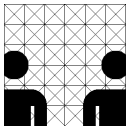
Diplomarbeit
im Studiengang Informatik

Merkmalsreduktion und Ensemble-Lernen mit AdaBoost am Beispiel der Kennzeichenlokalisierung

am Arbeitsbereich für
Technische Aspekte Multimodaler Systeme,
Universität Hamburg

vorgelegt von
Nils Meins
Dezember 2010



betreut von
Prof. Dr. Jianwei Zhang
Prof. Dr. Leonie Dreschler-Fischer

Abstract

In this assignment we are analysing the ensemble learning AdaBoost according to the example of the licence plate localization. We analyse the behaviour of the learning method using different classifier which are combined to an ensemble.

One disadvantage of AdaBoost as an ensemble-learning method is the long training duration.

We can varify AdaBoost a bit by making only minor changes. That way we are able to reduce a large quantity of informations.

Using these fewer informations as basis we are able to train good essemble classifier in a short time.

To train the final ensemble classifier we use AdaBoost in its unmodified form.

Finally we compare our results with two different cascades. One with an exhaustive search and one that was trained with a randomised limited attribute quantity. Using Threshold-, Binning-, FuzzyBinning-classificators and the "Joint Haar-like Feature"we furnish proof that our method is giving good results.

Zusammenfassung

In dieser Arbeit untersuchen wir das Ensemble-Lernverfahren AdaBoost am Beispiel der Kennzeichen-Lokalisierung. Wir untersuchen das Verhalten des Lernverfahrens unter Nutzung verschiedener Klassifikatoren, die zu einem Ensemble kombiniert werden.

Ein Nachteil von AdaBoost als Ensemble-Lernverfahren ist die lange Trainingszeit. Durch wenige Einschränkungen verändern wir AdaBoost leicht und erreichen dadurch, dass wir mit ihm schnell große Mengen von Merkmalen auf kleine Mengen reduzieren können.

Auf Grundlage dieser kleinen Mengen können wir in kurzer Zeit gute Ensemble-Klassifikatoren trainieren.

Zum Trainieren der finalen Ensemble-Klassifikatoren nutzen wir AdaBoost in seiner unveränderten Form.

Wir vergleichen unsere Ergebnis-Kaskaden zum einen mit Kaskaden, die mit einer umfassenden Suche auf der gesamten Merkmalsmenge trainiert wurden und zum anderen mit solchen, die mit einer zufällig eingeschränkten Merkmalsmenge trainiert wurden.

Wir zeigen mit Schwellwert-, Binning-, FuzzyBinning-Klassifikatoren und den "Joint Haar-like Feature", dass unsere Methode gute Ergebnisse liefert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Einleitung

1

Kennzeichenerkennung ist weitreichend untersucht worden und hat viele erfolgreiche Lösungen. Es wurden die unterschiedlichsten Ansätze betrachtet und geprüft. Sowohl Systeme mit fester Kamera als auch mit mobiler Kamera wurden erstellt und eingesetzt.

So werden Kennzeichenerkennungssysteme z.B. für Zugangskontrollen eingesetzt, bei denen die Software nicht nur kontrolliert, wer einen bestimmten Bereich befahren darf, sondern auch protokolliert, welche Fahrzeuge wann in diesen Bereich fuhren und wann sie ihn wieder verließen.

Auch für Zahlsysteme bei Mautstrassen und Parkplätzen wird entsprechende Software für die Kennzeichenerkennung eingesetzt. In Deutschland wird für die LKW-Maut die Kennzeichenerkennung genutzt, um zahlpflichtige Fahrzeuge, die die Mautpflicht nicht wahrnehmen, zu identifizieren. Die "London Congestion Charge" ist ein gebührenpflichtiges Gebiet, das einfahrende Autos über die Kennzeichenerkennung registriert und abrechnet.

Ein weiteres Gebiet ist die Kennzeichenerkennung zur Unterstützung der Polizei und weiteren Strafverfolgungsbehörden, bei der automatischen Überprüfung von Kraftfahrzeugen.

Um Kennzeichen einer Erkennung zu unterziehen, müssen zwei Probleme gelöst werden. Erst muss das Kennzeichen gefunden werden und dann müssen die Zeichen innerhalb des Kennzeichens erkannt werden. Häufig wird als dritter Punkt zusätzlich die Zeichensegmentierung genannt, die in den meisten Verfahren vor der Zeichenerkennung verwendet wird.

Das Ziel dieser Arbeit ist die Untersuchung, welche weitergehenden lernenden Techniken bzgl. HaarFeature und Boosting geeignet sind, das Problem der Kennzeichen-Lokalisierung (Licence Plate Localization) zu lösen. Im Vordergrund steht das Szenario einer Parkplatzüberwachung. Viele Kennzeichenerkennungssysteme arbeiten in vorgegebenen und vereinfachenden Umgebungsbedingungen. So können z.B. Systeme die im Eingangsbereich von Parkhäusern wirken, den Bereich, in dem sich das Kennzeichen befindet, räumlich sehr stark eingrenzen. Auch sind die Lichtverhältnisse in solchen Szenarien weitesgehend ähnlich und der Hintergrund (sofern er überhaupt ins Bild kommmt) weist meist keine große Varianz auf.

In dieser Arbeit wird mit Trainings- und Testbildern gearbeitet, die keinen besonderen Einschränkungen unterliegen. Die Kennzeichen sollen unabhängig von den Wetter und Lichtverhältnissen die zur Zeit der Aufnahme herrschten gefunden werden und unabhängig von der verwendeten Hardware zur Erstellung der Bilder.

Boosting liefert insbesondere dann gute Ergebnisse, wenn die zur Auswahl stehende Merkmalsmenge besonders groß und vielfältig ist. Der schwache Punkt bei den Boosting-Verfahren ist ihre lange Trainingsdauer, was konträr zu der großen Merkmalsmenge steht. So beschäftigt sich der wesentliche Teil dieser Arbeit mit der Frage, wie die Trainingszeit verkürzt werden kann und trotzdem noch adäquate Klassifikatoren gewonnen werden können.

Ein wesentlicher Punkt ist die Anzahl der Merkmale so zu reduzieren, dass die guten Merkmale erhalten bleiben und eine gewisse Verschiedenartigkeit gewahrt bleibt.Das Gebiet, das sich mit dieser Fragestellung beschäftigt, ist die "Merkmals-Reduktion/Merkmals-Auswahl".

Der Weg den wir hier prüfen ist, AdaBoost selbst, mit wenigen Einschränkungen, zur Reduzierung der Merkmale zu nutzen.

So werden wir erst prüfen, in wie weit wir mit dem von uns angepassten AdaBoost gute und stark reduzierte Merkmalsmengen erhalten können, um darauf aufbauend mehr Varianten von Klassifikatoren auf den reduzierten Mengen zu trainieren und zu testen.

1.1 Aufbau dieser Arbeit

Im Kapitel 2.1 geben wir einen Überblick über Arbeiten, die die Kennzeichenerkennung und die Merkmals-Reduktion/-Auswahl zum Thema haben.

Kapitel 3 stellt die methodischen Grundlagen vor, die wir in dieser Arbeit nutzen werden. Das Ensemble-Learning und Boosting wird vorgestellt, dass aus mehreren einfachen Klassifikatoren (engl. WeakClassifier) einen deutlich verbesserten Klassifikator kombiniert. Nachfolgend werden die Verbesserungen von Viola und Jones [27, 39] vorgestellt. In ihren Arbeiten nutzen sie zur Erkennung für die einfachen Klassifikatoren Rechteck-Merkmale (sogenannte Haar-like-Feature oder auch Haar-Feature) und zeigen, dass diese mit einer speziellen Bildrepräsentation, dem sogenannten "Integral-Image", sehr effizient berechnet werden können. Außerdem führen sie als Klassifikator die Kaskade ein, eine Struktur aus Klassifikatoren, die den Klassifikationsprozess deutlich beschleunigt.

In Kapitel 3.7, 3.8 und 3.9 erläutern wir auf den Ergebnissen des vorigen Kapitels aufbauende Arbeiten. Erweiterungen und Alternativen zu dem von Viola und Jones vorgeschlagenden AdaBoost-Algorithmus werden in dem Kapitel 3.7 vorgestellt. Das nachfolgende Kapitel 3.8 beschäftigt sich mit Erweiterungen in den genutzten

Rechteck-Merkmalen, während wir in dem Kapitel 3.9 alternative Klassifikatoren vorstellen.

Mit der Möglichkeit den zeitintensiven Lernvorgang zu verkürzen, beschäftigt sich das Kapitel 4. Hier stellen wir das von uns, über zwei Vermutungen, leicht abgewandelte AdaBoost-Verfahren vor, mit dem die Menge der Klassifikatoren verringert werden kann.

Im Kapitel 7 werden die Ergebnisse und Vergleiche der Kaskaden vorgestellt, die zur Lokalisierung von Nummerschildern umgesetzt wurden.

In dem Kapitel 7.1 zeigen wir, dass selbst wenn wir dieselbe Anzahl an Merkmalen wählen, auf dessen Grundlage auch unserer "Viola-Jones-Klassifikator" eine umfassende Suche durchführt, vergleichbar gute Ergebnisse erzielt werden. Es zeigt sich allerdings auch, dass die Kaskaden auf der reduzierten Menge etwas mehr einfache Klassifikatoren innerhalb ihres Ensembles benötigen.

Letzteres ändert sich deutlich, sobald wir mehr Merkmale für die initiale Reduktion verwenden. Dadurch wird es möglich mit mehr Merkmalen in adäquater Zeit umzugehen und damit auch bessere Ergebnisse zu erreichen, was wir in 7.2 zeigen.

Darauf aufbauend vergleichen wir in dem Kapitel 7.3, ob sich auch andere Klassifikatoren durch das Verfahren auf gute und kleine Mengen reduzieren lassen. Wir werden vergleichen, wie und ob sich das Ergebnis mit diesen Klassifikatoren als Ersatz für die Schwellwert-Klassifikatoren verbessern läßt.

In dem Kapitel 7.4 werden wir diese Vergleiche mit den "Joint Haar-like Feature" wiederholen. "Joint Haar-like Feature" kombinieren selber Klassifikatoren, die dann als Merkmale für einen "Bayes"-Klassifikator dienen. Letzterer wird dann wieder über AdaBoost zu einem Ensemble kombiniert.

In manchen Bereichen sind die englischen Begrifflichkeiten die allgemein bekannteren und treffenderen und daher nutzen diese in Einzelfällen, ohne sie zu übersetzen. Auch setzen wir gewisse Kenntnisse der Bildverabeitung voraus.

Vergleichbare Arbeiten

2

2.1 Arbeiten zur Kennzeichenerkennung

Kennzeichenerkennungssysteme bestehen meist aus drei Komponenten:

1. Lokalisierung des Kennzeichens,
2. Segmentieren der einzelnen Zeichen
3. Erkennung der Zeichen

Anagnostopoulos et al. vergleichen in ihrer Arbeit [6] insbesondere alle diejenigen Lösungen, die dem oben genanntem drei-geteiltem Schema folgen und gibt einen Überblick über Arbeiten auf dem Gebiet.

Wir betrachten hier nur Arbeiten zu dem erstgenannten Punkt, der Kennzeichenlokalisierung.

In Kennzeichenregionen ist die Änderung der Helligkeitswerte eine der auffallendsten und herausragensten Eigenschaften. So versuchen viele Arbeiten diesen Aspekt herauszuarbeiten. Ein Ansatz führt über Kantendetektoren, die sich der Verfahren zur Kantenstatistik und mathematischer Morphologie bedienen. Allerdings reagieren diese recht empfindlich auf andere Kanten in der Umgebung, weshalb Kantendetektoren alleine meist ungenügend sind. Ein Ansatz diese ungewollten Kanten zu entfernen, ist vorweg morphologische Algorithmen auf das Bild anzuwenden. Häufig ist bei solchen Verfahren das Wissen um die Größe des gesuchten Objektes innerhalb des Bildes relevant für eine hohe Trefferquote.

In den Arbeiten von Hong [13] erreicht dieser mit seinem Ansatz der auf Kantenstatistik und Morpholgie aufsetzt, für ein Strassenmautsystem eine Erkennungrate von 99,6%. Der Abstand sowie Winkel zwischen Kamera und Wagen ist in diesem Szenario bekannt, was daher ein wichtiger Grund für die hohe Erkennungsrate ist.

Bei der “Connected Component Analysis” werden in einem bereits binarisiertem Bild alle Pixel in zusammengehörige Komponenten gruppiert. Für diese Komponenten werden dann verschiedene Eigenschaften ihrer Ausdehnung und Struktur berechnet. Diese Vermessungen der räumlichen Ausprägungen, wie Fläche/Größe, Orientierung und Form/Seitenverhältnis, der Komponenten werden dann herangezogen, um zu entscheiden, ob es sich um ein Kennzeichen handelt oder nicht.

Die Autoren Comelli et al. in [26] suchen in einem rechteckigen Fenster den Bereich mit dem größten lokalen Kontrast.

In den Arbeiten [3, 36, 5, 17, 16] verfolgen die Autoren einen vergleichbaren Ansatz, der nur partiell das Bild analysiert. Sie durchsuchen das Bild, in dem sie alle N Zeilen betrachten und dort die existierenden Kanten zählen. Liegen diese über einem bestimmten Schwellwert, so wird angenommen, dass es sich um ein Kennzeichenbereich handelt. Bei Broumandnia und Fathy [3] wird der Suchvorgang mit einem verringerten Schwellwert wiederholt, wenn kein Kennzeichen gefunden wurde. Da nur wenige Zeilen durchsucht werden, ist dieses Vorgehen sehr schnell. Allerdings ist es zu einfach, um in verschiedenen Szenarien gute Ergebnisse zu erbringen. Auch ist es nicht unabhängig von der Kameraentfernung zum Objekt.

Auf statistischen Messwerten richten Wang et al in ihrer Arbeit [11] die Aufmerksamkeit. Dort wird nach einem rechteckigen Bereich mit hohen Kantenwerten oder hohen Kantenvarianzen gesucht. Entsprechen die zusammenliegenden Bereiche gewissen geometrischen Kriterien, so werden diese als Kennzeichen betrachtet. Sie berichten in [11] bei 180 Bildern von einer Genauigkeit von 92,5%

In [44] wird eine Methode zur Kennzeichenerkennung vorgestellt, die auf Vektorquantifizierung (VQ) beruht.

Matas und Zimmermann stellten in [23] ein Verfahren zum Finden von Kennzeichen vor, in dem sie besonderen Wert auf robustes Verhalten legen. Sie forderten, dass das Verfahren nicht anfällig ist für Änderungen in der Größe, der Ausrichtung und affinen Deformationen. Auch sollte es mit Verdeckungen umgehen können. Sie nutzten hierfür die von Matas et al. in [18] vogestellte Methode der “Maximally Stable Extremal Region (MSER)”. MSER sind die Regionen in einem (Grauwert)Bild, die unter Anwendung aller möglichen Schwellwerte am stabilsten ihre geometrischen Formen bewahren. Mit einer Untermenge dieser zusammenhängenden Regionen und Methoden des maschinellen Lernens erreichen Matas und Zimmermann eine Trefferquote von 95%.

MSER wird auch von den Autoren Bischof und Donoser in [24] genutzt, um Kennzeichen zu erkennen. Sie unterscheiden zwei Varianten von MSER. Zum einen MSER+, das helle Regionen mit dunklen Grenzen erkennt und zum anderen MSER-, das dunkle Regionen mit hellen Grenzen erkennt. Sie stellten so fest, dass MSER+ eher das Schild als ganzes detektiert, während MSER- eher die Zeichen herausstellt. Ihr Ansatz ist die beiden Varianten zu analysieren und zu prüfen, wo größere MSER+ Regionen kleinere MSER- Regionen umfassen. Solche Regionen interpertierten sie als Kennzeichen. Auf Einzelbildern erreichten sie damit eine Erkennungsrate von etwas über 80% und in Sequenzbildern 94%.

Die Autoren Ho et al. in [40] trainierten AdaBoost mit den Basis HaarFeature von Viola und Jones. Ihre Intention war es ebenfalls Kennzeichen zu erkennen, ohne Einschränkungen in den Rahmenbedingungen (Licht, Lage etc.). Sie vermuteten deutliche Vorteile, wenn auf Kantenbildern gearbeitet wird. So haben sie die Bilder

mit einem Canny Edge Detektor vorverarbeitet. Auf ihren Testbildern landeten sie mit Grauwertbildern bei einer Detektionsrate von ca. 0.1. Bei Kantenbildern jedoch erreichten sie eine Detektionsrate von ca 0.8.

Chen und Yuille wollten in ihrer Arbeit [41] nicht speziell Kennzeichen finden sondern als Hilfe für sehbehinderte Menschen Schilder im Strassenverkehr erkennen. Sie nutzten auch einen Boosting Ansatz. Sie argumentierten, dass die Detektion von Text wenig Gemeinsamkeiten mit der Detektion von Gesichtern hat. Die räumliche Verteilung der Merkmale variiert bei Text sehr viel stärker als bei Gesichtern. Denn Augen und Nase sind immer an ähnlichen Positionen, während die Position, Verteilung und Größe von Buchstaben stark schwankt. Sie analysierten die Antworten von Ableitungsfiltern in x und y Richtung für jedes Pixel. Der Durchschnitt der Ableitungen zeigten dabei ein klares Muster. Die Ableitungen sind klein in den Hintergrundregionen unterhalb und oberhalb des Textes. Die x-Ableitungen tendieren dazu im Zentrum des Textes groß zu sein, während die y Ableitungen am Kopf und am Fuß des Textes groß sind und schmal im Zentrum des Textes. Aber die Varianz der x-Ableitung ist sehr groß innerhalb der zentralen Region, da Buchstaben unterschiedliche Form und Position haben. Die y-Ableitung tendiert dazu geringe Varianz zu haben und damit auch geringe Entropie. Sie erweiterten die Haar-Merkmale von Viola und Jones, um eben diese Gegebenheiten fassen zu können.

Für die Kennzeichenerkennung bauten Dlagnekov und Belongie [22] auf der Arbeit von Chen und Yuille [41] auf. Sie erstellen zusätzlich zu ihren 359 manuell ausgeschnittenen, weitere Positivbeispiele, in dem sie für jedes Positivbeispiel weitere 10 (mit zufälligem Offset von bis zu 1/8 der Weite und 1/4 der Höhe eines Ausschnitts) Beispiele ausgeschnitten wurden, so dass 3590 zusätzliche Positivbeispiele vorhanden waren. Sie erreichten eine Falsch-Positiv-Rate von 0,002% und eine Detektionsrate von 96,97% Darüber hinaus konnten sie ihre Falsch-Positiv-Rate noch weiter senken, in dem sie die nahe beieinander liegenden Treffer gruppiert hatten und solche Gruppen zurückwiesen, die nur wenige Treffer beinhalteten.

Zu der Arbeit Chen und Yuille [41], bemerkten die Autoren Zhang, Jia, He und Wu in [14], daß die bei Chen und Yuille genutzten statistischen Merkmale zu hohen Falsch-Positiv-Raten neigten, wenn sie überwiegend genutzt werden. Bei ihrem Ansatz benutzten sie sowohl globale statistische Merkmale als auch Haar-Merkmale. Sie berechneten die Varianz der sogenannten “Kanten-Dichte“, die sie zusätzlich zu den Haar-Merkmalen von Viola und Jones nutzten. Die Kanten-Dichte bezeichnet die Intensität innerhalb eines Ausschnittes, der mit einem Kantenfilter bearbeitet wurde. Um die Varianz der Kanten-Dichte zu erhalten wurde der betrachtete Ausschnitt in 12 gleich große Blöcke unterteilt. Für jeden Block wurde dann die Kanten-Dichte berechnet. Aus diesen Werten konnte schließlich die Varianz über die einzelnen Blöcke ermittelt werden. In ihrer Testmenge von 169

Kennzeichen in 160 Beispielbildern erreichten sie eine Detektionsrate von 96.4% und dabei nur eine Anzahl von insgesamt acht Falsch-Positiven.

2.2 Merkmalsreduktion, Merkmalsauswahl

Die Merkmalsreduktion/Merkmalsauswahl ist ein wichtiges Gebiet und wird von uns in den verschiedensten Bereichen untersucht. Sie hat zum Ziel gute Merkmale aus dem vorhandenen Material herauszusuchen bzw. bei der Merkmalsreduktion wird eine Untermenge der vorhandene Merkmale erstellt.

Ein wesentlicher Punkt für ein Ensemble ist die Verschiedenartigkeit der einzelnen Klassifikatoren. Wichtig ist die Fähigkeit möglichst viele unterschiedliche Aspekte des zu klassifizierenden Objektes beschreiben zu können.

In den meisten Fällen ergeben ähnliche Klassifikatoren auch ähnliche Resultate. Hat man also einen guten Klassifikator gefunden, so werden die Klassifikatoren, die zu dem ersten sehr ähnlich sind auch "nur" ähnlich gute Fehlerraten erzeugen und vor allem auf ähnliche Aspekte des gesuchten Objektes passen.

Die Merkmalsauswahl soll die Eigenschaft haben, relevante Merkmale zu finden und irrelevante und redundante Merkmale zu entfernen, so dass auf der gewonnenen Datenmenge ein robustes Lernen möglich wird.

Merkmalsreduktion ist zum einen wichtig wegen dem "Fluch der Komplexität" und zum anderen wegen der aufkommenden Datenmenge und Datenkomplexität durch Disziplinen wie z.B. "Machine Learning", "Pattern Recognition", Bioinformatik und "Data Mining".

Es gibt sehr viele verschiedene Ansätze mit denen gute Lösungen gefunden wurden.

Das Wählen einer zufälligen Untermenge von Merkmalen wird als "Random Subspace Method" bezeichnet. Jeder Klassifikator in einem zu erstellendem Ensemble wird dabei aus einer Menge der Merkmale zufällig ausgewählt. (vergleiche [12], [20])

Korrelationen zwischen Objektklassen und Merkmalswerten nutzen Oza und Tumer für ihre Methode die sie "Input Decimation" nannten und in ihrer Arbeit [25] vorstellten.

Es gibt viele unterschiedliche Ansätze Genetische Algorithmen zur Auswahl zu verwenden (vergl. [10], [20]).

In [31] nutzen die Autoren Redpath und Lebart das AdaBoost-Verfahreb in der "Soft Margin" Variante von Rätsch et al. [32] und "Floating Search", um Merkmale für die Klassifikatoren eines Ensembles auszuwählen. AdaBoost kombiniert dabei die Klassifikatoren und über das "Floating Search" und das "Soft Margin" Kriterium werden die Merkmale für einen Klassifikator bestimmt.

Die Komponenten des Erkennungsverfahrens

3

In diesem Kapitel stellen wir die Komponenten des Erkennungsverfahrens vor und die Aufgabe, die diese übernehmen.

2001 stellten Viola und Jones in ihren Arbeiten [27, 39] ein Verfahren zum Finden von Gesichtern in Grauwertbildern vor. Das Erkennungsverfahrens von Viola und Jones hob sich von anderen bis dahin üblichen Verfahren, durch seine sehr hohe Erkennungsgeschwindigkeit und geringe Falsch-Positiv-Rate ab.

Ein wesentlicher Bestandteil des Nachfolgenden ist das Ensemble-Lernen und hier speziell der Algorithmus AdaBoost (Abschnitt 3.1). Das Ensemble-Lernen beinhaltet, wie aus mehreren Klassifikatoren ein neuer verbesserter Klassifikator kombiniert werden kann.

Schapire gibt in seiner Arbeit [34] eine Einführung in die Boosting-Algorithmen. Er schreibt zu der Entwicklung des Boostings, dass er 1989 in seiner Arbeit [33] den ersten beweisbaren polynomialzeit Boosting-Algorithmus vorgestellt hat. Ein Jahr später stellte Freund in [8] eine verbesserte Variante vor und zusammen veröffentlichten sie 1995 den AdaBoost-Algorithmus [9].

Die Bestandteile des Ensemble-Klassifikators werden häufig auch als "einfache Klassifikatoren" (engl. WeakClassifier) bezeichnet. Viola und Jones nutzten für ihre Implementierung zur Gesichtserkennung einen Schwellwert-Klassifikator (siehe Kapitel 3.2). Dieser hat den Vorteil, dass er leicht umgesetzt werden kann und, sowohl während des Trainingsprozesses als auch während des Klassifikationprozesses, effizient berechenbar ist.

Viola und Jones schreiben in ihrer Arbeit [27], dass ihr Erkennungssystem, motiviert durch die Arbeiten von Papageorgiou et al. [4], nicht direkt auf Pixelwerten arbeitet. Wie bei Papageorgiou et al. nutzen sie eine Menge von "Haar-Merkmalen", die den Haar Basisfunktionen entlehnt sind.

Nach eben diesen Haar-Merkmalen, die einen besonderen Anteil an der hohen Geschwindigkeit des gesamten Erkennungssystems haben, treffen die Schwellwert-

Klassifikatoren ihre Entscheidung. Jeder Schwellwert-Klassifikator benutzt dafür genau eine Haar-Merkmal. Wie in Kapitel 3.3 näher erläutert sind diese Haar-Merkmale (engl. Haar-Feature oder Haar like Feature) Differenzen der Mittelwerte von Rechteckflächen.

Dass diese Haar-Merkmale so effizient berechenbar sind, begründet sich in der Einführung einer besonderen Art der Bildrepräsentation, dem sogenannten "Integral-Image", das wir in 3.4 näher erläutern werden. Das Integral-Image, schreiben Viola und Jones, ist sehr ähnlich zu dem "Summed Area Table (SAT)", der in der Arbeit von Crow [7] für Texturmapping genutzt wird.

Das Ganze ist eingebettet in eine weitere Struktur, der sogenannten "Kaskade", die die Erkennungsgeschwindigkeit noch weiter erhöht. Viola und Jones schrieben, dass diese Struktur im wesentlichen einen "degenerierten Entscheidungsbaum" darstellt, wie er in den Arbeiten von Amit und Geman [42] vorgestellt wurde.

Mehrere Ensemble-Klassifikatoren bilden die Bestandteile einer solchen Kaskade. Sie zeichnet sich durch die Fähigkeit einen großen Teil des Hintergrundes frühzeitig und mit wenig Rechenaufwand als solchen zu erkennen aus. Nur bei schwierigen Ausschnitten prüft der Kaskaden-Klassifikator kritisch und mit mehr Rechenaufwand.

Um das gesuchte Objekt in einem vorgegeben Bild zu finden, wird in verschiedenen Skalierungen ein Fenster über das zu untersuchende Bild geschoben. In diesen Bildrahmen wird für jeden Teil des Bildes geprüft, ob sich innerhalb des Fensters das gesuchte Objekt befindet.

Ein sonst übliches Verfahren ein Objekt in einem Bild in unterschiedlichen Größen zu finden, ist es das Bild selbst in unterschiedlichen Größen zu skalieren und dann, mit der jeweils gleichen Ausschnittsgröße, das entsprechende Bild zu analysieren. Der Aufwand hierfür ist hoch, da die Skalierung, je nach Größe des Bildes, aufwendig ist.

Dass wir keine Bilder neu skalieren, sondern lediglich Fenster in verschiedenen Skalierungen über das Bild legen müssen, ist der Art und Weise gedankt, wie wir Haar-Merkmale berechnen können, was wir in Abschnitt 3.3 und 3.4 näher betrachten werden.

3.1 AdaBoost

Beim Ensemble-Lernen geht es um Verfahren, die zum Ziel haben, eine Menge von Klassifikatoren zu verbessern, indem diese zu einem einzigen Klassifikator zusammengefügt werden. AdaBoost trifft dabei einen gewichteten Mehrheitsentscheid über die einzelnen Klassifikatoren. Diese einzelnen Klassifikatoren werden in dem Zusammenhang auch häufig als "einfache" Klassifikatoren (engl. WeakClassifier) bezeichnet.

Beim Boosting wird die Klassifikationsentscheidung des Ensembles unter Beachtung aller Einzelergebnisse der beteiligten Klassifikatoren getroffen. Der Grundgedanke ist, dass Viele gemeinsam eine bessere Entscheidung treffen als Einer alleine. Wenn man die einzelnen Klassifikatoren als Experten betrachtet kann das Boosting mit einem Expertenrat verglichen werden, der über das endgültige Resultat abstimmt.

Die einfachen Klassifikatoren, derer sich AdaBoost bedient, liefern "boolsche" Werte, also einfache Ja/Nein-Entscheidungen. Liegt diesen der zu bewertende Bildausschnitt vor, so geben sie eine Entscheidung darüber ab, ob sich in dem Ausschnitt das gesuchte Objekt befindet oder nicht befindet.

AdaBoost ermittelt das Ergebnis der Klassifikation, in dem die Entscheidungen der einfachen Klassifikatoren ausgezählt werden. Die Mehrheit bestimmt das Ergebnis. Wobei AdaBoost den einzelnen Klassifikatoren eine Gewichtung hinzufügt. Klassifikatoren die besser bewertet sind, haben auch mehr Einfluss auf die finale Entscheidung.

Formal ausgedrückt:

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

α_t ist hierbei die Gewichtung und $h_t(x)$ das Ergebnis des $t$ten einfachen Klassifikators.

An den einfachen Klassifikiator wird für den AdaBoost-Algorithmus lediglich die Anforderung gestellt, dass er besser sein soll als der Zufall, in Zahlen also nicht größer als 0.5.

Viola und Jones schreiben: "Our hypothesis, which is borne out by experiment, is that a very small number of these features can be combined to form an effective classifier. The main challenge is to find these features."

