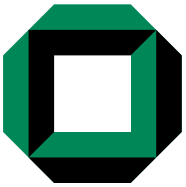

Universität Karlsruhe (TH)
Fakultät für Informatik
Interactive Systems Labs
Prof. Dr. A. Waibel

Diploma thesis

Combination of Classifier Cascades and Training Sample Selection for Robust Face Detection

Submitted by

Lorant Szasz-Toth

January 2009

Advisors

M.Sc. Hazım Kemal Ekenel, Universität Karlsruhe (TH)
Dr. Jie Yang, School of Computer Science, Carnegie Mellon University
Dr. Wende Zhang, General Motors
Dr.-Ing. Rainer Stiefelhagen, Universität Karlsruhe (TH)

Interactive Systems Labs
Institut für Anthropomatik
Universität Karlsruhe (TH)
Title: Combination of Classifier Cascades and Training Sample Selection for Robust Face Detection
Author: Lorant Szasz-Toth

Lorant Szasz-Toth
Rudolfstr. 15
76131 Karlsruhe, Germany
email: LorantToth@gmail.com

Statement of authorship

I hereby declare that this thesis is my own original work which I created without illegitimate help by others, that I have not used any other sources or resources than the ones indicated and that due acknowledgement is given where reference is made to the work of others.

Karlsruhe, 30. January 2009

.....

(Lorant Szasz-Toth)

Abstract

Face detection is one of the most fundamental tasks in human-computer-interaction, surveillance, and, more recently, image retrieval. Determining the location and size of faces in input images is a prerequisite for many other applications, including face recognition. In recent years several breakthroughs have been made in this field. These days, face detectors deliver high detection rates, low false alarm rates and run in real-time.

Despite the efforts and publicly available tools, training high-performance face detectors from scratch remains a challenge. Mostly, because training time for a single cascade can be in the order of days and various training parameters have to be chosen carefully. Usually, training involves acquiring heuristics and a feeling for the intricacies of the training process and the influence of training parameters. A substantial amount of time is spent training classifiers iteratively and modifying parameters, while usually discarding intermediate results.

The goal of this work is to overcome some of the problems of training cascade classifiers and to promote the use of custom-trained classifiers. Specifically, two problems are addressed in this work. First, an approach to combine several trained cascade classifiers into a single cascade is presented and evaluated. Second, a technique to optimize the training set is explored.

A major challenge during cascade training is the choice of training parameters. There is no ideal way to choose these parameters and optimization is not feasible. Usually, the process involves several attempts or guesses at the right parameters and, finally, the best performing classifier is selected. Instead of discarding intermediate results, several of these classifiers are combined into a single new classifier. Unlike previous work, the base classifiers are not run in parallel but a fixed number of individual classifier stages are optimized, selected and combined into a new classifier without added run-time overhead.

Experiments have shown the importance of a proper choice of training samples. Classifiers trained with a reduced amount of well-chosen samples can outperform a classifier that was trained on a far larger training set. The use of less training samples to achieve the same performance decreases the required training time, especially with large training sets, where results cannot be cached. Additionally, forcing the classifier to focus on difficult training examples has shown to increase classification performance. Therefore, a method to select an optimized set of training samples from a large set with the help of support vector

machines is explored.

The results of both presented approaches have been evaluated on the widely used, publicly available CMU+MIT database. Both the SVM-based training sample selection and the cascade combination approaches are shown to improve the performance over the base classifiers. Cascade combination allows to generate a classifier within a single day that performs nearly as well as a single, high-performance classifier trained in more than ten days. Additionally, classifiers generated by cascade combination outperform the orignal base cascades. Face detectors trained with SVM-based training set selection perform better than equally trained base classifiers with a random choice of training samples. Both presented approaches were able to produce cascade classifiers that clearly outperform the publicly available OpenCV face detectors.

Zusammenfassung

Die Gesichtsdetektion ist eine der grundlegenden Aufgaben der Mensch-Maschine-Interaktion, Videoüberwachung und, neuerdings, der Verwaltung von digitalem Videomaterial. Das Auffinden von Gesichtern ist die Grundvoraussetzung für viele weitere Aufgaben, wie zum Beispiel der Gesichtserkennung. Mehrere wissenschaftliche Durchbrüche gelangen in den letzten Jahren auf diesem Feld. Diese ermöglichen heutigen Gesichtserkennern hohe Detektionsraten in Echtzeit bei einer niedrigen Zahl falscher Detektionen.

Trotz einer Vielzahl wissenschaftlicher Veröffentlichungen zu diesem Thema und der öffentlichen Verfügbarkeit von Toolkits zur Gesichtserkennung, bleibt das Training von kaskaden-basierten Detektoren eine Herausforderung. Der Hauptgrund hierfür ist die hohe Trainingszeit von meist mehreren Tagen, gepaart mit der Notwendigkeit mehrere Parameter manuell zu wählen. Normalerweise umfasst der Trainingsprozess daher das Erstellen mehrer Detektoren, um ein Gefühl für den Einfluss der Parameter und die Eigenheiten des Trainingsprozesses zu erlangen. Viel Zeit vergeht daher mit dem wiederholten Training dieser Klassifikatoren bis letztendlich der beste Parametersatz gewählt wird.

Das Ziel dieser Arbeit ist es die genannten Probleme zu entschärfen. Folgende zwei Ansätze werden in dieser Arbeit im Besonderen behandelt. Erstens wird eine Technik zum Zusammenführen verschiedener, bereits trainierter Klassifikatoren in einen einzigen Detektor präsentiert. Der zweite Ansatz dient der Optimierung des Trainingsdatensatzes.

Da die Wahl optimaler Trainingsparameter schwierig und eine Optimierung nahezu unmögich ist, befasst sich der erste Ansatz mir der Kombination verschiedener, nicht optimal trainierter Detektoren in einen einzigen finalen Klassifikator. Dieser kombinierte Detektor verbessert die Leistungsfähigkeit der Ausgangsdetektoren und führt diese nicht einfach parallel aus, sondern ist so schnell wie ein einziger Klassifikator.

Experimente haben die Bedeutung einer guten Wahl der Trainingsbeispiele gezeigt. Ein kleiner, gut-gewählter Trainingssatz kann einen Klassifikator hervorbringen, der die gleiche Leistung erbringt wie ein mit größerem Datensatz trainierter Klassifikator. Dies widerum reduziert den Trainingszeitbedarf. Außerdem hat sich gezeigt, dass es hilfreich ist das Training des Detektors auf schwere Beispiele zu lenken. Darauf beruhend wird in dieser Arbeit ein Ansatz zur optimierten Wahl der Trainingsbeispiele mittels SVMs vorgestellt.

Die Leistung beider vorgestellten Ansätze wird auf dem vielfach genutzten, öffentlich verfügbaren CMU+MIT Datensatz evaluiert. Sowohl die Optimierung des Trainingssatzes,

als auch die Kombination verschiedener Klassifikatoren, zeigen eine gesteigerte Leistung im Vergleich zu herkömmlich trainierten Kaskaden. Die Kombination verschiedener Kaskaden kann dazu genutzt werden innerhalb eines Tages einen Klassifikator zu generieren, der mit einem über den Zeitraum von zehn Tagen trainierten Detektor vergleichbar ist. Zum Schluss dieser Arbeit folgt noch ein Vergleich der hier präsentierten Ansätze mit öffentlich verfügbaren Gesichtserkennern und State-of-the-Art Detektoren. Die resultierenden Klassifikatoren beider vorgestellten Verfahren erreichen bessere Resultate als die öffentlich verfügbaren OpenCV Kaskaden.

Acknowledgements

The completion of this thesis would not have been possible without the help and support of several individuals. The author would like to thank his advisors Dr. Jie Yang, Carnegie Mellon University, Dr. Wende Zhang, General Motors, M. Sc. Hazım K. Ekenel and Dr.-Ing. Rainer Stiefelhagen from Universität Karlsruhe (TH) for the valuable discussions, suggestions and corrections.

The author would also like to thank everyone involved with the InterACT exchange program for offering the opportunity to pursue the research at the Carnegie Mellon University in Pittsburgh, PA.

This research was partially supported by General Motors through GM-CMU collaborative Lab, NIH (under contract 1U01HL09173601) and by OSEO, French State agency for innovation, as part of the Quaero Programme.

Contents

1 Introduction **1**

1.1 Motivation 2

1.2 Contribution 3

1.3 Outline 4

2 Overview of Face Detection Approaches **5**

2.1 Knowledge-based Face Detection 5

2.2 Feature-based Face Detection 6

2.3 Template-based Face Detection 8

2.4 Appearance-based Face Detection 9

2.5 Cascades of Boosted Ensembles 13

2.5.1 Alternative Boosting Algorithms 13

2.5.2 Alternative Feature Sets 14

2.5.3 Weak Learners 16

2.5.4 Cascade Training Procedure 17

2.5.5 Cascade Architecture 18

2.5.6 Training Time Improvement 19

2.5.7 Run-time Efficiency Improvement 20

2.5.8 Classifier Combination 21

3 Methodology **23**

3.1 Face Detection 23

3.1.1 Haar Features 23

3.1.2 Integral Image Representation 24

3.1.3 Boosting 25

3.1.4 Weak Learners 28

3.1.5 Cascades of Boosted Ensembles . . . 28
3.1.6 Optimal Thresholding for Cascaded Ensembles . . . 29
3.1.7 Bootstrapping . . . 30
3.1.8 Matrix-Structural Learning . . . 30
3.2 Entropy and Mutual Information . . . 31
3.3 Support Vector Machines . . . 33
3.3.1 Linear classification . . . 33
3.3.2 Soft-margin linear classification . . . 35
3.3.3 Non-linear classification . . . 36

4 Cascade Combination and SVM-based Sample Selection 37
4.1 Cascade Training . . . 38
4.2 Cascade Combination . . . 39
4.2.1 Optimization Problem Formulation . . . 39
4.2.2 Cascade Combination with Threshold Optimization . . . 41
4.2.3 Cascade Combination with Conditional Mutual Information Maximization . . . 42
4.2.4 Cascade Combination with Mutual Information Maximization . . . 43
4.2.5 Cascade Combination with Correlation Maximization . . . 44
4.2.6 Sample Extraction and Subsampling . . . 45
4.3 Training Set Selection . . . 45
4.3.1 SVM-based Training Set Selection . . . 46

5 Experiments 49
5.1 Experimental setup . . . 49
5.1.1 Dataset . . . 49
5.1.2 Bounding Boxes . . . 49
5.1.3 Face Matching Metric . . . 50

5.2 Receiver Operating Characteristic Curves 51
5.3 Base Classifiers 53
5.3.1 Training Set 53
5.3.2 Base Classifiers 53
5.3.3 Training Time 57
5.3.4 Motivation for Training Set Selection 58
5.3.5 Motivation for Cascade Combination 60
5.3.6 Large Training Sets with Matrix-Structural Learning 61
5.4 Classifier Combination 62
5.4.1 Threshold Optimization 62
5.4.2 Subsampling for Cascade Combination 64
5.4.3 Comparison of Cascade Combination Cost Functions 64
5.4.4 Combination Stage Count 66
5.4.5 Comparison against Base Classifiers 67
5.5 SVM-based Training Set Selection 69
5.5.1 Histogram Equalization vs. Raw Image Intensity Features 69
5.5.2 SVM-based Training Set Resampling 69
5.6 Combining SVM-based Resampling and Cascade Combination 71
5.7 Comparison against other Cascade Classifiers 72

6 Conclusion **75**

7 Future Work **77**

List of Figures

1 Example Haar-like features 24
2 Matrix-Structural Learning overview 32
3 Maximum margin classification 34
4 Non-linear classification using kernel functions 36
5 ROC curve smoothing 52
6 ROC curves 3000 samples 54
7 ROC curves 4500 samples 54
8 ROC curves 3000, 4500, 9000 samples 55
9 ROC curve 20 vs. 25 stages 56
10 Training times during 25 stage training 58
11 Training times during 20 stage training 59
12 Comparison of identical cascade classifiers with training sets 59
13 Cascade combination of low-performance classifiers vs. high-performance cascade 61
14 Cascade training with large training set via MSL 62
15 The influence of threshold optimization 63
16 Subsampling negative samples for cascade combination 65
17 Comparison of cascade combination cost functions 66
18 Performance of increased final stage count through cascade combination . . 67
19 Performance of cascade combination from 20-stage base classifiers 68
20 Performance of cascade combination from 25-stage base classifiers 68
21 SVM bootstrap training 70
22 Performance of classifiers trained with resampled training set 71
23 Cascade combination and SVM-based training set selection vs 9000 sample base cascade 72
24 Comparison against OpenCV default cascade 73
25 Comparison against state-of-the-art 74

List of Abbreviations

CART	Classification and regression tree
ESS	Enormous sample set
HCI	Human-computer interaction
MSL	Matrix-Structural Learning
ROC	Receiver operating characteristics
RSAT	Rotated summed area table
SAT	Summed area table
SVM	Support vector machine
ASM	Active shape model
CT	Modified census transform
FFS	Forward feature selection
HMM	Hidden Markov Model
ICA	Independent component analysis
LDA	Linear discriminant analysis
ML	Machine learning
MLP	Multi-layer perceptron
MVFD	Multi-view face detection
PCA	Principal component analysis
RBF	Radial basis function
RNDA	Recursive non-parametric discriminant analysis

1 Introduction

Several reasons have lead to a tremendous amount of research in the field of face detection. It is one of the classic problems in computer vision as it is the basis for many applications such as human-computer-interaction, surveillance, identification and smart environments. Despite the work that has been done in this field robust, real-world, real-time face detection remains an open research issue.

Face detection and recognition are integral parts of human-human interaction. Faces convey a lot of information about the interaction partner, obviously identity but also mood, age and gender. Humans have the innate ability to detect, recognize and interpret faces.

The most important motivation to study face detection is, therefore, its integral role in many human-computer interaction (HCI) tasks. HCI research tries to enhance the usability of computer systems by providing other means of input and output than traditional keyboard and display-based systems. In order to ease the interaction with human users, machines need to be able to perceive their environment in a similar way as their users and supply more natural forms of interaction. Face detection is a key component of many next-generation human-computer-interfaces. Current identification methods require the user to identify themselves actively, by logging into a computer or by other means, whereas during human-human-interaction the interaction partner is automatically recognized. Knowledge about the partner may ease interaction by allowing the computer system to incorporate existing information to customize the interaction process. HCI applications have been continuously refined with the availability of more computing power and cheap video recording devices for computers. Additionally, face detection is the first processing step in solving many other HCI related problems like recognizing and identifying people, gender or facial expressions. These tasks require a human face to be located first. Once the face has been detected other classifiers can be applied to establish a person's identity, emotional state and extract additional information with which smart environments can enhance the user's experience.

Face detection has also been in research focus due to its complex, non-trivial nature. It attempts to find a non-rigid, transformable, 3-dimensional object in 2-dimensional images. Faces exhibit great variance due to lighting, different camera parameters, in-plane head and image rotation, facial expressions, occlusion by other objects or accessories like glasses or scarfs and out-of-plane head rotation (pose). Albeit being more challenging than rigid

object detection in controlled settings, faces exhibit limited amounts of deformation and thus variance in appearance as opposed to, for example, hands. Therefore, face detection is more challenging than other object detection tasks and remains a challenge, while several promising advances have already been made and quick progress is possible.

Additionally, face detection poses a challenge to most machine learning algorithms due to the high-dimensional raw data space of even small input images. The handling of large training sets is necessary in order to cover most of the variance presented by faces as discussed before. The use of more than 10,000 or even 100,000 training samples can be mandatory in order to model all variations in the appearance of faces. These large data sets and the complex nature of the classification task allow for testing and improving general machine learning algorithms.

Early work on face detection was either based on the detection of facial components, skin-color models or modeling facial appearance using the respective state-of-the-art machine learning algorithms. These early approaches were either fast or achieved high detection rates, but their computational requirements were immense. Consequently, the high computational overhead limited their applications. While many machine learning algorithms achieve good face/non-face classification results, they cannot meet real-time requirements of real-world applications.

With the introduction of Haar feature-based cascade classifiers by Viola and Jones [64], the first robust real-time applications became feasible and interest in the field of face detection surged. Since then, many variations of and additions to their work have been published. Viola and Jones contributed the combination of three important ideas. First, the use of very basic Haar-like features whose computation is extremely efficient. Second, the use of a cascade of classifiers for real-time detection and third, the use of AdaBoost for the aggressive selection of discriminative features and ensemble classifier construction.

1.1 Motivation

Despite the recent work and breakthroughs in face detection systems, training high-performance face detectors is non-trivial. A major challenge in training Haar feature-based cascade classifiers is their high training time. In order to fully exploit the potential of Haar feature-based cascade classifiers, several training parameters, i.e. the training set and the number of stages, need to be hand-tuned. Implementations of Haar feature-based

cascade classifiers are publicly available, e.g. the widely used OpenCV toolkit, along with pretrained face detectors. Unfortunately, these face detectors are far behind the results of state-of-the-art systems on publicly available databases, whereas most applications would strongly benefit from retrained specific face detectors. The need to fine-tune many parameters for optimal performance is a big challenge.

Since finding optimal parameters is not feasible computationally, the process usually involves training several cascade classifiers with a varying number of final stages, different training sets with various ratios of positive to negative training samples and several individual stage false acceptance and detection rate targets. Therefore, several cascade classifiers with varying performance will be the result of determining optimal parameters, but there are no guarantees that any of those classifiers will perform optimally.

Usually, a lot of time is thus spent on developing heuristics for cascade training. Several cascade classifiers are generated in the process and, at some point, the cascade with the best performance is selected. The other cascades are usually discarded. On the other hand, various attempts were made to combine several trained classifiers into one superior classifier. These methods usually run the base classifiers in parallel and then combine their classification results. This combination can improve the performance, but the run-time increases linearly with the number of combined classifiers.

This work explores two ways to overcome the difficulties of the training process and delivers improved cascade classifiers without increased run-time. The presented approaches focus on optimizing the training set and combining multiple trained cascades to overcome problems associated with the complexities of training. The approaches try to find a smaller, optimized training set in order to reduce training time and try to combine several trained cascade classifiers into a single classifier with improved classification performance. The final combined classifier is a combination of individual classifiers' stages, therefore the base classifiers are not run in parallel.

1.2 Contribution

The main contribution of this work can be summed up as follows:

- While classifier combination has been applied several times before, most approaches run multiple classifiers in parallel and apply more or less complex arbitration and

combination schemes. In the field of face detection, run-time efficiency is one of the most important factors. Combination efforts negate the recent run-time performance gains, while improving classification performance. The presented method to cascade combination seeks to combine several stages of multiple pretrained cascades into a single cascade. This method is, therefore, avoiding the need to run several classifiers in parallel. Training of classifier cascades is not trivial. Especially the optimization of the training parameters is time-consuming and requires a lot of experience and possible hand-tuning of stage-thresholds. Additionally, negative (and possibly positive) samples are bootstrapped, so the training sets of equally trained cascades may vary. Therefore, several cascades trained with the same training parameters may exhibit different performances. So, careful selection of individual stages from different classifiers allows to overcome the deficiencies of individual classifiers and create an improved classifier.

- The other aspect of this work is training set resampling. While many aspects of the design of cascaded classifiers have been examined, the choice of an ideal training set has not received a lot of attention except for the work of Chen et al. [11]. We examine a simple method to rank training samples and force a trained cascade to concentrate on the classification boundary between face and non-face samples. This approach is motivated by the already existing bootstrapping of the negative set for each cascade stage, where the classifier is continually forced to concentrate on more difficult examples. Also, this strategy has proved to be useful for classification in support vector machines.

1.3 Outline

The rest of this work is organized as follows. Chapter 2 provides an overview of the numerous face detection algorithms. Chapter 3 introduces the basic concepts used in the presented methods, namely cascaded Haar feature-based face detectors, information theory basics and support vector machines basics. Chapter 4 outlines the proposed methods for cascade combination and training set resampling by means of support vector machines. Chapter 5 presents several experiments that evaluate the performance of the presented approaches and tries to present some general insights into training of cascaded classifiers. Chapter 6 and 7 conclude by summarizing this work and presenting possible future work.

2 Overview of Face Detection Approaches

Face detection is defined as the task of determining the presence of faces in input images. Given an image, face detection systems determine whether faces are present in the image and, if present, supply the location and size of each face within the image. Overall at least 150 approaches to face detection have been reported by 2002 [75] and many more since then due to more recent breakthroughs.

Generally, face detection algorithms have been differentiated into the following four categories [75]:

1. Knowledge-based approaches try to apply human knowledge about the characteristics of faces. These approaches mainly try to classify faces by the relation of facial features such as eyes, nose and mouth.

2. Feature-based approaches aim to extract features invariant to changes in pose, viewpoint and lighting.

3. Template-based approaches match one or more standard patterns of faces or facial features and try to maximize the correspondence with a template.

4. Appearance-based methods classify parts of the image by applying models learned from a set of training images that represent the face and non-face classes.

Due to the immense impact of boosted classifier cascade architectures on face detection an additional subsection will detail the most recent developments in that area separately.

2.1 Knowledge-based Face Detection

Early attempts at face detection mainly consisted of knowledge-based methods. Researchers tried to encode intuitive human knowledge into algorithms or rules. These rules encode the features of faces and the relationships of the features. Care has to be taken to not make these rules too strict or relaxed resulting in few correct detections or a high number of false positives. Yang and Huang [72] developed a hierarchical knowledge-based system. Their system consists of layers of rules, from coarse rules that describe the appearance of a face applied first, to more specific rules describing features once the input image has been pre-scanned.

2.2 Feature-based Face Detection

Knowledge-based methods proved mostly only useful for face localization where only a single face has to be located within an image. Therefore, research lead to approaches that try to establish a face's location by finding features invariant to pose, viewpoint or lighting changes. These methods are called feature-based approaches. They can further be divided into methods that use facial features, texture, skin color or a combination of the aforementioned to increase the robustness.

A number of facial feature based approaches have been suggested. Generally, detection of local facial features is not very robust and leads to many false detections, while multiple passes over the image may be necessary to detect different features using different classifiers.

Facial features: A method based on horizontal and vertical projections to locate boundaries of a face and the facial features has been used by Kotropoloulos and Pitas [31]. Large changes in horizontal and vertical projections [29] are interpreted as the borders of faces. Within the designated face region, local minima of those projections determine the location of lips, eyes and nose. Another facial feature-based approach was devised by Leung et al. [33]. They use local feature detectors to detect eyes, nostrils and the nose and lip region. The face detection problem is expressed as a random graph matching problem between a graph-model of a typical face that contains distributions of typical pair-wise facial feature distances learnt from training samples and a graph constructed from face candidates based on the local facial feature detections. Leung et al. [7] have refined their approach to avoid problems associated with modeling mutual distances scale-, rotation- and translation-invariantly by using shape statistics. Further feature-based methods, for example, apply edge-maps and heuristics to locate faces [72]. Another method is based on eye and eyebrow detection after morphological pre-processing and a geometrical face model guided by eye detection [23]. Many further edge- or local feature-detector-based approaches have been published.

Texture: The second category of feature-based approaches use the unique texture of a face to distinguish it from other background objects. Using texture may be problematic due to input images of different resolution and difficult lighting, complex backgrounds may also pose a problem. Augusteijn et al. [2] developed a system for human face texture

classification based on hair, skin and other features. They use second-order statistical features (space gray-level dependence matrix) to model the textures and a cascade correlation neural network for classification. Only texture classification results were published. Dai and Nakano [15] use similar statistical features and embed color information into their models to enhance skin-colored areas in images.

