the detection performance while reducing the representation dimensionality.

2.6 Summary

This chapter provided background on the basic areas used in this thesis. Viola-Jones Rapid Object detector, and some enhancements on it have been discussed. Feature selection, its categories and importance is provided. An overview on Genetic Algorithm is given. Finally research using both GAs in feature selection is examined.

Since the last section has proved the effectiveness of combining GA in Feature Selection with various problems and since the previous work was concerned with enhancing the accuracy or speed of detection regardless of the overhead posed on the training time. This work aims to examine the effects on increasing the speed of training using GAs in feature selection and how this might affect the accuracy in the Viola Jones Rapid Object Detector, with its cascaded classifier structure.

CHAPTER (3): PROPOSED APPROACH

The outcome of the proposed methodology is building a cascade classifier that is efficient and does not require too much time to train, without a significant effect on the detection accuracy. The sections of this chapter describe the methodology of building such classifiers, how to implement them and the tools used for achieving the required goal, since the basis of this methodology is to incorporate GAs in the training process of the Classifiers, the details of the GA used are to be discussed. Also, the training and testing details are discussed.

3.1 OpenCV

In order to integrate the use of GA, Open Source Computer Vision Library (OpenCV)(Itseez, 2015) was used. OpenCV is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms (“The OpenCV Reference Manual,” 2014) . OpenCV contains the implementation of the Viola-Jones cascade classifier in the form of 2 applications: `Opencv_haartraining`, and `Opencv_traincascade`. Table 3-1 provides a brief summary of both applications.

3.1.1.1 `Opencv_haartraining`

`Opencv_haartraining` supports only Haar features. The drawback of this application is that it has become obsolete and has been removed from newer versions of OpenCV.

3.1.1.2 `Opencv_traincascade`

`Opencv_traincascade` is the newer version of training a cascade classifier in OpenCV. This application supports Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG) along with Haar features. `Opencv_traincascade` also supports the use of Threading Building Blocks (TBB) for multi-threading in a multi-core environment.

Both of these applications store the trained classifier with different file formats. `Opencv_traincascade` is able to store the resultant classifier in the old format, yet none of the applications can load the other’s format to continue the training if the training was interrupted at any point.

Table 3-1 Comparison of OpenCV Application used to train a cascade classifier.

	Opencv_haartraining	Opencv_traincascade
Difference	Older version of cascade classifier implementation.	Newer version of cascade classifier implementation.
Advantages	<ul style="list-style-type: none"> • Less code, easier to manipulate, and add functions to. 	<ul style="list-style-type: none"> • Supports LBP in addition to Haar. • Supported in newer versions of OpenCV. • Supports multi-threading • Saves the saved classifier in both old and new formats
Disadvantages	<ul style="list-style-type: none"> • Obsolete (not supported in newer OpenCV versions. • Only saves and loads the old version of template for saving the classifier. 	<ul style="list-style-type: none"> • Only loads the old version of template for saving the classifier. • Lots of modules in the code, not well documented, thus harder to manipulate and add functions to.

As shown, Opencv_traincascade surpasses Opencv_haartraining in the advantages, thus opencv_traincacade was chosen to be modified by adding necessary functions, in order to implement the proposed idea.

3.2 GAdaboost Overview

The proposed method (Named: GAdaBoost) applies GA to select a set of features, to have Adaboost choose from, instead of going through the set of all possible features. The original Adaboost algorithm was proposed by Freund and Schapire (1995) the generalized version works as follows: For the training of each stage in the stage classifier, the algorithm passes through the set of all possible features and calculates the error of each feature on each given image. After that, it chooses the best feature (the one with the lowest error, i.e best classifies the image correctly) as the first weak classifier. It then updates the sample images and their corresponding weights, by putting more weights on the wrongly classified images. The procedure is repeated until the set of chosen features reaches a preset false alarm and hit rate set for classification.

Incorporating the use of GA will increase the training speed by avoiding the error calculation of the set of all possible features and only providing the Adaboost algorithm with a representative set of features, that have been chosen based on their classification power. This set of representative candidate features is to be prepared by the GA before the training of each stage in the final classifier. For example if the final classifier is to have 10 stages the added GA technique is to be repeated 10 times. The stage training utilizes Adaboost technique to choose multiple weak classifiers from the mentioned representative set, in order to reach the desired false alarm and hit rate preset for the stage. Figure 3-1 shows a block diagram that explains the proposed GAdaBoost technique.

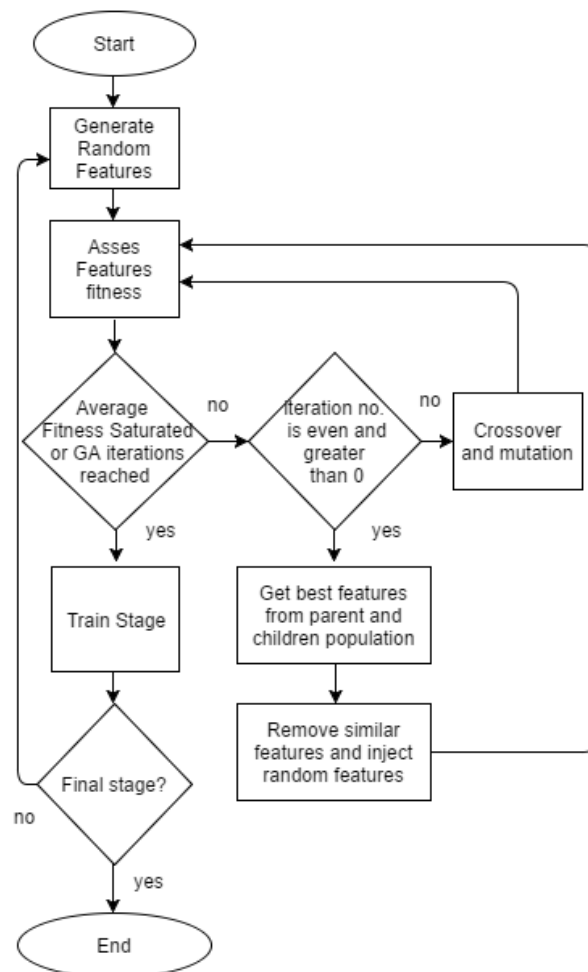


Figure 3-1 GAdaBoost algorithm flowchart

On the first iteration the GAdaboost chooses a preset number of features randomly to create the first generation of the given population size. Those randomly chosen features are marked so that they are not to be used again when more random features are to be generated. This is done to explore more of the set of all possible features. In order to assess the predictive power of these features, they are passed to a learning algorithm. The way this has been implemented is by creating a temporary (dummy) stage where the features are trained in the same way the original stage training works, i.e the dummy stage is an Adaboost training algorithm. The number of weak classifiers chosen by the Adaboost algorithm in the dummy stage is a variable that is preset. The Adaboost algorithm associates the features with scores that are a representation of their predictive power (how well they are able to correctly classify images). After that the best features are then selected and have mutation and crossover processes performed on them to get the next generation of an even better performing set of features. The new generation is then passed by a dummy stage for scoring. The process is repeated until the average fitness of the population saturates or a predefined number of iterations are reached.

As a form of exploring more of the set of all possible features, for each iteration with an even number (2nd, 4th, etc. generations) that is greater than zero, the best set of parents and their children produced are chosen. Then a spatial comparison is formed to remove the redundant features and random features are inserted instead to complete the population size. The spatial comparison is done using the pasacal criterion where two features are considered of spatial similarity if the ratio of the intersection of the two features over the union of the two is greater than 0.4. This method is described in more detail in section 4.2.3.2. The use of only even iterations entails that the spatial comparison is done on half the number of iterations (eg. for 50 iterations, the spatial comparison is done 25 times). The final set of features obtained by the GA is passed through a real stage where the weak classifiers selected by this stage are to be used in the resultant final classifier. The afore-mentioned technique ensures that the Adaboost algorithm will only evaluate the population size chosen instead of going through the whole set of features when selecting the weak classifiers of the resultant final stage classifier. The following is the pseudo code of the GAdaboost algorithm.

Algorithm GAdaboost

For each stage:

populationArray = **selectRandomFetures**(popSize)

fitnessValues = **Adaboost**(*populationArray*, *no WeakClassifiers*)

do

if (evenIteration and iteration !=0)

 newGeneration= **GetFittestIndivisuals**(previous2Generations)

 newGeneration = **specialCompare**(newGeneration)

if (newGneration.size<*popSize*)

fillPopluation(newGeneration, *selectRandFeatures*())

end if

else if

tournamantPlayers = **selectBestParents**(*populationArray*, *fitnessValues*)

newGeneration = **crossover**(*tournamantPlayers*)

 newGenration = **Mutate**(*newGeneration*)

end if

fitnessValues = **Adaboost**(newGenration, *no WeakClassifiers*)

While (! converged and !presetItertionNo)

reducedfeatureSpace= *newGeneration*;

TrainStage(*reducedfeatureSpace*);

end For

3.3 GA Details

3.3.1 Initial population

In the first GA generation a random set of features out of the whole search space is chosen, each feature is then mapped to a chromosome and added to the population. The individual chromosome in this case is just the representation of one feature. It represents here one feature which is one weak classifier, which can be considered the complete solution for each iteration. In other words, the complete solution in this case is just one weak classifier (feature) for each iteration. Figure 3-2 illustrates how the population is represented in the proposed method.

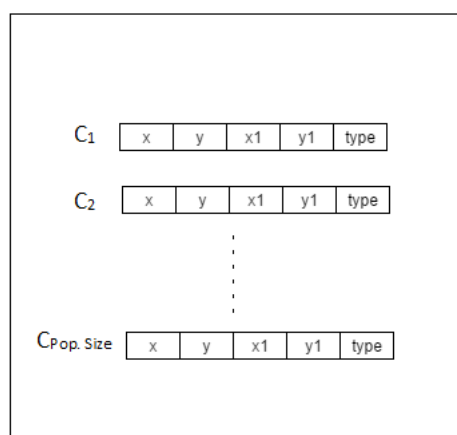


Figure 3-2 Population illustration

3.3.2 Chromosome Representation

Each chromosome represents one Haar feature. The chromosome is of an integer representation. The values of the chromosome are x, y, x1, y1, type, where x, y are the integer values of the upper left co-ordinates of the feature rectangle and x1, y1 are the integer values of the lower right corner of the feature rectangle. The type is an integer value from 0-4 where each number represents one of the Haar feature types used for upright frontal faces detection. 0, 1, 2, 3, 4 represent the Haar types of haar_x2, haar_y2, haar_x3, haar_y3 and haar_x2y2 respectively. Figure 3-2 explains the mapping of a feature of type haar_x2 to a chromosome in a given image. As shown, the chromosome carries decoded information about the type of the feature and its orientation in a given image, the way the chromosome is represented facilitates the mutation and the crossover processes which provide new features.