Bisher haben wir uns angeschaut, wie ein mit AdaBoost trainiertes Ensemble eine Entscheidung trifft. Jetzt betrachten wir, wie AdaBoost diese kleine Menge an Merkmalen findet.

Wichtig für den Lernalgorithmus AdaBoost ist die Gewichtung der Trainingsbilder. Jedes Trainingsbild erhält einen eigenen Gewichtungsfaktor. Die Summe der Gewichtsfaktoren über alle Trainingsbilder sollen dabei stets 1 ergeben, so dass die

Gewichte einer statistischen Verteilung genügen. Die Objekte, die zu finden AdaBoost trainiert wird, bestehen meist aus unterschiedlichen Eigenschaften, wie z.B. der häufige Wechsel der Intensität durch die Zeichen. Diese unterliegen meist unterschiedlicher Ausgeprägung, wie z.B. die unterschiedlichen Lichtverhältnisse in den verschiedenen Bildern. Zum einen sollen möglichst viele Eigenschaften gelernt werden und zum anderen diese so gelernt werden, dass sie auch in den verschiedenen Ausprägungen in den Bildern erkannt werden. Das Ziel ist es, für möglichst viele Eigenschaften und Ausprägungen einen Klassifikator zu finden, der diese beschreibt, um sich in der Summe auf möglichst vielfältige und unabhängige Entscheidungen stützen zu können. Dass möglichst viele dieser Eigenschaften und Ausprägungen gelernt werden, wird durch die Gewichtsfaktoren der Trainingsbilder erreicht.

Zu Beginn werden die Gewichte jeweils für Positive und Negative Beispiele gleich über die Anzahl der Trainingsbilder berechnet. $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ für $y_i = 0, 1$, wobei m die Anzal der negative Beispielbilder ist und l die Anzahl der positiven Beispielbilder.

Während jedes Trainingsdurchlaufs wird derjenige Klassifikator ausgewählt, der den geringsten gewichteten Fehler erreicht.

Der Lernalgorithmus für die einzelnen Klassifikatoren, die durch AdaBoost dem Ensemble zugefügt werden sollen, ist im allgemeinen auch in der Lage die Gewichte auf den Trainingsbildern zu berücksichtigen.

Ein Klassifikator, der auf einem bestimmten Merkmal beruht, kann also in verschiedenen Durchgängen auf den gleichen Trainingsbildern verschiedene Ausprägungen annehmen. Sein absoluter Fehler kann hier durchaus größer sein als bei einem Klassifikator der nicht gewählt wurde, der also einen schlechteren gewichteten Fehler hat.

Nachdem ein Klassifikator gewählt wurde, werden die Gewichte so verändert, dass diejenigen Beispiele, die falsch klassifiziert wurden stärker hervorgehoben werden. Der als nächstes gewählte Klassifikator soll den Bildern eine größere Beachtung schenken, die bisher falsch klassifiert wurden. Deshalb werden die Gewichte der korrekt klassifizierten Beispiele entsprechend des Fehlers des zuletzt gewählten Klassifikators verringert. Abschließend werden die Gewichte normalisiert, so dass diese in der Summe über alle Beispiele wieder 1 ergeben und damit wieder einer statistischen Verteilung genügen.

So soll gewährleistet werden, dass im Laufe der Durchgänge jede Eigenschaft des gesuchten Objektes “seinen“ guten Klassifikator erhält.

Zu jedem Klassifikator wird ein Faktor α mitgespeichert, der seine ”Güte“ wiedergibt. Diese Güte wird über den gewichteten Fehler des betrachteten Klassifikators berechnet.

Freund und Schapire stellten fest, dass der Trainingsfehler des Ensembles exponentiell mit der Anzahl der Runden gegen 0 strebt. Viola und Jones [27] schreiben, dass der Schlüssel zur Generalisierungsfähigkeit abhängig ist von dem Abstand den die

- Given example images $(x_1, y_1)...(x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- for $t = 1, ..., T$:
  1. Normalize the weights, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$ so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i \mid h_j(x_i) - y_i \mid$.
  3. Choose the classifier, h_t with the lowest error ϵ_j.
  4. Update the weights: $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$ where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is: $h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$ where $\alpha_t = log\frac{1}{\beta_t}$.

Tabelle 3.1: Der AdaBoost Algorithmus zur Erstellung eines Ensemble-Klassifikators nach [39, 27]

Entscheidungskurve zu den Trainingsbeispielen erreicht. AdaBoost erreicht schnell große Abstände [29].

3.2 Schwellwert-Klassifikator

Als einfachen Klassifikator benutzen Viola und Jones einen boolschen Schwellwert-Klassifikator. Dieser Klassifikator teilt eine Menge in zwei Klassen. Werte die größer als ein gewisser Schwellwert sind, werden der einen Klasse zugeordnet und solche Werte, die kleiner als der Schwellwert sind, werden der anderen Klasse zugeordnet.

Wie im Kapitel 3.3 über Haar-Merkmale erläutert, handelt es sich bei einem Haar-Merkmal um eine Gruppierung von Rechteck-Merkmalen, die einem Filter ähnlich dem Bereich, den sie in einem Bild einnehmen, einen Wert zuordnen. Die Schwellwert-Klassifikatoren von Viola und Jones nutzten als Eingangswerte den Wert genau eines Haar-Merkmales. Entsprechend des Wertes des gerade genutzten Haar-Merkmales und des Schwellwertes entscheidet der Klassifikator, ob es sich um das gesuchte Objekt handelt oder um Hintergrund.

Die Trainingsbilder erhalten durch den Boosting-Algorithmus unterschiedliche Gewichtungen, wie in Kapitel zu AdaBoost 3.1 dargelegt. Der Schwellwert-Klassifikator

soll nun so trainiert werden, dass dieser unter Beachtung der aktuellen Gewichte auf den Trainingsbildern die Anzahl der Fehlklassifikationen minimiert.

Für das Training eines Schwellwert-Klassifikators muss der Wert des aktuell betrachteten Haar-Merkmales an einer festen Stelle für alle Bilder der Trainingsmenge berechnet werden. Aus diesen Werten wird dann der Schwellwert so bestimmt, dass die positiven und negativen Trainingsbeispiele am besten getrennt werden, dass also der Schwellwert so berechnet wird, dass er unter Brücksichtigung der aktuellen Trainingsgewichte den Fehler minimiert.

Ein solcher Schwellwert-Klassifikator $h_j(x)$ wird zusammengestzt aus einem Haar-Merkmal f_j, einem Schwellwert θ_j und einer Parität p_j, die die Richtung der Ungleichung beschreibt.

$$h_j(x) = \begin{cases} 1 & p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

Viola und Jones nennen in [27] Fehlerraten von 0.1 bis 0.3 für die Schwellwert-Klassifikatoren, die früh beim Boosten zugefügt wurden und Fehlerraten von 0.4 bis 0.5 bei später zugefügten Schwellwert-Klassifikatoren.

3.3 Haar-Merkmale

Viola und Jones benutzten als Erkennungsmerkmal in den Bildern Haar-Merkmale (engl. “Haar Like Feature“ oder auch ”HaarFeature“). Diese berechnen im wesentlichen die Grauwertsumme innerhalb eines Rechteckes und setzen sie mit anderen Rechtecksummen in Beziehung. In der Arbeit von Viola und Jones und den anderen uns bekannten Arbeiten wird die Rechtecksumme des einen Teils von der Rechtecksumme des anderen Teils abgezogen. Diese gebildete Differenz ist der Ausgangspunkt der weiteren Berechnungen. Viola und Jones nutzten verschiedene Typen als Basis Haar-Merkmale, wie sie beispielhaft in Abbildung 3.4 auf Seite 16 dargestellt sind.

Die Merkmale werden als Maske über das zu betrachtende Bild geschoben und an allen Stellen der entsprechende Wert des Merkmales berechnet. Die Anzahl aller Merkmale ist sehr groß, da die Basis Haar-Merkmale im Rahmen der betrachteten Bildgröße beliebig positioniert und skaliert werden. Damit gibt es für ein Bild erheblich mehr Haar-Merkmale als Pixel.

3.4 Integralimage

Ein Teil der hohen Verarbeitungsgeschwindigkeit basiert auf einer von Viola und Jones neu eingeführten Repräsentation der Bilder, die sie Integral-Image nannten.

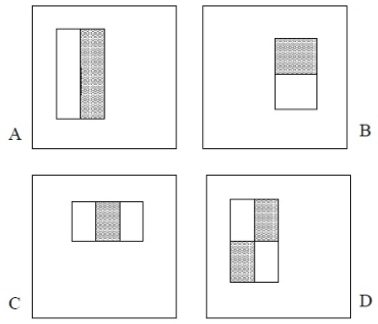


Abbildung 3.1: Basis Haar-Merkmale nach Viola und Jones. Die Summe der Pixelwerte in den grau schraffierten Rechtecken werden von den Pixelsummen der weißen Rechtecke abgezogen.

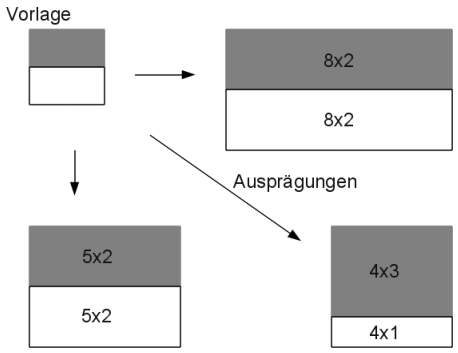


Abbildung 3.2: Generierung der konkreten Ausprägungen mit einer festen Höhe und einer festen Breite nach einer Vorlage.

Der andere Teil, der zu der hohen Verarbeitungsgeschwindigkeit führt, liegt in der weiter unten genauer vorgestellten Kaskadenstruktur des finalen Klassifikators.

Durch dieses Integral-Image können Rechteckmerkmale, aus denen die Haar-Merkmale bestehen, sehr effizient berechnet werden.

Das Integral-Image wird berechnet, indem die Pixelwerte innerhalb eines Bildes summiert werden. Der Wert des Integral-Images an einer bestimmten Position ergibt sich aus der Summe aller Pixelwerte oberhalb und links des betrachteten Punktes.

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'), \tag{3.2}$$

Wobei $ii(x, y)$ das Integral-Image ist und $i(x', y')$ das Originalbild darstellt.

Ist das Integral-Image berechnet, kann mit lediglich vier Zugriffen die Summe aller Pixelwerte innerhalb eines Rechteckes berechnet werden.

Das Integral-Image selbst kann in einem Durchgang mit folgenden Formeln berech-

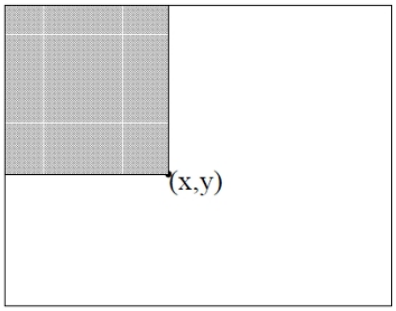


Abbildung 3.3: Der Wert des IntegralImages am Punkt (x, y) ist die Summe aller Pixelwerte innerhalb der grauen Fläche.

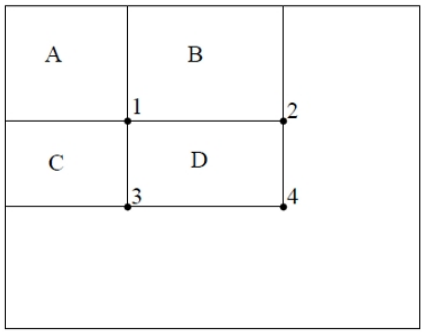


Abbildung 3.4: Der Wert eines Rechtecks kann mit dem Integral-Image aus vier Punkten berechnet werden. Der Wert des Integral-Images am Punkt 1 ist die Summe der Pixelwerte im Rechteck A. Der Wert an Punkt 2 ist $A + B$, an Punkt 3 $A + C$ und an Punkt 4 $A+B+C+D$. Somit kann die Summe von D wie folgt berechnet werden: $D = 4+1-(3+2)$

net werden.

$$s(x, y) = s(x, y - 1) + i(x, y) \tag{3.3}$$

$$ii(x, y) = ii(x - 1, y) + s(x, y) \tag{3.4}$$

$s(x, y)$ ist hierbei die Zeilensumme, $s(x, -1) = 0$ und $ii(-1, y) = 0$

3.5 Kaskade

Objekte in einem Bild zu suchen, in dem ein Fenster in kleinen Schritten über das zu untersuchende Bild geschoben wird, ist sehr rechenintensiv. Schließlich muss für sehr viele Positionen geprüft werden, ob sich das gesuchte Objekt in dem gerade betrachteten Ausschnitt befindet.

Häufig ist bei dem größeren Teil des Bildes leicht zu entscheiden ob es Hintergrund ist und nur bei dem geringeren Teil ist es notwendig, den Ausschnitt einer genaueren Analyse zu unterziehen. Da die meisten Bilder aus deutlich mehr Hintergrund

bestehen als aus dem gesuchten Objekt, wäre ein Klassifikator wünschenswert, der mit wenig Rechenaufwand feststellen kann, dass es sich nicht um das gesuchte Objekt handelt. Aber trotzdem gründlich in den Regionen ist, die das gesuchte Objekt enthalten.

Für diesen Zweck nutzen Viola und Jones eine Struktur, die sie, begzugnehmend auf Amit und Geman [42], als degenerierten Entscheidungsbaum beschrieben und "Kaskade" nannten.

Jeder Knoten dieses Entscheidungsbaums hat einen binären Klassifikator und genau einen Nachfolger. In jedem dieser Knoten wird entschieden, ob es sich bei dem betrachteten Ausschnitt um Hintergrund handelt oder um das gesuchte Objekt.

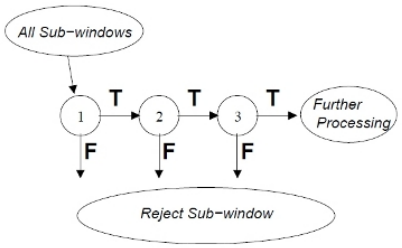


Abbildung 3.5: Kaskadenstruktur nach Viola/Jones.

Wird entschieden, dass es sich um Hintergrund handelt, wird die Kaskade verlassen und der Ausschnitt als negativ klassifiziert. Der Klassifikationsprozess ist damit an dieser Stelle beendet.

Wird jedoch entschieden, dass es sich um das gesucht Objekt handelt wird der Ausschnitt dem nachfolgenden Knoten zur Klassifikation übergeben. Nur wenn bis zum Ende der Kaskade an keinem Knoten ein Klassifikator für Hintergrund votiert, wird der Ausschnitt als das gesuchte Objekt klassifiziert. Das entscheidende Element dieser Kaskade ist, dass frühe Knoten wenig komplexe und schnelle Klassifikatoren beinhalten. Je weiter hinten ein Klassifikator angesiedelt ist, desto komplexer wird er.

Bei Viola/Jones befinden sich in den Knoten der Kaskade Ensemble-Klassifikatoren, die mit AdaBoost aus Schwellwert-Klassifikatoren trainiert wurden. Um nun den Geschwindigkeitsvorteil zu erreichen, sind diese Ensemble-Klassifikatoren der ersten Knoten aus nur wenigen Schwellwert-Klassifikatoren zusammengesetzt. Im weiteren Verlauf der Kaskade wird die Komplexität der Ensemble-Klassifikatoren immer weiter erhöht, was hier bedeutet, dass die Anzahl der Schwellwert-Klassifikatoren innerhalb eines Ensembles immer weiter erhöht wird.

Die Ensemble-Klassifikatoren werden so traininiert, dass sie eine sehr hohe Detektionsrate haben, was bei einfachen Klassifikatoren meist aber auch eine hohe Falsch-Positiv-Rate zur Folge hat. Diese hohe Anzahl falsch positiver Entscheidungen der

frühen Knoten wird jedoch für die hohe Detektionsrate in Kauf genommen, da erstere durch die nachfolgenden Knoten immer weiter reduziert wird. Auf diese Weise soll an jedem Knoten sichergestellt werden, dass frühzeitig Hintergrund rausgefiltert wird, aber mit großer Sicherheit positive Objekte weitergereicht werden. Wird also frühzeitig Hintergrund durch den Ensemble-Klassifikator mit geringer Komplexität als solcher erkannt, kann die Analyse des Bildausschnitts mit vergleichsweise wenig Rechenaufwand beendet werden.

Wie in 3.1 dargestellt besitzt der Ensemble-Klassifikator den AdaBoost erstellt einen Schwellwert. Liegt die gewichtete Summe der Ensemble-Mitglieder über diesem Schwellwert, so klassifiziert der Ensemble-Klassifikator positiv. Um die gewünschte hohe Detektionsrate zu erhalten, wird der Schwellwert des Ensembles gesenkt. Begrenzt ist dies nur durch den Wert der Falsch-Positiv-Rate die wir bereit sind, in Kauf zu nehmen.

So erhält man schon zu einem Zeitpunkt an dem noch nicht viele einfache Klassifikatoren dem Ensemble hinzugefügt wurden einen Klassifikator, der zwar eine hohe Falsch-Positiv-Rate aber auch eine sehr hohe Detektions-Rate hat.

Die gesamte Detektionsrate D und die gesamte Falsch-Positiv-Rate F kann dabei wie folgt abgeschätzt werden.

$$D = \prod_{i=1}^{K} d_i \tag{3.5}$$

$$F = \prod_{i=1}^{K} f_i \tag{3.6}$$

f_i ist dabei die Falsch-Positiv-Rate des Klassifikators im i-ten Knoten der Kaskade und d_i die Detektionsrate des i-ten Klassifikators in der Kaskade. K ist die Gesamtanzahl der Kaskadenknoten.

Für das Training der Kaskade wird für jeden Knoten die minimale Detektionsrate und die maximale Falsch-Positiv-Rate vorgegeben. Wie oben beschrieben, wird zusätzlich geprüft, ob durch die Senkung des Schwellwertes des Ensemble-Klassifikators diese Vorgaben erreicht werden können.

3.6 Bootstrapping

Das “Bootstrapping” (siehe [35]) bezeichnet ein Verfahren, das es erlaubt die Anzahl der negativen Trainingsbeispiele erheblich zu erhöhen, ohne Einbußen bei der Geschwindigkeit des Verfahrens in Kauf zu nehmen. Eingesetzt wird es innerhalb des Kaskaden-Lernens wie folgt: Wurde ein Ensemble für einen Kaskaden-Knoten

gelernt, so werden alle diejenigen negativen Beispiele aus den Trainingsbildern entfernt, die korrekt als Hintergrund erkannt wurden. An deren Stelle kommen neue Negativ-Beispiele aus der Menge der Trainingsbilder. Da jeder Kaskaden-Knoten nur diejenigen Ausschnitte zur Klassifikation erhält, die der Vorgänger nicht verworfen hat, werden auf die Gesamtkaskade betrachtet erheblich mehr Negativbeispiele gelernt, als man mit einer unveränderlichen Menge lernen würde. Die positiven Beispielbilder werden nicht geändert.

3.7 Erweiterungen und Alternativen zu AdaBoost

Wir möchten an dieser Stelle auf drei Erweiterungen bzw Alternativen zu dem Boosting-Algorithmus AdaBoost eingehen.

Die erste Variante, Asymmetric AdaBoost, unterscheidet sich zu AdaBoost lediglich dadurch, dass die positiven Beispiele durch eine Änderung der Gewichtung stärker betrachtet werden als die negativen Beispiele (3.7.1).

RealBoost der Autoren Wu, Ai und Huang [2] erwähnen wir hier nur der Vollständigkeit halber, um eine Boosting-Variante vorzustellen, die viel genutzt wird. Im Gegensatz zu AdaBoost aber nicht mit binären Klassifikatoren arbeitet, sondern mit solchen, die kontinuierliche Entscheidungswerte liefern (siehe 3.7.2).

Als letzte Variante stellen wir das “Direct Feature Selection” von den Autoren Wu, Rehg und Mullin [19] vor, die sich dem Ensemble-Lernen auf eine gänzlich andere Weise nähern als Viola und Jones.

3.7.1 Asymmetric AdaBoost

2002 stellten Viola und Jones in ihrer Arbeit [28] eine veränderte Variante des AdaBoost Algorithmus vor: Asymmetric AdaBoost.

Asymmetric AdaBoost arbeitet genauso wie AdaBoost, so wie wir es im Abschnitt 3.1 beschrieben haben.

Bei Asymmetric AdaBoost kommt hinzu, dass eine unterschiedliche (asymmetrische) Gewichtung der positiven und der negativen Trainingsbeispiele vorgenommen wird.

Viola und Jones führen zu diesem Zweck einen Faktor $\sqrt{k}$ bzw. $\frac{1}{\sqrt{k}}$ ein, der in die Gewichtung der Trainingsbeispiele mit einfließt.

Da AdaBoost diese asymmetrische Gewichtung in wenigen Runden wieder glätten und damit unwirksam machen würde, muss diese Änderung der Gewichtung in jeder Runde wiederholt werden.

Damit bei steigender Rundenzahl auch die Falsch-Positiven Entscheidungen reduziert werden, soll der Einfluss im Laufe der Trainingsrunden geringer ausfallen, so dass Asymmetric AdaBoost sich in späten Runden wie AdaBoost verhält.

Dazu werden in jeder Runde die Trainingsbeispiele mit

$$(\sqrt[2T]{k})^{y_i} \tag{3.7}$$

multipliziert. Wobei T die aktuelle Trainingsrunde ist und y_i die Werte -1 für negative Beispielbilder und $+1$ für positive Beispielbilder annehmen kann.

3.7.2 RealBoost

Wu et al. geben in [2] als Hauptnachteil des Schwellwert-Klassifikators an, dass dieser zu einfach ist, um sich an komplexe Verteilungen anzupassen.

Sie wollten einen Klassifikator für das AdaBoost nutzen, der kontinuierliche Werte anstatt der boolschen Werte lieferte.

Den Algorithmus zum Trainieren dieser nicht-boolschen Klassifikatoren nannten sie Realboost und nutzten einen Binning-Klassifikator, ähnlich dem den wir in Kapitel 3.9.1 vorstellen. Der Rückgabewert des von ihnen genutzten Binning-Klassifikators stellt eine Funktion des Quotienten der positiven und negativen Trefferwahrscheinlichkeiten dar.

Realboost-Ensembles treffen, ebenso wie AdaBoost, ihre Entscheidung über einen gewichteten Mehrheitsentscheid. Im Gegensatz zu AdaBoost werden die Entscheidungsgewichte der einzelnen Klassifikatoren durch ihren jeweiligen Rückgabewert geregelt. Je besser also der Rückgabewert des einzelnen Klassifikators, je sicherer sich der Klassifikator also über seine Entscheidung ist, desto stärker beeinflusst seine Entscheidung die Gesamtentscheidung. Die $\alpha h(x)$ beim AdaBoost entfallen hier und werden zu:

$$H(x) = sign(\sum_{t=1}^{T} h_t(x)) \tag{3.8}$$

3.7.3 Direct Feature Selection

Die Autoren Wu, Rehg und Mullin [19] präsentieren in ihrer Arbeit die Methode "Direct Forward Feature Selection", um einen Ensemble-Klassifikator zu trainieren. Die Kaskadenstruktur ist diejenige, wie von Viola/Jones vorgeschlagen und das Bootstraping wie in [35].

Als Ausgangspunkt stellen sie die Frage, was genau die Rolle des Boostings ist. Ihre Hyphothese ist, dass der Erfolg des Boostings an einer ausreichend großen Merkmalsmenge hängt.

Diesem Gedanken folgend, verzichten sie auf eine Gewichtung der Trainingsbeispiele. Dementsprechend werden die Klassifikatoren, die zu dem Ensemble kombiniert

werden sollen, nur einmal zu Beginn der Prozedur trainiert. Es wird dann in jeder Runde in der das Ensemble erweitert wird, der Klassifikator ausgewählt, der das Ergebnis des Ensembles am stärksten verbessert, also die Fehlerrate am meisten senkt.

1. Given a training set. Given d, the minimum detection rate and f, the maximum false positive rate.
2. For every feature, j, train a weak classifier h_j , whose false positive rate is f.
3. Initialize the ensemble H to an empty set, i.e. $H \leftarrow \phi$. $t \leftarrow 0$, $d_0 = 0 : 0$, $f_0 = 1 : 0$.
4. while $d_t < d$ or $f_t > f$
   a) if $d_t < d$, then, find the feature k, such that by adding it to H, the new ensemble will have largest detection rate d_{t+1}.
   b) else, find the feature k, such that by adding it to H, the new ensemble will have smallest false positive rate f_{t+1}.
   c) $t \leftarrow t + 1$, $H \leftarrow H \cup \{h_k\}$.
5. The decision of the ensemble classifier is formed by a majority voting of weak classifiers in H, i.e. $H(x) = \begin{cases} 1 & \sum_{h_j \in H} h_j(x) \geq \phi \\ 0 & \text{otherwise} \end{cases}$, where $\phi = \frac{T}{2}$. Decrease ϕ if necessary.

Tabelle 3.2: Direct Feature Selection Methode zur Erstellung eines Ensemble-Klassifikators der Autoren Wu, Rehg und Mullin [19]

3.8 Erweiterungen der Haar-Merkmale

Seit Viola und Jones ihre Basistypen der Haar-Merkmale vorgestellt haben, wurden viele weitere Varianten anderer Formen von Haar-Merkmalen vorgeschlagen, die wir in diesem Kapitel kurz vorstellen.

3.8.1 Diagonale Haar-Merkmale

Viola und Jones stellten in ihrer Arbeit [38] fest, dass die von ihnen genutzten Basis Haar-Merkmale nicht in der Lage sind, Gesichter die nicht frontal und aufrecht aufgenommen sind, mit ausreichender Genauigkeit zu erkennen. Um mit solchen Gesichtsprofilen besser umgehen zu können und um die Blickrichtung abschätzen zu können, führten sie diagonale Haar-Merkmale ein, wie sie in der Grafik 3.6 abgebildet sind.

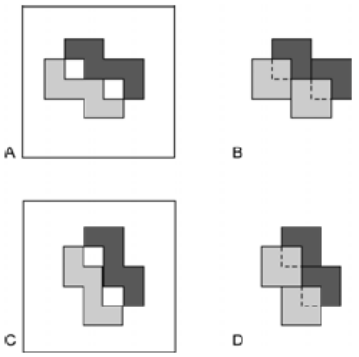


Abbildung 3.6: Diagonale Haar-Merkmale nach Viola und Jones, zur Abschätzung der Blickrichtung

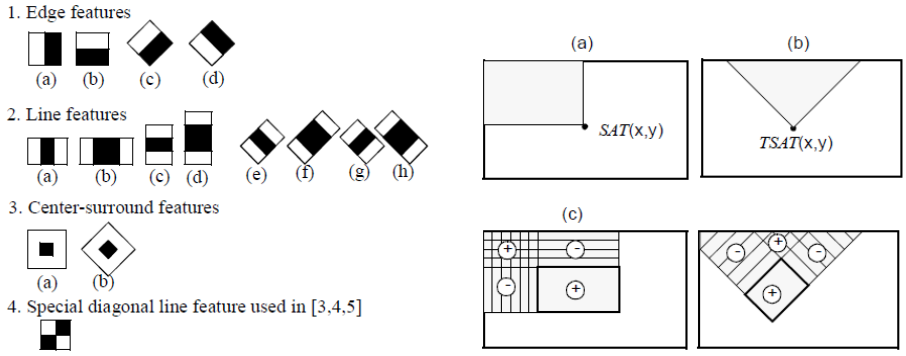


Abbildung 3.7: Die neue Haar-Merkmale von Lienhart et al. und die Gegenüberstellung des IntegralImages nach Violoa/Jones und des rotierten IntegralImages nach Lienhart et al..

3.8.2 45 Grad gedreht nach Lienhart

Um 45 Grad gedrehte Haar-Merkmale stellen die Autoren Lienhart et al. in ihrer Arbeit [30] vor. Sie berechnen analog zu dem Integral-Image von Viola und Jones ein rotiertes IntegralImage, so dass die gedrehten Haar-Merkmale auch mit lediglich vier Zugriffen berechnet werden können (siehe Grafik 3.7).

$$rsat(x,y) = \sum_{y' \leq y, y' \leq y - |x - x'|} i(x', y'), \tag{3.9}$$

Wobei $rsat(x,y)$ das rotierte Integral-Image ist und $i(x',y')$ das Originalbild darstellt.

Ist das Integral-Image berechnet, kann mit lediglich vier zugriffen die Summe aller Pixelwerte innerhalb eines Rechteckes berechnet werden.

Das rotierte Integral-Image kann in einem Durchgang mit folgender Formel berechnet werden.

$$rsat(x,y) = rsat(x-1,y-1)+rsat(x+1,y-1)-rsat(x,y-2)+i(x,y)+i(x,y-1) \tag{3.10}$$

wobei gilt
$rsat(-1,y) = rsat(x,-1) = rsat(x,-2) = rsat(-1,-1) = rsat(-1-2) = 0$

Berechnet wird die Rechtecksumme dann über folgende Formel:

$$rsum(r) = rsat(x–h+w, y+w+h–1)+rsat(x, y–1)–rsat(x–h, y+h–1)–rsat(x+w, y+w–1) \tag{3.11}$$

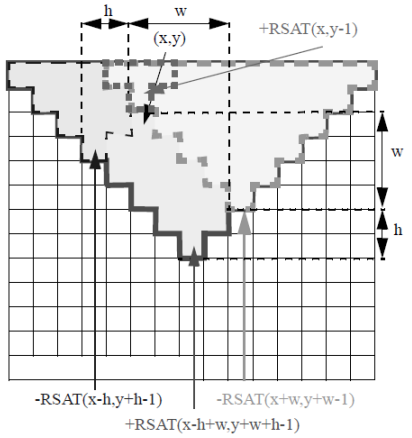


Abbildung 3.8: Berechnung der Rechteckfläche des rotierten Integral-Images nach Lienhart et al. $RecSum(r) = rsat(x–h+w, y+w+h–1)+rsat(x, y–1)-rsat(x–h, y+h–1)-rsat(x+w, y+w–1)$

3.8.3 Block Merkmale

Ishii et al. führen in ihrer Arbeit [15] sogenannte Block-Merkmale ein. Sie überziehen das Trainingsbild dafür mit einem gleichmäßigem Raster und vergleichen die Intensitätssummen der Flächen. So können sie, im Gegensatz zu den Basis Haar-Merkmalen von Viola und Jones, verschiedene Flächen miteinander in Beziehung setzen.