Skin color: Skin color has been successfully used to detect faces. Several studies have been conducted and established the suitability of skin color for face detection, e.g. [73]. Different color spaces, including RGB, normalized RGB, HSV, YIQ and different models for the skin-color distribution have been used. Unfortunately, skin color is sensitive to changes in lighting and the resulting chrominance changes, so, it may have to be adapted. On the other hand, the detection of skin-colored pixels and image patches is very efficient. The most basic approaches use thresholds to define a skin-color region within the color space, e.g. Chai and Ngan [10] threshold Cr and Cb channels to classify skin-color values, the thresholds are learnt from training data. Methods based on histograms have also been used, e.g. histogram intersection in HSV space [49]. Other approaches use unimodal Gaussian density functions [8], mixtures of Gaussians [74] with multimodal Gaussian distributions to model the skin-color distribution. Additional work has been done to overcome the lighting problem, i.e. that the color appearance varies highly due to differences in lighting in different scenes. McKenna [40] et al. introduced an adaptive color model for tracking faces in complex environments, whereas Forsyth [18] derives equations based on a physical model to address color constancy.

Combined approaches: Since most feature-based approaches are not robust by themselves, several systems have been proposed that combine several of the aforementioned approaches. These systems tend to pre-process input images using large-scale features such as skin color or shape and verify the resulting matches using local features such as eyes, eyebrows, mouth or nostrils - Sobottka and Pitas present such a system [56]. HSV-based skin-color segmentation is followed by a connected-component analysis. Then, candidates that can be well represented by ellipses are selected and verified using facial features as eyes and mouth that are assumed to be darker than the rest of the face.

2.3 Template-based Face Detection

Another rather intuitive way to detect faces is based on matching previously prepared prototypical face templates against parts of the input image - as opposed to appearance-based methods these templates are not learnt from training data, but specifically designed by experts. Straight-forward template-based methods have difficulties dealing with changes in pose, rotation and scale. Therefore, multi-resolution, multi-scale and deformable templates have been proposed to overcome these problems.

Static templates: Samal and Iyengar [47] presented a face detection systems using silhouette templates. A set of face silhouettes, represented as arrays of bits, is generated via principal component analysis (PCA) from selected training samples. The generalized Hough-transform is then used in conjunction with the resulting principal silhouettes to localize faces. Sinha [55] proposed the use of brightness differences between facial regions, because, while the overall brightness may change, the relative brightness differences remain similar. The brightness differences are further reduced to just the sign of the difference to provide a robust feature. Then a template is generated by dividing a face into several regions that roughly correspond to facial landmarks as eyes, nose, mouth, cheeks, eyebrows and the mutual brightness differences between these regions are modeled using training data. A face is detected if these brightness-difference constraints are met.

Deformable templates: To overcome some of the previously mentioned limitations, deformable templates were introduced. They are parametrized templates which implement an elastic model that can be matched to different input face shapes. The most prominent approaches were active shape models (ASMs). Cootes and Taylor [14] as well as Kirby and Sirovich [30] use statistical models of shape and orientation of key features to detect faces. Cootes and Taylor first select a set of key features and then create a model of the statistics of the relative positions and orientations of these feature points to create a deformable active shape model. Kwon and Vitoria Lobo [32] propose a method using snakes. So-called snakelets are used to find edges, these are then used to match ellipses to face candidate regions by checking which snakelets lie along the perimeter of an ellipse via voting. These face candidates are verified with facial feature detectors using a face template with deformable eye and face outline models.

2.4 Appearance-based Face Detection

As opposed to the previously presented template-based approaches, appearance-based face detection systems try to learn the distinguishing qualities of faces and non-faces from raw training examples without human assistance.

Machine learning algorithm-based approaches: Many appearance-based algorithms use specific machine learning algorithms or statistical analysis tools to perform the pattern recognition task of classifying face and non-face image patches. Most machine learning (ML) algorithms have been successfully applied to face detection, usually in a supervised learning context where all samples are labelled correctly before training. Simply applying ML algorithms, though, is usually very time intensive, since in the appearance-based approach every possible sub-window, that is windows within the image of different location and scale, has to be classified as face or non-face.

Distribution-based approaches: Sung and Poggio claim that appearance-based approaches have a few advantages over, for example, template-based, manually designed systems [55, 61]. Appearance-based methods learn from large amounts of training data and, therefore, reduce the dependence on domain-specific knowledge and reduce the risk of mistakes due to incomplete or inaccurate knowledge. They also claim that, by adding misclassified patterns to the training set, the number of falsely accepted non-faces and the number of correctly detected faces can be arbitrarily tuned. The last claim is not generally accepted as, although the quantity of training samples is important, quality of training samples becomes more and more important as more samples are used to train a system.

Sung and Poggio [61] presented a method to detect faces based on modeling the distribution of face and non-face patterns. Their approach models the distribution of face and non-face patterns as several multivariate Gaussian distributions using a cluster mean and a co-variance matrix. These clusters are extracted via a modified k-means algorithm from training data that is represented as vectors by concatenating image rows. Experiments found 6 clusters for both face and non-face classes to perform best. Then, a multilayer perceptron (MLP) is used to perform the face/non-face classification using a special distance metric. The distance consists of two components, the first is the normalized Mahalanobis distance of the input pattern projected onto a subspace constructed from

the largest eigenvectors of the clusters. The second component is the Euclidean distance between the test pattern and the reconstruction of its projection onto the subspace. An important aspect of their work is the selection of training samples for the non-face class. Since the non-face class contains substantially more possible patterns than the face class, they select the negative samples in a bootstrap fashion. Starting with a small initial random set of non-face samples and a set of face patterns an MLP face detector is trained, then, misclassified non-face samples are added to the negative training set and the process is repeated until a desired amount of samples has been collected. This method of selecting negative training samples has since been used by other groups, e.g. Rowley et al. [45] or, recently, boosted classifier cascades. While Sung and Poggio model the distribution of global face patterns in a high-dimensional space, Schneiderman and Kanade [53] model the joint probability of spatial relationships and local appearance of faces. They estimate the posterior probability function of an input pattern being a face or non-face.

Linear transformation-based methods: Subspace methods project input patterns into a lower-dimensional subspace in order to ease classification. Several projections are known and have been used, principal component analysis (PCA), linear discriminant analysis (LDA) and independent component analysis (ICA) are the most widely used transformations. While PCA tries to retain as much variance of the input data as possible and is, therefore, optimal for compression, LDA has been designed for classification by producing a projection that best separates the different classes, i.e. it maximizes the ratio of between-class scatter to within-class-scatter. Turk and Pentland presented a well-known PCA-based face detection and recognition method [62]. They create a subspace using PCA from face training samples and project face patterns into the subspace to model their distribution. For classification, an input pattern is projected into that space and the distance of the sample and the facespace is computed as the likeliness of the pattern being a face.

Neural network face detectors: Neural networks have been widely used successfully to perform many different pattern recognition tasks in the past and they have also been applied to the face detection task. Neural networks are able to model highly non-linear decision boundaries in the input data space. Therefore, neural networks allow the construction of complex decision functions for face/non-face classification. On the other hand, network type and network topology have to be carefully chosen in order to avoid

overfitting while retaining a network that is powerful enough to perform the classification task. The optimization of these parameters is time-consuming and only aided by heuristic guidelines. The most important neural network-based face detection approach has been developed by Rowley et al. [45] - their approach is still cited in some more recent publications as a baseline algorithm. They designed a multi-layer neural network face detection system based on receptive fields that were chosen to allow the network to detect individual or combined facial features. First, multiple neural networks are applied to the input image to classify all possible sub-windows. Then, heuristics are applied to handle overlapping detections of the same face at different scales and slightly different positions to discard false positives that were found not to exhibit these multiple detections. Instead of heuristics an arbitration between the detection results of the networks, like AND or OR operations or an arbitration neural network, can be applied. Rowley et al. used a special, faster pre-scanning network to identify likely face candidates quickly in order to speed up the detection process. This first network was larger than the original networks and was able to roughly locate face candidates quickly before the more extensive classifiers were applied to the face candidates, this can be interpreted as an early cascaded architecture. The presented approach cannot detect rotated faces, therefore Rowley et al. proposed an extension of their method [46]. They used a prepended router network that was trained to estimate the rotation of face image patches. These patches were then rotated to the upright pose before they were passed to the original system. Other neural network based approaches have been devised, Agui et al. [1] used two parallel networks fed with direct intensity values and Sobel-filtered intensities, Soulie et al. [58] applied time-delay neural networks on wavelet transformed input images for scale invariance.

Support vector machine-based approaches: Osuna et al. [43] were the first to apply another successful machine learning algorithm to face detection, support vector machines (SVMs). SVMs construct a linear hyperplane that separates two sets of input data represented as image row concatenated vectors. As opposed to other linear classifiers, SVMs aim to maximize the margin between the classes and the separating hyperplane and thus minimize the expected generalization error instead of just the training error. Additionally, the data points are projected into a high-dimensional space by means of a kernel function before classification to allow for non-linear classification in the input space. The system of Osuna et al. is trained in a similar manner as Sung and Poggio [61] have done, by bootstrapping the negative non-face samples. They trained the classifier

on normalized, vectorized images (rows concatenated) 19x19 input images. The detection is performed by applying the classifier to all possible sub-windows of the input image at different scales. The SVM-based approach slightly outperforms the system of Sung and Poggio. Another SVM-based system is presented by Heisele et al. [24], but their use of overlapping appearance-based component classifiers, albeit not invariant to rotation and scale, makes it a mixture of feature-based and appearance-based methods. They train 14 linear-SVM component classifiers that mostly cover the whole face. The component-based layout is supposed to make the approach more robust to pose and illumination changes than traditional holistic appearance-based approaches. Finally, an SVM is used to verify that the configuration of the individual feature detectors constitutes a valid face.

Several other machine learning paradigms have been applied to face detection. For example, Samaria [48] applied Hidden Markov Models (HMMs) for face detection by representing facial regions scanned line-by-line as states of the model, Yang et al. [76] applied a sparse network of winnows as classifier.

Classifier cascades: More recently, a new face detection approach has been proposed by Viola and Jones [64], referred to as cascades of boosted ensembles. As opposed to previous works, their method was able to perform real-time face detection in a robust manner achieving state-of-the-art performance on evaluation data sets. The success of the approach is due to several contributions. In order to achieve high performance a so-called cascaded classifier design was applied. The cascaded design is efficient and fast for rare-event detection tasks, like face detection, by rejecting rather easy cases early with only few feature computations. Additionally, conceptually simple and, by themselves, not very powerful thresholded rectangle features were used. These are, on the other hand, fast to evaluate using an integral image representation. Since a large quantity of these features exists even within small scanned windows, a greedy approach based on AdaBoost is used to select the most suitable features. A linear combination of these thresholded features is used to build each stage of the classifier cascade. The combination coefficients are chosen according to the AdaBoost algorithm. This seminal work has sparked high interest in the research community and many alternative approaches to several aspects of the work have been published. Fellow researchers have explored alternative features, weak learners, methods to train the weak learners and boosting schemes.

2.5 Cascades of Boosted Ensembles

The following Chapter will present some of the recent research in face detection associated with cascades of boosted ensembles.

2.5.1 Alternative Boosting Algorithms

Most of the immediate work inspired by the publication of Viola and Jones [63] has been the exploration of alternative boosting methods. Boosting algorithms combine several simple learners into a single powerful classifier - see Chapter 3 for details. The boosting algorithm has seen many improvements over the initially devised AdaBoost-based approach by Viola and Jones [63]. Several variants of boosting have been applied to object detection tasks in conjunction with Haar-features and classifier cascades, e.g. Discrete AdaBoost [19, 52] Real AdaBoost [26, 52], asymmetric AdaBoost [65], FloatBoost [79], GentleBoost [20], Kullback-Leibler boosting[37] and WaldBoost [57]. Among the plethora of boosting variants, Discrete AdaBoost and Real AdaBoost seem to be the most widely used variants.

Initially, Viola and Jones [63] used discrete AdaBoost proposed by Freund and Schapire [19, 52]. It was the first practical, polynomial-time boosting algorithm. AdaBoost works by iteratively finding a simple weak-learner that best classifies the labelled, weighted training data. Iteratively, the algorithm reweighs all samples, increasing weights for misclassified samples, and then finds a new best-performing classifier on the newly weighted training set. At the end of each iteration the new classifier is added to the ensemble of the classifiers with an appropriate weight.

Boosting algorithms seem prone to overfitting and learning noise, in order to overcome that weakness Gentle AdaBoost has been proposed [19, 52]. Gentle AdaBoost tries to reduce the susceptibility to noise by reducing the weights assigned to outliers. A study by Lienhart et al. [35] features a comparison of Discrete AdaBoost, Real AdaBoost and GentleBoost and suggests that GentleBoost outperforms the other two boosting algorithms for face detection.

Another variant of AdaBoost is Real AdaBoost [20], it was used by Lienhart et al. [35] and a generalized version of Real AdaBoost for multi-class-multi-label classification was used by Huang et al. [25, 26] in multi-view face detection. Real AdaBoost is an extension of AdaBoost that uses real-valued weak learner outputs instead of simple binary class

labels in the basic two-class case. Real-valued outputs can improve the classification over the coarse binary class label information.

Kullback-Leibler Boosting, or KLBoosting, has been proposed by Liu and Shum [37]. Feature-selection and boosting are based on Kullback-Leibler divergence, a measure of distance between two probability distributions from information theory. The authors seek to maximize the Kullback-Leibler divergence of histograms of face/non-face classes that were generated by projecting the data to 1D-histograms of KL features that are approximated with simple wavelets. Also, they use the Kullback-Leibler divergence to derive the weight coefficients of weak classifiers when added to the ensemble.

AdaBoost turned out to be not ideal for applications in cascaded frameworks, since the original algorithm optimizes the overall error, whereas a cascaded framework is mostly concerned with the minimization of false negatives. The cascade architecture is strongly dependent on high correct classification rates of individual stages, while modest false acceptance rates per stage are tolerable. Viola and Jones [64] updated their face detection approach by introducing asymmetric AdaBoost that modifies the AdaBoost sample reweighting scheme after each round of boosting to increase the weight of positive examples proportionally.

Li and Zhang proposed FloatBoost, an extended boosting strategy that includes backtracking after each iteration of boosting in order to remove ineffective or harmful weak classifiers that increase the error rate [79].

For a near optimal trade-off between computational efficiency and minimal error rate, Sochman and Matas [57] presented a new boosting algorithm based on Wald's sequential probability ratio test that optimizes the average decision time given a target error rate.

2.5.2 Alternative Feature Sets

The initial work by Viola and Jones [63, 64] used basic rectangle features, called Haar-wavelets or Haar-filters. These are simple features consisting of a weighted sum of two to four rectangular areas of summed image intensities - see Chapter 3 for details. This large set of possible features forms an overcomplete set of bases that is able to represent the original image intensities. Their advantage is their conceptual simplicity and the ability to compute them very efficiently. Using an integral image representation only four table lookups per rectangle are necessary. A disadvantage is their limited discriminational ability for classification.

Lienhart et al. [35] introduced an extended set of features that include features rotated by 45^{o} that are equally efficient. The new features are a generalization of the concept introduced by Viola and Jones [63] and some are inspired by biological features of human vision. These features increase the set of available basis functions for classification and allow for slightly better detection performance of about 10% decreased false alarm rate at the same hit rate [35].

A simple modification for profile face detection has been proposed by Ishii et al. [27]. Instead of connected rectangle features, this approach allows the use of differences of two disconnected rectangles to improve the classification of profile faces that are hard to model with the traditional feature set due to the lack of many discriminative edges or lines as in frontal faces.

Completely different feature sets were explored by Zhang et al. [78] and Wang and Ji [67, 66]. Weak classifiers based on Haar-wavelets are fast to evaluate but may lack the ability to discriminate well between face and non-face samples in later cascade training stages, since the bootstrapping approach successively concentrates on hard samples. In order to overcome this problem, different feature sets were suggested. On the other hand, other features are usually computationally expensive, thus, Haar-wavelets are still used in the initial stages. Zhang et al. [78] suggest the use of global PCA-based features at later stages once the local Haar-features reach the discriminatory limit and error rates approach 50%. After the face eigenvectors have been constructed via PCA from training data, the boosting algorithm selects the new PCA-based global features based on their discrimination ability using 1D feature value histograms, not their eigenvalue rank. Wang and Ji [66] suggest the use of the recursive non-parametric discriminant (RNDA) analysis, since PCA is designed for retaining the most variance in the data after projection and is not optimal for discrimination. RNDA does not assume a Gaussian form of class distributions and is thus a generalization of LDA and is also a global feature. Again, feature histograms were used to model the distribution of features and, thereby, find the most discriminative feature corresponding to eigenvectors of the RNDA process. Wang and Ji applied their boosted RNDA framework to multi-view face detection [67].

Liu and Shum [37] used global projection features based on Kullback-Leibler-Features (KL-features) that are linear projections that maximize the Kullback-Leibler divergence of 1D-histograms of the two classes. They approximated the optimal KL-features by linearly combining a subset of Gabor-wavelet or Haar-wavelet features at different orientations,

scales and translations.

Another different feature type was used by Fröba and Ernst [21]. They propose the use of the modified census transform, a local transform that generates a string of bits representing which neighborhood pixels have a lower intensity than the current location. The modified census transform (CT) uses the mean neighborhood intensity of the current locations as an anchor to be more expressive. The neighborhood size is theoretically not limited, but sizes of 3×3 were used to capture local details and keep computational overhead low. For every image pixel and its neighborhood a CT feature can be constructed that describes the resemblance to a structural kernel. These features can be efficiently evaluated by using the resulting bit-sequences as an index into a lookup table. A cascade of boosted classifiers is then trained as in other approaches with a small number of image locations selected as features, the last stage is trained to evaluate all locations. The presented approach performs well on the CMU+MIT database and is very fast due to the efficient and sparse calculation of CT features.

2.5.3 Weak Learners

Basic, thresholded Haar-features have been widely used as features for face detection. Lienhart et al. [35] and Brubaker et al. [5] doubt that these simple, thresholded features suffice for difficult classification tasks. Both used low height Classification and Regression Trees (CARTs) [4] with Haar-wavelet-based nodes instead. Their results are somewhat contradictory, Lienhart et al. [35] reported modest improvements whereas Brubaker et al. [6] found significant improvements.

Huang et al.[25] introduced piece-wise functions as weak learners. Piece-wise functions divide the feature space into n equal bins and output a constant value for each bin that describes the divergence of positive and negative samples' feature values. The authors claimed that piece-wise function learners allow the learning procedure to converge faster, while allowing for more robust classification at later stages. It is not clear whether the reported results are due to the new weak learner or the other novelty presented in the paper, the Vector Boosting approach that will be explained later.

2.5.4 Cascade Training Procedure

The cascade training process as proposed by Viola and Jones [64] has a few draw-backs. The most obvious problem is the ad-hoc choice of individual cascade training goals. The authors chose the same hit rate and false alarm rate for all trained stages. These choices may not be ideal. Luo [38] described a method to globally optimize the cascade architecture post-training and set the final stage thresholds accordingly. Instead of simultaneously optimizing stage classifiers and stage thresholds, the approach optimizes the thresholds of a pre-trained cascade architecture. Two solutions were presented, a simple algorithm assuming that stage classifiers can be optimized independently and a dependent greedy search solution to find an optimal operating point by adjusting stage thresholds. Since the approach does not address the adjustment of thresholds during the training phase that influences bootstrapping, McCane et al. [39] suggested to model the cost of execution of individual stages. The cost function is two-dimensional and depends on both false acceptance and detection rates and is therefore difficult to model. Hence, McCane et al. [39] created a family of ROC curves based on an incrementally built monolithic classifier that were used to estimate the parameters of the stage-execution-cost model. Then global false acceptance and detection rates could be minimized using the cost function and optimal individual operating points for stages could be found. Brubaker et al. [5, 6, 59] viewed the overall performance of the cascade as a random variable and trained individual cascade stages with a minimal number of features so that the probability of meeting the overall cascade goals was sufficiently high. Each stages' performance iwas evaluated on an independent evaluation set and the probability of meeting the overall cascade goals was tested given the current results on the evaluation set. A cost function was defined based on the probabilities of meeting the overall goals and minimized during training. Chen and Yuille [13] proposed a method to learn a cascade that is as fast as possible given a desired overall accuracy. They tried to minimize the average processing time, basically trading off false acceptance rate against a stage's execution time.

Dundar and Bi [16] proposed a different training architecture, called AND-OR learning. Their method avoids the independent treatment of stages during the training process and all stages were trained in a joint fashion. Instead of the usual greedy scheme that trains classifiers sequentially, the approach optimizes all classifier stages in parallel by providing mutual feedback between stages. Positive and negative sample are assigned different loss functions as positive samples have to pass all stages (AND-operation) and negatives have

to be rejected by a single stage only (OR-operation). The approach iteratively optimizes the overall cascade performance by adjusting a single stage's parameters while fixing all other stage's parameters.

Chen et al. [11] presented an approach to optimize the training set for training a classifier cascade. The choice of good training samples plays an important role in the quality of a trained classifier. The presented approach covers generating a large training set that should cover the whole face space and a resampling technique that selectes samples as to densely and evenly cover the whole face space. An initial training set was first expanded by using a genetic algorithm that crossed parts of faces (crossover operation) and applied relighting (mutations) to generate new samples. After each round of the genetic algorithms, the current generation of generated samples was checked against a face classifier trained with the last-generation data. Samples with too large variations from the last generation were discarded because of their low resemblance with faces. The process was repeated several times. Once a sufficient amount of samples had been generated, a manifold space was created by means of the Isomap algorithm to represent the local distances of the face space in a lower-dimensional space. Then, sparse areas of the face space were filled by interpolated samples. Finally, the set was resampled to a sufficiently large set of samples that evenly covered the original face space without dense or sparse areas. The final classifier presents one of the best reported results on the CMU+MIT database. A one-class SVM was used as an additional last step to reject further false positives, thereby lowering the false alarm rate compared to strict cascade classifiers.

2.5.5 Cascade Architecture

Most of the complexity of current face detectors lies in their use of the cascaded classifier architecture, several parameters like individual stage goals, number of stages, thresholds and more have to be tuned to achieve state-of-the-art performance. Due to the training time in the order of days, optimizing these parameters is difficult or impossible. Therefore, several enhancements have been proposed to overcome the problem of threshold choices by giving up a strict cascade architecture. Bourdev and Brandt [3] coined the term soft cascade for an architecture that departs from strictly separated stages. Instead of deciding whether to accept a sample after a sequence of features, i.e. after every stage, and restarting the summation and thresholding, feature values are continuously thresholded

in a monolithic classifier so that a sample may be discarded after every feature. The approach by Bourdev and Brandt [3] goes one step further than the Boosting Chain presented by Xiao et al. [70]. The Boosting Chain avoids resetting the feature value sums after thresholding at each stage. This is done in order to allow samples to pass a stage they might have otherwise failed, because they had a large margin on previous stages. Sochman and Matas [57] also implemented a similar monolithic design for making decisions after every feature evaluation in their WaldBoost framework.