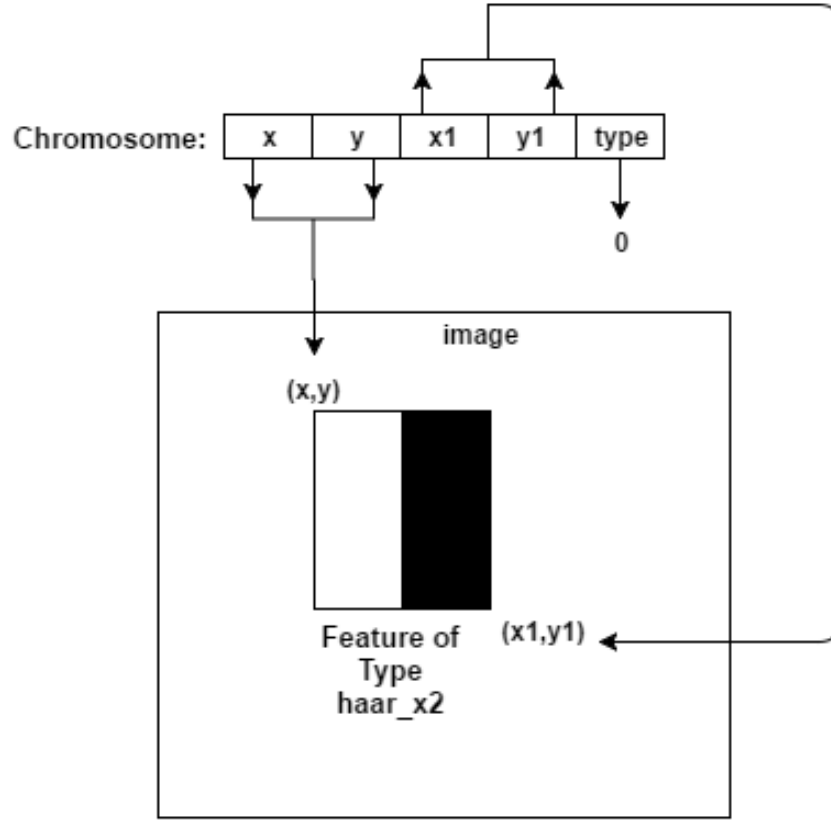


Figure 3-3 Chromosome to feature mapping

3.3.3 Fitness Function

The fitness function is a measure of how well this features splits between the negative and the positive images or in other words predictive power of this feature in classifying the images correctly. The OpenCV implementation uses decision stumps as weak classifiers, these decision stumps are Classification and Regression trees (CART). In CART the regression tree's best split quality is calculated by the minimization of Equation 3.1.

$$\sum_{i=0}^n (TR_i - PR_i)^2 \quad (3.1)$$

Where TR is the ground truth of the image, PR is the predicted response by the decision stump and n is the number of sample images. Yet, for simplicity the OpenCV traincascade developer mentioned that in implementation the minimization criteria is reduced to equivalent simpler maximization ones (Dimashova, 2012). In conclusion, the fitness of the feature used is the split quality measure provided by OpenCV's CVDTree class. Thus, in the implementation the best feature is the one the largest quality.

3.3.4 Selection Mechanism

The selection mechanism used in GAdaboost is the Roulette Wheel selection method. After each individual in the population is associated with a fitness function, the roulette wheel selects pairs from the population. Each pair chosen will undergo a crossover mechanism to produce child chromosome to participate in the next generation. This selection mechanism ensures that the individuals with higher fitness will have a higher probability of contributing to the next generation, since the probability of an individual of being chosen is directly proportional to its fitness. In other words, the larger the fitness value of the chromosome the larger its segment on the roulette wheel is, the higher the probability of its selection becomes.

3.3.5 Crossover

A simple one-point crossover has been used, in order to combine the genes of the 2 parents that are chosen according to their fitness by the selection mechanism to participate in the next generation. The one-point crossover allows the exploration of the search space by choosing one point then cutting the chromosomes at this point and exchanging the parts of the chromosome after the cut. This type of crossover has been chosen as it best fits the chromosome representation applied in GAdaboost. Since the chromosome is short and consists of the upper left corner, lower left corner and the type of the rectangular feature, the cut point has been chosen to be at the lower right corner of the two candidate features. Figure 3-4 illustrates the crossover done by the GAdaboost technique.

In order to ensure the validity of the produced children, some checks are preformed. Each child chromosome coordinates (coordinates of the feature it represents) are checked. If the produced chromosome is unvaild, for example, its lower right horizontal coordinate (x_1) is smaller then its upper left horizontal coordinate (x), the choromose is fixed by exchanging the 2 values. The same process is done for the vertical cooridinales of the produced child. Similariy, if the horizontal coordinates, or the vertical cordinales of the upper left corner and lower right corner of the feature the chromosome represnts are equal they are fixed. All the alterations to build a valid choromosme are done in accordance to the width and height of the images.

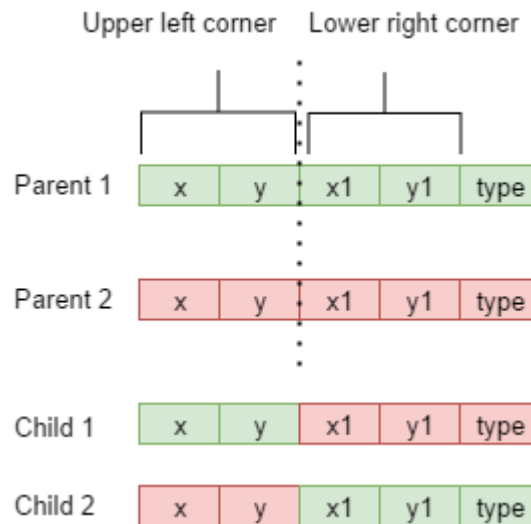


Figure 3-4 GAdaboost crossover illustration

3.3.6 Mutation

The type of the feature is highly dependent on its width and height. In order to reduce the time taken by validation of the correctness of the feature, the mutation is designed to assign the type to the feature according to how suitable this type is, given the co-ordinates of this feature. So for each produced child chromosome when check if the width is divisible by two then its assigned a haar_x2 type, if not we check if it is divisible by three and if it is then it is assigned a type of haar_x3, and so on.

CHAPTER (4) EXPERIMENTS

In this chapter the experiments done on the proposed method are discussed. In order to test a classifier 2 main processes are involved. These processes are: Training and testing. The first 2 sections describe these procedures and provides information that is beneficial for the experiments. Then we start our experiments by building a baseline, which is the normal rapid object detector proposed by Viola-Jones, without any additions of feature selection methods or GA. The second experiment examines the fitness values of the best individuals and the average fitness of the population through the generations of the Genetic Algorithm. The effect of varying the population size on the training time is observed in the third experiment. The fourth experiment builds a classifier with 20 iterations for the GA per stage and tests the accuracy of both the Caltech Web Faces and the Fddb datasets. The final experiment compares the baseline performances versus the performances of the 2 variations of GAdaboost with respect to time and accuracy. All the training occurred on the same computer with an Intel Core i7-4510U @ 2.00GHz processor and 8 GB RAM.

4.1 Training

Principally GAadaboost is concerned with training the cascade classifier faster with minimal loss of accuracy in the detection process of the resultant classifier. Training is the process where the application is given positive images, which are images that contain the desired object. The positive images are annotated. i.e the places where the desired object lies in these images is given. The training must also be given a set of negative images, which are images that do not contain the desired object. The training must be given a stopping criteria, in order to stop training when these results are reached on the validation set.

4.1.1 Positive images

The positive images used to train the detectors are acquired from the trainingfaces_24-24 .vec file provided with OpenCV. This file encompasses information about 1000 images containing upright frontal faces. A sample of these positive training images is shown in Figure 4-1.



Figure 4-1 Positive training images

4.1.2 Negative images

The negative images were picked randomly from the dataset of 101 objects developed at Caltech (Fei-Fei, Fergus, & Perona, 2004). Figure 4-2 provides a sample of the negative images used.

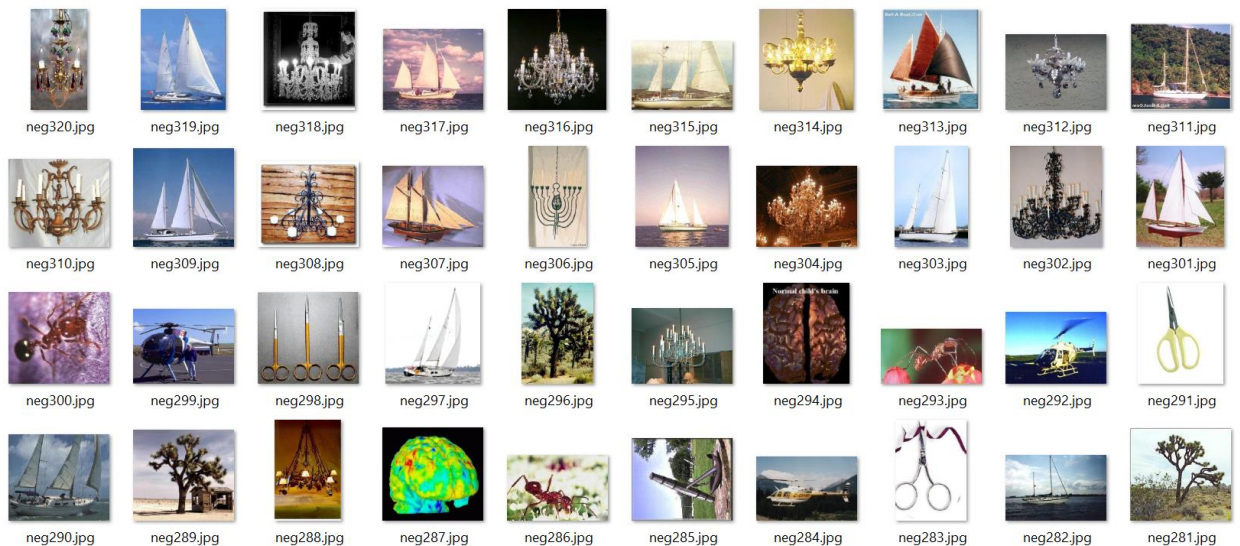
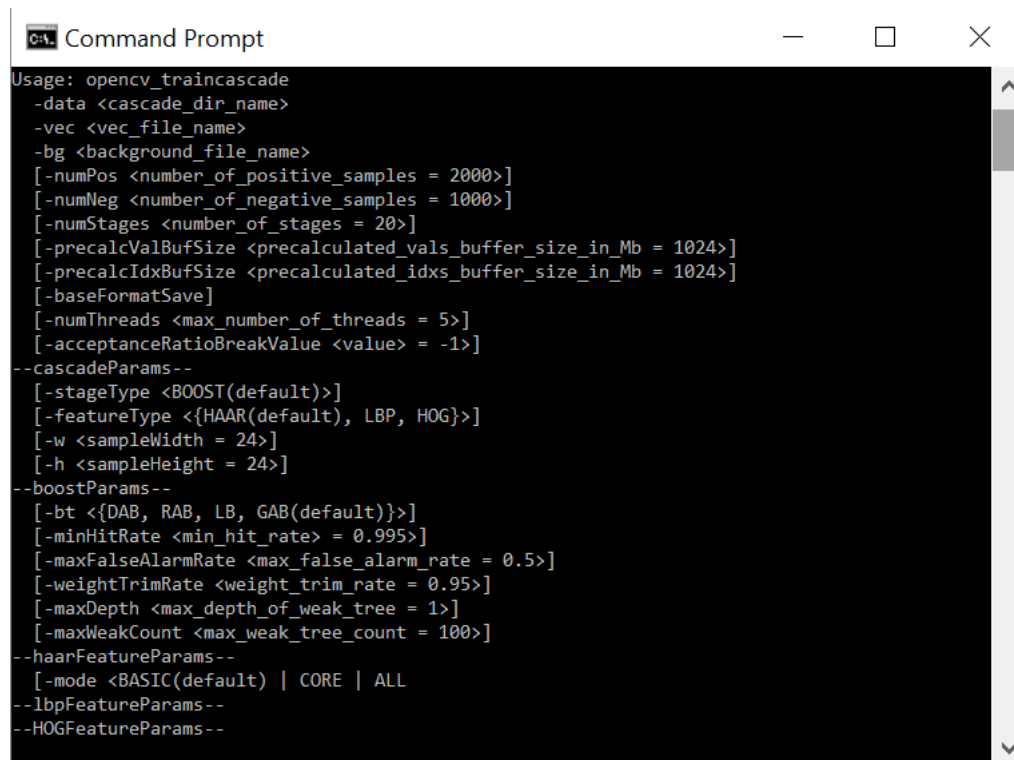