3.8.4 Frei Zusammengesetzte HaarFeature

In der Arbeit [43] präsentieren Zhang et al. Haar-Merkmale, deren Teile frei im Bild positioniert werden und nicht mehr, wie bei den Basis Haar-Merkmalen von Viola

und Jones, einander berühren. Um die Komplexität der Permutationen etwas abzuschwächen, halten Zhang et al. Restriktionen in der Positionierung aufrecht. Zum Beispiel dürfen sich diese nicht überlappen und zwei Flächen, die zusammengerechnet werden, sollen auf einer Linie liegen. Weiter geht der Autor Kölsch in seiner Arbeit [21]. Dort dürfen die Flächen der Haar-Merkmale nach Belieben auf dem Bild positioniert werden, insbesondere auch überlappend.

3.9 Weitere (einfache) Klassifikatoren

Die Klassifikatoren die wir hier vorstellen, nutzen wie der Schwellwert-Klassifikator aus Abschnitt 3.2 die Haar-Merkmale als Grundlage für die Klassifikation eines Bildausschnittes. In 3.9.1 stellen wir den Binning-Klassifikator vor, der für seine Entscheidung auf Treffer-Wahrscheinlichkeiten in definierten Wertebereichen zurückgreift.

Den “Joint Haar like Feature”-Klassifikator nach [37] erläutern wir in dem Kapitel 3.9.2. Dieser kombiniert wenige einfache Klassifikatoren (z.B. Schwellwert Klassifikatoren) zu einem Ganzen und lernt die kombinierten Trefferwahrscheinlichkeiten.

3.9.1 Binning, LUT, Naive Bayes

Der Schwellwert-Klassifikator, wie Viola und Jones ihn genutzt haben und wie wir ihn in Kapitel 3.2 vorgestellt haben, sucht einen Schwelltwert, der die Menge der positiven und negativen Beispiele gut trennen kann. Aber nicht immer ist die Verteilung der Werte so, dass diese durch einen Schwellwert gut getrennt werden können. Befinden sich die Mittelwerte der Verteilungen der positiven und negativen Werte sehr dicht beieinander, wird ein Schwellwert-Klassifikator nur schwerlich gute Ergebnisse erbringen können.

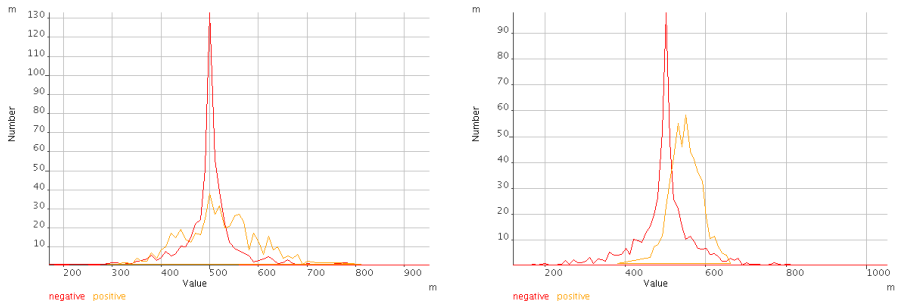


Abbildung 3.9: Verteilung der Werte, die durch ein Haar-Merkmal erzeugt wurden auf unserer Standard Test-Bildermenge. Zu sehen ist, dass die Verteilung in der rechten Graphik gut über einen Schwellwert trennbar ist, während die Anordnung in der ersten Graphik kaum zu befriedigenden Ergebnissen führen wird. Bei solchen Verteilungen kann ein Binning-Klassifikator Vorteile bringen.

Ein anderer sehr einfach aufgebauter Klassifikator ist der sogenannte "Binning-Klassifikator". Dieser zerlegt den Wertebereich den die Problemmenge annehmen kann, in kleinere Teile (Bins). Für jede dieser Teilbereiche werden die Wahrscheinlichkeiten für das Auftreten der verschiedenen Problemklassen abgespeichert (LUT = Look Up Table). In unserem Zwei-Klassenproblem werden die Anzahl der positiven Treffer und die Anzahl der negativen Treffer vermerkt.

Der Klassifikator trifft die Entscheidung, zu welcher Problemklasse ein Ausschnitt zu zählen ist, in dem er zuerst den Teilbereich heraussucht, zu welchen der Wert der Anfrage gehört und dann die abgespeicherten Häufigkeiten vergleicht (Naive Bayes).

Dieser Klassifikator ist schnell und einfach zu trainieren und kann auch leicht mit den Gewichten, die AdaBoost vergibt, umgehen.

Rasolzadeh et.al. kommen in ihrer Arbeit [1] zu dem Ergebnis, dass ein Binning-Klassifikator in den meisten Fällen in der Lage ist, die Klassifikation eines Ensembles zu verbessern.

In dem Kapitel 5 stellen wir die Realisierung unseres Binning-Klassifikators vor und eine davon abgewandelte Form, die wir "FuzzyBinning-Klassifikator" nannten und in denen wir die starren Grenzen der einzelnen Bins aufgeweicht haben.

3.9.2 Joint Haar-like Feature

Die Autoren Mita, Kaneko und Hori stellen in ihrer Arbeit [37] die sogenannten "Joint Haar-like Feature" vor. Bezugnehmend auf die Aussage von Viola und Jones, dass früh von AdaBoost ausgewählte Schwellwert-Klassifikatoren Fehlerraten zwischen 0.1 bis 0.3 und spätere Fehlerraten 0.4 bis 0.5 haben, bemerken Mita, Kaneko und Hori in [37], dass diese späten Schwellwert-Klassifikatoren zwar den Trainingsfehler reduzieren, aber nicht viel zur Generalisierung beitragen.

Um diesem Problem zu begegnen schlagen sie vor mehr markante Merkmale zu suchen, die die bezeichnenden Unterschiede der gesuchten Objekte deutlicher wiedergeben. Hierfür führten sie eine neues Merkmal ein, das "Joint Haar-like Feature".

"Joint Haar-like Feature" bezeichnet dabei die Methode mehrere binäre Klassifikatoren zu einem neuen verbesserten Merkmal zu kombinieren. Sie wählten dafür die Haar-Merkmale nutzenden Schwellwert-Klassifikatoren, wie sie auch Viola und Jones verwendet haben.

Laut Mita et al. können diese unabhängig von der Auflösung schnell berechnet werden und sind robust gegen Zufügen von Rauschen und Änderungen der Lichtverhältnisse.

Wie in dem Abschnitt 3.2 beschrieben, wird solch ein Schwellwert-Klassifikator trainiert in dem die Werte des genutzten Haar-Merkmals für alle Trainingsbilder berechnet werden und daraus der Schwellwert und die Richtung der Ungleichung bestimmt wird, die mit dem geringsten Fehler die positiven und negativen Beispiele

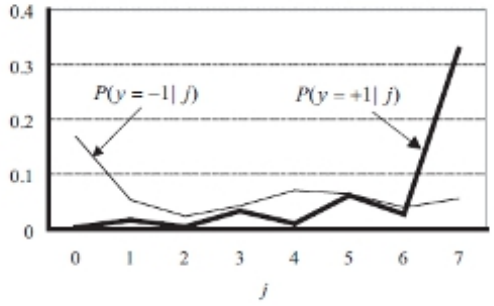


Abbildung 3.10: Beispiel für ein Joint Haar like Feature mit Featurewahrscheinlichkeiten. In der von [37] angegebenen Grafik bringt der Klassifikator nur für den Merkmalswert $(111)_2 = 7$ ein positives (+1) Klassifikationsergebnis.

trennen. Während sich bei Haar-Merkmalen der Wert über die Intensitäts-Summen der enthaltenen Rechtecke berechnet, berechnet sich der Wert der "Joint Haar-like Feature" aus dem zusammengesetzten binären Entscheidungen der Schwellwert-Klassifikatoren.

Somit kann der Wert eines "Joint Haar-like Feature", der aus drei Schwellwert-Klassifikatoren zusammengestzt ist, die Binärzahlen (000) bis (111) annehmen, je nachdem, wie die Ergebnisse der Schwellwert-Klassifikatoren ausfallen. Zum Beispiel wird der Merkmalswert eines "Joint Haar-like Feature", wenn die ersten beiden Schwellwert-Klassifikators $s(x) = 1$ zurückgeben und der dritte $s(x) = 0$, zu $j = (110) = 6$.

Der Klassifikator der nun mit diesen "Joint Haar-like Feature" arbeitet, basiert auf der "Bayes decision rule" (letztlich ein Binning-Klassifikator wie in 3.9.1) und wird wie folgt beschrieben.

$$\forall x \in j, h_t(x) = \begin{cases} +1 \text{ if} P(y=+1|j) > P(y=-1|j) \\ -1 \text{ otherwise} \end{cases}, \tag{3.12}$$

wobei j ein Merkmalswert des "Joint Haar-like Feature" ist.

Die $P(y = +1|j)$ und $P(y = -1|j)$ werden dabei unter Beachtung der Trainingsbeispielgewichte $D_t(i)$ wie folgt berechnet.

$$P(y = +1|j) = \sum_{i:x_i \in j \wedge y_i = +1} D_t(i), \tag{3.13}$$

$$P(y = -1|j) = \sum_{i:x_i \in j \wedge y_i = -1} D_t(i). \tag{3.14}$$

In dem von Mita et al. [37] angegebenen Beispiel (siehe 3.10) bringt der Klassifikator nur für den Merkmalswert $(111)_2 = 7$ ein positives $(+1)$ Klassifikationsergebnis. In den anderen Fällen ist die Wahrscheinlichkeit für ein negatives Ergebnis (-1) stets höher, als für ein positives $(+1)$.

Die größte Ausbeute an guten "Joint Haar like Feature" würde man mit einer erschöpfenden Suche über alle möglichen Kombinationen der zugrunde liegenden Schwellwert-Klassifikatoren erreichen. Die Komplexität, um F Merkmale aus einer Menge M zu besimmen, ist $O(M^F)$, was schnell und praktisch nicht machbar ist.

Mita et al. [37] nutzen in ihrer Arbeit die "Sequential Forward Selection" Methode. Bei dieser Methode werden in jedem Schritt, in dem ein weiteres Merkmal dem zusammengesetzten Klassifikator zugefügt wird, nur einmal alle vorhanden Merkmale geprüft. Es wird stets das Merkmal hinzugefügt, das die Güte des Klassifikators am meisten verbessert. Schritte zurück, in dem ein bereits zugefügtes Merkmal entfernt wird um weitere Permutationen zu testen, ist nicht vorgesehen. Nutzt ein Klassifikator also fünf Merkmale, so wird genau 5-mal die gesamte Merkmalsmenge geprüft, unter Zufügung des "Joint Haar-like Feature" der bestehende Klassifikator am stärksten verbessert wird.

Damit verringert sich die Komplexität auf $O(MF)$ von $O(M^F)$, wenn alle Möglichkeiten duchgetestet würden. M ist hier die Gesamtanzahl der Merkmale und F die Anzahl der Merkmale, die zu einem Klassifikator hinzugefügt werden sollen.

Die Anzahl F der zusammengesetzten Merkmale ist wichtig. Ist F zu groß, so neigen die Klassifikatoren zur Überanpassung.

Die Klassifikatoren, die die Joint Haar-like Feature nutzen, werden mit AdaBoost zu einem Ensemble trainiert.

Mita et al. machen Vergleiche mit verschiedenen Gesichtsdatenbanken und kommen zu dem Ergebnis, dass sie mit weniger Schwellwert-Klassifikatoren auskommen, als wenn sie die Schwellwert-Klassifikatoren direkt genutzt hätten. Sie trainierten in verschiedenen Varianten ein Ensemble von 1000 Schwellwert-Klassifikatoren, nach der Methode von Viola und Jones.

Zum Vergleich trainierten sie ein Ensemble von 333 Klassifikatoren, die die "Joint Haar-like Feature" mit jeweils 3 Schwellwert-Klassifikatoren nutzten. Damit ist die Gesamtzahl der benutzten Schwellwert-Klassifikatoren quasi gleich und damit die Berechnungskomplexität in der Klassifikation auch.

In ihren Experimenten benötigten die nach Viola/Jones trainieren Klassifikatoren deutlich mehr Merkmale, als die mit Joint Haar like Feature trainierten, um auf dieselbe geringe Fehlerquote zu kommen.

Reduktion der Merkmalsmenge 4

In diesem Kapitel zeigen wir ein paar einfache Möglichkeiten die Merkmalsmenge zu reduzieren, um die Trainingszeiten für das Ensemble-Lernen im Kontext von AdaBoost und Haar-Merkmalen zu verkürzen.

In dem Kapitel 4.1 stellen wir die Teile heraus, die dazu führen, dass AdaBoost sehr lange Trainingszeiten benötigt. Darauf aufbauend, betrachten wir in den Kapiteln 4.2 und 4.3 einfache und naheliegende Beschleunigungsmöglichkeiten.

Als wesentlichen Punkt erläutern wir in dem Kapitel 4.4 die von uns geprüfte Variante, in der wir den Boosting-Algorithmus AdaBoost selbst, in einer leicht geänderten Form, nutzen, um die Merkmalsmenge zu reduzieren. Wir erläutern, dass dies eine direkte und schnelle Möglichkeit ist, die Anzahl der Merkmale bzw. in dem Fall die Anzahl der Klassifikatoren, stark zu reduzieren.

Vergleichen werden wir dieses Verfahren mit einem Lernen aller vorhanden Merkmale (der erschöpfenden Suche) und einer zufälligen Auswahl von Merkmalen.

4.1 Problemanalyse

Folgende Betrachtungen beziehen sich auf AdaBoost und den Schwellwert-Klassifikator von Viola/Jones. Die Überlegungen betreffen jedoch analog die meisten "artverwandten" Methoden.

Der problematische Punkt für das Boosting-Lernen ist die große Menge an Merkmalen und die damit einhergehende lange Trainingsdauer. Ebenso ist die große Menge an Merkmalen ein wichtiger Punkt für den Erfolg des Boostings. Für den Boosting-Algorithmus wird derjenige Klassifikator gewählt, der den geringsten Fehler in der Trainingsmenge hat. Zum Training dieses Klassifikators für das Ensemble muss also ein bestimmtes Haar-Merkmal an einer bestimmten Stelle für alle Bilder der Trainingsmenge berechnet werden.

Der Aufwand für eine erschöpfende Suche die der AdaBoost Algoritmus ausführt, hängt damit zum einen von der Anzahl der Haar-Merkmale und der sich daraus ergebenden Anzahl der konkreten Ausprägungen ab (siehe 3.3). Und zum anderen hängt sie von der Größe und der Anzahl der Trainingsbilder ab.

Für ein Haar-Merkmal kann dessen Größe, beginnend mit der kleinstmöglichen, in X-Richtung und in Y-Richtung in Schritten vergrößert werden, bis die durch die Höhe und Breite der Trainingsbilder vorgegebene maximale Größe erreicht ist. Für jeden Schritt erhält man damit ein neues Haar-Merkmal in einer bestimmten Größe.

Jedes dieser Haar-Merkmale wird nun, wieder in Schritten in X-Richtung und Y-Richtung, an jeden Punkt des Trainingsbildes positioniert. Damit erhält man die Gesamtzahl aller möglichen Haar-Merkmale durch das Produkt der drei oben beschriebenen Teile. Nimmt man alle nur erdenklichen Merkmale, so sind die Schrittweiten X und Y entsprechend 1.

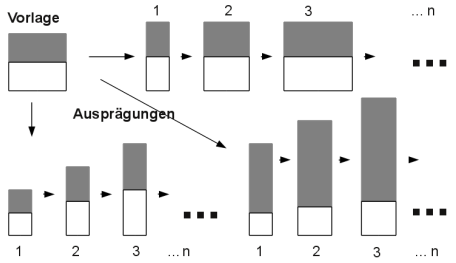


Abbildung 4.1: Generierung der konkreten Ausprägungen nach einer Vorlage. Aufgeführt sind Beispiele, wie aus der Vorlage konkrete Ausprägungen von Haar-Merkmalen generiert werden, indem diese Schrittweise in der Höhe und der Breite vergrößert werden.

Gesamtanzahl aller Merkmale ist demnach = Anzahl der Basis Haar-Merkmale x die Anzahl der Ausprägungen (X x Y) x die jeweilige Anzahl der Positionen im Bild (X x Y).

$$m_{sum} = b_m a_m \sum_{i=1}^{n} p_{m_i} \tag{4.1}$$

Hier bei ist m_{sum} die Summe aller Merkmale, b_m die Anzahl der Basismerkmale und a_m die Anzahl der Ausprägungen der Basis-Merkmale. Da die hier genutzten Basis-Merkmale alle die gleiche äußere Form haben, nämlich Rechtecke sind, sind auch die konkreten Variationen bzgl. ihrere äußeren Form gleich, womit für jedes Basis-Merkmal die Anzahl der Ausprägungen gleich ist. n ist die Anzahl aller Ausprägungen und p_{m_i} ist die Anzahl der Positionen, die diese i-te Ausprägung einnehmen kann.

Für ein konkretes Haar-Merkmal, also ein Haar-Merkmals mit einer bestimmten Form und Größe, wird nun der Wert des Merkmals für jede Position im Bild gemessen. Mit diesen Werten für alle Trainingsbilder wird schließlich der “einfache Klassifikator”, z.B. der Schwellwert-Klassifikator berechnet.

Der Aufwand für diesen letzten Schritt ist abhängig von der Anzahl der zur Verfügung stehenden Trainingsbilder.

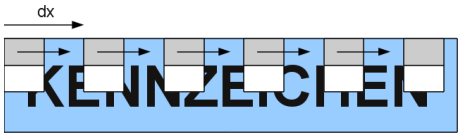


Abbildung 4.2: Jede Ausprägung der Haar-Merkmale wird über das Trainingsbild positioniert und sein Wert an der entsprechenden Stelle gemessen. Für jede Ausprägung kommen n Positionen hinzu. Für jedes Haar-Merkmal auf allen Positionen werden dann die entsprechenden Schwellwert-Klassifikatoren erstellt.

Wenn die Merkmale als eine Ursache für die lange Trainingsdauer und die Trainingsbilder als ein zweite betrachtet wird, so kann noch ein dritter Punkt herausgestellt werden, der im Wesen von AdaBoost begründet liegt. AdaBoost (vergleiche Abschnitt 3.1) sucht den Klassifikator mit dem geringsten gewichteten Fehler. Dazu trainiert er auf den aktuellen Gewichten den Klassifikator neu und berechnet dann den Fehler auf den Gewichten. Da der Klassifikator sich ändert (z.B. durch einen neuen Schwellwert), müssen die Ergebnisse seiner Klassifikation auf den Trainingsbildern auch neu berechnet werden, um den Fehler zu erhalten.

4.2 Vergößerung der Schrittweite

Eine sehr naheliegende Reduktionsmöglichkeit ist die Schrittweite für die Vergrößerung von Rechteckflächen und die Prüfung der Punkte auf dem Trainingsbild zu erhöhen. Dadurch werden systematisch Merkmale ausgelassen und die Anzahl aller Merkmale bisweilen drastisch reduziert.

Allerdings vermögen wir damit auch nicht zu sagen, ob wir nicht die besten Merkmale ausgelassen haben. Bei Merkmalen die sich nur wenig in ihrer Geometrie und auch nur wenig in ihrer Positionierung unterscheiden, können wir vermuten, dass deren Werte auch ähnlich sind, da die Veränderung der Werte von Haar-Merkmalen auf Grauwertbildern meist sehr stetig ist.

Die Einsparung an Merkmalen ist aber schon dann sehr groß, wenn wir die Schrittweite der Vergrößerung der Merkmale als auch die Schrittweite für die Positionen in den Beispielbildern auf 2er Schritte, statt 1er Schritten, erhöhen.

4.3 Filtern maximaler Fehler

Man könnte die Merkmalsmenge einfach dadurch verringern, dass man alle einfachen Klassifikatoren vorberechnet und nur die Klassifikatoren in den Boosting-Algorithmus mit einbezieht, die einen gewählten Fehler f nicht überschreiten. AdaBoost selbst erfordert lediglich, dass der Fehler der einzelnen Ensemble-Mitglieder

kleiner 0.5 ist. Grundsätzlich stände es uns aber frei, alle Klassifikatoren von vornhinein zu verwerfen, deren (ungewichteter) Fehler größer als z.B. 0.3 ist.

AdaBoost gewichtet die Trainingsbeispiele so, dass die Aufmerksamkeit auf diejenigen gelenkt wird, die bisher nur wenig berücksichtigt wurden. Nachdem also die "normalen" gelernt worden sind, werden die "Ausreisser" betrachtet. Klassifikatoren die eher auf diese Ausreisser spezialisiert sind, haben auf der ungewichteten Trainingsmenge meist einen höheren Fehler, so dass es gut sein kann, daß diese "Ausreisser-Klassifikatoren" bereits aus der Merkmalsmenge entfernt wurden.

Es bedarf einer Merkmalsmenge, die eine gewisse Verschiedenartigkeit aufweist und behält. Senken wir den Fehler zu stark, erhalten wir womöglich schlechte Ensemble-Klassifikatoren. Senken wir den Fehler zu gering, gewinnen wir nur wenig.

Die Frage, der wir in 7.5 kurz nachgehen, ist, ob es Fehlerwerte gibt, die uns einen Vorteil erlangen lasssen.

4.4 Auswahl durch Boosting

In diesem Abschnitt erläutern wir unsere Änderungen an AdaBoost, um es für die Merkmalsreduktion zu nutzen. Nachfolgend nennen wir diese Verfahren "Preboost".

Im folgenden Abschnitt mischen wir zuweilen die Bezeichnung Klassifikator und Merkmal. Da ein Klassifikator in diesen Betrachtungen genau ein Merkmal benutzt, ist der Sinnzusammenhang im nachfolgenden stets derselbe.

Viola und Jones schreiben in ihrer Arbeit [39]: "Our hypothesis, which is borne out by experiment, is that a very small number of these features can be combined to form an effective classifier. The main challenge is to find these features."

Die gefundene Antwort war, dass das Boosting diese Auswahl trifft, welche Merkmale zu dieser "kleinen Anzahl" gehören. Insofern ist der Boosting-Algorithmus auch ein Verfahren zur Reduzierung der Merkmalsmenge, wobei dann das gebildete Ensemble die reduzierte Merkmalsmenge darstellt.

Wie auch früher hier in der Arbeit schon kurz diskutiert, ist eine wichtige Eigenschaft, die alle Verfahren zur Merkmalsreduktion und Merkmalsauswahl erreichen bzw. erhalten wollen, die Diversität der Klassifikatoren. Diversität wollen wir hier als eine problemadäquate Verschiedenartigkeit beschreiben.

Wir wollen AdaBoost selbst, mit kleinen Veränderungen, nutzen, um die Menge der Merkmale zu reduzieren. Wir gehen davon aus, dass wir die angesprochene problemadäquate Verschiedenartigkeit durch die Gewichtung der Trainingsbeispiele erhalten und behalten.

Wenn wir Boosting nun zur Reduktion der Merkmalsmenge einsetzen wollen, müssen wir ihn schneller machen. Nachfolgend erläutern wir die Möglichkeiten dazu, die wir zu diesem Zweck betrachten wollen.

Wie in 3.1 dargestellt, werden in jedem Iterationsschritt die einfachen Klassifikatoren mit den angepassten Gewichten neu trainiert. Der Schwellwert wird also unter Beachtung der Gewichte der Trainingsbilder erneut berechnet, ebenso wie der damit einhergehende gewichtete Fehler.

Die Berechnung der Merkmalswerte in den Trainingsbildern ist unabhängig von der Gewichtung und damit in jeder Iteration derselbe. Diesen könnten wir also im Speicher halten. Wir hätten aber immer noch die Neuberechnung des einfachen Klassifikators und die Berechnung des damit einhergehenden neuen Fehlers, wobei letzteres vielfach mit wenigen Schritten geschehen kann.

Wir wollen die Geschwindigkeit von AdaBoost durch folgende drei Punkte erhöhen.

Zum einen berechnen wir die Klassifikatoren des Ensembles nicht in jeder Iteration neu. Der einfache Klassifikator selbst wird also nur einmal mit der Startgewichtung der Trainingsbilder trainiert.

Der zweite Punkt ist, dass die Klassifikationsergebnisse auf den Trainingsbildern sowie die Referenz zu den entsprechenden Trainingsbildern im Speicher gehalten werden. Es findet also nur einmal zum Beginn des Boosting-Prozesses eine Berechnung auf den Bildern statt. Danach werden alle Daten aus dem Speicher entnommen.

Die Reduzierung der Anzahl der Trainingsbilder ist der dritte Punkt.

Zusammenfassend halten wir fest, dass unsere wesentlichen Annahmen sind, die wir nachfolgend noch erweitern werden, dass wir auch dann eine brauchbar verschiedenartige Menge an Merkmalen durch AdaBoost erhalten, wenn wir die Klassifikatoren nicht in jeder Runde neu trainieren und auch nur auf einer kleinen Menge von Beispielbildern trainieren.

Nun ist aber die Merkmalsmenge für gewöhnlich zu groß, als dass man sie komplett in den Speicher schreiben könnte. Doch müssen wir die Ergebnisse für alle Klassifikatoren auf allen Trainingsbildern auch in den Speicher bekommen.

Um dies zu erreichen, gehen wir in Schritten durch die Merkmalsmenge. Wir bilden solche Untermengen der Merkmalsmenge, die wir, wie oben beschrieben, im Speicher halten können. Diese Untermengen erhalten wir, in dem wir die ersten N Merkmale nehmen und die ensprechenden Klassifikatoren berechnen.

Schließlich iterieren wir über die so entstandene Klassifikatoren-Menge entsprechend Kapitel 3.1, nur mit dem Unterschied, dass die einfachen Klassifikatoren selbst nicht neu berechnet werden. Der Fehler des einzelnen Klassifikators wird wie gehabt in jeder Runde neuberechnet, so wie auch die Gewichte in jeder Runde angepasst werden, was aber mit den Werten im Speicher sehr effizient und schnell gelingt. Auf

diese Weise erhalten wir einen durch AdaBoost trainierten Ensemble-Klassifikator. Einen für jeden Teil der ursprünglichen Merkmalsmenge.

Die Mitglieder des Ensemble-Klassifikators werden zu einer Gesamtmenge wieder zusammengefügt. Diese, jetzt schon reduzierte, Menge wird daraufhin, wie oben, wieder in die entsprechenden Teile zerlegt und es werden wieder Ensemble-Klassifikatoren erstellt. Wie eine auf dem Kopf stehende Pyramide wird dieser Prozess so lange fortgesetzt, bis nur noch ein Ensemble-Klassifikator übrig ist. Die Klassifikatoren dieses Ensembles stellen dann unsere reduzierte Merkmalsmenge dar.

Um die Verschiedenartikeit noch zu erhöhen, erstellen wir auf jeder Teilmenge an Merkmalen nicht einen Ensemble-Klassifikator, sondern mehrere. Die Klassifikatoren, die für das Vorgänger-Ensemble genutzt wurden, werden für den neuen Lauf des Boosting-Algorithmus aus der gerade bearbeitenden Menge entfernt, so dass sie nicht zweimal genutzt werden können.

Für die wesentlichen Aspekte unserer positiven Trainingsbeispiele sollte ein Klassifikator vorhanden sein, der in der Lage ist diesen zu beschreiben. Wir gehen davon aus, dass wir auf diese Weise unsere wichtige, gut "durchmischte" Merkmalsmenge erhalten, die aber deutlich reduziert ist. Wenn wir uns eine solche gut durchmischte Klassifikatorenmenge erstellen, haben wir eine gute Ausgangsbasis, um von hier aus einen neuen guten Ensemble-Klassifikator zu erstellen.

Die Umsetzung in dieser Arbeit geht von der Annahme aus, dass ähnlich gute, durchmischte Merkmale ausgewählt werden, wenn die einfachen Klassifikatoren nicht in jeder Runde neu trainiert werden, sondern nur durch ihren Fehler durch AdaBoost selektiert werden. Ebenso gehen wir davon aus, dass wir auch auf einer geringeren Anzahl an Trainingsbildern diese Merkmale ausgewählt bekommen.

Dass exakt dieselben Klassifikatoren in dem resultierenden Ensemble sein werden, wenn wir Bereichsweise ein "Preboost" ausführen oder das Verfahren von Viola und Jones auf der Gesamtmenge mit einer erschöpfenden Suche anstoßen, erwarten wir nicht. Was wir uns allerdings erhoffen, ist mit der reduzierten Menge in erheblich kürzerer Zeit einen nahezu gleichwertigen Klassifikator zu erstellen. Und, da wir leicht deutlich mehr Merkmale und Klassifikatoren verarbeiten können, dass wir dadurch auch zu besseren Ensemble-Klassifikatoren gelangen können. Dies dann allerdings nur durch den "unfairen" Vorteil, dass wir mehr Merkmale in unserer Ausgangsbasis zur Verfügung haben.