Another problem of the traditional cascade architecture design is the handling of multi-view face detection (MVFD). MVFD has additional requirements in terms of run-time performance and the classification task becomes more difficult. Huang et al. [25] propose the use of width-first search trees in conjunction with a multi-class, multi-label version of Real AdaBoost called Vector Boosting. The width-first search tree outperforms a simple depth-first tree design because multiple paths may be taken simultaneously before making a final decision. Trees in general allow for better multi-view classification, since the sample sets at nodes become smaller and the discrimination from background samples is thus easier due to less within-class variance. Other tree-based approaches are a pyramid-based structure presented by Li et al. [34] and a tree-based approach by Viola and Jones [28] with detectors trained for different views and rotations.

2.5.6 Training Time Improvement

Although cascades of boosted ensembles exhibit real-time run-time performance, training time ranges from days to weeks. Several approaches have been suggested to reduce the required amount of time. The factors that affect the training time are the amount of training samples and feature-set size that depends on the actual image dimensions of training samples. The traditional training approach has a run-time of $O\left(N \cdot T \cdot log\left(N\right)\right)$ where N represents the number of samples and T is the number of features. The most successful and straight-forward approach to reduce the training time is caching [68]. In order to execute AdaBoost, the classification of each prospective classifier for each individual sample has to be computed based on the current weight distribution. In the case of Haar feature-based weak learners, the classification is a simple thresholded decision on the scalar difference of rectangle areas. Therefore, for several iterations of AdaBoost, only the weighting of samples but not the computed rectangle areas themselves change. Precalculating and caching the differences of these rectangle sums for all samples is, thus,

the fastest way to train a face detector. Caching, on the other hand, is limited by the amount of available memory, training with 40000 samples and a 20×20 window ($\tilde{}$ 65000 basic features) requires several GB of RAM. Another method to reduce training time is to avoid the constant reevaluation of feature values. A method that treats features as high-dimensional random vectors and models the feature values with Gaussian distributions was presented by Pham et al. [44]. This approach avoids the recalculation of feature values to determine optimal thresholds by using Gaussians to model the distribution of feature values. Therefore the training time can be reduced to $O(Nd^2 + T)$, where d is the amount of pixels within the probed feature region. Another possibility to reduce training time is to select features directly to either decrease the false acceptance rate or to increase the maximal hit rate directly using Forward Feature Selection as described in [69], but the guarantees proved for boosting may not hold for FFS. Also, feature filtering has been explored by Brubaker et al. [6], they explore methods to reduce the size of the feature pool. They suggest the use of different filters. The simplest approach is a ranking filter that eliminates features whose values have poor correlation with class labels. But that may leave only redundant features whose combination is not useful. Therefore slower filters based on pairwise mutual information of features have also been tried. They were compared to a random filter selection of features. While ranking and mutual information filters performed better than randomly selected features, there was a decrease in detection rates of final classifiers with about factor four performance gains. The use of filtering to eliminate Haar features did not meet the authors' expectations.

Another approach to reduce training time is the Matrix-Structural Learning (MSL) approach by Yan et al. [71]. The approach extends bootstrapping to the positive set as well as the negative set. Negative samples are bootstrapped at the end of each stage's training, whereas positive samples are bootstrapped during the training of stages. Initially, a cascade stage is trained with the current negative set and a random positive set, then misclassified positive samples are added to the training set and the stage is retrained until the detection rate on the overall global set is sufficient. Then, after bootstrapping the negatives, the process is repeated for the next stage. See Chapter 3 for details.

2.5.7 Run-time Efficiency Improvement

Bourdev and Brandt's [3] soft cascades allow to optimize both cascade performance and run-time speed. Their approach produces monolithic classifiers that outperform compa-

rably fast cascaded classifiers. The trade-off between run-time efficiency and detection performance can be optimized by analyzing receiver operating characteristics (ROC) surfaces instead of static ROC curves. Like the soft cascades, the work of Sochman and Matas [57] and Xioa et al. [70] implicitly improves classification performance by allowing for arbitrarily preempted points to stop execution of the classification.

McCane et al. [39] built an empirical cost model based on ROC families of a monolithic classifier to model the trade-off between run-time and detection rate. They uses a cost model that includes both false alarm and detection rates to optimize the cascade training parameters.

Brubaker et al. [5] presented another method to optimize the run-time performance of a cascade architecture by means of an abstract cascade execution model. This execution cost model allows to decide the minimal amount of required weak hypothesis and a better exploitation of the false alarm vs. correct detection rate trade-off by optimal choice of stage thresholds. Additionally, existing stages can be split into smaller stages in order to discard negative samples with the evaluation of less features. The work by McCane et al. [39] and Brubaker et al. [5, 6] address the issue of choice of thresholds during training as opposed to Luo's [38] post-training threshold optimization strategy.

2.5.8 Classifier Combination

Classifier combination has been explored by Rowley et al. [45], Sung and Poggio [60] as well as Viola and Jones [63]. Rowley et al. [45] were the first to build a face detection system that incorporates a combination and arbitration of several classifiers. First, the neural network-based face detector was run over the entire image and detections were collected. Then, a heuristic was applied to these detections where detections were grouped and single detections were removed, as face detections occur at slightly different positions and scales whereas false positives usually do not. Finally, arbitration between multiple networks was applied to further reduce the number of false alarms. The arbitration schemes included logical AND, OR and voting operations as well as a separate network trained to arbitrate detections of different classifiers.

Viola and Jones' [63] first publication also contained a simple combination scheme to improve classification performance. A simple majority voting was carried out between three similarly trained cascades. They reported an increased hit rate and a reduction of false alarms even though the cascade classifiers were similarly trained and thus redundant.

Grosvenor [22] presented a general object detection integration framework. The integration approach takes a set of cascades and statistical information of their interdependence and creates a classification tree. The method is supposed to allow the automatic combination of more specific detectors for problem subsets, e.g different poses in multi-view face detection, into a general classifier that is more efficient than parallel evaluations of the individual classifiers.

3 Methodology

This chapter presents the basic principles used within this work and explains their theoretical basis.

3.1 Face Detection

Face detection is the process of determining whether a face is present within an input image and, if present, returning the location and size of the human faces. The following sections explain the basic concepts that are used to perform face detection in this work. One key criterion for face detection systems is real-time run-time performance. Viola and Jones [63, 64] were first to present a state-of-the-art performance, real-time face detection system. The concepts that allow for highly accurate object and face detection at high speeds are explained in this Chapter.

Haar wavelets form the basic features for face detection, they are simple but fast to compute. Face detectors are trained using AdaBoost, a powerful machine learning algorithm. In order to further improve detector performance, a chain or so-called cascade of classifiers is used to reduce the average execution time of classifiers.

3.1.1 Haar Features

Haar features are simple rectangular features reminiscent of Haar basis functions first suggested for use in face detection by Papageogiou et al. in [41]. The over-complete feature set consists of weighted combinations of two or more rectangles' intensity sums.

Individual features can described as

$$feature_I = \sum_{i \in I = \{1,\ldots,N\}} \omega_i RecSum(r_i), \tag{1}$$

with weights ω_i, rectangles r_i and number of rectangles N, the rectangle area intensity sum function $RecSum(r)$ will be defined later. The weights are restricted to opposite signs in order to yield rectangle area differences, the number of rectangles is usually restricted to a maximum of four, although more elaborate features are possible. The features resemble Haar-basis functions, they are constructed to capture edges, lines and

diagonals, sometimes center-surround features are added to the set. See Figure 1 for examples and Lienhart et al. [35] for details.

3.1.2 Integral Image Representation

A contribution of Viola and Jones approach to object detection [65] was the use of integral images to calculate rectangle sums necessary for the calculation of the Haar-like features, thus allowing fast computation and evaluation. An integral image, also referred to as summed area table (SAT), contains the sum of an input image's pixel intensities within a rectangle from the top left corner of the image to the current position (x, y). The integral image value $ii(x, y)$ at coordinates (x, y) is defined as

$$ii(x, y) = \sum_{0 \leq x' \leq x, 0 \leq y' \leq y} I(x', y'), \tag{2}$$

with $I(x, y)$ being the image pixel intensity at coordinate (x, y).

The integral image can be calculated with a single pass over the image as

$$ii(x, y) = ii(x, y-1) + ii(x-1, y) + I(x, y) - ii(x-1, y-1), \tag{3}$$

with $ii(-1, y) = ii(x, -1) = ii(-1, -1) = 0$.

Using the integral image representation rectangle area sums can be calculated using only four lookups. The rectangle sum $RecSum(r)$ with rectangle $r = (x, y, w, h)$ at position (x, y) with width w and height h, is defined as

$$\begin{aligned} RecSum(r) \quad = \quad & ii(x-1, y-1) + ii(x+w-1, y+h-1) \\ & -ii(x-1, y+h-1) - ii(x+w-1, y-1). \end{aligned} \tag{4}$$

Figure 1: Example Haar-like features consisting of two, three and four base rectangles.

Lienhart et al. [36] proposed an extension to the set of Haar-features to include rotated features. The calculation overhead is only slightly increased as the rotated summed area table can be calculated using in a single pass over the input image as well. The rotated summed area table (RSAT) is defined in analogy to the summed area table above, but represents the sum of pixel intensities of rectangles rotated by 45° whose left corner lies at the image origin and the bottom corner at (x, y). The $RSAT$ at point (x, y) is defined as

$$RSAT(x, y) = \sum_{x' \leq x, x' \leq x - |y - y'|} I(x', y'). \tag{5}$$

The calculation of RSAT elements can be performed using the following formula

$$\begin{aligned} RSAT(x, y) &= RSAT(x - 1, y - 1) + RSAT(x + 1, y - 1) \\ &\quad - RSAT(x, y - 2) + I(x, y) + I(x, y - 1), \end{aligned} \tag{6}$$

with $RSAT(x, y) = 0, \forall x, y : x < 0, y < 0$.

Rotated rectangle's, $r_{rot} = (x, y, w, h)$, pixel intensity $RecSum(r_{rot})$ can thus be calculated using

$$\begin{aligned} RecSum(r_{rot}) &= RSAT(x - h + w, y + w + h - 1) + RSAT(x, y - 1) \\ &\quad - RSAT(x - h, y + h - 1) - RSAT(x + w, y + w - 1). \end{aligned} \tag{7}$$

3.1.3 Boosting

Boosting is a meta-algorithm used to improve the classification performance of a base classifier, also called a weak classifier, by continuously combining weak classifiers trained with different sample weights into a final, accurate classifier called a strong classifier. The basic concept of boosting is the combination of simple base classifiers h_i into an accurate classifier $f(x)$, where x denotes a sample:

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x).$$

The α_t denote coefficients or weights of the individual weak classifiers in the final strong classifier.

Boosting is based on the idea that many simple weak classifiers are easier to construct than a single complex, strong classifier. Therefore, boosting algorithms combine several weak classifiers into a strong classifier that is nonetheless able to perform the desired classification task well. Boosting algorithms iteratively run a base learning algorithm with a set of training examples that is differently weighted each round to put emphasis on misclassified samples. The two main challenges of boosting are the choice of training sample weights each round and the combination rule for building a strong classifier from weak classifiers. Boosting algorithms put emphasis on training samples that are misclassified by increasing those samples' weights each round. The final classifier is a simple linear combination of weak classifiers.

The boosting approach is independent of the base machine learning algorithm, also called weak learner or weak classifier. Several algorithms have been used in conjunction with boosting, for example decision trees have been found to work well for certain classification tasks. Face detection applications use very primitive weak learners, simple thresholded features. The threshold is set so that the feature values after projection separate the classes as well as possible.

Several boosting algorithms have been devised, AdaBoost (Adaptive Boosting) by Schapire and Freud [19, 51, 52] is the most popular boosting approach. Schapire and Freud were the first to present a practical polynomial time algorithm that found wide acceptance. A schematic overview of the boosting algorithm can be seen in Algorithm 1. AdaBoost is given a training set $(x_1, y_1), \ldots, (x_n, y_n)$ as input, consisting of samples x_i from the domain or instance space X and labels y_i from the label set Y, usually $\{-1, +1\}$. The initial weight distribution is set so that all samples have equal weights $1/n$. The parameter α_t is chosen to reflect the importance of a weak learner h_t, in the binary case usually

$$\alpha_t = \frac{1}{2} ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right), \tag{8}$$

where ϵ_t is the error of the classifier h_t, i.e. probability of h_t misclassifying samples $x_i \in X$ given the current weight distribution D_t. Each round, the distribution of weights D_t is updated according to the rule in Algorithm 1.

By adding a weak classifier to the ensemble that minimizes the error given the current weight distribution and weighting it according to Equation 8, AdaBoost decreases the training error, i.e. the number of misclassified training samples, exponentially. It does so by aggressively maximizing the so-called margin of misclassified training examples by increasing their weight. The choice of D_t determines how fast the training error can be reduced. A choice of D_t according to Algorithm 1 reduces the training error as fast as possible. The margin of a training example x_i is defined as

$$m(x_i) = \frac{y_i \sum_{t=1}^{T} \alpha_t h_t(x)}{\sum_{t=1}^{T} |\alpha_t|}. \tag{9}$$

It lies between $[-1, +1]$ and correlates with the confidence in the classification of sample x_i. Therefore, it is positive if a sample x_i is correctly classified. Large margins have been proved to yield lower generalization errors on unseen testing data. AdaBoost has been shown to maximize the margin aggressively, since it concentrates on those examples that have the smallest margins. As can been be seen in Equations 9 and 8 α_t is chosen to minimize the classification error of samples.

Algorithm 1 AdaBoost boosting algorithm[50]

Initialize $D_1(i) = 1/n$
For t = 1, . . . ,T:

- Train base learner using distribution D_t.
- Get base classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in R$,
- Update:
  $D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$, Z_t is chosen so that D_{t+1} will be a distribution

Final classifier:
$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

3.1.4 Weak Learners

Boosting algorithms do not rely on a pre-determined base learning algorithm. Therefore, the choice of a weak learner to be boosted remains. Popular choices are decision trees. Viola and Jones [63] initially proposed the use of thresholded Haar-features as weak learners. A Haar-feature is evaluated on the corresponding input image sample. If the feature value exceeds a certain threshold, the output is 1, otherwise 0. The threshold is chosen to satisfy the stages minimal detection rate on all training samples. Lienhart et al. introduced Classification and Regression Trees (CART) as weak learners [35].

This work uses simple thresholded weak learners constructed from Haar-features. Using CARTs can slightly improve performance, but studies do not agree [35, 39]. The performance gain does increase the complexity of features and increases the difficulty of cascade performance analysis. Therefore simple thresholded Haar-features were used.

3.1.5 Cascades of Boosted Ensembles

In order to decrease the average execution time of rare event detection systems, for example face detection, the cascade architecture has been introduced. Rare event detection tasks typically consist of a very imbalanced ratio of positive and negative samples during testing, thus the majority of execution time is spent rejecting negative samples. Viola and Jones [64] inspired the wide-spread use of cascaded detectors, especially in face detection. The basic idea of cascaded detectors is to reduce the average execution time by applying a chain of detectors that mimic a degenerate decision tree. For each presented testing sample, several decisions are made whether the sample should be rejected as negative or passed to the next element of the chain for further inspection. Only if the sample passes all classifiers of the chain, it is accepted as a positive sample. The chain, or cascade, architecture is organized so that the first classifiers mainly deal with simple samples and let later stages handle more complicated decisions. Thus, early classifiers are kept simple and the execution time low. Stages need to have very high detection rates, while fairly high false acceptance rates around 50% are acceptable. That means that the first stage is already able to reject 50% or more of all presented negative samples in a time-efficient manner. There is no need to evaluate all stages as in a slow monolithic classifier. All stages are only executed in the rare case of classifying a positive sample.

The global detection rate D_g and false acceptance rate F_g are the product of the individual,

chained stage detection and false acceptance rates d_i and f_i of stage i. They are thus defined as

$$D = \prod_{i=0}^{N} d_i, \tag{10}$$

$$F = \prod_{i=0}^{N} f_i \tag{11}$$

in a cascade with N stages.

In face detection, cascades consist of classifiers trained via AdaBoost based on thresholded Haar-feature weak learners. Each stage of the classifier is trained using a modified AdaBoost approach. AdaBoost is used to select and combine the features into a strong classifier. A weak classifier is constructed from a Haar-feature by thresholding the resulting feature value $h(x)$, so that the error on all training samples is minimized. Then, weak classifiers are added to the stage ensemble until desired stage false acceptance and detection rates are met. Equations 10 and 11 give rough ad-hoc guidelines for choosing the detection rates. Viola and Jones [64] therefore chose all stage detection rates to be set at $d_i = D^{1/N}$and $f_i = F^{1/N}$. Accordingly, each stage has to have a very high stage detection rate, in excess of 99.9%, whereas false acceptance rates may be moderately high around 50% for 20-stages architectures for example. These choices are not necessarily optimal.

3.1.6 Optimal Thresholding for Cascaded Ensembles

A cascade architecture consists of a number of stages with individually defined operating points. During training, several parameters have to be optimized - the stage classifiers, i.e. the features, have to be constructed, optimal thresholds have to be determined for each stage and the number of stages has to be decided. Viola and Jones [64] pointed out that optimizing all these parameters simultaneously is not trivial. Therefore, ad-hoc stage detection rate and false alarm rate targets, that is stage thresholds, were chosen that may not be optimal globally. Also, the number of stages is fixed initially or the training is aborted early if the goals cannot be met after a number of stages. Luo [38] devised a high-level abstraction model for representing cascade architectures. In this model, a cascade consists of n sequential stages, each stage consists of a thresholded node classifier that projects input samples x_i to a class label $y_i \in Y = \{-1, +1\}$. Each stage classifier

is represented as a function $C(t, x_i) \rightarrow Y$, or simply $C(t)$, whose decision boundary is defined by a threshold t. The goal of the abstraction model is to devise a method to optimize the stage thresholds t_i for all stages $i \leq n$, since the previous ad-hoc targets may not be ideal. Reducing the threshold and false alarm rate, thus raising the detection rate, of stage i and increasing the threshold of stage j, $i \neq j$, thus lowering its false alarm rate may raise the overall detection rate if done correctly.

3.1.7 Bootstrapping

The term bootstrapping is derived from "pulling oneself up by the strap of the boot" and describes a method to incrementally train a classifier from an exhaustively large set of samples or, in general, a process that generates a complex system from a simple initial system. It is an iterative method that enlarges an initial set of rules or samples by generating new rules or acquiring new samples with help of the current state. In classification, first, an initial population is randomly sampled from the large set of samples. Then a classifier is trained with this initial set. In order to improve the performance, the classifier is forced to focus on misclassified samples. Therefore, after each training step the training set is augmented with a number of misclassified samples and the classifier is retrained. This helps the classifier to correctly classify a larger set of samples and leads to a steady improvement of classifier performance.

3.1.8 Matrix-Structural Learning

The main drawbacks of Haar feature-based face detection are the immense amount of base features and the need for a large number of training samples. These necessities, in conjunction with a boosting training scheme, result in long training times, in the order of days. Several approaches have been presented to reduce training time, see Chapter 2.5.6. Of these, the Matrix-Structural Learning [71] approach has the least limitations. Other approaches discard possibly valuable information, require large amounts of memory, do not evaluate all possible features or may break AdaBoost's proved generalization properties. The Matrix-Structural Learning approach extends the idea of bootstrapping to the entire training set instead of just the negative samples as usually done. The approach is illustrated in Figure 2.

Matrix-Structural Learning (MSL) works by alternating a positive and a negative boot-

strapping phase. In the original Viola and Jones [64] approach, the bootstrapping process is limited to negative samples in between stages. This ensures that later stages have to deal with a more challenging classification task, hence the specialization of later stages on hard samples and their increased computational complexity. Usually, a training set consists of a number of cropped face images and several large background images without any faces. A very large amount of negative samples can be extracted from these background images. Enormous positive training sets might make the Viola and Jones [64] training approach unfeasible because of very long training times. Therefore, MSL introduces an additional bootstrapping phase of positive samples.

Initially, from the original enormous sample set (ESS) with m positive samples a small random positive set of $n < m$ samples is drawn. $C(i, j)$ will denote a trained classifier for stage i that has been trained with a training set from the jth bootstrapping iteration. This initial set $P(1, 1)$ forms the positive training set for the first cascade stage. The initialization of the negative set N_1 is performed similarly to the original approach, by random selection of samples from images known not to contain any face images. By means of the initial training set consisting of $P(1, 1)$ and N_1 a first classifier is trained with a target false alarm rate f_{min} and a target detection rate d_{min}. Iteratively, each newly trained classifier $C(i, j)$ is evaluated on the whole ESS. If the target detection rate d_{min} is missed, a number of positive samples that have been misclassified is added to $P(i, j)$ to form the new positive training set $P(i, j + 1)$ for the next training iteration. If the detection rate target is met, the classifier $C(i, j)$ is the final classifier for stage i, denoted as $C(i, M_i)$. After a stage has been finalized, the negative set is bootstrapped by creating a new negative training set N_{i+1} that consists of samples gathered form misclassifications of $C(i, M_i)$ on the background training images without faces. This alternating process continues until the desired number of stages has been trained.

3.2 Entropy and Mutual Information

Information theory is concerned with the information content or randomness of signals, i.e. the complexity of signals. A key concept of information theory is entropy, a measure for the randomness or uncertainty of a signal. Instead of signals, more generally random variables are quantified. So, given a set of n symbols or a random variable X with n outcomes, i.e. $X = \{x_1, x_2, x_3, \ldots, x_n\}$, and corresponding probabilities for their occurrences

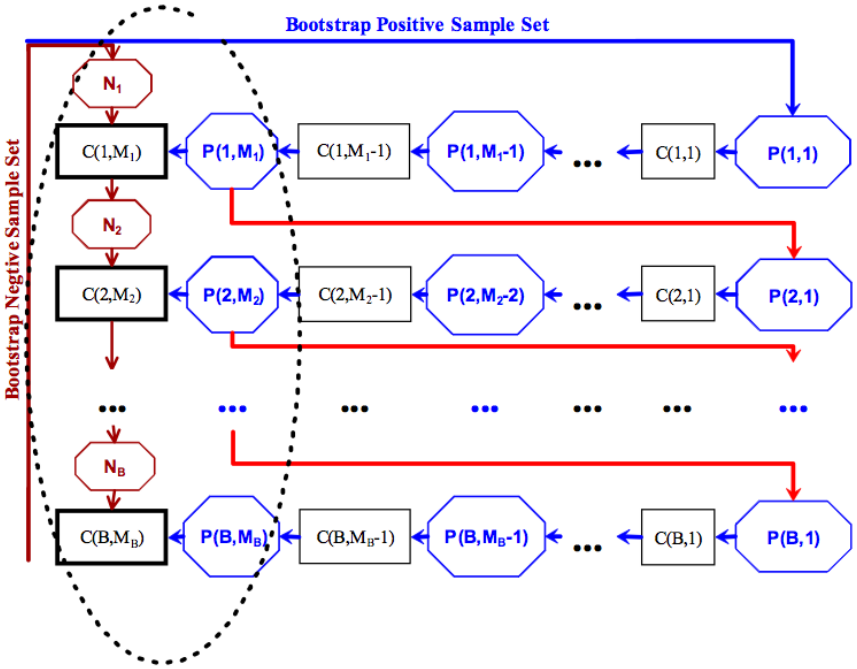

Figure 2: Matrix-Structural Learning overview [71] - alternating positive and negative sample bootstrapping phases.