Figure 4-2 Negative images samples

4.1.3 Training Parameters.

To train a cascade classifier using `Opencv_traincascade`, some parameters values have to be set, such as the number of positive images and number of negative images per stage, the number of stages the final classifier will have, the type of features used and the hit and false

alarm rates per stage. Figure 4-3 provides a screenshot of the required parameters by the `opencv_traincascade` application.

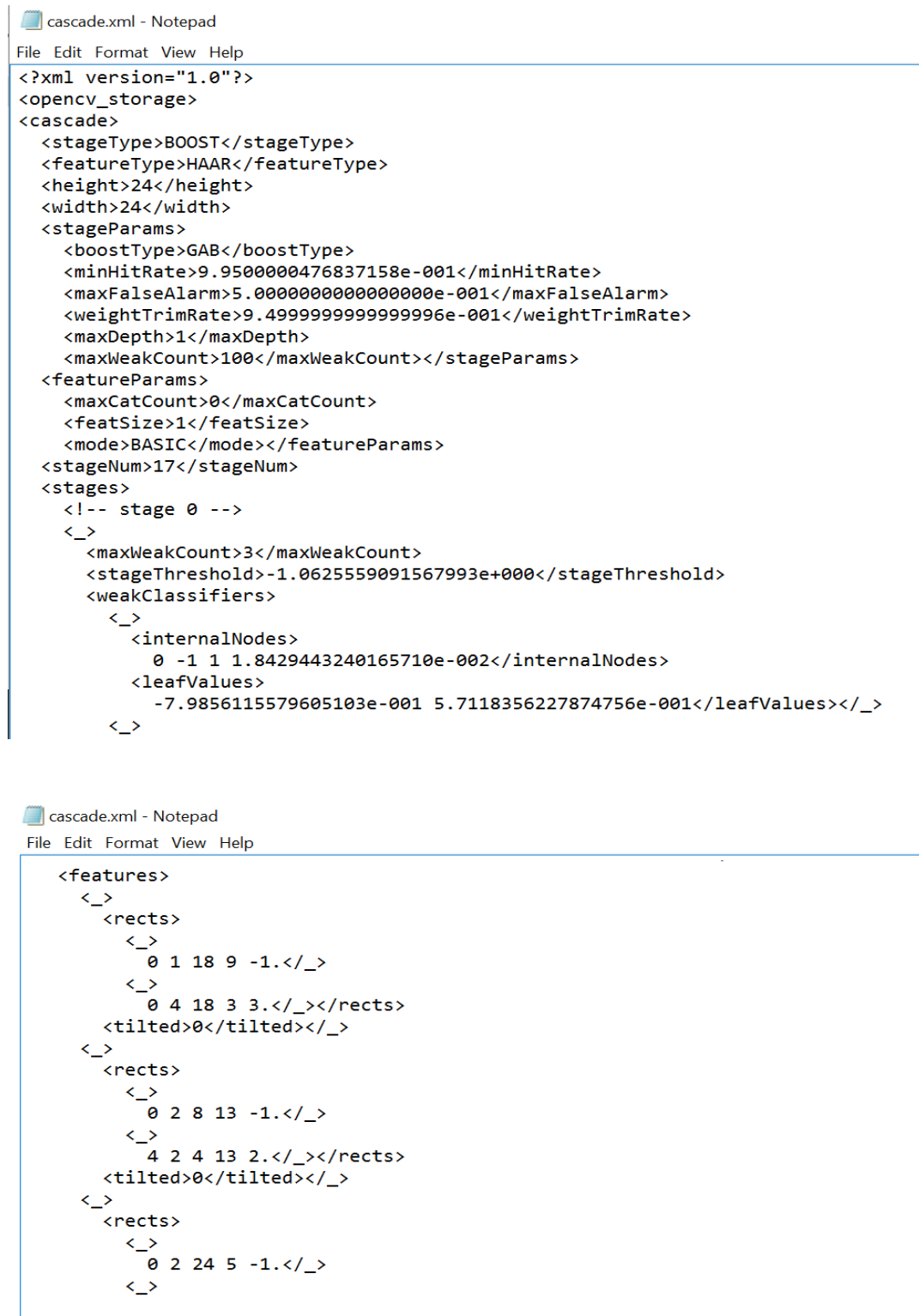


```
Usage: opencv_traincascade
-data <cascade_dir_name>
-vec <vec_file_name>
-bg <background_file_name>
[-numPos <number_of_positive_samples = 2000>]
[-numNeg <number_of_negative_samples = 1000>]
[-numStages <number_of_stages = 20>]
[-precalcValBufSize <precalculated_vals_buffer_size_in_Mb = 1024>]
[-precalcIdxBufSize <precalculated_idx_buffer_size_in_Mb = 1024>]
[-baseFormatSave]
[-numThreads <max_number_of_threads = 5>]
[-acceptanceRatioBreakValue <value> = -1]
--cascadeParams--
[-stageType <BOOST(default)>]
[-featureType <{HAAR(default), LBP, HOG}>]
[-w <sampleWidth = 24>]
[-h <sampleHeight = 24>]
--boostParams--
[-bt <{DAB, RAB, LB, GAB(default)}>]
[-minHitRate <min_hit_rate> = 0.995]
[-maxFalseAlarmRate <max_false_alarm_rate = 0.5>]
[-weightTrimRate <weight_trim_rate = 0.95>]
[-maxDepth <max_depth_of_weak_tree = 1>]
[-maxWeakCount <max_weak_tree_count = 100>]
--haarFeatureParams--
[-mode <BASIC(default) | CORE | ALL>]
--lbpFeatureParams--
--HOGFeatureParams--
```

Figure 4-3 *Opencv_traincascade parameters*

4.1.4 Resultant trained classifier

After the training finishes the resultant classifier is stored as an xml file. The xml file represents the stages, the nodes (features) to be applied on the testing images per stage, and the threshold per each stage. Figure 4-4 shows snapshots of the saved classifier.



```
cascade.xml - Notepad
File Edit Format View Help

<?xml version="1.0"?>
<opencv_storage>
<cascade>
  <stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>24</height>
  <width>24</width>
  <stageParams>
    <boostType>GAB</boostType>
    <minHitRate>9.9500000476837158e-001</minHitRate>
    <maxFalseAlarm>5.0000000000000000e-001</maxFalseAlarm>
    <weightTrimRate>9.4999999999999996e-001</weightTrimRate>
    <maxDepth>1</maxDepth>
    <maxWeakCount>100</maxWeakCount></stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount>
    <featSize>1</featSize>
    <mode>BASIC</mode></featureParams>
  <stageNum>17</stageNum>
  <stages>
    <!-- stage 0 -->
    <_>
      <maxWeakCount>3</maxWeakCount>
      <stageThreshold>-1.062559091567993e+000</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 1 1.8429443240165710e-002</internalNodes>
          <leafValues>
            -7.9856115579605103e-001 5.7118356227874756e-001</leafValues></_>
        <_>

```

```
cascade.xml - Notepad
File Edit Format View Help

<features>
  <_>
    <rects>
      <_>
        0 1 18 9 -1.</_>
      <_>
        0 4 18 3 3.</_></rects>
      <tilted>0</tilted></_>
    <_>
      <rects>
        <_>
          0 2 8 13 -1.</_>
        <_>
          4 2 4 13 2.</_></rects>
      <tilted>0</tilted></_>
    <_>
      <rects>
        <_>
          0 2 24 5 -1.</_>
        <_>

```

Figure 4-4 Resultant cascade classifier

As shown from Figure 4-4 the classifier is saved as an xml file containing information about the parameters on which this classifier has been trained on, like the feature type and the width and height of the training images, etc. It also contains the stages, their internal nodes and the feature rectangles at the end of the xml file.

A stage is represented as follows (Figure 4-5)

```
<!-- stage 0 -->
<_>
  <maxWeakCount>1</maxWeakCount>
  <stageThreshold>9.5771014690399170e-001</stageThreshold>
  <weakClassifiers>
    <_>
      <internalNodes>
        0 -1 13 -1.2366019655019045e-003</internalNodes>
      <leafValues>
        -9.3442624807357788e-001 9.5771014690399170e-001</leafValues></_></weakClassifiers></_>
```

Figure 4-5 Example of a stored stage of resultant classifier

The values of the internal node are, node.left node.right feature index (features are written as rectangles at the end of the xml file as shown from Figure 4-4) and the node threshold.