Konkret müssen für den Ablauf folgende Entscheidungen getroffen werden. Wir müssen festlegen, wieviele Haar-Merkmale die Ursprungsmenge enhalten soll. Dies ist natürlich eine Entscheidung, die unabhängig von der Methode ist, die wir benutzen.

Daraufhin berechnen wir für alle Merkmale den entsprechenden einfachen Klassifikator.

Nun bestimmen wir, wieviele Klassifikatoren ein einzelner Bereich umfassen soll und wie groß die Trainingsmenge ist. Einen Bereich füllen wir später so mit Klassifikatoren, dass die verwendeten Merkmale möglichst verschieden sind.

Schließlich müssen wir noch festlegen, wieviele Klassifikatoren dem Ensemble-Klassifikator zuzufügen sind. Wieviele Runden also der AdaBoost-Algorithmus laufen soll oder welche Detektionsrate und Falsch-Positiv-Rate das so erstellte Ensemble erreichen soll.

Um, wie oben erwähnt, die Durchmischung noch zu erhöhen, können wir angegeben, wieviele Ensemble-Klassifikatoren auf einer Bereichsmenge zu erstellen sind. Wenn wir z.B. die Menge der Bereichsmerkmale auf 600 Klassifikatoren reduzieren wollen, so können wir dies erreichen, indem wir einmal ein Ensemble mit 600 einfachen Klassifikatoren erstellen oder wir generieren sechs Ensemble-Klassifikatoren mit je 100 Mitgliedern.

Alternativ kann die Detektionsrate und die Falsch-Positiv-Rate vorgegeben werden. Mit dieser Einstellung verlieren wir allerdings die Kontrolle über die Anzahl der Ensemble-Mitglieder und damit über die Stärke der Reduktion.

Unsere Experimente führten wir meist mit allen drei Angaben durch. Wir gaben eine minimale Detektionsrate, eine maximale Falsch-Positiv-Rate und eine maximale Anzahl an Klassifikatoren pro Ensemble vor.

Es gilt zu klären, ob wir auf Basis einer kleinen, durch oben beschriebenes "Preboost"-Verfahren erzeugten Menge einen, zu einer erschöpfenden Suche und im Vergleich mit einer zufälligen Wahl, ebenbürtigen Ensemble-Klassifikator erstellen können. Desweiteren interessiert uns, wie die Einstellungen für das Preboost sein sollten, um hier besonders gute Ergebnisse zu erlangen und wie weit wir die Merkmalsmenge reduzieren könnnen.

- Gegeben eine Menge mit Beispielbildern $(x_1, y_1)...(x_n, y_n)$ wo $y_i = 0, 1$ für die negativen und positive Beispiele stehen.
- Weiter ist Gegeben eine Menge M von Klassifikatoren, die auf auf den Trainingsbildern trainiert wurden.
- Festgelegt wird:
  - N_{kb}, die Anzahl der Klassifikatoren pro Bereich
  - T, die Anzahl der Trainingsrunden (oder dr die Detektionsrate und fpr die Falsch-Positiv-Rate)
  - r, die Anzahl der Wiederholungen (die Anzahl der zu erstellenden Ensemble-Klassifikatoren
- $RM = GM$, wobei RM die reduzierte Menge ist und GM die Gesamtmenge aller Klassifikatoren.
- $NB = RM/N_{kb}$
- Solange $NB \geq 1$:
  1. Zerlege die Gesamtmenge der Klassifikatoren so in K Bereiche, daß ein Bereich B_i, N_{kb} Klassifikatoren enthält.
  2. für alle B_i $(i = 1\ bis\ K)$
     a) Erstelle Ensemble-Klassifikator E_i.
  3. $RM = \bigcup_i^K E_i$ (füge die Mitglieder aller Ensemble-Klassifikatoren E_i der neuen reduzierten Menge RM zu.)
  4. $NB = RM/N_{kb}$
- RM ist die finale reduzierte Menge.

Tabelle 4.1: Der Proboost-Algorithmus reduziert auf Teilbereichen, unter Nutzung von AdaBoost, die Eingangsmenge der Klassifikatoren. Die Zerlegung in Teilbereich geschiet so, daß die Anzahl der ähnlichen Merkmale innerhalb eines Bereichs gering ist.

Beschreibung der Realisierung 5

In diesem Kapitel stellen wir unsere Realisierung vor.

In dem Abschnitt 5.1 erläutern wir die “einfachen” Klassifkatoren, die wir für das Ensemble-Lernen implementiert und getestet haben.

Einen Überblick über die Software, die wir erstellt haben, geben wir in dem Abschnitt 5.3.

5.1 Klassifikatoren für das Ensemble-Lernen

5.1.1 Binning-Klassifikator

Der Binning-Klassifikator, den wir hier verwenden, zerlegt den Wertebereich in gleich große Teile. Die Anzahl der Teile kann für das Training konfiguriert werden.

Während des Trainings dieses Binning-Klassifikators, werden die Gewichte der Trainingsbilder für die entsprechenden Bereiche aufsummiert. So erhält man am Ende eine Datenstruktur (LUT), in dem die Bereichsgrenzen festgehalten sind und die Treffer- bzw. Hintergrund-Wahrscheinlichkeiten.

In unserer Umsetzung merken wir uns die minimalen und maximalen Werte sowie einen Offset, so dass das Minimum sicher ein positiven Wert hat oder 0 ist. Die Bereichsgrenzen ergeben sich, indem der Gesamtbereich durch die Anzahl der Bereiche geteilt wird. Die einzelnen Bereiche halten wir in einer Liste. Die Nummer des Bereichs, zu dem ein Wert gehört, können wir dann wie folgt berechnen.

$$interval = \frac{rangeMax - rangeMin}{numberBins} \tag{5.1}$$

$$binNumber = \lfloor \frac{featureValue + rangeOffset}{interval} \rfloor \tag{5.2}$$

Hierbei ist $interval$ der Wertebereich eines Bereiches, $rangeOffset$ die Verschiebung, um negative Werte zu vermeiden und $numberBins$ die Anzahl der Bereiche. $binNumber$ ist schließlich die Nummer des Bereichs, in dem wir nun prüfen können,

ob die positiven Gewichte $W+$ größer sind als die negativen Gewichte $W-$. Der boolesche Rückgabewert ist entsprechend.

$$featureValue = (W+ \geq W-) \tag{5.3}$$

5.1.2 FuzzyBinning-Klassifikator

Die Wahrscheinlichkeiten, die in dem Binning-Klassifikator gespeichert sind, werden auf den Trainingsbildern gemessen. Ihre Bereiche sind fest und es macht keinen Unterschied, ob der aktuell zu klassifizierende Wert dicht an den Grenzen des Bereichs liegt oder mittendrin. Je näher er an einem Rand des Bereichs liegt, desto mehr kann er auch einem Nachbarbereich zugeordnet werden.

Wenn man eine hohe Anzahl an Bereichen zu einem Binning-Klassifikator hinzufügt, so kann er die zu lernenden Daten besser abbilden, läuft aber auch Gefahr, auswendig zu lernen, also einer Überanpassung anheim zu fallen.

Diesem Gedanken folgend erweitern wir den Binning-Klassifikator um eine "Aufweichung" der starren Grenzen und nennen diesen FuzzyBinning-Klassifikator. Wobei dieser nur in dem Sinne "Fuzzy" ist, dass wir nicht nur die Werte innerhalb der starren Bereichsgrenzen beachten, sondern Nachbarbereiche mit abnehmendem Einfluss auch für die Ermittlung des Ergebniswertes hinzuziehen.

Der Bereich, in dem der zu klassifizierende Wert liegt, wird mit 1 bewertet. Je nachdem, wie dicht der Wert an einer Bereichsgrenze liegt, werden die Werte der Nachbarbereiche unterschiedlich gewichtet zur Gesamtberechnung der Trefferwahrscheinlichkeiten hinzugezogen.

$$k = \frac{fuzzyBins - i}{fuzzyBins\ z} \tag{5.4}$$

Dabei ist $fuzzyBins$ die Anzahl der Bereiche, die mit in das Ergebnis hereinspielen sollen, i ist die Entfernung zu dem Bereich (gemessen als Anzahl Bereiche), in dem der Wert liegt und z ist ein Faktor, um diesen Einfluss zu verändern.

Für die linksseitig benachbarten Bereiche:

$$W+, W- = \sum_{\frac{fuzzyBins}{2}} \alpha k_i \tag{5.5}$$

Für die rechtsseitig benachbarten Bereiche:

$$W+, W- = \sum_{\frac{fuzzyBins}{2}} (1 - \alpha) k_i \tag{5.6}$$

α ist der Abstand zum rechten Rand.

5.2 Training der Kaskade

Für das Training der Kaskade gibt es zwei unabhängige Abbruchkriterien. Zum einen wird nach der Berechnung eines Kaskadenknotens geprüft, ob die geforderte Gesamtdetektionsrate und maximale Falsch-Positiv-Rate der gesamten Kaskade erfüllt ist und zum anderen kann eine maximale Anzahl an Kaskadenknoten angegeben werden.

Für jeden Kaskadenknoten wird eine Mindestdetektionsrate und eine maximale Falsch-Positiv-Rate gefordert. Ob die geforderte Detektionsrate und Falsch-Positiv-Rate erfüllt ist, wird für die Kaskade als Ganzes und für die jeweiligen Knoten auf unterschiedlichen Teilmengen der Trainingsbilder geprüft. Die betrachtete Untermenge der Trainingsbilder für die Kaskaden-Erkennungsrate bleibt über den gesamten Trainingsprozess unverändert. Diese Untermenge bestand aus der Hälfte aller positiven Beispiele (1800) und ungefähr einem Zehntel der Negativbeispiele (18.000).

Die Menge der Trainingsbilder, auf denen die Detektionsrate und die Falsch-Positiv-Rate der einzelnen Knoten getestet wurde, wurde über Bootstrapping variiert. Es wurden also sämtliche als richtig erkannte Negativbeispiele aus der Trainingsmenge entfernt und durch neue Hintergrundbilder ersetzt. Zusätzlich wurden weitere, noch nicht betrachtete Negativbeispiele hinzugefügt, so dass die zu lernenden negativen Beispielbilder mit jedem Knoten mehr wurden. Gleichzeit wurde die geforderte maximale Falsch-Positive-Rate gesenkt.

Dageben blieb die geforderte Detektionsrate für jeden Knoten gleich. Auch die Positivbeispiele änderten sich nicht und umfassten die Gesamtmenge in Höhe von 3600 Bildern.

Zum Senken der Falsch-Positiv-Rate müssen 3 Werte angegeben werden. Einen Startwert für die Falsch-Positiv-Rate $fprStart$, einen Endwert $fprEnd$ und die Anazahl der Schritte, in denen der Endwert für die Falsch-Positiv-Rate erreicht werden soll $steps$. Beginnend beim Startwert wird bis zum erreichen des Endwertes dann die aktuell geforderte Falsch-Positive-Rate um $ds = (fprStart - fprEnd)/steps$ verringert.

Zusätzlich haben wir zwei weitere Abbruchkriterien angefügt.

Zum einen bricht unsere Realisierung des AdaBoost-Algorithmus ab, wenn der α_t-Wert (die Gewichtung des einzelnen Klassifikators, siehe auch 3.1), der sich aus dem Fehler berechnet, so gering wird, dass der dazugehörige Klassifikator kaum noch Einfluss auf das Gesamtergebnis hat.

Zum anderen fordern wir, dass jedes Merkmal nur einmal für das aktuelle Ensemble verwendet werden darf. Damit verbraucht sich der Vorrat an Klassifikatoren.

5.3 Aufbau der Software

In diesem Kapitel stellen wir die wichtigsten Funktionen unserer Software vor und beschreiben in groben Zügen den Aufbau.

Die Software zum Training der einfachen Klassifikatoren, der Ensembles und Kaskaden sind eigene Implementierungen und wurden in Java umgesetzt. Zurückgegriffen wurden auf verschiedene Opensource APIs.

Als DI (Dependency Injection)/IoC (Inversion of Control) Umgebung wurde das Spring-Framework genutzt.

Die Oberflächen wurden mit dem Framework Swixat erstellt. Swixat nutzt selbst das Framework Swixml, dass es erlaubt, Swing Oberflächen mit XML zu erstellen. Die XML Beschreibung der Oberfläche ist dabei dicht an die Swing Objekte und Methoden angelehnt. Swixat erweitert Swixml um eine MVC Umgebung und nutzt als DI Container das Spring-Framework.

Zum Erstellen der Graphiken wurde die API JChart2D genutzt.

Die Software läßt sich im Wesentlichen in drei Bereiche aufteilen.

Im Bereich “Training“ sind die Funktionen angesiedelt, die zur initialen Erstellung aller Klassifikatoren, zur Preboost Reduzierung und zum Trainieren der finalen Kaskade notwendig sind.

”Trainings-und Testbilder“ beinhaltet die Funktionen zum Verwalten und Manipulieren der Trainings- und Testbilder.

Zum Testen der Klassifikatoren stehen Funktionen im Bereich “Klassifikatoren-Tests” zur Verfügung.

Alle erzeugten Daten werden im XML Format gespeichert.

Für alle Trainings, Tests und Charterstellungen werden eigene Spring-Config Dateien mit den aktuellen Einstellungen generiert. In diesen Spring-Config Dateien sind alle Parameter fesgehalten, die Einfluss auf den Ablauf des Programmes haben. Die Spring-Config Datei wird bei den Ergebnissen belassen, so dass jederzeit nachvollziehbar bleibt, mit welcher Konfiguration ein Test, Training etc. durchgeführt wurde.

5.3.1 Training

Der Bereich “Training” läßt sich in die drei Teile “Klassifikatoren-Suche”, ”Preboost-Reduktion“ und “Kaskaden-Training” unterteilen.

Klassifikatoren-Suche

Diese Funktion erstellt alle Klassifikatoren für eine bestimmte Haar-Merkmalsmenge und speichert diese ab. Das ist der erste Schritt auf dem Weg, die Menge aller zur Verfügung stehenden Klassifikatoren zu reduzieren.

Einstellbar sind an dieser Stelle:

- Haar-Merkmalsmenge: Die Menge der Haar-Merkmale, die als Basis für die Erstellung der Klassifikatoren herangezogen wird.
- Weite-X, Weite-Y: Diese Werte legen fest, in welcher Schrittweite das jeweilige Haar-Merkmal über das Bild positioniert werden soll, um einen Klassifikator zu erstellen.
- Klassifikator: Hier wird gewählt, welcher Klassifikator trainiert werden soll.
- Trainingsbilder-Menge: Liegen mehrere Mengen von Trainingsbildern vor, können diese hier ausgewählt werden.
- Kalkulator: Der Kalkulator legt fest, wie die Flächen der HaarFeature in Beziehung gesetzt werden. Bei Viola und Jones wird die Differenz der Flächen berechnet. Wir nutzen dagegen weitetsgehend den Quotienten der Fläche.

Während des Testens der Klassifikatoren wird der aktuelle Status gespeichert. Sollte der Suchlauf gestoppt werden, so kann das Programm an der Stelle die Suche wieder aufnehmen, an der sie angehalten wurde.

Preboost Reduktion

An dieser Stelle wird die eigentliche Reduktion der Merkmals-Menge vorgenommen. Die Gesamtmenge der im Schritt “Klassifikatoren-Suche” erstellten Klassifikatoren wird so in Teile zerlegt, dass alle notwendigen Objekte im Speicher gehalten werden können. Für jeden Teilbereich werden dann durch den AdaBoost-Algorithmus ein oder mehrere Ensemble-Klassifikatoren erstellt. Die einfachen Klassifikatoren, aus denen der Ensemble-Klassifikator aufgebaut ist, bilden dann die Ausgangsmenge für den nächsten Reduktionsschritt.

Einstellbar sind an dieser Stelle:

- Klassifikatoren: Hier wird gewählt, welches Verzeichnis Grundlage für die zu reduzierenden Klassifikatoren sein soll, die unter dem vorher genannten Abschnitt erstellt wurde.
- Trainingsbilder-Menge: Liegen mehrere Mengen von Trainingsbildern vor, können diese hier ausgewählt werden.
- Positiv-Indikator: Die Software erkennt an dem Prefix des Dateinamens, ob es ich um ein positives oder negatives Beispiel. Dieser Prefix kann hier eingestellt werden. Vorgabe ist “positive”.

- Fehler-Filter: Es kann ein Filter für den maximalen Fehlerwert angegeben werden. Es werden dann nur solchen Klassifikatoren herangezogen, die den vorgegebenen Wert nicht überschreiten.
- Anzahl maximaler Trainingsrunden: die Anzahl der Trainingsrunden, die AdaBoost ausführen soll, wird an dieser Stelle nach oben begrenzt.
- Anzahl Wiederholungen: Dieser Wert gibt an, wie viele Ensemble-Klassifikatoren auf einem Teilbereich erstellt werden sollen. Die Gewichte werden dann wieder zurückgesetzt und die Klassifikatoren, die in der vorhergehenden Runde gewählt wurden, stehen nicht mehr zur Verfügung.
- Anzahl positiver Trainingsbilder: Der Wert legt die maximale Anzahl an positiven Beispielen fest.
- Anzahl negativer Trainingsbilder: Der Wert legt die maximale Anzahl an negativen Beispielen fest.

Kaskaden-Training

Unter diesem Menüpunkt wird die Kaskade trainiert. Wie in 5.2 beschrieben, wird für jeden Knoten eine minimale Detektionsrate und eine maximale Falsch-Positiv-Rate bestimmt. Der Klassifikator eines Knotens ist dann der über AdaBoost trainierte Ensemble-Klassifikator. Die Komplexität des Ensemble-Klassifikators wird über zwei Einstellungen gesteuert. Zum einen steigt mit jedem Knoten (bis zu einem Maximalknoten) die Anforderung an die Falsch-Positiv-Rate. Diese wird für jeden fortlaufenden Knoten gesenkt. Zum anderen steigt die Anzahl der negativen Trainingsbilder mit jedem Knoten. Zusätzlich wird Bootsraping verwendet, so dass der nachfolgende Knoten nur auf negativen Beispielbildern lernt, die der Vorgänger falsch klassifiziert hat.

Einstellbar sind an dieser Stelle:

- Min Detektionsrate Kaskade: An dieser Stelle wird die minimal erwartete Detektionsrate eingestellt. Diese gilt sowohl für die Kaskade als Ganzes als auch für jeden einzelnen Ensemble-Klassifikator im Knoten.
- Max Falsch-Positiv-Rate Kaskade: Der Wert legt die maximale Falsch-Positiv-Rate für die gesamte Kaskade fest. Ist diese und die "Min Detektionsrate Kaskade" erreicht, endet das Kaskadenlernen.
- Min Ensemble-Schwellwert: Hier kann der Schwellwert des Ensemble-Klassifikators nach unten beschränkt werden. Bis zu diesem Wert wird dann der Schwellwert des Ensemble-Klassifikators gesenkt, um die geforderte Detektionsrate und Falsch-Positiv-Rate zu erreichen.
- Anzahl Erhöhung Negativbeispiele: Um die hier angegebene Anzahl werden die negativen Trainingsbeispiele in jeder Runde erhöht.

- Max Kaskadenknoten: Mit diesem Wert wird die Anzahl der Kaskadenknoten nach oben beschränkt.
- Lern-Modul: Mit dem Lern-Modul wird gesteuert, ob die erschöpfende Suche eine Menge der Preboost Klassifikatoren nutzen oder das Joint Haar-like Feature Lernen gestartet werden soll.
- Max Falsch-Positiv-Rate Knoten: Die maximale Falsch-Positiv-Rate für den ersten Knoten des Kaskadenlernen.
- Min Falsch-Positiv-Rate Knoten: Der Minimalwert, den ein einzelner Knoten erreichen soll.
- Schritte Falsch-Positiv-Rate Knoten: Dieser Wert gibt vor, innerhalb wievieler Knoten der Wert von "Max Falsch-Positiv-Rate Knoten" auf "Min Falsch-Positiv-Rate Knoten" zu senken ist.

Während des Kaskadentrainings werden die verschiedenen relevanten Statusinformationen gespeichert. Die wesentlichen Statusinformationen sind die folgenden: Im Status des Bootstrapping wird festgehalten, welche Bilder für den aktuell trainierten Ensemble-Klassifikator verwendet werden. Für den Ensemble-Klassifikators wird stets der letzte Stand abgespeichert, also der aktuelle Ensemble-Klassifikator selbst. Darüber hinaus werden die Gewichte für die verwendeten Trainingsbilder abgespeichert. Der Status des Kaskaden-Klassifikators schließlich beinhaltet die Informationen, welcher Knoten gerade trainiert wird und verweist auf die XML-Repräsentation des letzten Kaskaden-Klassifikators. So ist gewährleistet, dass das Training dort wieder starten kann, wo es gestoppt wurde.

5.3.2 Trainings-und Testbilder

Die Funktionen im Bereich "Trainings- und Testbilder" ermöglichen es, eine Bildermenge in Test- und Trainingsbilder zu zerlegen, die interessanten Bereiche innerhalb eines Bildes zu markieren, diese mit Attributen zu versehen und Ausschnitte zum Training zu extrahieren.

Unter dem Menüpunkt "Image Tagging" kann ein Verzeichnis mit Bildern geöffnet werden. Nacheinander kann man sich alle Bildern anzeigen lassen und bearbeiten. Mit der Maus können nun die Bereiche markiert werden, auf die wir unsere Aufmerksamkeit richten wollen.

Einen solchen Bereich bezeichnen wir als "Interest Region".

Zu jedem Bereich können zusätzlich verschiedene Attribute eingetragen werden. Für die Kennzeichen wurden z.B. der Attributname "Ausrichtung" mit den Attributwerten "gerade", "schief" etc. und das Attribut "Lesbarkeit", mit Werten wie "gut", "kaum" oder z.B. "nicht" zugefügt.

Die so markierten und mit Attributen versehen Bereiche werden dann für jedes Bild gespeichert. Wir nutzen hier XML, um die "Interest Regions" und deren Attribute festzuhalten.

Zum Erstellen der Trainingsausschnitte bedient man sich des Menü-Punktes "Extract Images". Dort kann eingestellt werden, wieviele negative und positive Bilder man erhalten möchte und wie diese variiert werden sollen.

In den vorgegebenen Bildmengen der Originalbilder wird dann nach den markierten Bereichen gesucht. Die Angabe von Attributen wird als Filter genutzt, so dass nur diejenigen Bereiche ausgeschnitten werden, die den geforderten Attributen entsprechen.

Im selben Schritt werden aus jedem Bild auch die Negativ-Beispiele ausgeschnitten. Diese können auf zwei Arten bestimmt werden. Zum einen werden Bereiche aus den Bildern ausgeschnitten, die dicht bei den Nummerschildern liegen. Und zum anderen werden zufällig Ausschnitte aus dem ganzen Bild gewählt. Die negativen Ausschnitte schwanken in ihrer Ausdehnung sowohl in der Breite als auch in der Höhe. Bei den Ausschnitten, die dicht bei den Nummerschildern liegen, wurden die Vorgaben gemacht, daß die Schwankungen dort gering ausfallen.

Im einzelnen gibt es folgenden Einstellungen:

- Anzahl Negativer pro Positiv: Der Wert bestimmt die Anzahl der negativen Beispiele, die für jedes positive Beispiel ausgeschnitten und gespeichert werden.
- Anzahl zufällige Negative: Hier wird festgelegt, wieviele negative Beispiele, zusätzliche und zufällig verteilt, aus dem Bild ausgeschnitten werden sollen.
- Positiver Offset: Die positiven Ausschnitte können an dieser Stelle vergrößert werden. Es wird dann mehr Rand um das eigentlich Objekt mit ausgeschnitten.
- Anzahl zufällige Positive: Soviele zufällig variierende positive Beispiele werden zusetzlich erstellt.
- Zufalls-Offset-Breite: Um diesen Wert wird die Breite zufällig variiert
- Zufalls-Offset-Höhe; Um diesen Wert wird die Höhe zufällig variiert.

Unter dem Menüpunkt "Shuffle" gibt es die Möglichkeit die Bilder zu vermischen. Wenn die Bilder ausgeschnitten werden, so liegen sie erst einmal in der Reihenfolge vor, wie diese ausgeschnitten wurden. Dieser Punkt soll verhindern, daß sich schwer zu lernende Beispiele häufen.

5.3.3 Klassifikatoren-Tests

Es gibt zwei Arten, die Klassifikatoren zu testen. Zum einen kann sie gegen die Ausschnitte der Test- und Trainingsbilder getestet werden und zum anderen kann gegen komplette Bilder getestet werden.

Beides ist sowohl für die Kaskaden-Klassifikatoren als ganzes möglich als auch für die einzelnen Ensemble-Klassifikatoren der Kaskaden-Knoten.

Die Ergebnisse der Tests werden sowohl als ausführliche Log-Datei gespeichert als auch als CSV.

Des weiteren existiert ein Programm, mit dem man diese einzelnen CSV-Dateien zu einer ganzen zusammenfassen kann. Dies haben wir oft genutzt und ist hilfreich, um sich einen Überblick über die "Fähigkeiten" der einzelnen Kaskaden zu verschaffen.

In diesem Bereich ist auch die Erstellung der Grafiken angesiedelt.

Trainings- und Testrahmen

6

In diesem Kapitel beschreiben wir den Rahmen, in dem unsere Versuche stattgefunden haben.

So beschreiben wir in 6.1, welche Trainingsbilder und Testbilder wir genutzt haben.

In Kapitel 6.2 stellen wir die Haar-Merkmalsmengen vor, die wir genutzt haben.

Im besonderen Fokus ist die Frage, wie sehr sich die Kaskaden in ihrer Erkennungsleistung unterscheiden, wenn die Merkmalsmenge zur Trainingsbeschleunigung eingeschränkt wird. In dem Abschnitt 6.3 stellen wir die verschiedenen Konfigurationen vor, mit denen Preboost arbeiten kann.

Detektionsrate, Falsch-Poistiv-Rate und die Anzahl der Klassifikatoren in einem Ensemble sind für uns die wichtigen Kriterien für die Güte einer Kaskade. In unseren Ergebnissen (Kapitel 7) kommen wir immer wieder auf diese Werte zurück, weshalb wir sie in dem Kapitel 6.4 näher beschreiben und begründen.

6.1 Trainings- und Testbilder

Die hier verwendeten Trainings- und Testbilder sind zu verschiedenen Zeiten fotografiert. Es wurden sowohl Bilder mit einer Digitalkamera als auch Bilder mit einem Handy gemacht.

Des weiteren haben wir griechischen Autobilder der Autoren Anagnostopoulos et al. [6], die sie auf ihrer Website zum Download anbieten, als Testbilder verwendet. Es wird also nie mit griechischen Nummerschildern gelernt.

Die Menge der selbst erstellten Bilder wird zufällig in zwei Gruppen geteilt, eine Testmenge und eine Trainingsmenge. Da unsere eigene Bildermenge nicht so umfangreich ist, verzichten wir, wie andere Autoren auch, auf die Validierungsmenge.

Wir schneiden 350 positive Beispiele für die Trainingsmenge aus, die wir durch zufälligen Versatz in der Höhe und der Breite, angelehnt an die Arbeit [22] der Autoren Dlagnekov und Belongie, auf 3.600 erhöhen.

Es werden im selben Schritt zwei Formen von negativen Beispielen ausgeschnitten. Zum einen wird beliebig ein Ausschnitt aus dem Bild gewählt und gespeichert. Während hierbei die Position ohne Einschränkung und zufällig gewählt wurde,

schwankte die Ausdehnung zufällig um eine festgesetzte durchschnittliche Kennzeichenausdehnung in den Bildern.

Zum anderen werden Ausschnitte gewählt, die dicht bei dem Kennzeichen liegen und in ihrer Form leicht um die Ausdehnung des gerade betrachteten Kennzeichens schwanken.

So erstellen wir am Ende eine Menge von knapp 190.000 negativen Trainingsbeispielen.

Alle Ausschnitte werden zu Grauwertbildern gewandelt und auf 60x20 Pixel skaliert.

Die Kennzeichen der Autos innerhalb der Bilder wird markiert und mit Attributen versehen. Wie in Kapitel 5.3.2 dargestelt können wir die Objekte, die wir ausschneiden wollen über eben diese Attribute einschränken. So werden in dieser Arbeit für das Training ausschließlich Kennzeichen verwendet, die bei dem Attribut “Lesbarkeit” zwischen “sehr gut” und “knapp” liegen und bei dem Attribut “Ausrichtung” “gerade” oder “leicht schief” aufweisen. Als Trainingsmenge werden also nur Nummernschilder verwendet, die halbwegs gerade und lesbar waren.

Da wir uns hier insbesondere für den Vergleich der Reduktionsmöglichkeiten der Merkmalsmenge interessieren, ist diese Einschränkung kein Nachteil.

Unsere Testbilder sind nicht durch solche Attribute eingeschränkt. Es gibt also durchaus Bilder in den Testmengen, die deutlich schiefere und schlechter lesbare Nummernschilder enthalten, als sie in den Trainingsdaten vorhanden sind. Da diese aber wieder alle untersuchten Verfahren trifft, sehen wir auch das nicht als Nachteil an.