$\{p_1, p_2, \ldots, p_n\}$, entropy is defined as the expected information content

$$H(X) = -\sum_{i=0}^{n} p_i \cdot log(p_i) = -\sum_{x \in X} p(x) \cdot log\left(p(x)\right).$$

Conditional entropy measures the amount of uncertainty of a random variable X given the value of another random variable Y. Therefore, conditional entropy is a measure for the remaining entropy of X after the value of Y is known. Conditional entropy is defined as

$$\begin{aligned} H(X|Y) &= -\sum_{y \in Y} p(y) \sum_{x \in X} p(x|y) \cdot log\left(p(x|y)\right) \\ &= -\sum_{y \in Y, x \in X} p(x,y) \cdot log\left(\frac{p(x,y)}{p(y)}\right) = H(X,Y) - H(X), \end{aligned}$$

where $H(X,Y)$ is the joint entropy of X and Y. Another important concept in information

theory is mutual information that measures the information shared between two random variables X and Y, i.e. the information of one variable that can be recovered by observing the other. The mutual information $I(X;Y)$ is defined as

$$I(X;Y) = \sum_{x \in X, y \in Y} p(x,y) \cdot log\left(\frac{p(x,y)}{p(x) \cdot p(y)}\right).$$

Conditional mutual information represents the mutual information of two variables conditioned on a third variable. It is defined as

$$I(X;Y|Z) = \sum_{z \in Z, y \in Y, x \in X} p(z) \cdot p(x,y|z) \cdot log\left(\frac{p(x,y|z)}{p(x|z) \cdot p(y|z)}\right).$$

3.3 Support Vector Machines

Support vector machines (SVMs) are maximum margin binary classifiers that solve a classification task using a linear separating hyperplane in a high-dimensional projection-space. This hyperplane is chosen to maximize the distance between positive and negative samples. Real-world problems seldom present linearly separable data, therefore a transformation into a higher-dimensional space is applied before classification with hopes of being able to linearly separate data in the new space. The advantage is that the hyperplane does not need to be projected down into the original space, instead the classification takes place right in the high-dimensional space implicitly. SVMs are conceptually simple yet powerful and the results are interpretable, all good reasons to employ SVMs. Further introduction into SVMs can be found in [42, 54].

3.3.1 Linear classification

Let $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ again denote the training data, consisting of a training vector x_i and a corresponding classification class $y_i \epsilon \{-1, 1\}$. Under the assumption that the provided data can be separated linearly in an n-dimensional space, we can construct an infinite amount of $n-1$ dimensional hyperplanes that correctly separate the training data, because there are no restrictions on placement or orientation of the hyperplane as long as the data is correctly classified. The idea of maximum margin classification is then to choose the hyperplane with the maximum separating margin between the two classes

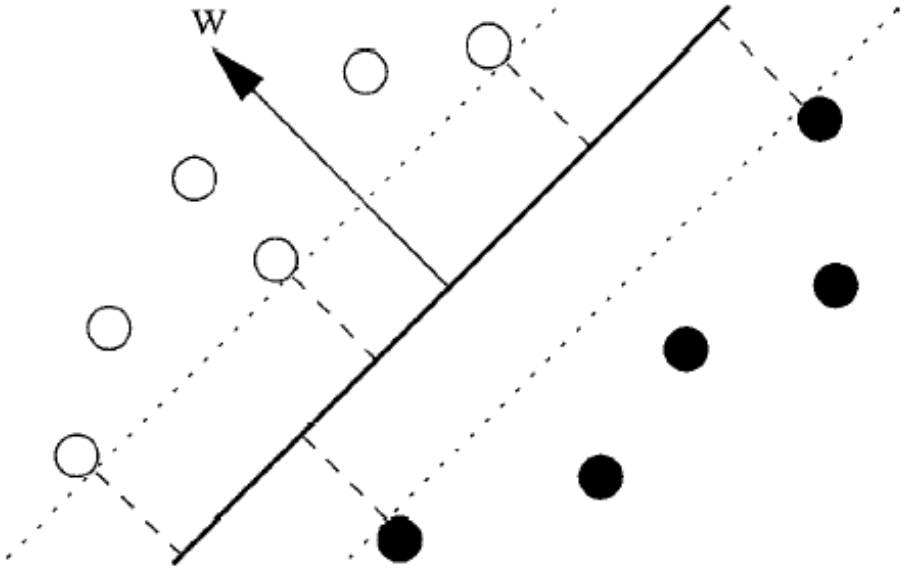

Figure 3: Linear classifier and margins - the margin is proportional to the expected generalization ability. Taken from [42].

because it can be expected that this maximum margin hyperplane is best at generalization, i.e. the margin is proportional to generalization ability of the classifier.

A hyperplane can be described as

$$\{x \epsilon \mathbb{S} : wx + b = 0, (w, b) \epsilon \mathbb{S} \times \mathbb{R}\} \tag{12}$$

and the maximum margin separating hyperplane has to minimize the condition $\min_{i=1\ldots n} |wx_i + b| = 1$ where x_i are the training examples. Consequently, the distance between two samples x_i and x_j relative to the hyperplane can be defined as $\frac{w \cdot (x_i - x_j)}{\|w\|}$. Then the distance between the two classes is $\frac{2}{\|w\|}$. So in order to classify training data correctly the hyperplane can be found by maximizing $\frac{2}{\|w\|}$ or minimizing $\|w\|^2$ under the condition

$$y_i(wx_i + b) \geq 1 \quad \text{for} \quad i = 1 \ldots n, \tag{13}$$

that guarantees that all samples are correctly classified. This minimization can be achieved

using Lagrange multipliers once rewritten into

$$L_p = L(w, b, a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(wx_i + b) - 1)\,. \tag{14}$$

with $\alpha_1, \alpha_2, \ldots, \alpha_n$ being Lagrange multipliers. After solving the optimization problem most α_i are zero because their conditions are fulfilled. Those x_i whose $\alpha_i > 0$ are chosen as support vectors to represent the margins, they are the closest vectors to the hyperplane. Then w can be computed as a linear combination of these α_i: $w = \sum_{i=1}^{n} \alpha_i y_i x_i$.

3.3.2 Soft-margin linear classification

In order to allow a certain number of misclassifications a soft-margin is introduced. The optimization condition is changed to $y_i(wx_i + b) \geq 1 - \xi_i$ for $i = 1 \ldots n, \xi_i \geq 0$, where ξ_i is the sample x_i's distance from the correct margin, it is sometimes also referred to as slack term. Therefore if $\xi_i > 1$ the sample is misclassified, if $0 < \xi_i < 1$ the sample is correctly classified but within the margin (margin error) and if $\xi_i = 0$ the vector lies on the margin. Consequently, the minimization term becomes

$$\min_{w,b,\xi_i} \|w\|^2 + C \left(\sum_{i=1}^{n} \xi_i \right) \tag{15}$$

where C is a weighting parameter that controls the rate of misclassifications. Small values of C maximize the margin, large values of C yield few misclassifications.

So soft-margin classifiers allow a certain number of misclassifications and can therefore better cope with data that is not exactly linearly separable.

3.3.3 Non-linear classification

Given that most real-world data is not linearly separable, the data has to be altered in some way in order for the previous linear maximum-margin classifiers to be useful. The idea is to transform the data into a higher-dimensional space and scatter the data suitably so that it can then be classified using linear separation.

The transformation is usually of the form $\Phi : \mathbb{R}^n \to \mathbb{R}^m, m > n$. But transformations and computations in high-dimensional spaces are usually computationally expensive. Therefore the so-called "kernel-trick" is employed, a kernel function is defined as the dot product of the projection of two vectors - $K(x, y) = \Phi(x) \cdot \Phi(y)$. As the previous equations exclusively use dot products in the high-dimensional space there is no need to explicitly transform the data or to transform the hyperplane. Instead, all equations can be evaluated using kernel functions. Popular kernel functions are
Polynomial:

$$K(x, y) = (x \cdot y + c)^d \tag{16}$$

Radial basis functions:

$$K(x, y) = \exp \frac{-\|x - y\|^2}{2\sigma^2} \tag{17}$$

Sigmoid:

$$K(x, y) = \tanh(\kappa(x \cdot y) + \theta) \tag{18}$$

c, d, σ, κ and θ are parameters and have to be chosen to optimize the classification.

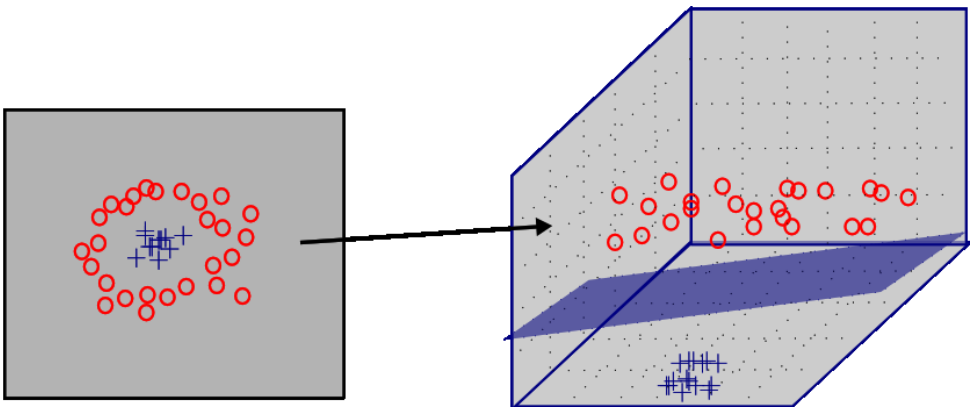

Figure 4: Non-linear classification using kernel function. A kernel function is used to transform data into a higher-dimensional space where the data is linearly separable.

4 Cascade Combination and SVM-based Training Sample Selection

Building real-time face detection systems is still an open research issue. The real-time requirement of many applications requires fast detectors with high precision and low false alarm rates. The work of Viola and Jones [63, 64] proposed a very promising and now wide-spread approach. While many aspects of the approach have been in the focus of research and several improvements were published, training state-of-the-art face detection systems remains a challenge and achieving good results requires large amounts of time for parameter tuning, training set creation and sample selection. Both the classifier training parameters and the training set impact the final classifier's performance.

It is widely believed that training with a large training set improves classification results due to the better coverage of variance in the data. But Chen et al. [12] have shown the importance of selecting suitable training samples for face detection and that shrinking the training set may actually be beneficial. By means of expanding and then down-sampling positive training examples they have created high-performance face detectors. Their algorithm utilizes a sophisticated method based on manifolds. Part of this work attempts to achieve similar results by simpler means. The basic idea to sample selection is not new. Especially in speech recognition and grammar parsing, sample selection played an important role. Speech recognizers and grammar parsers require large quantities of training examples, usually more than can be labeled. Therefore several sample selection approaches were explored in order to pick the most promising training samples to be labeled and used for the next training round. In face detection, Chen et al. have created a large positive training set consisting of more than 100,000 samples overall. The problem here though is not an insufficient amount of samples, but too many samples. As Chen et al. have shown, using optimal training samples can improve the classifier's performance.

Another problem that this work tries to address is to overcome possibly non-optimal selection of training samples by bootstrapping and non-optimal training parameters. Especially in asymmetric detection tasks as face detection, training heavily relies on bootstrapping to select suitable negative examples from the millions of available samples. Usually, according to the bootstrapping algorithm, those samples are added to the training set that are misclassified by the classifier of the previous training iteration. This training scheme, along with the selection of positive training samples, introduces random-

ness into the training process. Two classifiers trained with identical parameters usually exhibit different performance. Additionally, classifiers with different training parameters and training sets may perform better in particular situations. In order to exploit these findings, classifier combination has been explored by Rowley et al. [45]. They use simple arbitration schemes of several neural networks to improve the overall classification performance. Viola and Jones [63] also presented a system consisting of three parallel cascades that were combined by voting, Sung [60] used neural fusion to combine classifiers. The problem with these approaches of combining classifiers is their run-time performance impact. They require to run several classifiers in parallel.

Training good classifiers with several stages is a time consuming task. Training time increases tremendously as the amount of memory needed to cache feature values surpasses the available memory. Then, many calculations have to be repeated for each feature selection iteration. The cache size depends on the number of training samples and the number of features, thus training several cascades in parallel on parts of the training set would be a straight-forward way to minimize training time as that limit is reached. Another possibility to reduce training time by combination stems from the fact that stage training times increase with the number of samples and especially with the number of stages. After a certain point, the negative bootstrapping process is the most time consuming task.

4.1 Cascade Training

In order to evaluate the effect of large amounts of training data on classifier performance, several classifiers had to be trained with many training samples. Training cascade classifiers is tedious and time consuming, since a feeling for the influence of several parameters has to be acquired first. The number of positive and negative samples, each stage target hit rate, the ratio of positive and negative samples and the number of stages have to be determined. In order to train classifiers, especially with a large amount of training samples in excess of 100.000 samples, optimized training algorithms are necessary. Caching has the largest impact on training time without having any side-effects on the final classifier's performance, but it is limited by the amount of available memory, the number of samples and the number of features, i.e. feature set and window size. Once the memory for caching runs out, each feature value for every sample has to be recomputed every feature selection round. Thus, the training performance degrades sharply. In order to overcome

this limit, the Matrix-Structural Learning [71] approach was implemented.

See Chapter 3.1.8 for details of the Matrix-Structural Learning algorithm. The implementation was straight-forward. The training algorithm was extended with another loop to bootstrap the positive sample set in between the negative bootstrap executions until the target detection and false alarm rates are met on the extensive sample set.

4.2 Cascade Combination

Generally, cascade combination attempts to combine the stages of several pre-trained cascades into a new stronger cascade. Cascade combination is supposed to bring the benefits of combining several cascades in parallel, i.e. the benefits achieved by voting or other arbitration techniques, without the need of running cascades in parallel. Also, when combining cascades, several cascades have to be trained, but the amount of necessary samples and stages is lower to achieve a similar performance. Therefore, the overall training time can be less, depending on the extent of caching allowed by available memory, feature set, and training set size.

4.2.1 Optimization Problem Formulation

The cascade combination task can be formulated as an optimization task. The goal of cascade combination can be defined as selecting k optimal stages $\nu(1), \ldots, \nu(k)$ that maximize a cost function from a set of available stages taken from several cascades. Cascade combination is quite similar to feature selection which has been studied widely in the field of machine learning. Fleuret [17] has studied feature selection for face detection in combination with several classifiers. Although cascade combination can be examined in the same framework there are a few crucial differences. First, the choice of the optimal feature or stage is slightly different.

1. In feature selection a good next feature has a large correlation with the class label given all previous features' classifications. But no additional information is gained by adding another feature that highly correlates with the class labels and makes the same errors as previously selected features, i.e. a feature that is redundant. Since positive samples are only accepted by the cascade if they pass all stages, all stages have to correctly classify positives. Thus, the decisions for positive samples need to be coherent and redundant, as opposed to feature selection.

2. During feature selection, the final combination scheme of all selected features is not usually easy to evaluate incrementally. The simple decision making of cascade classifiers, on the other hand, is. Therefore, large lookup tables (caches) may be computed that store each stage's feature sum for every training sample. The classification vector can then be calculated by thresholding these stage feature sums given stage thresholds. The combination of previous stages' decisions is easy and fast, the final decision is positive if all stages accept the sample, negative otherwise.

Due to these two differences, standard feature selection algorithms do not perform quite as well in cascade stage combination. On the other hand, these properties may be exploited to optimize the decisions.

Given a training data set consisting of m samples with label y_i and training sample x_i, i.e. $\{(y_1, x_1), \ldots, (y_m, x_m)\}$, n cascade stages S_n from different classifiers, we can construct a class label vector Y and each stage's classification vector s_n. The class label vector $Y = (y_1, \ldots, y_m)$ contains the binary class labels for all training samples and each stage's classification vector $s_n = (H_n(x_1), \ldots, H_n(x_m))$ denotes stage S_n's strong classifier's output $H_n(x_i)$ for sample x_i. The class labels y_i and the strong classifier outputs $H_n(x_i)$ are binary, i.e. $y_i, H_n(x_i) \in \{0, 1\}$.

Therefore, cascade combination algorithms try to select k final stages $S_{\nu(1)}, \ldots, S_{\nu(k)}$, each iteration the next stage to select is determined by

$$\nu(l) = argamx_n \left\{ cost(Y, s_n, \{s_{\nu(1)}, \ldots, s_{\nu(l-1)}\}) \right\}.$$

The cost function takes the class labels Y, the target stage s_n's classification vector and all previously selected stages' classification vectors as inputs. The following cost functions were evaluated:

- conditional mutual information maximization:

$$cost_{CMIM}(Y, s_n, S) = min_{s_k \in S} I(Y; s_n | s_k),$$

- mutual information maximization:

$$cost_{MIM}(Y, s_n, S) = I(Y^*; s_n^*),$$

- correlation:

$$cost_{correlation}(Y, s_n, S) = r(Y^+, s_n^+).$$

s_n is the classification vector of stage n only,

s_n^* is the classification vector of stage S_n consisting of only negative samples that have passed the selected $n-1$ stages and all positives, Y^* are the corresponding labels,

s_n^+, Y^+ are the same as s_n^*, Y^* but only positives that passed all previous stages are included.

The corresponding cost functions will be explained in the following Chapters.

4.2.2 Cascade Combination with Threshold Optimization

Cascade stages are trained to be used in conjunction with certain other stages. Since each stage's negative training data is bootstrapped by using the previously trained stages, its stage threshold has been chosen on training data that does not take previously discarded samples into account. Additionally, several researchers have shown that the independent, ad-hoc choice of stage thresholds as proposed by Viola and Jones [64] is not optimal. Therefore, stage thresholds have to be re-adjusted when different cascades' stages are combined.

The optimization of stage thresholds is done using the same cost functions as outlined in the previous section and described in more detail later. Instead of trying to find the optimal next stage, here the threshold t_i of the current stage S_i is optimized. Let s_n^t denote the classification vector of stage S_n after its threshold has been modified to t. The best threshold for the current stage maximizes the given cost function $cost()$, i.e.

$$t_i = argamx_t \left\{ cost(Y, s_n^t, \{s_{\nu(1)}, \ldots, s_{\nu(l-1)}\}) \right\}.$$

The range of possible values of t is restricted by the possible range of feature values $H_n(x_i)$ and extreme cases need not be evaluated. We performed a linear search to find the optimal threshold for $t \in [-2.5, +2.5]$.

4.2.3 Cascade Combination with Conditional Mutual Information Maximization

Conditional mutual information maximization (CMIM) has been used as a feature selection scheme in machine learning (ML) before. Fleuret [17] provides a good introduction. In the CMIM scheme, in order to combine several stages of different cascades, we build a new cascade from existing stages by iteratively choosing the next best stage to be removed from the pool and added to the final cascade. We try to select the next stage s_m which shares the least information with all previously selected stages, but correlates well with the true class labels Y, where $m-1$ stages have already been selected. As stated before, a strong correlation with the class label is not useful if most stages correctly classify the majority of samples, i.e. they are each individually "good", but make the same, redundant errors.

Ideally, we would maximize the term

$$max\left\{I\left(Y|s_{\nu(1)}, s_{\nu(2)}, \ldots, s_{\nu(k)}\right)\right\} \tag{19}$$

that denotes the joint mutual information between the class label Y and all stages to select. Optimizing that term, specifically estimating the required probabilities, is computationally infeasible. Maximizing the mutual information between each selected stage's classification vector and the class labels does not necessarily yield optimal results since the chosen stages are examined completely independently, thus they may be redundant and the final combination may not be ideal. A compromise is the use of CMIM.

Conditional mutual information is an estimate for the information that X_1 and X_2 share when X_3 is known - $I(X_1, X_2|X_3) = H(X_1|X_3) - H(X_1|X_3, X_2)$. So I is large if X_3 carries information about X_1 that was not already revealed by X_2, if X_2 and X_3 contain the same information I is zero. In the cascade combination case, therefore, the optimal combination of stages is the one that maximizes the conditional mutual information of all selected stages, i.e. $max(I(Y|X_{s(1)}, X_{s(2)}\ldots., X_{s(n)}))$. Since that term's calculation is not feasible, an approximation suggested by Fleuret [17] is to select each feature, or stage in

our case, so that

$$argmax_n \left\{ min_{k \leq n} I(Y|s_n, s_{\nu(k)}) \right\},$$

i.e. select the stage that has the maximum conditional mutual information matched with the worst case previously selected stage.

The first stage is simply selected by maximizing the mutual information between the class labels and the stages classification, i.e.

$$argmax_n \left\{ I(Y|s_n) \right\}. \tag{20}$$

While CMIM was shown to perform well for feature selection, the use of conditional mutual information for cascade combination has limitations. The main problem is the different decision process for positive samples when combining cascade stages. In feature selection, different features ideally produce different classification results that are then ultimately consistent with the class labels when combined. In cascade combination, the non-redundancy of stages is only desirable for negative samples, whereas all classifiers have to make the same decisions on positive samples in order to accept them.

4.2.4 Cascade Combination with Mutual Information Maximization

While CMIM has been shown to outperform mutual information maximization (MIM) for feature selection tasks [17], the different prerequisites for cascade combination make it less useful than MIM for cascade combination. Since the combination scheme of classifications simple in cascade classifiers, the classification vector of all previously selected stages can be obtained efficiently by caching stage classifications of samples. Therefore, instead of optimizing the ideal global mutual information as in Equation 19, we can greedily approximate its optimization by continually maximizing the mutual information between the class label and the joint classification vector of all previously selected stages and the current stage.

The optimization term then becomes

$$argmax_n \left\{ I(Y^*; s_n^*) \right\}.$$

Y^* and s_n^* require some more explanation. Since we cannot optimize Equation 19, an

approximation is necessary. The joint mutual information of k arbitrary stages that need to be selected and the class label, as described in Equation 19, cannot be feasibly optimized, since it requires the estimation of 2^{k+1} probabilities. Hence, the ideal choice of the next stage is approximated by choosing the next stage, so that the classification result of all previously selected stages and the candidate stage has maximum mutual information with the class label. Samples that were rejected by previous stages should not have any further influence on the current decision for the next candidate, since even a correct classification would not make a difference in the overall classification. Positive samples on the other hand are important and ideally stages should not discard any positive samples, so we require the next stage to comply with all positive labels, not just the remaining. Therefore, the terms Y^* and s_n^* denote the class labels of all remaining samples that have passed all previously selected stages and the corresponding classification vector of stage S_n. The calculation of $Y^*$and s_n^* is straight-forward, Y^* can be attained by discarding labels for rejected samples and s_n^* can be generated with the help of a list of remaining samples to classify and evaluation of stage S_n on those samples to yield the classification vector. Since these operations have to be repeated each iteration, caching the feature values of all stages is very helpful. The first stage is selected according to Equation 20.

4.2.5 Cascade Combination with Correlation Maximization

So far the use of mutual information and conditional mutual information for selection of stages in cascade combination has been explored. They have been used to measure a correlation between stages' classification vectors, i.e. classification results, and the true class labels. Therefore, the formulation of the same problem with a simple correlation metric is obvious. Instead of the mutual information between classification and class label, the linear correlation is optimized, another widely used feature selection strategy. The difference is that linear correlation tries to minimize the mean-square error of a line fit to the data and class labels, thus considering the linear relationship between classification vector and labels of individual samples. Linear correlation is expected to perform less well but is useful for comparison. If it turns out to perform better than CMIM or MIM, there is strong evidence for further possible optimizations.