4.2 Testing

In order to test the performance of the trained classifier, three processes take place: Collecting test images, using OpenCV's detection function, then matching the detected rectangles to the ground truth of the test images.

4.2.1 Datasets

Face detection datasets were chosen since Haar features were used originally to detect upright frontal faces. The Face Detection Dataset and Caltech Web Faces are the datasets used For testing the built classifiers.

4.2.1.1 The Face Detection Dataset

The Face Detection Dataset (FDDB)(Jain & Learned-Miller, 2010) is a benchmark dataset designed for studying the unconstrained face detection problem. This dataset has been used in many studies and is considered one of the difficult datasets, due to occlusions, out of focus faces and difficult poses (Jain & Learned-Miller, 2010).This dataset contains annotations for 5171 faces in a set of 2845 images. The faces are annotated in the form of ellipses, and their major and minor axes as shown in Figure 4-6.



Figure 4-6 Example of annotated Fddb dataset(Jain & Learned-Miller, 2010).

4.2.1.2 Caltech

Caltech Web Faces is a dataset of human faces collected from the web (Angelova, Abu-Mostafa, & Perona, 2005). It is a challenging dataset since it contains difficult examples such as; extreme face orientations, occlusion like hats and glasses and variable light conditions (Angelova et al., 2005). The Caltech Web Faces data set consists of 10,524 annotated faces. This dataset provides the images along with text files containing the co-ordinates of the mouth, eyes and nose for each face in the image. In order to overcome the extreme face orientation problem, the images were processed and faces where the difference between the y-coordinates of the left and the right eyes are more than 20 percent of the face width, are disregarded. This removed approximately 1,000 images of the Caltech dataset.

4.2.2 Detection with OpenCV.

Using the built classifier to detect faces can be achieved by using openCv's *detectMultiScale* function. It works by loading the xml file of the built stage classifier, then it provides the detected rectangles of the faces on the test images.

In order to calculate a score for each detected rectangle, the information about the stage at which the outputted rectangle has been rejected is required, thus a variation of the

detectMultiScale function of OpenCV. The header of the former function is as follows:

detectMultiScale(Mat image, MatOfRect objects, MatOfInt rejectLevels, MatOfDouble levelWeights, double scaleFactor, int minNeighbors, int flags, Size minSize, Size maxSize, boolean outputRejectLevels)

However, this overloaded method of the *detectMultiScale* function has a bug in opencv 3.1 and has been reported to work in earlier opencv versions especially OpenCV 2.4. Hence, to use this function, OpenCV 2.4.9 was used for detection.

4.2.3 Evaluation Tools

After the trained classifier is used to detect faces, the acquired results have to be validated against the image annotation in order to acquire the accuracy of detection. The evaluation has been done with different tools for each of the 2 datasets chosen for testing.

4.2.3.1 FDDB Evaluation Tool

The FDDB developers provide their tool for evaluation(Jain & Learned-Miller, 2010). The software can be downloaded from their website along with some gnuplot scripts to draw the Receiver Operator Curves (ROC) of previous published papers using their dataset as a benchmark. Their tool is fairly simple to use, the detection file has to be fed to the tool in the following format:

```
<image name i>
<number of faces in this image =im>
<face i1>
<face i2>
...
<face im>
```

Where each face is represented as follows:

Rectangular regions:

```
<left_x top_y width height detection_score>
```

OR,

Elliptical regions

```
<major_axis_radius minor_axis_radius angle center_x center_y
detection_score>.
```

Also the image ordering in the detection file is expected to be the same as the annotated file.

In this work, the Rectangular representation was used in the detected file. The detection score is computed using the following Equation 4.1 as mentioned by the FDDB authors:

$$K * rejectionStage + StageSumofRejectionStage \quad (4.1)$$

Where K is a large number in order to ensure that the window rejected by stage i will have a much higher score than the one rejected by stage $i-1$. Both the `rejectionStage` and the `stageSumofRejectedStage` can be computed by using the *rejectLevels* and the *levelWeights* obtained from the variation of the *detectMultiScale* function of OpenCV.

4.2.3.2 Caltech Webfaces Evaluation Tool

Caltech Webfaces dataset has no evaluation tool thus we built a tool to match the detected faces with the ground truth of the annotated file. This has been achieved through the following steps:

1. Building a tool to convert the given annotation which are the co-ordinates of the mouth, eyes and nose for each face into rectangular regions by assuming that the face width equals double the distance between the 2 eyes. We also assume that the width of the face equals the height of the face.
2. Building another tool that reads both the ground truth rectangles provided by the previous tool and the detected faces from our cascade classifier and matching them to acquire the classifier's accuracy.

For matching, we use the pascal criteria shown in Equation 4.2 (Everingham, Gool, Williams, & Winn, 2010)

$$\frac{area(B_{gt} \cap B_{det})}{area(B_{gt} \cup B_{det})} > 0.4 \quad (4.2)$$

Where B_{gt} is the ground truth bounding box and B_{det} is the detected bounding box. Thus the ratio of the area of intersection between the two boxes to the area of their union has to exceed 0.4 in order for the detected box to be counted as a face.

We followed the same template for the detection file and the annotated file used in the FDDB evaluation tool. Also the detection score has been computed the same way it has been computed for the FDDB.

4.3 Individual and Population fitness

4.3.1 Objective

In this experiment we observe the progression of the best individual fitness, and the average fitness of the population per generation.

4.3.2 Method

In this Experiment a 17 stage cascade classifier has been trained with 500 positive images, and 500 negative images, and a hit rate of 0.995, and a false alarm rate of 0.5 per stage. The GAdaBoost discussed in the proposed method (section 3) has been used to train this cascade classifier. A population size of a 1000 and 50 iterations are the parameters set for the GAdaBoost. Each dummy stage has been trained for only one weak classifier, with no carrying on of the image weights between dummy stages. In addition the check on the special proximity and its removal wasn't utilized.

4.3.3 Results

Figure 4-7 shows the progress of the best individual, and the average fitness of the population. They are shown over the course of 50 iterations of the GA performed before the 17th stage.

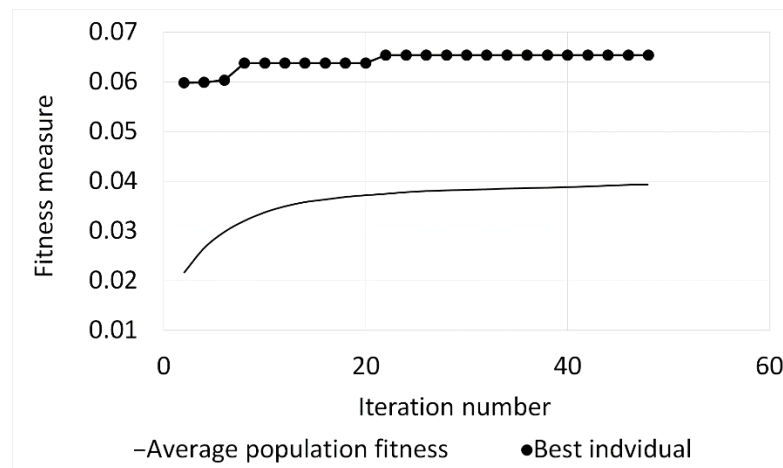


Figure 4-7 Best individual fitness and average population fitness over 50 iterations

4.3.4 Discussion

The best individual fitness either increases or stays constant, this can be attributed to elitism, since in our algorithm the best few features survive and are passed on to the next generation. i.e. not all the individuals of the parent population are replaced by the produced children.

As for the average population fitness, it follows the expected behavior observed in genetic algorithm. The fitness begins to increase significantly then starts to saturate as a maxima is reached.

4.4 Population Size versus Training Time

4.4.1 Objective

In this experiment we emphasize the effect of varying the population size of the GA on the time taken by GAdaboost in training the cascade classifier.

4.4.2 Method

In this experiment a 17 stage cascade classifier is trained 3 times, each with a different population sizes while keeping the other training parameters constant. The constant parameters are: 500 positive images and 500 negative images, a hit rate of 0.995 and a false alarm rate of 0.5 per stage. Each dummy stage is trained for 3 weak classifiers and the sample image weights are carried on between the dummy stages, 20 iterations are set for GAdaboost. Each classifier has been trained multiple times and the average time taken by each have been calculated.

4.4.3 Results

As can be observed from Figure 4-8 as the population size increase the time required to train the cascade classifier increases.

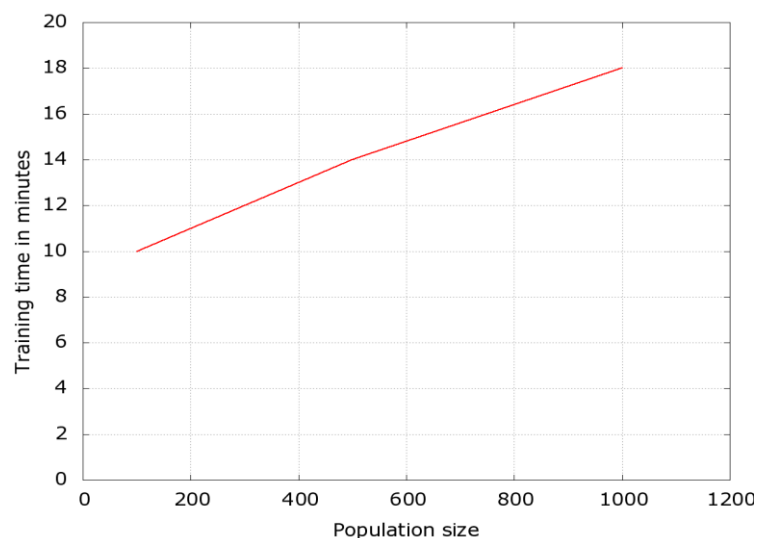


Figure 4-8 Population size vs training time.

4.4.4 Discussion

The expected increase of the training time when increasing the population size can be attributed to two factors: The first one is that the less the population size, the less mutations and crossover processes take place. Consequently, less checking on feature validity will be made, which will eventually save time. The second factor is that the GA provides the Adaboost with a smaller feature set to go through in a brute force manner, which is less time consuming.

4.5 Baseline

4.5.1 Objective

The objective of this experiment is to have a baseline to compare our method with. Our aim is to examine how a cascade classifier without any modification performs on the chosen datasets of faces. The observation of the performances focuses on the accuracy of the detected results, and the time taken to train this classifier.