Wir benutzten folgende Test-Bildmengen, auf die wir uns in den Ergebnissen beziehen werden und die wir beispielhaft im Appendix aufgeführt haben.

- Standard Testbilder 37: Diese Testmenge enthält 37 Bilder mit 38 Kennzeichen mit einer eine Größe von 380x280 Pixeln. Es sind viele Winterbilder, häufig mit bedecktem Himmel und mittelmäßig viel Hintergrund. (Beispiele siehe A.1)

- Parkplatz Testbilder 43: Diese Testmenge enthält 43 Bilder mit 129 Kennzeichen mit einer Größe von 1.024x1.280 Pixeln. Es sind Parkplatzmotive mit hellem und bedecktem Himmel und sehr viel Hintergrund. (Beispiele siehe A.2)

- Griechische Testbilder 67: Diese Testmenge enthält 67 Bilder mit 71 Kennzeichen mit einer Größe von 480x360 Pixeln. Es sind meist einzelne Autos mit griechischen Nummernschildern. Diese Bilder stammen von Anagnostopoulos et al. [6]. Die meisten sind hell mit nicht so viel Hintergrund. (Beispiele siehe A.3)

6.2 HaarFeature

In dem Abschnitt 3.3 haben wir beschrieben, was Haar-Merkmale sind und wie sie berechnet werden. In unserer Problemanalyse zur Reduktion der Merkmale in dem Abschnitt 4.1 haben wir genauer dargelegt, unter welchen Faktoren die Haar-Merkmale aus der Vorlage generiert werden.

So wird aus einer Vorlage, die lediglich die Form festlegt, eine konkrete "Ausprägung". Diese Ausprägung stellt letztlich erst das konkrete Haar-Merkmal dar, das wir verwenden werden.

Die Ausprägungen der Haar-Merkmale werden generiert, in dem die Vorlage mit unterschiedliche Höhen und Breiten für die einzelnen Flächen versehen wird. Den verschiedenen Mengen geben wir Namen, aus welcher "Basis" sie kommen, und wieviele Ausprägungen existieren.

So steht die Menge "BaseHaarFeature 393" für die Menge der HaarFeature Ausprägungen, die aus den Grundformen nach Viola und Jones gebildet wurden und 393 HaarFeature umfassen.

Die 393 HaarFeature Ausprägungen ergeben sich, wenn die Flächen sukzessive in X-Richtung in 8er Schritten und in Y-Richtung in 5er Schritten vergrößert werden. Dabei werden für alle Höhen jeweils alle Permutationen der Breiten erstellt, was dann wie erwähnt eine Menge von 393 konkreten Ausprägungen der HaarFeature ergibt.

In dieser Arbeit benutzen wir noch mehr Mengen. Die Namensgebung der Mengen sind alle nach obigen Schema aufgebaut.

Folgende Haar-Merkmalsmengen haben wir häufig genutzt.

- Basis Haar-Merkmale 393: 393 Ausprägungen der Basis Haar-Merkmale (siehe auch 6.1) ergeben 27.000 mögliche Klassifikatoren.
- Basis Haar-Merkmale 3081: 3081 Ausprägungen der Basis Haar-Merkmale (siehe auch 6.1) ergeben 240.000 mögliche Klassifikatoren.

Folgende Formen haben wir für die Generierung unserer Merkmalsmengen verwendet.

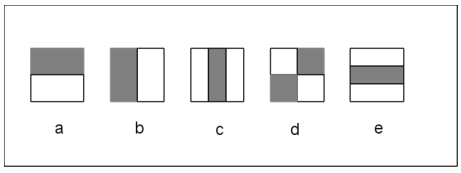


Abbildung 6.1: Die Basis Haar-Merkmale nach Viola und Jones.

Im Gegensatz zu anderen Autoren, die Haar-Merkmale genutzt haben, haben wir für die Berechnung des Haar-Merkmal-Wertes nicht die Summe berechnet, sondern

den Quotienten der Flächen genommen. Also

$$hfValue = \frac{F_A}{F_B} \tag{6.1}$$

statt

$$hfValue = F_A - F_B \tag{6.2}$$

wobei F_A die Pixelsumme der Rechteckfläche A und die F_B die Pixelsumme der Rechteckfläche B ist.

Dafür wurde auf die Varianz-Normalisierung verzichtet.

6.3 Preboost

Für das Preboosting wird auf einer geringen Menge an Trainingsbildern gearbeitet. So werden meistens 610 von 3600 positiven Beispielen und 1.500 von 180.000 negativen Beispielen verwendet. (Siehe auch 6.1

Wie oben in Abschnitt 6.2 dargestellt können mit den "Basis Haar-Merkmalen 3081" 240.000 Klassifikatoren erstellt werden. In einem ersten Schritt werden alle Klassifikatoren entfernt, deren Fehler auf Trainingsbilder schlechter als 0.45 ist, was die Anzahl der Klassifikatoren auf etwas mehr als 200.000 reduziert.

Die Vorberechnung der Klassifikatoren geschieht in der Reihenfolge, wie die Merkmale vorliegen. Da für jedes Basismerkmal nacheinander die Ausprägungen berechnet werden und dann für jeweils jede Ausprägung die Positionierungen auf dem Trainingsbild, liegen die ähnlichen Klassifikatoren in einer Liste direkt nebeneinander. Um nun aber in den einzelnen Bereichen eine möglichst hohe Verschiedenartigkeit zu erhalten, füllen wir die Bereiche auf, auf denen wir das Preboost durchführen, in dem wir jedes $n$te Merkmal hinzufügen. Wir lassen also stets ein paar Merkmale in der Liste aus, um nicht die unmittelbaren Nachbarn ein einer Menge zu haben. Wie viele Merkmale wir beim Auffüllen der Mengen auslassen, berechnet sich nach der Größe der gesamten Merkmalsmenge und an der Anzahl der Merkmale, die wir in einem Bereich zusammenfassen wollen.

In der Umsetzung geschieht dies wie folgt. Wir berechnen den Quotienten aus der Gesamtzahl aller Klassifikatoren und der Anzahl der Klassifikatoren, die in einen Bereich sollen. ($step = GM/BM$) Aus diesem Quotienten erhalten wir die Schrittweite $step$, in der wir die Bereiche auffüllen. Wir beginnen also mit $i = 1$ und nehmen dann jeden $step$ten Klassifikator für den $i$ten Bereich. Danach folgt $i = i + 1$ und wieder kommt jeder $step$te Klassifikator in den entsprechenden Bereich. Dies wird solange wiederholt, wie $i \leq step$ gilt.

Preboost können wir mit unerschiedlichen Einstellungen laufen lassen. Es hat sich aber gezeigt, daß mit folgender Konfiguration am sichersten gute Ergebnisse erziehlt werden:

- Detektionsrate: 0.99
- Falsch-Positive-Rate: 0.01
- Anzahl Wiederholungen: 2-3

6.4 Bewertungskriterien der Kaskaden

In diesem Abschnitt stellen wir die Kriterien vor, nach denen wir unsere Klassifikatoren bewerten.

Wir beurteilen die Güte über die Anzahl der einfachen Klassifikatoren in dem Ensemble, die Detektionsrate und die Falsch-Positiv-Rate, was wir in den Abschnitten 6.4.1 und 6.4.2 näher erläutern werden.

Mit welchen Tests wir die Ergebnisse ermitteln, auf Grund derer wir die Güte unserer Kaskaden beurteilen, legen wir im Abschnitt 6.4.3 dar.

6.4.1 Anzahl der einfachen Klassifikatoren im Ensemble

Ein wichtiger Punkt ist die Anzahl der Klassifikatoren pro Kaskadenknoten. Sie ist wichtig für die Gesamtgeschwindikeit des Systems und gibt uns darüber hinaus Auskunft über die Güte unserer Vorauswahl.

Wir wollen ein einfaches und schnelles System. Jeder Klassifikator benötigt bei seiner Ausführung Rechenzeit. Je weniger Klassifikatoren bemüht werden müssen, um so effizienter erreicht das Ensemble eine Entscheidung, um so effizienter ist dementsprechend der gesamte Klassifikationsprozess.

Je mehr Klassifikatoren genutzt wurden, um das Ensemble-Ziel zu erreichen, umso schlechter waren die einzelnen Klassifikatoren. Das insbesondere läßt Schlüsse auf die Güte unserer Vorauswahl zu. Waren sonst alle Vorbedingungen für die beiden Ensemble-Trainings gleich und differierten diese nur in der Wahl der Klassifikator-Menge, aus denen sie auswählen konnten, so können wir darauf schließen, daß die eine Menge weniger geeignet war als die andere. Da wir unsere Methode zur Merkmalsreduktion prüfen, ist dieser Schluss für uns wichtig.

Ein weiterer positiver Effekt einer guten und kleinen Merkmalsmenge ist, daß das Training der Kaskade sich erheblich beschleunigt, da nicht soviele Runden trainiert werden müssen, um mit den einzelnen Ensemble-Klassifikatoren die geforderte Detektions- und Falsch-Positiv-Rate zu erreichen.

Eine Kaskade mit weniger Klassifikatoren pro Kaskadenknoten ziehen wir also einer mit mehr Klassifikatoren vor.

6.4.2 Detektions- und Falsch-Positiv-Rate

Die Detektionsrate und die Falsch-Positive-Rate betrachten wir im Wechselspiel miteinander, da wir nur wenig gewinnen, wenn wir eine 100% Trefferquote haben, aber die Anzahl der Falsch-Positiven so hoch ist, daß die Aussagefähigkeit des Systems stark sinkt und der Aufwand für die nachfolgenden Prozesse stark steigt.

Je nach Größe eines Testbildes befinden sich schnell viele Zehntausend negative Ausschnitte innerhalb eines Bildes, die betrachtet werden müssen. In einer unserer Testbildmenge mit Bildern, die im Schnitt 380x280 Pixel groß sind, ergeben das 60.000 Auschnitte, die allein in einer Skalierung klassifiziert werden müssen. Die Falsch-Positiv-Rate wird hier sehr gering und unanschaulich. Wir geben daher die Gesamtzahl Falsch-Postiven auf einer Testmenge an.

Für die Bewertung einer Kaskade ist uns eine geringe Anzahl an Falsch-Positiven wichtiger als eine maximale Detektionsrate. Wir gehen davon aus, daß wir die Detektionsrate noch erhöhen können, wenn wir mit Sequenzen von Bildern arbeiten, da wir mehr Informationen aus unterschiedlichen Blickwinkeln zur Verfügung haben. Währenddessen läuft eine zu hohe Falsch-Positiv-Rate Gefahr, unser Gesamtsystem in seiner Effizienz stark zu beeinträchtigen, da wir davon ausgehen, dass nachgelagerte Prozesse der Erkennung aufwändiger sind. Diese würden aber auch auf den Falsch-Positiven ausgeführt werden müssen.

6.4.3 Tests: Mit und ohne Merging

Schwierig ist die Festlegung, wann wir ein Resultat als Treffer bewerten wollen. Für nachgelagerte Prozesse ist dieses klar zu beantworten. Der Treffer-Ausschnitt muss mindestens so groß sein, wie das Kennzeichen, sollte aber auch nicht viel größer sein, da das ein nachfolgendes Erkennen erschweren würde.

Innerhalb unserer Testumgebung berechnen wir die Fläche, die der vermeintliche Treffer mit dem markierten Kennzeichen gemeinsam hat, also den Durchschnitt der beiden Flächen. Berechnet wird immer die gemeinsame Fläche im Verhältnis zu der Fläche des Nummernschildes. Gibt es keine Überschneidungen der Flächen, so ist dies klar ein Falsch-Positiver “Treffer”.

Gibt es jedoch Überschneidungen, müssen wir ein Maß festlegen, ab wann wir dies als Treffer werten wollen.

Eine automatische Auswertung mit einer kompletten Überschneidung zu fordern, würde den Ergebnissen aufgrund der unterschiedlichen Geometrie und Ansichten in den Bildern nicht gerecht. Auch macht sich bemerkbar, dass wir unsere Beispielbilder zufällig und unterschiedlich mit Rand versehen haben, wodurch die Streuung um das Nummernschild herum größer ist. Manche Autoren sind dazu übergegangen, die Bilder manuell auszuwerten.

Auf den Bildern können wir zwei verschiedene Tests betrachten. Zum eine das direkte "ungeschminkte" Ergebnis der Klassifikatoren und das Ergebnis nach einem Verschmelzungsprozess, was es uns erlauben wird, bestimmte Aspekte leichter herauszuarbeiten.

Auf diese Weise ohne einen Verschmelzungsprozess sehen wir am deutlichsten, wie groß unsere Falsch-Positiven sind. Testen wir mit einem Verschmelzungsprozess, so kann es geschehen, dass besonders schlechte Ergebnisse nicht sofort als solche zu erkennen sind. Wir können dann eine Kaskade mit einer hohen Detektionsrate und vermeintlich wenigen Falsch-Positiven haben, bei der sich bei genauerer Betrachtung herausstellt, dass viele Falsch-Positive lediglich zu einer sehr großen Fläche zusammengefügt wurden.

Zum anderen können wir diese Falsch-Positiven zuweilen durch Zusammenfügen dicht beieinander liegender Treffer leicht reduzieren. Um dies zu erkennen geben wir eine Zahl mit aus, die uns etwas über die Größe der Fläche im Verhältnis zum Bild sagt. Sie ist aber meist unanschaulich.

Für das Verschmelzen spricht, daß tatsächlich häufig viele Treffer dicht beieinander liegen und sich stark überlappen. Ein Treffer mitten im Nummernschild kann ohne Verschmelzen als Falsch-Positiv gewertet werden, da die gemeinsame Überlappung mit dem Nummernschild zu gering ist. Wir möchten einen Unterschied machen können zwischen zwei Kaskaden, von denen eine die Falsch-Positiven über das Bild verstreut und die andere wenige Bereiche falsch klassifiziert, diese dafür mehrfach.

Gibt es keine oder nur wenige Überlappungen mit anderen vermeintlichen Treffern, so ist dies ein gutes Indiz dafür, daß es sich um einen Falsch-Positiven handelt. In ihrer Arbeit [22] nutzen die Autoren genau diese Beobachtung. Sie entfernen "Treffer", die für sich alleine stehen und schaffen es damit, die Falsch-Positiven noch weiter zu reduzieren.

So führen wir einen Test durch, bei dem die gemeinsame Fläche mit dem Nummernschild nur 0.5 betragen muss, um als Treffer bewertet zu werden. Dafür findet kein Zusammenfügen dicht beeinander liegender Flächen statt. Die Anzahl der Falsch-Positiven kommt hier deutlich hervor. Da für gewöhnlich mehrere Treffer dicht beieinander liegen, gibt dieses Vorgehen die tatsächlichen Gegebenheiten gut wieder.

Der zweite Test wird mit einem Zusammenfügen der nah beieinander liegenden Flächen durchgeführt. Für solch ein Zusammenfügen der Flächen werden in der Literatur verschiedene Methoden verwendet. Da wir hier aber insbesondere den Vergleich zwischen Klassifikatoren anstellen, nutzen wir hier eine relative einfache Variante des Verschmelzungsverfahrens, indem wir Flächen zusammenfügen, so bald sie sich um einen bestimmten Wert überlappen. Bei unseren Tests haben wir den Wert 0.5 gewählt, gemessen an der jeweils größeren Fläche.

In den meisten Fällen führen wir die Tests ohne ein Verschmelzen durch. Nur bei den guten Kaskaden prüfen wir die Ergebnisse noch einmal mit dem Verschmelzungsprozess.

Abbildung 6.2: Ergebnisbilder zur Illustrierung der Unterschiede mit und ohne Zusammenfügen dicht beieinander liegender Flächen. Grün ausgefüllt zeigt die gemeinsame Fläche der "Treffer" mit dem blau eingezeichnet Nummernschild. Die einzelnen Treffer sind mit einem roten (beim Merge mit gelben) Rechteck markiert. Als Treffer gewertete Bereiche sind mit einem grünen Rechteck markiert.

Experimentelle Ergebnisse

7

In diesem Kapitel betrachten wir die Ergebnisse der Tests zur Lokalisierung. Wir vergleichen die Klassifikatoren, die mit AdaBoost auf unterschiedlichen Merkmalsmengen trainiert wurden.

Zu Beginn zeigen wir in Abschnitt 7.1 Kaskaden, die auf den Basis Haar-Merkmalen mit 393 Ausprägungen trainiert wurden. Auf dieser Menge, die 27.000 Klassifikatoren ergeben, führen wir noch eine erschöpfende Suche für das AdaBoost-Training aus. Mit diesem und mit einer zufällig reduzierten Auswahl an Klassifikatoren vergleichen wir dann die Kaskaden, die auf den durch Preboost reduzierten Mengen trainiert wurden. Wir können zeigen, daß die Ergebnisse der Kaskaden dicht beieinander liegen, wobei die "Preboost-Kaskaden" insbesondere bei der Stärke der Reduzierung beachtliche Ergebnisse produzieren. In den darauf folgenden Kapiteln gehen wir auf ein paar Eigenheiten und Probleme der Preboost-Kaskaden ein und zeigen, wie diese abzumildern sind.

In dem Abschnitt 7.2 erhöhen wir die Ausgangsmenge an Merkmalen. Damit dauert ein Kaskaden-Training nach Viola und Jones mit einer erschöpfenden Suche so lange, dass wir es nicht mehr durchgeführt haben. Die Ergebnisse der Preboost-Kaskaden sind jetzt bereits deutlich besser als bei den letzten, auf einer geringeren Menge mit der erschöpfenden Suche trainierten Kaskaden. Hier zeigt sich bereits deutlich der Vorteil des Preboostings.

Mit Binning- und FuzzyBinning-Klassifikatoren wiederholen wir im Abschnitt 7.3 die Experimente und können zeigen, dass sich die Ergebnisse bestätigen.

Als weitere Bestätigung, dass wir mit unserem Preboosting gute Klassifikatoren-Mengen erhalten, prüfen wir, ob auch die Joint Haar-like Feature, die mit solchen Mengen gebildet werden, gute Ergebnisse erzielen.

Schließlich zeigen wir in dem Abschnitt 7.6 die Ergebnisse einer erweiterten Variante des Preboost, in der einige Klassifikatoren während des Preboostings neu berechnet werden.

Durch die starke Reduktion ist das nachfolgende Kaskaden-Training erheblich schneller, unabhängig davon, wie groß die Ausgangsmenge war, da die minimale Größe nur von der Güte der einzelnen Klassifikatoren abhängt. Der Mehraufwand liegt hier nur im Preboosting.

So zeigen wir im letzten Abschnitt 7.7 ein paar Varianten von Kaskaden, die mit anderen Mengen und Klassifikatoren trainiert wurden als wir sie bis dahin betrachtet haben.

Wir nutzen für die Kaskaden eine bestimmte Namensgebung, die es erlaubt, wichtige Daten aus dem Namen abzuleiten. So beginnen wir den Namen mit der Haar-Merkmalsmenge, gefolgt von der Art der Reduzierung und der Anzahl der Klassifikatoren die für das Training zur Verfügung standen.

Z.B. steht "HF3081 Preboost K150 VJ" für eine Kaskade, für deren Training die Haar-Merkmalsmenge "Basis Haar-Merkmale 3081" mit "Preboost" auf 150 ("K150") Klassifikatoren reduziert wurde und dann mit AdaBoost nach Viola und Jones ("VJ") trainiert wurde.

Im Laufe dieser Arbeit haben wir mehr Kaskaden trainiert und getestet, als wir hier vorstellen. Die Kaskaden, die wir hier präsentieren, sind die Auswahl, die die Ergebnisse unsere Experimente am besten repräsentieren.

7.1 Vergleich mit kleiner Merkmalsmenge

In diesem Kapitel vergleichen wir Kaskaden, die mit reduzierten Mengen trainiert wurden und Kaskaden, die mit einer erschöpfenden Suche trainiert wurden. Um die erschöpfende Suche in annehmbarer Zeit durchführen zu können, haben wir die Haar-Merkmalsmenge "Basis Haar-Merkmale 393" gewählt, womit AdaBoost mit der erschöpfende Suche in jeder Runde 27.000 Klassifikatoren testen muss (vergleiche 6.2). Die Merkmals-Mengen, mit denen die anderen Kaskaden trainiert werden, werden zum einen durch unser Preboost Verfahren reduziert und zum anderen durch eine zufällige Auswahl.

Unsere Ergebnisse zeigen, dass die Preboost-Kaskaden ähnliche Ergebnisse erzielen wie die Kaskaden, die nach dem Verfahren von Viola und Jones mit einer erschöpfenden Suche trainiert werden. Dafür benötigen die Preboost-Kaskaden nur einen geringen Teil der Zeit für das Training.

In den darauf folgenden Kapiteln 7.1 erläutern wir ein paar Eigenheiten und Nachteile der Proboost-Kaskaden und wie diese abzuschwächen sind.

Auf die Variante, auf das Neu-Trainieren der Klassifikatoren in den Boosting-Iterationen zu verzichten, wie wir es beim Preboost machen, gehen wir in 7.1 ein und zeigen, daß die so erzielten Ergebnisse nicht befriedigen.

Preboost im Vergleich mit VJ 393 und VJ 160

393 steht, wie im Abschnitt 6.2 beschrieben, für die Menge von 393 Ausprägungen der Basis Haar-Merkmale.

Die anderen Kaskaden wurden mit der durch Preboost (siehe 4.4) reduzierten Merkmalsmenge bzw. den daraus erstellten Klassifikatoren trainiert.

Für alle Klassifikatoren gilt, dass nicht die Differenz der Rechtecksummen, sondern der Quotient berechnet wurde, um den Wert des Haar-Merkmals zu ermitteln. Auf die Varianznormalisierung wurde verzichtet.

Die VJ-Kaskaden

Für das Verfahren von Viola und Jones werden für jedes der Haar-Merkmale die entsprechenden Klassifikatoren erstellt. Hierfür werden in X-Richtung auf jedes 2-te Pixel und in Y-Richtung auf jedes Pixel nacheinander die Haar-Merkmale gelegt und für diese Positionen ein Schwellwert-Klassifikator trainiert. Ohne Einschränkung ergeben das 27.000 Klassifikatoren, die in jeder Runde des AdaBoost-Algorithmus trainiert und getestet werden.

Der Klassifikator, der auf diese Weise trainiert wurde, trägt in den Grafiken den Namen “HF393 K27T VJ”. Für den “HF160 K10T VJ” wurden lediglich 160 Ausprägungen der Basis Haar-Merkmale verwendet, womit die Suche nur über etwas mehr als 10.000 Klassifikatoren durchgeführt werden musste.

Die “HF393 K27T VJ”-Kaskade wurde so trainiert, dass sie mit einer Falsch-Positive-Rate von 0,45 gestartet ist und diese über 8 Kaskadenknoten auf eine maximale Falsch-Positiv-Rate von 0,001 reduziert wurde. Die “HF160 K10T VJ”-Kaskade startete mit einer Falsch-Positiv-Rate von 0,5, die in 10 Schritten auf 0,001 gesenkt wurde.

Die Preboost-Kaskaden

Die drei hier betrachteten “Preboost-Kaskaden” wurden mit den durch Preboost reduzierten Mengen trainiert. Ansonsten geschah das Training der Kaskaden analog zu den “VJ-Kaskaden”.

Wir rufen uns hier kurz den Kern des Preboost-Verfahrens in Erinnerung, der in 4.4 ausführlich dargestellt ist. Der Reduktionsalgorithmus teilt die Ursprungsmenge in N Teile mit je M Klassifikatoren. Für jeden dieser N Teile werden nun diese M Klassifikatoren auf K Klassifikatoren reduziert. Die K Klassifikatoren sind genau die Klassifikatoren, die durch den Boosting-Algorithmus zu “seinem“ Ensemble hinzugefügt wurden. Dieser Vorgang kann wiederholt werden, nachdem die bereits gewählten Klassifikatoren aus dem N-ten Teil entfernt wurden. Nachdem alle N Teile abgearbeitet wurden, werden die N Ergebnismengen zu einer Menge mit dann $N \cdot K$ Klassifikatoren zusammengefügt. Der Vorgang wird abgebrochen, wenn das erste Mal die Ergebnismenge nicht mehr in kleinere Mengen zerlegbar ist.

Die hier vorgestellten Klassifikatoren wurden mit den "Basis Haar-Merkmalsmenge 393" trainiert, also 393 Ausprägungen der Basis Haar-Merkmale, für die auf allen Positionen in den Beispielbildern ein Schwellwert-Klassifikator berechnet wurde (siehe auch 6.2). In der Summe ergibt das 27.000 Klassifikatoren, deren Fehler auf der Untermenge von 610 positiven und 1.500 negativen Trainingsbildern (siehe 6.3) 0,45 nicht überschreitet.

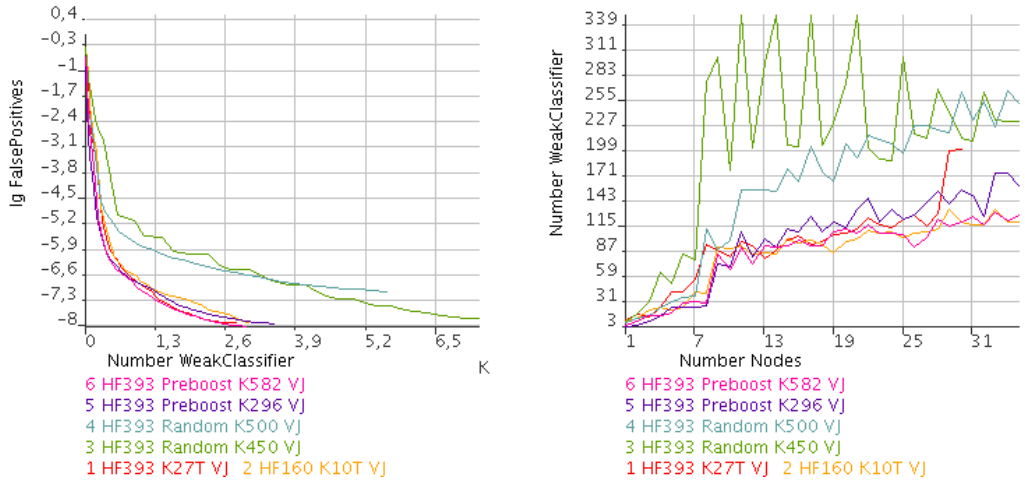


Abbildung 7.1: Vergleich AdaBoost mit erschöpfenden Suche, zufälliger Auswahl und Reduktion durch Boosting auf den Testbildern. Links zu sehen ist der Logarithmus der Falsch-Positiv-Rate, ermittelt mit der Beispielbilder-Menge "Standard Testbilder 37". Rechts ist die Anzahl der Schwellwert-Klassifikatoren gegen die Kaskaden-Knoten aufgetragen.

Unterschiede zeigen sich in der Grafik (7.1 rechts), in der die Anzahl der Schwellwert-Klassifikatoren gegen die Kaskaden-Knoten aufgetragen ist. Die Kaskade "HF393 Random K450 VJ" benötigt deutlich mehr Schwellwert-Klassifikatoren, um das Ziel zu erreichen, während die Ergebnisse der Detektionsrate und der Falsch-Positiv-Rate dicht bei denen der anderen Kaskaden liegen (7.4).

Wie in der Tabelle 7.4 zu sehen, liegen alle Kaskaden dicht beieinander. Die "HF393 Random K500 VJ" hat mehr Falsch-Positive, aber auch eine höhere Detektionsrate.

Die Preboost Kaskaden weisen im Vergleich zu der erschöpfenden Suche sowohl ähnliche Detektionsraten als auch eine ähnliche Anzahl Falsch-Positiver. Bei der Kaskade "HF393 K27T VJ" ist zu beachten, daß diese aus zeitlichen Gründen nur bis zum 30ten Kaskaden-Knoten trainiert wurde, während alle anderen bis zum 35ten Kaskaden-Knoten trainiert wurden.

Kaskaden werden selten exakt die gleiche Detektions- und Falsch-Positiv-Rate aufweisen. Da wir unser Augenmerk besonders auf die reduzierten Mengen richten, ist die Anzahl der Klassifikatoren in den Knoten der wichtigste Faktor, sobald die Detektions- und Falsch-Positiv-Rate nicht zu weit voneinander entfernt sind.

Klassifikator	DR	FP	Klassifikatoren	Knoten
HF393 K27T VJ	0.87	652	2019	25
HF393 K27T VJ	0.87	541	2788	30
HF160 K10T VJ	0.84	724	2407	30
HF160 K10T VJ	0.84	545	3013	35
HF393 Random K450 VJ	0.84	728	6126	30
HF393 Random K450 VJ	0.82	598	7293	35
HF393 Random K500 VJ	0.92	1461	4350	30
HF393 Random K500 VJ	0.92	1251	5577	35
HF393 Preboost K296 VJ	0.89	601	2712	30
HF393 Preboost K296 VJ	0.87	512	3491	35
HF393 Preboost K582 VJ	0.87	558	2342	30
HF393 Preboost K582 VJ	0.87	479	2963	35

Tabelle 7.1: Ergebnisse der Kaskaden, die mit 160 (HF160) bzw. mit 393 (HF393) Ausprägungen der Basis Haar-Merkmale trainiert wurden, getestet auf einer Menge von 37 Bildern mit 38 Kennzeichen. Etwas mehr als 1,5 Millionen Ausschnitte wurden klassifiziert. Bei 393 Ausprägungen der Haar-Merkmale ergeben sich 27.000 mögliche Klassifikatoren. Die Preboost Kaskaden wurden mit einer Menge von 296 (K296) und 582 (K582) Klassifikatoren trainiert. Die Random-Kaskaden wurden mit einer zufällig gewählten Menge von 500 bzw. 450 Klassifikatoren trainiert. FP=Anzahl Falsch-Positive, DR=Detektions-Rate, Klassifikatoren=Summe der (einfachen) Klassifikatoren bis zu dem entsprechenden Knoten.