The optimization term is

$$argmax_n \left\{ r(Y^*, s_n^*) \right\}.$$

$Y^*$and s_n^* are defined as outlined in the previous subsection.

4.2.6 Sample Extraction and Subsampling

Since classifier combination is unlike the usual classifier training and more like an optimization of pre-trained cascades, the samples used for the cascade combination algorithms are different from those used for training. One reason to use a different sample set is to avoid overfitting of the classifier. Thus, the training sets for training and combination are completely independent. Because cascade classifiers are applied in a sliding window-fashion, i.e. moving a window of different scales over the input image and testing for positive detections, there are multiple acceptable windows for a single labeled face. Some approaches, like Multiple-Instance-Pruning by Zhang and Viola [77], take advantage of this. This work indirectly acknowledges this by extracting numerous samples from manually labeled full input images, instead of using cropped faces, and maintaining multiple positive samples for a single labeled face, i.e. slightly offset positions and slightly different scales - see Chapter 5.1.3 for details on the face matching metric.

One problem that occurs is that the imbalance of positive and negative samples may bias the cost function, e.g. the prior probability of class labels influences the entropy of the different classes. In order to mitigate the imbalance, the negative samples have been subsampled. Subsampling in this case means that only every k-th extracted negative sample from the input image is added to the combination data set.

4.3 Training Set Selection

As outlined in the beginning of Chapter 4 resampling promises to improve classification performance while reducing cascade training time as long as the sample selection itself does not negate the gain. Chen et al. [11] have explored resampling. They have employed a data-centric approach by modeling the distribution of samples using manifolds, which are lower-dimensional spaces that preserve the local neighborhood of an initial space. Using manifolds they resample the training set to have the same density at all locations in the manifold, underpopulated regions are filled with interpolated samples. This approach can be described as data-centric because its goal is a uniform coverage of the positive sample space. The approach chosen by us is more classifier-centric. Support vector machines have recently become one of the most-widely used classifiers. Their advantage lies in their conceptional simplicity and solid theoretic foundation while achieving very good classification rates. Another advantage is that their internal state, i.e. the choice of

the separating hyperplane, is interpretable as opposed to some other classifiers. Support vector machines construct a hyperplane that separates positive and negative samples. A kernel function allows to project samples into a higher-dimensional space so that non-linear decision functions in the input space are possible. The projections of the closest samples to the decision hyperplane are called support vectors. The separating hyperplane is chosen so that all vectors of each class lie on either side of the hyperplane and the support vectors have a distance of 1 from the hyperplane. This distance from hyperplane is also called the margin. Therefore, samples further away from the decision hyperplane (higher margin) can be thought of as being easier to classify than those closer to it. The hyperplane margin can then be used as an indicator for the difficulty or importance of samples for the classification.

Unfortunately, support vector machines do not perform as well as classifiers of boosted ensembles for face recognition. Their main drawback is the evaluation speed. Thus the overall approach is to train an SVM-based face detector, use the distance-from-hyperplane of samples to quantify each samples difficulty and then train a cascaded classifier based on the resulting ranking of samples.

4.3.1 SVM-based Training Set Selection

Each stage of a cascade classifier architecture has the purpose of rejecting a certain subset of samples as negatives. The bootstrapping process used to gather negative training samples forces later stages continually focus more and more on the decision surface between positive and negative samples. SVM-based training set selection attempts to construct a positive set so that the classifier is focusing on the decision hyperplane on the positive side as well [43].

SVM-based Face Detector. In order to resample the training set with the help of support vector machines, a suitable face detector has to be built first. There have been several approaches for face detection with SVMs. Heisele et al. [24] built a component-based face detection system, where local feature-detectors based on linear SVMs are trained with artificially generated samples and the components' relative positions are verified with another SVM. Earlier approaches used SVMs on raw input data [43].

The SVM-based face detector used for resampling is based on raw image intensity inputs, no feature extraction is applied. The training set consists of a number of positive and

negative training samples whose raw intensity values are vectorized and used for training. The positive samples are manually extracted and cropped face images, the same as used for cascade classifier training. Since an almost arbitrarily large amount of negative samples can be extracted from the training data, a bootstrapping approach such as proposed by Sung and Poggio [61] and used by Rowley et al. [45] was also used here. First, an initial set is randomly selected from large background images that are verified not to contain any faces. Then, an SVM is trained with the given positive and negative training samples. Next, misclassified negative samples are added to the training set. Once an upper bound on sample set size is reached, bootstrapped samples are randomly replaced, in order to avoid too strong imbalances.

Support vector machines have several parameters - the kernel projection and the corresponding kernel parameters. In order to find the best parameters, a simple grid search was used without bootstrapping the negative samples. A grid search simply tries all pairwise combination of parameters, in this case RBF ($\Phi(u,v) = exp\left(-\gamma\|u-v\|^2\right)$) or polynomial kernel ($\Phi(u,v) = \left(\gamma \cdot u^T \cdot v\right)^d$), and the kernel parameters γ for RBF, the exponent $d \in \{2,3\}$ and γ for polynomial kernels and the misclassification cost parameter C that allows for a number of misclassifications in order to avoid overfitting. A polynomial kernel with $d = 2$, $C = 10, \gamma = 1$ worked best according to the grid-search. All components of the input feature vectors were mapped from image grey-value intensities $[0, 255]$ to $[0, 1]$.

Lighting normalization has been widely employed in face detection and face recognition. Therefore, normalization approaches were explored, namely histogram equalization. Histogram equalization tries to increase the global contrast of images by spreading the image intensities in the histogram. It is computationally inexpensive.

Support vector machines trained with histogram equalization performed better than those trained on raw input data, the detection rate was higher for a given false alarm rate and, after the same number of iterations, the SVM trained on normalized data achieved a higher detection rate.

SVM-based Training Set Selection. Once the support vector machine is bootstrapped and trained sufficiently, samples have to be selected for the following cascade classifier training. The presented approach tries to focus the attention of classifier training on the decision boundary between positive and negative samples. Boosted classifiers, among others, tend to overfit training data when not carefully chosen due to their property

of concentrating on misclassified data. Using only samples along the decision boundary, a single negative outlier within a positive region would most likely split that region and enforce a negative region within the region of positives. Therefore, while concentrating on the decision boundary, sampling the whole population is helpful to reduce the effect of outliers.

Two strategies for selecting samples were examined. Both try to force the trained classifier to focus on the decision boundary. The first strategy simply takes the l hardest samples, where is $l < n$ and n is the number of overall samples. The second strategy tries to focus on hard examples without solely selecting those samples. When examining the resulting SVM-based ranking of training samples, about 0.1% were misclassified as negative (distance-from-hyperplane $d_{SVM} < -1$), 23.5% were not clearly negative or positive ($d_{SVM} \in [-1, 1]$) and 76.4% were correctly classified as faces, d_{SVM} was in the range of $[-3.2, 5.2]$, $d_{SVM} < 0$ for 0.6% and $d_{SVM} \geq 0$ for 99.4%. Therefore, the second strategy for selecting $l < n$ samples simply divides the range of distances into l segments and selects a sample closest to the segment boundary for each segment. This approach covers the whole range of distances, not just the hard samples. As most samples have positive distances and only a limited amount of segments will cover positive d_{SVM}, relatively more samples with $d_{SVM} < 0$ will be selected compared to the original distribution of samples, thus focusing on hard training samples. In other words, more segments fall onto the fewer negatively misclassified samples, therefore increasing their probability of being included in the final set.

Cascade Retraining. Once the training set was resampled, the cascade was trained in a straight-forward manner using Matrix-Structural Learning to speed up the learning process. The resulting cascaded classifier has the same properties as any other trained classifier, since the training process remains exactly the same. The only difference is the choice of training samples that has had more attention.

5 Experiments

This chapter describes the experiments that were conducted in order to evaluate the effectiveness of the proposed algorithms.

5.1 Experimental setup

This section describes the basic setup of the experiments, i.e. the data set, how face bounding boxes for matching faces are created and how a detection is determined to be correct.

5.1.1 Dataset

The data set used to evaluate the classifiers that were trained with the proposed approaches is the CMU and MIT database (CMU+MIT). The dataset consists of images collected at CMU, Pittsburgh, PA, and MIT, Boston, MA. These images were initially used to evaluate the neural network-based face detector developed by Rowley et al. [45]. Since then, the dataset has become the most widely used benchmark in face detection. Most groups report results on the CMU+MIT dataset, thus the same set was used to allow for comparison with related work. Usually, results on 507 labeled faces are reported, while the ground-truth labels obtained from the CMU neural network-face detection page contain 512 faces. Potential extreme side-views or non-human faces may have been removed in other work but specific information could not be found. The dataset consists of 117 images of medium to high resolution, containing one to several faces. The images are constituted of mixed subjects, from soccer teams, to theater pamphlets, to photos of graduation classes. The provided CMU+MIT ground-truth labels mark positions of the eyes, nose, as well as left and right mouth corners and mouth center.

5.1.2 Bounding Boxes

Our face classifiers return face bounding boxes, (x, y, w, h) consisting of the location (x, y) and width and height (w, h) of the face detection. Since the ground-truth labels do not explicitly provide face bounding boxes, these have to be generated from the data labels. Face labels are provided as (x, y) locations of eyes, nose and mouth left and right corner

and the mouth center. These locations are denoted as *leye*, *reye*, *nose*, *lmouth*, *rmouth* and *cmouth* with added subscripts x,y for the coordinate. A face bounding box is obtained from the labels by taking

$$\begin{aligned} eyedist_x &= reye_x - leye_x, \\ facecenter_x &= reye_x + \frac{1}{2} eyedist_x, \\ moutheye_y &= max(reye_y, leye_y) - cmouth_y, \\ x &= facecenter_x - eyedist_x, \\ w &= 2 \cdot eyedist_x, \\ y &= max(reye_y, leye_y) - \frac{1}{2} moutheye_y, \\ h &= 2 \cdot moutheye_y. \end{aligned}$$

5.1.3 Face Matching Metric

The definition of a correct or false detection is not trivial and most papers seldom report exact face matching metrics. While detection rate by itself is not conclusive, since efficiency and speed are also important factors, most authors solely report detection rates on the CMU+MIT dataset. The detection rate d is the fraction of positive samples that were correctly classified as face samples and the false acceptance rate f is the number of negative samples that were also classified as faces. The absolute number of false acceptances is denoted as F. Given n test samples, n_{pos} positive and n_{neg} negative of which m_{pos} were correctly detected and m_{neg}were falsely accepted, detection and false acceptance rates are defined as

$$d = \frac{m_{pos}}{n_{pos}},$$

$$f = \frac{m_{neg}}{n_{neg}}.$$

Once face bounding boxes from face labels have been constructed, the problem of defining a correct detection still remains. According to Lienhart et al. [35] a face is considered as

correctly detected, if

- the distance of centers of bounding boxes of a detection and a labeled face is less than 30% of the actual face width and
- the size difference between the two bounding boxes, in our case both width and height, is less than ±50% of the actual face width.

The same detection metric was used in this work.

5.2 Receiver Operating Characteristic Curves

The receiver operating characteristics curve (ROC curve) is defined as a plot of detection rate d vs. false alarm rate f. It provides a way of comparing binary classifiers, as well as setting desired operating points where, for example, only a certain maximum false acceptance rate is acceptable.

Usually, ROC curves are generated by modifying the classification threshold of the binary classifier. Since cascades of boosted ensembles consist of several stages with an own threshold each, this simple scheme cannot be applied straight-forward.

Since the work of Rowley et al. [45] heuristic post-processing steps have been widely used. Neighboring detections are grouped and a face detection is only reported if at least n neighbors are in a group. This method provides an effective way to reduce f with a slight impact on d. Another way to influence the detection rate d indirectly is modifying the scale s at which the sliding window is scaled up, allowing for a finer-scaled search of faces. Additionally, the number of stages used by the classifier can be reduced and thus d and f rise. Varying thresholds is more complicated, since m stages interact and each stage $i < m$ has an individual threshold t_i. In order to explore all possibilities, an exponentially growing number of thresholds would have to be evaluated. Therefore, we vary the stage thresholds t_i individually in k steps, within a range $[t_i - \Delta t, t_i + \Delta t]$. In our experiments, $\Delta t = 1.0, k = 4, n \in \{0, \ldots, 8\}, s \in \{1.01, 1.1\}$ and the number of stages was not reduced. We chose small scales s since larger scales barely contribute to the interesting region of the ROC curve with values of $F < 50$. Since the cascade architecture does not allow to vary the thresholds in a straight-forward way, several of the aforementioned approaches have to be combined to estimate as many operating points as

possible. Overall, 96 parameters are evaluated per stage per image, resulting in several thousand operating points. Nonetheless, many of these operating points lie at high false alarm rates. Therefore, the following plots may not be perfectly convex because the combination of the aforementioned parameters could not capture an optimal operating point at a certain false alarm rate.

In order to smooth the plots, the following procedure was applied and is demonstrated in Figure 5. Each point of the ROC curve (x_i, y_i) has to have a y value that is greater than the value of a point on the line connecting the previous point on the curve with any later point, i.e. it has to satisfy

$$y_i \geq (x_i - x_{i-1}) \cdot \frac{(y_j - y_{i-1})}{(x_j - x_{i-1})}, \ \forall j > i, i > 1.$$

Thus, the smoothing operation removes data points whose value is below the connecting line of the previous point with any following point. The reasoning is that these points can be removed because the ROC generation algorithm only produced sub-optimal results. A better data point should be producible with an optimal combination of the above parameters and thresholds. The generation of these optimal points is impractical due to the immense computational requirements necessary to find these points.

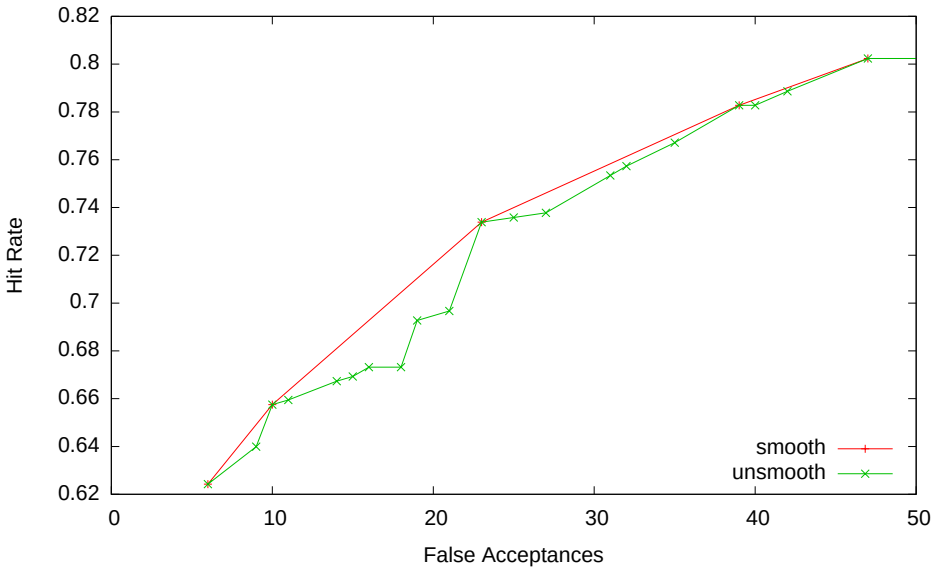

Figure 5: ROC curve smoothing. The originally generated ROC curve is compared against the smoothed curve. Data points with smaller values than a line connecting any previous and following points are removed.

5.3 Base Classifiers

5.3.1 Training Set

Several base classifiers have been trained to be used for classifier combination and as a baseline to compare against for SVM-based resampling method. The global data set used for training of these classifiers, unless otherwise noted, consists of 9000 positive face images and 339 large background images. Negative samples are sampled via bootstrapping from the background images that do not contain faces. Our training set is a subset of the 100,000 sample positive set created by Chen et al. [12].

The 9000 positive samples have been randomly partitioned into subsets consisting of 4500 (two sets) and 3000 samples (three sets), all subsets are disjunctive. These sets were used to train the following base classifiers.

5.3.2 Base Classifiers

The following base classifiers were trained from the aforementioned training sets by using Matrix-Structural Learning. The number of negative samples is equal to the number of positive samples. Therefore, one set with 9000 negative and positive samples, two sets with 4500 samples each and three sets with 3000 used samples each were available. The classifiers were trained to produce 20 stages, width and height of the resulting classifier were 20×20, a basic feature set with 41910 features overall was used and the minimal hit rate per stage was $d_{min} = 0.999$. No tree-based weak classifiers were used, only simple tree stump classifiers which correspond to thresholded single features.

Figure 6, 7 and 8 show the ROC curves of classification results on the CMU+MIT dataset of cascade classifiers trained on three different training set sizes. These classifiers were trained with 3000, 4500 and 9000 positive and negative samples. If multiple sets of training data were available, multiple cascades were trained. All resulting classifiers for all available training data sets are shown below. Even though the training parameters are the same, the cascades exhibit different performances. This is most likely due to the different positive training sets and the bootstrapping procedure used for training. Since the bootstrapping process starts with an initial random set of negative samples, these classifiers were also trained with different negative sets. All training sets were of the same size. These results indicate that a careful choice of the training has an influence on the performance of the classifier.

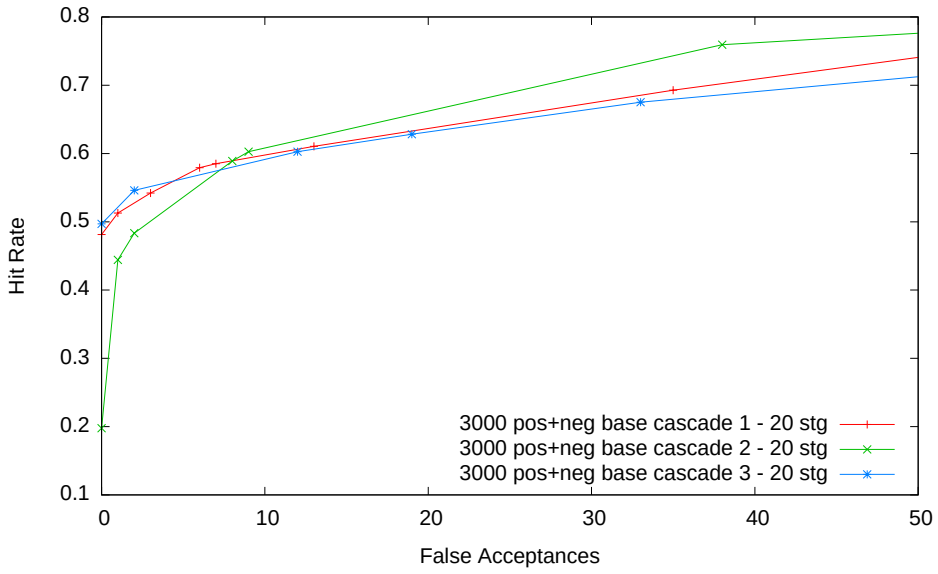

Figure 6: ROC curve for cascades trained on 3000 positive and negative samples.

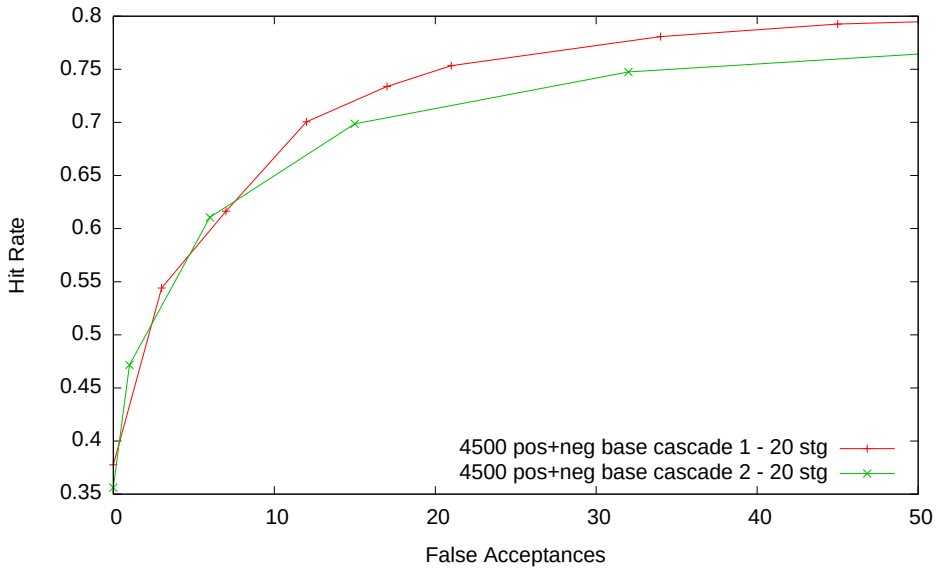

Figure 7: ROC curve for cascades trained on 4500 positive and negative samples.

Figure 8 compares all cascades that were obtained by training on the original training set and all derived subsets. The cascades were trained with the same parameters as described above. It is interesting to note that while training with 4500 samples yields an improvement over 3000 samples, training with full 9000 samples results in no performance improvement. Additionally, 3000 samples base cascade 2 seems to perform almost as well

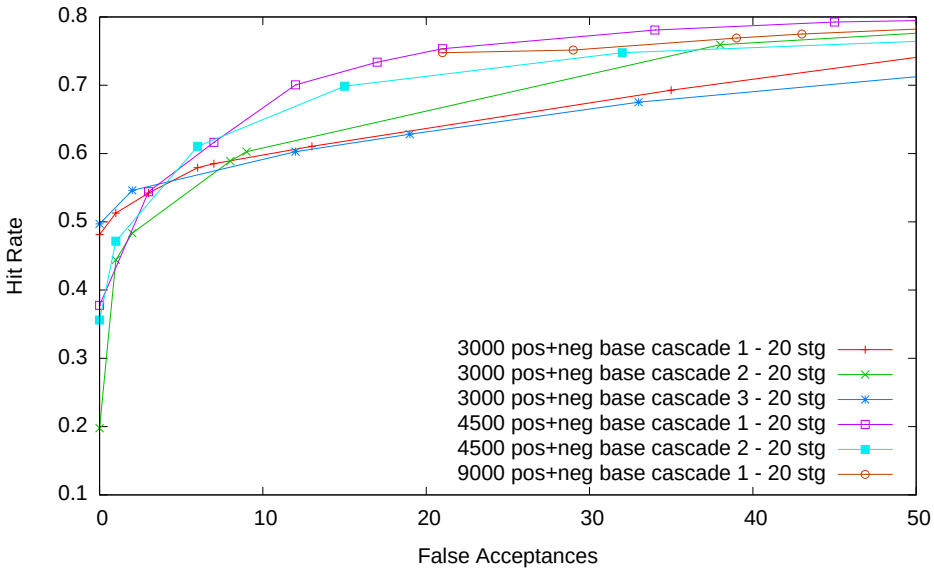

Figure 8: ROC curve for cascades trained on 3000, 4500 and 9000 positive and negative samples.

as the 4500 sample cascades. There is also a substantial difference in cascade performances trained with the same parameters and amount of samples. Out of the cascade classifiers trained with 3000 samples, the detection rate d of cascade 2 is 5% higher than the rivalling cascades at higher false alarm rates, 4500 cascade 2 performs about 3% better than cascade 1. These considerations were, along with the work of Chen et al. [12], the motivations to attempt to resample the positive training set.