4.5.2 Method

Opencv_traincascade application has been used to train a cascades classifier, without any modification in the code. The classifier is trained to have 17 stages. The parameters for training are: 500 positive images and 500 negative images per stage. The selected features are of type Haar basic features. A hit rate of 0.995 and a false alarm rate of 0.5 have been chosen per stage. The rest of the parameters are left with the default values.

The obtained classifier has been used to detect faces in 290 images found in the first fold of the FDDB images, and 500 images from the Caltech Webfaces dataset. Then the detected faces are passed to the evaluation applications, to assess their correctness of detecting a face, and observe the accuracy of the trained classifier.

4.5.3 Results

The classifier has finished training in **67 minutes**. The detection power has been evaluated and the results are used to draw a Receiver operator curve (ROC) to provide a visualization of how well the baseline classifier performs on the two chosen datasets. Figure 4-9 shows examples of detection on images. Both Figure 4-10 and Figure 4-11 show the performance of the baseline classifier in detecting faces in both FDDB and Caltech Webfaces respectively.



Figure 4-9 Examples of detection of baseline on images

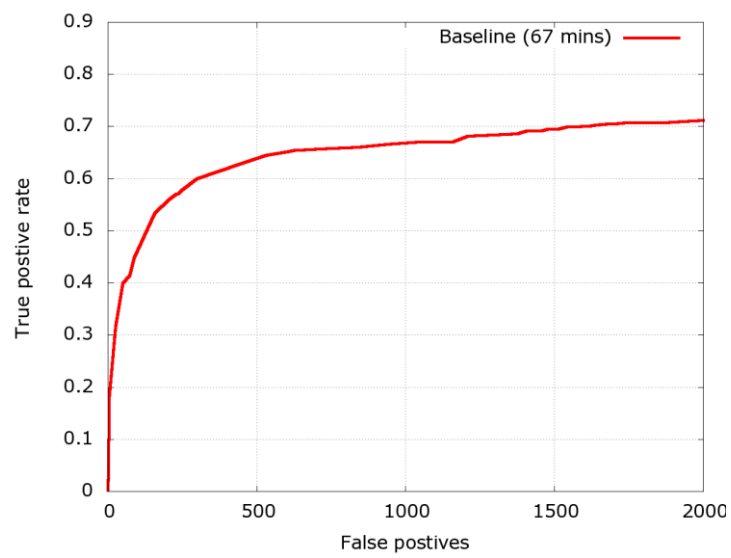


Figure 4-10: Baseline performance on Fddb dataset

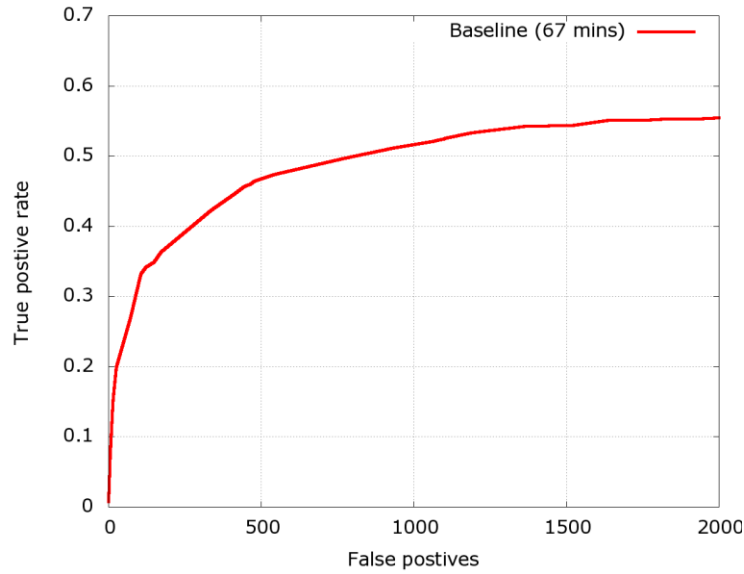


Figure 4-11: Baseline performance on the Caltech dataset

4.5.4 Discussion

As shown in from both figures, at 500 false positives, the baseline true positive rate is 64% and at 1000 false positives the baseline achieved 67% true positive rate on the Fddb dataset.

While on the Caltech Webfaces dataset the baseline at 500 false positives, the baseline true positive rate is 46 % and at 1000 false positives the baseline achieved 51%. These low percentages can be attributed to the fact that the basic types of Haar features perform well in detecting upright frontal faces, while both datasets are hard benchmarks since they contain images with severe face rotations, occlusion and light variations. One possible enhancement would have been to increase the number of positive and negative images used for training, or set a larger value for the number of stages (greater than 17) while training the cascaded classifier.

4.6 GAdaboost 20 iterations

4.6.1 Objective

The objective of this experiment is to test the results of the GAdaboost. In summary, GAdboost is implemented by injecting a feature selection mechanism using GA into the original training mechanism of the cascade classifier. By conducting this experiment we observe the effects of adding GA with 20 iterations to the original Viola-Jones Rapid Object

Detector on the training time and accuracy of detection on both FDDB and the Caltech Webfaces dataset.

4.6.2 Method

A 17 stage classifier using the GAdaboost method has been trained with the following parameters: 500 positive images and 500 negative images per stage, the selected features are of type Haar basic features, a hit rate of 0.995 and a false alarm rate of 0.5 were chosen per stage. The rest of the parameters are left with the default values. Each stage is trained for 3 weak classifiers, and the maximum number of iterations for the GA is 20 iterations and a population size of a 1,000.

The training process has been repeated several times, each time a classifier is obtained. After the training process of the classifier is finished, we acquire each saved final classifier and tested its performance on the FDDB and Caltech Webfaces datasets. The results are presented in the following section.

4.6.3 Results

4.6.3.1 Training time

Since the training process was repeated many times, the training times have been recorded and averaged. Table 4-1 shows the timings of each run and their average.

Table 4-1 Training time for each run of training GAdaboost 20

Run Number	Time in Minutes
1	17
2	17
3	18
4	21
5	16
6	17
7	18
8	16
9	18
10	22
Average	18

4.6.3.2 Results on the Two Datasets

Each classifier obtained from the multiple training runs is tested. Figure 4-12 shows examples of detections on images, While Figure 4-13 and Figure 4-14, show the ROC curves of the results on both the Fddb and the Caltech Webfaces respectively.



Figure 4-12: Examples of detection of Gadaboost20 on images

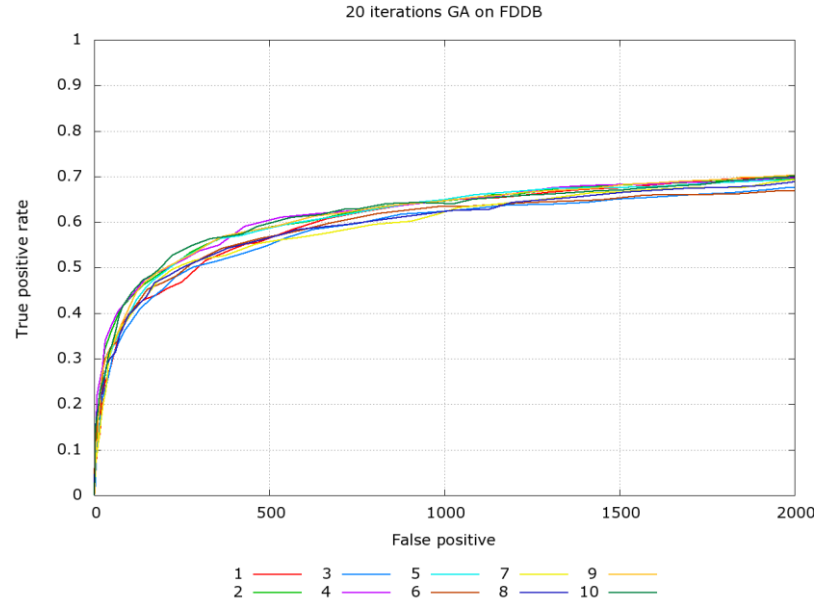


Figure 4-13: ROC curve of multiple GAdaboost20 Classifiers on Fddb

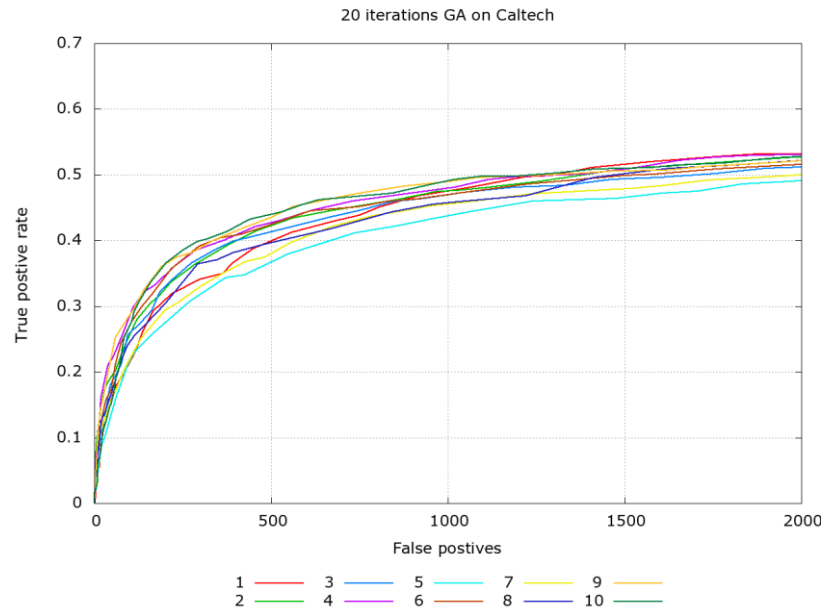


Figure 4-14: ROC curve of multiple GAdaboost20 Classifiers on Caltech

4.6.4 Discussion

As shown from Table 4-1 Training time for each run of training GAdaboost 20 the training time of each run varies slightly from the other with their average being 18 minutes, and the lowest value is 16 minutes and the highest is 22 minutes. The slight variation in the training time can be attributed to the randomness factor that is a part of the GA's nature. In order words, the randomly chosen initial population differ multiple times in each run which might provide better features on some runs over the others, worse features in general will also

require more time in the Adaboost training of each stage in order to reach the specified hit and false alarm rate set for each stage. In addition, non-representative features may take more time to converge thus having more mutations and crossovers done on them than fitter (more representable features), consequently consuming more time. The variation of time may also be attributed to an implementation detail done in GAdboost, which is the marking of used features, this technique has been deployed in order to explore more of the search space, yet its drawback is that it might require more time to search for an unseen feature while randomly choosing features from the original search space.