Nach unseren Kriterien bewerten wir daher die Preboost-Kaskaden an dieser Stelle besser als die Random-Kaskaden, da letztere deutlich mehr Schwellwert-Klassifikatoren aufbringen müssen, um ihr Ziel zu erreichen.

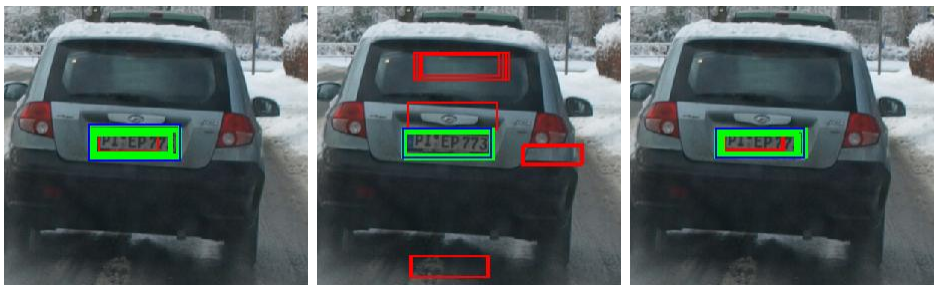


Abbildung 7.2: Beispiele der Ergebnisbilder der Kaskaden Preboost HF393 K27T VJ (links), HF393 Preboost HF393 Random K450 VJ (mitte) und K582 VJ (rechts).

Spitzen

Beim Trainieren von Kaskaden, die nur eine geringe Menge an Merkmalen zur Verfügung haben, treten leicht "Spitzen" bei der Anzahl der einfachen Klassifikato-

ren in den Knoten auf. Die Anzahl der Klassifikatoren in den jeweiligen Kaskaden-Knoten steigt nicht so gleichmäßig wie bei den Kaskaden, die mit einer großen Menge an Merkmalen trainiert wurden. Der "Vorrat" an Klassifikatoren für diese Kaskaden ist beschränkt und somit können diese an Stellen der Trainingsbilder geraten, die sie mit ihrem Vorrat an Klassifikatoren nur schwer abbilden können. Das Resultat dieser schwierigen "Passagen" ist, dass sie erheblich mehr Klassifikatoren benötigen, um das vorgegebene Ziel zu erreichen, was sich in den Spitzen abzeichnet.

Die Kaskaden dagegen, die eine große Anzahl an Merkmalen zur Verfügung haben, haben soviele Klassifikatoren, dass sie auch für solch schwierigen Aneinanderreihungen stets Klassifikatoren finden, die sich der Gegebenheit gut anpassen. Selbst bei der Kaskade "HF160 K10T VJ" in der Grafik 7.1, der lediglich 160 Ausprägungen der Basis Haar-Merkmale zur Verfügung standen, findet mit den daraus resultierenden 10.000 Schwellwert-Klassifikatoren für alle Trainingsbeispiele, die "passenden" Klassifikatoren für das Ensemble.

Unsere Trainingsausschnitte lagen in der Reihenfolge vor, wie sie aus den Bildern ausgeschnitten wurden. Ist also ein Bild mit besonders viel und schwer zu unterscheidendem Hintergrund dabei, so treten diese Ausschnitte gehäuft auf. Daraus leitete sich die Vermutung ab, daß es durchaus Sequenzen in den Trainingsausschnitten gab, die schwerer zu lernen waren als andere. Nachdem wir die Bilder vermischt hatten, wurden die Spitzen weniger, verschwanden aber nicht komplett.

Die Spitzen selbst beeinflussen das Ergebnis nur wenig. Wurden diese bei ein paar Test-Kaskaden gestutzt und mit den Originalen verglichen, änderte sich in den Testreihen die Detektionsrate nicht. Lediglich die Falsch-Positiven erhöhten sich leicht.

Die Spitzen sind auch abhängig von der geforderten Detektionsrate. Da sich die Gesamtdektektionsrate aus den Einzeldetektionsraten ergibt, ist es geboten, die geforderte Detektionsrate der einzelnen Knoten möglichst hoch anzusetzen (vergleiche auch 3.5). Der maximale Wert ist hier die 1,0. Relevante Unterschiede in der "Spitzenbildung" ergeben sich bei geforderten Detektionsraten von 0,99 bis 1,0.

Wenn wir diese Forderung abschwächen und lediglich eine Detektionsrate von 0,99 fordern statt 1,0 wählen, so verschwinden auch die Spitzen in den Kaskaden-Knoten. Die Kaskade "HF393 Preboost K227 DR099 VJ" in der Grafik 7.3 wurde z.B. mit einer geforderten Detektionsrate von 0,99 in jedem Knoten trainiert, während die Kaskade "HF393 Preboost K227 DR100 VJ" mit 1,0 trainiert wurde. Die restlichen Einstellungen für beide Kaskaden waren identisch. Es ist in der Grafik 7.3 und auch in der Tabelle 7.2 zu sehen, dass diese Änderung die Spitzen verschwinden läßt.

Wenn wir in den späteren Abschnitten zu mehr Haar-Merkmalen in den Grundmengen übergehen, werden alleine dadurch die Spitzen selten.

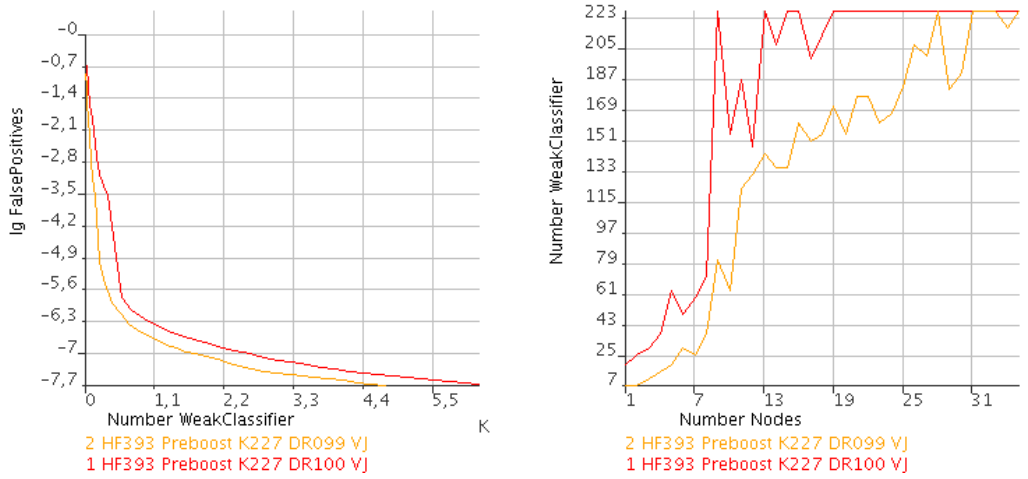


Abbildung 7.3: In der Grafik ist der Logarithmus der Falsch-Positiv-Rate gegen die Anzahl der Klassifikatoren in der Kaskade aufgetragen (links) und die Anzahl der Klassifikatoren in dem jeweiligen Knoten (rechts). Die Spitzen sind deutlich reduziert (links) und die Falsch-Positiven werden ähnlich gut reduziert (rechts).

Klassifikator	DR	FP	Klassifikatoren	Knoten
HF393 Preboost K227 DR100 VJ	0.87	945	3957	25
HF393 Preboost K227 DR100 VJ	0.87	683	6227	35
HF393 Preboost K227 DR099 VJ	0.89	945	2619	25
HF393 Preboost K227 DR099 VJ	0.87	663	4751	35

Tabelle 7.2: Vergleich zweier Kaskaden, die sich nur in der geforderten Detektionsrate pro Knoten unterschieden. Getestet auf einer Menge von 37 Bildern mit 38 Kennzeichen. FP=Anzahl Falsch-Positiver und DR=Detektions-Rate.

Kein Neutrainieren der Klassifikatoren in der Boosting-Iteration

In diesem Abschnitt zeigen wir, beispielhaft an einer Kaskade, die Ergebnisse, wie sich unsere Kaskaden verhalten, wenn wir während des Kaskadentrainings auf das Neu-Trainieren der Schwellwert-Klassifikatoren verzichten. Motiviert dies zu prüfen wurden wir durch die Arbeiten von Wu et al. zum "Direct Feature Selection" und durch die Analogie zu unserem Preboost, wo wir ebenso kein Neu-Trainieren der Klassifikatoren vornehmen. Dazu kommt, dass sich die Trainingszeiten für das Kaskadentraining merkbar verringern, würde das Verfahren auf diese Weise Erfolg haben.

In dem Kapitel 3.7.3 stellen wir das Ensemble-Lernverfahren "Direct Feature Selection" der Autoren Wu, Rehg und Mullin [19] vor. Ihre Kernthese ist, daß der Erfolg des Ensemble-Lernens im wesentlich an einer ausreichend großen Merkmalsmenge hängt. Bei ihrer Methode wird in jedem Durchgang derjenige

Klassifikator dem Ensemble hinzugefügt, der das Ergebnis am meisten verbessert. Die Trainingsbeispiele werden nicht neu gewichtet und der Klassifikator wird auch nicht in jeder Iteration neu trainiert. In ihrer Arbeit schreiben sie, daß sie dadurch um einen Faktor 100 schneller sind als Viola und Jones beim Trainieren einer Kaskade.

Da AdaBoost ganz anders als die Methode von Wu et al. arbeitet, können wir uns an dieser Stelle nicht mit ihnen vergleichen. Da uns die Ergebnisse im Zusammenhang mit dem Preboost-Reduktionsverfahren interessiert, werden wir auch nur Preboost-Kaskaden vergleichen.

Diesen Gedanken folgend, vergleichen wir, wie sich die finale Preboost-Kaskade verhält, wenn wir auch dort auf ein Neu-Trainieren der Klassifikatoren für das Ensemble verzichten.

Festzustellen ist, wie erwartet, daß sich das Lernen beschleunigt, wenn wir den Aufwand für das Wählen der einzelnen Klassifikatoren betrachten. Absolut sinkt die Trainingszeit kaum, teilweise steigt sie sogar, was an dem deutlichen Anstieg der Anzahl der benötigten Klassifikatoren liegt, wie wir folgend zeigen.

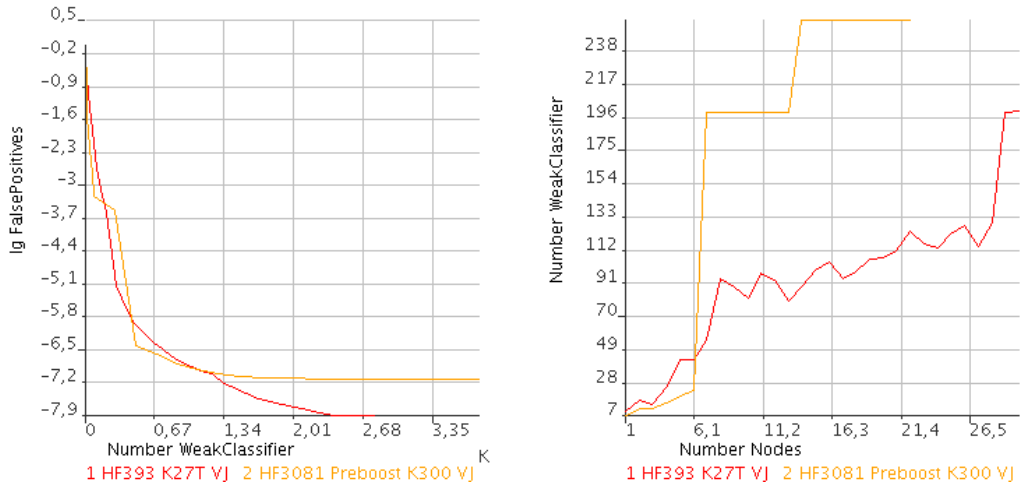


Abbildung 7.4: In der Grafik ist der Logarithmus der Falsch-Positiven gegen die Anzahl der Klassifikatoren in der Kaskade aufgetragen (links) und die Anzahl der Klassifikatoren in dem jeweiligen Knoten (rechts). Es ist deutlich zu sehen, daß eine Kaskade “HF3081 Preboost K300 VJ“, deren Klassifikatoren während des Trainingsprozesses nicht neu trainiert wurden erheblich mehr Klassifikatoren benötigen, als unsere Vergleichskaskade ”HF393 K27T VJ“.

Obwohl die Kaskade “HF3081 Preboost K300 VJ” eine größere Ausgangsmenge zur Verfügung hatte, benötigt sie erheblich mehr Klassifikatoren um die Falsch-Positiven zu senken und um die geforderten Detektionsraten beim Training zu erfüllen, wie in den beiden Grafiken gut zu sehen ist (siehe 7.4).

Klassifikator	DR	FP	Klassifikatoren	Knoten
HF393 K27T VJ	0.95	1677	924	15
HF393 K27T VJ	0.95	972	1433	20
HF393 K27T VJ	0.95	781	1668	22
HF3081 Preboost K300 VJ	0.84	1209	1993	15
HF3081 Preboost K300 VJ	0.84	1179	3278	20
HF3081 Preboost K300 VJ	0.84	1177	3792	22

Tabelle 7.3: Vergleich mit einer Kaskade, bei der auf ein neu trainieren in jeder AdaBoost-Runde verzichtet wurde. Getestet auf einer Menge von 37 Bildern mit 38 Kennzeichen. FP=Falsch-Positive-Rate und DR=Detektions-Rate.

Wie wir im nachfolgenden Abschnitt 7.2 sehen werden, verbessert die größere Ausgangsmenge "Basis Haar-Merkmale 3081", wie sie von der Kaskade "HF3081 Preboost K300 VJ" genutzt wurde, die Leistung einer Kaskade deutlich. Doch der negative Effekt, die einfachen Klassifikatoren nicht neu zu trainieren, so wie wir es beim Preboost tun, ist so groß, dass die Kaskade schlechter ist, als unsere, auf einer kleineren Menge trainierten Vergleichskaskade "HF393 K27T VJ".

Schlussbemerkung

Wir halten hier fest, das es deutliche Nachteile bringt, wenn im AdaBoost Lernprozess die Schwellwert-Klassifikatoren auf den aktuellen Gewichten nicht neu gelernt werden. Für den Anpassungsprozess an die zu lernenden Beispiele müssen erheblich mehr Klassifikatoren dem Ensemble zugefügt werden, als bei den Vergleichskaskaden, was auch das schnellere Lernen pro Schwellwert-Klassifikator wieder zu nichte macht.

Im Vergleich mit den anderen "Preboost Kaskaden" benötigt die nach Viola und Jones trainierte Kaskade eine etwas geringere Anzahl an Schwellwert-Klassifikatoren, um das Ensembles-Ziel zu erreichen. Die Detektionsrate und die Falsch-Positiv-Rate ist dabei vergleichbar. Nach unseren Kriterien bewerten wir die Kaskaden besser, die über die erschöpfende Suche trainiert wurden. Die erschöpfende Suche findet die besseren Klassifikatoren. Sie ist allerdings bereits bei einer kleinen Ausgangsmenge von Haar-Merkmalen so zeitintensiv, dass wir auf diese Weise nicht viele Experimente durchführen könnten.

Der große Vorteil bei den Preboost-Varianten ist, dass wir schnell viele unterschiedliche Kaskaden testen können. Variationen, die sonst durch die hohe Trainingszeit außerhalb des machbaren liegen.

Die Kaskaden, die mit einer zufällig reduzierten Menge trainiert werden, benötigen deutlich mehr Klassifikatoren als die Preboost-Kaskaden. Damit ist der Trainingsaufwand bei ihnen auch höher als bei den Preboost-Kaskaden.

Mit dieser Erkenntnis widmen wir uns einer Anzahl von Merkmalen, für die wir keine erschöpfende Suche mehr durchgeführt haben, da die Trainingszeit zu groß ist.

7.2 Vergrößerung der Grundmenge der Basis Haar-Merkmale

In diesem Kapitel stellen wir Ergebnisse vor, die zeigen, dass die Einbeziehung einer größeren Grundmenge an Haar-Merkmalen die Ergebnisse deutlich verbessern. Und dies auch, wenn wir die Grundmenge auf wenige hundert Klassifikatoren reduzieren, mit denen dann die Kaskade trainiert wird.

Als Grundmenge wurde die Menge "Basis Haar-Merkmale 3081" betrachtet (siehe auch 6.2), die 3.081 Ausprägungen der Basis Haar-Merkmale umfasst.

Für das Training der Preboost-Kaskanden wurden die 240.000 möglichen Klassifikatoren auf eine Anzahl von 2.547, 425 und 125 Klassifikatoren reduziert.

Auf das Training mit einer umfassenden Suche wurde verzichtet, da das Training einen zu langen Zeitraum eingenommen hätte. Dafür haben wir drei Kaskaden trainiert, bei denen wir die Merkmalsmenge durch eine zufällige Wahl reduziert haben. Bei den beiden Kaskaden "HF3081 Random K650 VJ" und "HF3081 Random K2500 VJ" wurden zu Beginn des Prozesses 650 bzw. 2.500 Klassifikatoren zufällig gewählt und diese Menge wurde während des Trainingsprozesses so beibehalten. Diese Kaskaden unterlagen damit den analogen Gegebenheiten wie die Preboost-Kaskaden, die ja auch mit einer unveränderlichen Menge arbeiten. Für die "HF3081 Random K200 VJ" dagegen wurde in jeder Trainingsrunde 200 Klassifikatoren erneut zufällig gewählt, so dass über alle Knoten betrachtet erheblich mehr verschiedene Klassifikatoren zur Verfügung standen.

Auch hier ist zu sehen, wie gut die Ergebnisse der Klassifikatoren sind, die auf der kleinen durch Preboost reduzierten Menge trainiert wurden. Es wird deutlich, wieviel besser die Ergebnisse werden, wenn wir die (Grund)Menge an Haar-Merkmalen erhöhen. Dies zeigt der Vergleich mit der Kaskade "HF393 K27T VJ" in der Tabelle 7.4, die immerhin mit einer erschöpfenden Suche auf 27.000 Klassifikatoren trainiert wurde.

Der Vergleich mit den Random-Kaskaden in der Grafik 7.5 zeigt, dass die über Preboost reduzierten Mengen, die besseren Klassifikatoren beinhalteten. Die Anzahl der Klassifikatoren pro Ensemble ist bei den Preboost-Kaskaden geringer. Lediglich die Kaskade "HF3081 Random K2500 VJ", die 2.500 Klassifikatoren zur Verfügung hatte, benötigt nur geringfügig mehr Klassifikatoren als ihre Preboost Vergleichskaskaden.

An zwei Kaskaden zeigen wir exemplarisch, wie sich die Anzahl der Falsch-Positiven noch weiter reduziert, in dem wir die gefundenen Regionen verbinden, die dicht

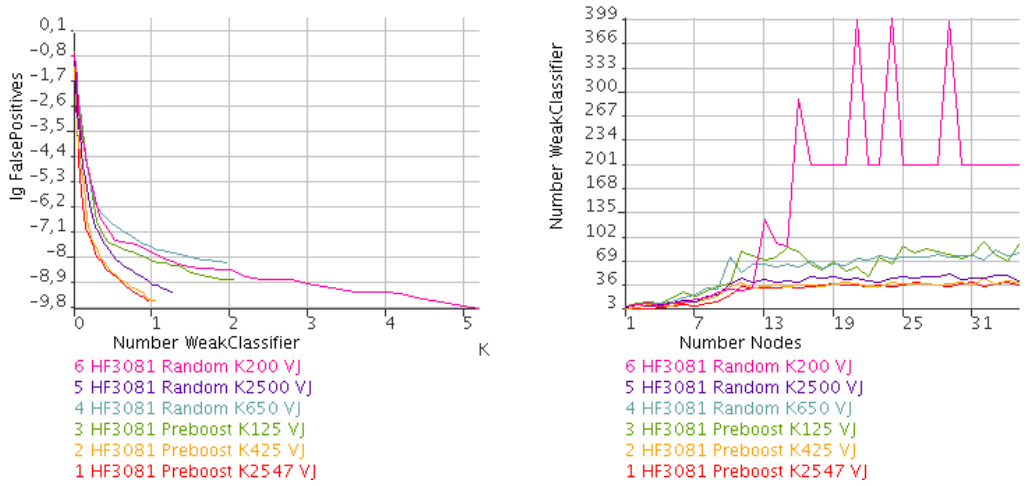


Abbildung 7.5: In der Grafik ist der Logarithmus der Falsch-Positiven gegen die Anzahl der Klassifikatoren in der Kaskade aufgetragen (links) erstellt auf der Testbildermenge "Standard 37". Rechts ist die Anzahl der Klassifikatoren gegen die Kaskaden-Knoten aufgetragen.

Klassifikator	DR	FP	Klassifikatoren	Knoten
HF393 K27T VJ	0.87	541	2788	30
HF160 K10T VJ	0.84	545	3013	35
HF3081 Preboost K2547 VJ	0.82	140	766	30
HF3081 Preboost K2547 VJ	0.79	104	949	35
HF3081 Preboost K425 VJ	0.82	144	842	30
HF3081 Preboost K425 VJ	0.82	102	1039	35
HF3081 Preboost K125 VJ	0.89	254	1642	30
HF3081 Preboost K125 VJ	0.89	214	2049	35
HF3081 Random K650 VJ	0.79	469	1578	30
HF3081 Random K650 VJ	0.79	407	1959	35
HF3081 Random K2500 VJ	0.84	181	1028	30
HF3081 Random K2500 VJ	0.82	138	1257	35
HF3081 Random K200 VJ	0.79	131	4196	30
HF3081 Random K200 VJ	0.74	77	5196	35

Tabelle 7.4: Ergebnisse der Kaskaden, die auf reduzierten Mengen trainiert wurden. Zum einen wurde mit Preboost reduziert und zum anderen mit eine zufälligen Auswahl. Die "HF393 K27T VJ" und "HF160 K10T VJ" aus dem vorigen Abschnitt, sind mit einer erschöpfenden Suche trainiert. Getestet wurde auf einer Menge von 37 Bildern mit 38 Kennzeichen. FP=Falsch-Positive-Rate und DR=Detektions-Rate.

beieinander liegen. Wir haben dabei nur solchen Regionen verschmolzen, die sich um einen Faktor von mindestens 0,6 überlappen. Wie in der Tabelle 7.5 zu sehen,

reduzieren sich die Falsch-Positiven noch einmal deutlich.

Klassifikator	DR	FP	Knoten
HF3081 Preboost K2547 VJ	0.76	33	35
HF3081 Preboost K462 VJ	0.76	35	35

Tabelle 7.5: Ergebnisse auf 37 Testbilder mit 38 Kennzeichen und 35 Kaskadenknoten

Abbildung 7.6: Zu sehen sind drei Ergebnisbilder der Preboost-Kaskaden, nach dem Verschmelzen von Regionen. Grün eingezeichnet sind die Treffer, gelb sind die als positiv Klassifizierten Ausschnitte, die zu einer größeren Region verschmolzen wurden und rot sind die Falsch-Positiven.

Schlussbemerkung

Wir haben die Grundmenge an Haar-Merkmalen erhöht, mit dem Effekt, daß die darauf trainierten Kaskaden bessere Detektionsraten und bessere Falsch-Positiv-Raten zeigen. Dafür benötigen sie weniger Klassifikatoren als die Kaskaden des vorangegangenen Abschnittes 7.1. Im Vergleich zeigt sich auch hier, wie in dem vorangegangenen Abschnitt 7.1, dass Kaskaden, die mit der durch Preboost reduzierten Menge trainiert wurden, weniger Klassifikatoren benötigen als die Vergleichskaskaden, die mit einer zufällig reduzierten Menge trainiert wurden. Mit der Ausnahme der Kaskade "HF3081 Preboost K125 VJ" galt das auch, wenn die Anzahl der Klassifikatoren in der zufällig reduzierten Menge höher war.

7.3 Binning- und FuzzyBinnning-Klassifikatoren

Nach dem wir uns in den vorangegangen Kapiteln davon überzeugt haben, dass wir mit dem Preboost und größeren Ausgangsmerkmalsmengen bessere Ergebnisse erzielen als mit den Vergleichskaskaden und Preboost, betrachten wir nun weitere "einfache" Klassifikatoren.

Zum einen wollen wir überprüfen, ob das Preboost-Verfahren hier ebenso einen Vorteil bringt und zum anderen wollen wir analysieren, ob die Klassifikatoren bessere Ergebnisse erbringen als der Schwellwert-Klassifikator.

Dafür vergleichen wir erst 3 Binning-Klassifikatoren, die auf einer durch Preboost reduzierten Menge trainiert wurden und 3 Binning-Klassifikatoren, die mit einer zufällig reduzierten Menge trainiert wurden. Im zweiten Teil dieses Abschnittes wiederholen wir dies ebenso mit FuzzyBinning-Klassifikatoren.

Binning-Klassifikatoren

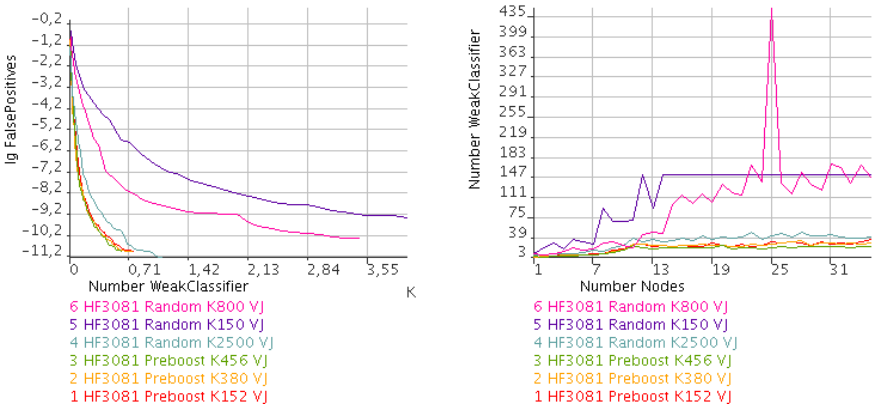


Abbildung 7.7: In der Grafik ist der Logarithmus der Falsch-Positiven gegen die Anzahl der Klassifikatoren in der Kaskade aufgetragen (links) und die Anzahl der Klassifikatoren in dem jeweiligen Knoten (rechts). Es ist gut zu sehen, daß die insbsondere die “Random 800” und die “Random 150” deutlich mehr Klassifikatoren benötigt, um die Falsch-Positiven zu senken.

Wenn wir die Ergebnisse in der Tabelle 7.11 und der Grafik 7.13 betrachten, ist deutlich der Unterschied bei der Anzahl der Klassifikatoren in den Kaskaden-Knoten, zu sehen.

Die Kaskaden, die mit einer zufälligen Reduktion der Merkmalsmenge trainiert wurden, benötigen mehr Binning-Klassifikatoren, um das vorgegebene Ziel zu erreichen, als diejenigen, die mit einer durch Preboost reduzierten Menge trainiert wurden. Die Random-Kaskaden benötigen teils erheblich mehr Klassifikatoren, um die Falsch-Positiven zu senken.

Die Ergebnisse selber, d.h. die Detektionsraten und die Anzahl der Falsch-Positiven, liegen meist nicht soweit auseinander.

FuzzyBinning-Klassifikatoren

Links in der Grafik **??** zeigt sich, dass zwei der drei Random-Kaskaden mehr Klassifikatoren benötigen, um die Falsch-Positiven zu senken. Der Unterschied zu der Kaskade “HF3081 Random K1500 VJ” fällt gering aus.

Klassifikator	DR	FP	Klassifikatoren	Knoten
HF3081 Preboost K152 VJ	0.74	49	463	25
HF3081 Preboost K152 VJ	0.71	14	463	25
HF3081 Preboost K152 VJ	0.68	25	757	35
HF3081 Preboost K152 VJ	0.65	7	757	35
HF3081 Preboost K380 VJ	0.61	45	457	25
HF3081 Preboost K380 VJ	0.57	13	457	25
HF3081 Preboost K380 VJ	0.55	25	745	35
HF3081 Preboost K380 VJ	0.55	4	745	35
HF3081 Preboost K456 VJ	0.63	53	389	25
HF3081 Preboost K456 VJ	0.63	20	389	25
HF3081 Preboost K456 VJ	0.58	25	615	35
HF3081 Preboost K456 VJ	0.57	10	615	35
HF3081 Random K2500 VJ	0.71	35	679	25
HF3081 Random K2500 VJ	0.65	16	679	25
HF3081 Random K2500 VJ	0.66	19	1087	35
HF3081 Random K2500 VJ	0.57	9	1087	35
HF3081 Random K150 VJ	0.82	245	2506	25
HF3081 Random K150 VJ	0.73	83	2506	25
HF3081 Random K150 VJ	0.74	125	4006	35
HF3081 Random K150 VJ	0.65	54	4006	35
HF3081 Random K800 VJ	0.79	143	1989	25
HF3081 Random K800 VJ	0.68	54	1989	25
HF3081 Random K800 VJ	0.68	48	3290	34
HF3081 Random K800 VJ	0.6	24	3290	34

Tabelle 7.6: Ergebnisse der Binning-Kaskaden, die mit 3081 (HF3081) Ausprägungen der Basis Haar-Merkmale trainiert wurden, getestet auf einer Menge von 37 Bildern mit 38 Kennzeichen. FP=Falsch-Positive-Rate und DR=Detektions-Rate. Der Name gibt wieder, ob mit "Preboost" oder "Random" die Vorauswahl getroffen wurde und der Faktor "KN" gibt die Anzahl der Klassifakoren an, die für das Training zur Vefügung standen. Jede zweite Zeile erhält das Ergebnis, wenn die Trefferflächen zusammengefügt werden. Es werden dabei nur solche Flächen vereinigt, die sich mehr als 0.5 überlappen.