The improvement of classification results with additional stages has been shown in Figure 9. Two of the three 25-stage cascades show an improvement over the 20-stage cascades. As the figure illustrates, adding more cascade stages raises the ROC curve as is to be expected since adding stages mostly means rejecting further negative samples while retaining a large majority of positives.

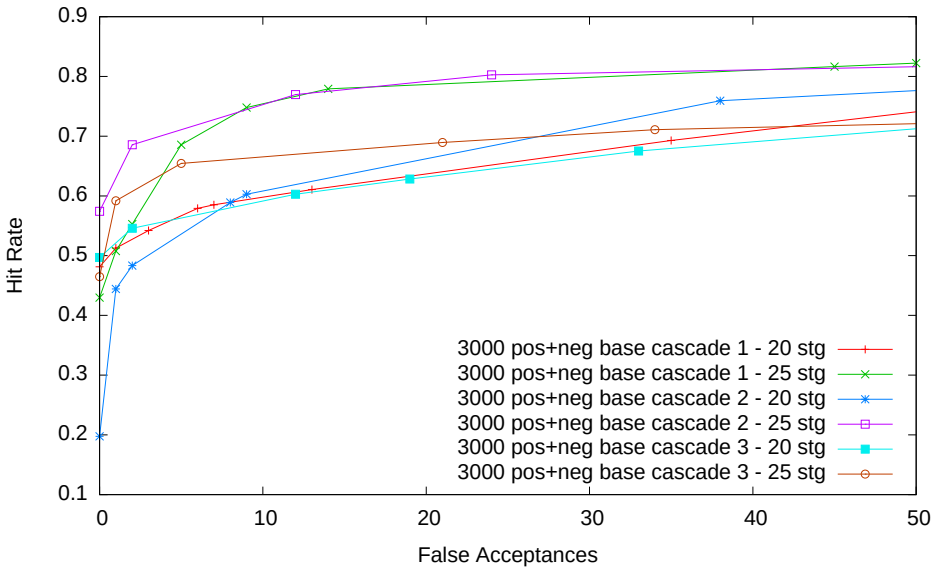

Figure 9: Comparison of ROC curves of cascades with 20 and 25 stages.

5.3.3 Training Time

The need to train several cascades led to an interesting observation. Figure 10 shows that after training about 20 stages, the bootstrapping time rises almost exponentially, whereas training time itself (feature selection and threshold estimation) rises nearly linearly. The values fluctuate considerably in Figure 10, in order to clarify the effect, the fluctuation was reduced in Figure 11 and the linear increase is more visible. As can be seen from the figures, overall training time is mostly dominated by the bootstrapping process and not the stage training itself after about 20 stages. Two possible bottlenecks for the negative bootstrapping have been examined, network-stored image files with slow access and possibly inefficient selection of search windows within background images by frequent changes between several background images. Slow file access via a networked file system has been avoided by moving the files to a local hard disk volume. The bootstrapping time only slightly decreased, loading the files is therefore not a critical factor. Then, the code that samples negative examples from image files was checked not to perform too many unnecessary disk accesses. A new file is loaded after about 20,000 patches have been samples which should not warrant a high overhead. Hence, the reason for the high bootstrapping time is the need to sample many random image patches before a suitable negative sample can be added to the bootstrapped training set. Many image patches have to be extracted and checked against the currently trained cascade stages, only misclassified samples are added. One explanation for the disproportional rise would be the size of the negative background images set. These 339 full-size images with diverse, cluttered background may not contain enough suitable samples. So finding misclassified patches becomes very time-consuming with random sampling. Therefore, a larger set may solve the issue.

Figure 10 shows that the training time does not increase linearly but fluctuates. Therefore, the number of selected features was also plotted to verify that the classifier is constantly growing in complexity. The fluctuations in training time are due to the MSL-style learning, where positive samples are bootstrapped. Feature reuse was not implemented, therefore after every bootstrapping run all features have to be reselected. In some cases, a single positive bootstrap run was enough to fulfill the stage goal on the whole set whereas the next stage required, e.g., 4 bootstrap runs, thus taking about four times as long. Figure 11 shows training time, bootstrapping time and number of features as well for a cascade trained with 1500 (instead of 500 before) of 3000 positive samples already in the initial

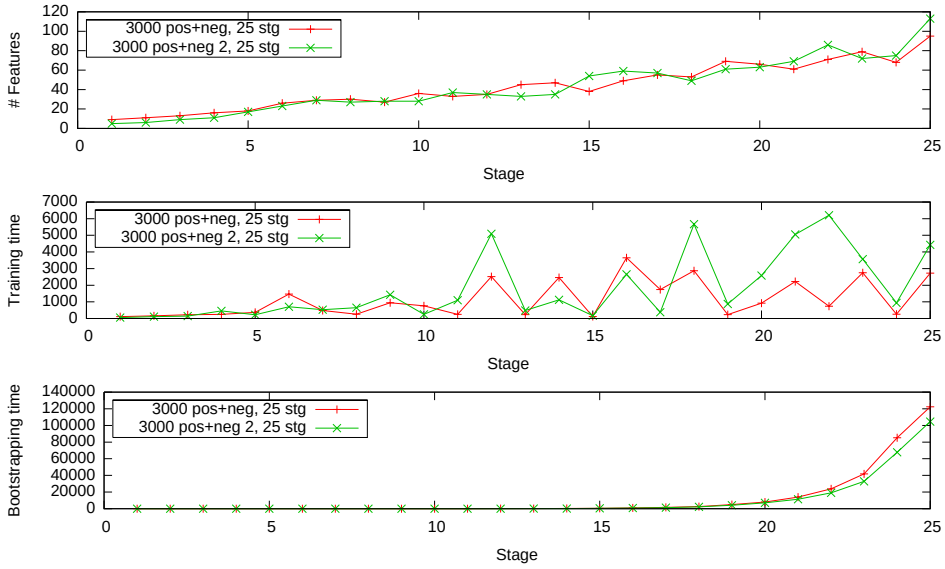

Figure 10: Training time, bootstrapping time and number of features for 25 stage cascade training. The bootstrapping time increases exponentially after 20 stages were trained. MSL-style learning causes the fluctuations in training time.

bootstrap set, i.e. reduced MSL bootstrapping. The training time curve is smoother and the exponential increase is still visible towards the end, but it is more pronounced in Figure 10.

5.3.4 Motivation for Training Set Selection

An optimal training set can have a great influence on the classifiers performance. In order to examine the effect of different training sets, the 9000 sample base set was randomly divided into three 3000 sample sets as described in Section 5.3.1. Figure 12 demonstrates the resulting classifiers' performances trained with these different sets. The classifiers were all trained with the same training parameters. Figure 12 clearly shows that while cascade 1 and 2 roughly have the same performance, cascade 3 underperforms in comparison. Training set selection is supposed to make sure that cascades trained with an optimized set will perform better.

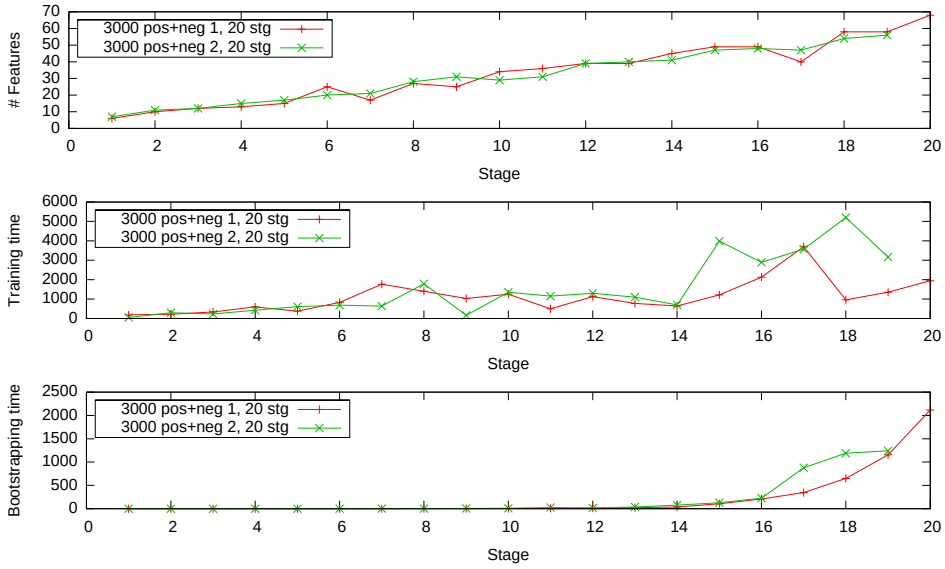

Figure 11: Training time, bootstrapping time and number of features for a 20 stage cascade with minimal MSL bootstrapping. The exponential increase in bootstrapping time begins around stage 18. The variations in training time are caused by the MSL bootstrapping of positive samples.

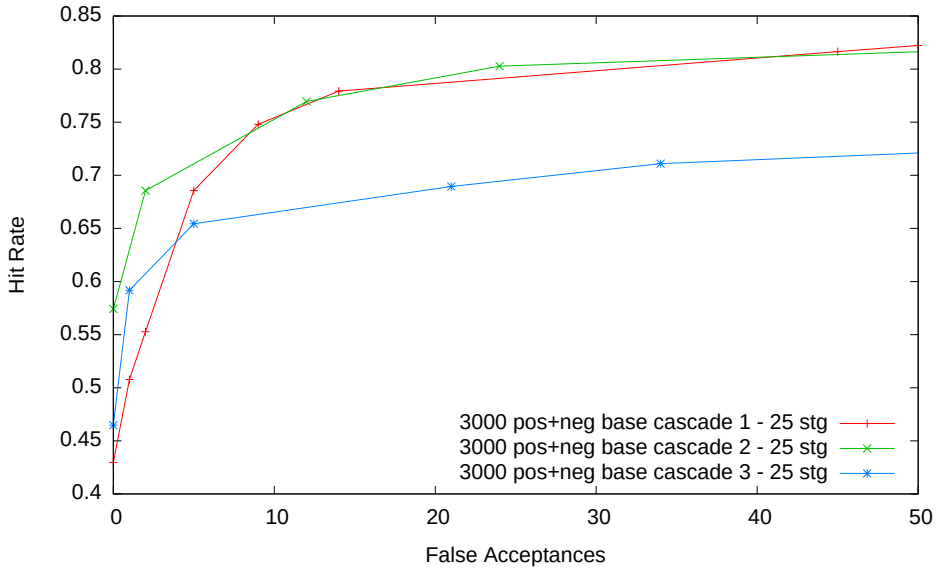

Figure 12: Comparison of three cascades trained with identical parameters but different training sets of the same size. The performance of cascade 3 is far below that of cascade 1 and 2 and must be due to the choice of training set since all other parameters were the same.

5.3.5 Motivation for Cascade Combination

Figure 13 demonstrates another advantage of cascade combination. Cascade combination can be used to improve the results of low-performance base cascades and therefore generate cascade classifiers that are comparable to classifiers that were trained more extensively. The extensive classifiers were trained with more final stages and higher stage target detection rate $d_{min} = 0.9999$, as opposed to 20 stages and $d_{min} = 0.999$ for the base cascades. Figure 13 compares the performance of a 20-stage combined cascade classifier constructed from three 20-stage base cascades with an extensively trained 25-stage cascade classifier. As was shown in Section 5.3.3, training time increases sharply after twenty stages. Section 5.3.2, on the other hand, shows that substantial performance gains are possibly with extended training.

Figure 13 shows the resulting classifiers' performances. The plot displays the best 20-stage base cascade, the best extensively trained 25-stage cascade and the resulting combined cascade classifier. In order to avoid clutter, only the best classifiers for 20 and 25 stages are shown, although three of each were trained - see Section 5.3.2 for details. The results indicate that cascade combination was able to produce a cascade classifier that substantially outperforms the base classifiers, as can be seen in Section 5.4.5. More importantly, the combined cascades can almost match the performance of the 25-stage base cascade that took considerably longer to train.

Overall, the training time for the depicted extensively trained, high-performance cascade was 267 hours. The training time for the depicted low-performance base cascade was only 6 hours and 58 minutes. This striking difference in training times is due to an effect shown in Section 5.3.3, the exponential increase in bootstrapping time after 20 stages. Overall, the three 20-stage base cascades were trained in about 22 hours, the combination requires another 30 minutes to an hour, depending on the amount of extracted samples. Therefore, the total training time for the combined cascades was about one day as opposed to 11 days for the extensively trained cascade. That is a substantial improvement and should outweigh the fact that the 25-stage cascade performs slightly better, two to five percent increased detection rate for a given false alarm rate.

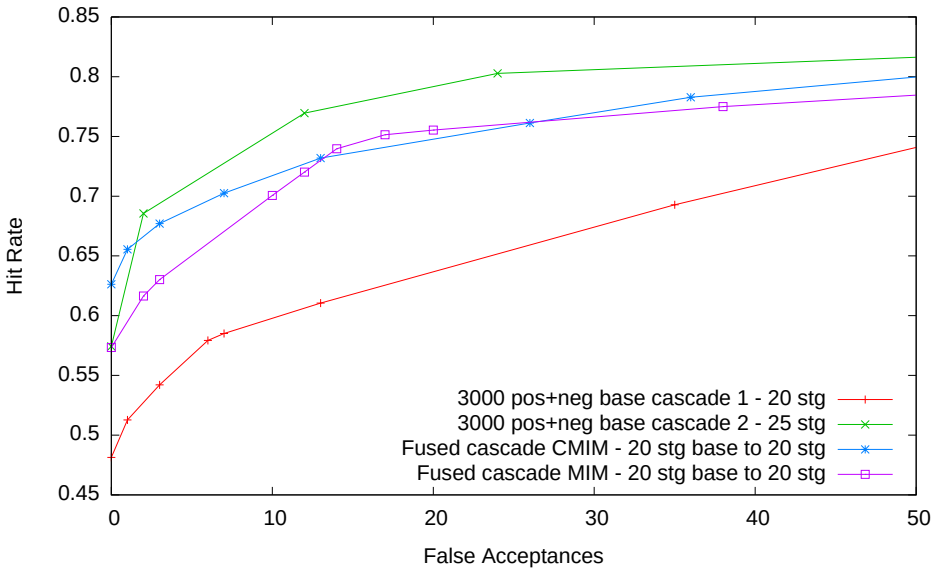

Figure 13: Cascade combination of three 20-stage, low-performance cascades trained with 3000 positive and negative samples vs. 25-stage, high-performance cascades trained on the same training sets. The resulting combined classifer's performance approaches the performance of the high-performance cascades. Training time for all three 20-stages base cascades and the combination process is less than a day, whereas the 25-stage cascades were trained for more than 11 days each. Cascade combination improves the base cascade classifiers significantly while keeping training time low. Only the best base and 25-stage cascade classifiers are shown for comparison.

5.3.6 Large Training Sets with Matrix-Structural Learning

Figure 14 compares three cascades with different training set sizes trained to twenty stages. Three cascades were trained with the same training parameters but different training set sizes. Two 20-stage cascades were trained with small training sets of 1,000 and 2,000 samples whereas a third 20-stage cascade was trained with a large 40,000 sample set. The 40,000 sample set classifier clearly outperforms the other classifiers as was to be expected. The increase in performance is at the cost of training time. Training with 40,000 samples requires a cache size of about 8GB of RAM and is therefore not always practical. If the amount of available memory is insufficient to cache feature values for all samples, the training time will increase strongly. Almost similar performance can be achieved with a 25-stage classifier trained with only 3,000 training samples.

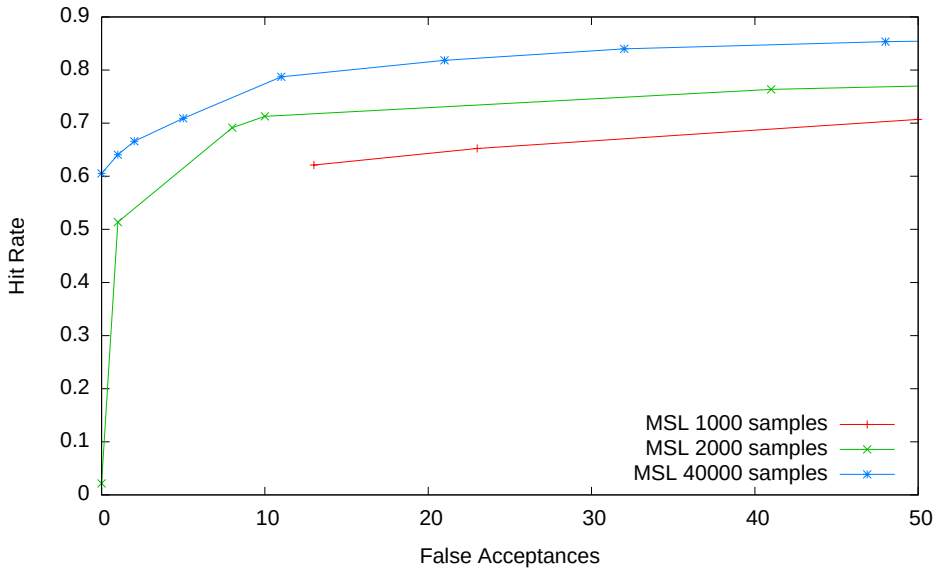

Figure 14: Comparison of cascades trained with different amount of samples. Increasing the training set size improves the performance at the cost of training time and memory consumption.

5.4 Classifier Combination

The combination of cascade classifiers is analyzed in the following subsections. First, we analyze whether potential performance gains are due to combination or solely due to the associated threshold optimization. Then, the necessity and effects of subsampling the amount of negative samples for cascade combination are examined. Finally, different cascade combination cost functions are evaluated and additional experiments include determining the effect of combining stages from more than two cascades.

5.4.1 Threshold Optimization

Threshold optimization of pre-trained cascade classifiers has been studied before, i.e. by Luo [38]. The published works have shown substantial improvements over the initial base cascade by adjustinging stage thresholds. The algorithms proposed in this work do not specifically aim at improving the stage threshold, but in order to successfully recombine stages, the thresholds need to be adjusted as well. Since the stages were trained to be used at a certain position within a cascade, the threshold determined during training may not be ideal anymore. Therefore, each stage i's threshold t_i is optimized based on the

same cost function that is used to combine the classifiers. Thresholds are adjusted using a coarse-to-fine search within a certain range from the original value. The search interval is $[t_i - 0.5, t_i + 0.5]$, which has been found empirically. This subsection analyzes the influence of the threshold optimization on the overall performance to ensure that the gains are not solely due to choosing better thresholds.

Figure 15 shows the initial base cascade, the cascade after threshold optimization and the CMIM combination of three base cascades' stages with the same amount of final stages. As can be seen from the plot, threshold optimization by itself provides an increase in the detection rate for a given false alarm rate. Using a combination of stages of different base classifiers can further improve results. Therefore, the performance gains of cascade combination are not solely due to the integrated threshold optimization.

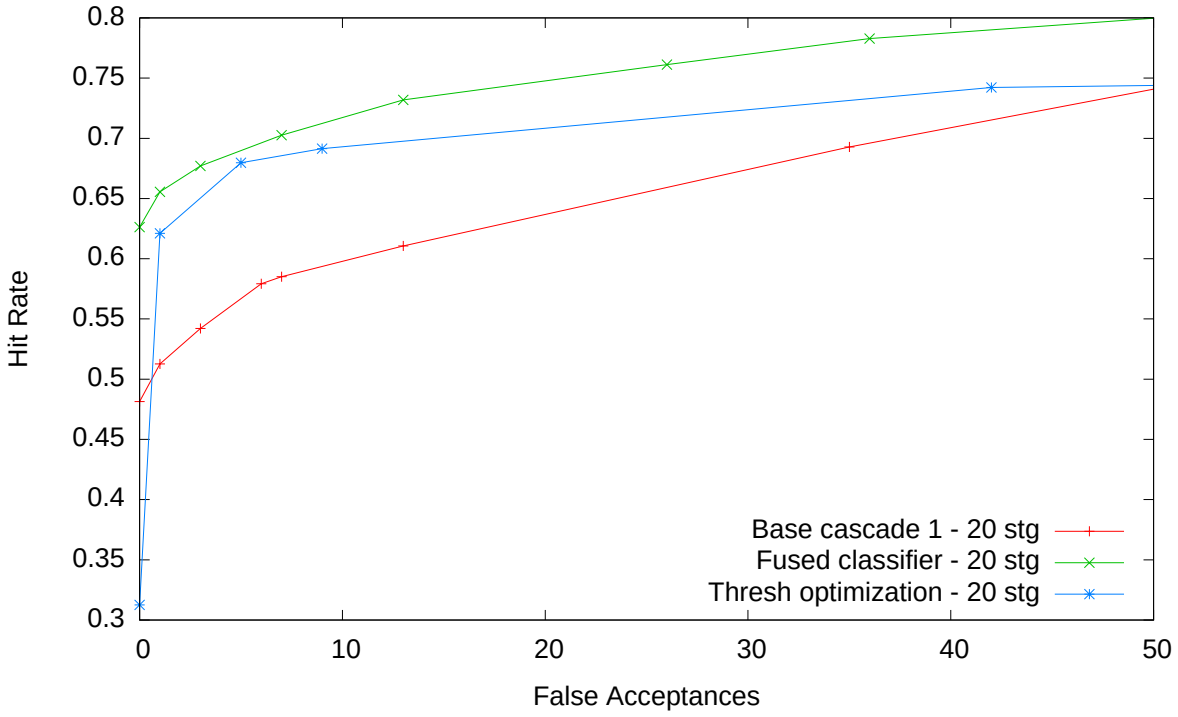

Figure 15: Cascade combination vs. threshold optimization of a single cascade. Threshold optimization improves the classification results. Additional performance gains can be achieved with cascade combination.

5.4.2 Subsampling for Cascade Combination

As noted before, due to the strong imbalance of negative and positive samples extracted for the optimal cascade combination and the bias of cost functions regarding class sample sizes, there is a need to reduce the imbalance for the combination to perform best. If the optimization set contains too many negative samples, the cost function may prefer stages that correctly reject more negative samples at the expense of also rejecting more positives and degrading the overall performance. On the other hand, forcing the combination process to adopt a slight bias may be helpful. Once the optimal parameter for subsampling has been determined, that parameter does not need to be changed much for different cascades trained with different parameters or count of stages. Figure 16 shows the influence of subsampling on each of the combination cost functions. As it was assumed, correlation is not highly influenced by subsampling because the class priors play no role. CMIM and MIM are more dependent on subsampling, especially MIM. Figure 16 shows the varying classification performance for different subsampling parameters. The MIM cost function does not handle imbalanced data sets well, whereas the other two methods seem to be less influenced. The necessity to determine optimal subsampling parameters complicates the combination process and is therefore not very desirable, this is a major draw-back for the MIM cost-function. For the other cost functions, subsampling turns out to be a vast speed improvement since, after subsampling, much fewer samples have to be analyzed when selecting each stage.

5.4.3 Comparison of Cascade Combination Cost Functions

Figure 17 shows a comparison of the three different cost functions. The MIM cost function outperforms both correlation and CMIM cost functions. While the CMIM cost function performs better than the correlation cost function. The graph was generated with optimal subsampling parameters for each cost function, all other parameters were the same.