Concerning the detection accuracy, from both Figure 4-13 and Figure 4-14 it can be shown that the behavior of the runs are similar in both FDDB and Caltech Webfaces dataset. Though the performance of the runs vary, most of them perform well with the exception of some of the runs which perform slightly worse. It can also be noted that at lower thresholds the performances of the runs become more similar (the results become closer to each other). The difference in performances of the runs can also be attributed to the randomness factor of the GA. Where each run may discover a different area of the search space. Not only this, but also the randomness happens before each stage in each run, and while injecting some random features after removing the especially similar features. Another factor may be that the population size was 1000 chromosomes (features) which is a small portion of the whole search space that consists of more than 160,000 features in this case.

4.7 GAdaboost 50 iteration

4.7.1 Objective

The objective of this experiment is to test the results of yet another variation of the GAdaboost. By conducting this experiment we observe the effects of adding GA with 50 iterations to the original Viola-Jones Rapid Object Detector on the training time and accuracy of detection on both FDDB and the Caltech Webfaces datasets.

4.7.2 Method

A 17 stage classifier using the GAdaboost method has been trained with the following parameters: The parameters for training are: 500 positive images and 500 negative images per stage. The selected features are of type Haar basic features. A hit rate of 0.995 and a false alarm rate of 0.5 have been chosen per stage. The rest of the parameters are left with the default

values. The parameters of the Genetic algorithm are: Each stage is trained for 3 weak classifiers, and the maximum number of iterations for the GA is 50, and a population size of a 10,000.

The training process has been repeated several times, each time a classifier is obtained. After the training process of the classifier is finished, we acquire each saved final classifier and tested its performance on the Fddb and Caltech Webfaces datasets. The results are presented in the following section.

4.7.3 Results

4.7.3.1 Training time

Since the training process was repeated many times, the training times have been recorded and averaged. Table 4-2 shows the timings of each run and their average.

Table 4-2 Training time for each run of training GAdaboost 50

Run Number	Time in Minutes
1	30
2	32
3	27
4	33
5	38
6	27
7	29
8	26
9	28
10	28
Average	29.8

4.7.3.2 Results on the Two Datasets

Each classifier obtained from the multiple training runs is tested. Figure 4-15 shows examples of detections on images, while Figure 4-16 and Figure 4-17, show the ROC curves of the results on both the Fddb and the Caltech Webfaces respectively.



Figure 4-15: Example of detections of GAdaboost50

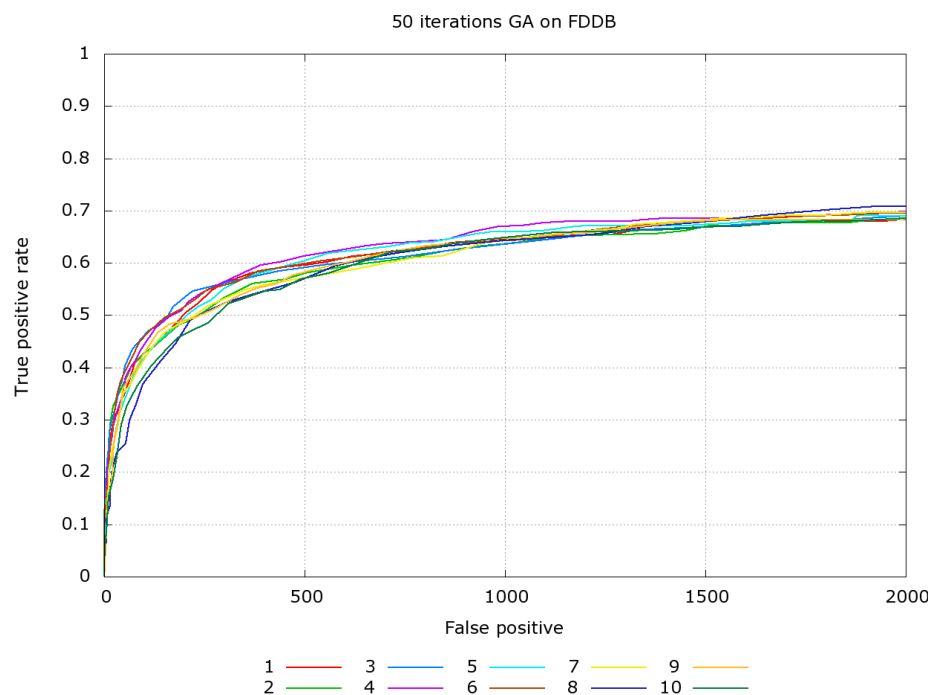


Figure 4-16: ROC curve of multiple GAdaboost50 Classifiers on Fddb

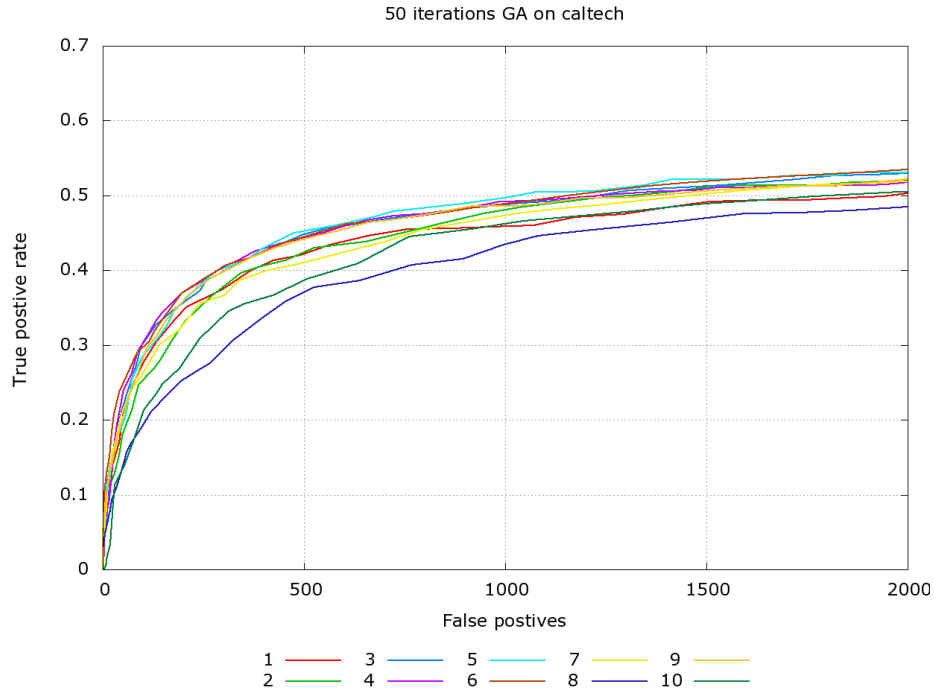


Figure 4-17 ROC curve of multiple GAdaboost50 Classifiers on Caltech

4.7.4 Discussion

As shown from Table 4-1 Training time for each run of training GAdaboost 20 the training time of each run varies slightly from the other with their average being 30 minutes, and the lowest value is 26 minutes and the highest is 38. The GAdaboost with 50 iterations exhibits the same behavior of the GAdaboost with 20 iterations (shown in the previous experiment: Section 4.6). The randomness in the GA nature is deemed responsible for the slight variation in time difference between the runs. The randomness affects the initial quality of features, meaning that the first population may have been better in some runs than the others. This also means that the explored part of the search space differs between the runs. The quality of explored features can be held accountable for the difference in the time taken by the Adaboost for training. As the case with GAdaboost 20 iterations, the variation of time can also be due to the marking of used features. This entails that more time will be consumed while searching for an unseen feature, to be selected from the original search space.

Concerning the detection accuracy, GAdaboost 50 showed similar behavior as GAdaboost 20. From both Figure 4-16 and Figure 4-17 it can be shown that the behavior of the runs are similar in both FDDB and Caltech Webfaces dataset. The performance of the runs vary, yet only a smaller portion of the runs perform worse than the rest. In other words, most of the runs perform well. It can also be noted that the results of the runs at lower thresholds are

closer to each other. The difference in performances of the runs can also be attributed to the randomness factor of the GA which denotes that each run may discover a different area of the search space according to the random seed. In GAdaboost, the randomness also happens before each stage in each run, and also while injecting some random features after removing the spatially similar features, which was meant to avoid redundancy. The population size of a 1000, which is a small sub-set of the 160,000 feature search space, might have also contributed to the reason that some of the runs perform poorly along with the random seed dependency.

4.8 Training speed versus Accuracy

4.8.1 Objective

In this sub-section, the objective is to have an overview of the performance of the variations of the proposed method versus the baseline, which is the original Viola-Jones cascade classifier. In order to achieve the former goal, the results obtained from 3 previous experiments are compared. The three experiments are: The baseline, GAdaboost with 20 iterations and GAdaboost with 50 iterations. The performance of the 3 classifiers are obtained and ROC on both FDDB and Caltech Webfaces datasets are drawn and compared. In addition the time variation in training the 3 classifiers is compared and visually emphasized on a graph.

4.8.2 Results

4.8.2.1 Time Comparison Graph

Figure 4-18 plots the time taken to train each of the 3 experiments: The baseline, GAdaboost with 50 iterations and GAdaboost with 20 iterations.

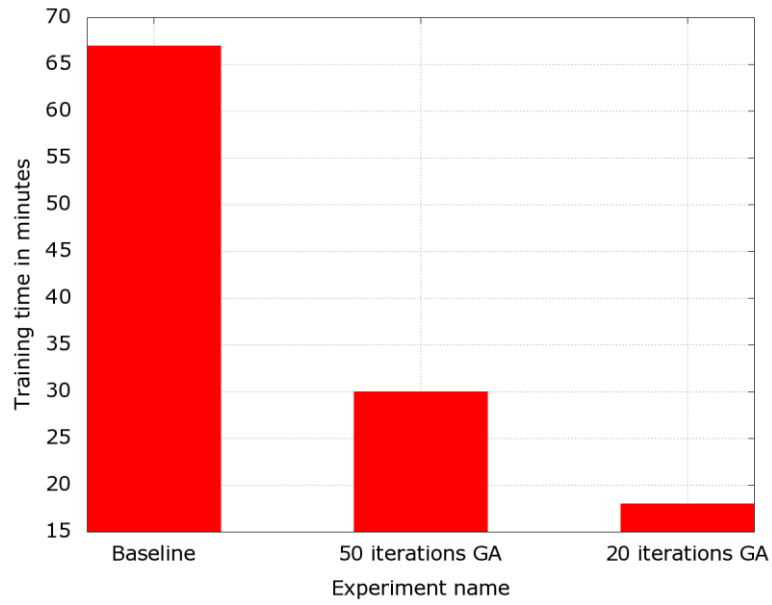


Figure 4-18 Training time in minutes of each of the experiments.

4.8.2.2 Accuracy on Fddb and Caltech Webfaces Datasets

Figure 4-19 and Figure 4-20 provide the baseline results versus the Y error bars, showing the maximum, minimum and average results, for all the runs of both the 20 and the 50 iterations GAdaBoost on Fddb dataset

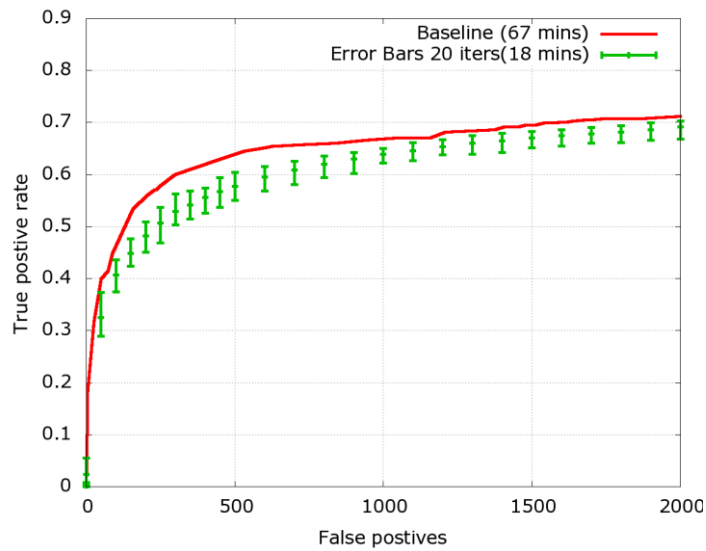


Figure 4-19 Y error bars for all the runs of the 20 iterations GAdaBoost on Fddb.

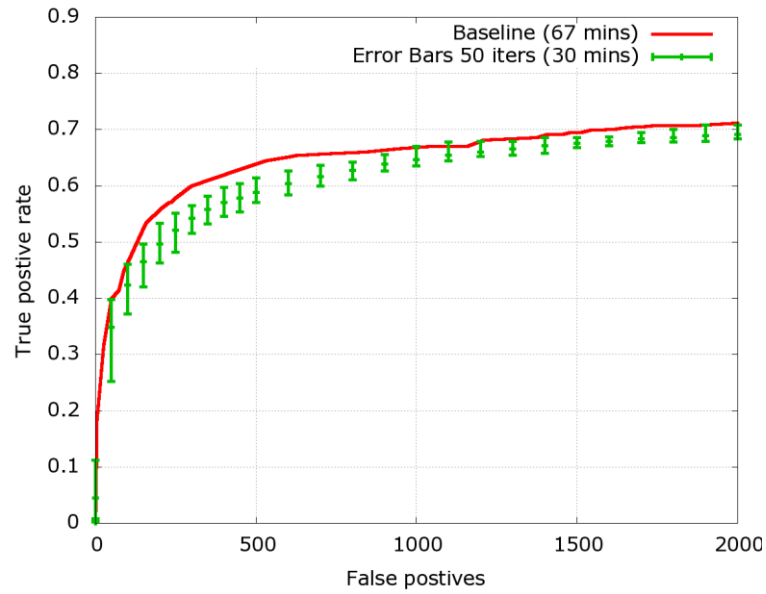


Figure 4-20: Y error bars for all the runs of the 50 iterations GAdaBoost on Fddb

Figure 4-21 and Figure 4-22 provide the baseline results versus the Y error bar graphs, showing the maximum, minimum and average results, for all the runs of both the 20 and the 50 iterations GAdaBoost on Caltech Webfaces dataset.

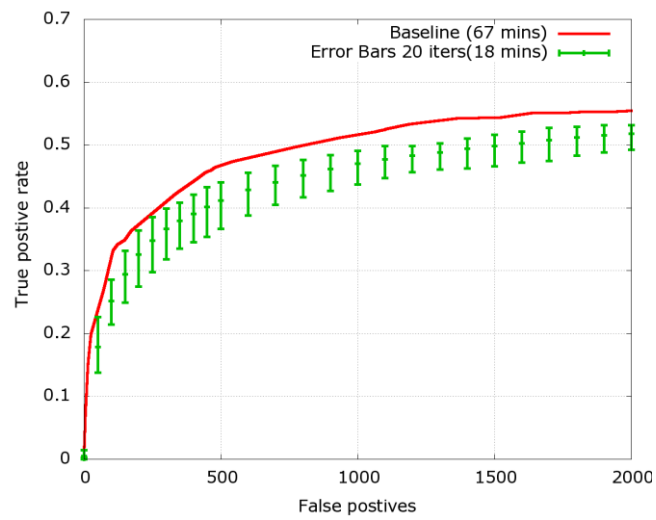


Figure 4-21: Y error bars for all the runs of the 20 iterations GAdaBoost on Caltech Web Faces.

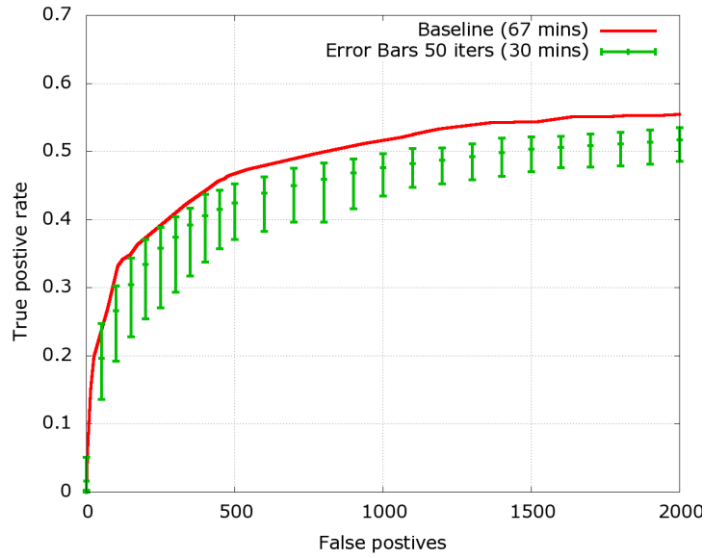


Figure 4-22: Y error bars for all the runs of the 50 iterations GAdaBoost on Caltech Web Faces.

4.8.3 Discussion

From both Figure 4-19 and Figure 4-20, by examining the average point on the Y error bars it can be observed that at 500 false positives the baseline true positive rate is 64% and the GAdaBoost 20 and 50 iterations achieved 58% 59% true positive rate respectively. While at 1000 false positives the baseline achieved 67% true positive rate versus about 64% and 65% for the GA 20 and 50 iterations respectively.

Figure 4-21 and Figure 4-22, by examining the average point on the Y error bars we find that at 500 false positives the baseline true positive rate is 46 % and the GAdaboost 20 and 50 iterations achieved 41% , 43% true positive rate respectively. While at 1000 false positives the baseline achieved 51% true positive rate versus about 47% and 48% for the GAdaBoost 20 and 50 iterations respectively.

Collectively from the provided figures, it can be noted that GAdaBoost with 50 iterations has performed slightly better than the GAdaBoost with 20 iterations. It can also be observed that at lower thresholds the GA provides closer true positive rates compared with the baseline, than it does at higher thresholds. By drawing the Y error bars with the averages it can be observed that most of the runs achieved high detection rates with the exception of a couple of outliers, which showed worse performance than the majority of the runs. It can also be noted that some of the GAdaboost runs had almost reached the same accuracy of the baseline classifier.

The decrease in performance of both the baseline and GAdaboost can be attributed to the fact that both FDDB and Caltech Web Faces dataset include occlusions and light variations, as was mentioned at the beginning of this section. Table 4-3 provides a summary of the performance of the baseline, GAdaboost 50, and GAdaboost 20.

Table 4-3 Summary of performance of the baseline, GAdaboost50, and GAdaboost20

Experiment name	Training Time	TPR on FDDB		TPR on Caltech Webfaces	
		500 FP	1000 FP	500 FP	1000 FP
Baseline	67 minutes	64%	67%	46 %	51%
GAdaboost 50	30 minutes	59%	65%	43 %	48%
GAdaboost 20	20 minutes	58%	64%	41 %	47%

CHAPTER (5): CONCLUSIONS

With the constant automation of processes, much focus has been given to machine learning techniques. Machine learning is the process of learning from collected data. As the data increases, the need for techniques to reduce the dimensionality of data to reach efficient classifiers becomes unavoidable. One of the areas that suffer from the curse of dimensionality is the area of computer vision, specifically object detection. In this study, first the enhancements done on the Viola-Jones Object detector are reviewed. Then comprehensive overviews on Feature Selection methods have been assessed. Due to the multiple evidence which suggest the powerfulness of the Genetic Algorithms and their wide use in Feature Selection techniques, this work incorporated the use of GA into the Viola-Jones Rapid Object Detector aiming to enhance the training time of this detector without a significant loss of accuracy. Incorporating the use of GAs will speed up the training process by developing a set of representative features to present to the Adaboost learning algorithm instead of going through the set of all possible features multiple times due to its brute force nature. The motivation behind this technique is that the feature space of such detectors is huge. For example, for a 24X24 image the feature space can include more than 160,000 features. In order to build the proposed method, the implementation of the Viola-Jones detector in the OpenCV library has been modified. Functions that apply GA before the training of each stage were added to provide the stage training (that uses the Adaboost machine learning technique) with a meaningful set of features, disregarding the insignificant features, by doing so, we were able to train classifiers using our proposed technique (GAdaboost). The training time taken has been recorded and compared against that of a trained baseline classifier with the same parameters but without the use of GAs. The accuracy of detection of the classifiers trained with the GAdaboost technique were compared to that of the baseline classifier by testing them on both the Face Detection Dataset and the Caltech 10,000 Webfaces dataset, and the results have proven to be somewhat promising.

5.1 Contributions

We showed the effect of incorporating Genetic Algorithms with the Viola-Jones Rapid Object Detector on enhancing the training speed. Experiments to show the progression of the best individual and the average population fitness were provided. Other experiments showed the speedup of that training process, which can be gained by the reduction of the population size.

Also, two variations of the GAdaboost were examined, one with 20 iterations and the other with 50 iterations. Both experiments were run multiple times to observe the effect of the number of iterations on the performance using the FDDB and Caltech Web Faces dataset. We experienced that the training process became up to 3.7 times faster than the original algorithm with a mere decrease of 3% to 4% in accuracy. We noted that the 50 iterations performed better than the 20 iterations, and both had best case scenarios of almost reaching the baseline accuracy at some thresholds.

5.2 Future Work

Although GAdaboost provided evidence of the successfulness of introducing GAs to the Viola-Jones Rapid Object Detector, there is still more room for enhancements to achieve better results with this technique. The future extension of this contribution can be done by experimenting with more GAdaboost parameters by varying the iteration numbers, the population size, or finding a better stopping criteria for the GA. The parallelizable nature of the GA can be utilized to gain an even faster training process. Another enhancement over the GAdaboost is to introduce guided randomness to the initial population of the Genetic Algorithms. Also the use of filter techniques in picking an originally representative population instead of a completely random set of features which might originally include useless features.

REFERENCES

- Amit, Y., Geman, D., & Jedynak, B. (1998). Efficient Focusing and Face Detection. In H. Wechsler, P. J. Phillips, V. Bruce, F. F. Soulié, & T. S. Huang (Eds.), *Face Recognition: From Theory to Applications* (pp. 157–173). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-72201-1_8
- Andrade, A. V., & Errico, L. De. (2008). Analysis of Selection and Crossover Methods used by Genetic Algorithm-based Heuristic to solve the LSP Allocation Problem in MPLS Networks under Capacity Constraints, (June), 1–5.
- Angelova, A., Abu-Mostafa, Y., & Perona, P. (2005). Pruning training sets for learning of object categories. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1*, 494–501. <http://doi.org/10.1109/CVPR.2005.283>
- Ayala-ramirez, V., Garcia-capulin, C. H., Perez-garcia, A., & Sanchez-yanez, R. E. (2006). Circle detection on images using genetic algorithms, 27, 652–657. <http://doi.org/10.1016/j.patrec.2005.10.003>
- Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. In *Proceedings of an International Conference on Genetic Algorithms and their applications* (pp. 101–111).
- Chaaroui, A. A., & Flórez-Revuelta, F. (2013). Human action recognition optimization based on evolutionary feature subset selection. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation* (pp. 1229–1236).
- Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. *Computers and Electrical Engineering*, 40(1), 16–28. <http://doi.org/10.1016/j.compeleceng.2013.11.024>
- Chouaib, H., Terrades, O. R., Tabbone, S., Cloppet, F., Vincent, N., & Nancy, B. P. (2008). Feature selection combining genetic algorithm and Adaboost classifiers. *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, 5(Ea 2517), 4. <http://doi.org/10.1109/ICPR.2008.4761264>
- Dezhen, Z., & Kai, Y. (2008). Genetic Algorithm Based Optimization for AdaBoost. In *Computer Science and Software Engineering, 2008 International Conference on* (Vol. 1, pp. 1044–1047). <http://doi.org/10.1109/CSSE.2008.1040>
- Dimashova, M. (2012). How is Decision Tree Split Quality Computed. Retrieved from <http://answers.opencv.org/question/566/how-is-decision-tree-split-quality-computed/>
- Everingham, M., Gool, L. Van, Williams, C. K. I., & Winn, J. (2010). The PASCAL Visual Object Classes (VOC) Challenge. *International Journal*, 303–338. <http://doi.org/10.1007/s11263-009-0275-4>
- Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning Generative Visual Models From Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In *IEEE CVPR Workshop of Generative Model Based Vision (WGMBV)*.

- Ferri, F., & Pudil, P. (1994). Comparative study of techniques for large-scale feature selection. *Machine Intelligence and Pattern ...*. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.4369&rep=rep1&type=pdf>
- Freund, Y., & Schapire, R. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. *Computational Learning Theory*, 55, 119–139. <http://doi.org/10.1006/jcss.1997.1504>
- Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms* (pp. 69–93). Morgan Kaufmann.
- Harb, H. M. H., & Desuky, A. A. S. (2011). Adaboost Ensemble with Genetic Algorithm Post Optimization for Intrusion Detection. *Update*, 2(5), 1. <http://doi.org/10.1.1.402.9250>
- Hasançebi, O., & Erbatur, F. (2000). Evaluation of crossover techniques in genetic algorithm based optimum structural design. *Computers and Structures*, 78(1), 435–448. [http://doi.org/10.1016/S0045-7949\(00\)00089-4](http://doi.org/10.1016/S0045-7949(00)00089-4)
- Hjelmås, E., & Low, B. K. (2001). Face Detection: A Survey. *Computer Vision and Image Understanding*, 83(3), 236–274. <http://doi.org/10.1006/cviu.2001.0921>
- Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press.
- Itseez. (2015). Open Source Computer Vision Library.
- Jacobson, L. (2012). Applying a genetic algorithm to the traveling salesman problem. Retrieved May 8, 2015, from <http://www.theprojectspot.com/tutorial-post/applying-a-genetic-algorithm-to-the-travelling-salesman-problem/5>
- Jain, V., & Learned-Miller, E. (2010). *FDDB: A Benchmark for Face Detection in Unconstrained Settings*.
- Jeong, Y.-S., Shin, K. S., & Jeong, M. K. (2014). An evolutionary algorithm with the partial sequential forward floating search mutation for large-scale feature selection problems. *Journal of the Operational Research Society*, 1–10. <http://doi.org/10.1057/jors.2013.72>
- Lee, J., & Lee, J. (2014). An Efficient Prediction for Heavy Rain from Big Weather Data using Genetic Algorithm.
- Li, Q., Niaz, U., & Merialdo, B. (2012). An improved algorithm on Viola-Jones object detector. *Proceedings - International Workshop on Content-Based Multimedia Indexing*, 55–60. <http://doi.org/10.1109/CBMI.2012.6269796>
- Li, R., Lu, J., Zhang, Y., & Zhao, T. (2010). Dynamic Adaboost learning with feature selection based on parallel genetic algorithm for image annotation. *Knowledge-Based Systems*, 23(3), 195–201. <http://doi.org/10.1016/j.knosys.2009.11.020>

- Liang, D., Tsai, C. F., & Wu, H. T. (2014). The effect of feature selection on financial distress prediction. *Knowledge-Based Systems*, 73(1), 289–297. <http://doi.org/10.1016/j.knosys.2014.10.010>
- Lienhart, R., & Maydt, J. (2002). An extended set of Haar-like features for rapid object detection. *Proceedings. International Conference on Image Processing*, 1, 900–903. <http://doi.org/10.1109/ICIP.2002.1038171>
- Lillywhite, K., Lee, D. J., Tippetts, B., & Archibald, J. (2013). A feature construction method for general object recognition. *Pattern Recognition*, 46(12), 3300–3314. <http://doi.org/10.1016/j.patcog.2013.06.002>
- Magalhães-Mendes, J. (2013). A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem. *WSEAS Transactions on Computers*, 12(4), 164–173.
- Manikas, T. W., & Cain, J. T. (1996). Genetic Algorithms vs . Simulated Annealing : A Comparison of Approaches for Solving the Circuit Partitioning Problem Genetic Algorithms vs . Simulated Annealing : A Comparison of Approaches for Solving the Circuit Partitioning Problem by.
- Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press.
- Noraini, M., & Geraghty, J. (2011). Genetic algorithm performance with different selection strategies in solving TSP. *World Congress on Engineering*, II(978-988-19251-4-5), 4–9. Retrieved from <http://umpir.ump.edu.my/2609/>
- Oreski, S., & Oreski, G. (2014). Genetic algorithm-based heuristic for feature selection in credit risk assessment. *Expert Systems with Applications*, 41(4 PART 2), 2052–2064. <http://doi.org/10.1016/j.eswa.2013.09.004>
- Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the curses of dimensionality* (Vol. 703). John Wiley & Sons.
- Santana, L. E. A., Silva, L., Canuto, A. M. P., Pinto, F., & Vale, K. O. (2010). A Comparative Analysis of Genetic Algorithm and Ant Colony Optimization to Select Attributes for an Heterogeneous Ensemble of Classifiers.
- Schölkopf, B., Luo, Z., & Vovk, V. (2013). Empirical inference: Festschrift in honor of Vladimir N. Vapnik. *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, 1–287. <http://doi.org/10.1007/978-3-642-41136-6>
- Srinivas, M., & Patnaik, L. M. (1994). Genetic Algorithms: A Survey. *Computer*, 27(6), 17–26. <http://doi.org/10.1109/2.294849>
- Sun, Z., Bebis, G., & Miller, R. (2004). Object detection using feature subset selection. *Pattern Recognition*, 37(11), 2165–2176. <http://doi.org/10.1016/j.patcog.2004.03.013>

- Tabassum, M., & Mathew, K. (2014). A Genetic Algorithm Analysis towards Optimization solutions. *International Journal of Digital Information and Wireless Communications (IJDWC)*, 4(1), 124–142. Retrieved from <http://sdiwc.net/digital-library/a-genetic-algorithm-analysis-towards-optimization-solutions.html>
- The OpenCV Reference Manual. (2014, April).
- Tsai, C. F., Eberle, W., & Chu, C. Y. (2013). Genetic algorithms in feature and instance selection. *Knowledge-Based Systems*, 39, 240–247. <http://doi.org/10.1016/j.knosys.2012.11.005>
- Vignolo, L. D., Milone, D. H., & Scharcanski, J. (2013). Feature selection for face recognition based on multi-objective evolutionary wrappers. *Expert Systems with Applications*, 40(13), 5077–5084. <http://doi.org/10.1016/j.eswa.2013.03.032>
- Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 1. <http://doi.org/10.1109/CVPR.2001.990517>
- Xia, H., Zhuang, J., & Yu, D. (2014). Multi-objective unsupervised feature selection algorithm utilizing redundancy measure and negative epsilon-dominance for fault diagnosis. *Neurocomputing*, 146, 113–124. <http://doi.org/10.1016/j.neucom.2014.06.075>
- Xue, B., Fu, W., & Zhang, M. (2014). Multi-objective Feature Selection in Classification: A Differential Evolution Approach. In G. Dick, W. N. Browne, P. Whigham, M. Zhang, L. T. Bui, H. Ishibuchi, ... K. Tang (Eds.), *Simulated Evolution and Learning: 10th International Conference, SEAL 2014, Dunedin, New Zealand, December 15-18, 2014. Proceedings* (pp. 516–528). Cham: Springer International Publishing. http://doi.org/10.1007/978-3-319-13563-2_44
- Xue, B., Zhang, M., Browne, W. N., & Yao, X. (2016). A Survey on Evolutionary Computation Approaches to Feature Selection. *IEEE Transactions on Evolutionary Computation*, 20(4), 606–626. <http://doi.org/10.1109/TEVC.2015.2504420>
- Yusta, S. C. (2009). Different metaheuristic strategies to solve the feature selection problem. *Pattern Recognition Letters*, 30(5), 525–534. <http://doi.org/10.1016/j.patrec.2008.11.012>