Schlussbemerkung

Die Trainingszeiten und die Erkennungsgeschwindigkeiten sind bei den meisten Preboost-Varianten besser. Bei den Trainingszeiten benötigt selbst der "HF3081 Random K150 VJ" die dreifache Zeit für das Training wie der "HF3081 Preboost K152 VJ". Die Random-Kaskaden mit einer größeren Menge an Klassifikatoren benötigen im Vergleich mit den Preboost-Kaskaden die 5-10 fache Zeit für das Training. In den Detektionsraten und den Falsch-Positiv-Raten liegen die verschiedenen Kaskaden alle nicht sehr weit auseinander. Nur dann, wenn wir die Anzahl der Klas-

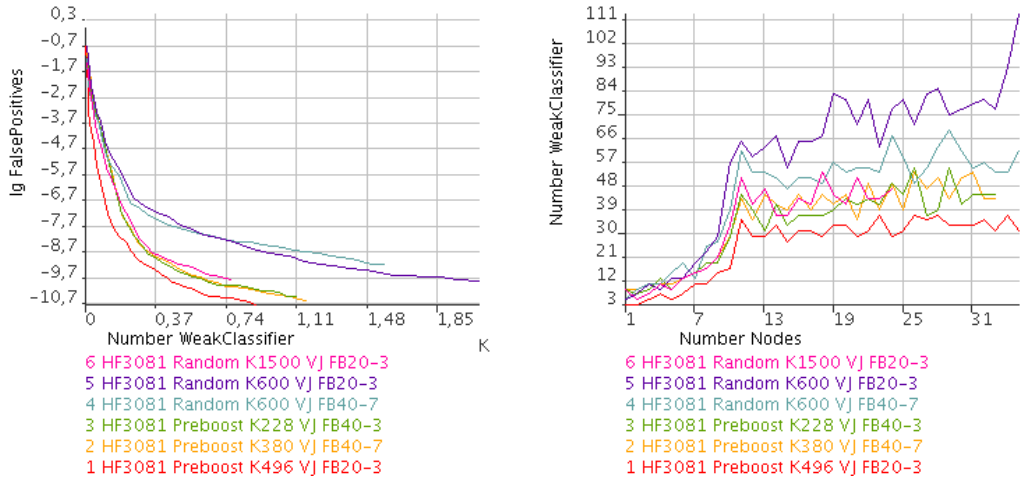


Abbildung 7.8: In der Grafik ist der Logarithmus der Falsch-Positiven gegen die Anzahl der Klassifikatoren in der Kaskade aufgetragen (links) und die Anzahl der Klassifikatoren in dem jeweiligen Knoten (rechts). Es ist gut zu sehen, daß die insbsondere die “Random 800” und die “Random 150” deutlich mehr Klassifikatoren benötigt, um die Falsch-Positiven zu senken.

Klassifikator	DR	FP	Klassifikatoren	Knoten
HF3081 Preboost K496 VJ FB20-3	0.66	58	551	25
HF3081 Preboost K496 VJ FB20-3	0.61	31	893	35
HF3081 Preboost K380 VJ FB40-7	0.68	62	775	25
HF3081 Preboost K380 VJ FB40-7	0.68	37	1159	33
HF3081 Preboost K228 VJ FB40-3	0.71	67	743	25
HF3081 Preboost K228 VJ FB40-3	0.68	44	1105	33
HF3081 Random K600 VJ FB40-7	0.82	309	989	25
HF3081 Random K600 VJ FB40-7	0.76	147	1565	35
HF3081 Random K600 VJ FB20-3	0.74	147	1229	25
HF3081 Random K600 VJ FB20-3	0.71	77	2063	35
HF3081 Random K1500 VJ FB20-3	0.71	76	807	25
HF3081 Random K1500 VJ FB20-3	0.66	55	1056	30

Tabelle 7.7: Ergebnisse der FuzzyBinning-Klassifikatoren, auf 37 Testbilder mit 38 Kennzeichen. FB in den Namen steht für FuzzyBinning, die Zahl nach dem FB ist die Anzahl der Bins und die darauffolgende Zahl bezeichnet die Anzahl der Nachbar-Bins, die mit betrachtet werden. Die Ergebnisse basieren auf 37 Testbilder mit 38 Kennzeichen.

sifikatoren mit betrachten, sind die Preboost-Kaskaden besser zu bewerten.

Wie in den vorangegangen Kapiteln zeigen die Ergebnisse, dass die mit einer durch Preboost reduzierten Menge trainierten Kaskaden in den meisten Fällen weniger Klassifikatoren benötigen als die Kaskaden, die mit einer zufällig reduzierten Menge

trainiert wurden. Die Ergebnisse selber liegen, gerade wenn man die jeweils besten betrachtet, dicht beieinander. Der Aufwand für das Training und für den Erkennungsprozess ist für die Preboost-Kaskaden meist deutlich geringer.

Wenn wir die Binning-Klassifikatoren und FuzzyBinning-Klassifikatoren mit einander vergleichen, die mit einer durch Preboost reduzierten Menge trainiert wurden, fällt auf, dass die FuzzyBinning-Kaskaden mehr Klassifikatoren benötigen als die Binning-Kaskaden. Die Detektions- und Falsch-Positiv-Raten der FuzzyBinning-Klassifikatoren wiegen nach unserer Einschätzung diesen Nachteil nicht auf. Mit den Kaskaden, die wir trainiert haben, haben sich die Erwartungen nicht bestätigt, die wir in die FuzzyBinning-Klassifikatoren hatten.

7.4 Joint Haar-like Feature

Die Joint Haar-like Feature kombinieren mehrere binäre Klassifikatoren zu einem neuen Merkmal. Auf diesem Merkmal wird entsprechend der "Bayes decision rule" ein Klassifikator trainiert, wie wir ihn in Kapitel 3.9.2 beschrieben haben.

In dem ersten Teil dieses Abschnittes vergleichen wir die Joint Haar-like Feature mit unserer Schwellwert Kaskade "HF393 VJ", die noch mit einer erschöpfenden Suche trainiert wurde. Wir wollen prüfen, ob wir die Ergebnisse der Autoren Wu et al., dass die Joint Haar-like Feature weniger Schwellwert-Klassifikatoren benötigen, bestätigen können.

Die anderen vier Kaskaden wurde mit Joint Haar-like Feature trainiert. Die Joint Haar-like Feature setzen sich in allen Beispielen aus 3 Schwellwert-Klassifikatoren zusammen.

Eine, die "HF393 K27T Joint VJ" wurde auf der vollen Menge der zur Verfügung stehenden Schwellwert-Klassifikatoren erstellt. Für jedes Joint Haar-like Feature mussten also dreimal 27.000 Schwellwert-Klassifikatoren getestet werden.

Für die verbleibenden Kaskaden wurde auf eine Menge an Klassifikatoren zurückgegriffen, die durch Preboost reduziert wurde. Bei zwei Kaskaden wurde die Menge "Basis Haar-Merkmale 393" zum reduzieren durch Preboost genutzt und für eine weitere, die "Basis Haar-Merkmale 3081", so dass deutlich mehr Haar-Merkmale zur Verfügung standen.

In der Tabelle 7.8 sind die Detektionsraten und die Anzahl der Falsch-Positiven auf der 37 Bilder umfassenden Testmenge zu sehen. Wenn man die Ergebnisse für die beiden Kaskaden vergleicht, die ohne die Reduzierung der Klassifikatoren trainiert wurden, sieht man das Ergebnis von Mita et al. durchaus bestätigt. In der Grafik 7.9 ist ersichtlich, dass die Kaskade, die die Joint Haar-like Feature nutzt, weniger Schwellwert-Klassifikatoren benötigt, um die Falsch-Positiven zu senken, als die Kaskade "HF393 K27T VJ", dessen Ensemble direkt aus Schwellwert-Klassifikatoren zusammengesetzt ist.

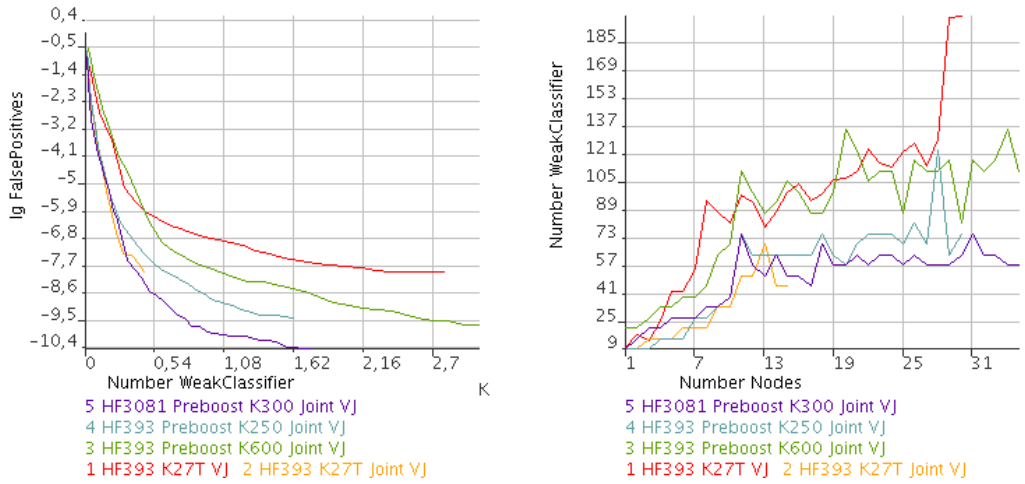


Abbildung 7.9: In der Grafik ist der Logarithmus der Falsch-Positiven gegen die Anzahl der Klassifikatoren in der Kaskade aufgetragen (links) und die Anzahl der Klassifikatoren in dem jeweiligen Knoten (rechts). Auf der Grafik ist gut zu sehen, daß die Kaskaden, die die Joint Haar-like Feature verwenden weniger Schwellwert-Klassifikatoren benötigen, um die Falsch-Positiven zu senken.

Klassifikator	DR	FP	Klassifikatoren	Knoten
HF393 K27T VJ	0.87	541	2788	30
HF393 K27T Joint VJ	0.89	548	453	15
HF393 Preboost K600 Joint VJ	0.89	694	885	15
HF393 Preboost K600 Joint VJ	0.74	92	3057	35
HF393 Preboost K250 Joint VJ	0.89	732	525	15
HF393 Preboost K250 Joint VJ	0.82	116	1620	30
HF3081 Preboost K300 Joint VJ	0.84	250	549	15
HF3081 Preboost K300 Joint VJ	0.74	43	1743	35

Tabelle 7.8: Ergebnisse der Kaskaden, die Joint Haar-like Feature nutzen, getestet auf einer Menge von 37 Bildern mit 38 Kennzeichen. FP=Falsch-Positive-Rate und DR=Detektions-Rate.

Wenn man die beiden Joint-Kaskaden vergleicht, die mit einer Ausgangsbasis von 393 Basis Haar-Merkmalen trainiert wurde, sieht man, dass auch hier, analog zu den Ergebnissen im Abschnitt 7.1, die Ergebnisse ohne das Preboost etwas besser sind. Die Kaskade, die mit Joint Haar-like Feature und der erschöpfenden Suche trainiert wurde, erreicht etwas bessere Ergebnisse als ihre Preboost-Varianten. Allerdings betrug die Trainingszeit der “HF393 K27T Joint VJ“-Kaskade bis zum 15-ten Knoten mehr als das 10-fache dessen, was z.B. die ”HF3081 Preboost K300 Joint VJ“-Kaskade bis zum 15-ten Knoten benötigte. Und da in jedem Knoten die Anzahl der negativen Beispiele gesteigert werden, dauern die später gelernten Knoten noch

einmal länger.

Deutlich fallen die Unterschiede aus, wenn wir die Kaskade betrachten, die mit 3081 Basis Haar-Merkmalen trainiert wurde. Schon mit 15 Knoten erreicht sie knapp die Ergebnisse der im Kapitel 7.2 trainierten Kaskaden, die direkt mit den Schwellwert-Klassifikatoren trainiert wurden.

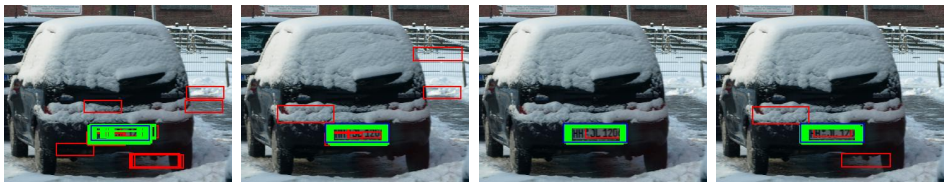


Abbildung 7.10: Beispielergebnisse der Kaskaden HF393 K27T VJ (links), HF393 K27T Joint VJ (mitte-links), HF393 Preboost K250 Joint VJ (mitte-rechts), HF393 Preboost K600 Joint VJ (rechts) auf den Testbildern "Standard 37" mit 37 Bildern und 38 Kennzeichen.

In dem zweiten Teil diesen Abschnitts vergleichen wir Kaskaden, die alle mit reduzierten Mengen trainiert wurden. Als Grundlage diente die Menge "Basis Haar-Merkmale 3081".

Drei Kaskaden wurden mit Mengen trainiert, die mit Preboost reduziert wurden und zwei Kaskaden mit Mengen, die durch zufällige Wahl reduziert wurden. Für die "Random-Kaskaden" wurde die Menge für jeden Knoten neu gezogen, während die Menge für die Preboost-Kaskaden gleich blieb. So wurde z.B. für die Kaskade "HF3081 Random K250 Joint VJ" über alle Knoten $250 \cdot 35 = 8750$ Klassifikatoren aus der Gesamtmenge zufällig ausgewählt.

Klassifikator	DR	FP	Klassifikatoren	Knoten
HF3081 Preboost K100 Joint VJ	0.79	235	381	15
HF3081 Preboost K100 Joint VJ	0.63	29	1257	35
HF3081 Preboost K500 Joint VJ	0.79	121	339	15
HF3081 Preboost K500 Joint VJ	0.66	7	1083	35
HF3081 Preboost K150 Joint VJ	0.87	285	345	15
HF3081 Preboost K150 Joint VJ	0.71	24	1137	35
HF3081 Random K450 Joint VJ	0.97	1826	357	15
HF3081 Random K450 Joint VJ	0.76	61	3366	32
HF3081 Random K250 Joint VJ	0.89	851	591	15
HF3081 Random K250 Joint VJ	0.76	128	2091	35

Tabelle 7.9: Ergebnisse der Kaskaden, die Joint Haar-like Feature nutzen, getestet auf einer Menge von 37 Bildern mit 38 Kennzeichen. FP=Falsch-Positive-Rate und DR=Detektions-Rate.

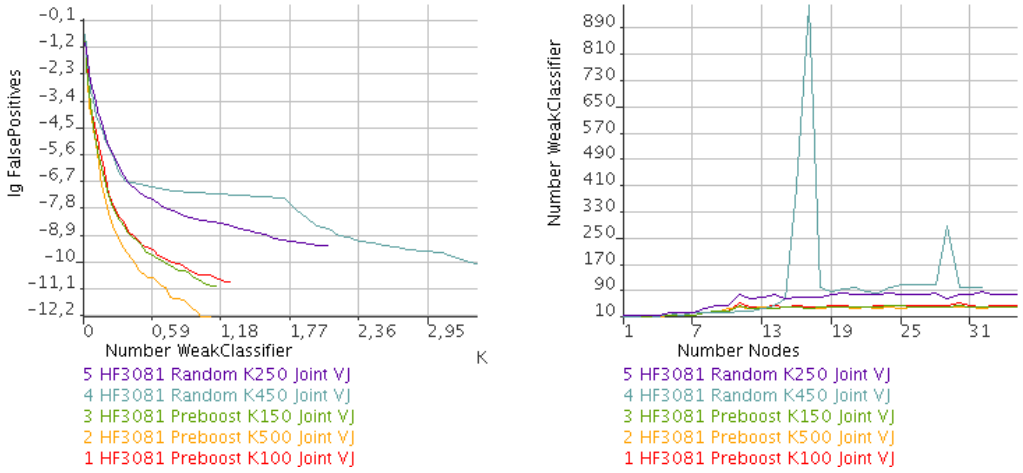


Abbildung 7.11: In der Grafik ist der Logarithmus der Falsch-Positiven gegen die Anzahl der Klassifikatoren in der Kaskade aufgetragen (links) und die Anzahl der Klassifikatoren in dem jeweiligen Knoten (rechts). Auf der Grafik ist gut zu sehen, daß die Kaskaden, die die Joint Haar-like Feature verwenden weniger Schwellwert-Klassifikatoren benötigen, um die Falsch-Positiven zu senken.

Abbildung 7.12: Beispielergebnisse der Kaskaden HF3081 Preboost K500 Joint VJ (links), HF3081 Preboost K150 Joint VJ (mitte), HF3081 Random K450 Joint VJ (rechts) auf den Testbildern "Parkplatz 43" mit 43 Bildern und 129 Kennzeichen. Insgesamt müssen etwas mehr als 15 Millionen Ausschnitte klassifiziert werden.

Bei den Joint Haar-like Feature nutzenden Kaskaden fällt der Unterschied zwischen den Preboost- und den Random-Varianten deutlicher aus als bei den Kaskaden, die direkt die Schwellwert-Klassifikatoren nutzen.

7.5 Fehlerfilter

An dieser Stelle werden wir wieder drei Kaskaden betrachten. Die erste wurde mit dem Verfahren von Viola/Jones trainiert und ist die selbe, die auch in den vorigen Beispielen gewählt wurde.

Bei den anderen Kaskaden wurden die einfachen Klassifikatoren vorberechnet und

über Preboost reduziert.

Für das Training der Preboost-Kaskaden wurde nicht die finale Menge an Klassifikatoren genommen, da die eh schon sehr gering ist, sondern eine früher anfallende Menge an Klassifikatoren. Bei den 3081er Kaskaden umfasste die Menge 6000 Klassifikatoren. Für den Trainingslauf kann ein Filter gesetzt werden, der Klassifikatoren verwirft, die einem bestimmten Kriterium, hier der Fehler, nicht entsprechen. Der Filter betrachtet die Klassifikatorenfehler, die beim Vorberechnen bestanden und nur auf einer kleinen Menge von Trainingsbildern ermittelt wurden.

Bei den folgenden Kaskaden wurde in dem einen Fall (ME02) nur Klassifikatoren weiter einbezogen, deren Fehler (auf der Trainingsmenge) geringer als 0,2 waren und in dem anderen Fall wurde ein Fehler geringer als 0,3 (ME03) erwartet.

Von den 6000 Klassifikatoren bleiben für den "Preboost VJ 3081 ME02 K272" noch 272 Klassifikatoren übrig. Eine Zahl, die, wie wir in Abschnitt 7.2 sehen, durchaus ausreicht, um gute Ensemble-Klassifikatoren zu erstellen. Die Kaskade "Preboost VJ 3081 ME03 K1971" erhält 1971 Klassifikatoren zum Erstellen eines Ensembles.

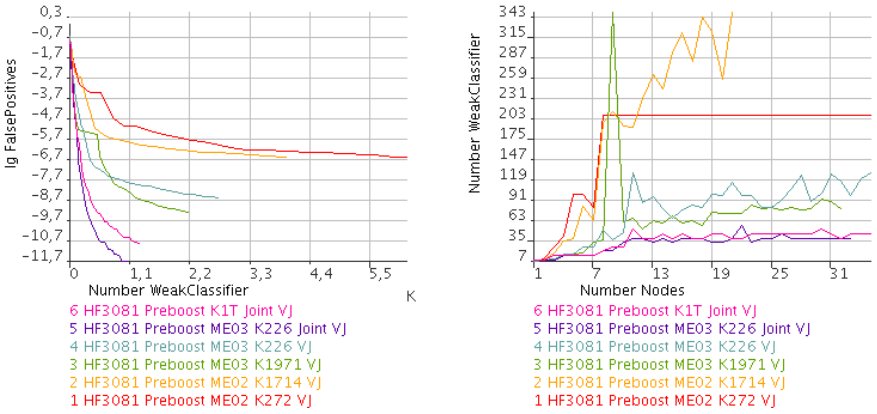


Abbildung 7.13: In der Grafik ist der Logarithmus der Falsch-Positiven gegen die Anzahl der Klassifikatoren in der Kaskade aufgetragen (links) und die Anzahl der Klassifikatoren in dem jeweiligen Knoten (rechts). Die Kaskaden wurden auf Mengen trainiert, die zusätzlich zu dem Preboosting über einen maximalen Fehler reduziert wurden. Wird der Fehler zu stark abgesenkt, erreicht die damit trainierte Kaskade keine guten Ergebnisse - auch dann nicht, wenn noch vergleichbar viele Klassifikatoren zur Verfügung stehen.

Insbesondere bei Joint Haar-like Feature ist zu sehen, dass es sogar von Vorteil sein kann, den maximalen Fehler leicht einzuschränken. Es ist aber zu bedenken, das dieser Fehlerwert mit 610 positiven und 1500 negativen Testbildern berechnet wurde. Werden andere positive und negative Testbilder verwendet, ergeben sich auch andere Fehlerwerte. Der hier gewählte Wert von 0,3 kann in anderen Variationen bereits zu schlechten Ergebnissen führen.

Eine zu starke Reduktion durch einen Fehlerfilter oder auf eine zu kleinen Ausgangsmenge führt zu keinen erfolgreichen Lösungen. Die verbleibenden Klassifikatoren

Klassifikator	DR	FP	Klassifikatoren	Knoten
HF3081 Preboost ME02 K272 VJ	0.95	5015	2021	15
HF3081 Preboost ME02 K272 VJ	0.89	1991	6161	35
HF3081 Preboost ME02 K1714 VJ	0.92	2838	2081	15
HF3081 Preboost ME02 K1714 VJ	0.92	2025	3955	21
HF3081 Preboost ME03 K1971 VJ	0.89	490	865	15
HF3081 Preboost ME03 K1971 VJ	0.82	135	2158	32
HF3081 Preboost ME03 K226 VJ	0.89	801	705	15
HF3081 Preboost ME03 K226 VJ	0.89	278	2719	35
HF3081 Preboost ME03 K226 Joint VJ	0.82	103	345	15
HF3081 Preboost ME03 K226 Joint VJ	0.63	12	1053	33
HF3081 Preboost K1T Joint VJ	0.79	235	381	15
HF3081 Preboost K1T Joint VJ	0.63	29	1257	35

Tabelle 7.10: Ergebnisse der Kaskaden, die auf Mengen trainiert wurden, die zusätzlich zum Preboosting durch einen Fehlerfilter reduziert wurden. Getestet auf einer Menge von 37 Bildern mit 38 Kennzeichen. FP=Falsch-Positive-Rate und DR=Detektions-Rate.

sind kaum in der Lage, sich an die Problemstellung anzupassen. In der Grafik **??** ist deutlich zu sehen, wie die beiden Kaskaden "Preboost VJ 393 ME03 K363" und "Preboost VJ 3081 ME02 K272" an Schwellwert-Klassifikatoren in jedem Knoten zunehmen.

Während der Boosting-Algorithmus arbeitet, zeigt sich, dass der einzelne Klassifikator dadurch an Bedeutung im Ensemble abnimmt, dass der Alpha-Wert, der sich durch den Fehler und die Gewichte berechnet, immer geringer wird. Diese Verringerung des Alpha-Wertes bedeutet letztlich, dass wiederholt kein Klassifikator gefunden werden kann, der in der Lage ist, genau die Beispiele zu erkennen, auf die der AdaBoost Algorithmus durch seine Gewichtung gerade die Aufmerksamkeit lenkt. Das Zufügen eines solchen Klassifikators bei solch einem geringen Alpha-Wert wird immer "belangloser", da dieser durch die Gewichtung $\alpha h(x)$ kaum noch Einfluss auf das Gesamtergebnis hat. $h(x)$ ist hier der Wert des Klassifikators h. (vergleiche auch Kapitel 3.1)

Nur die "Preboost VJ 3081 ME03 K1971" bringt sehr gute Ergebnisse. Diese Kaskade hat - mit Ausnahme einer Stelle - , stets genug Klassifikatoren in ihrem "Reservoir", um gute Ensemble-Klassifikatoren zu kreieren.

7.6 Preboosting Variante: "Beste neu trainieren"

In dem Abschnitt 7.1 haben wir gesehen, welche große Bedeutung dem Neu-Trainieren der einzelnen Klassifikatoren während des Kaskadentrainings zukommt.

So wollen wir in diesem Abschnitt der Frage nachgehen, ob ein Neu-Trainieren der Klassifikatoren in den einzelnen Boosting-Iterationen auch bei unserem Preboost Vorteile bringt.

Wenn wir alle Klassifikatoren neu berechnen würden, würde der zeitliche Aufwand hierfür so groß, daß der Vorteil des schnelleren Trainierens kaum noch vorhanden wäre.

Um dies zu umgehen, trainieren wir nur die besten Klassifikatoren neu.

D.h, wir berechnen die Fehler der Klassifikatoren auf den neuen Gewichten wie gehabt, nehmen nun aber nicht den besten, sondern merken uns die N Klassifikatoren mit dem geringsten Fehler. N ist dabei frei bestimmbar. Diese N Klassifikatoren trainieren wir mit den aktuellen Gewichten neu und wählen daraus den Klassifikator mit dem geringsten Fehler. Wir führen also lediglich ein partielles Neu-Trainieren durch.

Unser Gedanke ist, daß unter den N nicht neu trainierten besten Klassifikatoren auch "der" beste Klassifikator ist.

Wir haben Preboosting in drei Varianten durchgeführt. Es wurden 5, 15 und 50 Klassifikatoren mit den geringsten Fehlern neu berechnet. Aus den neu berechneten wurde dann für die reduzierte Menge der jeweils beste ausgewählt. Mit diesen Mengen wurden die Kaskaden "HF393 Preboost RB5 K296 VJ", "HF393 Preboost RB15 K276 VJ" und "HF393 Preboost RB50 K296 VJ" trainiert.

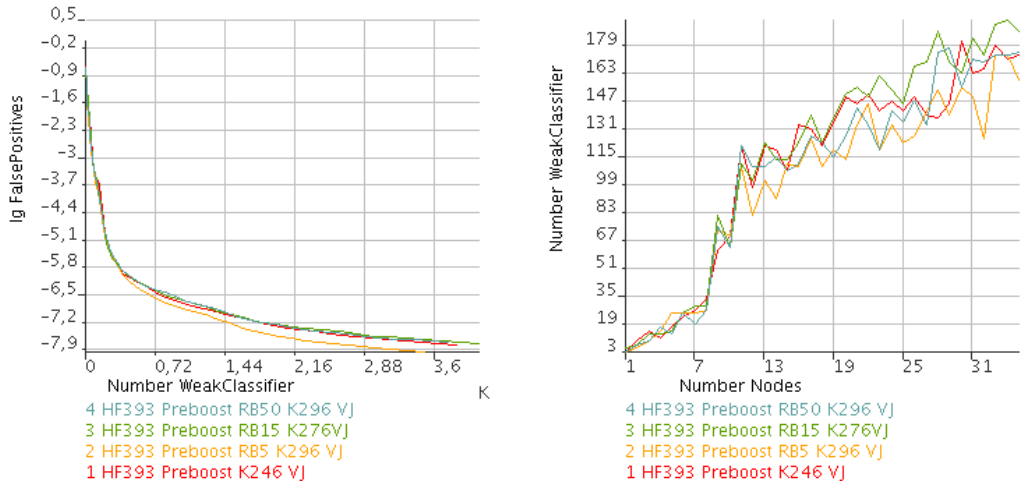


Abbildung 7.14: In der Grafik ist der Logarithmus der Falsch-Positiven gegen die Anzahl der Klassifikatoren in der Kaskade aufgetragen (links) und die Anzahl der Klassifikatoren in dem jeweiligen Knoten (rechts). Die verschiedenen Kaskaden in der Grafik wurden mit einer unterschiedlichen Anzahl an Klassifikatoren reduziert, die neu berechnet wurden.

Es lassen sich für die oben aufgeführten unterschiedlichen Varianten nicht die exakt gleichen Vorbedingugen für das Kaskadentraining schaffen. So variieren die reduzierten Mengen in ihrer Anzahl. Damit ist es schwer, nur den Faktor zu bewerten,

den wir gerade betrachten. Die Größe einer Menge hat Einfluss auf das Ergebnis, auch wenn wir gerade andere Faktoren betrachten. Die hier betrachteten Kaskaden haben alle eine ähnliche Anzahl an Klassifikatoren für das Training zur Verfügung, aber eben nicht die gleiche Anzahl.

In Betrachtung mit anderen Experimenten scheint es von Vorteil zu sein, eine kleine Anzahl an besten Klassifikatoren während des Preboostings neu zu trainieren. Der zeitliche Mehraufwand für eine kleine Anzahl, wie z.B. 5 und 15 neu trainierte Klassifikatoren, ist gering.

In der Grafik 7.14 verhalten sich die verschiedenen Kaskaden sehr ähnlich. Lediglich die Kaskade "HF393 Preboost RB5 K296 VJ" benötigt weniger Klassifikatoren zum Reduzieren der Falsch-Positiven.