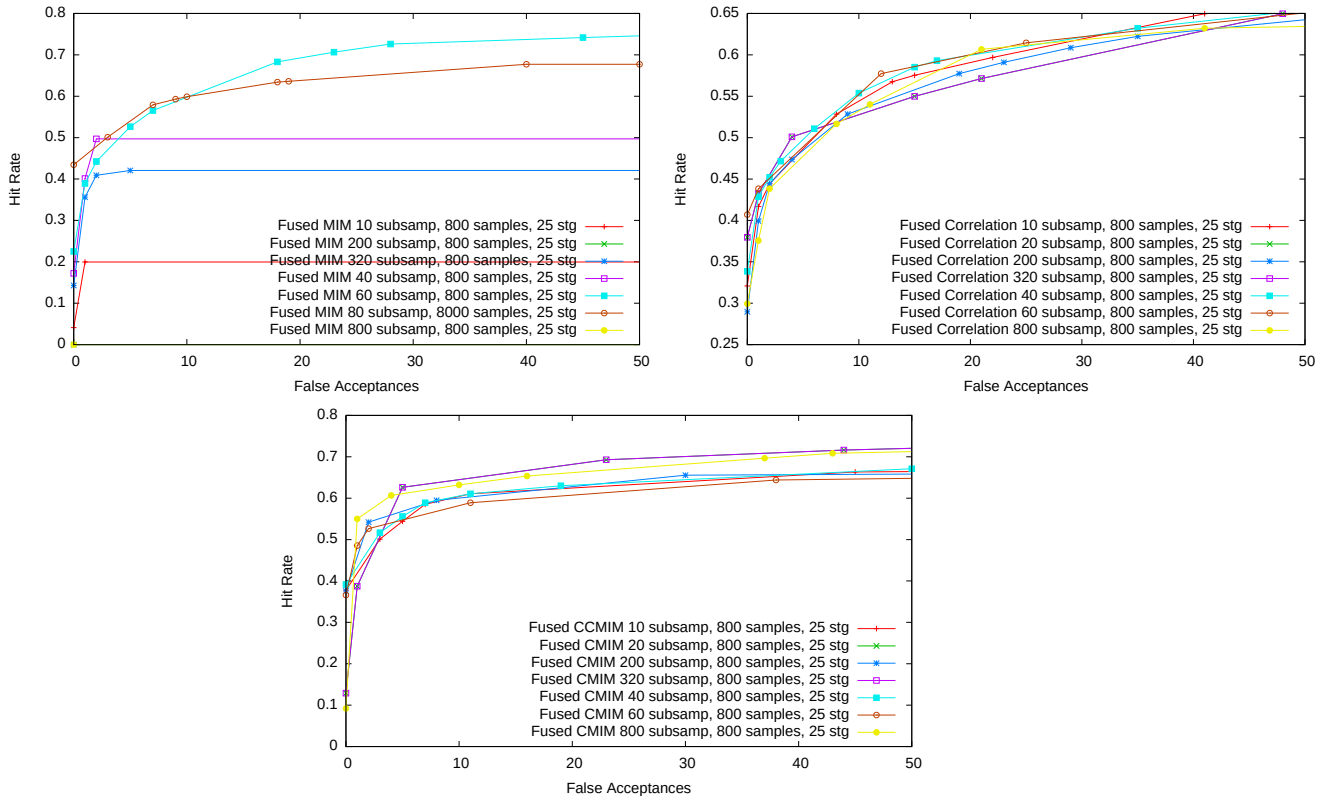

Figure 16: Cascade combination, the effect of subsampling for MIM, CMIM and correlation cost functions. While CMIM and correlation cost functions are not very sensitive to imbalances of positive and negative sets, the MIM cost function is.

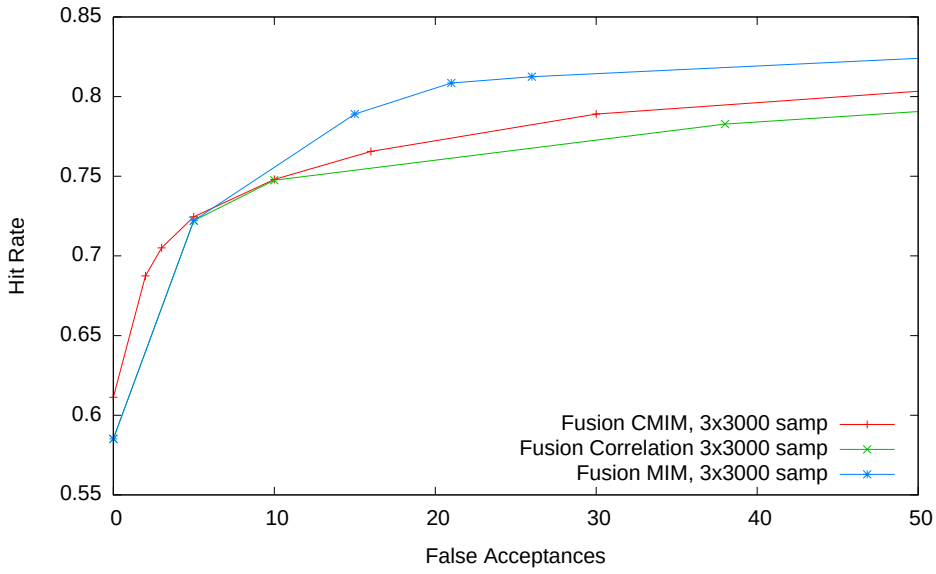

Figure 17: Comparison of the MIM, CMIM and correlation cost functions for combined 3000 sample trained cascades. The MIM cost function outperforms the other cost functions.

5.4.4 Combination Stage Count

The use of cascade combination would allow to construct a final cascade with more stages than the base cascades were made up of. Theoretically, adding more stages to the cascade should further reduce the false alarm rate - at the cost of slightly decreasing the correct detection rate. Figure 18 demonstrates the result of the associated experiment. Reducing the final stage count roughly keeps the detection rate at higher false alarm rates but reduces the detection rate at low false alarm rates. Adding more stages seems to yield a mild increase of the detection rate at low false alarm rates, whereas the detection rate at higher false alarm rates suffers.

The false alarm rate of a cascade classifier is determined by its stages. It is possible to reduce the false alarm rate by tightening the thresholds so that fewer negative samples are accepted as positive. This also decreases the correct detection rate. Another way to reduce the false alarm rate is to add additional stages. In ordinary cascades, later stages are trained with more difficult training examples, therefore they tend to be more complex. In cascade combination, none of these complex stages trained with more challenging training examples are available. Thus, when adding new stages, the only possibility to decrease false alarm rates is to utilize stricter thresholds for later stages. Figure 18 shows that

adding more stages is not beneficial in this case - the same effect could be achieved with stricter thresholds on earlier stages.

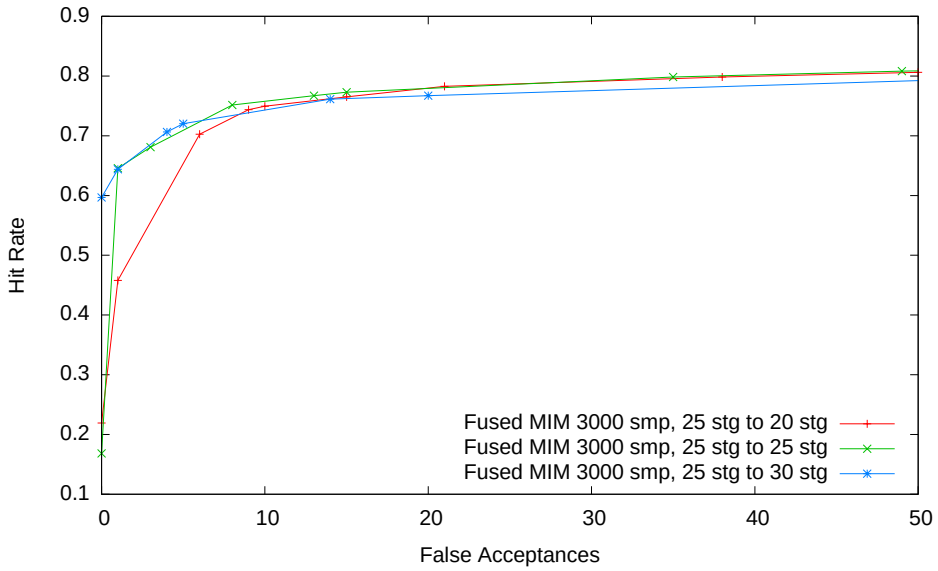

Figure 18: Combining 25 stage base cascades into 20, 25 and 30 stage final cascades. Increasing the number of final stages increases the hit rate at low false alarm rates but also decreases the overall hit rate slightly.

5.4.5 Comparison against Base Classifiers

Figures 19 and 20 show a comparison of combined classifiers versus the base classifiers from which the combination was constructed.

Figure 19 shows a clear improvement of both combined cascades over the initial base cascades. The detection rate, especially at low false alarm rates, is improved for a given false alarm rate. The CMIM and MIM cost function perform almost similarly. The base cascades were 20-stage classifiers as described in Section 5.3.2.

Figure 20 shows the same comparison for more extensively trained cascades. The base cascades whose results are shown in Figure19 were trained in about seven to eight hours each, the 25-stage cascades in Figure 20 were trained for about ten days each. The extensive training seems to have improved the cascades significantly. Compared to Figure 19, the detection rate at 50 false acceptances rises slightly. But especially in lower false alarm regions, the detection rate has been improved substantially. The curve is near the maximum detection rate even for lower false alarm rates and drops slower than before.

Therefore, the combination approach does not seem able to produce any substantial gains over the 25-stage base cascades anymore. Although there is a slight improvement, the performance gains are much lower than those seen in Figure 19.

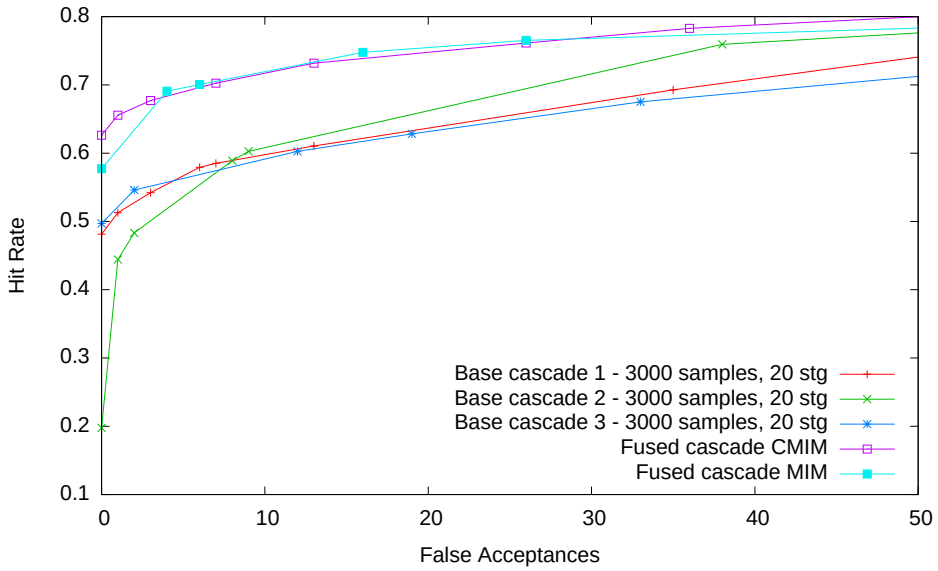

Figure 19: Comparison of 20-stage base cascades and combined cascades with 20 stages. A substantial improvement is possible here with both MIM and CMIM cost functions.

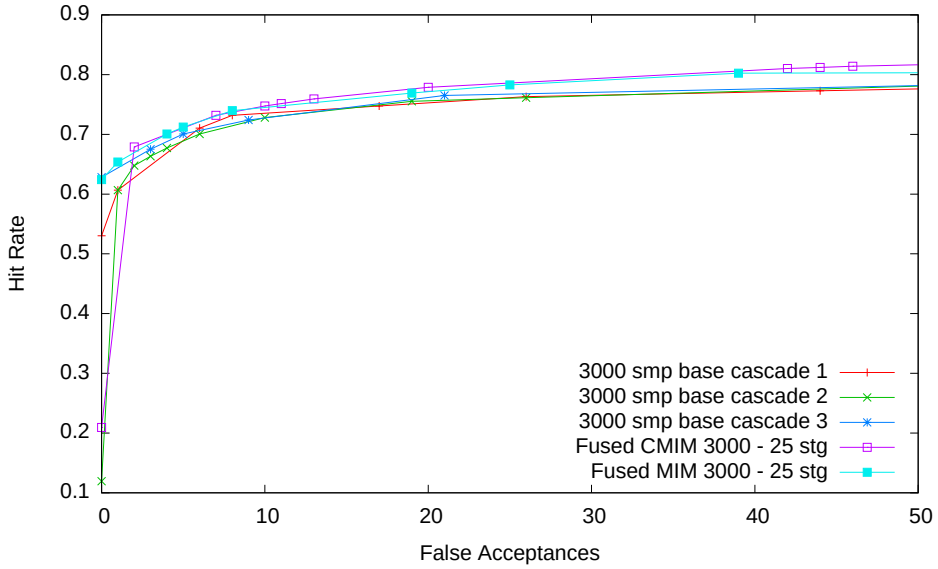

Figure 20: Comparing cascade combination against well-trained 25-stage base classifiers. Only marginal improvements can be achieved over well-trained base cascades.

5.5 SVM-based Training Set Selection

The following experiments examine the effect of resampling the positive training set with use of support vector machines. First, we compare the training of an SVM-based face detector trained using histogram equalization-based features and raw image intensity-based features. The next experiment performs an analysis of the possible performance gains by SVM-based sample set selection.

5.5.1 Histogram Equalization vs. Raw Image Intensity Features

In order to resample the positive training set with the help of support vector machines, an SVM has to be trained first. The SVM is trained by bootstrapping the negative training samples. In face detection, the SVM has to distinguish face and non-face patterns. Since face detection is a complex task with a lot of variance, partially due to different lighting situations, normalization techniques have been shown to improve classification when applied before-hand. Histogram equalization worked well and Figure 21 shows that classifiers trained with histogram equalized data performed better. False alarm rates were lower and detection rates higher when compared to raw intensity features after the same number of bootstrapping iterations.

5.5.2 SVM-based Training Set Resampling

Figure 22 illustrates the performance of classifiers trained with SVM-based training set resampling compared to the best similar base cascade. The figure compares the base cascades trained with 3000 positive and negative samples against two cascades with samples selected based on SVM distance to hyperplane. An SVM-based face detector was used to create a ranking of 9,000 initial training samples. These 9,000 samples are the same that were partitioned and used to train the base classifiers. The SVM-based feature ranking was then used to select 3000 positive samples with which cascade classifiers were trained. The label “hardest” describes the sample set comprised of 3000 samples closest to the SVM hyperplane, “equidist” describes samples selected from the whole range of samples. Both resampling techniques achieve similar performance. Covering the whole range of samples seems slightly advantageous, as was to be expected, because the whole face space is covered. But the difference is minimal and might as well be due to the bootstrapping of negative samples. In experiments, two iterations of bootstrapping were found to be

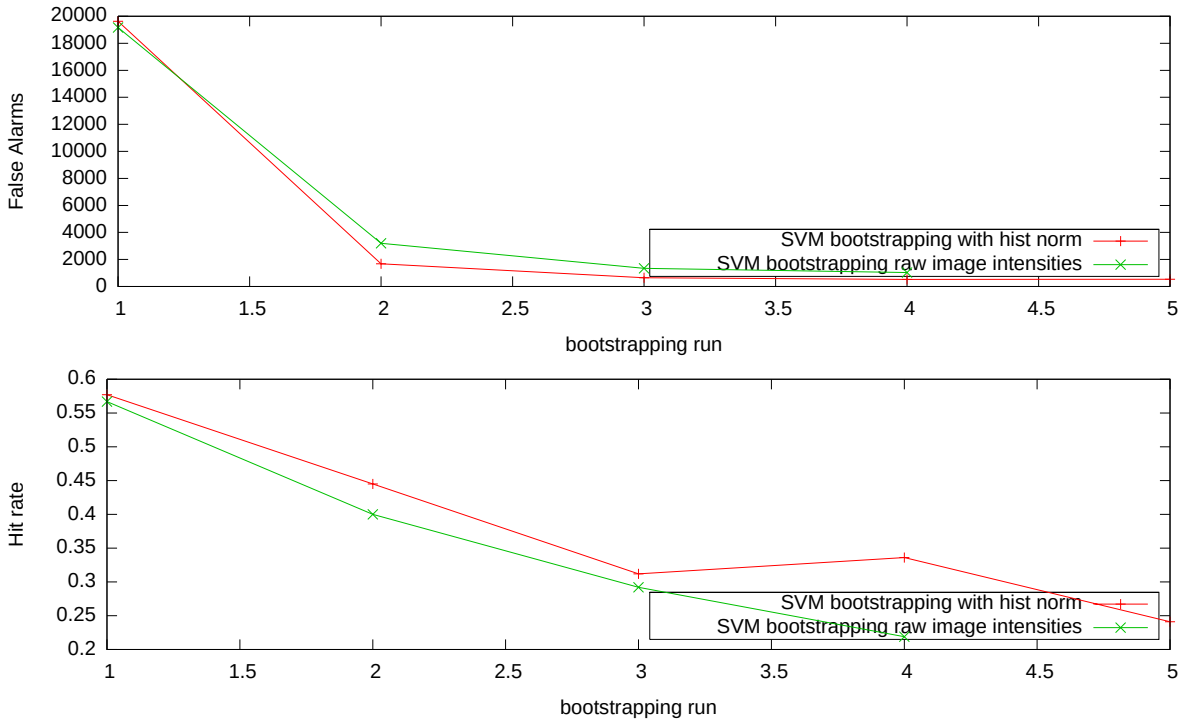

Figure 21: SVM bootstrap training - 3000 positive, 5000 negative samples, with and without histogram normalization of input images.

enough to produce a useful ranking. More bootstrapping iterations had little to no impact on the classifier's performance.

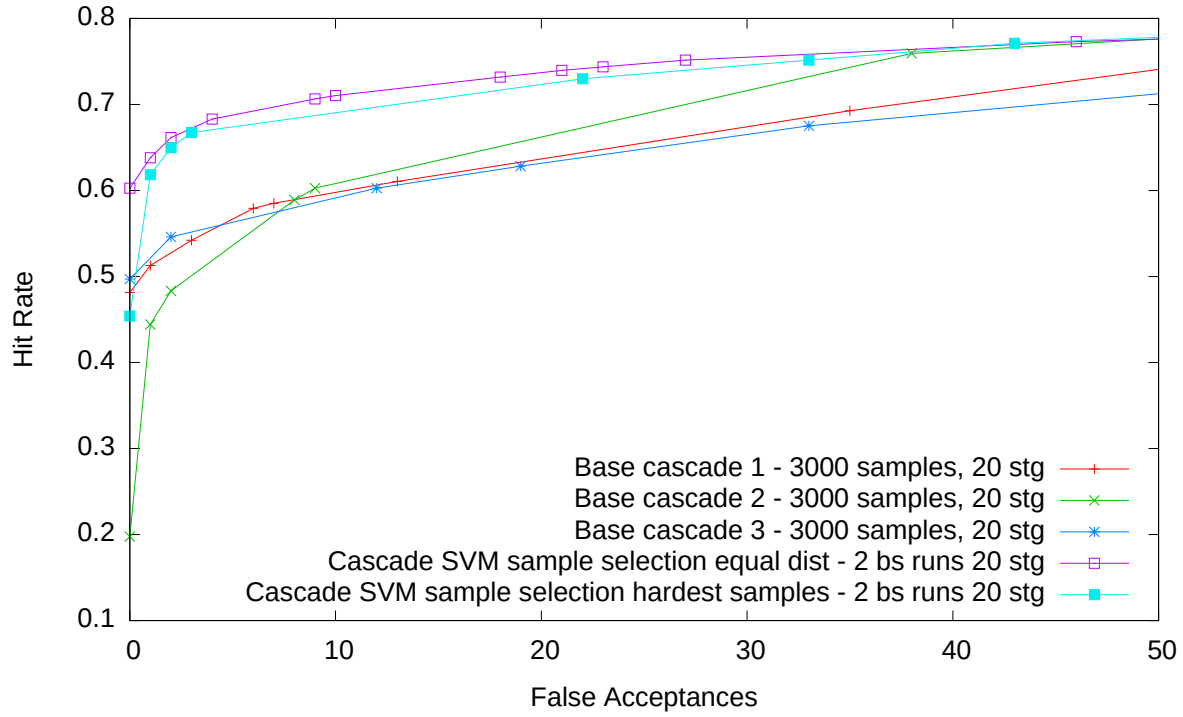

Figure 22: Cascades trained with SVM-based bootstrapping, 3000 training samples and 20 stages vs. 20-stage base cascades. Cascades trained with SVM-based training sample selection outperform the base cascades.

5.6 Combining SVM-based Resampling and Cascade Combination

In this subsection, the application of both training set optimization and cascade combination has been studied. Figure 23 compares the base cascade trained with 9000 training samples with a combination of two cascades whose 3000 training samples were acquired with both different SVM-based resampling methods. Before applying the cascade combination algorithm with the MIM cost function, two cascade classifiers were trained with 3000 training samples each. These training samples were selected using the SVM-based training sample selection as outlined in Section 4.3.1. The figure shows that it is possible to achieve almost similar and in part better performance with combination of two carefully trained 3000 sample cascades as with a single 9000 sample cascade. The combined

cascade resulted in 20 stages, the two base cascades were trained to 20 stages and the 9000 sample cascade also contained 20 stages. The 3000 training samples for the cascades were selected from the 9000 sample set used to train the larger cascade.

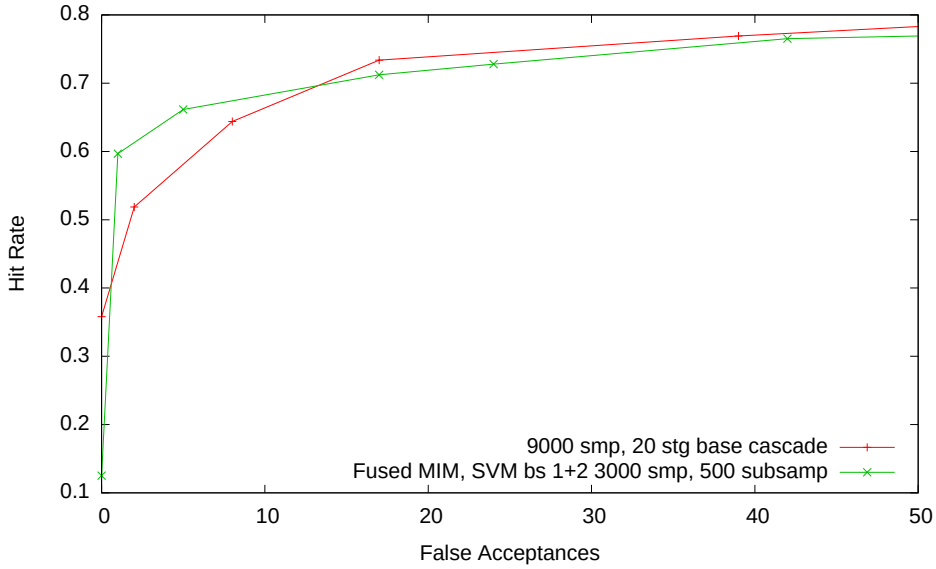

Figure 23: Comparison of a base cascade trained with 9000 positive and negative samples with a combined cascade. The combined cascade's input classifiers were trained with SVM-based sample selection, 3000 samples for training were selected from 9000 initial samples. The two input cascades' samples were selected with the help of an SVM and two different sample selection strategies. The MIM cost function was used. Both the 9000 sample base cascade and the combined cascade have 20 stages.

5.7 Comparison against other Cascade Classifiers

This section compares the presented methods to other publicly available classifiers. Figure 24 shows a comparison of several classifiers that were created for this work against publicly available OpenCV cascades. The figure demonstrates that our cascades clearly outperform the OpenCV face detectors. Some classifiers are not visible in the graph, because, in order to preserve details in that most important part of the ROC, the window was focused on false alarm rates in $[0, 50]$. The invisible classifiers were unable to produce any hits at false alarm rates below 50. The comparison was done under the same premises, all cascades were evaluated with the ROC generation approach outlined in Section 5.2 and the same parameters.

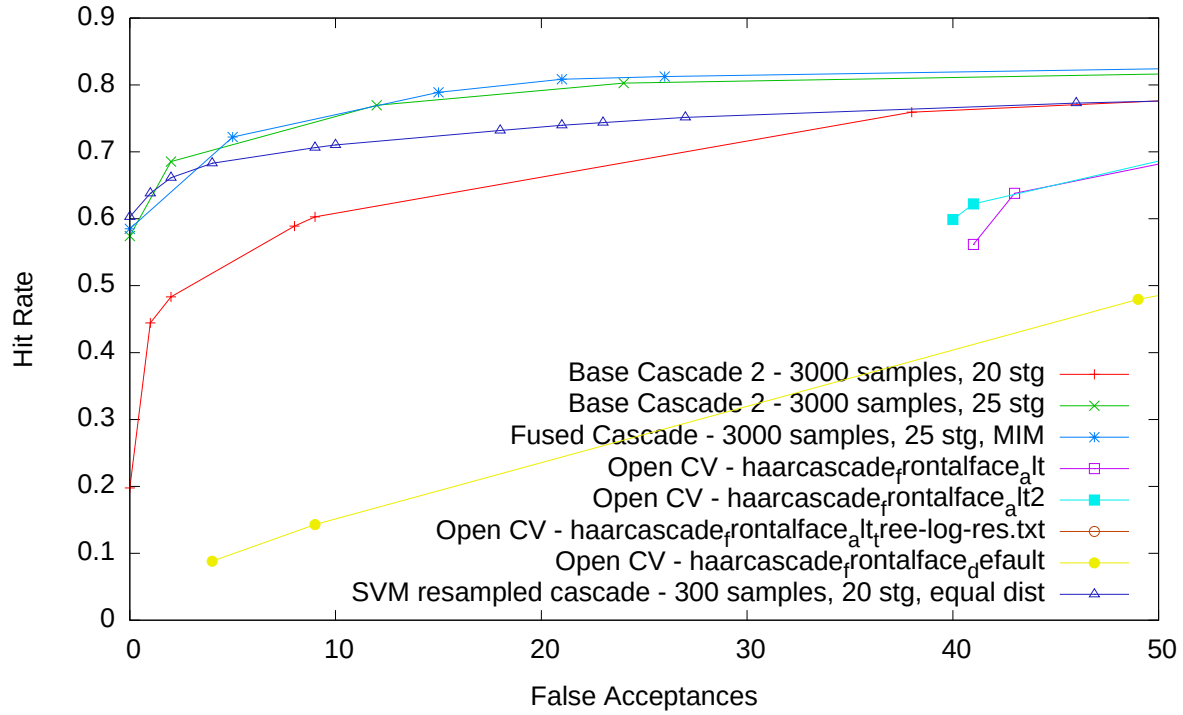

Figure 24: Comparison against OpenCV default cascade

Figure 25 compares our cascades against a state-of-the-art classifier by Luo [38]. Luo's classifiers [38] are constructed by automatic threshold optimization and outperform the presented classifiers. It should be noted that the base classifier used for the threshold optimization already performed much better than our base classifiers, even our final classifiers. Luo's base cascade [38] before the threshold optimization, which is not shown in the figure, has a detection rate that is about 10% higher on average compared to our base classifiers. Our results indicate that cascade combination can improve classification results but they cannot achieve state-of-the-art performance. For better comparison, base cascades would have to be trained that match the performance of the initial cascades in other publications.

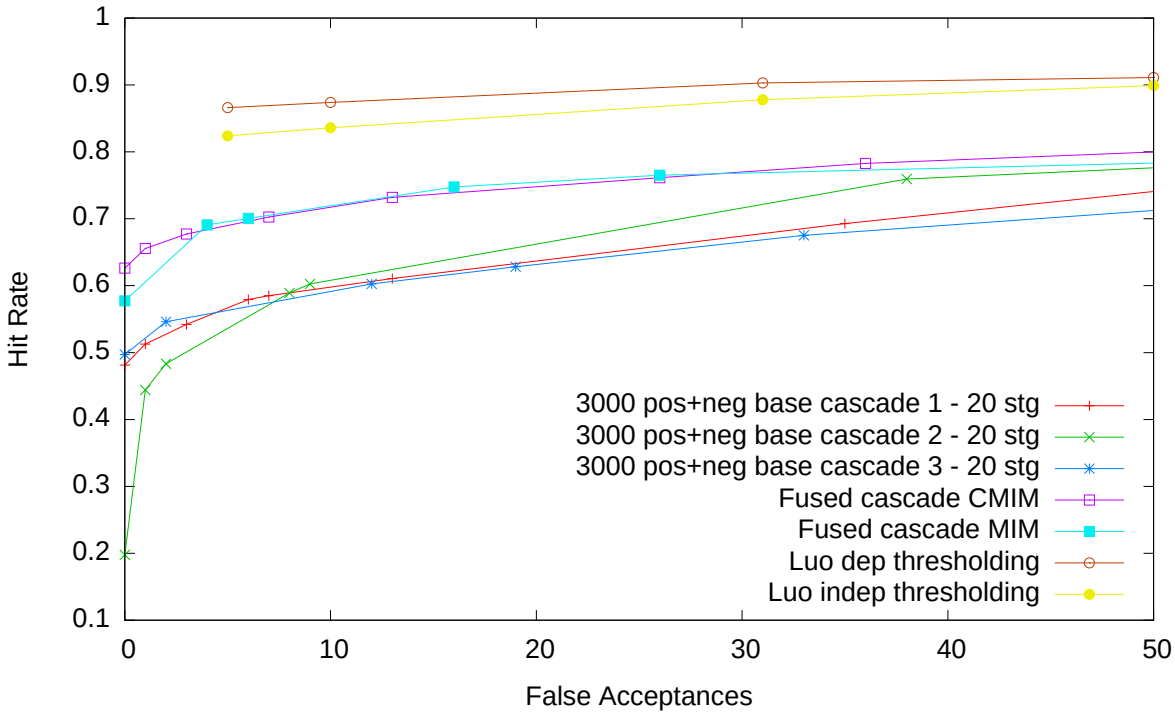

Figure 25: Comparison against state-of-the-art classifiers [38].

6 Conclusion

This work presents two methods to improve the training of face detection cascades. The first method, cascade combination, attempts to combine the advantages of classifier combination without the additional overhead of running several classifiers in parallel. Second, SVM-based training set selection seeks to optimize the training set in a conceptually simple fashion in order to force the final classifier to focus on the decision boundary. Both methods have been shown to improve the classification performance over base classifiers. All experiments were conducted on the widely used, public CMU+MIT benchmark database in order to allow comparison with other work.

In this study, several base classifiers were trained so as to create a baseline for evaluating the performance of the proposed cascade optimization techniques. An initial training set was divided into several smaller sets for various experiments. Several classifiers were trained based on these sub-sets and used as base classifiers for cascade combination and as a comparison baseline for the training set resampling.

A framework to represent cascade combination as an optimization and three methods for cascade combination have been presented. The first method is based on conditional mutual information as a cost function and is akin to a successful feature selection approach that has been studied. Another method is based on simple mutual information between each stage's classification combined with previously selected stage classifications and the true sample class labels. These methods seek to directly optimize the correlation between the set of selected stages plus a prospective next stage and the class labels. Another method based on linear correlation has also been explored for comparison.

The cascade combination approach was shown to improve the performance of the base classifiers by utilizing and combining several stages of different cascade classifiers. It was also shown that cascade combination can be used to reduce training times by combining several stages of quickly trained classifiers into a combined classifier with superior performance that can achieve comparable performance as an extensively trained classifier. The required overall time could be reduced from more than ten days to a single day, with only a slight performance drop. Cascade combination is especially useful for quickly trained cascades where large performance benefits can be achieved. Extensively trained cascades on the other hand do not have as much potential for improvement.

The second method explored in this work is SVM-based training set selection. By resam-

pling the training set, emphasis is put on samples near the decision hyperplane, where most of the errors occur. An SVM is trained and used to generate a ranking of samples by difficulty, by using their distance from the hyperplane of the SVM. Two resampling techniques have been studied, one selects samples closest to the decision boundary, the other selects samples from the whole set. Both resampling techniques perform almost equally well with a slight advantage for the set sampled from the whole initial training set. Cascade classifiers trained with a resampled training set outperformed comparable base cascades trained with random training sets.

Both the SVM-based training set resampling and cascade combination methods proved useful for increasing the detection rate of cascade classifiers. Additionally, a combination of both methods can be used to create a cascade classifier that is trained with 3000 samples that can achieve the performance of a classifier that was trained with 9000 samples. This property proves to be useful for cascade classifiers trained with large training sets and many stages as the bootstrapping time increases exponentially and large amounts of memory would be necessary in order to cache feature value.

7 Future Work

Several improvements to the proposed methods are possible and could be addressed in future work.

Cost functions: Other cost functions may have a positive impact on the performance of combined classifiers and may be less prone to the imbalances of the validation set used for combination. A cost function that can handle unbalanced sets eliminates another training parameter that has to be optimized in a time-consuming fashion.

Improved base cascades: In order to improve the overall performance of the combined cascades better base cascade classifiers are necessary. Although a substantial amount of time was spent on training and evaluating base cascades, the presented cascades are unable to match the base cascades used in state-of-the-art methods, like those in [38]. Generally, cascade training is not trivial and requires a lot of know-how and engineering effort. Heuristics and a feeling for the importance of the many parameters must be acquired which is a lengthy process. A survey of publicly available cascades by Castrillón-Santana et al. [9] suggests that publicly available Haar cascade classifiers based on OpenCV tend to be inferior to state-of-the-art implementations. The biggest problem so far is the exponential increase in bootstrapping time after about 20 stages. Most publications report cascades trained with 30-40 stages, but in our experiments, even after 10 days, only 28-stage cascade classifiers had been trained. More stages should decrease the false alarm rate. But the decrease in false alarms may come at a reduced detection rate. Therefore, it has to be seen whether the benefit of adding stages is not negated by the decrease of the detection rate. Another possibility to increase the performance of base cascades is the use of a larger training set, although care has to be taken to have sufficient free memory for caching. Otherwise training time will increase dramatically.

Threshold optimization: Besides improved subsampling, an improved threshold optimization algorithm could also help improve the detection rates. As demonstrated before, part of the improvement is due to better choices of thresholds than in the original cascades. Several threshold optimization algorithms have been proposed in other publications. An approach based on the work by Luo [38] would be a prime candidate for implementation in future work. Instead of using an iterative algorithm as suggested by Luo [38] that

shifts ROC mass from weaker stages to more powerful stages, thresholds could be directly chosen to be optimal using cached feature values and classification tables.

Bootstrapping of cascade combination stage negative sets: Since in cascade combination, as during cascade training, stage thresholds have to be adapted for a specific point in a cascade, bootstrapping may be helpful to adjust the stages better for each selection round. Early stages are mostly meant to reject trivial negatives; later stages should reject more difficult examples. It may be helpful if the validation set is chosen accordingly.

Reuse of stage feature value sums: The idea of a boosting chain or soft cascade [3, 70] and the implementation of multiple instance pruning [77] should help to increase the classification performance. The experiments and the cascade combination framework were initially designed to be used in conjunction with some form of continuous feature sums that are not reset after stage boundaries as introduced by Bourdev and Brant [3] or Xiao et al. [70]. Several publications that use continuous feature sums show substantial improvements in classification performance.

Reducing training time: As stated in Section 5.3.3, in the training of late stages the training time itself for feature selection and combination of weak classifiers becomes negligible compared to the bootstrapping time. Presumably, the high bootstrapping times could be due to the fact that the background images do not contain enough difficult image patches. Although the background images seem to contain a large variety of motives with clutter, adding more background images or more complex images may be necessary.

References

[1] T. Agui, Y. Kokubo, H. Nagashashi, and T. Nagao. Extraction of face recognition from monochromatic photographs using neural networks. *Proc. Second Int'l Conf. Automation, Robotics, and Computer Vision*, 18:81–85, 1992.

[2] M.F. Augusteijn and T.L. Skujca. Identification of human faces through texture-based feature recognition and neural network technology. *Proc. IEEE Conf. Neural Networks*, pages 392–398, 1993.

[3] L. Bourdev and J. Brandt. Robust object detection via soft cascade. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2:236–243, 2005.

[4] L. Brieman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[5] S.C. Brubaker, M.D. Mullin, and J.M. Rehg. Towards optimal training of cascaded detectors. *European Conference on Computer Vision*, 1:325–337, 2006.

[6] S.C. Brubaker, J. Wu, J. Sun, M.D. Mullin, , and J.M. Rehg. On the design of cascades of boosted ensembles for face detection. *Int'l Journal of Computer Vision*, (77):65–86, 2008.

[7] M. C. Burl, T. K. Leung, and P. Perona. Face localization via shape statistics. *Proc. First Int'l Workshop Automatic Face and Gesture Recognition*, pages 154–159, 1995.

[8] J. Cai, A. Goshtasby, and C. Yu. Detecting human faces in color images. *Proc. 1998 Int'l Workshop Multi-Media Database Management Systems*, pages 124–131, 1998.

[9] M. Castrillón-Santana, O. Déniz-Suárez, L. Antón-Canalís, and J. Lorenzo-Navarro. Face and facial feature detection evaluation performance evaluation of public domain haar detectors for face and facial feature detection. *VISAPP*, pages 167–172, 2008.

[10] D. Chai and K.N. Ngan. Locating facial region of a head-and-shoulders color image. *Proc. Third Int'l Conf. of Automatic Face and Gesture Recognition*, pages 124–129, 1998.

[11] J. Chen, X. Chen, J. Yang, S. Shan, R. Wang, and W. Gao. Optimization of a training set for more robust face detection. *Pattern Recognition Journal*.

[12] J. Chen, R. Wang, S. Yan, S. Shan, X. Chen, and W. Gao. Enhancing human face detection by resampling examples through manifolds. *IEEE Transactions on Systems, Man, and Cybernetics*, 6:1017–1028, 2007.

[13] X. Chen and A.L. Yuille. A time-efficient cascade for real-time object detection: With applications for the visually impaired. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 20–26, 2005.

[14] T.F. Cootes and C.J. Taylor. Locating faces using statistical feature detectors. *Proc. Second Int'l Conf. Automatic Face and Gesture Recognition*, pages 204–209, 1996.

[15] Y. Dai and Y. Nakano. Face-texture model based on sgld and its application in face detection in a color scene. *Pattern Recognition*, 29(6):238–242, 1996.

[16] M. Dundar and M.J. Bi. Joint optimization of cascaded classifiers for computer aided detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[17] F. Fleuret. Fast binary feature selection with conditional mutual information. *The Journal of Machine Learning Research*, 5:1531–1555, 2004.

[18] D. Forsyth. A novel approach to color constancy. *Int'l Journal of Computer Vision*, 5(1):5–36, 1990.

[19] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[20] J. Friedman, T. Hastie, , and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:337–407, 2000.

[21] B. Fröba and A. Ernst. Face detection with the modified census transform. *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 91–96, 2004.

[22] D. Grosvenor. Integrating object detectors. Technical Report HPL-2007-66, Media Technologies Laboratory HP Laboratories Bristol, 2007.

[23] C.-C. Han, H.-Y.M. Liao, K.-C. Yu, and L.-H. Chen. Face face detection via morphology-based pre-processing. *Proc. Ninth Int'l Conf. Image Analysis and Processing*, pages 469–476, 1998.

[24] B. Heisele, T. Serre, M. Pontil, and T. Poggio. Component-based face detection. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1:pp. 657, 2001.

[25] C. Huang, H. Ai, Y. Li, and Sh. Lao. Vector boosting for rotation invariant multi-view face detection. In *Proc. IEEE International Conference on Computer Vision*, pages 446–453, 2005.

[26] C. Huang, H. Z. Ai, Y. Li, and S. H. Lao. High-performance rotation invariant multiview face detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(4):671–686, 2007.

[27] Y. Ishii, K. Imagawa, E. Fukumiya, K. Iwasa, and Y. Ogura. Profile face detection using block difference feature for automatic image annotation. *Proc. IEEE International Conference on Consumer Electronics*, pages 337– 338, 2006.

[28] M. Jones and P. Viola. Fast multi-view face detection. *MERL TR2003 96*, July 2003.

[29] T. Kanade. *Picture Processing by Computer Complex and Recognition of Human Faces.* PhD thesis, Kyoto University, 1973.

[30] M. Kirby and L. Sirovich. Application of the karhunen-loeve procedure for the characterization of human faces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(1):103–108, 1990.

[31] C. Kotropoulos and I. Pitas. Rule-based face detection in frontal views. *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 4:2537–2540, 1997.

[32] Y. H. Kwon and N. da Vitoria Lobo. Face detection using templates. *Proc. Int'l Conf. Pattern Recognition*, pages 764–767, 1994.

[33] T.K. Leung, M.C. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graph matching. *Proc. Fifth IEEE Int'l Conf. Computer Vision*, pages 637–644, 1995.

[34] S. Z. Li, L. Zhu, and Z. Q. Zhang. Statistical learning of multi-view face detection. *Proc. European Conference on Computer Vision*, pages 67–81, 2002.

[35] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *DAGM03*, pages 297–304, Madgeburg, Germany, 2003.

[36] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. *Proc. IEEE International Conference on Image Processing*, 1:900–903, 2002.

[37] C. Liu and H.Y. Shum. Kullback-leibler boosting. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, (587-594), 2003.

[38] H. Luo. Optimization design of cascaded classifiers. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 480–485, 2005.

[39] B. McCane, K. Novins, and M. Albert. Optimizing cascade classifiers. *Unpublished.*

[40] S. McKenna, Y. Raja, and S. Gong. Tracking colour objects using adaptive mixture models. *Pattern Recognition*, 31(12):1883–1892, 1998.

[41] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(4):349–361, 2001.

[42] K.R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Schoelkopf. An introduction to kernel-based learning algorithms. *IEEE Trans. on Neural Networks*, 12(2):181–201, 2001.

[43] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[44] M.-T. Pham and T.-J. Cham. Online learning asymmetric boosted classifiers for object detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR07)*, pages 1–8, 2007.

[45] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 203–208, 1996.

[46] H. A. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 38–44, 1998.

[47] A. Samal and P.A. Iyengar. Human face detection using silhouettes. *Int'l J. Pattern Recognition and Artificial Intelligence*, 9(6):845–867, 1995.

[48] F.S. Samaria. *Face Recognition Using Hidden Markov Models*. PhD thesis, Univ. of Cambridge, 1994.

[49] D. Saxe and R. Foulds. Toward robust skin identification in video images. *Proc. Second Int'l Conf. Automatic Face and Gesture Recognition*, pages 379–384, 1996.

[50] R. Schapire. The boosting approach to machine learning: An overview. *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[51] R. E. Schapire, Y. Freund, P. Barlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *International Conference on Machine Learning*, pages 322–330, 1997.

[52] R.E. Schapire and Y.Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.

[53] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 45–51, 1998.

[54] B. Schoelkopf. Support vector learning. *DAGM'99*, 1999.

[55] P. Sinha. Object recognition via image invariants: A case study. *Investigative Ophthalmology and Visual Science*, 35(4):1735–1740, 1995.

[56] K. Sobottka and I. Pitas. Face localization and feature extraction based on shape and color information. *Proc. IEEE Int'l Conf. Image Processing*, pages 483–486, 1996.

[57] J. Sochman and J. Matas. Waldboost – learning for time constrained sequential detection. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 150–156, 2005.

[58] F. Soulie, E. Viennet, and B. Lamy. Multi-modular neural network architectures: Pattern recognition applications in ptical character recognition and human face recognition. *Int'l J. Pattern Recognition and Artificial Intelligence*, 7(4):721–755, 1993.

[59] J. Sun, J. M. Rehg, and A. Bobick. Automatic cascade training with perturbation bias. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:276–283, 2004.

[60] K.-K. Sung. *Learning and Example Selection for Object and Pattern Detection.* PhD thesis, EECS, MIT, February 1996.

[61] K.-K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(1):39–51, 1998.

[62] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[63] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 511–518, 2001.

[64] P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. *Advances in Neural Information Processing Systems*, 14:1311–1318, 2002.

[65] P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. *Advances in Neural Information Processing Systems*, 14:1311–1318, 2002.

[66] P. Wang and Q. Ji. Learn discriminant features for multi-view face and eye detection. In *Proc. IEEE International Conference on Computer Vision*, pages 373–379, 2005.

[67] P. Wang and Q. Ji. Multi-view face and eye detection using discriminant features. *Computer Vision and Image Understanding*, pages 99–111, 2007.

[68] J. Wu, S.C. Brubaker, M.D. Mullin, and J.M. Rehg. Fast asymmetric learning for cascade face detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 30, pages 369–382, 2007.

[69] J. Wu, J. Rehg, and M. Mullin. Learning a rare event detection cascade by direct feature selection. *Advances in Neural Information Processing Systems*, 16:1523–1530, 2003.

[70] R. Xiao, L. Zhu, and H.-J. Zhang. Boosting chain learning for object detection. *Proc. Fifth IEEE Int'l Conf. Computer Vision*, 1:709–715, 2003.

[71] S. Yan, S. Shan, X. Chen, W. Gao, and J. Chen. Matrix-structural learning (msl) of cascaded classifier from enormous training set. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1–7, 2007.

[72] G. Yang and T.S. Huang. Human face detection in complex background. *Pattern Recognition*, 27(1):53–63, 1994.

[73] J. Yang and A. Waibel. A real-time face tracker. *Proc. Third Workshop Applications of Computer Vision*, pages 142–147, 1996.

[74] M.-H. Yang and N. Ahuja. Gaussian mixture model for human skin color and its application in image and video databases. *Proc. ACM Human Factors in Computing Systems Conf. (CHI 98)*, pages 142–147, 1998.

[75] M.-H. Yang, David J. Kriegman, and Narendra Ahuja. Detecting faces in images: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(1):35–58, January 2002.

[76] M.-H. Yang, D. Roth, and N. Ahuja. A snow-based face detector. *Advances in Neural Information Processing Systems*, pages 855–861, 2000.

[77] C. Zhang and P. Viola. Multiple-instance pruning for learning efficient cascade detectors. *Advances in Neural Information Processing Systems*, 2007.

[78] D. Zhang, S.Z. Li, and D. Gatica-Perez. Real-time face detection using boosting in hierarchical feature spaces. *Proc. Int'l Conf. Pattern Recognition*, 2:411–414, 2004.

[79] Z. Q. Zhang, M. Li, S. Z. Li, and H.J. Zhang. Multi-view face detection with floatboost. *Proc. Sixth IEEE Workshop on Applications of Computer Vision*, page pp. 184, 2002.