Klassifikator	DR	FP	Klassifikatoren	Knoten
HF393 Preboost K246 VJ	0.95	1310	1502	20
HF393 Preboost K246 VJ	0.89	723	2978	30
HF393 Preboost K246 VJ	0.89	610	3829	35
HF393 Preboost RB5 K296 VJ	0.95	1224	1344	20
HF393 Preboost RB5 K296 VJ	0.89	601	2712	30
HF393 Preboost RB5 K296 VJ	0.87	512	3491	35
HF393 Preboost RB15 K276 VJ	0.95	1316	1514	20
HF393 Preboost RB15 K276 VJ	0.87	750	3132	30
HF393 Preboost RB15 K276 VJ	0.87	630	4059	35
HF393 Preboost RB50 K296 VJ	0.89	1445	1422	20
HF393 Preboost RB50 K296 VJ	0.87	760	2880	30
HF393 Preboost RB50 K296 VJ	0.87	646	3741	35

Tabelle 7.11: Ergebnisse der Kaskaden, die Joint Haar-like Feature nutzen, getestet auf einer Menge von 37 Bildern mit 38 Kennzeichen. FP=Falsch-Positive-Rate und DR=Detektions-Rate. Der Name gibt wieder, ob mit "Preboost" oder "Random" die Vorauswahl getroffen wurde und der Faktor "KN" gibt die Anzahl der Klassifakoren an, die für das Training zur Vefügung standen. Jede zweite Zeile erhält das Ergebnis, wenn die Trefferflächen zusammengefügt werden. Es werden dabei nur solche Flächen vereinigt, die sich mehr als 0.5 überlappen.

In den wenigen Experimenten, die wir mit Asymmetic AdaBoost durchgeführt haben, scheint das eine höhere Relevanz zu haben.

7.7 Verschiedene Varianten

In diesem Abschnitt stellen wir ein paar trainierte Klassifikatoren vor, die von den bisher betrachteten abweichen. Wir wollen hier zeigen, wie gut und schnell wir unsere Experimente durch die Zeit, die wir beim Training sparen, ausweiten können. Wir

wollen Kaskaden zeigen, die nicht mehr dem Vergleich mit anderen Verfahren dienen, sondern für die Suche nach guten Kaskaden für die Kennzeichen-Lokalisierung trainiert wurden.

Für die Kaskade “Preboost Joint Base+ VJ” haben wir unsere Basis Haar-Merkmale um Ecken und sich umfassende Haar-Merkmale erweitert (siehe Grafik 7.15).

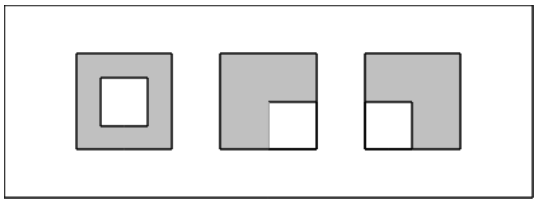


Abbildung 7.15: Weitere Haar-Merkmale, Ecken und Einfassungen.

FuzzyBinning-Klassifikatoren haben wir für die Kaskade “Preboost Joint 3xFuzzyBin VJ” anstatt der Schwellwert-Klassifikatoren genutzt.

Bei der Kaskade “Preboost Joint VarQuo VJ” haben wir zum einen unseren Schwellwert-Klassifikator genutzt, der wie sonst in den bisher gezeigten Experimenten den Quotienten der Intensitäten der Flächensummen berechnet. Zum anderen einen Schwellwert-Klassifikator, der die Varianz der Intensitäten in den Flächen der Haar-Merkmale berechnet. Für diese Varianz der Flächen wurde dann wieder der Quotient berechnet.

Die letzte Kaskade wurde mit fünf Schwellwert-Klassifikatoren trainiert, anstatt der sonst genutzten drei Schwellwert-Klassifikatoren.

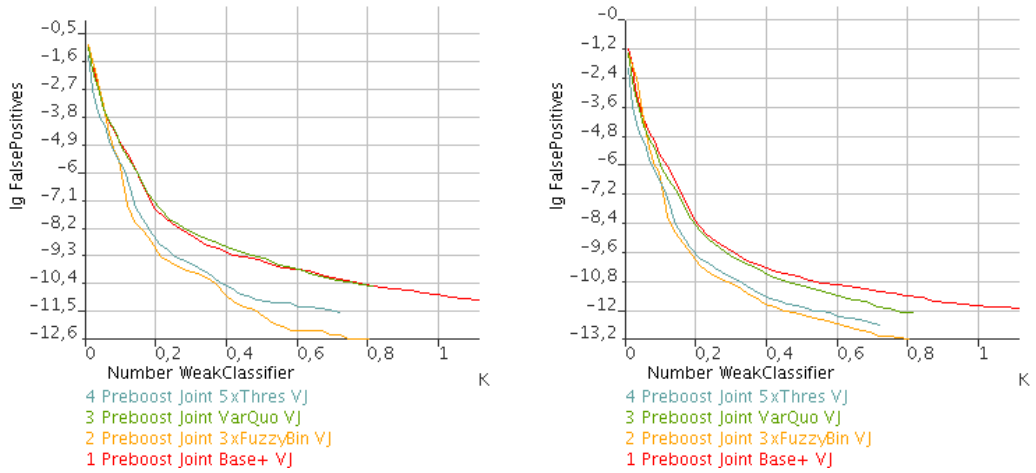


Abbildung 7.16: In der Grafik ist der Logarithmus der Falsch-Positiven gegen die Anzahl der Klassifikatoren in der Kaskade aufgetragen. Links erstellt auf der Testbildermenge “Standard 37“ und rechts auf der Testbildermenge “Parkplatz 43”

Die Kaskaden wurden mit Mengen von 150 bis 400 Klassifikatoren trainiert. Auf den Testbildern "Parkplatz 43“ haben sie im 35 Kaskaden-Knoten nur noch eine Detektionsrate von knapp 0,5. Dafür haben sie bei diesen Testbildern aber auch weniger als 100 Falsch-Positive bei etwas mehr als 15 Millionen Ausschnitten, die klassifziert worden sind (vergleicht Tabelle 7.12).

Abbildung 7.17: Beispiele der Kaskade "Preboost Joint Base+ VJ" auf der Testmenge "Parkplatz 43" (links), "Griechisch 67" (mitte) und "Standard 37" (rechts). Treffer sind grün eingezeichnet und Falsch-Positive rot.

7.8 Schlussbetrachtung

Um einen guten mit AdaBoost trainierten Klassifikator zu erstellen, ist es sehr von Vorteil, ein hohe Anzahl an Merkmalen zur Verfügung zu haben. Für ein schnelles Training können diese durch das Preboosting auf eine erheblich geringere Anzahl reduziert werden. Dass dabei das Preboosting auf einer geringen Anzahl an Traingsbildern seine Ergebnisse erzeugt, tut der Qualität der auf den Mengen trainierten Kaskaden keinen Abbruch. Auch dass nicht alle einfachen Klassifikatoren in jeder Runde auf den aktuellen Trainingsgewichten neu trainiert werden, stellt kein Problem dar. Das Wählen des besten Klassifikators unter Bezugnahme auf die aktuellen Trainingsgewichte reicht auch ohne eine neue Anpassung des Klassifikators aus, um eine gute Verteilung für alles Problemausprägungen zu bekommen.

Wenn wir über Preboost die Merkmale reduzieren, wird die Anzahl der Merkmale in der finalen Menge meist differieren. Da die Größe der Merkmalsmenge aber Einfluss hat auf die Güte der damit trainierten Kaskaden, behält ein Vergleich stets eine gewisse Unsicherheit.

Auch ist eine Bewertung schwierig, wenn sowohl die Detektionsrate und als auch die Falsch-Positiv-Rate höher als bei einer Vergleichskaskade ist, da es “quasie das natürliche Verhalten” darstellt. Deshalb haben wir als wesentlichen Punkt zur Beurteilung der Güte einer Kaskade die Anzahl der Klassifikatoren betrachtet. Denn je geringer die Anzahl der Klassifikatoren in den Ensemble der Kaskaden-Knoten ist, desto passender waren die einzelnen Klassifikatoren zu dem Problembereich. Damit läßt sich der Schluss zeihen, dass auch die Menge der Klassifikatoren, auf der diese Kaskaden trainiert wurden, ein höhere Güte hat.

Klassifikator	DR	FP	Klassifikatoren	Knoten	Testbilder
Preboost Joint Base+ VJ	0.84	195	345	15	Standard 37
Preboost Joint Base+ VJ	0.79	53	717	25	Standard 37
Preboost Joint Base+ VJ	0.79	24	1113	35	Standard 37
Preboost Joint 3xFuzzyBin VJ	0.71	88	261	15	Standard 37
Preboost Joint 3xFuzzyBin VJ	0.61	9	531	25	Standard 37
Preboost Joint 3xFuzzyBin VJ	0.55	5	801	35	Standard 37
Preboost Joint VarQuo VJ	0.76	346	297	15	Standard 37
Preboost Joint VarQuo VJ	0.66	75	615	25	Standard 37
Preboost Joint VarQuo VJ	0.61	41	813	31	Standard 37
Preboost Joint 5xThres VJ	0.89	172	231	15	Standard 37
Preboost Joint 5xThres VJ	0.74	27	459	25	Standard 37
Preboost Joint 5xThres VJ	0.66	14	717	35	Standard 37
Preboost Joint Base+ VJ	0.72	310	345	15	Griechisch 67
Preboost Joint Base+ VJ	0.55	75	717	25	Griechisch 67
Preboost Joint Base+ VJ	0.44	29	1113	35	Griechisch 67
Preboost Joint 3xFuzzyBin VJ	0.68	235	261	15	Griechisch 67
Preboost Joint 3xFuzzyBin VJ	0.55	40	531	25	Griechisch 67
Preboost Joint 3xFuzzyBin VJ	0.46	16	801	35	Griechisch 67
Preboost Joint VarQuo VJ	0.77	787	297	15	Griechisch 67
Preboost Joint VarQuo VJ	0.68	134	615	25	Griechisch 67
Preboost Joint VarQuo VJ	0.65	79	813	31	Griechisch 67
Preboost Joint 5xThres VJ	0.72	361	231	15	Griechisch 67
Preboost Joint 5xThres VJ	0.55	71	459	25	Griechisch 67
Preboost Joint 5xThres VJ	0.42	33	717	35	Griechisch 67
Preboost Joint Base+ VJ	0.58	702	345	15	Parkplatz 43
Preboost Joint Base+ VJ	0.5	204	717	25	Parkplatz 43
Preboost Joint Base+ VJ	0.44	102	1113	35	Parkplatz 43
Preboost Joint 3xFuzzyBin VJ	0.59	355	261	15	Parkplatz 43
Preboost Joint 3xFuzzyBin VJ	0.5	68	531	25	Parkplatz 43
Preboost Joint 3xFuzzyBin VJ	0.47	28	801	35	Parkplatz 43
Preboost Joint VarQuo VJ	0.64	912	297	15	Parkplatz 43
Preboost Joint VarQuo VJ	0.51	160	615	25	Parkplatz 43
Preboost Joint VarQuo VJ	0.5	83	813	31	Parkplatz 43
Preboost Joint 5xThres VJ	0.63	667	231	15	Parkplatz 43
Preboost Joint 5xThres VJ	0.46	118	459	25	Parkplatz 43
Preboost Joint 5xThres VJ	0.4	50	717	35	Parkplatz 43

Tabelle 7.12: Ergebnisse der Kaskaden auf 37 Testbilder mit 38 Kennzeichen (Standard 37), 67 griechischen Bildern mit 71 Kennzeichen (Griechisch 67) und 43 Parkplatzbildern mit 129 Kennzeichen (Parkplatz 43). Auf der Bildermenge “Standard 37“ mussten 1,5 Millionen Ausschnitte klassifiziert werden, auf der “Griechisch 67” 2,8 Millionen und auf der “Parkplatz 43“ 15 Millionen.

Erweiterungen und Fazit

8

8.0.1 Erweiterungen am Preboost-Verfahren

Nachfolgend führen wir ein paar Punkte auf, bei denen wir davon ausgehen, das sie Preboost-Verfahren verbessern könnten.

Preboost beschleunigen

Das Preboosting kann hervorragend den Speicher ausnutzen, so dass die Ursprungsmenge sehr schnell reduziert wird. Wie in Abschnitt 4.4 dargestellt, werden Teilbereiche herausgegriffen, für die dann eine reduzierte Menge über AdaBoost konstruiert wird. Da diese Teilbereiche einen wesentlich Punkt für die gesamte Verarbeitungszeit darstellen, kann diese Methode leicht skaliert werden.

Die teuren Prozesse sind die Berechnungen der Klassifikatoren auf den Trainingsbildern. Liegen die Ergebnisse erst einmal im Speicher, so kann diese Methode leicht durch mehr Speicher beschleunigt werden.

Es ist aber auch mit wenig Aufwand möglich die einzelnen Teilbereiche auf einen anderen Rechner zu übertragen und dort ausführen zu lassen. Da es nicht notwendig oder wichtig ist, dass die Teilbereiche die gleich Größe haben, ist es auch denkbar unterschiedlichste Rechner mit verschiedener Leistung einzubeziehen. Natürlich kommt ein Mehraufwand für die Verwaltung und das Zusammenfügen der Ergebnismengen hinzu.

Jeden Teilbereich mit verschiedenen Bildern trainieren

Wir vermuten, dass es dem Ziel einer besonderen Durchmischung mit möglichst verschiedenartigen Merkmalen förderlich wäre, wenn wir in jedem Preboost-Schritt eine neue Trainingsmenge für die negativen Ausschnitte bestimmen. Nachdem ein Teilbereich der Klassifikatoren bestimmt wurde, der reduziert werden soll, werden als erstes die Ergebnisse der Klassifikatoren auf den Trainingsbildern berechnet. Würden wir bei jedem dieser Schritte eine andere Menge zum trainieren wählen, als bei dem Vorgänger, würde sich an der Performance kaum etwas ändern. Es müssten lediglich

die Menge der Trainingsbilder neu geladen werden, die man sonst im Speicher halten könnte.

Wir würden uns davon erhoffen, dass sich das Verhältnis und die Verteilung der Klassifikatoren über den Merkmalsraum der Problemstellung noch besser anpassen würde.

Verschiedene Untermengen

Im Rahmen der Merkmalsauswahl wird häufig mit unterschiedlichen Teilmengen an Merkmalen gearbeitet. Schon bei einem frühen Ensemble Lernalgorithmus, dem Bagging, wird so vorgegangen.

Der Gedanke hier ist, die Teilmengen, die während des Reduktionsprozesses auftreten, nicht zu einer Menge zu vereinigen, sondern zu K Teilmengen. Diese Teilmengen könnten dann genauso reduziert werden, wie es bisher geschieht. Und auch die Vereinigung dieser Teilmengen könnte wie gehabt reduziert werden.

Wenn wir jedoch solche Teilmengen zur Verfügung haben, so könnten wir während des Kaskadenlernens, jedem Ensemble für einen Knoten, eine andere Merkmalsmenge zur Verfügung stellen. Da wir in unserem Prozess alle Klassifikatoren berechnet haben, entsteht hier kein Mehraufwand.

8.0.2 Fazit

Im Vergleich mit anderen Autoren berechnen wir den Quotienten der Haar-Merkmals-Flächen statt der Differenz.

Wir nutzen AdaBoost, in einer leicht veränderten Variante, zur Reduktion einer Merkmalsmenge. Um AdaBoost zu beschleunigen und einen zeitlichen Vorteil zu erreichen, verzichten wir auf ein neu trainieren der Klassifikatoren für das Ensemble. Außerdem arbeiten wir mit geringen Mengen an positiven und negativen Beispielbildern. Mit diesen beiden Änderungen können wir AdaBoost nutzen und in kurzer Zeit mehrere Ensembles trainieren, die die Grundlage für unsere reduzierte Menge darstellen.

Wichtig für das Lernen eines Ensembles ist die Verschiedenartigkeit der Klassifikatoren, die diesem Ensemble zugefügt werden sollen. Unsere Annahme war, dass wir diese Verschiedenartigkeit auch dann noch in dem von AdaBoost trainiertem Ensemble vorfinden, wenn wir den Algorithmus wie beschrieben einschränken. So ist ein wesentlicher Punkt, den wir feststellen, dass unsere, durch Preboost reduzierten, Klassifikatoren-Mengen eine bessere Durchmischung an verschiedenen und guten Klassifikatoren haben, als zufällig reduzierte Merkmalsmengen. Und das auch, wenn die zufällig reduzierten Mengen deutlich mehr Klassifikatoren beinhalteten.

Die Vergleiche, die wir in dem Kapitel 7 vorstellen, bestätigen unsere Annahmen. Wir konnten in kurzer Zeit viele verschiedene Kaskaden trainieren, die auf unseren Testbildern sehr gute Ergebnisse erzielen. Besonders hervorheben möchten wir die Parkplatzbilder, die viel und schwierigen Hintergrund aufweisen.

AdaBoost in der von uns leicht geänderten Variante, eignet sich dazu Merkmalsmengen stark zu reduzieren und die Verschiedenartigkeit zu erhalten.

Wir finden mit unserer Variante des Preboostings nicht das (schwer zu fassende) Optimum. Ein, mit einer erschöpfenden Suche und AdaBoost durchgeführtes Training, wird mit großer Wahrscheinlichkeit die besseren Kaskaden erbringen. Der zeitliche Aufwand hierfür läßt jedoch nur wenig experimentieren mit anderen Konfigurationen zu.

Die Kaskaden, die auf den mit Preboost reduzierten Mengen trainiert wurden, bringen gute Ergebnisse in kurzer Trainingszeit. Damit wird es möglich Experimente durchzuführen, die sonst zeitlich nicht möglich wären, wie verschiedene Klassifikatoren in einer Kaskade zu verbinden und noch größere Merkmalsmengen zu nutzen.

Appendix

A

A.1 Beispiele der verwendeten Testbildmengen

Abbildung A.1: Beispielbilder der Testmenge “Standard Testbilder 37”, die 37 Bilder mit 38 Kennzeichen enthält in einer Auflösung von 380x280

Abbildung A.2: Beispielbilder der Testmenge “Parkplatz Testbilder 43”, die 43 Bilder mit 129 Kennzeichen enthält in einer Auflösung von 1024x1280

Abbildung A.3: Beispielbilder der Testmenge “Griechische Testbilder 67”, die 67 Bilder mit 71 Kennzeichen enthält int einer Auflösung von 480x360. Es sind keine griechischen Nummernschilder in die Trainingsbilder eingeflossen.

A.2 Ergebnisbilder einiger Kaskaden

Abbildung A.4: Ergebnisbilder der Kaskade “HF3081 Preboost ME03 K1971 VJ“, 37 Bilder, Detektionsrate 0.81, Falsch-Positive 201.

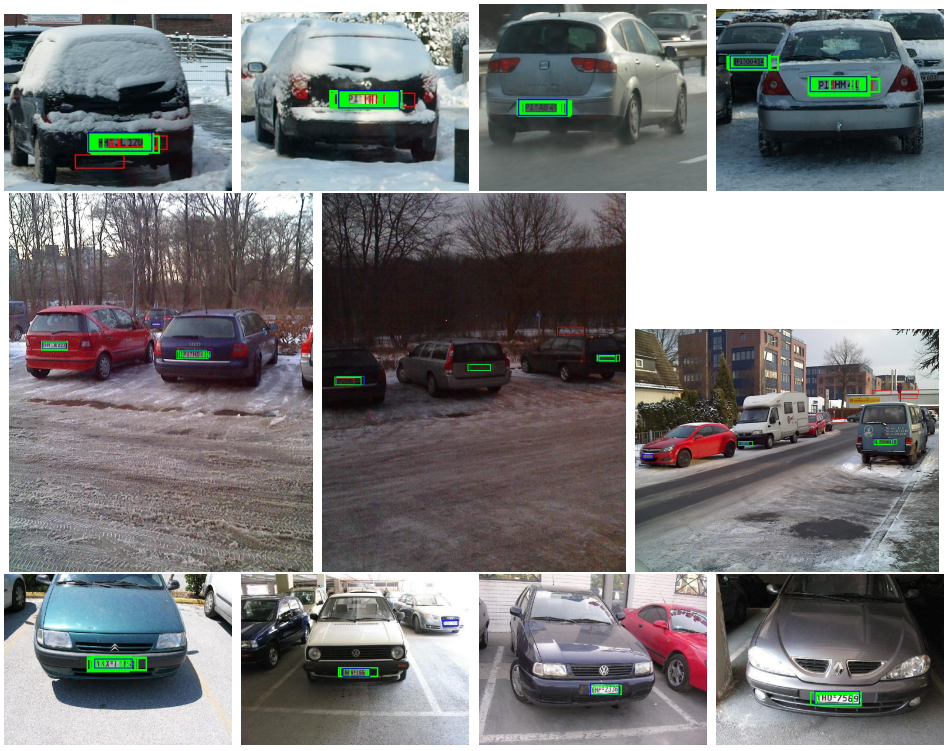


Abbildung A.5: Ergebnisbilder der Kaskade ”HF3081 Preboost K500 FB20-3” auf der Standard Testmenge 37 (oben), Parkplatz 43 (mitte), Griechische 67 (unten)

Abbildung A.6: Ergebnisbilder der Kaskade "HF3081 Preboost Multi K350 Joint VJ" auf der Parkplatz 43 (oben, mitte), Griechische 67 (unten) trainiert mit den Basis Haar-Merkmalen und den zusätzlichen "Ecken und Einfassungen" (Grafik **??**.

Literaturverzeichnis

[1] L. Petersson B. Rasolzadeh and N. Pettersson. *Response binning: Improved weak classifiers for boosting.* IEEE Intelligent Vehicle Symposium, 2006.

[2] Chang Huang Bo Wu, Haizhou Ai. *Fast Rotation Invariant Multi-View Face Detecting based on Real AdaBoost.* International Conference on Automatic Face and Gesture Recognition, 2004.

[3] A. Broumandnia and M. Fathy. *Application of pattern recognition for Farsi license plate recognition.* Proc. Int. Conf. GVIP, Cairo, Egypt, 2005.

[4] M. Oren C. Papageorgiou and T. Poggio. *A general framework for object detection.* In International Conference on Computer Vision, 1998.

[5] J. Cano and J. C. Perez-Cortes. *Vehicle License Plate Segmentation in Natural Images.* vol. 2652, F. J. Perales et al., Eds. New York: Springer-Verlag, 2003, pp. 142–149, 2003.

[6] IEEE Ioannis E. Anagnostopoulos Member IEEE Ioannis D. Psoroulas Vassili Loumos Member IEEE Christos-Nikolaos E. Anagnostopoulos, Member and IEEE Eleftherios Kayafas, Member. *License Plate Recognition From Still Images and Video Sequences: A Survey.* IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 9, NO. 3,, 2008.

[7] F. Crow. *Summed-area tables for texture mapping.* In Proceedings of SIGGRAPH, volume 18(3), pages 207–212, 1984.

[8] Yoav Freund. *Boosting a weak learning algorithm by majority.* Information and Computation, 121(2):256–285, 1995.

[9] Yoav Freund and Robert E. Schapire. *A decision-theoretic generalization of on-line learning and an application to boosting.* Journal of Computer and System Sciences, 55(1):119–139, August 1997, 1997.

[10] Whitley D Guerra-Salcedo, C. *Feature selection mechanisms for ensemble creation: a genetic search perspective.* AAAI-99, 1999.

[11] S.-Y. Chen H.-J. Lee and S.-Z. Wang. *Extraction and recognition of license plates of motorcycles and vehicles on highways.* in Proc. ICPR, 2004, pp. 356–359, 2004.

[12] Tin Kam Ho. *The random subspace method for constructing decision forests.* IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 20, NO. 8, 1998.

[13] B. Hongliang and L. Changping. *A hybrid license plate extraction method based on edge statistics and morphology.* in Proc. ICPR, 2004, pp. 831–834, 2004.

[14] Xiangjian He Huaifeng Zhang, Wenjing Jia and Qiang Wu. *Learning-Based License Plate Detection Using Global and Local Features.* Computer Vision Research Group, University of Technology, Sydney, 2006.

[15] K.; Fukumiya E.; Iwasa K.; Ogura Y. Ishii, Y.; Imagawa. *Profile face detection using block difference feature for automatic image annotation.* Consumer Electronics, 2006. ICCE '06. 2006 Digest of Technical Papers, 2006.

[16] A. Rafael J. Barroso, E. Dagless and J. Bulas-Cruz. *Number plate reading using computer vision.* in Proc. IEEE Int. Symp. Ind. Electron., pp. 761–766.

[17] Y. Lu J. Kong, X. Liu and X. Zhou. *A novel license plate localization method based on textural feature analysis.* in Proc. IEEE Int. Symp. Signal Process. Inf. Technol., Athens, Greece, 2005, pp. 275–279, 2005.

[18] M. Urban J. Matas, O. Chum and T. Pajdla. *Robust wide baseline stereo from maximally stable extremal regions.* In BMVC02, volume 1, pages 384–393, London, UK, 2002.

[19] M. Mullin J. Wu, J. Rehg. *Learning a Rare Event Detection Cascade by Direct Feature Selection.* Advances in Neural Information Processing Systems, Nr. 16, 2004.

[20] Ludmina I. Kuncheva. *Combining Pattern Classifiers.* Wiley Interscience - John Wiley & Sons, Inc.

[21] M. Kölsch. *Vision Based Hand Gesture Interfaces for Wearable Computing and Virtual Environments.* Dissertation, University of California, Santa Barbara, 2004.

[22] Serge Belongie Louka Dlagnekov. *Recognizing Cars.* Technical Report CS2005-0833, UCSD University of California, San Diego, 2005.

[23] J. Matas and K. Zimmermann. *Unconstrained licence plate detection.* Proc. of the 8th International IEEE Conference on Intelligent Transportation Systems, pages 572–577, Heidelberg, Germany, 2005.

[24] Horst Bischof Michael Donoser, Clemens Arth. *Detecting, Tracking and Recognizing License Plates.* Computer Vision–ACCV 2007, Springer, 2007.

[25] N. Oza and K. Tumer. *Input decimation ensembles: Decorrelation through dimensionality reduction.* MCS '01 Proceedings of the Second International Workshop on Multiple Classifier Systems, 2001.

[26] M. N. Granieri P. Comelli, P. Ferragina and F. Stabile. *Optical recognition of motor vehicle license plates.* IEEE Trans. Veh. Technol., vol. 44, no. 4, pp. 790–799, 1995.

[27] Michael Jones Paul Viola. *Robust Real-time Object Detection.* SECOND INTERNATIONAL WORKSHOP ON STATISTICAL AND COMPUTATIONAL THEORIES OF VISION – MODELING, LEARNING, COMPUTING, AND SAMPLING, 2001.

[28] Michael Jones Paul Viola. *Fast and robust classification using asymmetric adaboost and a detector cascade.* Advances in Neural Information Processing Systems, 2002.

[29] P. Bartlett andW. S. Lee R. E. Schapire, Y. Freund. *Boosting the margin: a new explanation for the effectiveness of voting methods.* Proceedings of the Fourteenth International Conference on Machine Learning, 1997.

[30] V. Pisarevsky R. Lienhart, A. Kuranov. *Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection.* MRL Technical Report, 2002.

[31] & Lebart K Redpath, D.B. *Boosting feature selection.* In Proceedings of the international conference on advances in pattern recognition (pp. 305,314), 2005.

[32] Onoda T. Müller K.R. Rätsch, G. *Soft margins for adaboost.* achine Learning 42 287–320, 2001.

[33] Robert E. Schapire. *The strength of weak learnability.* Machine Learning, 5(2):197–227, 1990.

[34] Robert E. Schapire. *The Boosting Approach to Machine Learning: An Overview.* MSRI Workshop on Nonlinear Estimation and Classification, 2002.

[35] K. Sung and T. Poggio. *Example-based learning for view-based human face detection.* IEEE Trans. on Pattern Analysis and Machine Intelligence, 20(1):39–51, 1998.

[36] K.-T. Li T.-H. Wang, F.-C. Ni and Y.-P. Chen. *Robust license plate recognition based on dynamic projection warping.* Proc. IEEE Int. Conf. Netw., Sens. Control, 2004, pp. 784–788, 2004.

[37] Osamu Hori Takeshi Mita, Toshimitsu Kaneko. *Joint Haar-like Features for Face Detection.* 2005.

[38] M. Viola, Michael J. Jones, and Paul Viola. Fast multi-view face detection. In *Proc. of Computer Vision and Pattern Recognition*, 2003.

[39] P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* Proc. CVPR, pages 511–518, 2001.

[40] Yong Haur Tay Wing Teng Ho, Wooi Hen Yap. *Learning-based License Plate Detection on Edge Features.* Computer Vision and Intelligent Systems (CVIS) Group Universiti Tunku Abdul Rahman, Malaysia, 2007.

[41] Alan L. Yuille Xiangrong Chen. *Detecting and Reading Text in Natural Scenes.* CVPR. Volume: 2, pp. 366–373, 2004.

[42] D. Geman Y. Amit and K. Wilder. *Joint induction of shape features and tree classifiers*. 1997.

[43] S. Z. Li H. Zhang Z. Zhang, L. Zhu. *Real-Time Multi-View Face Detection*. International Conference on Automatic Face and Gesture Recognition, 2002.

[44] R. Zunino and S. Rovetta. *Vector quantization for license-plate location and image coding*. IEEE Trans. Ind. Electron., vol. 47, no. 1, pp. 159–167, 2000.

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereiches einverstanden.

Hamburg, den

Unterschrift: