Ressourcenoptimierte Objektdetektion und teilüberwachtes Lernen zur Echtzeitanwendung mit konfidenzbasierten, kaskadierten Klassifikationssystemen

Dissertation

zur Erlangung des Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität Dortmund

vorgelegt von

Dipl.-Inform. Armin Staudenmaier

Dissertation in der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität Dortmund

Tag der mündlichen Prüfung: 3. Dezember 2014
Dissertationsort: Dortmund

Hauptgutachter:
Prof. Dr. rer. nat. Christian Wöhler

Zweitgutachter:
apl. Prof. Dr. rer. nat. Frank Hoffmann

Zusammenfassung

In dieser Arbeit werden ressourcenoptimierte Trainings- und Detektionsverfahren zur bildbasierten Echtzeitobjektdetektion vorgestellt. Angewendet werden diese auf anspruchsvolle amerikanische Geschwindigkeitsbegrenzungen. Für die maschinellen Lernverfahren wird ein umfangreicher Datensatz bestehend aus 11000 Trainingsbeispielen und einem separaten Testdatensatz bestehend aus 6000 Testbildern verwendet.

Um echtzeitfähige Detektoren zu generieren werden kaskadierte Klassifikationssysteme verwendet, für die ein im Rahmen dieser Arbeit entwickeltes Bayes'sches Wahrscheinlichkeitsmodell aufgestellt wird. Mit diesem Modell werden zum einen die Eigenschaften vorhergehender Stufenklassifikatoren in die Entscheidung der darauf folgenden Klassifikatoren miteinbezogen, indem die Entwicklung der Wahrscheinlichkeiten über die Stufen berechnet wird. Zum anderen wird mit diesem Wahrscheinlichkeitsmodell ein ebenfalls in dieser Arbeit entwickeltes Verfahren zur Optimierung der Laufzeiten und damit des Energieverbrauchs kaskadierter Klassifikationssysteme vorgestellt, indem ein Kostenmodell aufgestellt wird, bei dem die wahrscheinlich auftretenden Fehloperationen zwischen unterschiedlich komplexen Klassifikatoren betrachtet werden. Es kann gezeigt werden, dass die Laufzeiten mit diesem Verfahren um den Faktor 20 reduziert werden können im Vergleich zur reinen klassifikationsfehlerbasierten Betrachtung. Darüber hinaus werden ebenfalls neue auflösungsabhängige Trainings- und Detektionsverfahren vorgestellt, die im Gegensatz zu den Laplacepyramiden die Filtereigenschaften der Merkmale ausnutzen, womit sich eine in das Kaskadentraining und die Detektion integrierte Einschränkung des Suchraumes mit einer wahrscheinlichkeits- und auflösungsabhängigen Feinsuche ergibt. Es wird erst ab einer bestimmten Wahrscheinlichkeit der Zugehörigkeit zur Objektklasse eine Feinsuche, die mit erhöhtem Energieaufwand verbunden ist, durchgeführt. Mit diesem Verfahren werden sowohl die Trainings- als auch die Detektionszeiten drastisch reduziert. Bei der Ausführung der Klassifikatoren kann die Laufzeit um den Faktor 10 bei identischen Klassifikationsraten im Vergleich zu den gängigen Verfahren reduziert werden.

Die Optimierung des damit verbundenen Energieverbrauchs ist insbesondere bei mobilen Geräten mit beschränkten Rechenkapazitäten wie z.B. bei Fahrerassistenzsystemen oder bei Mobiltelefonen von großem Vorteil, da durch die hohen Stückzahlen eine beachtliche Effizienzsteigerung für den Gesamtenergieverbrauch erzielt werden kann.

Mit dem Wahrscheinlichkeitsmodell werden außerdem neue Verfahren zur Konfidenzwertberechnung vorgestellt, die eine Fusion der Konfidenzen der unterschiedlichen Stufenklassifikatoren ermöglichen. Mit den Konfidenzwerten können Intraklassenmuster differenziert beurteilt werden und es wird ein neues Verfahren vorgestellt, mit dem multiple Detektionen anhand ihrer Konfidenzwerte zu einer gewichteten Gesamtdetektion fusioniert werden. Außerdem wird ein konfidenzbasierter Selektionsmechanismus für die Auswahl der Muster der Stufenklassifikatoren vorgestellt, mit dem auf "schwierigeren" Beispielen trainiert wird und somit der Fehler bei der Ausführung auf "einfacheren" Daten erheblich reduziert werden kann.

Mit den Konfidenzwerten werden außerdem integrierte teilüberwachte Kaskadendetektionsverfahren ermöglicht, die im Rahmen dieser Arbeit entwickelt wurden. Dabei wird dem Klassifikator die Möglichkeit eingeräumt, Negativbeispiele aus der Hintergrundklasse als "migrierte" Objekte umzudefinieren und als Trainingsbeispiele in der Objektklasse zu verwenden. Aus der überproportional großen Hintergrundklasse werden diese Beispiele mit dem Verfahren identifiziert, wodurch eine gleichzeitige Datenexploration erfolgt. Es wird gezeigt, dass die Klassifikationsraten mit dem teilüberwachten Verfahren insbesondere in Bereichen geringer Falsch-Positiv-Detektionen den vollständig überwachten Klassifikatoren bereits mit der Hälfte der verfügbaren Objektdaten überlegen sind. Die mit den teilüberwachten Verfahren erzeugten Detektoren liegen ebenfalls im Echtzeitbereich. Durch die geringeren benötigten Mengen an Objektdaten können Einsparungen bei der aufwändigen Annotation durch menschliche Experten der Daten erzielt werden.

Als Stufenklassifikatoren werden Ensembleverfahren eingesetzt, wobei das bekannte Gradientenboosting-Verfahren mit den hier entwickelten Konfidenzwerten zur Gewichtung und Modellierung der Entscheidungsfunktionen sowie mit einer multidimensionalen Optimierung erweitert wurde. Mit diesem Verfahren konnten bei hohen Detektionsraten von 98 % die energieeffizientesten Klassifikatoren erzeugt werden. Außerdem wird ein neues, im Rahmen dieser Arbeit entwickeltes Verfahren zur Klassifikation mit Unterteilungsbäumen und multiplen konfidenzbasierten Entscheidungspfaden vorgestellt. Bei diesem Verfahren kann bei der Ausführung des Klassifikators eine adaptive Regulierung zwischen geringer mittlerer Laufzeit bei höherem Klassifikationsfehler und einer höheren Laufzeit bei niedrigerem Klassifikationsfehler erfolgen. Damit ist das Verfahren insbesondere bei mobilen rechnergestützten Multifunktionssystemen, bei denen Schwankungen in den Rechenkapazitäten auftreten, von großem Vorteil.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Algorithmenverzeichnis

Kapitel 1

Einleitung

Die menschliche visuelle Wahrnehmung, bei der visuelle Reize mit dem Auge wahrgenommen werden und im Gehirn zur Erkennung von Objekten und Elementen weiterverarbeitet werden, ist erstaunlich. Die Erkennung von Objekten unterschiedlichster Art geschieht für den Menschen scheinbar mühelos. Beim Vergleich der Rechenleistung eines Computers mit dem Menschen, liegt das menschliche Gehirn bei etwa 10^{13} analogen Rechenoperationen pro Sekunde, bei einer Leistung von etwa 15 bis 20 Watt. Im Gegensatz dazu erreicht zum Beispiel der Supercomputer BlueGene/L von IBM bis zu $3,6 \times 10^{14}$ Gleitkommaoperationen pro Sekunde bei einer Leistung von $1,2 \times 10^{6}$ Watt [62]. Eine interessante Frage ist, wie weit sich diese Fähigkeit anhand von digitalen Bildern und Computern, also die maschinelle Objekterkennung, bei möglichst geringem Energieverbrauch simulieren lässt, da der Einsatzbereich maschineller Detektions- und Klassifikationsaufgaben mit fortschreitender Technologisierung und Miniaturisierung immer größer wird und auf immer kleiner werdenden elektronischen Geräten anwendbar ist. Die Objekterkennung in der Bildverarbeitung bezieht sich oft auf das Erkennen von Objektklassen, wie zum Beispiel “Fahrzeug”, “Fußgänger” oder “Hindernis”. Die Anwendungsbereiche reichen von der Gesichts- und Lächelerkennung sowie Erkennung der Augen und Blickrichtung bei Digitalkameras und Mobiltelefonen über die Erkennung von Produktionsfehlern bei Bauteilen in der industriellen Bildverarbeitung. Im Automobilbereich existieren zahlreiche Assistenzsysteme, bei denen die bildbasierte Detektion von Fahrspur, Autorückfronten oder Verkehrszeichen die Kernfunktionalität bilden [81]. Eine große Anzahl von Systemen zur Objekterkennung basieren auf einem zweistufigen Verfahren. Im ersten Schritt wird eine schnelle Suche nach Objektkandidaten auf dem ganzen Bild durchgeführt. In einem zweiten Schritt werden dann die identifizierten Kandidaten mit einem komplexen, berechnungsintensiven Verfahren, meistens einem Klassifikator, detektiert, bzw. klassifiziert. Die schnelle Kandidatensuche im ersten Schritt wird dabei meist anhand einer modellbasierten Suche mit der Hough-Transformation [63] durchgeführt, wobei nach geometrischen Formen wie Geraden, Kreisen oder Ellipsen gesucht wird. Ein Nachteil des Verfahrens sind Objekte, die keine prägnanten geometrischen Formen enthalten oder deren Erscheinungsformen stark variieren. Ein weiterer Nachteil ist, dass die Parameter nicht mittels einer definierten Statistik an Objekt- und Hintergrunddaten ermittelt werden können, und damit keine Aussagen über die Signifikanz der Erkennungsleistung im ersten Schritt getroffen werden kann. In dieser Arbeit soll vor allem erforscht werden, ob und mit welchen Mechanismen eine laufzeitef-

fiziente Objektdetektion durch ein maschinelles Lernverfahren möglich ist. Dabei liegt der Schwerpunkt auf Optimierungs- und Klassifikationsverfahren sowie Ensemblemethoden, bei denen während des Trainingsvorgangs ein komplexer Ensembleklassifikator aus einzelnen Basisklassifikatoren zusammensetzt wird. Das Ziel ist, Klassifikatoren im Echtzeitbereich zu erzeugen, die trotz hoher Detektionsraten geringe Laufzeiten aufweisen, woraus eine große Energieersparnis resultiert, da die Klassifikatoren im industriellen Umfeld in außerordentlich hohen Stückzahlen eingesetzt werden. Bereits geringe Verbesserungen der Energieeffizienz bei den Algorithmen können durch die hohen Stückzahlen eine große Gesamtenergieersparnis bewirken.

Allerdings soll die Optimierung nicht durch eine Parallelisierung erfolgen, sondern mit Einkernprozessoren durchführbar sein. Deshalb werden, um eine energieeffiziente Anwendung zu garantieren, alle unterschiedlichen Klassifikatoren in einer hierarchischen Kaskadenstruktur eingebettet. Ein weiterer Teil der Arbeit beschäftigt sich mit dem teilüberwachten Lernen und der Datenexploration der Detektionssysteme, wobei die Menge der Hintergrundklasse nicht gekennzeichnete Objekte der Objektklasse enthält. Der Klassifikator besitzt ein Konfidenzmaß und wird mit der Fähigkeit ausgestattet, Beispiele aus der Hintergrundklasse "umzudefinieren" und in die Objektklasse aufzunehmen. Dieser Prozess ist in den Trainingsmechanismus des Kaskadendetektors integriert.

Die Hauptanwendung ist dabei das Erkennen von anspruchsvollen amerikanischen Verkehrszeichen, prinzipiell ist das Verfahren aber auch auf andere Aufgabenstellungen anwendbar. Der aktuelle Stand der Technik in der Mustererkennung und beim maschinellen Lernen sind kaskadierte Klassifikatoren [116, 115], wobei ein Klassifikator eine mathematische Funktion ist, die die Einteilung eines Bildausschnitts in Objekt oder Hintergrund vornimmt. Das beschriebene Zweischrittverfahren wird dabei in ein N-Schrittverfahren mit N Stufenklassifikatoren aufgeteilt, wobei die einzelnen Klassifikatoren die Kandidatenmenge immer stärker aussortieren und eine steigende Berechnungskomplexität besitzen. Bei der bildbasierten Objekterkennung ist die ungleichmäßige Verteilung der Objekt- und Hintergrunddaten ein grundlegendes Problem. Die Anzahl der unterschiedlichen Skalierungen der Bildbereiche, in denen Objekte vorkommen können, beträgt bei einer feinen Rasterung typischerweise bis zu 500000 Möglichkeiten. Bei einem realistischen Datensatz von 10000 Bildern, die Objekte enthalten, ist die Größe der Hintergrundklasse also 5×10^9.

Eine grundlegende Idee und die entsprechende algorithmische Umsetzung

dieser Arbeit ist es, durch Einschränkungen der Frequenzeigenschaften der Merkmale der Klassifikatoren, diese Menge zu minimieren, um hierarchisch von einer groben Suche in einer niedrigen Auflösung in höhere Auflösungen mit feineren Details zu wechseln, falls der zu betrachtende Bildausschnitt mit einer bestimmten Wahrscheinlichkeit Objektcharakter aufweist. Ähnlich wie beim Menschen, dessen Aufmerksamkeit sich auf die wesentlichen Details bei der Objekterkennung fixiert und irrelevante Details ausblendet, werden die Merkmale von Stufe zu Stufe aufwändiger, da die Wahrscheinlichkeit der Zugehörigkeit zur Objektklasse steigt. Dazu wird ein statistisches Optimierungsverfahren vorgestellt, dass auf den zu erwartenden Fehloperationen beruht, und somit in der Lage ist, aus einem großen Pool unterschiedlicher Klassifikatoren den effizientesten auszuwählen, wobei die Klassifikatoren unterschiedlich komplexe Merkmale beinhalten können. Die Skalierung auf definierte Signifikanzniveaus garantiert eine hohe Klassifikationsleistung. Das Optimierungsverfahren wird in Kapitel 4 beschrieben.

Im Gegensatz zu den häufig verwendeten eindimensionalen Haarähnlichen Merkmalen bestehen die Merkmale aus einer variablen Anzahl an Rechtecken und Merkmalstypen, sodass einzelne Klassifikatoren auch hochdimensionale Merkmalsvektoren extrahieren können, falls dies energetisch von Nutzen ist. Die hierarchische, auflösungsabhängige Einteilung hat sowohl beim Training der Klassifikatoren den Vorteil, dass weniger Kombinationen betrachtet werden müssen und damit das Training schneller durchführbar ist, als auch einen beachtlichen Laufzeitgewinn bei der Ausführung der Klassifikatoren, der eine Echtzeitanwendung ermöglicht. Die Merkmale und die merkmalsbasierten Klassifikatoren sind in Kapitel 3 beschrieben. Außerdem wird ein Kommunikationsmechanismus vorgestellt, bei dem Eigenschaften des Vorgängerklassifikators an den Nachfolgenden weitergeben werden und in dessen Entscheidung mit einfließen. Dieser Mechanismus basiert auf der Bayes'schen Statistik und ermöglicht es, eine hohe Anzahl an einfacheren Klassifikatoren zu verwenden, ohne dass die Erkennungsleistung merklich absinkt. Das Prinzip und die grundlegende Funktionsweise des Verfahrens ist in Kapitel 2.3 geschildert.

In Kapitel 5 werden Verfahren zur Konfidenzberechnung der Kaskadenklassifikatoren vorgestellt und ausgewertet. Eine konfidenzbasierte Musterauswahl innerhalb des Trainings ermöglicht eine deutliche Steigerung der Klassifikationsrate, indem sich der Klassifikator auf Beispiele mit niedriger Konfidenz konzentriert. Außerdem wird anhand der Konfidenzwerte ein semiüberwachtes Verfahren zum Kaskadentraining vorgestellt. Die Hintergrund-

klasse enthält nicht zugewiesene Beispiele der Objektklasse. Während des Trainings ist der Kaskadenklassifikator in der Lage, bei hoher Konfidenz der Objektklasse Beispiele aus der Hintergrundklasse in die Objektklasse zu "migrieren", indem sie umdefiniert und verschoben werden.

Kapitel 6 beschreibt Ensembleverfahren, wobei ein leistungsstarker Klassifikator trainiert wird, der aus unterschiedlichen Basisklassifikatoren zusammengesetzt wird. Um die Diversität der Einzelklassifikatoren zu erzeugen werden sowohl Mechanismen zur Neugewichtung der Trainingsbeispiele, als auch Mechanismen zur Aufteilung, verwendet. Da das Ziel ist, möglichst laufzeiteffiziente Klassifikatoren zu erzeugen passen sich alle Verfahren automatisch an die (steigende) Komplexität der Trainingsaufgaben in den Stufen an.

US-Verkehrszeichen

Die voranschreitende Entwicklung von Mikroprozessoren bzw. Mikrocontrollern und ihre Miniaturisierung ermöglichen ein immer größer werdendes Einsatzspektrum.Vor allem in der Automobilindustrie kommen immer mehr Fahrerassistenzsysteme (englisch: Advanced Driver Assistance Systems (ADAS)), das sind elektronische Zusatzeinrichtungen in Kraftfahrzeugen zur Unterstützung des Fahrers in bestimmten Fahrsituationen [119, 121], zum Einsatz. In Tabelle 1.0.1 sind einige der verfügbaren Assistenzsysteme gelis-

Fahrerassistenzfunktionen
Ampelassistent
Bremsassistent
Fernlichtassistent
Nachtsicht-Assistent
Spurhalte-/Spurwechselassistent
Fahrerzustandserkennung
Verkehrszeichenerkennung

Tabelle 1.0.1: Aktuell verfügbare Fahrerassistenzfunktionen.

tet. Der nächste Schritt zwischen assistiertem Fahren und autonomen Fahren, bei dem der Fahrer nicht einschreitet, ist das hochautomatisierte oder pilotierte Fahren. Das Fahrzeug fährt anhand der Sensorik und Steuerungs- sowie Auswertungsalgorithmen im Hintergrund mit und kann die Fahraufgabe, vor allem in Gefahrensituationen, vollständig übernehmen. Jedoch sind

die derzeitigen Fahrerassistenzsysteme so konzipiert, dass der Fahrer jederzeit die vollständige Kontrolle übernehmen kann, da dies gesetzlich vorgeschrieben ist. Ein weiterer Grund, dass autonome Systeme vermieden werden, ist die nicht ausreichende Zuverlässigkeit, vor allem bei der Erkennung und Klassifikation von Objekten und der Interpretation der Szenerie im Umfeld des Fahrzeugs [81]. “Derzeitig verfügbare Sensoren und bekannte Signalverarbeitungsansätze können noch keine zuverlässige Umfelderkennung unter allen möglichen Fahrzuständen und Wetterbedingungen bieten” [119]. Die Verkehrszeichenerkennung anspruchsvoller Schilder in Echtzeit steht im

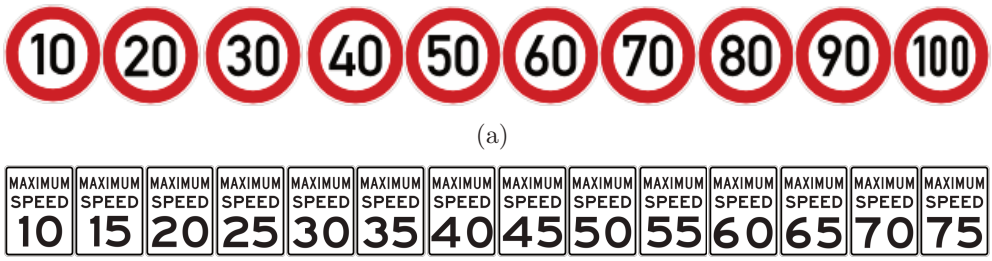
(a) (b)

Abbildung 1.0.1: Verkehrszeichenerkennungssystem für runde Schilder. (a): Im Fahrzeug integrierte Monokamera. (b): Aufnahme der Szene und maschinelle Erkennung, sowie Darstellung der aktuellen Geschwindigkeitsbegrenzung auf dem Armaturenbrett.

Mittelpunkt dieser Arbeit. Da sich immer mehr Assistenzsysteme die Rechenleistung des On-Board-Systems teilen, sind laufzeiteffiziente und damit energiesparende Lösungen unumgänglich.

Das Erkennungssystem für europäische Geschwindigkeitsbegrenzungen ist in Abbildung 1.0.1 dargestellt. Die Monokamera (Abb. 1.0.1(a)) zeichnet die aktuelle Szene auf. Eine Bildverarbeitungssoftware sucht das Bild nach Geschwindigkeitsbegrenzugen ab, klassifiziert diese bei erfolgreicher Detektion und teilt die Informationen dem Bordcomputer mit, der dann das Ergebnis darstellt. Der Fahrer bekommt somit immer die aktuelle Geschwindigkeitsbegrenzung angezeigt. Bei Überschreitung der Geschwindigkeit kann ein Warnsignal oder ein haptisches Signal am Gaspedal den Fahrer aufmerksam

machen. Die Vorteile gegenüber einer rein GPS-basierten Lösung oder einer Lösung mittels integrierten Sendern und Empfängern, wie z.B. RFID-Transpondern, ist, dass das das Verfahren nur von der Kamera abhängig und nicht auf aktuelle Datenbanken angewiesen ist. Damit können die Verkehrsschilder auch in Baustellenbereichen, in denen das Einhalten der vorgeschriebenen Geschwindigkeit besonders wichtig ist, erkannt werden. Durch das Einhalten der Geschwindigkeitsbegrenzungen werden Gefahrensituationen und schwere Unfälle durch überhöhte Geschwindigkeiten reduziert. Außerdem wird der Kraftstoffverbrauch gemindert, was vor allem in Kombination mit Verkehrsleitsystemen mit Wechselverkehrszeichen zu einem flüssigen Verkehr und zur Vermeidung von Staus führt. Abb. 1.0.1(b) zeigt dieses Funktionsprinzip. Das Grundlegende Problem bei der Detektion der Schilder ist



(a)

(b)

Abbildung 1.0.2: Gegenüberstellung von (a) deutschen und (b) amerikanischen Geschwindigkeitsbegrenzungen.

es, die große Menge an Hintergrundobjekten herauszufiltern und gleichzeitig alle relevanten Objekte, in diesem Fall die Geschwindigkeitsbegrenzugen, zu identifizieren. Da die Objekte in unterschiedlichen Skalierungen vorkommen können, ergibt sich eine große Menge an Positionen und Skalierungen. Der Vorteil bei den europäischen Schildern ist die ausgeprägte runde Form, die die Ziffern umschließt (siehe Abb. 1.0.2(a)) und durch die Wiener Konvention für Verkehrszeichen festgelegt ist [1]. Es existieren effiziente und robuste Algorithmen zur Identifikation der Kreise, wodurch ein hierarchischer Ansatz möglich ist.

Die Hauptanwendung in dieser Arbeit ist die Detektion amerikanischer Geschwindigkeitsbegrenzungen, die im Gegensatz zu den europäischen Schildern eine rechteckige Form aufweisen (siehe Abb. 1.0.2(b)). Allerdings ist der schwarze Rand kaum ausgeprägt und es existiert kein festes Höhen- zu

Breitenverhältniss. Bei der Kandidatensuche müssten dadurch zusätzlich alle unterschiedlich vorkommenden Verhältnisse überprüft werden. Die amerikanischen Verkehrsschilder sind zwar durch das “Manual on Uniform Traffic Control Devices” (MUTCD) spezifiziert [2], allerdings bleibt es den einzelnen Staaten offen, sich den Regelungen des MUTCD voll, teilweise oder mit bestimmten Zusätzen anzuschließen [85]. Die Verkehrszeichen der meisten anderen Länder, wie z.B. Japan und China, folgen größtenteils der Wiener Konvention. Eine weitere Ausnahme bilden die zentral- und südamerikanischen Länder, bei denen überhaupt kein Standard existiert, die jedoch meist nach nordamerikanischem Vorbild gestaltet sind. Abbildung 1.0.3 zeigt, wie

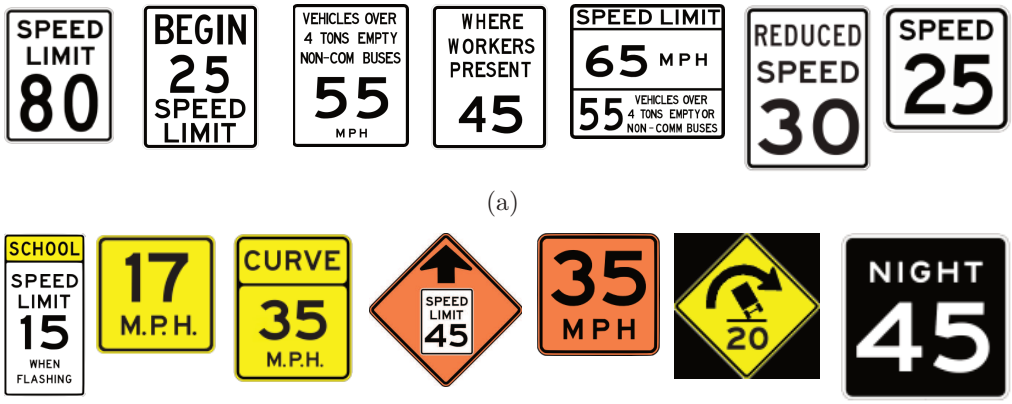


Abbildung 1.0.3: In die US-Speed-Limits integrierte Zusatzschilder. (a): Integrierte Einschränkungen, unterschiedliche Beschriftungen und Seitenverhältnisse. (b): Empfehlungen, Baustellenschilder und nächtliche Einschränkungen.

Zusatzschilder in die Begrenzungen integriert sind. In Abb. 1.0.3(a) sind die unterschiedlichen Seitenverhältnisse sowie die unterschiedliche Beschriftung der Schilder zu erkennen. Im Gegensatz zu den europäischen Schildern sind die Zusatzschilder integriert. In Abb. 1.0.3(b) bedeuten die gelben Schilder lediglich Empfehlungen oder Hinweise. Orange gefärbte Schilder markieren Baustellenbereiche und das schwarze “NIGHT” Schild gilt nur bei Nacht. An diesen Beispielen ist zu erkennen, dass eine modellbasierte Kandidatensuche als Vorstufe nur schwierig zu realisieren ist. Die Rechteckstruktur ist durch den sehr dünnen schwarzen Rand und die unterschiedlichen Breite- zu

Höheverhältnisse, sowie schwierige Beleuchtungsverhältnisse an den Rändern durch die Reflexionsschicht, nur bedingt für eine Kandidatensuche geeignet. Die dominantesten Strukturen sind in den Ziffern enthalten, die auch die größte Fläche in der Beschriftung der Schilder einnehmen. Außerdem sind die Umrisse der Ziffern durch den Schwarz-Weiß-Kontrast gut sichtbar und enthalten außer Schlagschatten und Sonnenreflexionen keine variablen Hintergrundstrukturen.

Für eine modellbasierte Ziffernsuche müsste getrennt nach einer Null und nach einer Fünf als zweite Ziffer gesucht werden. Für die Suche der Null wäre eine fünfdimensionale Houghtransformation [63] in den Ellipsenparameterraum nötig, was mit einem hohen Rechenaufwand verbunden wäre. Bei der Fünf existiert kein einzelnes Modell einer geometrischen Form, deshalb müsste getrennt nach einem Kreis für die Rundung und den Geraden gesucht werden. Eine weitere Möglichkeit bietet die generalisierte Houghtransformation, die jedoch durch den immensen Rechenaufwand eine Echtzeitanwendung derzeit ausschließt [88]. Aus diesen Gründen wird in dieser Arbeit eine klassifikatorbasierte Detektion mit maschinellen Lernverfahren betrachtet, wobei das Ziel eine Echtzeitanwendung für die US-Verkehrszeichen ist und der Einfluss unterschiedlicher Merkmale und Optimierungsmethoden sowie Klassifikatorstrukturen untersucht werden. Die Eingabe für den maschinellen Lernalgorithmus sind die Bereiche, welche die Ziffernpaare umschließen, was in Abb. 1.0.4 dargestellt ist.

Abbildung 1.0.4: Unterschiedliche US-Geschwindigkeitsbegrenzungen mit annotierten Bereichen über den Ziffernpaaren als Trainingsmuster für ein maschinelles Lernverfahren.

Kapitel 2

Induktive, Bayes‘sche Statistik kaskadierter Klassifikationssysteme

Im ersten Teil dieses Kapitels wird der aktuelle Stand der Technik beschrieben. Im zweiten Teil werden wichtige Grundlagen, sowie die grundlegenden Ideen und Mechanismen erläutert, die bei dieser Arbeit Verwendung finden, indem probabilistische Eigenschaften von Klassifikatoren durch Bayes'sche Regeln in unterschiedlichen Klassifikationssystemen erläutert werden. Im Gegensatz zu den in der Literatur bestehenden Verfahren wird durch die Bayes'sche Formulierung eine getrenne Fehlerbetrachtung möglich, wobei durch ein Zustandsmodell die Falsch-Positiv-Fehler von den Falsch-Negativ-Fehlern getrennt betrachtet werden. Außerdem wird durch die Bayes'sche Formulierung eine Integration der kaskadierten Klassifikatoren ermöglicht, indem die durch die Stufenklassifikatoren induzierten Wahrscheinlichkeiten für die Entscheidungen der nachfolgenden Klassifikatoren genutzt werden. Abschließend wird das zu Grunde liegende Verfahren prinzipiell beschrieben. Die entsprechende algorithmische Umsetzung der Theorie in den Unterkapiteln des zweiten Teils wird in den darauffolgenden Kapiteln aufgezeigt und anhand experimenteller Versuche untersucht.

2.1 Stand der Technik

Die klassische Aufteilung eines Systems zur Objektdetektion lässt sich in zwei Schritte einteilen. Im ersten Schritt werden mit einem effizenten, meist modellbasierten Verfahren alle möglichen Regionen, in denen sich das Objekt befinden kann, auf eine möglichst geringe Kandidatenmenge reduziert. Die Kandidaten werden dann mit einem komplexeren Verfahren, wie zum Beispiel einem Klassifikator in eine oder mehrere Objektklassen und eine Hintergrundklasse aufgeteilt. Allerdings ist es für Objektklassen mit variablen Strukturen schwierig, die Kandidatenmenge effizient einzuschränken. Modellbasierte Verfahren wie die Houghtransformation finden lediglich einzelne geometrische Strukturen wie Linien, Kreise oder Ellipsen. Für komplexere Strukturen existiert die generalisierte Houghtransformation [7], die allerdings durch den hohen Rechenaufwand eine Echtzeitanwendung derzeit ausschließt [88, 85].

Für die Verkehrszeichendetektion existiert eine gute Einführung von Fu und Huang [44], die jedoch nicht sehr umfassend ist. Eine Beschreibung der unterschiedlichen Methoden zur Verkehrszeichendektektion wird in dem Artikel von Brkic [20] gegeben. Eine umfangreiche Übersicht und Gegenüberstellung der unterschiedlichen Verkehrszeichendatensätze sowie Methoden im Bezug auf Fahrerassistenzsysteme wird in der Einführung von Mogelmose

[85] gegeben. Dort werden über 40 Systeme tabellarisch verglichen, wobei die Segmentierungsmethoden, verwendete Merkmale und die Detektionsmethoden gelistet sind. Da die Auswertungen der Systeme allerdings alle auf unterschiedlichen Datensätzen berechnet wurden, lassen sich die Ergebnisse nur schwer vergleichen. Die Detektionsraten liegen jedoch meistens über 90 % bei relativ niedrigen Falsch-Positiv-Detektionen pro Bild. Mehrere Verfahren verwenden eine Segmentierung der Eingabebilder als Vorstufe. Dabei werden entsprechende Farbkanäle im normierten RGB- oder HSI-Farbraum durch einen oder mehrere Schwellwerte verändert, und die Farbinformation von farbigen Verkehrsschildern ausgenutzt. Allerdings sind diese Verfahren bei stark schwankenden Lichtverhältnissen instabil. Für Systeme, die in Dämmerung oder Dunkelheit funktionieren sollen, kann diese Vorstufe nicht angewendet werden, da die Farbinformation dann verloren geht. Ein Vergleich unterschiedlicher Segmentierungsverfahren für die Verkehrszeichendetektion ist in dem Artikel von Gómez-Moreno et ala [52] gegeben. Unterschiedliche Hough-Transformationen zum Auffinden der geometrischen Strukturen werden in der Arbeit von Houben [60] beschrieben und verglichen. In den Artikeln [50, 53, 86] werden Systeme beschrieben, die auf der Hough-Transformation basieren. Erwähnenswert ist ein weiteres Verfahren, der sogenannte "radial symmetry detector" [76, 8], der Ähnlichkeiten zur Hough-Transformation aufweist und auf reguläre Polygone bzw. Rechtecke erweitert wurde [77].

Unabhängig von einer eventuellen Segmentierung als Vorverarbeitungsschritt lässt sich der Klassifikationsschritt durch die Art der Merkmale, die verwendet werden, und den Klassifikationsmechanismus beschreiben. Merkmale, die häufig zur Verkehrzeichendetektion Verwendung finden, sind die HOG-Merkmale [31] (Histogram of Oriented Gradients), bei denen Kantenrichtungen abhängig von der Kantenstärke in lokalen Regionen in Histogramme akkumuliert werden, welche dann aneinandergereiht den mehrdimensionalen Merkmalsvektor ergeben. Zur Steigerung der Stabilität überlappen sich unterschiedliche Regionen und die Merkmale werden normiert. In den Artikeln [49, 126, 91, 3, 28] und [92] werden HOG-Merkmale zur Verkehrszeichendetektion verwendet. Eine weitere wichtige Klasse von Merkmalen sind die diskreten Haar-Wavelets [54], die einen orthogonalen Satz von Basisfunktionen bilden, die unter anderem zur Transformation von Bildsignalen in die Spektraldarstellung zur Signalcodierung und Signalübertragung verwendet werden. Die Haar-Wavelets sind durch eine rekursive Funktionalgleichung und einem "Vater-Wavelet" definiert und lassen sich im diskreten Fall als skalierte Differenzen von Stufenfunktionen darstellen. Im zweidimensionalen

Fall entsprechen die Funktionen Rechteckbereichen, die voneinander abgezogen werden. Die orthogonalen Haar-Merkmale wurden in [116] zur Gesichtsdetektion um weitere Rechteckgeometrien erweitert und bilden so einen überbestimmten Satz an Basisfunktionen, die "Haar-ähnliche Merkmale" genannt werden. Durch die Verwendung von sogenannten Integralbildern [116] können die Haar-ähnlichen Merkmale unabhängig von der Größe und Geometrie mit einer konstanten Anzahl von wenigen Integralbildzugriffen (vier Zugriffe für ein Rechteck) sehr effizient berechnet werden. Eine Erweiterung auf rotierte Merkmale und eine empirische Analyse ist in dem Artikel von Lienhart [73] gegeben. Im Bezug auf die Verkehrszeichendetektion werden Haar-ähnliche Merkmale in den Arbeiten [9, 67, 96, 6] verwendet, wobei das System von Keller [67] auf amerikanische Verkehrszeichen angewendet wird und Bahlmann et al. [6] die Haar-Merkmale auf bestimmte Farbkanäle anwenden. Als Klassifikatoren für die Detektion werden Support Vector Machines (SVMs) in den Artikeln [79, 71, 126] und [28] eingesetzt. Neuronale Netze werden von Nguwi et al. [87] eingesetzt.

Eine weitere interessante Klassifikatorstruktur ist der sogenannte Kaskadenklassifikator, der von Viola und Jones [116, 115] zur echtzeitfähigen Gesichtsdetektion mit den erweiterten Haar-ähnlichen Merkmalen eingesetzt wurde. Bei dieser Klassifikatorstruktur werden mehrere Stufenklassifikatoren hintereinandergeschaltet, indem ein zu klassifizierendes Beispiel die Stufen nacheinander durchläuft. Sobald ein Stufenklassifikator das Beispiel als Hintergrund klassifiziert, wird das Beispiel verworfen. Für eine erfolgreiche Detektion muss ein Beispiel alle Stufen durchlaufen. Die Stufenklassifikatoren bestehen wiederum aus mehreren sogenannten schwachen Klassifikatoren, die mittels einem Boostingverfahren wie AdaBoost [39] trainiert werden. Zur Verkehrszeichendetektion werden Kaskadenklassifikatoren mit Haar-ähnlichen Merkmalen ohne Segmentierungsschritt [6] und mit einer speziellen Segmentierung [67] verwendet. HOG- und HOG-ähnliche Merkmale werden mit einer Kaskade ohne Segmentierungsschritt verwendet [92, 91] sowie zusätzlich mit dem LogitBoost-Verfahren [41] und fünf Stufenklassifikatoren verwendet. Ein Kaskadenklassifikator mit "dissociated dipoles" als Merkmalen wird von Baró et al. [9] eingesetzt.

Die kaskadierte binäre Klassifikation wurde erstmals zur echtzeitfähigen Gesichtsdetektion von Viola und Jones vorgestellt [116, 115]. Bei der klassifikatorbasierten Objektdetektion wird im Allgemeinen eine sogenannte "Sliding-Window"-Detektion durchgeführt, bei der die Detektionsfenster, in denen nach Objekten gesucht wird, an allen möglichen Positionen und

Skalierungen an denen Objekte vermutet werden, generiert werden. Bei einer kompletten Bildsuche (full scan), die den Vorteil besitzt, dass Objekte überall im Bild gefunden werden können, wird abhängig von der Bildgröße eine große Menge an Detektionsfenstern im gesamten Bild generiert. Für jedes Fenster wird dann eine Klassifikation in Objekt- oder Hintergrundklasse durchgeführt. Falls eine Echtzeitanforderung an das Klassifikationssystem besteht, ist eine effiziente Klassifikation pro Detektionsfenster ausschlaggebend.

Kaskadierte Klassifikationssysteme haben mehrere Vorteile. Sie ermöglichen eine echtzeitfähige Objektdetektion ohne spezifischen Vorverarbeitungsschritt und sind damit schnell auf unterschiedliche Problemstellungen adaptierbar. Außerdem lassen sich, je nach verfügbarer Rechenleistung, eine hohe Anzahl unterschiedlicher Merkmale verwenden. Die Art der Stufenklassifikatoren ist theoretisch ebenfalls frei wählbar. Allerdings existieren sehr viele offene Fragen im Bezug auf die Realisierung und Automatisierung der Trainingsprozesse und der Klassifikationsmechanismen bei kaskadierten Klassifikationssystemen. Es existiert zum Beispiel kein Trainingsmechanismus der Stufenklassifikatoren, welcher sich an das entsprechende Teilproblem der jeweiligen Stufe anpasst. Die Struktur der Stufenklassifikatoren ist ein Designparameter. Zudem sind die Trainingsmechansimen der "schwachen Klassifikatoren" lediglich auf das Finden eines Schwellwertes beschränkt, da diese Klassifikatoren nur aus eindimensionalen Merkmalen bestehen. Des Weiteren resultiert aus der einfachen Hintereinanderschaltung der Stufenklassifikatoren eine Verschlechterung der Detektionsrate mit jeder neuen Stufe. Daher wird in den meisten Systemen eine möglichst geringe Anzahl an Stufenklassifikatoren verwendet, was allerdings komplexere Stufenklassifikatoren erfordert und damit die mittleren Laufzeiten des Kaskadenklassifikators erhöht.

Im Bezug auf die Boostingverfahren werden meist die Haar-ähnlichen Merkmale verwendet und es ist unklar, wie das System bei anderen Merkmalen reagiert. Darüber hinaus werden als Stufenklassifikatoren schwellwertbasierte Entscheidungsfunktionen verwendet, die lediglich eine Ja/Nein-Entscheidung liefern können. Mit einem Konfidenzwert für die Entscheidung könnten zum einen Muster mit gleicher Klassenzugehörigkeit differenziert bewertet werden und zum anderen würde dadurch die Tür zum teilüberwachten Lernen mit kaskadierten Klassifikationssystemen geöffnet werden, bei denen der Klassifikator die Möglichkeit besitzt, sich selbst Lernbeispiele für das Training auszuwählen. Das ist insbesondere bei der Detektion interessant, da eine überproportional große Menge an Hintergrunddaten existiert. Außerdem werden standardmäßig nur Falsch-Positiv-Beispiele für die Negativ-

Lernbeispiele der Stufen verwendet. Anhand der Konfidenzwerte kann eine stärkere Fokussierung auf Problembeispiele angewendet werden. Ein weitere offene Fragestellung ist die Art und Weise der Erzeugung der Bildausschnitte pro Bild bei der Anwendung im Detektionsschritt, die einen starken Einfluss auf die mittlere Laufzeit des Klassifikators nimmt. Für feine Merkmale muss auch eine hohe Anzahl an Ausschnitten verwendet werden um keine Objekte "zu übersehen", was jedoch auch bei schnell ausführbaren Merkmalen zu einer drastischen Erhöhung der Laufzeit führt und somit ein kritischer Punkt für die Echtzeitfähigkeit ist. Bei groben Merkmalsstrukturen ist hingegen die Trennfähigkeit der Klassifikatoren stark beeinträchtigt.

2.2 Hauptfragestellungen und Schwerpunkte

Da die Anzahl an rechnergestützten Systemen in mobilen Geräten, wie zum Beispiel bei Fahrerassistenzsystemen oder in Mobiltelefonen, immer stärker anwächst und damit die verfügbare Rechenleistung für die entsprechenden Anwendungen stark beschränkt ist, ist ein zentraler Schwerpunkt in dieser Arbeit die

- Entwicklung und Analyse von ressourcenoptimierten Klassifikationssystemen mit Mechanismen zur Regulierung der Optimierungskriterien Klassifikationsleistung und Laufzeit,

wobei kaskadierte Klassifikationssysteme untersucht werden. In den in der Literatur verfügbaren Verfahren beschränken sich die Optimierungskriterien lediglich auf den Klassifikationsfehler, ohne dass ein Kostenmodell für die Berechnungskosten in Betracht gezogen wird. Da die Klassifikatoren im industriellen Umfeld in Fahrerassistenzsystemen Anwendung finden, kann durch die somit verbesserten Laufzeiteigenschaften der Klassifikationsalgorithmen, mittels Einsatz in hohen Stückzahlen, eine beachtliche Energieersparnis in der Gesamtsumme erreicht werden.

Aus der Sicht der maschinellen Lernverfahren ist das teilüberwachte Lernen aktueller Forschungsgegenstand. Gerade bei Detektionsaufgaben existiert eine überproportional große Menge an Hintergrunddaten, die teilüberwachte Verfahren für Detektionsaufgaben besonders interessant machen. Aus dieser Menge an Hintergrunddaten kann sich der Klassifikator selbst "interessante oder neue" Beispiele zum Training aussuchen, wodurch die Klassifikationsleistung gesteigert werden kann. Andererseits existieren in dieser Men-

ge auch Objekte, die vom Menschen übersehen wurden oder die sehr hohe Ähnlichkeiten mit der Objektklasse aufweisen. Durch semi-überwachte Algorithmen können diese vom Klassifikationsalgorithmus identifiziert werden und in die Objektklasse für das Training übernommen werden. Der zweite Schwerpunkt in dieser Arbeit ist deshalb die

- Entwicklung und Analyse von ressourcenoptimierten, teilüberwachten Trainingsverfahren zur integrierten Kaskadenklassifikation.

In der Literatur existieren wenige Verfahren die auf Kaskadenklassifikatoren basieren. Die meisten davon sind als Wrapper-Ansätze formuliert. Allerdings bietet es sich gerade bei kaskadierten Klassifikatoren durch die Hinzunahme weiterer Stufen und die Auswahl von Trainingsbeispielen für die entsprechenden Stufen an, integrierte teilüberwachte Trainingsverfahren einzusetzen. Um die teilüberwachten Algorithmen für kaskadierte Systeme zu realisieren werden in dieser Arbeit Methoden zur Konfidenzwertberechnung entwickelt. Ein weiterer Vorteil der teilüberwachten Lernverfahren ist der reduzierte Aufwand zur Annotation der Objekte, da weniger Daten benötigt werden. Um diese Aufgaben zu lösen wird im nächsten Abschnitt ein Wahrscheinlichkeits- und Kostenmodell für kaskadierte Klassifikationssysteme vorgestellt.

2.3 Induktive Bayes‘sche Statistik kaskadierter Klassifikationssysteme

Im Folgenden wird das grundlegende Verfahren beschrieben, das in dieser Arbeit entwickelt wurde. Im Gegensatz zu den klassischen Verfahren wird durch die Bayes'sche Formulierung eine getrennte Fehlerbetrachtung möglich, indem über ein Zustandsmodell die Entwicklung der Wahrscheinlichkeiten in den unterschiedlichen Stufenklassifikatoren berechnet und deren Entwicklung verfolgt wird. Somit werden die durch die Stufenklassifikatoren induzierten Auftrittswahrscheinlichkeiten für die Entscheidungen der nachfolgenden Klassifikatoren verwendet. Die im Vorfeld dieser Arbeit publizierten eigenen Arbeiten des Autors sind im Folgenden mit "*" gekennzeichnet.

Die Bayes'sche Formulierung mit den entsprechenden Algorithmen wurde zum Teil in dem Artikel [108]* vorgestellt. Außerdem werden die Signifikanzniveaus der Stufenklassifikatoren verändert, um hohe Auftrittswahrscheinlichkeiten der Objektklasse zu garantieren, indem eine getrennte Feh-

leranalyse zur Optimierung der Klassifikatoren vollzogen wird. Darüberhinaus ermöglicht diese Formulierung eine fusionierte Konfidenzberechnung aller Stufenklassifikatoren, was ebenfalls in dem Artikel [108]* beschrieben ist. Anhand dieser Konfidenzwerte wird ein neues, in das Kaskadentraining integriertes, teilüberwachtes Verfahren zur Echtzeitklassifikation vorgestellt, dessen Detektionsraten diejenigen des vollständig überwachten Verfahrens übertreffen. Weiterhin wird mit der wahrscheinlichkeitsbedingten Formulierung ein wiederum neues kombinatorisches Optimierungsverfahren vorgestellt, bei dem die wahrscheinlich auftretenden Operationen zur Optimierung verwendet werden. Dieses neue Verfahren wurde erstmals in der Publikation [107]* beschrieben.

Das Ziel der Optimierung von Klassifikatoren, die zu bildbasierten Detektionsaufgaben im Echtzeitbereich eingesetzt werden können, ist es, sowohl hohe Detektionsraten zu erzielen, als auch ein schnelles Detektionsergebnis zu ermöglichen. Es handelt sich also um ein multikriterielles Optimierungsproblem mit den Kriterien Detektionsleistung und Laufzeit. Ein Verfahren, das sich als effektiv für diese Art von Problemen erwiesen hat, ist die Kombination eines “Sliding Window”-Verfahrens (SLW) mit einem kaskadierten Klassifikationssystem [116, 115]. Bei dem SLW-Verfahren wird eine große Menge an Bildausschnitten bzw. Fenstern an unterschiedlichen translatorischen Positionen und in unterschiedlichen Skalen generiert, die nötig ist um den Suchraum komplett abzudecken. Jeder Bildausschnitt wird dann von einem Klassifikator als Objekt oder als Hintergrund klassifiziert, wobei der Klassifikator sehr kurze Laufzeiten für die Klassifikation eines einzelnen Bildausschnitts aufweisen muss, um eine effiziente Gesamtlaufzeit pro Bild zu ermöglichen. Die effiziente Klassifikation eines Fensters wird mit dem Kaskadenklassifikator durch eine geordnete Hintereinanderschaltung von Einzelklassifikatoren erzeugt, die alle die Möglichkeit besitzen, einen Bildausschnitt als Objekt zu verwerfen und als Hintergrund zu klassifizieren. Diese verworfenen Bildausschnitte müssen dann auch nicht mehr von den restlichen Klassifikatoren klassifiziert werden, sondern lediglich die als Objekte klassifizierten Ausschnitte.

Es wird somit ein reduktionistischer Ansatz verfolgt, indem eine anfangs sehr große Kandidatenmenge auf eine immer kleiner werdende Menge durch eine kausale Entscheidungskette reduziert wird, welche durch die geordneten Entscheidungen der Einzelklassifikatoren beeinflusst werden. Eine direkte Folge dieser Struktur ist eine immer stärker wachsende Komplexität der Teilprobleme, die jeder Einzelklassifikator zu lösen hat, da eine immer größere

Ähnlichkeit zwischen Objekt- und Hintergrundklasse auftritt.

Ein zentraler Ansatz für die Erzeugung kaskadierter Klassifikationssysteme in dieser Arbeit ist die statistische Beschreibung dieser Systeme durch einen Bayes-Ansatz [69], anhand von bedingten Wahrscheinlichkeiten, die mathematisch durch das Bayestheorem beschrieben werden können [108]*. Dieser Ansatz hat mehrere Vorteile.Zum einen wird die steigende Komplexität der Teilprobleme durch bedingte Wahrscheinlichkeiten beschrieben, und die Teilprobleme können isoliert betrachtet werden, indem die durch die Einzelklassifikatoren induzierten Veränderungen der Wahrscheinlichkeiten miteinbezogen werden. Außerdem können die Wahrscheinlichkeiten für die Steuerung der Algorithmen verwendet werden, woraus sowohl eine adaptive Regelung der Gesamtstruktur, also der Anzahl an Einzelklassifikatoren, als auch eine Regelung der Struktur der Einzelklassifikatoren resultiert, indem die Wahrscheinlichkeiten als Regelkriterien für die Algorithmen verwendet werden.
Des Weiteren können Konfidenzwerte berechnet werden, die aus den Einzelkonfidenzen aller Klassifikatoren fusioniert werden und ein Unterscheidungsmerkmal für Objekte derselben Klasse liefern, was wiederum verwendet wird um ähnliche Klassifikationsergebnisse zu einem über die Konfidenzen gewichteten Gesamtergebnis zu vereinen. Um das multikriterielle Optimierungsproblem zu lösen wird ein Kostenmaß zur Berechnung der erwarteten Berechnungskosten eingeführt, das eine problem- und stufenabhängige Auswahl der Einzelklassifikatoren während dem Klassifikatortraining ermöglicht und somit die strukturelle Komplexität der Einzelklassifikatoren steuert und die Verwendung von unterschiedlichen Klassifikatortypen, sowie Merkmale mit stark differierenden Berechnungskosten ermöglicht. Außerdem wird anhand der Konfidenzwerte ein neuer, semiüberwachter Trainingsalgorithmus zur Echtzeitdetektion entwickelt, welches ein neues integriertes Kaskadenverfahren beschreibt, das dem Klassifikator die Möglichkeit einräumt, die Klassenzugehörigkeit von Beispielen ohne weiteres menschliches Eingreifen umzudefinieren. Darüber hinaus können durch die probabilistische Beschreibung höchst modulare Verfahren und Strukturen entwickelt werden, bei denen die Typen der Einzelklassifikatoren ausgetauscht werden können. Im Folgenden werden die statistischen Eigenschaften der Einzelklassifikatoren, sowie die des Kaskadenklassifikators, beschrieben.

2.3.1 Klassifikation

Für ein Detektionssystem gibt es für gewöhnlich zwei unterschiedliche Klassen. Die Klasse der Objekte, die durch ω_1 beschrieben wird, und die Hintergrundklasse ω_0 werden durch eine endliche Menge an Beispielen $\mathcal{D}_1 = \{(\underline{x}_i, y_i), \ldots, (\underline{x}_N, y_N)\}$ und einer Menge $\mathcal{D}_0 = \{(\underline{x}_i, y_i), \ldots, (\underline{x}_N, y_N)\}$ bestehend aus Hintergrundbeispielen repräsentiert, wobei die Klassenzugehörigkeit durch die Werte $\underline{y}_i = y(\underline{x}_i) = \omega_1$ für die Objektklasse und $\underline{y}_i = y(\underline{x}_i) = \omega_0$ für die Hintergrundklasse beschrieben wird. Die Werte $\underline{x}_i$ sind Werte einer Zufallsvariablen $\underline{X} \in \mathbb{R}^N$ und werden für gewöhnlich durch einen ein- oder mehrdimensionalen Merkmalsvektor beschrieben. Die Klassenkategorien ω_i sind ebenfalls Werte einer Zufallsvariablen $Y \in \{\omega_0, \ldots, \omega_{c-1}\}$, wobei beim Detektionsfall zwischen zwei Klassen unterschieden wird.

Die Art und Anzahl der Merkmale hat großen Einfluss auf die Detektions- und auch Laufzeiteigenschaften des Klassifikators. Deshalb ist die Merkmalsbestimmung ein wichtiger Bestandteil des Klassifikatortrainings. Die Verteilung der Zufallsvariablen hängt von der unbekannten tatsächlichen Verteilung ab und wird durch $P(\underline{X} = \underline{x}|Y = \omega_i, \mathcal{D}_i)$, der klassenabhängigen Wahrscheinlichkeitsdichte, beschrieben. Um eine Entscheidung zu treffen ist die A-posteriori-Wahrscheinlichkeit $P(Y = \omega_i|\underline{X} = \underline{x})$ gesucht. Außerdem gibt es noch die A-priori-Wahrscheinlichkeit $P(Y = \omega_i)$, welche die Auftrittswahrscheinlichkeit des Ereignisses im Voraus beschreibt. Aus Übersichtlichkeitsgründen werden im Folgenden die eigentlichen Zufallsvariablen nicht mehr explizit dargestellt. Die Bayesformel stellt diese Größen in Bezug [35]

$$P(\omega_i|\underline{x}, \mathcal{D}) = \frac{P(\underline{x}|\omega_i, \mathcal{D}_i)P(\omega_i)}{\sum_{j=1}^{c} P(\underline{x}|\omega_j, \mathcal{D}_j)P(\omega_j)} \tag{2.3.1}$$

Das Grundproblem des Bayes'schen Lernens ist es also, aus c unterschiedlichen Mengen an Beispielen, die unabhängig voneinander gezogen werden und von einer unbekannten Verteilung stammen, die klassenabhängige Wahrscheinlichkeitsdichte

$$P(\underline{x}(h, \theta_1^h, \ldots, \theta_N^h)|\omega_j, \mathcal{D}_j) \tag{2.3.2}$$

zu bestimmen [35], denn die Regel zur Bestimmung der A-posteriori-Wahrscheinlichkeit ist durch die Bayesformel gegeben. Die klassenbedingten Wahrscheinlichkeitsdichten sind auf eine Gesamtfläche von Eins normiert.

Hier soll verdeutlicht werden, dass im Falle dieser Betrachtung die Zufallsvariable $\underline{x}$ eine von einem *Klassifikator* h abhängige Größe ist, die wiederum von mehreren Parametern θ_i^h abhängt und anhand von den Trainingsbeispielen $\mathcal{D}$ mit einem *Trainingsalgorithmus* bestimmt werden. Für das Zweiklassenproblem ist ein *binärer Klassifikator* bzw. eine Hypothese h durch seine Klassifikationsfunktion gegeben,

$$h(\underline{x}(\theta_1, \ldots, \theta_N)) = \begin{cases} \omega_1 & \text{Klassifikator wählt Objektklasse} \\ \omega_0 & \text{Klassifikator wählt Hintergrundklasse} \end{cases} \tag{2.3.3}$$

der die Repräsentation $\underline{x}$ eines Beispiels entweder als Objekt, das zur Klasse ω_1 gehört, oder als Hintergrund ω_0 klassifiziert. Die Abhängigkeit der Zufallsvariable von den Parametern wird aus Übersichtlichkeitsgründen im Folgenden weggelassen, soll aber weiterhin für die Formeln gelten. Die Arten von Trainingsalgorithmen hängen vom Klassifikatortyp ab und können die Laufzeiten und Detektionsleistungen stark beeinflussen. Im Folgenden sei ein *eigenständiger Klassifikator* als Klassifikator definiert, der nicht von den Entscheidungen anderer Klassifikatoren abhängt. Die Trainingsalgorithmen für diese Klassifikatoren werden in Kapitel 3 erklärt. Im Allgemeinen werden Beispiele nicht fehlerfrei klassifiziert, sondern es tauchen Zuordnungen zu den falschen Klassen auf. Indem man die relativen Häufigkeiten der Fehler analysiert können quantitative Fehlermaße abgeleitet werden, um die Klassifikationsleistung eines Klassifikators zu beurteilen. Für einen binären Klassifikator gibt es vier unterschiedliche Entscheidungszustände. Zwei fehlerfreie Zustände, Richtig-Positiv und Richtig-Negativ, also die richtige Zuordnung eines negativen und eines positiven Beispiels und zwei fehlerhafte Zustände, Falsch-Positiv und Falsch-Negativ, wobei ein negatives Beispiel als positiv und ein positives Beispiel als negativ gewählt wurde. In Tabelle 2.3.1 sind die unterschiedlichen Zustände aufgelistet.

Mit den Entscheidungszuständen können die entsprechenden Detektionsraten für eine Menge an Beispielen berechnet werden (2.3.2). Diese Raten können auch als approximierte Wahrscheinlichkeiten interpretiert werden [112, 35], und werden in dieser Arbeit für die Regelung der Trainingsalgorithmen und für die Klassifikationsfunktionen verwendet. Die Klassifikationsfunktion selbst kann ebenfalls als Zufallsvariable interpretiert werden und die binäre Relation die von den atomaren Ergebnissen $h(\underline{x}_i) \in \underline{X}$ und $y(\underline{x}_i) \in Y$ abhängig ist, und deren Realisierungen die vier Entscheidungszustände TP,

Entscheidungszustände		
wahr / gewählt	$y(\underline{x}) = \omega_1$	$y(\underline{x}) = \omega_0$
$h(\underline{x}) = \omega_1$	Richtig-Positiv (TP)	Falsch-Positiv (FP)
$h(\underline{x}) = \omega_0$	Falsch-Negativ (FN)	Richtig-Negativ (TN)

Tabelle 2.3.1: Konfusionsmatrix eines binären Klassifikators. Die obere Reihe zeigt die tatsächlichen Klassen, ω_1 für Objekte und ω_0 für Hintergrund. Die linke Spalte zeigt die vom Klassifikator h gewählten Klassen.

Rate	**Definition**
Sensitivität (TPR)	Anzahl Richtig-Positiv / Anzahl positiv
Falsch-Positiv-Rate (FPR)	Anzahl Falsch-Positiv / Anzahl negativ
Spezitivität (TNR)	Anzahl Richtig-Negativ / Anzahl negativ
Falsch-Negativ-Rate (FNR)	Anzahl Falsch-Negativ / Anzahl positiv

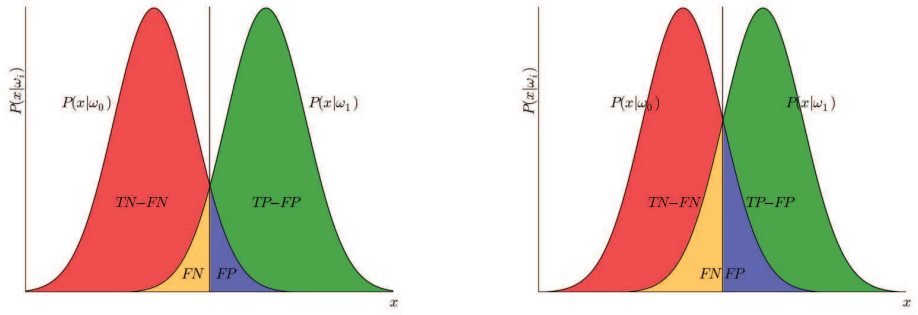
Tabelle 2.3.2: Detektionsraten für einen binären Klassifikator. Jede Rate nimmt Werte zwischen Null und Eins an.

FP, TN und FN sind. In Tabelle 2.3.3 sind die entsprechenden Zufallsvariablen mit den Realisierungen aufgelistet. Abb. 2.3.1 zeigt, wie die Detektionsraten mit den klassenabhängigen Wahrscheinlichkeitsdichten im Falle von zwei Gaussverteilungen über einer skalaren Zufallsvariable x und einer schwellwertbasierten Entscheidungsfunktion zusammenhängen. Die Entscheidungsgrenze wird durch einen Schwellwert θ dargestellt, wobei alle Beispiele rechts des Schwellwertes als positiv und alle Beispiele links des Schwellwertes als negativ klassifiziert werden. Die grüne Fläche stellt die Richtig-Positiven abzüglich den Falsch-Positiven dar, deshalb entsprechen die Richtig-Positiven der Vereinigung der grünen und blauen Flächen. Die Entscheidungsgrenze wurde so gesetzt, dass der Fehler minimal wird. Diese Entscheidung wird Bayes Entscheidung genannt.

Abb. 2.3.1(a) zeigt zwei Verteilungen, die eine bessere Unterscheidung und damit eine geringere Fehlerrate ermöglichen im Vergleich zu Abb. 2.3.1(b). Dort existiert eine größere Überschneidung der Verteilungen und damit ein größerer Fehler, was deutlich an den größeren Fehlerflächen erkennbar ist. Ein zentraler Teil beim Klassifikatortraining ist es deshalb, geeignete Merkmale zu finden, welche die Zufallsvariablen repräsentieren und eine gute Trennung der Klassen ermöglichen, woraus eine niedrige Fehlerrate resultiert. Eine wei-

Zufallsvar.	**Abbildung**	**Ergebnismenge**	**Realisierungen**
X	$id_{\underline{x}}(\underline{x})$	$\mathbb{R}^n$	$\mathbb{R}^n$
Y	$y(\underline{x})$	$\mathbb{R}^n$	$\{\omega_0, \omega_1\}$
H	$h(\underline{x})$	$\mathbb{R}^n$	$\{\omega_0, \omega_1\}$
HY	$(h(\underline{x}), y(\underline{x}))$	$(\{\omega_0, \omega_1\}, \{\omega_0, \omega_1\})$	$\{TP = (\omega_1, \omega_1),$ $FP = (\omega_1, \omega_0),$ $FN = (\omega_0, \omega_1),$ $TN = (\omega_0, \omega_0)\}$

Tabelle 2.3.3: Relevante Zufallsvariablen mit den entsprechenden Entscheidungszuständen und korrespondierenden Abbildungen, den Ergebnismengen und den entsprechenden Realisierungen.



(a) (b)

Abbildung 2.3.1: Normalverteilungen für Objekt- und Hintergrundbeispiele über einer skalaren Zufallsvariablen x. Die rechte Verteilung stellt die Wahrscheinlichkeitsdichte der Objektklasse dar, die linke Verteilung die der Hintergrundklasse. Um den Zusammenhang mit den Entscheidungszuständen (TP,FP,TN,FN) zu zeigen, ist eine Entscheidungsgrenze bei θ eingezeichnet.

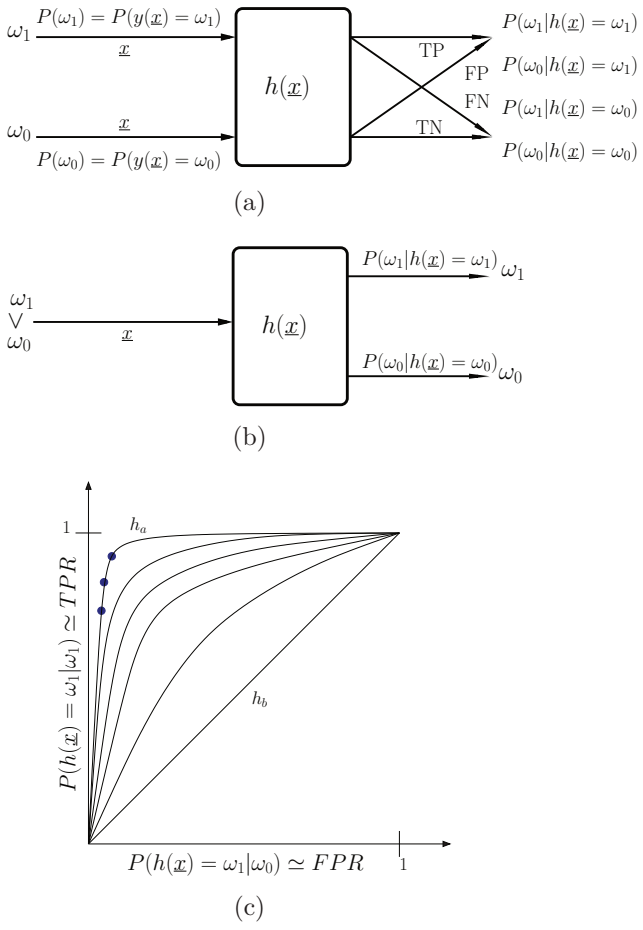


Abbildung 2.3.2:
(a): Der Wahrscheinlichkeitsgraph für bekannte Klassenzugehörigkeiten zeigt die Auswirkung einer Klassifikation durch den Klassifikator $h(x, \theta)$ auf die A-priori-Wahrscheinlichkeiten der Beispiele der Objekt und Hintergrundklasse für die zwei möglichen Entscheidungen $h(x) = \omega_1$ und $h(x) = \omega_0$.
(b): Der Wahrscheinlichkeitsgraph für unbekannte Klassenzugehörigkeiten als Anwendungsfall. Der Klassifikator schätzt die Klassenzugehörigkeit. Je kleiner der Fehler des Klassifikators ist, desto besser ist die Schätzung.
(c): Receiver operating characteristic (ROC) für gleiche A-priori-Wahrscheinlichkeiten $P(\omega_1) = P(\omega_0)$ und damit abhängig von den klassenabhängigen Wahrscheinlichkeitsdichten. Die Abszisse stellt die Wahrscheinlichkeit eines Falschalarms $P(h(x) = \omega_1|\omega_0)$ dar, die Ordinate die Wahrscheinlichkeit eines Treffers $P(h(x) = \omega_1|\omega_0)$.

tere statistische Eigenschaft der Klassifikation ist die entscheidungsbedingte Veränderung der A-posteriori-Wahrscheinlichkeiten für die Ausgänge eines Klassifikators. Aus Abb. 2.3.1(a) wird ersichtlich, dass eine Verschiebung der Entscheidungsgrenze θ nach rechts eine Absenkung der Wahrscheinlichkeit einer korrekten Zuordnung zu Klasse ω_1 zur Folge hat und gleichzeitig eine Erhöhung der Wahrscheinlichkeit für die Zuordnung zu Klasse ω_0. Dadurch wird gleichermaßen die Autrittswahrscheinlichkeit an den Ausgängen des Klassifikators von Beispielen, die zur Klasse ω_1 gehören gesenkt, und die Auftrittswahscheinlichkeiten der Beispiele von Klasse ω_0 erhöht. Beim Verschieben der Grenze nach links tritt der umgekehrte Fall ein. Betrachtet man nun einen bestimmten Ausgang des Klassifikators, so wird die A-posteriori-Wahrscheinlichkeit zur A-priori-Wahrscheinlichkeit für die jeweilige Klasse. Dieser Sachverhalt ist durch einen Wahrscheinlichkeitsgraphen in Abb. 2.3.2(a) dargestellt, wobei die Klassenzugehörigkeiten eines Beispiels bekannt sind. Dadurch ist eine Beurteilung der Klassifikationsgüte möglich. Der obere und untere linke Pfeil beschreibt den Eingang von positiven (oben) und negativen (unten) Beispielen mit den zugehörigen A-priori-Wahrscheinlichkeiten in den Klassifikator. An den Ausgängen existiert zu jedem Entscheidungszustand eine entsprechende gerichtete Kante, die auf die zwei möglichen Klassen ω_1 oben und ω_0 unten zeigen. Die Kante FP bedeutet zum Beispiel, dass ein negatives Beispiel einer positiven Klasse zugeordnet wurde. Fehlklassifikationen bewirken also eine Veränderung der Auftrittswahrscheinlichkeiten an den entsprechenden Ausgängen ω_1 und ω_0. Für jeden Entscheidungszustand kann die entsprechende A-posteriori-Wahrscheinlichkeit berechnet werden, sobald die klassenbedingten Wahrscheinlichkeitsdichten der Zufallsvariablen sowie die Klassifikationsfunktionen $h(\underline{x})$ und die A-priori-Wahrscheinlichkeiten, bekannt sind. Es existieren vier Fälle, ein Fall pro Entscheidungszustand:

$$
\begin{aligned}
P(\omega_1 = y(\underline{x}) \,|\, h(\underline{x}) = \omega_1) &: \text{ Posterior für Klasse } \omega_1 \text{ und Entscheidung } \omega_1,\\
P(\omega_0 = y(\underline{x}) \,|\, h(\underline{x}) = \omega_1) &: \text{ Posterior für Klasse } \omega_0 \text{ und Entscheidung } \omega_1,\\
P(\omega_1 = y(\underline{x}) \,|\, h(\underline{x}) = \omega_0) &: \text{ Posterior für Klasse } \omega_1 \text{ und Entscheidung } \omega_0,\\
P(\omega_0 = y(\underline{x}) \,|\, h(\underline{x}) = \omega_0) &: \text{ Posterior für Klasse } \omega_0 \text{ und Entscheidung } \omega_0.
\end{aligned}
$$

Im Folgenden sollen die Zusätze $\omega_i = y(\underline{x})$ implizit angenommen werden. Abb. 2.3.2(b) zeigt den Anwendungsfall eines Klassifikators für unbekannte Klassenzugehörigkeiten, wobei der Klassifikator selbst die Klassenzugehörig-

keit schätzt. Die A-posteriori-Wahrscheinlichkeiten, die von dem Klassifikator $h(\underline{x})$ erzeugt werden, können mit den Wahrscheinlichkeitsdichten $P(\underline{x}|\omega_i)$ berechnet werden:

$$P_{t+1}(\omega_i|h(\underline{x})=\omega_j) = P_t(\omega_i)P_t(h(\underline{x})=\omega_j|\omega_i) \;= P_t(\omega_i)\frac{\int_N P_t(\underline{x}|\omega_i)\mathrm{d}\underline{x}}{\int_D P_t(\underline{x}|\omega_i)\mathrm{d}\underline{x}} \tag{2.3.4}$$

$$\begin{aligned} &\text{mit} \\ N &= \{\underline{x} \in \underline{X}|h(\underline{x})=\omega_j\} \\ D &= \{\underline{x} \in \underline{X}|y(\underline{x})=\omega_i\} \end{aligned}$$

wobei $0 \leq i,j \leq 1$ und $P(w_i)$ die A-priori-Wahrscheinlichkeiten sind. Es ist hervorzuheben, dass diese Formulierung für beliebige Entscheidungsregionen gilt, da die Entscheidungsregionen durch die Klassifikationsfunktion h implizit festgelegt sind. Der Bruch in Gl. (2.3.4) beschreibt das gewichtete Verhältnis der Hypothesen des Klassifikators zu den tatsächlichen Klassenzuordnungen, indem über alle Entscheidungen integriert wird. Außerdem sind die klassenspezifischen Wahrscheinlichkeitsdichten auf den Flächeninhalt eins normiert, wodurch sich die weiteren folgenden Zusammenhänge ergeben:

$$\begin{aligned} P(h(\underline{x})=\omega_1|\omega_1) + P(h(\underline{x})=\omega_1|\omega_0) &= 1 \\ P(h(\underline{x})=\omega_0|\omega_1) + P(h(\underline{x})=\omega_0|\omega_0) &= 1 \end{aligned} \tag{2.3.5}$$

Falls die klassenspezifischen Wahrscheinlichkeitsdichten in Gl. (2.3.4) unbekannt sind, kann $P(h(\underline{x})=\omega_j|\omega_i)$ anhand des Klassifikationsergebnisses approximiert werden:

$$\begin{aligned} P(h(\underline{x})=\omega_1|\omega_1) &\approx TP/(TP+FN) = TPR \\ P(h(\underline{x})=\omega_1|\omega_0) &\approx FP/(TP+FN) = FPR \\ P(h(\underline{x})=\omega_0|\omega_1) &\approx FN/(TN+FP) = FNR \\ P(h(\underline{x})=\omega_0|\omega_0) &\approx TN/(TN+FP) = TNR \end{aligned}$$

Zielfunktion	$L(y_i, h(x_i))$	$-\partial L(y_i, f(x_i))/\partial h(x_i)$
Quadratischer Fehler	$\frac{1}{2}(y_i - h(x_i))^2$	$y_i - h(x_i)$
Betragsmäßiger Fehler	$\lvert y_i - h(x_i)\rvert$	$\text{sign}(y_i - h(x_i))$

Tabelle 2.3.4: Unterschiedliche Zielfunktionen und deren Ableitungen für die Optimierung von Klassifikationsproblemen [56].

Dieser Zusammenhang wir ebenfalls aus Abb. 2.3.2(a) ersichtlich. Ein Klassifikator kann als Schätzfunktion interpretiert werden, die weitere statistische Eigenschaften besitzt, welche als Kriterien für die Qualität verwendet werden. Eine wichtige Eigenschaft ist die Erwartungstreue. Ein Schätzer heißt erwartungstreu, wenn sein Erwartungswert dem wahren Wert der zu schätzenden Funktion entspricht. Ist eine Schätzfunktion nicht erwartungstreu, spricht man davon, dass der Schätzer verzerrt ist. Das Ausmaß der Abweichung seines Erwartungswerts vom wahren Wert nennt man Verzerrung oder Bias [122]. Je größer die Stichprobe an Beispielen, desto besser werden die tatsächlichen Wahrscheinlichkeiten angenähert.

Die *Optimierungskriterien* bezüglich der Klassifikationsleistung können nun anhand der Wahrscheinlichkeiten formuliert werden. Für den Fall, dass die A-priori-Wahrscheinlichkeiten bekannt sind, bezieht sich die Optimierung dabei auf die normierten klassenspezifischen Wahrscheinlichkeitsdichten. Das Optimierungskriterium für die Minimierung des Fehlers lautet:

$$\min_{h(\underline{x})}(P(h(\underline{x}) = \omega_i | \omega_j, \mathcal{D}_j)) \text{ mit } i \neq j \text{ und } 0 \leq, i, j \leq 1 \tag{2.3.6}$$

wobei eine Trennung der Fehlerzustände gegeben ist, nämlich der Falsch-Positiv Zustand für $i = 1, j = 0$ und der Falsch-Negativ Zustand mit $i = 0, j = 1$. Es werden also die Wahrscheinlichkeiten von Fehlklassifikationen minimiert, zusätzlich kann aber getrennt auf die spezifischen Fehlerzustände eingegangen werden. Mittels Gl. (2.3.5) können äquivalente Maximierungskriterien formuliert werden. Im Allgemeinen verwenden Optimierungsverfahren eine Zielfunktion L, die für den Fall der Minimierung meistens ein quantitatives Fehlermaß beinhaltet. Eine ausführliche Beschreibung der statistischen Zusammenhänge der Zielfunktion L ist in dem Buch [56] gegeben. In Tab. 2.3.4 sind häufig verwendete Zielfunktionen und deren Ableitungen aufgelistet.

Durch Variation der Klassifikationsfunktion bei konstanten Merkmalen verändert sich das Verhältnis der bedingten Wahrscheinlichkeiten bzw. das Verhältnis von Falsch-Positiv-Rate zur Sensitivität. Die variierenden Klassifikationsraten können zum Vergleich von unterschiedlichen Klassifikatoren benutzt werden, indem die Klassifikationsfunktion so eingestellt wird, dass für alle Klassifikatoren entweder eine identische Falsch-Positiv-Rate oder eine identische Sensitivität erreicht wird. Danach kann derjenige Klassifikator, der den geringsten Fehler an der nicht eingestellen Klassifikationsrate erzeugt, gewählt werden. Eine weit verbreitete Visualisierungsmethode ist die sogenannte Receiver Operating Curve (ROC) [35], welche durch die variierenden Wahrscheinlichkeiten $(P(h(\underline{x}) = \omega_1|\omega_0), P(h(\underline{x}) = \omega_1|\omega_0))$ für einen Klassifikator erzeugt wird, indem die unterschiedlichen 2D-Punkte aufgetragen werden. In Abb. 2.3.2(c) sind ROC-Kurven für mehrere Klassifikatoren eingezeichnet. Jeder Klassifikator bewegt sich dabei auf einer Kurve, was durch die blauen Punkte veranschaulicht wird. Die fehlerfreien Klassifikatoren sitzen auf dem Punkt $(0, 1)$. Da die ROC-Kurven eine direkte Interpretation mit Wahrscheinlichkeiten zulassen, werden in dieser Arbeit ausschließlich ROC-Kurven zur Beurteilung von Klassifikatoren verwendet.

Der Zusammenhang der A-posteriori- und A-priori-Wahrscheinlichkeit mit einen Klassifikator h, der in Gl. (2.3.4) hergestellt wurde, kann direkt zur Herleitung von Inferenzregeln und Trainingsalgorithmen bei komplexeren Klassifikationssystemen, die aus mehreren Basisklassifikatoren bestehen, angewendet werden. Die Klassifikationsfunktion des Klassifikatorsystems verwendet die einzelnen Klassifikationen der Basisklassifikatoren. Da ein Schwerpunkt dieser Arbeit die Echtzeitfähigkeit ist, wird eine spezifische Klassifikatorstruktur, der sogenannte Kaskadenklassifikator [116, 115], als Zielklassifikator verwendet, was im nächsten Abschnitt erläutert wird.

2.3.2 Kaskadierte Klassifikatoren

Wenn man die klassenspezifischen Wahrscheinlichkeiten als zeitlich abhängige Funktion betrachtet, dann kann man die Änderung der Wahrscheinlichkeitsdichten in einem Zeitschritt δt, die durch Anwendung einer Klassifikationsfunktion h induziert wird, unter Anwendung der Bayesformel in Gl.

(2.3.4) als Integral formulieren:

$$P(t+\delta t, \omega_i \,|\, \underline{x}(h(t)) = \omega_j) = P(t, \omega_i) \frac{\int_N P(t, \underline{x}|\omega_i)}{\int_D P(t, \underline{x}|\omega_i)} \qquad (2.3.7)$$

mit

$$N = \{\underline{x} \in \underline{X} | h(t, \underline{x}) = \omega_j\}$$
$$D = \{\underline{x} \in \underline{X} | y(\underline{x}) = \omega_i\}$$

Führt man mehrere Klassifikationen hintereinander aus, kann das Klassifikationsergebnis iterativ verbessert werden, indem die resultierenden Fehlklassifikationen $P(\omega_j|h(\underline{x}) = \omega_i)$ mit $i \neq j$ durch einen weiteren Klassifikator $h(t + \delta t)$ weiter reduziert werden. Im diskreten Fall wird dann die zeitliche Klassifikationsfunktion $h(t)$ durch eine geordnete Klassifikatorstruktur approximiert, indem eine zeitliche Diskretisierung durchgeführt wird. Diese Idee führt auf den *Kaskadenklassifikator* [116, 115]. Er ist eine geordnete Struktur von sogenannten *Stufenklassifikatoren*, wobei positiv klassifizierte Beispiele von einem Stufenklassifikator an den nächsten weitergereicht werden und negativ klassifizierte Beispiele sofort verworfen werden. Die Entscheidungsfunktion eines Kaskadenklassifikators h_C, der aus $h_1, \ldots, h_T$ Stufenklassifikatoren besteht, lässt sich wie folgt formulieren [114]:

$$h_C^T(\underline{z}) = (h_1(\underline{x}_1(\underline{z}), \ldots, h_T(\underline{x}_T(\underline{z})) = \begin{cases} \omega_0 & \text{falls } h_t(\underline{x}_t(\underline{z})) = \omega_0, \text{ für } t \in \{1, \ldots, T\} \\ \omega_1 & \text{sonst} \end{cases} \qquad (2.3.8)$$

Da die positiven Ausgaben der nachfolgenden Stufe übergeben werden, steigt die Komplexität der Teilprobleme in Abhängigkeit der Stufenklassifikatoren von Stufe zu Stufe an. Gl. (2.3.8) soll außerdem verdeutlichen, dass die Stufenklassifikatoren unterschiedliche Zufallsvariablen generieren, indem aus der Eingabe $\underline{z} \in \underline{Z}$ unterschiedliche Werte der Zufallsvariablen $\underline{x}_t(\underline{z})$ extrahiert werden. Diese sollten an das entsprechende Klassifikationsproblem angepasst sein, wobei dies Teil der Klassifikatoroptimierung ist. Ein Optimierungskriterium für die Zufallsvariablen wird im folgenden Abschnitt vorgestellt.

Die Ausgabe-Eingabe-Verkettung bei der Kaskade ist in Abb. 2.3.3(a) dargestellt. Wie im vorherigen Abschnitt erläutert, verändert jeder Stufenklassifikator damit die A-posteriori-Wahrscheinlichkeiten, wobei in diesem

Fall die negativen Entscheidungen nicht weitergeleitet werden. Da nur die positive Entscheidung $h_t(\underline{x}) = \omega_1$ weiterverfolgt wird, wird die A-posteriori-Wahrscheinlichkeit zur A-priori-Wahrscheinlichkeit für den darauf folgenden Klassifikator.

Abb. 2.3.3(b) zeigt den Wahrscheinlichkeitsgraph, der durch die Verkettung von mehreren Klassifikatoren anhand der Kaskadenstruktur entsteht. Solange kein Klassifikator involviert ist, befindet sich ein Beispiel $\underline{x}$ in einem der tatsächlichen Zustände ω_1 und ω_0. Nach einer Klassifikation erfolgt dann, je nach Entscheidung, der Übergang in einen der vier Zustände TP, FP, TN oder FN. Der Wahrscheinlichkeitsgraph zeigt den Fall für bekannte Klassenzugehörigkeiten und ist damit für den Trainingsalgorithmus ausschlaggebend, da sich die entsprechenden Wahrscheinlichkeiten der Zustände berechnen lassen. Abb. 2.3.3(c) zeigt den Entscheidungsgraph für den Anwendungsfall eines Kaskadenklassifikators. Ein Beispiel befindet sich in einem unbekannten Zustand und der tatsächliche Zustand wird vom Klassifikator geschätzt.

Je besser der Kaskadenklassifikator generalisiert bzw. schätzt, desto besser stimmen die Wahrscheinlichkeiten aus Abb. 2.3.3(b) mit den Wahrscheinlichkeiten für den Anwendungsfall aus Abb. 2.3.3(c) überein. Außerdem wird aus Abb. 2.3.3(b) ersichtlich, dass die Entscheidung des Kaskadenklassifikators von allen Stufenklassifikatoren abhängt, und die klassenbedingten A-Posteriori-Wahrscheinlichkeiten für die positiven Entscheidungen können somit durch eine *Markow-Kette* der Ordnung T dargestellt werden:

$$P(y(\underline{x}) = \omega_i | h_T(\underline{x}) = \omega_1, \ldots, h_1(\underline{x}) = \omega_1) \tag{2.3.9}$$

, wobei T die Anzahl der Stufenklassifikatoren ist. Die Ordnung T kann allerdings durch Anwendung von Gl. (2.3.4) für $j = 1$ schrittweise reduziert und in die lineare Kaskadenstruktur aufgelöst werden, wodurch ein iterativer Trainingsprozess entsteht, bei dem die A-priori-Wahrscheinlichkeiten durch die spezifischen Wahrscheinlichkeiten der Stufenklassifikatoren in jeder Stufe ermittelt werden und durch die Bayes'schen Entscheidungsfunktionen repräsentiert werden. Darüber hinaus wird ersichtlich, dass zur Schätzung der klassenbedingten Wahrscheinlichkeiten in der aktuellen Stufe nur diejenigen Beispiele $\mathcal{D}_t(\omega_j)$ verwendet werden sollten, die auch mit hoher Wahrscheinlichkeit in der Stufe auftreten, also alle Beispiele, die von den bestehenden Stufenklassifikatoren als positiv klassifiziert werden. Es hat sich jedoch gezeigt, dass bei einer strikten Auswahl von positiv klassifizierten Beispielen

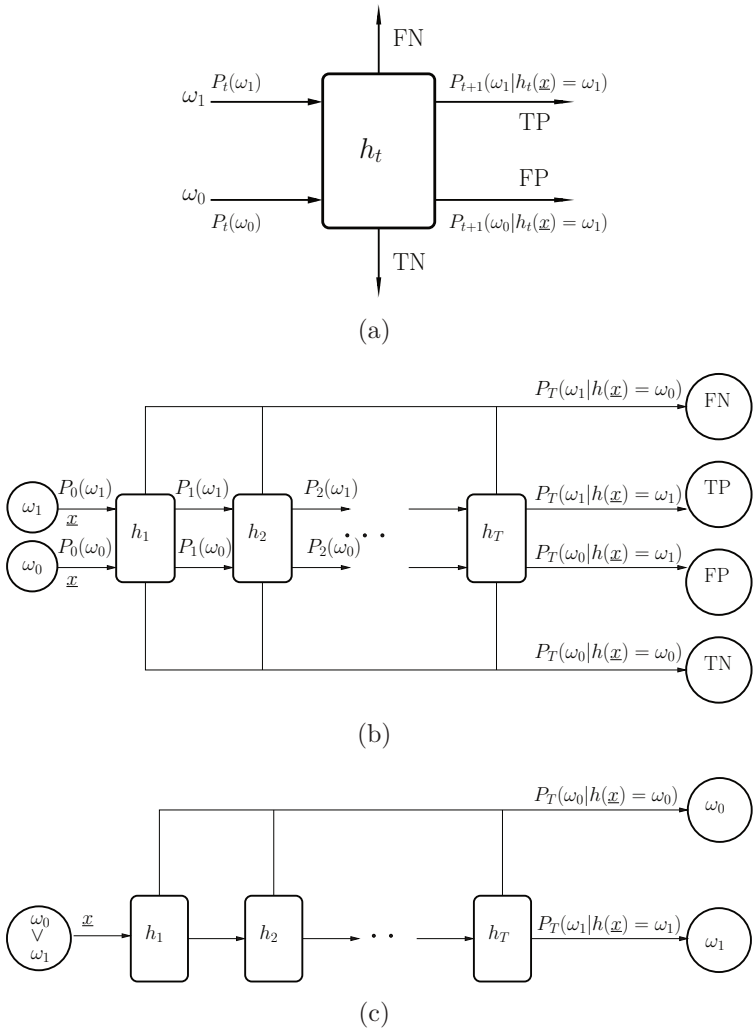


Abbildung 2.3.3:
(a): Veränderung der Wahrscheinlichkeiten durch einen Stufenklassifikator h_t.
(b): Wahrscheinlichkeitsgraph für den Kaskadenklassifikator für bekannte Klassenzugehörigkeiten. Durch die Weiterleitung für nur eine Entscheidung $h_t(\underline{x}) = \omega_1$ werden die A-posteriori-Wahrscheinlichkeiten zu A-priori-Wahrscheinlichkeiten.
(c): Entscheidungsgraph für den Anwendungsfall bei unbekannten Klassenzugehörigkeiten. Ein Beispiel $\underline{x}$ befindet sich in einem unbekannten Zustand. Der tatsächliche Zustand ω_1 oder ω_0 wird von den Klassifikatoren h_t geschätzt.

der Trainingsmenge die Klassifikationsleistung ab einer gewissen Anzahl an Stufen stark absinkt, da die geschätzten Verteilungen in den Stufen von den tatsächlichen Verteilungen auseinanderlaufen. Es tritt also eine Verzerrung oder ein Bias auf, sodass der Klassifikator nicht mehr erwartungstreu ist und der Klassifikationsfehler mit den Stufen ansteigt. In Kapitel 5 werden Methoden zur fusionierten Konfidenzwertberechnung der einzelnen Stufenklassifikatoren beschrieben und in Abschnitt 5.3 wird erklärt, wie eine konfidenzbasierte integrierte Fehlerminimierung erreicht werden kann, indem Beispiele selektiert werden, die mit hoher Wahrscheinlichkeit auftauchen:

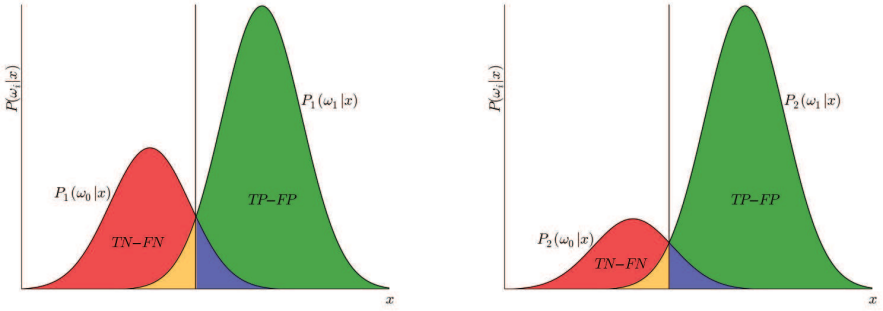
$$\mathcal{D}_t(\omega_j) = \{\underline{x} \in \mathcal{D}_0(\omega_j) \,|\, P(h_{1,\ldots,t-1}(\underline{x}) = \omega_1) \geq \rho\} \tag{2.3.10}$$

wobei ρ eine von der Fehlerkorrektur abhängige Konstante ist und $\mathcal{D}_t(\omega_j)$ die von den Stufenklassifikatoren gefilterten Beispiele sind, die zum Training des Stufenklassifikators h_t verwendet werden. Abb. 2.3.4(b) zeigt die Wirkung der Stufenklassifikatoren auf die A-priori-Wahrscheinlichkeiten. In diesem Beispiel halbiert sich das Verhältnis der A-priori-Wahrscheinlichkeiten, was zu einer immer kleiner werdenden Fehlerfläche führt. Man beachte, dass der Fehler nach der dritten Iteration nur noch den 8-ten Bruchteil des ursprünglichen Fehlers beträgt. Außerdem erkennt man die Verhältnisse der Wahrscheinlichkeiten aller Entscheidungszustände an den unterschiedlichen Flächen. Die Einzelklassifikationen induzieren somit eine Veränderung der klassenspezifischen Wahrscheinlichkeiten über die Zeit. Indem man Gl. (2.3.4) mit $j = 1$ rekursiv anwendet, erkennt man die Abhängigkeit von allen durch die Stufenklassifikatoren indizierten Wahrscheinlichkeiten:

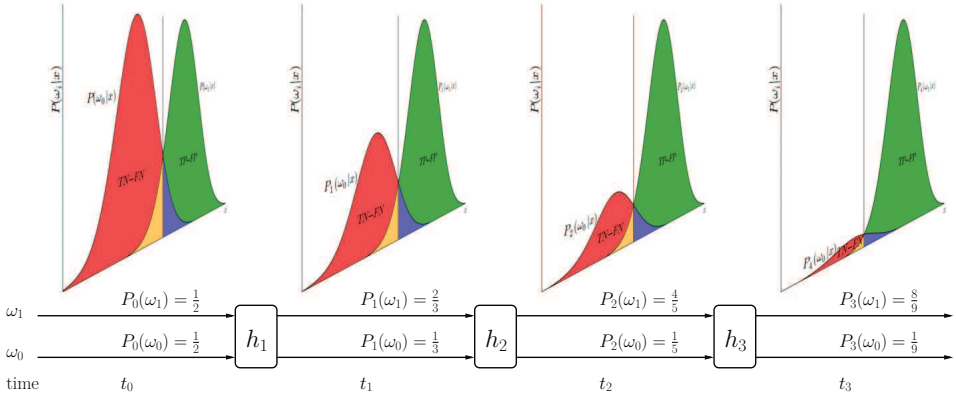
$$P_T(\omega_i) = P_0(\omega_i)\Pi_{t=1}^{T} P(h_t(\underline{x}) = \omega_1|\omega_i) \tag{2.3.11}$$

wobei i der Klassenindex ist und P_0 die initialen A-priori-Wahrscheinlichkeiten beschreiben. Alle Wahrscheinlichkeitsdichten der Vergangenheit haben somit auf die aktuelle A-piori-Wahrscheinlichkeit Einfluss. Die A-priori-Wahrscheinlichkeiten $P_T(\omega_i)$ geben direkt die Auftrittswahrscheinlichkeiten von Beispielen der Objektklasse und der Hintergrundklasse nach der Klassifikation an.

Das Optimierungskriterium für die Stufenklassifikatoren ist also die Minimierung der erwarteten Fehlentscheidungen bezüglich der klassenbedingten normierten Wahrscheinlichkeitsdichten, bei bekannten A-priori-Wahrscheinlichkeiten. Dies bedeutet für die positiven Ausgänge des Kaska-



(a)



(b)

Abbildung 2.3.4:
(a): Unterschiedliche A-posteriori-Wahrscheinlichkeiten verändern die Fehlerflächen der Verteilungen und damit auch die Bayes-Entscheidungsgrenze.
(b): Verkettungen der einzelnen Klassifikatoren führen zu einer Veränderung der A-priori-Wahrscheinlichkeiten an den Ausgängen der Klassifikatoren. Es resuliert eine durch Einzelklassifikationen induzierte zeitliche Entwicklung der Wahrscheinlichkeiten.

denklassifikators eine Maximierung der Auftrittswahrscheinlichkeit der positiven Klasse und eine Minimierung der Auftrittswahrscheinlichkeit der Hintergrundklasse.

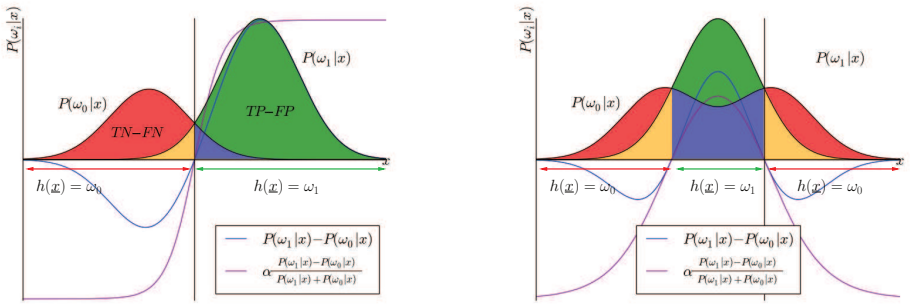
Die durch Gl. (2.3.11) herrührende Entwicklung der Fehlerwahrscheinlichkeiten soll nun anhand eines Beispiels kurz erläutert werden. Im Falle einer Kaskade mit $T = 100$ Stufenklassifikatoren und einer Sensitivität von $TPR_t = 0.95$ für jeden Stufenklassifikator resultiert eine Gesamtsensitivität von $TPR = 0.95^{100} \approx 6 \times 10^{-3}$, es resultiert also ein unbrauchbarer Klassifikator. Eine Falsch-Positiv-Rate jedes Stufenklassifikators von $TPR_t = 0.81$ ergibt hingegen eine erstaunlich niedrige Falsch-Positiv-Rate von $FPR = 0.81^{100} \approx 1 \times 10^{-9}$. Aus diesem Beispiel wird ersichtlich, wie wichtig die Anpassung der Stufenklassifikatoren und damit, wie wichtig die Trainings- und Klassifikationsalgorithmen sind um gute Klassifikationsleistungen für diese Art von Klassifikationssystemen zu erzielen. Zusätzlich kann der Fehler durch die in Kapitel 5, Abschnitt 5.3 beschriebene Methode reduziert werden.

2.3.3 Klassifikationskosten und Erhaltung der A-priori-Wahrscheinlichkeiten

Die Bayes'sche Entscheidungsregel, die zur Minimierung der Wahrscheinlichkeit eines Fehlers führt, lautet für zwei Klassen [35]:

$$h_s(\underline{x}) = \begin{cases} \omega_1 & \text{falls } g(\underline{x}) := P_s(\omega_1)P_s(\underline{x}|\omega_1) - P_s(\omega_0)P_s(\underline{x}|\omega_0) \geq 0 \\ \omega_0 & \text{sonst} \end{cases} \tag{2.3.12}$$

Abb. 2.3.4(b) zeigt, wie sich durch die veränderten A-priori-Wahrscheinlichkeiten auch die optimalen Bayes-Entscheidungsgrenzen ändern. Abb. 2.3.5 verdeutlicht den Unterschied zwischen einer schwellwertbasierten Entscheidungsfunktion und einer Bayesentscheidungsfunktion, also einer wahrscheinlichkeitsabhängigen Entscheidung, die bei multimodalen Verteilungen einen wesentlich geringeren Fehler erzeugt, da nicht zusammenhängende Entscheidungsregionen entstehen, was in Abb. 2.3.5(b) zu erkennen ist. Die blaue Funktion ist eine auf den Wahrscheinlichkeitsdichten berechnete Entscheidungsfunktion. In Magenta gefärbt ist eine normierte Entscheidungsfunktion, deren Wertebereich sich auf $[0, \ldots, 1]$ erstreckt und in der Abbildung entsprechend skaliert ist.



(a) (b)

Abbildung 2.3.5: Vergleich zwischen Bayes- und schwellwertbasierten Entscheidungsfunktionen:
(a) Im Falle zweier unimodaler Verteilungen sind die Entscheidungsregionen identisch.
(b) Für multimodale Verteilungen ergeben sich unterschiedliche Entscheidungsregionen. Bei der Bayes-Entscheidungsfunktion können nicht zusammenhängende klassenspezifische Entscheidungsregionen entstehen.

Beide Entscheidungsfunktionen erzeugen identische Entscheidungsbereiche, unterscheiden sich allerdings in ihrer Modulationshöhe und Steilheit. Für bekannte klassenspezifische Wahrscheinlichkeitsdichtefunktionen der Stufenklassifikatoren lässt sich die Bayesentscheidungsregel für jeden Stufenklassifikator und damit auf die Kaskadenstruktur anwenden, da die A-priori-Wahrscheinlichkeiten mit Gl. (2.3.4) bekannt sind [108]*. Indem die durch die Klassifikatoren veränderten A-priori-Wahrscheinlichkeiten in die Klassifikationsfunktionen einfließen, erfolgt eine *Erhaltung der initialen A-priori-Wahrscheinlichkeiten*. Somit kann bei der Optimierung die Menge der Gesamtbeispiele aufgeteilt werden und in den jeweiligen Stufen getrennt zur Optimierung der Stufenklassifikatoren verwendet werden. Durch eine Klassifikation der Beispiele aller vorhergehenden Stufen wird eine für die aktuelle Stufe repräsentative Menge an Beispielen erzeugt. Diese Auswahl ist jedoch ein wichtiges Kriterium und, wie oben erwähnt, eine potentielle Fehlerquelle. Deshalb wird in Kapitel 5 genau darauf eingegangen.

Bei der bildbasierten Objektdetektion ist die eigentliche Verteilung der Klassen und damit auch die A-priori-Wahrscheinlichkeiten jedoch höchst

asymmetrisch, da die Anzahl der Hintergrund-Objekte viel höher als die Anzahl der Objekte ist. Falls in einem Bild im Mittel für ein Objekt 10000 Hintergrundbeispiele auftreten, dann ist die geschätzte A-priori-Wahrscheinlichkeit für die Objektklasse $P(\omega_1) = 10^{-4}$ und für die Hintergrundklasse $P(\omega_0) = 1 - P(\omega_1)$. Sieht man die Wahrscheinlichkeiten als Gewichtungen an, so bekommen 10000 Positivbeispiele dasselbe Gewicht wie ein Negativbeispiel und damit werden auch die Fehler des Klassifikators, welche durch Positiventscheidungen entstehen, herunter gesetzt. In der Bayes'schen Entscheidungstheorie wird deshalb eine Verlustfunktion eingeführt, wobei für den Zweiklassenfall die A-posteriori-Wahrscheinlichkeiten mit entsprechenden Gewichtungsfaktoren λ skaliert werden. Um die Asymmetrie der Auftrittswahrscheinlichkeiten aufzuheben wird hier deshalb die A-priori-Wahrscheinlichkeit der zwei Klassen gleichgesetzt

$$\begin{aligned} P_1(\omega_1) &= \lambda(\omega_1)P(\omega_1) = 0.5 \\ P_0(\omega_0) &= \lambda(\omega_0)P(\omega_0) = 0.5 \end{aligned} \tag{2.3.13}$$

wobei $P(\omega_i)$ die tatsächlichen A-priori-Wahrscheinlichkeiten sind und $\lambda(\omega_i)$ dem entsprechende Skalierungsfaktoren. Somit besitzt ein Beispiel der Objektklasse dasselbe Gewicht wie ein Beispiel der Hintergrundklasse. Für einen Bayesklassifikator kann das Klassifikationsverhalten angepasst werden, indem die A-priori-Wahrscheinlichkeiten entsprechend skaliert werden. Die Modellierung der Wahrscheinlichkeitsdichten und die Bestimmung der Bayes-Entscheidung mit einem definierten Signifikanzniveau durch die Veränderung der A-priori-Wahrscheinlichkeiten wird in Kapitel 3, Abschnitt 3.5.1 beschrieben.

2.3.4 Konfidenzmaße kaskadierter Klassifikationssysteme

Eine *Konfidenz* soll die Sicherheit eines Klassifikators in seiner Entscheidung ausdrücken. Damit können mehrere Klassifikationsergebnisse derselben Klasse qualitativ verglichen werden und man erhält ein Maß, für das mit dieser Entscheidung verbundene Risiko. Da die in der Literatur vorhandenen Verfahren für Kaskadenklassifikatoren auf diskreten binären Entscheidungen

basieren, existieren lediglich Konfidenzmaße, die sich auf die Entscheidungsstufen beziehen. Klassenbedingte Wahrscheinlichkeiten eignen sich am besten für die Konfidenzwerte der Einzelklassen [101]. Deshalb basieren die hier vorgestellten Konfidenzmaße [108]* auf Wahrscheinlichkeiten und ermöglichen sowohl die Ermittelung der Konfidenzen der Stufenklassifikatoren, als auch eine Konfidenz des Kaskadenklassifikators, indem eine probabilistische Fusion der Einzelkonfidenzen erfolgt. In dieser Arbeit wird die Konfidenz in Bezug auf eine Zufallsvariable für den Zweiklassenfall als Distanz der erwarteten Wahrscheinlichkeiten berechnet [108]*:

$$\xi(h, \underline{x}) = |\langle P(\omega_1|\underline{x})\rangle_{\Delta\underline{x}} - \langle P(\omega_0|\underline{x})\rangle_{\Delta\underline{x}}| \tag{2.3.14}$$

, wobei sich der Erwartungswertoperator auf eine Umgebung $\Delta\underline{x}$ um den Wert $\underline{x}$ der Zufallsvariable bezieht. Der Erwartungswert bewirkt, dass auch die Wahrscheinlichkeiten in der Umgebung einfließen und erhöht somit die Stabilität. Die Modellierung der Wahrscheinlichkeiten und die Bildung des Erwartungswertes ist in Abschnitt 3.5.1 beschrieben.

Die Konfidenzberechnung stimmt mit der betragsmäßig lokal gemittelten Bayes-Diskriminanzfunktion aus Gl. (2.3.12) überein und ist unabhängig von der Entscheidung. Dies ist ein sinnvolles Maß, denn je größer die Differenz der klassenbedingten Wahrscheinlichkeiten ist, desto höher ist die Konfidenz in der Entscheidung des Klassifikators. Falls sich die Auftrittswahrscheinlichkeiten der Klassen nur gering unterscheiden, ist auch die Konfidenz klein. In Abb. 2.3.5 ist zu erkennen, dass für multimodale Verteilungen die Konfidenz an den Entscheidungsgrenzen minimal wird.

Bei der Betrachtung des Kaskadenklassifikators hat sich gezeigt, dass eine Klassifikation zu einem bestimmten Zeitpunkt die klassenspezifischen Wahrscheinlichkeiten verändert. Für bekannte zeitliche Klassifikationsfunktionen $h(t)$ und entsprechende zeitliche Wahrscheinlichkeitsdichten, lässt sich der Erwartungswert der Wahrscheinlichkeitsdichte über die Zeit berechnen:

$$\langle\, P(\omega_i \,|\, h(\underline{z}) = \omega_j)\,\rangle|_{t_0}^{t'} = \int_{t_0}^{t'} P(t, \omega_i \,|\, h(t, \underline{z}) = \omega_j)\mathrm{d}t \tag{2.3.15}$$

Für den Fall der diskreten Kaskadenklassifikation und bekannten Wahrscheinlichkeitsdichten an den Zeitpunkten $1 \leq t \leq T$ ist $h(t) \approx h_t$ der Stufenklassifikator in Stufe t des Kaskadenklassifikators h_C. Damit vereinfacht

sich Gl. (2.3.15):

$$\langle\, P(\omega_i \,|\, h_C(\underline{z}) = \omega_1) \,\rangle_{\Delta t} = \frac{1}{T} \sum_{t=1}^{T} |\langle\, P_t(\omega_i) P_t(h_t(\underline{x}_t, \underline{z}) = \omega_1 | \omega_i) \,\rangle_{\Delta \underline{x}_t}| \quad (2.3.16)$$

Somit lässt sich das Konfidenzmaß aus Gl. (2.3.14) durch Gl. (2.3.16) auch für die Kaskadenstruktur anwenden [108]*, wobei eine Fusion der Eigenschaften der Stufenklassifikatoren resultiert, indem eine zusätzliche zeitliche Mittelung verwendet wird und die Wahrscheinlichkeiten von den spezifischen Zufallsvariablen der Stufenklassifikatoren abhängen:

$$\xi(h_C, \underline{z}) = \frac{1}{T} \sum_{h_t(\underline{z})=\omega_1} \xi(h_t, \underline{x}_t(h_t, \underline{z})) \quad (2.3.17)$$

Wird das Beispiel verworfen, bevor es Stufe T erlangt, so werden die restlichen Konfidenzen implizit mit Nullen aufgefüllt.

2.3.5 Modellierung der Laufzeitkosten und erwarteter physikalischer Energiebedarf

Für die Anwendung von Klassifikatoren im Echtzeitbereich ist es wichtig, eine Abschätzung für die erwartete physikalische Leistung und damit bei bekannter Charakteristik des Rechenwerks für die erwartete Laufzeit eines Klassifikators zu erhalten. In der existierenden Literatur liegt jedoch das Hauptaugenmerk auf der Klassifikationsleistung. In dieser Arbeit wird deshalb im Folgenden ein Kostenmodell vorgestellt, das sowohl die Klassifikationsleistung als auch die physikalische Leistung beinhaltet und es damit ermöglicht unterschiedliche Klassifikatoren bezüglich dieser Merkmale zu bewerten [107]*.

Durch die Möglichkeit der Berechnung von Auftrittswahrscheinlichkeiten kann der erwartete Energieverbrauch des Klassifikators und damit auch die erwartete Laufzeit berechnet werden. Es existiert somit eine Möglichkeit, das Laufzeitoptimierungskriterium anhand der Eigenschaften eines Klassifikators zu beschreiben. Da die Laufzeit für eine Klassifikation von dem jeweiligen Rechenwerk abhängt, werden im Folgenden grundlegende Größen zur Beschreibung der Kriterien eingeführt.

Um ein Beispiel zu klassifizieren, d.h. zur Auswertung der Funktion $h(\underline{x}(\underline{z}))$, werden mehrere Berechnungsschritte durchgeführt. Hier sei nochmals verdeutlicht, dass der Bildausschnitt $\underline{z}$ im Allgemeinen auf den Wert $\underline{x}$ einer Zufallsvariable mit einer meist niedrigeren Dimension abgebildet wird, die dann über eine Klassifikationsfunktion ausgewertet wird, was mit einem gewissen Berechnungsaufwand verbunden ist. Diese Operation soll als Summe von Basisoperationen aufgefasst werden, indem eine *Basisoperation* als konstante atomare Energieeinheit definiert wird:

$$E_A = \int_t^{t+\Delta t} u(t)i(t)\mathrm{d}t \approx UI\Delta t = \text{const} \tag{2.3.18}$$

wobei $u(t)$ eine Spannungsfunktion und $i(t)$ eine Stromfunktion ist, die beide vom Rechenwerk abhängen. Über Gleichung 2.3.15 lässt sich der erwartete physikalische Energiebedarf für einen Entscheidungszustand angeben, indem jede Klassifikation mit der entsprechenden Energie gewichtet wird.

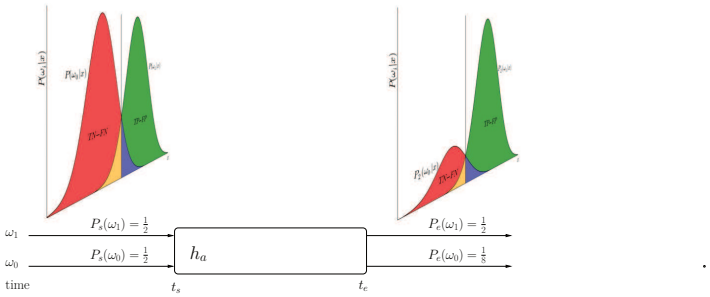
$$\langle\, C(\omega_i \,|\, h(\underline{z}) = \omega_j)\,\rangle|_{t_0}^{t'} = \int_{t_0}^{t'} u(t,h)i(t,h)P(t,\omega_i \,|\, h(t,\underline{z}) = \omega_j)\mathrm{d}t \tag{2.3.19}$$

Um wieder eine Diskretisierung vorzunehmen wird der Energiebedarf mittels Gl. (2.3.18) approximiert. Somit ist die approximierte Energie linear in der Zeit. Nun kann die Berechnungskomplexität eine Klassifikators anhand der Basisoperationen ausgedrückt werden:

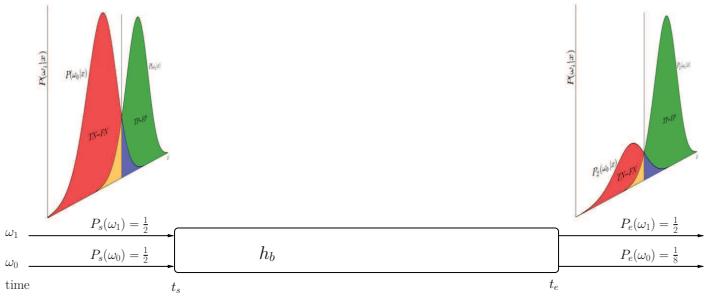
$$c(h) \approx oE_A \tag{2.3.20}$$

Der Faktor o kann durch eine gemittelte Laufzeitmessung des Klassifikators h bestimmt werden.

Abb. 2.3.6 zeigt das Laufzeitverhalten zweier Klassifikatoren h_a und h_b, indem die Anfangszeit t_s und die Endzeit t_e der Klassifikation pro Beispiel eingezeichnet sind. Die Klassifikatoren haben identische Fehlerwahrscheinlichkeiten und unterscheiden sich damit deutlich in ihren erwarteten Laufzeitkosten $C(\omega_0|h_a(\underline{z})) = \omega_1) < C(\omega_0|h_b(\underline{z}) = \omega_1)$. Physikalisch lässt sich das Maß $C(\omega_i|h(\underline{z}) = \omega_j)$ als die erwartete Energie interpretieren, die nötig ist um von den aktuellen Auftritts- und Fehlerwahrscheinlichkeiten, die durch



(a)



(b)

Abbildung 2.3.6:
Maß für die Effektivität eines Klassifikators. Beide Klassifikatoren h_a und h_b erzeugen identische Fehlerwahrscheinlichkeiten, wobei die Laufzeit des Klassifikators h_a in Abb. (a) deutlich kürzer als die Laufzeit des Klassifikators in Abb. (b) ist. Damit ist Klassifikator h_a zu bevorzugen.

die gegebene Verteilung bezüglich der Zufallsvariablen $\underline{z}$ gegeben ist, zu den Wahrscheinlichkeiten der Entscheidungszustände $P(\omega_i|h(\underline{x}) = \omega_i)$ bezüglich der Zufallsvariable $\underline{x}$ zu gelangen. Für den Kaskadenklassifikator addieren sich die Kosten der Stufenklassifikatoren [83]:

$$C(\omega_i|h_T(\underline{z}) = \omega_1) = C_0 + P_0(\omega_i)\, c(h_1) + \sum_{t=2}^{T} P(\omega_i|h_{t-1}(\underline{z}) = \omega_1)\, c(h_t) \quad (2.3.21)$$

wobei C_0 eine Konstante ist, die z.B. durch eine Initialisierung verursacht wird und $0 \leq i \leq 1$. Anhand dieser Formel wird ersichtlich, dass die Lauf-

zeit des Kaskadenklassifikators von den mit den Auftrittswahrscheinlichkeiten gewichteten Laufzeiten der Einzelklassifikatoren abhängt. Da die Auftrittswahrscheinlichkeit der Klassen zu Beginn am höchsten ist, wirkt sich die Berechnungskomplexität der ersten Stufenklassifikatoren im Allgemeinen am stärksten auf die Gesamtlaufzeit aus und sollten deshalb dementsprechend bei der Optimierung berücksichtigt werden. Bei diesem Kostenmodell wird allerdings vorausgesetzt, dass alle Kosten und Wahrscheinlichkeiten bekannt sind. Um die zu erwarteten Kosten für eine Menge an beliebigen Klassifikatoren abzuschätzen und damit ein Optimierungskriterium für eine globale Optimierung der Stufenklassifikatoren zu erzielen, wird die Funktion

$$C(\omega_i|h(\underline{z}) = \omega_j) = f(c(h), P(\omega_i|h(\underline{z}) = \omega_j)) \tag{2.3.22}$$

verwendet, wobei in einer Stufe der Kaskade der Wert für die Wahrscheinlichkeit der Falsch-Positiven, also $i = 0$ und $j = 1$ ausschlaggebend ist. Hierbei ist hervorzuheben, dass die Kosten durch die Auswertung der Entscheidungsfunktion zustande kommen, wobei im Allgemeinen eine Projektion auf eine niedrigdimensionale Zufallsvariable $\underline{x}(\underline{z})$ involviert ist. In Kapitel 4 wird ein neues Verfahren bezüglich dieses Kostenmodells vorgestellt und unterschiedliche Gewichtungsfunktionen f in Gl. (2.3.22) vorgestellt. Die Optimierung mit dieser Kostenfunktion ermöglicht die Verwendung von Merkmalen unterschiedlicher Berechnungskomplexität und damit können Stufenklassifikatoren ohne ein Boostingverfahren verwendet werden, was im Rahmen dieser Arbeit zur Echtzeitfähigkeit der Kaskadenklassifikatoren führte. Die Auswirkung der unterschiedlichen Gewichtungsfunktionen mit einem entsprechendnen Merkmalssatz wird durch eine Versuchsreihe in Abschnitt 4.8.3 gezeigt, indem die mittleren Laufzeitkosten zusätzlich zu den Klassifikationsleistungen dargestellt werden.

2.3.6 Statistisches Kaskadentraining durch lokale und globale Optimierungsstrategien

Bevor hier auf den prinzipiellen statistisch basierten Trainingsalgorithmus des Kaskadenklassifikators eingegangen wird, werden die zur Verfügung stehenden Merkmale beschrieben. Bei der Bildverarbeitung liegen Merkmale im Allgemeinen in einem diskreten Zustand vor, wie z.B. der Wert eines bestimmten Pixels. Für gewöhnlich existiert eine bestimmte Menge an Ba-

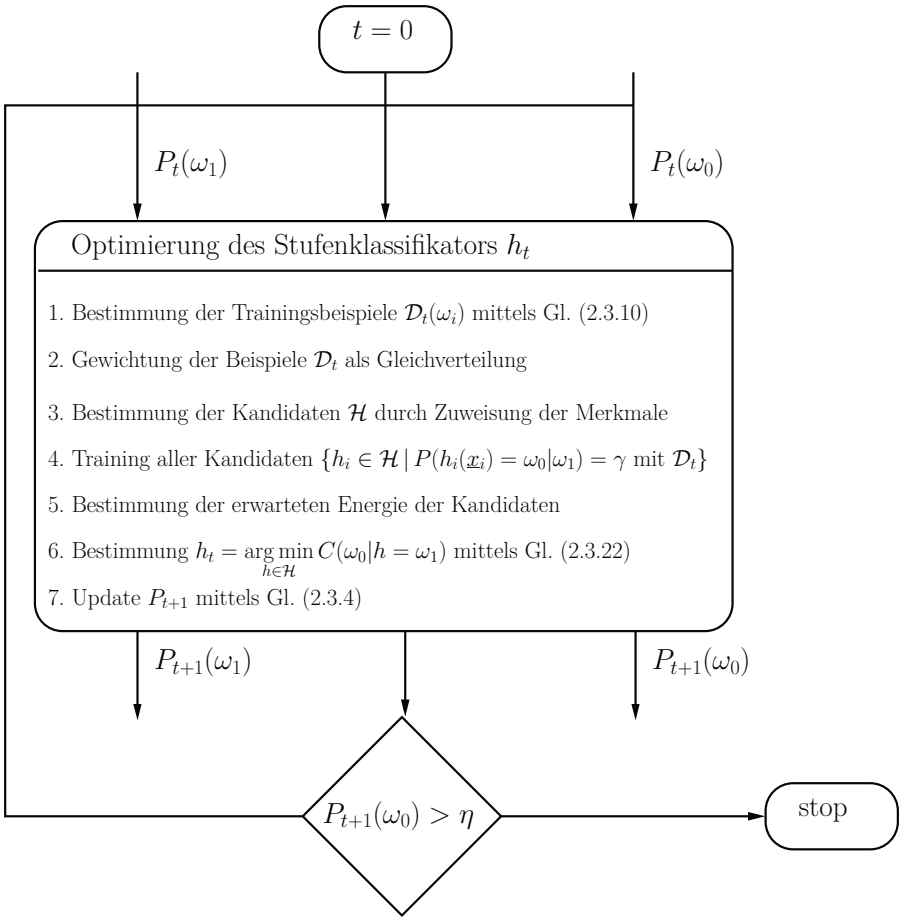


Abbildung 2.3.7: Prinzipielles Verfahren des statistischen Kaskadentrainings als iterativer Prozeß. In jeder Stufe wird der Stufenklassifikator h_t durch ein laufzeitsensitives Optimierungskriterium aus einer Menge an Kandidaten gewählt. Jeder Stufenkandidat wird auf eine vordefinierte Fehlerwahrscheinlichkeit γ skaliert. Die A-priori-Wahrscheinlichkeiten fließen in die stufenspezifischen Bayesentscheidungsfunktionen mit ein und werden für jede Stufe mitgeführt.

sismerkmalen, die als Zufallsvariable zur Verfügung stehen und sich auch gegenseitig kombinieren lassen. Seien M eine Menge an Basismerkmalen, und $\mathcal{X} = \mathcal{P}(M)$ die Potenzmenge aller Merkmale, so kann man jedes Merkmal einem Klassifikator zuordnen und erhält einen Klassifikatorpool

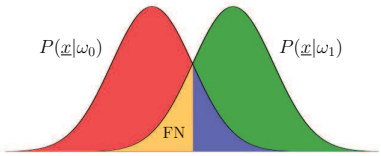
$\mathcal{H} = \{h_1, \ldots, h_K\}$ der Größe $K = 2^{|M|}$. Somit ist jedem Klassifikator ein Merkmal $\underline{x}_i(h_i)$ zugeordnet mit $1 \leq i \leq K$. Eine globale Optimierung des Kaskadenklassifikators würde bedeuten aus allen möglichen Kaskadenklassifikatoren, die aus dem bestehenden Pool gebildet werden können, die Kaskade mit minimalem Fehler zu wählen. Die Menge aller Kaskaden ist wiederum die Potenzmenge $\mathcal{P}(\mathcal{H})$ der Größe $2^{2^{|M|}}$. Für eine globale Optimierung mit $M = 10$ Basismerkmalen müßte man deshalb $|\mathcal{P}(\mathcal{H})| \approx 10^{300}$ Kaskadenklassifikatoren trainieren, was einen nicht realisierbaren Aufwand darstellt. Deshalb wird das Trainingsverfahren auf eine Greedy-Methode beschränkt, indem in jeder Stufe alle Kandidaten an Klassifikatoren $\mathcal{H}$ trainiert werden und die erwartete Energie jedes Klassifikators mittels einer gemittelten Laufzeitmessung bestimmt wird. Der Klassifikator mit dem besten Laufzeitverhältnis kann so durch Gl. (2.3.22) bestimmt werden. Es ist jedoch hervorzuheben, dass alle Kandidaten auf der repräsentativen Menge mittels Gl. (2.3.10) so trainiert werden, dass sie eine konstante Fehlerwahrscheinlichkeit γ auf den Beispielen $\mathcal{D}_t$ erzielen:

$$\{\, h \in \mathcal{H} \,|\, P(h(\underline{x}) = \omega_0 \,|\, y(\underline{x}) = \omega_1) = \gamma \,\} \tag{2.3.23}$$

Die Verteilungen werden also durch eine Korrektur-Wahrscheinlichkeit $P_c(\omega_0)$ so skaliert, dass die falsch klassifizierten positiven Beispiele genau das γ-Quantil aller positiven Beispiele bilden. Somit ist die Irrtumswahrscheinlichkeit einer Falsch-Negativ-Entscheidung auf den gleichverteilten Klassen das *Signifikanzniveau* $\gamma = P(h(\underline{x}) = \omega_0|\omega_1)$. Die Korrektur $P_c(\omega_0)$ fließt bei der Berechnung der neuen A-priori-Wahrscheinlichkeit mit ein. Unter der Annahme, dass in den positiven Beispielen auch Ausreisser vorkommen, wird somit ein vordefinierter Fehler geduldet, wobei die Sensitivität auf den Verteilungen dann $1 - \gamma$ beträgt. Ansonsten könnte bereits ein Stufenklassifikator die Gesamtsensitivität negativ beeinflussen. Der Skalierungsfaktor kann auf die A-priori-Wahrscheinlichkeiten verlagert werden, somit sind die klassenbedingten Wahrscheinlichkeiten weiterhin gleichverteilt und normiert. Das genaue Verfahren zur Skalierung der Wahrscheinlichkeiten auf definierte Quantile für multimodale Verteilungen wird in Kapitel 3, Abschnitt 3.5.1 erläutert. Damit ist die Optimierung des Stufenklassifikators ein globales Verfahren, das es ermöglicht aus einer Menge an Klassifikatoren mit unterschiedlichen Berechnungkomplexitäten bzw. physikalischem Energieverbrauch durch Gl. (2.3.22), den an das Problem in der jeweiligen Stufe angepassten Klassifika-

Training eines Kandidatenklassifikators h

1. Trainingsbeispiele $\mathcal{D}_t(\omega_i)$ sind als Werte von Z-Variable $\underline{Z}$ gegeben.
2. Gewichtung der Werte von $\underline{Z}$, sodass die Klassen gleichverteilt sind.
3. Bestimmung einer geeigneten Projektion $\underline{X} = pro(\underline{Z})$ mit $pro : \mathbb{R}^n \to \mathbb{R}^m$
4. Berechnung der Verteilungen für $\underline{X}$



5. Berechnung einer Korrektur $P_c(\omega_0)$, sodass FN das γ-fache von $P(\underline{x}|\omega_1)$
6. Bestimmung der Entscheidungsfunktion von h
7. Berechnung der Wahrscheinlichkeiten mittels Gl. (2.3.4) und $P_t(\omega_1) = P_t(\omega_0)$

Abbildung 2.3.8: Training der Kandidatenklassifikatoren in Schritt 4. aus Abb. 2.3.7. Der Falsch-Negativ-Fehler wird auf eine definierte Größe γ eingestellt. Somit bilden die falsch-klassifizierten positiven Beispiele das γ-Quantil aller positiven Beispiele.

tor mit entsprechender Berechnungskomplexität und Klassifikationsleistung zu wählen:

$$h_t = \arg\min_{h \in \mathcal{H}} C(\omega_0 \,|\, h = \omega_1) \tag{2.3.24}$$

Nachdem der Stufenklassifikator bestimmt ist, wird er dem Kaskadenklassifikator angehängt und die neuen A-priori-Wahrscheinlichkeiten werden mittels Gl. (2.3.4) bestimmt. Der Vorgang wird dann iterativ fortgesetzt, bis $P_{t+1}(\omega_0)$ eine kritische Schwelle überschreitet, d.h. bis die Auftrittswahrscheinlichkeit eines negativen Beispiels in einer Stufe zu hoch wird. Somit exisitiert ein Abbruchkriterium. Das Prinzip des Kaskadentrainings ist in Abb. 2.3.7 dargestellt.

Das Kaskadentraining als iterative Greedy-Strategie ist durchaus sinnvoll, da ein komplexer Klassifikator in den ersten Stufen die Gesamtlaufzeit negativ beeinflussen würde und es prinzipiell möglich ist, durch hinzufügen weiterer

Klassifikatoren die Klassifikationsleistung in einem gewissen Rahmen zu reduzieren. Allerdings ist damit nicht gewährleistet im globalen Optimum zu landen. Es hat sich jedoch gezeigt, dass diese Optimierungsmethode bereits eine große Laufzeitverbesserung zur Folge hat [108]*. Der Basiskaskadentrainingsalgorithmus ist in Abb. 2.3.7 dargestellt. Das Training der Kandidaten für eine Stufe wird in Abb. 2.3.8 gezeigt.

2.4 Überblick

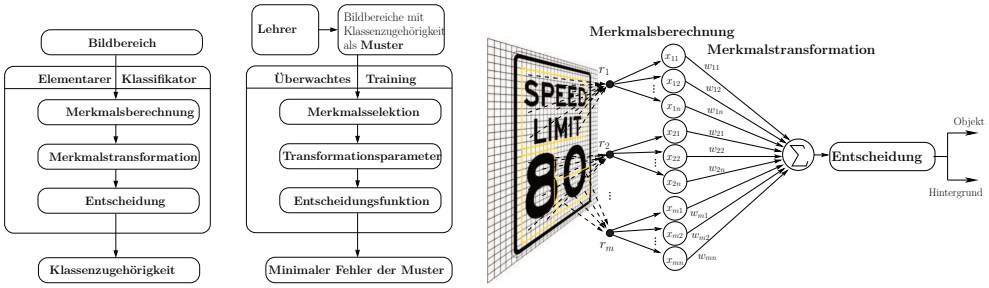
Bei bildbasierten maschinellen Lernverfahren liegen die Lernbeispiele für gewöhnlich als Bildausschnitte vor. Der erste Teil des Kapitels 3 beschreibt deshalb eine geeignete Darstellung von integralbildbasierten Merkmalen, sowie deren Extraktion. Danach wird eine Methode erklärt, die unterschiedliche Merkmale verbindet und damit eine geeignete Zufallsvariable erzeugt, welche dann für die Bayes'sche Betrachtung verwendet werden kann. Im Kapitel 4 werden Trainingsalgorithmen beschrieben, die abhängig von geeigneten Zufallsvariablen und Wahrscheinlichkeitsdichteverteilungen die beschriebenen Formeln realisieren. Des Weiteren wird eine Fusionsmethode vorgestellt, die es erlaubt die Klassifikationsresultate der Stufenklassifikationen zu einer Gesamtklassifikation für den Kaskadenklassifikator zu kombinieren. Außerdem wird ein Konfidenzmaß eingeführt, das ebenfalls aus den Einzelkonfidenzen der Stufenklassifikatoren berechnet wird. Da die Darstellung der Merkmale, die in Kapitel 3 eingeführt wurden, Bandpasscharakter aufzeigen, werden in Kapitel 4 auflösungsabhängige Trainings- und Detektionsalgorithmen beschrieben. Dort wird ebenfalls ein Maß zur Approximation der erwarteten operativen Kosten für einen Klassifikator eingeführt und Algorithmen zur kostenoptimierten Selektion von Klassifikatoren, basierend auf den Auftrittswahrscheinlichkeiten, vorgestellt. Kapitel 6 beschreibt unterschiedliche Klassifikationsysteme, die als Stufenklassifikatoren verwendet werden und deren Trainings- und Klassifikationsalgorithmen. Dort werden ebenfalls bereits eingeführte Klassifikatorarchitekturen geeignet kombiniert und hybride Klassifikatoren erzeugt. Im Kapitel 5 wird ein semiüberwachter Algorithmus zur bildbasierten Objektdetektion mittels Kaskadenklassifikatoren vorgestellt. Die spezielle Eigenschaft des semiüberwachten Klassifikators ist, dass der Klassifikator die Auswahl der Trainingsbeispiele bis zu einem gewissen Grad selbst vornehmen kann und sogar vom Menschen klassifizierte Beispiele umbenennen kann.

Kapitel 3

Merkmalsbasierte Klassifikation

In diesem Kapitel werden elementare Klassifikatoren, die aus Bildausschnitten extrahierte Merkmale zur Klassifikation verwenden, und deren überwachte Lernmethoden beschrieben. Die merkmalsbasierte Klassifikation bezieht sich im Folgenden auf bildbasierte Merkmale mit einer isolierten Klassifikationseinheit. Mit dieser Beschreibung soll der explizite Unterschied zur Klassifikation mit Ensemble-Verfahren verdeutlicht werden, die in Kapitel 6 beschrieben werden, wobei dort Merkmale eines Ensembleklassifikators aus der Ausgabe anderer Klassifikatoren herrühren können. Die grundlegende Idee, höherdimensionale Tensor- und Intensitätsmerkmale mit berechnungseffizienten Integralbildern für die Kaskadenklassifikation zu verwenden, wurde erstmals in der Publikation [109]* beschrieben. Im ersten Teil dieses Kapitels wird die Funktionsweise des Mehrkanal-Integralbilds und dessen Extraktion unterschiedlicher Merkmalstypen beschrieben. Danach wird die geometrische Merkmalsselektion erläutert, wobei aus unterschiedlich geformten rechteckigen Strukturen beliebige Merkmale extrahiert werden. Diese Bereiche werden *lokale rezeptive Felder* genannt. Sie wurden im Zusammenhang mit neuronalen Netzen unter anderem von Fukushima [46, 45, 47] und LeCun [30] beschrieben. Die Erweiterung auf raum-zeitliche Bildsequenzen vollzieht Wöhler [125]. Die lokalen rezeptiven Felder zeichnen sich dadurch aus, dass die Eingabeschicht der Merkmale eines neuronalen Netzes nicht vollständig mit der darauffolgenden Schicht vernetzt ist, wodurch eine berechnungseffiziente Darstellung der wichtigsten Merkmale gegeben ist. Zusätzlich zu den einzelnen rezeptiven Feldern, die hier durch integralbildbasierte Bereiche dargestellt werden, wird in diesem Kapitel eine kombinatorische Darstellung der Extraktionsregionen der Merkmale präsentiert. Danach wird auf unterschiedliche Normierungsmethoden und auf die Methoden zur Merkmalstransformation eingegangen. Die Entscheidungsverfahren für die Klassenzuweisung sind im Abschnitt Klassenzuordnung verfahrenstechnisch von den Verfahren zur Merkmalstransformation abgegrenzt. Hier wird beschrieben, wie die Wahrscheinlichkeitsdichten mit Frequenztabellen dargestellt werden können, und wie mittels der Wahrscheinlichkeitsdichtequotienten die Entscheidungsfunktionen auf ein vordefiniertes Signifikanzniveau skaliert werden können. Dieses Verfahren ist ebenfalls ein im Rahmen dieser Arbeit neu entwickeltes Verfahren, das eine enorme Steigerung der Detektionsleistung von Kaskadenklassifikatoren ermöglicht und durch seine lineare Komplexität auch sehr gut für den Einsatz mit der kombinatorischen Optimierung geeignet ist.

Abb. 3.0.1(a) zeigt den prinzipiellen Aufbau eines elementaren Klassifikators durch sequentielle Verarbeitungseinheiten für die Klassifikation eines



(a) (b) (c)

Abbildung 3.0.1:
(a): Prinzip der Klassifikation eines Bildbereichs mit einem Basisklassifikator durch sequentielle Verarbeitungseinheiten.
(b): Prinzip des überwachten Klassifikatortrainings zur Bestimmung der Merkmale, der Transformationsparameter und der Parameter der Entscheidungsfunktion mittels eines Fehlerminimierungsansatzes auf den Mustern mit bekannter Klassenzugehörigkeit.
(c): Beispiel eines linearen Klassifikators. Die gelben Rechtecke sind Merkmalsregionen, aus denen Merkmale extrahiert werden. Die extrahierten Werte werden mit Gewichten multipliziert und aufsummiert. Eine Entscheidungseinheit berechnet aus diesem Wert die zugehörige Klasse.

bestimmten Bildbereichs. Zuerst werden aus dem zu untersuchenden Bildbereich Merkmale berechnet, die dann mit einer Merkmalstransformation auf einen niedrigdimensionalen Unterraum abgebildet werden. Eine Entscheidungseinheit berechnet als Ergebnis die Klassenzugehörigkeit. Mittels eines Trainingsalgorithmus werden geeignete Merkmale, die Transformationsparameter, sowie die Parameter der Entscheidungsfunktion bestimmt. In Abb. 3.0.1(b) ist der Ablauf eines überwachten Trainingsalgorithmus mittels eines Fehlerminimierungsansatzes dargestellt. Die Klassenzugehörigkeit der Muster, die in das Trainingsverfahren einfließen, werden von einem Lehrer bestimmt. Ein Beispiel eines linearen Klassifikators ist in Abb. 3.0.1(c) dargestellt. Aus einem Bildbereich werden in bestimmten Regionen Merkmale berechnet und auf eine eindimensionale Variable projiziert, woraus dann über eine Entscheidungsfunktion die Klassenzugehörigkeit ermittelt wird. Dieses Kapitel beschäftigt sich in Abschnitt 3.1 mit unterschiedlichen Arten von Merkmalen sowie Verfahren zur effizienten Berechnung. In Abschnitt 3.1.2 werden geometrische und signaltheoretische Eigenschaften der Merkmale be-

züglich ihrer Form beschrieben sowie eine konsistente Gitterstruktur zur skalierungsunabhängigen Darstellung eingeführt. Diese Struktur erlaubt sowohl eine Suchraumeinschränkung beim Training als auch die Verwendung eines auflösungsabhängiges Abtastschemas zur Tiefensuche mit dem Kaskadenklassifikator. Im letzten Abschnitt 3.4 werden statistische Verfahren zur Merkmalstransformation beschrieben.

3.1 Merkmalsberechnung

Die Repräsentation eines Bildauschnitts bei bildverarbeitenden Systemen ist für gewöhnlich durch eine Matrix von Pixelwerten $\underline{Z}$ gegeben und wird statistisch als Zufallsvariable betrachtet. Ein Klassifikator verwendet Merkmale aus bestimmten Bereichen von $\underline{Z}$ um eine Klassifikation vorzunehmen. Um schnelle Klassifikationszeiten pro Bildausschnitt bei der Merkmalsextraktion zu erreichen, wird von Viola und Jones [116, 115] ein *Integralbild* verwendet, das ursprünglich in der Computergrafik als "Summed Area Table" bekannt ist. Die dort verwendeten Haar-ähnlichen Merkmale werden aus Differenzen benachbarter rechteckiger Bereiche berechnet und bilden eine überbestimmte Basis. In dieser Arbeit werden sowohl intensitätsbasierte, als auch strukturtensorbasierte Merkmale mit beliebigen Rechteckgeometrien durch Mehrkanalintegralbilder verwendet und ihre Vorteile im Folgenden beschrieben. Es können zusätzlich die haar-ähnlichen Merkmale verwendet werden [116, 115].

Das Verfahren wurde erstmals in [109]* vorgestellt. Das Integralbild beinhaltet an jedem Punkt (x, y) die Summe der Werte innerhalb des Rechtecks zwischen dem Ursprung $(0, 0)$ und dem Punkt (x, y) [116, 115]:

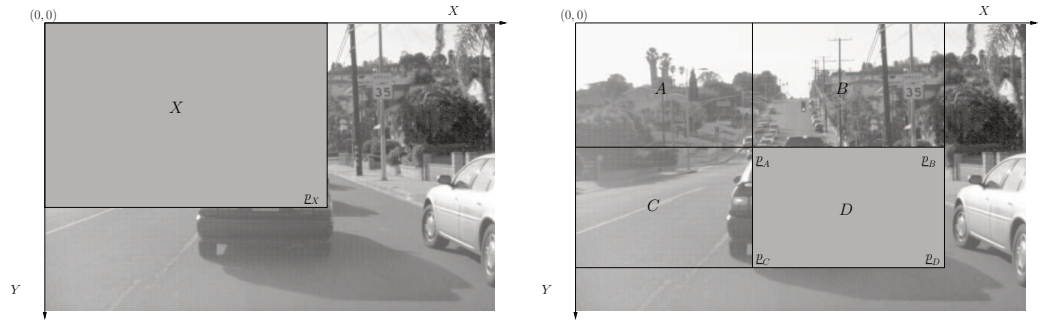
$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{3.1.1}$$

wobei i das Orginalbild und ii das Integralbild ist. Mit den rekursiven Gleichungen [116]:

$$s(x, y) = s(x, y - 1) + i(x, y) \tag{3.1.2}$$

$$ii(x, y) = ii(x - 1, y) + s(x, y) \tag{3.1.3}$$

(wobei $(s(x, -1) = 0$ und $ii(-1, y) = 0$ ist), kann das Integralbild in einem Durchgang linear berechnet werden. Mit dem Integralbild können die Summen zweidimensionaler Bildsignale innerhalb beliebiger Rechtecke mit



(a) (b)

Abbildung 3.1.1:
(a): Summe X der Intensitätswerte des Bildes i repräsentiert durch den Wert $ii(\underline{p}_X)$ des Integralbildes (Gl. (3.1.1)).
(b): Berechnung der grauen Fläche mit vier Integralbildzugriffen:
$ii(\underline{p}_D) + ii(\underline{p}_A) - ii(\underline{p}_B) - ii(\underline{p}_C)$.

oberer linker Ecke (x_1, y_1) und unterer rechter Ecke (x_2, y_2) mit vier Integralbildzugriffen berechnet werden, indem die Werte des Integralbilds an den Eckpunkten geeignet addiert werden:

$$\sum_{\substack{x_1 \le x' \le x_2 \\ y_1 \le y' \le y_2}} i(x', y') = ii(x_2, y_2) + ii(x_1, y_1) - ii(x_2, y_1) - ii(x_1, y_2) \qquad (3.1.4)$$

Abb. 3.1.1(a) zeigt die Repräsentation der Summe der Intensitätswerte innerhalb der Fläche X mit dem Wert $ii(\underline{p}_X)$ des Integralbildes. Die Berechnung der Fläche innerhalb eines Rechtecks ist in Abb. 3.1.1(b) dargestellt.

3.1.1 Merkmalstypen

Da sich das Integralbild für unterschiedliche Merkmalswerte berechnen lässt, liegt es nahe, weitere Merkmalstypen zu verwenden und in einem *Multikanal-Integralbild* mitzuführen, wobei jeder Pixelposition ein Vektor unterschiedlicher Merkmalswerte zugeordnet ist. Aus einem Intensitätsbild lässt sich die gemittelte Intensität innerhalb einer rechteckigen Region r durch das Integralbild mit Gl. (3.1.4) effektiv berechnen und als Faltungsoperation mit

einem Boxfilter ausdrücken:

$$\bar{i}(r) = \sum_{(x,y)\in r} w(x,y)i(x,y) = w_r * i \tag{3.1.5}$$

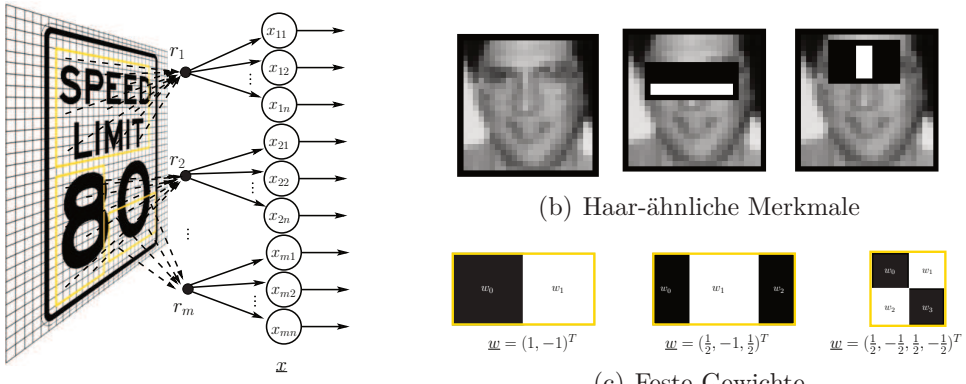
$$\overline{i^2}(r) = \sum_{(x,y)\in r} w(x,y)i(x,y)i(x,y) = w_r * i^2 \tag{3.1.6}$$

wobei die konstanten Filterkoeffizienten dem Kehrwert der Rechteckfläche entsprechen:

$$w(x,y) = \begin{cases} w_r = \frac{1}{(x_2-x_1+1)(y_2-y_1+1)} & \text{falls } (x,y) \in r(x_1,y_1,x_2,y_2) \\ 0 & \text{sonst} \end{cases} \tag{3.1.7}$$

und $\overline{i^2}(r)$ die Antwort des Boxfilters auf die quadrierten Intensitäten ist. Das zentrale Pixel dieser diskreten Faltung liegt im Schwerpunkt der Pixelpositionen. Die Werte entsprechen also dem arithmetischen Mittel der Intensitäten. Im Gegensatz zu den klassischen Boxfiltern können unterschiedliche Breiten und Höhen der Filter verwendet werden. Abb. 3.1.2(b) zeigt die Anwendung von Haar-ähnlichen Merkmalen bei der Anwendung der Gesichtserkennung [116, 115]. Die Pixelwerte der schwarzen Bereiche werden von den weißen Bereichen subtrahiert, woraus ein skalarer Merkmalswert resultiert. Es werden also feste Gewichte für die Bereiche verwendet, was in Abb. 3.2(c) gezeigt wird. Im Gegensatz dazu werden in dieser Arbeit mehrdimensionale Merkmale extrahiert (siehe Abb. 3.1.2(a)).

Um markante Punkte zu identifizieren werden in der Literatur sogenannte Corner Detection Verfahren [55, 103, 104, 84] verwendet, meistens in Kombination mit einem Matchingverfahren, um Punktkorrespondenzen in zwei ähnlichen Bildern zu ermitteln. Dabei wird nach Eckpunkten gesucht, indem für Ecken charakteristische Merkmale berechnet werden. Ein vielseitiger Ansatz zur Berechnung unterschiedlicher struktureller Eigenschaften ist durch den *Strukturtensor* [13, 57, 64, 65] gegeben, wobei Bigün [13] eine Berechnungsformel im Ortsraum für die Trägheitsmatrix der Energie im Fourierraum aufstellt und deren Äquivalenz beweist, sowie lineare Symmetrien anhand der Eigenwerte feststellen kann. Für den zweidimensionalen Anwendungsfall wird eine Formel zur Berechnung der Richtung der Hauptträgheitsachsen mit einer Trägheitsmatrix vorgestellt, sowie ein Sicherheitsmaß für die Korrektheit der Berechnung angegeben. Diese Trägheitsmatrix wird in den Arbeiten [57, 64] Strukturtensor genannt und in dieser Arbeit mit einer integralbildbasierten effizienten Methode zur mehrdimensionalen Merkmalsextraktion und



(a) Merkmalsvektor

(b) Haar-ähnliche Merkmale

(c) Feste Gewichte

Abbildung 3.1.2:
(a): Extraktion mehrerer Merkmalswerte x_{kj} aus einer Region r_k. Die Merkmale ergeben einen höherdimensionalen Merkmalsvektor $\underline{x}$.
(b): Haar-ähnliche Merkmale bei der Anwendung zur Gesichtserkennung [114].
(c): Feste Gewichte zur Differenzbildung eines einzelnen Merkmalwertes bei den Haar-ähnlichen Merkmalen.

Klassifikation verwendet [109]*. Bei den hier vorgestellten tensorbasierten Merkmalen ist die Idee, dass unterschiedliche charakteristische Tensorwerte als mehrdimensionale Merkmale innerhalb einer beliebigen Merkmalsregion r effizient extrahiert werden und von den Klassifikatoren mit einer geeigneten Merkmalstransformation belernt werden können. Da auch der Strukturtensor mit einem Integralbild berechnet werden kann [109]*, werden unterschiedliche strukturelle Eigenschaften durch einen mehrdimensionalen Merkmalsvektor in einer variablen Region effektiv berechnet. Der Strukturtensor lässt sich über einen "Sum of Squared Differences"-Ansatz (SSD) herleiten:

$$T(x, y) = \sum_u \sum_v w(u, v)(I(u, v) - I(u + x, v + y))^2 \tag{3.1.8}$$

wobei (u, v) die Positionen in einer Region um den Punkt (x, y) sind. Beinhaltet die Bildregion z. B. lediglich eine horizontale Linie, ergeben Verschiebungen in horizontaler Richtung keine quadrierten Differenzwerte. Bei isotropischen Strukturen ergeben sich Änderungen in allen Richtungen. Mit einer Taylor-Erweiterung und den Ableitungen $i_x = \frac{\partial i}{\partial x}$ und $i_y = \frac{\partial i}{\partial y}$ der Intensitä-

ten i,

$$i(u+x, v+y) \simeq i(u,v) + i_x(u,v)x + i_y(u,v)y \tag{3.1.9}$$

kann Gl. (3.1.8) wie folgt approximiert werden:

$$T(x,y) \simeq (x\ y) \begin{pmatrix} \sum_{u,v} w(u,v) i_x^2(u,v) & \sum_{u,v} w(u,v) i_x i_y(u,v) \\ \sum_{u,v} w(u,v) i_x i_y(u,v) & \sum_{u,v} w(u,v) i_y^2(u,v) \end{pmatrix} (x\ y)^T \tag{3.1.10}$$

wobei die 2×2-Matrix dem symmetrischen Strukturtensor entspricht. Das Integralbild wird also mit den Ableitungen i_x und i_y berechnet und beinhaltet die drei Kanäle i_x^2, i_y^2 und $i_x i_y$. Somit können die Strukturtensormerkmale analog zu den Gleichungen (3.1.5) und (3.1.6) mit variabel geformten rechteckigen Boxfiltern in konstanter Zeit berechnet und als Faltung dargestellt werden:

$$\overline{i_x^2}(r) = \sum_{(x,y)\in r} w(x,y) i_x^2(x,y) = w_r * i_x^2 \tag{3.1.11}$$

$$\overline{i_y^2}(r) = \sum_{(x,y)\in r} w(x,y) i_y^2(x,y) = w_r * i_y^2 \tag{3.1.12}$$

$$\overline{i_{xy}}(r) = \sum_{(x,y)\in r} w(x,y) i_x i_y(x,y) = w_r * i_x i_y \tag{3.1.13}$$

Basierend auf diesen Tensorwerten können weitere Merkmale berechnet werden, die unterschiedliche strukturelle Eigenschaften repräsentieren. Die Orientierung der Hauptträgheitsachsen der Grauwertverteilung und deren Winkel innerhalb der Region r kann mit dem Orientierungsoperator [13, 57, 64] berechnet werden:

$$o_x = \overline{i_y^2}(r) - \overline{i_x^2}(r) \tag{3.1.14}$$

$$o_y = 2\overline{i_{xy}}(r) \tag{3.1.15}$$

$$t_\alpha = \frac{1}{2} \tan^{-1}(o_y, o_x) \tag{3.1.16}$$

Weitere Eigenschaften werden durch die Eigenwerte λ_1 und λ_2 repräsentiert:

$$\lambda_{1,2} = \frac{1}{2}(\overline{i_x^2} + \overline{i_y^2} \pm \sqrt{(\overline{i_y^2} - \overline{i_x^2})^2 + 4\overline{i_{xy}}^2}) \tag{3.1.17}$$

Es kann ebenfalls ein Merkmal für die Gestrecktheit der Grauwerteverteilungen berechnet werden [13, 22] :

$$t_e = \frac{(\lambda_2 - \lambda_1)^2}{(\lambda_1 + \lambda_2)^2} = \frac{(\overline{i_y^2}(r) - \overline{i_x^2}(r))^2 + 4\overline{i_{xy}}(r)^2}{(\overline{i_x^2}(r) + \overline{i_y^2}(r))^2} \tag{3.1.18}$$

(a) Großer Bereich (b) Mehrere kleine Bereiche

Abbildung 3.1.3:
(a),(b): Visualisierung der Eigenschaften des Strukturtensors. Die Länge der Haupt- und Nebenachsen werden mit den Eigenwerten $\lambda_{1,2}$ (Gl. (3.1.17)) berechnet. Die Orientierung ist durch t_α (Gl. (3.1.15)) gegeben. Die Exzentrizität t_e ist durch die Länge der blauen Linie dargestellt. Hintergrundbild aus [120].

Mit dem Kohärenzmaß [57] können Bereiche mit konstanter Grauwertverteilung von Bereichen mit einer isotropen Grauwertstruktur ohne Vorzugsrichtung unterschieden werden:

$$t_c = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \tag{3.1.19}$$

Abb. 3.1.3 zeigt eine Visualisierung der unterschiedlichen Tensormerkmale. Die Länge der Haupt- und Nebenachsen werden mit den Eigenwerten $\lambda_{1,2}$ (Gl. (3.1.17)) berechnet. Die Orientierung ist durch t_α (Gl. (3.1.15)) gegeben. Die Exzentrizität t_e ist durch die Länge der blauen Linie dargestellt. Mit den Mehrkanalintegralbildern lassen sich mit fünf Kanälen $(i, i^2, i_x^2, i_y^2, i_x i_y)^T$ Merkmalsvektoren aus einer Region r extrahieren, die sowohl intensitäts- als auch tensorbasierte Merkmale enthalten. Somit können mit geringem Rechenaufwand höherdimensionale, und damit Merkmale mit besserer Unterscheidungsfähigkeit, berechnet werden. Das Prinzip der Extraktion von höherdimensionalen Merkmalsvektoren aus unteschiedlichen Regionen r ist in Abb. 3.1.2(a) dargestellt. Da die Entscheidungsgrenzen auch mit höherdimensionalen Merkmalsvektoren bei den meisten Realweltproblemen nichtlinear sind, werden in der Literatur die einzelnen Merkmale oftmals polynomiell erwei-

Tensormerkmale		
Id	**Berechnung**	**Beschreibung**
t_1	$\overline{i_x^2}$	Gl. (3.1.11)
t_2	$\overline{i_y^2}$	Gl. (3.1.12)
t_3	$\overline{i_{xy}}$	Gl. (3.1.13)
t_4	o_x	Orientierung X, Gl. (3.1.14)
t_5	o_y	Orientierung Y, Gl. (3.1.15)
t_6	o_x^2	Quadr. Erweiterung
t_7	o_y^2	Quadr. Erweiterung
t_8	$o_x o_y$	Quadr. Erweiterung
t_9	t_e	Excentrizität, Gl. (3.1.18)
t_{10}	t_c	Kohärenz, Gl. (3.1.19)
t_{11}	$\overline{i_x^2} + \overline{i_y^2}$	Spur des Tensors
t_{12}	$\sqrt{\overline{i_x^2}}$	Quadr. Erweiterung
t_{13}	$\sqrt{\overline{i_y^2}}$	Quadr. Erweiterung
t_{15}	t_α	Winkel, Gl. (3.1.16)
t_{16}	$(\overline{i_x^2} - \overline{i_y^2})^2 - 4\overline{i_{xy}}^2$	Invar.
t_{17}	$\overline{i_x}$	
t_{18}	$\overline{i_y}$	
t_{19}	$\sqrt{\overline{i_{xy}}}$	

(a)

Intensitätsmerkmale		
Id	**Berechnung**	**Beschreibung**
l_1	$\overline{i}$	Gl. (3.1.5)
l_2	$\overline{i^2}$	Gl. (3.1.6)
l_3	$\overline{i^2} - (\overline{i})^2$	Varianz
l_4	$\sqrt{\overline{i^2} - (\overline{i})^2}$	Standardabweichung

(b)

Kombinierte Tensormerkmale	
Id	**Merkmalsvektor**
$\underline{k}_1$	$(o_x, o_y)^T$
$\underline{k}_2$	$(o_x, o_y, o_x^2, o_y^2, o_x o_y)^T$
$\underline{k}_3$	$(\overline{i_x^2}, \overline{i_y^2}, \overline{i_{xy}})^T$
$\underline{k}_4$	$(\overline{i_x^2}, \overline{i_y^2}, \overline{i_{xy}}, \sqrt{\overline{i_x^2}}, \sqrt{\overline{i_y^2}}, \sqrt{\overline{i_{xy}}})^T$
$\underline{k}_5$	$(t_\alpha, t_e, t_c, o_x, o_y, o_x^2, o_y^2, o_x o_y, \sqrt{\overline{i_x^2}}, \sqrt{\overline{i_y^2}})^T$

(c)

Kombinierte Intensitätsmerkmale	
Id	**Merkmalsvektor**
$\underline{n}_1$	$(\overline{i}, \overline{i^2} - (\overline{i})^2)^T$
$\underline{n}_2$	$(\overline{i}, \overline{i^2}, \overline{i^2} - (\overline{i})^2)^T$
$\underline{n}_3$	$(\overline{i}, \overline{i^2}, \overline{i^2} - (\overline{i})^2, \sqrt{\overline{i^2} - (\overline{i})^2})^T$

(d)

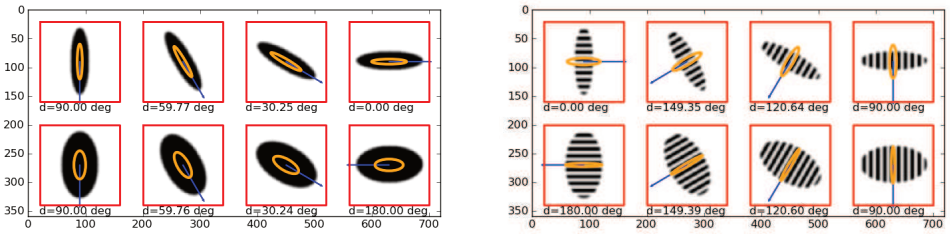
Kombinierte Tensor- und Intensitätsmerkmale	
Id	**Merkmalsvektor**
$\underline{c}_1$	$(\overline{i}, \overline{i_x^2} + \overline{i_y^2})^T$
$\underline{c}_2$	$(\overline{i}, \overline{i^2}, o_x, o_y)^T$
$\underline{c}_3$	$(\overline{i}, \sqrt{\overline{i^2} - \overline{i}^2}, o_x, o_y, \overline{i_x^2} + \overline{i_y^2}, t_e)^T$
$\underline{c}_4$	$(\overline{i}, \overline{i^2}, \overline{i_x^2}, \overline{i_y^2}, o_x, o_y, \overline{i_x^2} + \overline{i_y^2}, t_e)^T$

(e)

Abbildung 3.1.4: Liste der aus einer Region r berechneten Merkmale.
(a): Skalare Tensormerkmale
(b): Skalare Intensitätsmerkmale
(c): Kombinierte Tensormerkmale
(d): Kombinierte Intensitätsmerkmale
(e): Kombinierte Tensor- und Intensitätsmerkmale

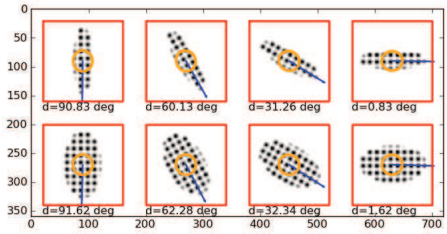
tert, indem sie miteinander multipliziert werden. Diese Erweiterung kann jedoch schnell zur Überadaption der Klassifikatoren mit einer schlechten Generalisierungsfähigkeit der Testdaten führen. Außerdem konnte im Versuch festgestellt werden, dass nur spezifische Merkmalskombinationen zu effektiven Merkmalen mit guter Trennfähigkeit führen.

In Abb. 3.1.4 sind unter (a) die skalaren tensorbasierten Merkmale und unter (b) die skalaren intensitätsbasierten Merkmale gelistet. Einige der Merkmale wurden quadratisch erweitert. Die vektoriellen Tensor- und Intensitätsmerkmale sind in Abb. 3.1.4(c) und 3.1.4(d) dargestellt und die kombinierten vektorwertigen Merkmale sind in Abb. 3.1.4(e) gelistet.



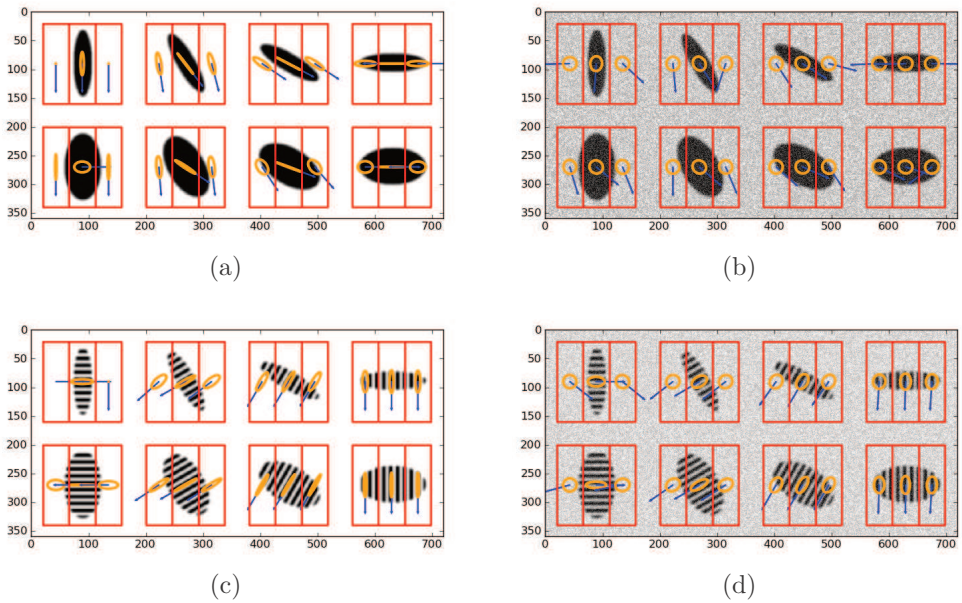
(a)

(b)



(c)

Abbildung 3.1.5:
(a): Visualisierung der exakten Bestimmung des Richtungswinkels der Grauwertverteilung um weniger als 0.5 Grad durch den blauen Richtungspfeil mit Gl. (3.1.16). Die Exzentrizität ist durch den gelben Kreis dargestellt (Gl. (3.1.18)).
(b): Das gestreifte Muster dominiert die Kanten. Dadurch ist der Richtungswinkel entlang des Streifenmusters ausgerichtet und die Exzentrizität hoch.
(c): Bei einem Schachbrettmuster stimmt der Richtungswinkel mit der Richtung der Ellipse überein, jedoch dominieren die Kanten des Musters, was zu geringer Exzentrizität führt (gelber Kreis).



(a) (b)

(c) (d)

Abbildung 3.1.6: Auswirkung der Merkmalsberechnung in unterschiedlichen rechteckigen Merkmalsfenstern. Es werden drei rechteckige Merkmalsfenster verwendet. Bei Abb. 3.6(b) und 3.6(d) wurden die Strukturen mit einem Rauschsignal überlagert.

Abb. 3.1.5 zeigt die Visualisierung der Merkmale, die innerhalb des roten Merkmalfensters berechnet werden, das die komplette Struktur umschließt. Der blaue Pfeil visualisiert den Richtungswinkel der Grauwertverteilung, der in Abb. 3.1.5(a) und 3.1.5(c) den Ellipsen entlang, zeigt und in Abb. 3.1.5(b) orthogonal zu den Ellipsen, zeigt, da die Kantenübergänge durch die Streifen dominieren. Die gelben Ellipsen stellen die Gestrecktheit bzw. Exzentrizität der Grauwertstruktur dar.

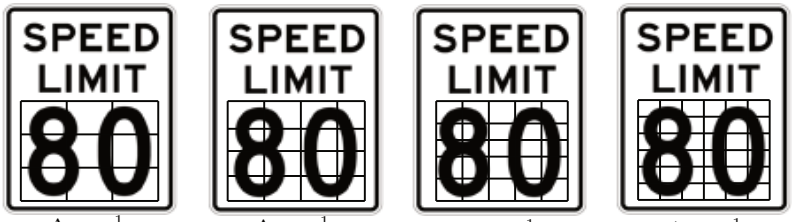
Abb. 3.1.6 zeigt die Auswirkung der Merkmalsberechnung mit mehreren Merkmalsfenstern. Die Fenster können eine beliebige rechteckige Struktur annehmen. Die Strukturen sind die bereits gezeigten Ellipsen. In den Abb. 3.1.6(b) und 3.1.6(d) wurden die Strukturen mit einem Rauschsignal überlagert.

3.1.2 Geometrische Merkmalsrepräsentation und -selektion

Da die geometrische Ausprägung der Bereiche, die als Merkmale verwendet werden, spezifische Filtereigenschaften, welche sich auf den Frequenzbereich auswirken, besitzt, wird in dieser Arbeit im Gegensatz zu gängigen Verfahren [116, 115] eine Gitterstruktur verwendet, woraus mehrere Vorteile resultieren. Zum einen kann für die Standardabtastung [65] die Abtastweite bei der Sliding-Window-Methode für beliebig skalierte Detektionsfenster genau angegeben werden. Dadurch kann durch das Verfahren sichergestellt werden, dass keine Details "übersehen" werden und die Anzahl der zu überprüfenden Bereiche möglichst klein bleibt, was wiederum einen Performancegewinn bedeutet. Zum Anderen wird eine hierarchische Einteilung der Merkmale von groben zu feinen Gitterstrukturen mit einem neuen Detektionsverfahren für Kaskadenklassifikatoren eingeführt, sodass der Suchraum aller möglichen Merkmalen beim Training stark eingeschränkt wird. Sobald eine Sättigung der Klassifikationsperformance durch hohe Auftrittswahrscheinlichkeiten der Hintergrundklasse eintritt, werden feinere Merkmale beim Training verwendet. Bei der Ausführung des Klassifikators wird eine hierarchische, rekursive Tiefensuche verwendet, und bei einer hohen Auftrittswahrscheinlichkeit für ein Objekt wird ausgehend von der aktuellen Region eine Feinsuche, abhängig von den Auflösungen der Merkmale des entsprechenden Klassifikators, durchgeführt. Dies ermöglicht schnelle Laufzeiten bei der Ausführung des Kaskadenklassifikators. Das Verfahren wurde erstmals in der Arbeit [107]* vorgestellt. Die Größe des Gitters wird durch eine feste Anzahl von Unterteilungen in X-, sowie in Y-Richtung, des lokalen Detektionsfensters für jede Auflösungsstufe erreicht, wobei sich die Anzahl der Unterteilungen mit der Anzahl der Stufen erhöht und unterschiedliche Unterteilungen in den beiden Richtungen möglich sind.

Im Gegensatz zu den klassischen Bildpyramiden, wie z.B. der Gauß-Laplace-Pyramide, bei denen eine Bandpasszerlegung des Bildes durch sukzessives Halbieren und Subtrahieren erreicht wird, kann mit dieser Methode eine feinere Gittereinteilung und damit zugleich eine feinere Frequenzaufspaltung erfolgen. Prinzipiell lassen sich mit einem Detektionsfenster mit minimaler Breite w bzw. Höhe h maximal so viele Unterteilungen, und damit Auflösungsstufen, bilden, wie die Anzahl der restfreien Teiler.

In Abb. 3.1.7 ist das Gitter des Detektionsfensters mit steigender Anzahl an Unterteilungen dargestellt, woraus für die Merkmale durch die Boxfil-



$\Delta r = \frac{1}{3}$ $\Delta r = \frac{1}{4}$ $\Delta r = \frac{1}{5}$ $\Delta r = \frac{1}{6}$

Abbildung 3.1.7: Gitterstruktur des Detektionsfensters mit steigender Anzahl an Unterteilungen und damit unterschiedlichen Auflösungsstufen mit steigender Auflösung. In jeder Auflösungsstufe werden die Merkmale aus möglichen Rechtecken, die am Gitter ausgerichtet sind gebildet.

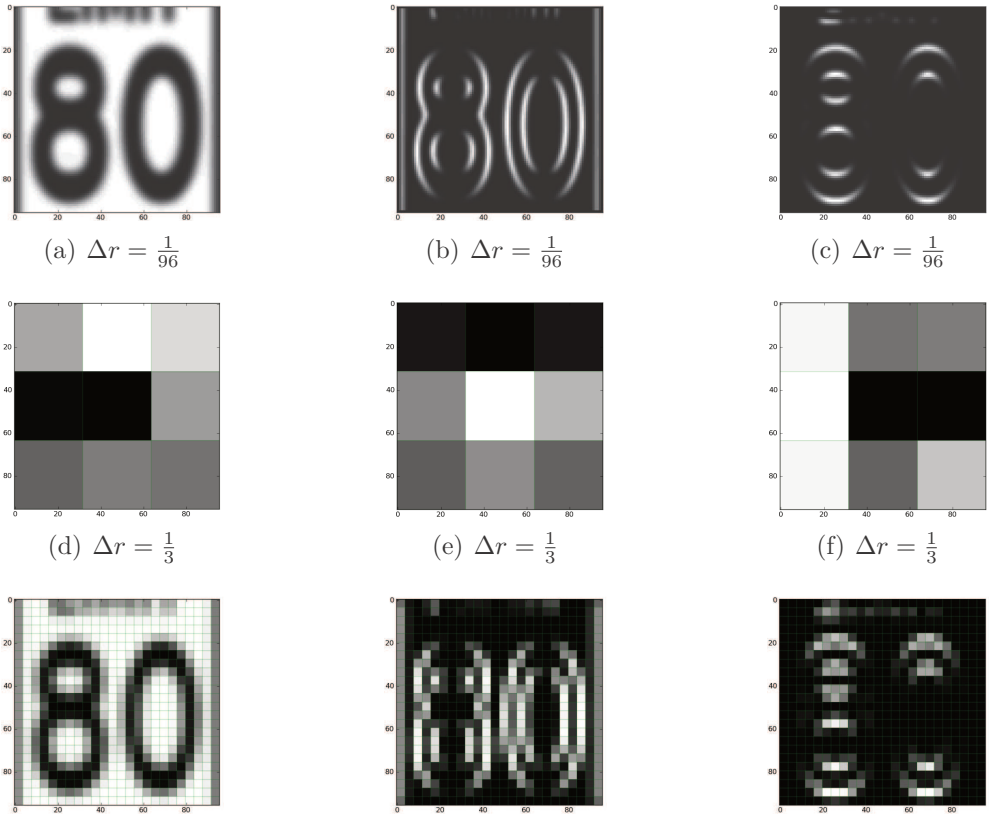
terung eine steigende Auflösung resultiert, denn die Merkmale werden aus möglichen Rechtecken, die am Grid ausgerichtet sind, gebildet. Es hat sich herausgestellt, dass bei den US-Verkehrszeichen die relativ großen Strukturen der Ziffern die beste Unterscheidungsfähigkeit in den Merkmalen liefern, wohingegen die rechteckigen Ränder der Schilder nur sehr schmal sind und bei ungünstigen Beleuchtungsverhältnissen unterbrochen werden oder verschwinden. Deshalb werden für das Training nur die Ziffernpaare mit einem äußeren Rand verwendet.

In Abb. 3.1.8 ist die Boxfilterung mit unterschiedlichen Gittereinteilungen, und damit unterschiedlichen Auflösungen, für drei unterschiedliche Merkmale dargestellt. Die Mittelwertbildung wird, wie beschrieben, durch die Integralbilder effizient erreicht. In der ersten Spalte (Abb. 3.1.8(a), 3.1.8(d), 3.1.8(g)) ist die Boxfilterung in den unterschiedlichen Auflösungen für das Intensitätsbild dargestellt (Gl.(3.1.5)). In der zweiten Spalte (Abb. 3.1.8(c), 3.1.8(f), 3.1.8(h)) ist die Filterung mit den quadrierten X-Gradienten des Strukturtensors (Gl. (3.1.11)) und in der dritten Spalte (Abb. 3.1.8(c), 3.1.8(f), 3.1.8(i)) die Filterung mit den quadrierten Y-Gradienten des Strukturtensors (Gl. (3.1.12)) dargestellt. Die *relative Auflösung* Δr wird durch die Anzahl der Unterteilungen beschrieben:

$$\Delta r_x = \frac{1}{x_d} \tag{3.1.20}$$

$$\Delta r_y = \frac{1}{y_d} \tag{3.1.21}$$

wobei x_d die Unterteilungen in X-Achsenrichtung und y_d die Unterteilungen



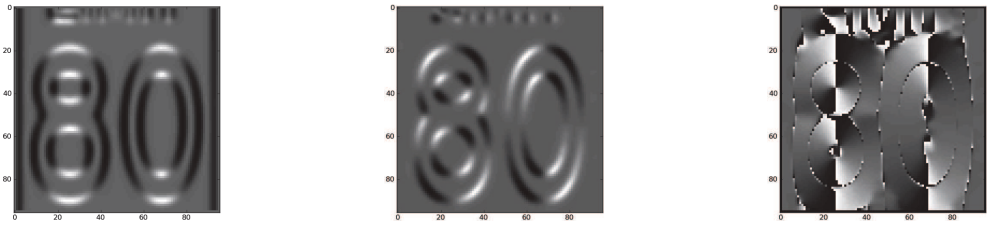
(a) $\Delta r = \frac{1}{96}$ (b) $\Delta r = \frac{1}{96}$ (c) $\Delta r = \frac{1}{96}$

(d) $\Delta r = \frac{1}{3}$ (e) $\Delta r = \frac{1}{3}$ (f) $\Delta r = \frac{1}{3}$

(g) $\Delta r = \frac{1}{24}$ (h) $\Delta r = \frac{1}{24}$ (i) $\Delta r = \frac{1}{24}$

Abbildung 3.1.8: Merkmalsberechnung durch Boxfilterung mit unterschiedlichen Gridgrößen durch unterschiedliche Unterteilungen.
(a), (d), (g): Boxfilterung der Intensitäten: $w_r * i$, (Gl.(3.1.5))
(c), (f), (h): Boxfilterung der quad. X-Gradienten: $w_r * i_x^2$, (Gl. (3.1.11))
(c), (f), (i): Boxfilterung der quad. Y-Gradienten: $w_r * i_y^2$ (Gl. (3.1.12))

in Y-Achsenrichtung sind. Die erste Zeile zeigt die Bilder in voller Auflösung (Abb. 3.1.8(a), 3.1.8(c), 3.1.8(c)). Die zweite Zeile zeigt die typische erste Auflösungsstufe mit drei Unterteilungen (Abb. 3.1.8(d), 3.1.8(f), 3.1.8(f)). In der letzten Zeile wird eine mittlere Auflösung von $x_d = y_d = 24$ verwendet (Abb. 3.1.8(g), 3.1.8(h), 3.1.8(i)). An diesen Bildern ist deutlich erkennbar, dass bereits bei der geringsten Auflösung mit $x_d = y_d = 3$ bei den Tensormerkmalen durch die ausgeprägten horizontalen Übergänge, erkennbar durch



(a) (b) (c)

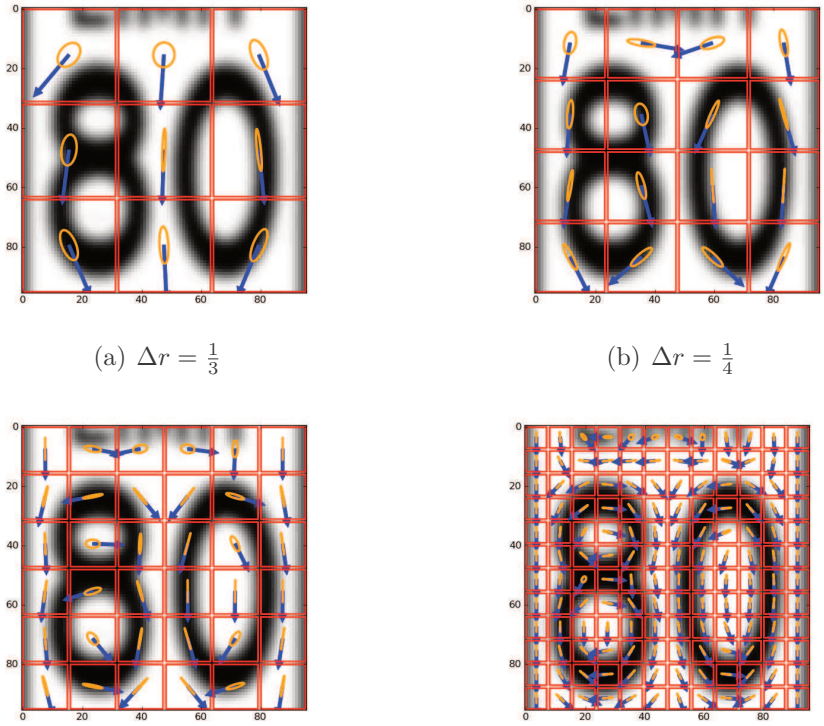
Abbildung 3.1.9: Orientierungs- und Winkelmerkmale bei voller Auflösung bei einem 80 - Speedlimit.
(a): o_x (Gl. (3.1.14))
(b): o_y (Gl. (3.1.15))
(c): t_α (Gl. (3.1.16))

die hellen Bereiche, starke Signalwerte für die quadrierten X-Gradienten und durch die dunklen Bereiche schwache Signalwerte für die quadrierten Y-Gradienten hervortreten. Dadurch ist bereits bei einer geringen Auflösung eine sehr gute Trennbarkeit durch einen zweidimensionalen Klassifikator gegeben. Während den Versuchen hat sich herausgestellt, dass tatsächlich diese zwei Tensormerkmale für den ersten Klassifikator ausgewählt werden.

Abb. 3.1.9 zeigt die Orientierungs- (Gl. (3.1.14) und (3.1.15)) sowie die Winkelmerkmale (Gl. (3.1.16)) bei voller Auflösung, d.h. für jedes Pixel wird das Merkmal berechnet und als Grauton dargestellt.

Abb. 3.1.10 zeigt die Visualisierung der Tensormerkmale bei steigender Auflösung. Die Richtungsvektoren passen sich bei steigender Auflösung immer genauer an die Strukturen der Ziffern an. Somit ist in jeder neuen Auflösungsstufe neue Information enthalten.

Als unterschiedliche Basismerkmale werden alle möglichen Rechtecke, die sich aus den Gitterrechtecken bilden lassen, verwendet. Jedes Rechteck wird wiederum mit allen möglichen Merkmalskombinationen verknüpft, woraus sich die absolute Anzahl der verwendeten Merkmale berechnen lässt, mit denen für jede Merkmalskombination und ihrer spezifischen Rechteckgeometrie entsprechende merkmalsbasierte Klassifikatoren berechnet werden. Die Anzahl n der möglichen Basisrechtecke, die sich in einem Grid mit x_d Unterteilungen in X-Achsenrichtung und y_d Unterteilungen in Y-Achsenrichtung



(a) $\Delta r = \frac{1}{3}$ (b) $\Delta r = \frac{1}{4}$

(c) $\Delta r = \frac{1}{3}$ (d) $\Delta r = \frac{1}{12}$

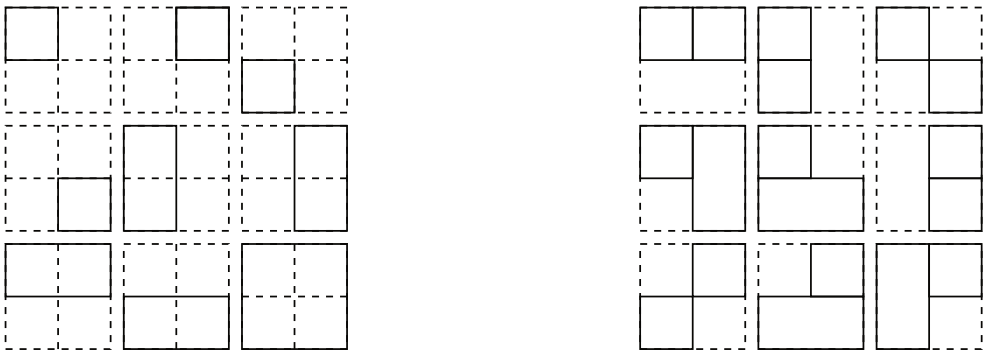
Abbildung 3.1.10: Visualisierung der Tensormerkmale bei steigender Auflösung. Die unterschiedlichen Auflösungen beinhalten unterschiedliche Informationen.

bilden lassen, kann mit der Gaußschen Summenformel berechnet werden:

$$n = \sum_{i=1}^{x_d} i \sum_{j=1}^{y_d} j = \frac{x_d y_d (x_d + 1)(y_d + 1)}{4} \tag{3.1.22}$$

Für geringe Auflösungen mit wenig Unterteilungen ist die Anzahl der Basisrechtecke gering. Um eine bessere Abdeckung des Lösungsraumes zu ermöglichen werden deshalb die Basisrechtecke miteinander kombiniert. Die Anzahl der kombinatorischen Möglichkeiten v berechnet sich bei einer k-stelligen Kombination über den Binomialkoeffizienten:

$$v = \binom{n}{k} \tag{3.1.23}$$



(a) (b)

Abbildung 3.1.11:
(a): Basisrechtecke für $\Delta r = \frac{1}{2}$.
(b): Mögliche Kombinationen der Rechtecke für $k = 2$.

Für $k = 2$ sind das zum Beispiel alle Rechteckpaare, die sich aus den Basisrechtecken bilden lassen. Die Trainingszeit der Klassifikatoren wird durch die damit verbundene Suchraumeinschränkung stark verringert. In Abb. 3.1.11(a) sind die 9 Basisrechtecke für $\Delta r = \frac{1}{2}$ dargestellt. Abb. 3.1.11(b) zeigt mögliche Kombinationen für $k = 2$.

In Tabelle 3.1.1 sind die Mengen der Basisrechtecke bei steigender Auflösung und die Menge der möglichen Kombinationen gelistet. Für kleine Auflösungen ist die Anzahl der Basisrechtecke gering und somit besteht die Möglichkeit, auch Kombinationen mit $k = 2$ in das Klassifikatortraining miteinzubeziehen. Jedem Klassifikatorkandidat $h_i \in \mathcal{H}$ wird dabei eine bestimmte

Δr	n	k_2	k_3	v_2	v_3
1/2	9	2	3	36	84
1/3	36	2	3	630	7140
1/4	100	2	3	4950	161700
1/5	225	2	3	25200	1873200
1/6	441	2	3	97020	14197260
1/12	6084	2	3	18504486	37514761284

Tabelle 3.1.1: Anzahl der Basisrechtecke n bei Auflösung Δr und die Anzahl der Kombinationen der Basisrechtecke v_2 bei $k = 2$ und v_3 bei $k = 3$.

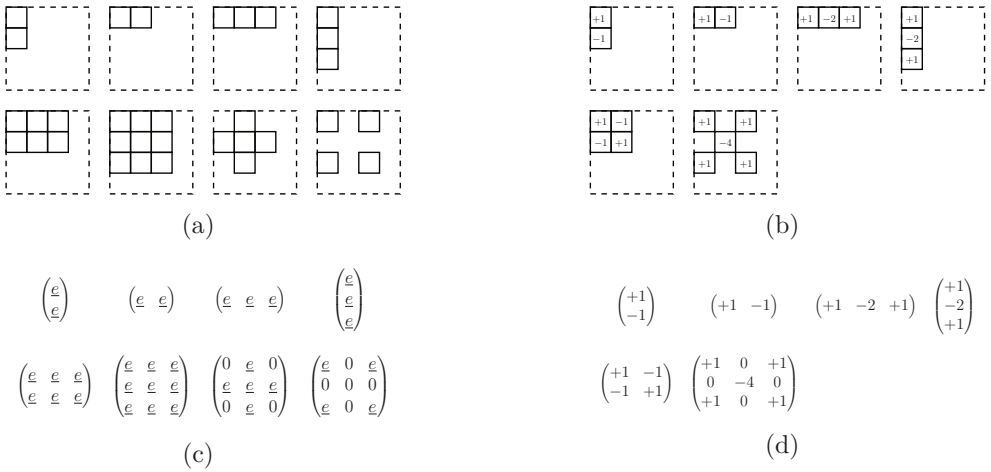


Abbildung 3.1.12:
Maskenmerkmale mit den zugehörigen Konfigurationen der Geometrien.
(a): Maskenmerkmale mit Extraktion von Merkmalen für jedes Rechteck.
(c): Entsprechende Konfiguration der Merkmale aus Abb. (a) über Matrizen.
(b): Maskenmerkmale mit festen Gewichten für jedes Rechteck.
(d): Entsprechende Konfiguration der Merkmale mit festen Gewichten aus Abb. (b).

Rechteckkombination zugewiesen:

$$\mathcal{R}_i = \{r_k | w(r_k) \geq \Delta r_x d_w, h(r_k) \geq \Delta r_y d_h\} \tag{3.1.24}$$

wobei d_w die Breite und d_h die Höhe des Detektionsfensters ist, sowie $w(r_k)$ die Breite und $h(r_k)$ die Höhe des Merkmalsrechtecks r_k ist. Aus Tabelle 3.1.1 wird ersichtlich, dass bei feineren Auflösungen die kombinatorischen Möglichkeiten für ein sequentielles Training aller Klassifikatoren zu groß werden. Deshalb werden zusätzlich noch fest konfigurierte Maskenmerkmale verwendet. Die Masken bestehen aus zweidimensional angeordneten Rechtecken, wobei einzelne Rechtecke innerhalb einer Maske aktiv sein können. Inaktive Rechtecke werden nicht berechnet.

Abb. 3.1.12 zeigt die Maskenmerkmale (Abb. (a)) und die zugehörige Konfiguration (Abb. (b)), wobei innerhalb von jedem Rechteck die zugewiesenen Merkmale extrahiert werden. Falls das 3x3-Grid z.B. mit den zweidimensionalen Orientierungen verknüpft ist, entsteht ein 18-dimensionaler Merkmalsvektor für den Klassifikator. Wird eine Null für ein Rechteck vergeben, so

wird dieses Rechteck nicht berücksichtigt. Es können damit komplexere Muster bzw. Templates für die Merkmale generiert werden. Die Maskenmerkmale können, wenn sie entsprechend gewichtet werden, als *Matched Filter* [113] bzw. *North Filter* [89] betrachtet werden. In der Signalverarbeitung sind diese Filter die optimalen linearen Filter, die das Signal-Rauschverhältnis maximieren [90]. Bei den Maskenmerkmalen wird die Faltung jedoch nicht pixelweise durchgeführt, sondern mit der integralbildbasierten Boxfilterung. Damit ist die Filterung skalierungsunabhängig und kann für beliebig große Masken angewendet werden. Außerdem können auch Maskenrechtecke mit unterschiedlichen Größen verwendet werden, was im Folgenden noch erklärt wird. Die Gewichte für den Filter werden mit einem überwachten Lernverfahren bestimmt.

Zusätzlich werden Maskenmerkmale mit festen Gewichten verwendet (Abb. 3.1.123.12(b)). Damit können Haar-ähnliche Merkmale realisiert werden, falls Intensitätswerte extrahiert werden. Darüber hinaus kann jedes beliebige andere Merkmal oder Merkmalskombinationen verwendet werden. Die Dimensionalität des Merkmalsvektors ist dann nur durch die Basismerkmale gegeben. Bei den Maskenmerkmalen werden die Masken über alle möglichen Positionen innerhalb des Detektionsfensters geschoben und so die entsprechenden Merkmalsgeometrien generiert und jedem Klassifikator zugeordnet (siehe Schritt 3 in Abb. 2.3.7). So werden zum Beispiel für eine 3x3-Maske bei $\Delta r = \frac{1}{4}$, 4 unterschiedliche Maskenmerkmale mit unterschiedlichen Positionen generiert. Eine weitere Konfigurationsmethode zur Erstellung spezifischer geometrischer Merkmale wird über eine spezielle numerische Notation und "Wildcards" erreicht:

$$\mathcal{R}_i = \{(x_0, y_0, w_0, h_0), \ldots, (x_N, y_N, w_N, h_N)\} \tag{3.1.25}$$

Hierbei sind $x_0 \in \{1, \ldots, x_d, *\}$ und $y_0 \in \{1, \ldots, y_d, *\}$ die möglichen Positionen innerhalb des Detektionsfensters, $w_0 \in \{1, \ldots, x_d, *\}$ die Breite und $h_0 \in \{1, \ldots, y_d, *\}$ die Höhe der Rechtecke. Bei der Generierung werden alle möglichen Rechtecke mit entsprechender Geometrie als Masken verwendet. Das Wildcardsymbol "$*$" entspricht allen möglichen Werten der entsprechenden Variable. So werden zum Beispiel mit der Notation $\{(*, *, *, *)\}$ alle möglichen Basisrechtecke verwendet, wodurch auch schon bei groben Auflösungen ein hochdimensionaler Klassifikator mit guter Trennfähigkeit erzeugt wird, der jedoch auch eine erhöhte Berechnungskomplexität besitzt. Eine automatisierte, komplexitätssensitive Auswahl der optimalen Klassifikatoren mit

den zugeordneten Merkmalen für ein bestimmtes Lernproblem in einer Kaskadenstufe wird durch die in Kapitel 2.3 beschriebenen Gleichungen (2.3.21) und (2.3.22) erreicht. Bei einer Auflösung von z. B. $\Delta r = \frac{1}{4}$ und Extraktion eines 5-dimensionalen Merkmalsvektors wird dann durch die 100 Rechtecke ein 500-Dimensionaler Klassifikator erzeugt (siehe Tabelle 3.1.1).

3.2 Voting-Verfahren zur Merkmalsberechnung

Mit dem Voting- oder Wahl-Verfahren kann eine weitere Klasse von Merkmalstypen beschrieben werden. Dabei werden in einer definierten Region mehrere Merkmalswerte in unterschiedlichen Bereichen berechnet, und für die Werte werden in einem unterteilten Parameterraum den entsprechenden Bereichen Stimmen (votes) zugeordnet, indem für diese Bereiche Werte akkumuliert werden. Ein bekanntes Beispiel, welches bei der Personendetektion eingesetzt wird, sind die sogenannten Histogram of Oriented Gradients Merkmale (HOG) [31, 32, 36, 128]. Innerhalb bestimmter Zellen, die wiederum in einer Gitterstruktur angeordnet sind, werden Gradientenrichtungen bestimmt und in ein Histogramm mit einer bestimmten Anzahl an Bins akkumuliert. Die Histogrammwerte aller Zellen bilden dann einen hochdimensionalen Merkmalsvektor. Bei einem Detektionsfenster der Dimension 64x128 resultiert auf diese Weise, abhängig von der Anzahl der Bins und der überlappenden Zellen, ein 3500-dimensionaler Merkmalsvektor. Allerdings haben diese Merkmale durch die zusätzliche Akkumulation der Werte eine hohe Berechnungskomplexität.

Abb. 3.2.1 zeigt eine Visualisierung typischer HOG-Merkmale zur Personendetektion [31]. In dieser Arbeit werden die HOG-Merkmale auf die bestehende gitterbasierte Merkmalsselektion und die Tensormerkmale angepasst, im Folgenden als Grid-HOG Merkmale bezeichnet, woraus mehrere Vorteile resultieren. Die tensorbasierten Orientierungsmerkmale (Gl. (3.1.16)) ermöglichen eine effektive Berechnungsmöglichkeit der Orientierungen, indem ein Basisrechteck (siehe Abb. 3.1.113.11(a)) in weitere gleichmäßige Rechtecke aufgeteilt wird, die zur Akkumulation der Orientierungen in ein Histogramm dienen. Damit können weiterhin unterschiedliche Auflösungsstufen verwendet werden und es können alle Basisrechtecke als Zellen, insbesondere auch langgestreckte Rechtecke, genutzt werden. Die Zellen werden bei unterschied-

(a) (b) (c) (d)

Abbildung 3.2.1: Histogram of Oriented Gradients - Merkmale für die Personendetektion, Abb. aus [31].
(a): Gemitteltes Gradientenbild über alle Trainingsbeispiele.
(b): Testbild einer Person.
(c),(d): R-HOG-Merkmale mit den positiven und negativen Gewichten der trainierten SVM gewichtet.

lichen Skalierungen der Detektionsfenster mitskaliert. Abb. 3.2.2 zeigt ein

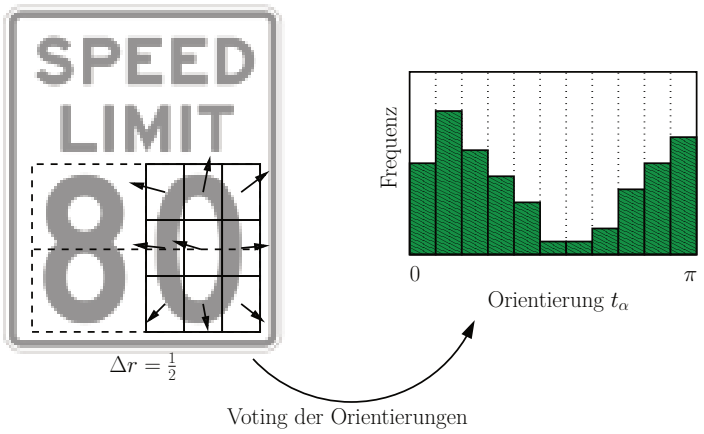


Abbildung 3.2.2: Grid-HOG Merkmalsberechnung im rechten langgestreckten Basisrechteck mit 9 Unterteilungen der Zelle bei der Auflösung $\Delta r = 1/2$. Die Orientierungen werden auf 11 unterschiedliche Bins aufgeteilt.

Beispiel mit der relativen Auflösung $\Delta r = 1/2$, wobei die Orientierungen innerhalb der rechten langgestreckten Zelle berechnet und in ein Histogramm mit elf Bins akkumuliert werden. Der Grid-HOG-Merkmalsvektor besteht aus den elf unterschiedlichen Frequenzen der Orientierungen. Somit lassen sich die Grid-HOG Merkmale mit den beschriebenen Maskengeometrien (verglei-

che Abb. 3.1.12 und Gl. (3.1.25)) verknüpfen und sogar mittels einer sequentiellen Merkmalsselektion mit anderen Merkmalen kombinieren. Ein weiterer Vorteil ist, dass sich nicht nur die Gradientenrichtungen, sondern prinzipiell alle Merkmale für das Voting in das Histogramm verwenden lassen. Es hat sich herausgestellt, dass sich auch die Exzentrizität (Gl. (3.1.18)) für das Histogrammvoting eignet und sehr gute Trenneigenschaften besitzt. Diese Merkmale werden im Folgenden als Grid-HEX Merkmale bezeichnet. Alg.

Algorithmus 3.1 Grid HOG

1. **Gegeben:** Detektionsfenster d, Unterteilungen n in X-Richtung, Unterteilungen m in Y-Richtung
2. Initialisiere den Merkmalsvektor $\underline{x} = \underline{0}$.
3. Berechne nm äquidistante Zellen $\mathcal{Z}$ innerhalb des Detektionsfensters d
4. **Iteriere** über die Zellen $z \in \mathcal{Z}$:
   (a) Berechne den Orientierungswinkel α innerhalb z mit Gl. (3.1.16)
   (b) Gradientenbetrag $g = \sqrt{\overline{i_x^2}(z) + \overline{i_y^2}(z)}$ (Gln. (3.1.11),(3.1.12))
   (c) Voting: $x(\lfloor \alpha \rfloor) \leftarrow x(\lfloor \alpha \rfloor) + g$
5. Normiere $\underline{x}$

(3.1) beschreibt das Grid-HOG Votingverfahren, wobei in den Merkmalsvektor gevotet wird. Die Winkelberechnung erfolgt mittels des Integralbildes und Gl. (3.1.16). In Schritt 4b wird der Gradientenbetrag berechnet, mit dem dann in Schritt 4c gevotet wird. Für das Grid-HEX Voting wird lediglich anstatt des Orientierungswinkels in Schritt 4a die Exzentrizität verwendet.

3.3 Merkmalsnormierung

Die Vorgehensweise zur Normierung bei Klassifikationsverfahren besteht darin, den zu klassifizierenden Bildausschnitt auf eine definierte Größe zu skalieren und dann diesen Bildauschnitt mit einem Normierungsverfahren, wie z. B. einer Helligkeits-, Kontrast- oder Histogrammnormierung zu bearbeiten. Für eine echtzeitfähige Detektion lässt sich dieses Vorgehen allerdings nicht

effizient durchführen, da alle Pixel mehrfach betrachtet werden müssen. Zur Intensitätsnormierung wird deshalb das integralbildbasierte Normierungsverfahren [116, 114] verwendet, das eine konstante und effiziente Normierung ermöglicht. Für die Tensormerkmale werden zwei unterschiedliche Normierungsverfahren vorgestellt.

Bei den Kaskadenklassifikatoren spielt die Normierung der Merkmale eine große Rolle, da die Stufenklassifikatoren sequentiell ausgeführt werden und sich der Effekt einer Normierung stark auf die Auswahl der Beispiele der folgenden Stufenklassifikatoren, sowohl beim Training, als auch bei der Ausführung des Klassifikators, auswirkt. Je mehr Stufenklassifikatoren verwendet werden, desto stärker spiegelt sich der Effekt wieder. Es lässt sich zwischen einer direkten Normierung der Merkmale bei der Extraktion unterscheiden und einer Normierung der Merkmale nach der Extraktion, bevor sie in den Klassifikationsprozess einfließen. Als Normierung bei der Extraktion wird für die Intensitätsmerkmale (Gl. (3.1.5),(3.1.6)) eines Merkmalrechtecks der Mittelwert des Detektionsrechtecks, also der gesamten Detektionsregion, abgezogen und durch die Varianz geteilt:

$$\bar{i}_n(r) = (\bar{i}(r) - \bar{i}(d))/\overline{\sigma^2}(d) \tag{3.3.1}$$

$$\overline{i^2}_n(r) = (\overline{i^2}(r) - \overline{i^2}(d))/\overline{\sigma^2}(d) \tag{3.3.2}$$

mit

$$\overline{\sigma^2}(d) = \overline{i^2}(d) - (\bar{i})^2(d) \tag{3.3.3}$$

wobei d das Detektionsfenster und $\overline{\sigma}(d)$ die Varianz ist, die sich effektiv durch das Integralbild berechnen lässt. Die weiteren Intensitätsmerkmale werden auf die gleiche Art und Weise normiert. Für die Tensormerkmale werden zwei unterschiedliche Normierungen bei der Extraktion verwendet. Bei den US-Verkehrszeichen hat sich die Normierung der Tensormerkmale (Gl. (3.1.11), (3.1.12) und (3.1.13)) mittels der Frobeniusnorm als wirksam herausgestellt:

$$\overline{i^2_{x,fn}}(r) = \overline{i^2_x}(r)/\overline{\phi}(r) \tag{3.3.4}$$

$$\overline{i^2_{y,fn}}(r) = \overline{i^2_y}(r)/\overline{\phi}(r) \tag{3.3.5}$$

$$\overline{i_{xy,fn}}(r) = \overline{i_{xy}}(r)/\overline{\phi}(r) \tag{3.3.6}$$

mit

$$\overline{\phi}(r) = \sqrt{\overline{i^2_x}(r) + \overline{i^2_y}(r) + 2\overline{i_{xy}}(r)} \tag{3.3.7}$$

Hierbei werden lediglich die Merkmalsrechtecke zur Normierung verwendet.

Eine weitere Möglichkeit zur Normierung, wobei die Werte des Detektionsrechtecks subtrahiert werden, ist:

$$\overline{i^2_{x,fn}}(r) = (\overline{i^2_x}(r) - \overline{i^2_x}(d))/\overline{\phi_s}(r) \tag{3.3.8}$$

$$\overline{i^2_{y,fn}}(r) = (\overline{i^2_y}(r) - \overline{i^2_y}(d))/\overline{\phi_s}(r) \tag{3.3.9}$$

$$\overline{i_{xy,fn}}(r) = (\overline{i_{xy}}(r) - \overline{i_{xy}}(d))/\overline{\phi_s}(r) \tag{3.3.10}$$

mit

$$\overline{\phi_s}(r) = \overline{i^2_x}(r) + \overline{i^2_y}(r) + \overline{i_{xy}}(d) \tag{3.3.11}$$

Bei der Normierung nach der Extraktion werden von den extrahierten Merkmalsvektoren jeder Klasse zuerst die Mittelwerte und Varianzen berechnet. Die Normierung besteht dann aus Subtraktion des Mittelwertvektors $\underline{\mu}_\omega$ und Division des Varianzvektors $\underline{\sigma}^2_{\omega_i}$ für jede Klasse ω_i:

$$\underline{x}_n = (\underline{x} - \underline{\mu}_{\omega_i})/\underline{\sigma}^2_{\omega_i} \tag{3.3.12}$$

Bei dieser Normierung muss also für jede Klasse ein Mittelwert- und Varianzvektor ermittelt und mitgeführt, bzw. abgespeichert werden. Außerdem wird die Normierung gleichermaßen beim Training und bei der Ausführung des Klassifikators durchgeführt.

3.4 Merkmalstransformation

Bei den meisten Klassifikatoren wird ausgehend von den Merkmalsvektoren eine Klassenzugehörigkeit bestimmt und damit eine Klassifikationshypothese aufgestellt. Die Entscheidung für eine Klasse wird oft mittels einer Schätzung der Klassendichte im Merkmalsraum bestimmt. So wird z. B. bei einem Nächste-Nachbarn-Klassifikator die Klassendichte über die Anzahl der klassenspezifischen Nachbarn geschätzt und für diejenige Klasse entschieden, für welche die meisten Nachbarn gefunden wurden. Da es in dieser Arbeit allerdings auf eine echtzeitfähige Klassifikation ankommt, wäre ein NN-Klassifikator durch die Nachbarsuche nicht effektiv berechenbar bei der Ausführung. Außerdem müssten alle klassenspezifischen Vektoren im Speicher mitgeführt werden. Somit wäre auch der Speicherbedarf für einen K-Stufigen Kaskadenklassifikator bei einer hohen Anzahl an Vektoren sehr hoch. Eine weitere Klassifikatorstruktur sind die Support Vector Machines (SVM). Bei diesen Klassifikatoren ist jedoch der Trainingsaufwand sehr hoch, so dass eine

globale Optimierung mit einer höheren Population an Kandidatenklassifikatoren in vertretbarem Aufwand nicht möglich ist. Darüberhinaus werden zur Klassifikationsfunktion im Allgemeinen eine hohe Anzahl an Supportvektoren für die Projektion verwendet, wodurch die Ausführung des Klassifikators eine hohe Berechnungskomplexität besitzt.

Bei dem Kaskadendetektionsverfahren von Viola und Jones werden im Gegensatz dazu, schwache Klassifikatoren ("weak classifier"), d. h. ein skalares Merkmal mit einem Schwellwert, verwendet. Diese weak classifier werden dann mit einem sequentiellen AdaBoost Verfahren einem starken Klassifikator ("strong classifier") zugeordnet, wobei die Gewichte der weak classifier reziprok proportional zu ihrer Fehlerwahrscheinlichkeit gewählt werden. Es gibt jedoch keine Vorschrift, wieviele weak classifier für die strong classifier in den Kaskadenstufen verwendet werden sollen. Die Bestimmung der Anzahl ist ein Designkriterium des Klassifikators.

In dieser Arbeit wird ein statistisches Optimierungsprinzip vorgestellt, das eine automatische Auswahl zwischen unterschiedlichen Klassifikatoren mit unterschiedlicher Berechnungskomplexität und Dimensionalität zur Reduktion der Berechnungskomplexität des Kaskadenklassifikators ermöglicht. Mit diesem Verfahren können effiziente Stufenklassifikatoren ohne Boosting-Verfahren verwendet werden, da eine hohe Anzahl an geboosteten Klassifikatoren die Berechnungskomplexität des Klassifikators erheblich erhöht. Die multidimensionalen Merkmalsvektoren werden auf eine skalare Variable projiziert, was bei der Anwendung effizient durchführbar ist. Außerdem ist die Berechnungskomplexität der statistischen Lernverfahren linear in der Anzahl der Trainingsbeispiele und damit auch für eine größere Anzahl an Kandidaten effizient durchführbar. Als Projektionsverfahren wird zum Einen die gewichtete Fisher's Lineare Diskriminante (FLD) verwendet [37, 4, 35]. Der Unterschied zur Linearen Diskriminanzanalyse (LDA) ist, dass bei der FLD die Voraussetzungen, dass die Klassen normalverteilt und identische Kovarianzen besitzen müssen, nicht existieren. Somit lässt sich die FLD auf beliebige Probleme anwenden, insbesondere auf die Teilprobleme der Kaskadenstufen mit steigender Komplexität. Das andere statistische Verfahren zur Projektion, das in dieser Arbeit verwendet wird, ist die gewichtete multiple lineare Regression (MLR). Die zusätzliche Gewichtung der Merkmalsvektoren bei beiden Verfahren ermöglichen die Kombination dieser Verfahren mit Verfahren der Datenanpassung (data alignment), wie z.B. Boostingverfahren, bei denen die Merkmalsvektoren je nach Klassifikationsergebnis und damit nach der Auftrittswahrscheinlichkeit gewichtet werden können. Die Boostingver-

fahren mit der Fisher's linearen Diskriminanten als Trainingsverfahren für die Basisklassifikatoren sind in Kapitel 6 genau beschrieben. Ein weiterer Vorteil beider statistischer Verfahren ist, dass keine Parameter zum Berechnen des Gewichtsvektors benötigt werden, was für die unterschiedlichen Teilprobleme der Stufen durch die sequentielle Struktur von großem Vorteil ist.

3.4.1 Fisher's lineare Diskriminante (FLD)

Das Optimierungsprinzip zur Berechnung der Gewichte für die Projektionsachse der Fisher's linearen Diskriminante (FLD) [37, 4, 35] ist eine Maximierung der Interklassentrennung und zugleich eine Minimierung der Intraklassentrennung über den Rayleigh-Quotienten. Es lässt sich zeigen, dass sich die FLD in eine Methode des kleinsten mittleren quadratischen Fehlers (MSE) überführen lässt [35]. Somit wird durch diese Methode der kleinstmögliche quadratische Fehler erreicht. Bei der FLD werden aus den Trainingsdaten $\mathcal{D} = \{(\underline{x}_1, \gamma_1, y_1), \ldots, (\underline{x}_N, \gamma_N, y_N)\}$, d. h. den Merkmalsvektoren $\underline{x}_j$ für jede Klasse $y_j = \omega_i$, die mit γ gewichteten Mittelwerte:

$$\underline{\mu}_{\omega_i} = \mathrm{E}[\underline{X}] \approx \sum_{j=1}^{N} \gamma_j \underline{x}_j \tag{3.4.1}$$

und die gewichteten Kovarianzen:

$$\begin{aligned} \mathbf{\Sigma}_{\omega_i} &= \mathrm{E}[(\underline{X} - \mathrm{E}[\underline{X}])(\underline{X} - \mathrm{E}[\underline{X}])^T] \\ &\approx \frac{\sum_{j=1}^{N} \gamma_i}{(\sum_{j=1}^{N} \gamma_i)^2 - \sum_{j=1}^{N} \gamma_i^2} \sum_{j=1}^{N} \gamma_i (\underline{x}_j - \underline{\mu}_{\omega_i})(\underline{x}_j - \underline{\mu}_{\omega_i})^T \end{aligned} \tag{3.4.2}$$

bestimmt, wobei die Gewichte normalisiert sind ($\sum_{j=1}^{N} \gamma_i = 1$). Mit der Normierung wird außerdem eine unterschiedliche Anzahl an Beispielen ausgeglichen, die in hohen Stufen auftritt, da die Beispiele der Hintergrundklasse mit steigenden Stufen stetig abnehmen. Durch Maximierung des Rayleigh-Quotienten:

$$J(\underline{w}) = \frac{\underline{w}^T (\underline{\mu}_{\omega_1} - \underline{\mu}_{\omega_0})(\underline{\mu}_{\omega_1} - \underline{\mu}_{\omega_0})^T \underline{w}}{\underline{w}^T (\mathbf{\Sigma}_{\omega_1} + \mathbf{\Sigma}_{\omega_0}) \underline{w}} \tag{3.4.3}$$

berechnet sich die Projektionsachse $\underline{w}$ der FLD [35]:

$$(\mathbf{\Sigma}_{\omega_1} + \mathbf{\Sigma}_{\omega_0}) \underline{w} \sim \underline{\mu}_{\omega_1} + \underline{\mu}_{\omega_0} \tag{3.4.4}$$

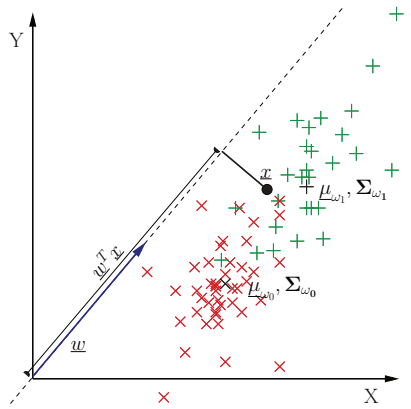


Abbildung 3.4.1: 2D-Visualisierung der blauen, normierten Projektionsachse. Die Berechnung beim Training erfolgt durch Gl. (3.4.4) mit den klassenspezifischen Merkmalsvektoren der Objektklasse (grün) und der Hintergrundklasse (rot).

Die Achse wird normiert, so dass $\|\underline{w}\| = 1$. Die Kovarianzmatrizen werden absichtlich nicht invertiert, da eine direkte Berechnung von Gl. (3.4.4) mit einer Cholesky-Zerlegung [95] , die auch ohne Pivotisierung numerisch extrem stabil ist, und die um den Faktor zwei schneller als alternative Gleichungslöser [95], effizienter ist. Bei der Ausführung wird ein unbekannter Vektor $\underline{x}$ auf die Projektionsachse projiziert:

$$s_{\mathrm{FLD}}(\underline{x}) = \underline{w}^T \underline{x} \tag{3.4.5}$$

Die Klassenentscheidung übernimmt eine Entscheidungseinheit die im Folgenden erklärt wird. Der Aufwand ist linear in der Anzahl der Merkmale bzw. in der Dimensionalität des Merkmalsvektors. Für niedrigdimensionale Merkmale kann diese Projektion also effizient berechnet werden.

In Abb. 3.4.1 ist eine Visualisierung der Projektionsachse für den 2D-Fall dargestellt. Mittels Gl. (3.4.4) wird die blaue Projektionsachse beim Training bestimmt. Bei der Ausführung des Klassifikators wird ein unbekannter Merkmalsvektor $\underline{x}$ auf die Achse projiziert (Gl. (3.4.5)) und mittels eines Entscheidungsmoduls eine Klassenzuweisung vorgenommen. Der Trainingsalgorithmus besteht also aus der Berechnung der Mittelwerte (Gl. (3.4.1)), der Kovarianzen (Gl. (3.4.2)) und aus der Lösung von Gl. (3.4.4), was für kleine bis mittelgroße Matrizen effektiv durchführbar ist. Das Berechnen der Mittelwerte und Kovarianzen ist linear in der Anzahl der Beispielsvektoren

der Klassen $\mathcal{O}(N)$ und die Cholesky-Zerlegung benötigt eine kubische Anzahl an Operationen in der Merkmalsdimension $\mathcal{O}(d^3/6)$. Das Trainingsverfahren ist in Algorithmus 3.2 dargestellt. Da eine ganze Population an Klassifikatoren trainiert wird, wird auf eine Regularisierung der Matrix verzichtet. Falls keine Lösung gefunden wird, so werden die Gewichte in Schritt 2b gleichgesetzt und normiert, was einer Mittelung aller Merkmalswerte entspricht.

Algorithmus 3.2 Trainiere FLD Projektion $\underline{w}$

1. Berechne $\mathbf{\Sigma}_{\omega_i}$ und μ_{ω_i} (Gln. (3.4.1),(3.4.2)).
2. Löse $\underline{w}$ mit Gl. (3.4.4).
   (a) Falls eine Lösung gefunden wurde, wird normiert $\|\underline{w}\| = 1$.
   (b) Sonst wird $w_j = c, 1 <= j <= d$ konstant gesetzt, $\|\underline{w}\| = 1$.

3.4.2 Multiple Lineare Regression (MLR)

Ein lineares Regressionsmodell hat die Form [56]:

$$s_{\mathrm{MLR}}(\underline{x}) = w_0 + \underline{w}^T \underline{x} = \hat{y} \tag{3.4.6}$$

und ist somit bis auf das Absolutglied w_0 identisch mit der Projektionsgleichung Gl. (3.4.5). Die Regressionskoeffizienten bzw. Parameter $\underline{w}$ sind unbekannt und die Kovariablenvektoren $\underline{x}$ stammen aus einer Zufallsvariablen. Der Vorteil dieses linearen Modells ist zum einen eine gute Prädiktionsgüte bei unbekannten Daten, zum anderen auch eine schnelle Ausführbarkeit bei der Anwendung. Außerdem sind lineare Modelle oftmals weniger anfällig bei Daten mit geringem Signal-Rauschverhältnis und tendieren weniger stark zur Überadaption als komplexere nichtlineare Modelle [56]. Das Gauss-Markov-Theorem bestätigt sogar, dass die Schätzfunktionen der Parameter $\underline{w}$ mittels der Kleinsten-Quadrate-Methode die kleinste Varianz unter allen unverzerrten linearen Schätzfunktionen besitzen [56] (Best Linear Unbiased Estimator). Für die Eigenschaften der Schätzfunktion muss keine Verteilungsinformation der Störgröße vorliegen. Die Bestimmung der Parameter erfolgt deshalb über die Minimierung des kleinsten mittleren quadratischen Fehlers

e:

$$e(\underline{w}) = (\underline{y} - \mathbf{X}\underline{w})^T(\underline{y} - \mathbf{X}\underline{w}) \tag{3.4.7}$$

wobei $\mathbf{X}^T = (\underline{x}_1, \ldots, \underline{x}_N)$ ist und $\underline{y}^T = (y_1, \ldots, y_N)$ aus den kategorischen Werten $y_i \in \{+1, -1\}$ für die Objekt- und Hintergrundklasse besteht. Unter der Annahme, dass die zweite Ableitung $2\mathbf{X}^T\mathbf{X}$ positiv definit ist, und durch Gleichsetzen der ersten Ableitung mit Null, resultiert die eindeutige Lösung:

$$\mathbf{X}^T\mathbf{X}\,\underline{w} = \mathbf{X}^T\underline{y} \tag{3.4.8}$$

Die zweite Ableitung und die rechte Seite von Gl. (3.4.8) lassen sich iterativ mit einer zusätzlichen Gewichtung γ berechnen:

$$\mathbf{X}^T\mathbf{X} = \sum_{i=1}^{N} \gamma_i^2 \underline{x}_i \underline{x}_i^T \tag{3.4.9}$$

$$\mathbf{X}^T\underline{y} = \sum_{i=1}^{N} \gamma_i \underline{x}_i y_i \tag{3.4.10}$$

Somit lässt sich dieses Verfahren auch mit Verfahren, die eine Datenanpassung durch Neugewichtung der Beispiele vornehmen, wie z.B. Boostingverfahren, verwenden. Falls das Absolutglied w_0 in Gl. (3.4.6) nicht verwendet wird, kann auf ein Auffüllen der ersten Spalte der Matrix $\mathbf{X}$ mit Einsen verzichtet werden. Unter der Voraussetzung der positiven Definitheit der zweiten Ableitung und der Tatsache, dass die Matrix symmetrisch ist, existiert wiederum die Cholesky-Zerlegung [95]. Die Parameter $\underline{w}$ aus Gl. (3.4.8) können dann wieder direkt berechnet werden, ohne dass eine explizite Matrixinversion stattfindet, wodurch die Berechnungskomplexität weiter vereinfacht wird. Der Trainingsaufwand und der Aufwand für die Ausführung der Projektion ist somit identisch zur FLDA. Das Berechnen von $\mathbf{X}^T\mathbf{X}$ und $\mathbf{X}^T\underline{y}$ benötigt linearen Aufwand (siehe Gl. (3.4.9) und (3.4.10)) und die Cholesky-Zerlegung benötigt kubischen Aufwand in der Merkmalsdimension. Das Verfahren ist in Algorithmus 3.3 dargestellt.

3.5 Klassenzuordnung

Die Zuordnung der Klassen und somit die Entscheidung des Klassifikators übernimmt ein separates Entscheidungsmodul. Dadurch können unterschiedliche Diskriminantenfunktionen verwendet und getestet werden. Als Eingabe

Algorithmus 3.3 Trainiere MLR Projektion $\underline{w}$

1. Berechne $\mathbf{X}^T\mathbf{X}$ und $\mathbf{X}^T\underline{y}$ (Gln. (3.4.9),(3.4.10)).
2. Löse $\underline{w}$ mit Gl. (3.4.8).
   (a) Falls eine Lösung gefunden wurde, wird normiert $\|\underline{w}\| = 1$.
   (b) Sonst wird $w_j = c, 1 <= j <= d$ konstant gesetzt, $\|\underline{w}\| = 1$.

für die Funktionen dienen die projizierten Merkmalswerte. Da sich bei einem Kaskadenklassifikator durch die sequentielle Ausführung der Stufenklassifikatoren die Wahrscheinlichkeiten multiplizieren, kann der Klassifikationsfehler drastisch reduziert werden, indem durch ein Lernverfahren ein spezifisches Signifikanzniveau sichergestellt wird. Dies kann durch die Modellierung der klassenspezifischen Wahrscheinlichkeiten erreicht werden, indem diese durch einen Korrektur-Prior geeignet skaliert werden, der mit den A-priori-Wahrscheinlichkeiten multipliziert wird. Bei einem Signifikanzniveau von z.B. 2% liegt die Fehlerwahrscheinlichkeit einer Falsch-Negativ-Entscheidung bei 2% auf den Trainingsdaten und damit die Detektionsrate bei 98%.

Der erste Teil dieses Abschnitts beschreibt die Algorithmen zur Bestimmung und zur Modifikation der Frequenztabellen. Die klassenspezifischen Wahrscheinlichkeiten, die zur Bayes-Entscheidungsfunktion benötigt werden, werden mit Frequenztabellen auf den transformierten Merkmalswerten modelliert. Pettersson [92] beschreibt ein ähnliches Modell, wobei "Histogramm-Merkmale" als schwache Klassifikatoren mit einem AdaBoost-Verfahren [39, 40] verwendet werden.

Bei den hier verwendeten Frequenztabellen existiert für jede Klasse ein Kanal mit Einträgen, die eine spezifische Frequenz besitzen, welche beim Training ermittelt wird. Bei der Ausführung müssen lediglich die entsprechenden Kanäle mit einer Lookup-Operation aus dem transformierten Merkmalswert ermittelt werden. Im zweiten Teil werden die Algorithmen beschrieben, die bei einer schwellwertbasierten Entscheidung zur Bestimmung des Schwellwerts verwendet werden.

Algorithmus 3.4 Bestimmung der Frequenztabelle

1. **Gegeben**:
   (a) Anzahl der Einträge b der Frequenztabelle f
   (b) klassenspezifische Merkmale $\mathcal{D}$
2. Normierung der klassenspezifischen Gewichte $\sum_{i=1}^{N} \gamma_i = 1$, $\gamma_i \in \mathcal{D}_{\omega_j}$.
3. Berechnung der mimimalen und maximalen Projektionswerte:
   $s_{\max} = \max_{\underline{x}_i \in \mathcal{D}} s(\underline{x}_i)$, $s_{\min} = \min_{\underline{x}_i \in \mathcal{D}} s(\underline{x}_i)$
4. **Iteriere** für jede Klasse $\underline{x}_i \in \mathcal{D}_{\omega_j}$:
   (a) Berechne Eintrag : $k = \lfloor \frac{s(\underline{x}_i) - s_{\min}}{s_{\max} - s_{\min}} b \rfloor$
   (b) Akkumulation: $f_k \leftarrow f_k + \gamma_i$ (Gewicht γ_i)

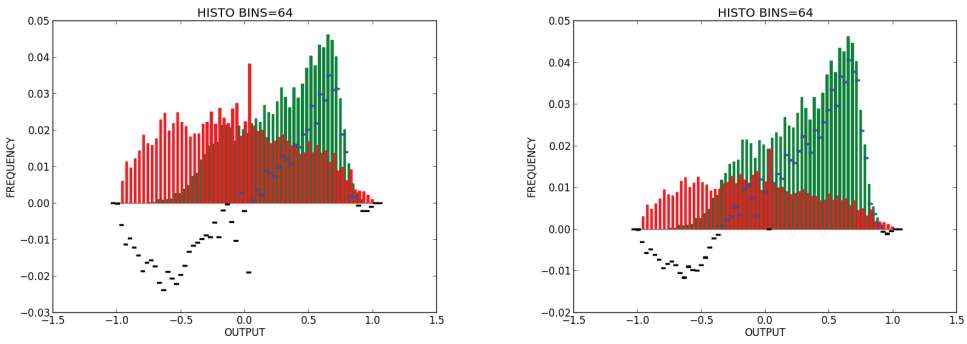
3.5.1 Bestimmung und Modifikation der Frequenztabellen

Indem die Wahrscheinlichkeiten durch Frequenztabellen modelliert werden, die für jede Klasse einen Frequenzkanal bereitstellen, kann ein effizienter, linearer Algorithmus entworfen werden. Im ersten Teil des Algorithmus wird das Modell der klassenspezifischen Wahrscheinlichkeiten mittels der Frequenztabelle berechnet. Das Verfahren ist in Algorithmus 3.4 dargestellt. Die Anzahl der Einträge haben direkten Einfluss auf den in Gl. (2.3.14) beschriebenen Erwartungswert der Wahrscheinlichkeiten in der lokalen Umgebung, der durch die Anzahl der Einträge gesteuert werden kann. Durch die normierten Gewichte wird sichergestellt, dass die Integrale der klassenspezifischen Wahrscheinlichkeiten auch für eine unterschiedliche Anzahl von Beispielvektoren für die Klassen identische Flächen besitzen. Anhand der Frequenztabelle kann eine Bayesentscheidung für jeden Eintrag mittels Gl. (2.3.12) erfolgen, oder es kann ein einzelner Schwellwert gesetzt werden.

Die Wahrscheinlichkeiten der Fehlklassifikationen bei gegebener Klassifikationsfunktion können ebenfalls effektiv anhand der Frequenztabelle berechnet werden. Durch die gewichtete Akkumulation beim Erstellen der Frequenztabelle ist das Verfahren auch mit einer Datenanpassung mit den Gewichten γ_i für die Merkmalsvektoren anwendbar.

Algorithmus 3.5 Ermittlung der Wahrscheinlichkeiten der Fehlklassifikationen

1. **Gegeben**: Frequenztabelle f mit b Einträgen und normierten Flächen
2. Setze $\text{fpr} = 0, \text{fnr} = 0$.
3. **Iteriere** über die Einträge f_i:
   (a) $\kappa = \text{classify}(f_i(\omega_0, \omega_1))$
   (b) Falls $\kappa = \omega_1$: $\text{fpr} \leftarrow \text{fpr} + f_i(\omega_0)$
   (c) Falls $\kappa = \omega_0$: $\text{fnr} \leftarrow \text{fnr} + f_i(\omega_1)$



(a) (b)

Abbildung 3.5.1: Klassenspezifische Verteilung der Häufigkeiten durch Frequenztabellen mit 64 Einträgen vor (Abb. (a)) mit $P(\omega_1) = 0.77$ und $P(\omega_0) = 0.38$ und nach (Abb. (b)) der Korrektur mit $P(\omega_1) = 0.95$ und $P(\omega_0) = 0.62$. Die negativen Frequenzen (rot) wurden mit $P_c(\omega_0) = 0.503$ skaliert (Alg. (3.6)).

Alg. (3.5) beschreibt die Bestimmung der Wahrscheinlichkeiten der Fehlklassifikationen $P(h(\underline{x}) = \omega_1|\omega_0) \approx \text{fpr}$ und $P(h(\underline{x}) = \omega_0|\omega_1) \approx \text{fnr}$ und damit auch der Auftrittswahrscheinlichkeiten der Kassen nach der Klassifikation mit h.

Die Auftrittswahrscheinlichkeit der Objektklasse liegt bei dem Beispiel in Abb. 3.5.13.1(a) nach der Klassifikation bei $P(\omega_1) = 0.77$, was als Stufenklassifikator die Detektionsperformance drastisch verschlechtert. Deshalb werden

Algorithmus 3.6 Training des Signifikanzniveaus

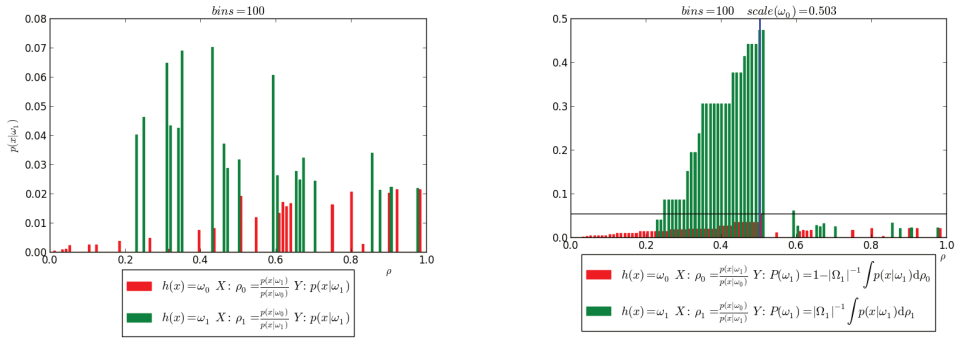
1. **Gegeben**: Signifikanzniveau α, Frequenztabelle f mit b Einträgen
2. Initialisierung einer Hilfsfrequenztabelle m mit k Einträgen
3. **Iteriere** über alle Einträge f_i:
   (a) Falls $f_i(\omega_1) \geq f_i(\omega_0)$, akkumuliere TPs:
   $j = \lfloor (k-1) f_i(\omega_0)/f_i(\omega_1) \rfloor, m_j(TP) \leftarrow m_j(TP) + f_i(\omega_1)$
   (b) Sonst, akkumuliere FNs:
   $j = \lfloor (k-1) f_i(\omega_1)/f_i(\omega_0) \rfloor, m_j(FN) \leftarrow m_j(FN) + f_i(\omega_1)$
4. Berechne die kumulierten Dichtequotienten für TP und FN:
   $M_i = \sum_{j=1}^{i} m_j$
5. **Iteriere** über die Einträge M_i:
   (a) Falls $M_i(FN) \geq \alpha$: $P_c(\omega_0) = i/(k-1)$, stop.
   (b) Falls $M_i(TP) \geq 1-\alpha$: $P_c(\omega_0) = (i/(k-1))^{-1}$, stop.
   (c) Sonst: Gehe zum nächsten Eintrag.
6. Multipliziere die Einträge $f_i(\omega_0)$ mit $P_c(\omega_0)$.

durch den folgenden Algorithmus die Wahrscheinlichkeiten auf eine definierte Fehlerwahrscheinlichkeit der Objektklasse skaliert. Daraus resultiert ein Klassifikator mit einem spezifischen Signifikanzniveau der Entscheidung. In Algorithmus 3.6 ist das Verfahren beschrieben. In einem ersten Schritt werden die kumulierten Wahrscheinlichkeiten berechnet und in eine Hilfsfrequenztabelle m akkumuliert. Für eine negative Entscheidung werden die kumulierten falsch negativen in m im FN-Kanal akkumuliert, ansonsten werden die kumulierten richtig positiven Entscheidungen im TP-Kanal akkumuliert. Die Entscheidung in Schritt 3a basiert auf der Bayes-Entscheidung. Bei der Zuweisung in Schritt 5a und 5b wird zusätzlich noch zwischen den beiden Einträgen M_{i-1} und M_i linear interpoliert um die Genauigkeit zu erhöhen. Da die Frequenztabelle m kumuliert wird, kann eine wesentlich höhere Anzahl an Einträgen gegenüber f verwendet werden um eine höhere Genauigkeit zu erreichen. Der resultierende Korrekturprior P_c wird dann entweder bei der Bayes-Entscheidungsfunktion verwendet, oder die Frequenzen $f_i(\omega_0)$ werden mit P_c skaliert. Somit ist auf den Trainingsdaten sichergestellt, dass

$$P(h(\underline{x}) = \omega_0 | \omega_1) \approx \alpha$$

Die modifizierte Frequenztabelle bildet bei der Bayes-Entscheidung die Entscheidungsfunktion des Klassifikators.

In Abb. 3.5.1 sind die Häufigkeitsverteilungen der Klassen für unterschiedliche Merkmale anhand von Frequenztabellen gezeigt. Die blauen Balken gehören zu Einträgen mit Entscheidungen für die Objektklasse und die schwarzen Balken stehen für Entscheidungen für die Hintergrundklasse. Es kann mehrere nichtzusammenhängende Entscheidungsregionen für jede Klasse geben, was an den schwarzen Balken auf der rechten Seite der positiven Verteilung zu erkennen ist. Die Auftrittswahrscheinlichkeit der Objekt- und Hintergrundklasse vor der Skalierung auf das Signifikanzniveau von 5% liegt bei Abb. 3.5.13.1(a) bei $P(\omega_1) = 0.77$ und $P(\omega_0) = 0.38$. Mit Algorithmus 3.6 wurde der Korrektur-Prior $P_c(\omega_0) = 0.503$ ermittelt, mit welchem die Häufigkeiten der Hintergrundklasse (rot) skaliert wurden, wodurch letztlich eine Auftrittswahrscheinlichkeit von $P(\omega_1) = 0.95$ und $P(\omega_0) = 0.62$ erreicht wird.



(a) (b)

Abbildung 3.5.2: (a): Wahrscheinlichkeitsdichtequotienten nach Schritt 3 in Alg. (3.6) in Tabelle m. (b): Kumulierte Dichtequotienten M nach Schritt 4.

Abb. 3.5.23.2(a) zeigt die Wahrscheinlichkeitsdichtequotienten für die TP-Entscheidung $P(h(\underline{x}) = \omega_1|\omega_1)$ und die FN-Entscheidung $P(h(\underline{x}) = \omega_0|\omega_1)$ nach Schritt 3 in Alg. (3.6). In Abb. 3.5.23.2(b) sind die kumulierten Dichtequotienten nach Schritt 4 dargestellt. In Schritt 5 wird dann über die Einträge M_i iteriert, bis das Signifikanzniveau α der FN-Entscheidung (rot) oder $1-\alpha$ der TP-Entscheidung (grün) erreicht wird. $P_c(\omega_0)$ wird dann auf der X-Achse abgelesen (blauer Balken). In dieser Abbildung wurde zur Visualisierung eine

Algorithmus 3.7 Lokal kumulierte Frequenztabelle

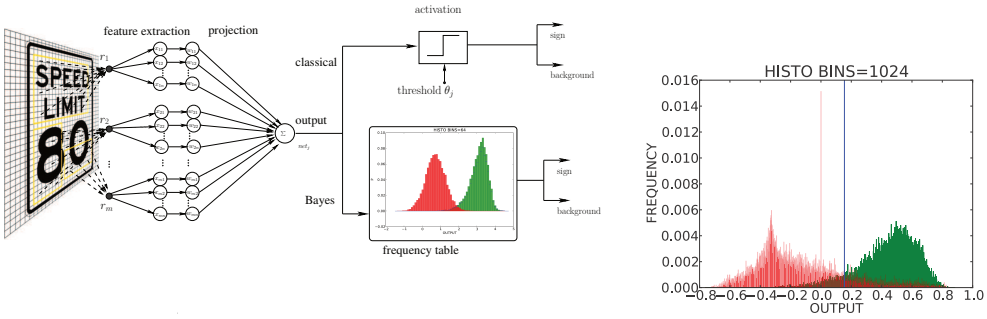
1. **Gegeben**:
   (a) Frequenztabelle f mit b Einträgen und normierten Flächen
   (b) Minimaler Anteil ξ
2. Initialisiere Hilfsfrequenztabelle t als Kopie von f.
3. **Iteriere** über die Einträge t_i:
   (a) $j = 0$, $s(\omega_1) = t_i(\omega_1)$, $s(\omega_0) = t_i(\omega_0)$
   (b) **Solange** $((s(\omega_1) < \xi) \vee (s(\omega_0) < \xi))$:
      i. $s(\omega_l) \leftarrow s(\omega_l) + t_{i+j}(\omega_l)$
      ii. $s(\omega_l) \leftarrow s(\omega_l) + t_{i-j}(\omega_l)$
      iii. $l \in \{0, 1\}$
   (c) $f_i(\omega_l) = s(\omega_l)$
4. Normiere f.

niedrige Anzahl an Bins verwendet. Für den Algorithmus werden $k = 10000$ Bins verwendet, da bei einer hohen Anzahl an Bins der Skalierungsfaktor genauer ermittelt werden kann. Die Laufzeit des Algorithmus ist linear in der Anzahl k der Einträge der Hilfstabelle m, $\mathcal{O}(k)$. Somit lässt sich das Signifikanzniveau für jeden Kandidatenklassifikator mit auf Frequenztabellen basierenden Entscheidungsfunktionen effizient einstellen.

Bei der experimentellen Bestimmung der optimalen Anzahl der Einträge für die Frequenztabellen hat sich gezeigt, dass die Anzahl der Einträge einen großen Einfluss auf den Klassifikationsfehler und die mittlere Laufzeit hat.

Deshalb wurde ein weiterer Algorithmus verwendet, der eine lokale Kumulation benachbarter Einträge in den Kanälen vornimmt, bis ein bestimmter Anteil an der Gesamtmenge des zugehörigen Kanals erreicht ist. Diese lokal kumulierte Frequenztabelle stellt dann sicher, dass mindestens ein Kanal des Eintrages einen Mindestanteil der Gesamtfläche besitzt. Somit werden Entscheidungen auf Basis von geringer Evidenz unabhängig von der Gesamtanzahl der Einträge vermieden.

Alg. (3.7) beschreibt das Verfahren. Die lokal kumulierte Frequenztabelle



(a) (b)

Abbildung 3.5.3: (a): Unterschied zwischen einer schwellwertbasierten und mittels Frequenztabelle realisierten Entscheidungsfunktion. (b): Frequenztabelle zur Berechnung von θ (blaue Linie) mit Alg. (3.8).

kann mit Alg. (3.6) wie besprochen auf ein bestimmtes Signifikanzniveau skaliert werden.

3.5.2 Bestimmung des Schwellwerts

Eine weit verbreitete Entscheidungsfunktion ist zum Beispiel die lineare Diskriminantenfunktion [35], die auf einem einzelnen Schwellwert basiert. Der Vorteil dieser Entscheidungsfunktion ist der geringere Speicherverbrauch. Die Entscheidungsfunktion mittels Frequenztabellen bietet jedoch eine bessere Anpassung an die Daten und vor allem einen Konfidenzwert für jede Entscheidung, welche die Sicherheit der Entscheidung quantifiziert.

Abb. 3.5.3(a) zeigt den Unterschied zwischen den Entscheidungsfunktionen und 3.5.3(b) zeigt eine Frequenztabelle des ersten Stufenklassifikators zur Bestimmung des Schwellwerts θ (blaue Linie) mit einem spezifischen Signifikanzniveau. Die Entscheidungsfunktion klassifiziert alle projizierten Werte rechts des Schwellwertes als Objekte, links als Hintergrund. Es muss lediglich der Schwellwert gespeichert werden, wohingegen bei der Frequenztabelle die gesamte Tabelle abgespeichert werden muss. Die Entscheidungsfunktion lautet:

$$h(\underline{x}) = \begin{cases} \omega_1 & \text{falls } \underline{w}^T \underline{x} + \theta \geq 0 \\ \omega_0 & \text{sonst} \end{cases} \tag{3.5.1}$$

Alg. (3.8) beschreibt die Berechnung des Schwellwerts θ mit einer Frequenz-

Algorithmus 3.8 Trainiere Schwellwert θ mit Signifikanzniveau

1. **Gegeben**: Frequenztabelle f mit b Einträgen, Signifikanzniveau α
2. Berechnung der mimimalen und maximalen Projektionswerte:
   $s_{\max} = \max_{\underline{x}_i \in \mathcal{D}} (\underline{w}^T \underline{x}_i)$, $s_{\min} = \min_{\underline{x}_i \in \mathcal{D}} (\underline{w}^T \underline{x}_i)$
3. $l(\omega_1) = 0$, **iteriere** über die Einträge f_i:
   (a) $l(\omega_1) \leftarrow l(\omega_1) + f_i(\omega_1)$ (Summierte FNs)
      i. Falls $l_t(\omega_1) > \alpha$: $\theta = s_{\min} + \frac{i-1}{b-1}(s_{\max} - s_{\min})$, stop.
      ii. Sonst gehe zum nächsten Eintrag.

tabelle. Da in Schritt 3a die Falsch-Negativen summiert werden, wird der Schwellwert umso genauer gesetzt, je mehr Einträge verwendet werden. In der Praxis wird $b > 1000$ gewählt. Der Algorithmus ist ebenfalls linear in der Anzahl der Einträge, $\mathcal{O}(b)$.

Das Standardverfahren zum Setzen eines Schwellwertes ist die Minimierung des Fehlers. Dazu werden für jeden Eintrag die akkumulierten Fehlklassifikationen berechnet und der Schwellwert an der Stelle gesetzt, an der der minimale Fehler erreicht wird.

Alg. 3.9 zeigt den Algorithmus zur Bestimmung des Schwellwertes bei minimalem Fehler. Die Verteilungen müssen dabei auf die Fläche eins normiert sein. In Schritt 3a werden die kumulierten Häufigkeiten der Klassen bestimmt, die dann zur Bestimmung des Eintrags mit minimalem Fehler in Schritt 4 verwendet werden. Der Schwellwert $\theta_{\min}$ wird dann über lineare Interpolation bestimmt.

3.6 Zusammenfassung

In diesem Kapitel wurden die isolierten merkmalsbasierten Klassifikationsverfahren beschrieben, welche in dieser Arbeit Verwendung finden. Die merkmalsbasierte Klassifikation bezieht sich hier auf Bildmerkmale, die für einen isolierten Klassifikator als Klassifikationseinheit verwendet werden. Im Gegensatz dazu werden in Kapitel 6 Ensmbleklassifikatoren beschrieben, die aus einzelnen Basisklassifikatoren zusammengesetzt sind, welche den in diesem Kapitel beschriebenen Klassifikatoren entsprechen. Die Merkmalsberechnung

Algorithmus 3.9 Trainiere Schwellwert θ mit minmalem Fehler

1. **Gegeben**: Frequenztabelle f mit b Einträgen und normierten Flächen
2. Berechnung der mimimalen und maximalen Projektionswerte:
   $s_{\max} = \max_{\underline{x}_i \in \mathcal{D}} (\underline{w}^T \underline{x}_i)$, $s_{\min} = \min_{\underline{x}_i \in \mathcal{D}} (\underline{w}^T \underline{x}_i)$
3. $e_i = 0$, **iteriere** über die Einträge f_i:
   (a) $e_i(\omega_0) = e_{i-1}(\omega_0) + f_i(\omega_0)$ (Akkumuliere Negative)
   $e_i(\omega_1) = e_{i-1}(\omega_1) + f_i(\omega_1)$ (Akkumuliere Positive)
4. $m = \arg\min_i (e_i(\omega_1) + (1 - e_i(\omega_0)))$
5. $\theta_{\min} = s_{\min} + \frac{m-1}{b-1}(s_{\max} - s_{\min})$

wurde in eine integralbildbasierte Berechnung unterschiedlicher Merkmalstypen, sowie in eine geometrische Repräsentation und Selektion der Merkmale unterteilt. Außerdem wurden Voting-Verfahren zur Merkmalsberechnung vorgestellt, worunter sich z. B. die HOG-Merkmale befinden. Das in diesem Kapitel vorgestellte allgemeine Verfahren lässt sich im Gegensatz dazu auf beliebige Merkmalstypen und Extraktionsregionen anwenden. Zur Normierung der Merkmale wurden ebenfalls unterschiedliche Methoden vorgestellt. Im Abschnitt zur Merkmalstransformation wurden die Fisher's lineare Diskriminante und die multiple lineare Regression als Projektionsverfahren vorgestellt, die einen mehrdimensionalen Merkmalsvektor auf einen skalaren Wert abbilden. Anhand dieses Wertes wurden dann im Abschnitt "Klassenzuordnung" Verfahren beschrieben, die eine Zuordnung zur entsprechenden Klasse ermöglichen. Hierbei wurde ein neues Verfahren beschrieben, mit dem sich ein spezifisches Signifikanzniveau aller Entscheidungen auf dem Trainingsdatensatz einstellen lässt, indem die Entscheidungsfunktion der Hintergrundklasse mit einer Korrekturwahrscheinlichkeit geeignet skaliert wird. Dieses Verfahren ist für die Kaskadenklassifikation mit einer höheren Anzahl an Stufenklassifikatoren unerlässlich, da sich die Detektionsraten durch die Form der Kaskadenklassifikationsregel multiplizieren. Das vorgestellte Verfahren ist linear in der Größe der Frequenztabelle berechenbar und somit auch effizient mit einer kombinatorischen Optimierung anwendbar.

Kapitel 4

Adaptive Ressourcenoptimierung kaskadierter Klassifikationssysteme

Die merkmalsbasierten Klassifikatoren, die im letzten Kapitel beschrieben wurden, bilden elementare Klassifikationseinheiten, die entweder direkt als Stufenklassifikatoren im Kaskadenklassifikator verwendet werden können oder aus denen komplexere Ensembleklassifikatoren gebildet werden können. Die unterschiedlichen Ensembleklassifikatoren werden in Kapitel 6 besprochen. Bei der sequentiellen Struktur des Kaskadenklassifikators nimmt jedoch sowohl die Berechnungskomplexität, als auch die Klassifikationsperformance der einzelnen Klassifikatoren, starken Einfluss auf die Gesamtperformance des Klassifikators. Bei den in der Literatur beschriebenen Boostingverfahren ist die Klassifikatorstruktur, also die Anzahl der schwachen Klassifikatoren für den starken Stufenklassifikator, jedoch ein Designkriterium, das vom Anwender festgelegt werden muss.

In diesem Kapitel wird ein neues, statistisch basiertes Optimierungsverfahren vorgestellt, das eine adaptive Regulierung der Ausprägung der Stufenklassifikatoren hinsichtlich der Berechnungskomplexität und der Klassifikationsleistung ermöglicht. Dadurch ist es möglich, lineare Stufenklassifikatoren zu verwenden, die sich strukturell an die Teilprobleme anpassen und im Gegensatz zu den geboosteten Klassifikatoren eine geringere Berechnungskomplexität besitzen. Diese ermöglichen eine Echtzeitanwendung der Detektoren. Außerdem kann damit die Anzahl der Basisklassifikatoren bei Boostingverfahren auf das entsprechende Klassifikationsproblem in einer Stufe angepasst werden. Das multikriterielle Optimierungsproblem wird durch eine Zielfunktion, die einen Kompromiss zwischen den zwei Kriterieneiner möglichst geringen mittleren Laufzeit und eines geringen Klassifikationsfehlers ermöglicht, und welche mit einer kombinatorischen Optimierung angewendet wird, gelöst. Aus diesem Mechanismus resultiert ein deutlicher Laufzeitgewinn und damit eine Energieersparnis [107]*, wodurch das Verfahren für Echtzeitanwendungen geeignet ist. Das Optimierungsverfahren kann außerdem mit beliebigen Ensemble-Verfahren kombiniert werden. Des Weiteren wird in diesem Kapitel ein neues Verfahren vorgestellt, das eine hierarchische Unterteilung des Kaskadenklassifikators in Unterkaskaden aus einer bestimmten Auflösungsstufe beschreibt, indem die entsprechenden Merkmale aus Gitterstrukturen mit identischen Unterteilungen beschränkt werden. Im Gegensatz zu den in der Literatur beschriebenen Bildpyramiden wird die steigende Auflösung durch die Struktur der Merkmale erreicht, deren Rechteckbereiche aus einer immer feiner aufgelösten Gitterstruktur mit einem kombinatorischen Optimierungsverfahren ermittelt werden. Diese hierarchische Unterteilung ermöglicht sowohl eine effektive Berechnung der kombinatorischen Optimierung durch

die resultierende Suchraumeinschränkung, sowie eine laufzeiteffiziente Echtzeitanwendung bei der Ausführung der Detektion, für die der entsprechende hierarchische, auflösungssensitive Detektionsalgorithmus in diesem Kapitel vorgestellt wird.

Teile dieses Kapitels und die entsprechenden Verfahren wurden bereits in der Publikation [107]* dargestellt.

4.1 Kombinatorische Ressourcenoptimierung

Das grundlegende Optimierungsverfahren ist die kombinatorische Optimierung:

$$(\mathcal{H}, C) \tag{4.1.1}$$

wobei $\mathcal{H}$ die Menge aller möglichen Lösungen bildet, also die Menge der Klassifikatorkandidaten. Jedem Kandidat $h \in \mathcal{H}$ wird als Merkmalsmenge eine spezifische Rechteckkombination mit Extraktionstypen zugewiesen (siehe Abschn. 3.1 und 3.1.2). Die Optimierungsfunktion C ist die Abbildung $C : \mathcal{H} \to \mathbb{R}$, die jedem Kandidaten einen Wert zuordnet, der einen Kompromiss aus Laufzeit und Klassifikationsleistung, abhängig von der Komplexität des Teilproblems in der jeweiligen Stufe, darstellt. Die Laufzeit lässt sich durch die Elementaroperationen pro Klassifikation abschätzen oder durch eine Zeitmessung ermitteln. Damit können Klassifikatoren mit unterschiedlichen Laufzeit- und Klassifikationseigenschaften unterschieden und mit einer globalen Optimierungsstrategie bestimmt werden.

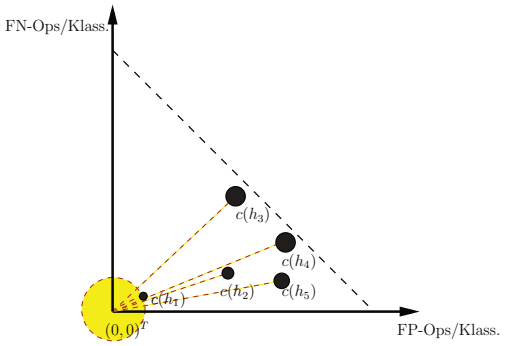


Abbildung 4.1.1: Darstellung der Kandidatenklassifikatoren als Teilchen mit den Eigenschaften Klassifikations-Leistung und mittlere Operationen pro Klassifikation.

Abb. 4.1.1 stellt die Analogie der Elementarklassifikatoren zu einem physikalischen Teilchen her. Die Teilcheneigenschaften sind die Operationen pro Klassifikation und die erwarteten Fehlklassifikationen, woraus mit einer multiplikativen Verknüpfung ein Maß für die *erwarteten Fehloperationen* resultiert.

Ein weiterer wichtiger Aspekt, der dieses komplexe kombinatorische Verfahren für die Klassifikatoroptimierung erst ermöglicht und zusätzlich sowohl beim Training, als auch bei der Anwendung, sehr schnell macht, ist die bereits besprochene hierarchische, gitterbasierte Einteilung der Merkmale, die eine drastische Reduktion der Kombinationen und damit der Kandidatenklassifikatoren von 18504486 auf 630 zur Folge hat (siehe Tabelle 3.1.1). Der Kaskadenklassifikator besteht also aus auflösungsabhängigen Stufenklassifikatoren, die aus Merkmalen einer bestimmten Auflösungsstufe durch die Gitterstruktur stammen. Für jeden Klassifikatorkandidaten wird ein Kostenwert $C(h)$ berechnet, der sowohl die mittlere Laufzeit pro Detektionsfenster $c(h)$, als auch die wahrscheinlichen Fehloperationen (Alg. (3.5)) bezüglich der A Posteriori Wahrscheinlichkeiten der Fehlklassifikationen beeinhaltet:

$$e(h) = P(\omega_1|h(\underline{z}) = \omega_0) + P(\omega_0|h(\underline{z}) = \omega_1) \tag{4.1.2}$$

Somit wird erreicht, dass ein von der Komplexität der Klassifikationsaufgabe abhängiger Kompromiss zwischen Klassifikationsleistung und Laufzeit zur Bestimmung des Stufenklassifikators herangezogen wird. Bei der Kostenfunktion werden die mittleren Laufzeitkosten durch unterschiedliche Funktionen $\nu(h)$ gewichtet

$$C(h) = e(h)c(h)^{\nu(h)} \tag{4.1.3}$$

Die Einheit dieser Funktion sind die erwarteten gewichteten Fehloperationen. Diese Gewichtung ist nötig um sich auf die unterschiedlich komplexen Teilprobleme in den Stufen anzupassen und dadurch bei komplexeren Teilproblemen höherdimensionale Merkmalsvektoren zu bestimmen. Für die Gewichtung werden folgende Funktionen verwendet:

$$\nu_1(h) = 0 \tag{4.1.4}$$

$$\nu_2(h) = 1 \tag{4.1.5}$$

$$\nu_3(h) = \frac{1}{s} \tag{4.1.6}$$

$$\nu_4(h) = P(\omega_1|h_T(\underline{z}) = \omega_0) \tag{4.1.7}$$

Das Verfahren läßt sich nicht nur auf Klassifikatoren, sondern auf beliebige Eigenschaften einer Funktion h anwenden, indem die entsprechenden Kosten $c(h)$ eingesetzt werden und ein Wahrscheinlichkeitsmodell aufgestellt wird, mit dem die Fehlerwahrscheinlichkeiten $e(h)$ geschätzt werden können.

Algorithmus 4.1 Ressourcenoptimiertes Training des merkmalsbasierten Klassifikators

1. **Gegeben**:
   (a) Für das Teilproblem repräsentative Trainingsmenge $\mathcal{D}$
   (b) Merkmalsvektor $\underline{v}$
   (c) Menge an Merkmalen $\mathcal{F}_{\Delta r}$ (Abschn. 3.1) in Auflösungsstufe Δr
   (d) Zielfehler $e(\omega_0) = P_{\max}(h(\underline{z}) = \omega_1|\omega_0)$
2. Initialisiere Lösungsmenge $\mathcal{H} = \emptyset$.
3. Für jede Kombination $\underline{m} \in \mathcal{F}_{\Delta r}$, trainiere h_m mit Merkmalen $\underline{v} \oplus \underline{m}$:
   (a) Bestimme die Projektionsachse $\underline{w}_m$ von h_m (Alg. (3.2) $\vee$ (3.3))
   (b) Bestimme die Entscheidungsfunktion von h_m (Alg. (3.4), (3.6) $\vee$ (3.8))
   (c) Bestimme die Kosten des Klassifikators $C_t(h_m)$ (Alg. (3.5), Gl. (4.1.3))
   (d) h_m wird neuer Kandidat: $\mathcal{H} \leftarrow \mathcal{H} \cup h_m$
4. Bestimme den Kandidaten h_{opt} bezüglich der vorgegebenen Fehlerwahrscheinlichkeit $e(\omega_0)$ (FPR) und der Berechnungskosten $C(h)$:
   (a) $e_{\text{sup}} = \max(e(\omega_0), \min(\{P(h(\underline{z}) = \omega_1|\omega_0) \,|\, h \in \mathcal{H}\}))$
   (b) $h_{\text{opt}} = \underset{C(h)}{\arg\min} \{h \in \mathcal{H} \,|\, P(h(\underline{z}) = \omega_1|\omega_0) \,|\, \omega_0) \leq e_{\text{sup}}\}$

Mit Gl. (4.1.4) werden ausschließlich die Klassifikationsfehler berücksichtigt. Mit Gl. (4.1.6) werden die Laufzeitkosten, je höher die Stufe s ist, herunter gewichtet. Bei Gl. (4.1.7) werden die Kosten anhand der Falsch-Positiv-Rate des bereits existierenden Kaskadenklassifikators gewichtet. Je kleiner die Auftrittswahrscheinlichkeit eines negativen Beispiels in der entsprechen-

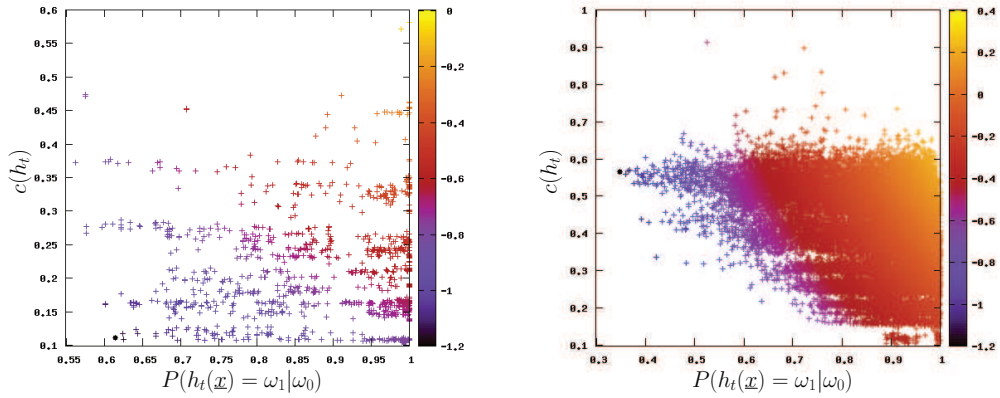
den Stufe, desto geringer wird die Laufzeit des Klassifikators gewichtet.

Alg. (4.1) beschreibt das Training der Klassifikatorkandidaten, welche die potentiellen Lösungen für das Klassifikationsproblem in der jeweiligen Kaskadenstufe bilden. Auf die Bestimmung der repräsentativen Trainingsmenge in Schritt 1a wird im folgenden Kapitel eingegangen. Der bestehende Merkmalsvektor in Schritt 1b wird mit dem Nullvektor initiiert. In Schritt 3 wird der bestehende Merkmalsvektor $\underline{v}$ mit dem Merkmalsvektor $\underline{m}$ durch die Operation $\oplus$ aneinandergehängt und dem Klassifikator h_m zugewiesen, der dann durch Bestimmung der Gewichte und der Entscheidungsfunktion trainiert wird. Die Zielfunktion wird in Schritt 3c ausgewertet und zugewiesen.

Das Ergebnis des Alg. (4.1) ist der optimale Klassifikatorkandidat h_{opt}. Ist der Zielfehler auf $e(\omega_0) = 1$ eingestellt, so berechnet sich h_{opt} als Minimum über die Zielfunktion mittels Gl. (4.1.3). Ansonsten wird in Schritt 4b das Minimum aus einer Untermenge gebildet, die kleiner als die obere Fehlerschranke e_{sup} ist, indem die Kandidaten aufsteigend nach dem Kostenkriterium sortiert werden. Falls kein Klassifikator die Fehlerschranke unterschreitet, wird der Klassifikator mit minimalem Fehler $\min\{P(h(\underline{z}) = \omega_1)|h \in \mathcal{H}\}$ gewählt.

Der Vorteil bei einer Fehlerschranke $e(\omega_0) < 1$ ist, dass bei komplexen Teilproblemen die Klassifikatoren mit minimalem Fehler unter den Kandidaten garantiert gewählt werden. Bei diffizilen Klassifikationsproblemen kann durch eine geringe Fehlerschranke eine stärkere Gewichtung der Klassifikationsleistung erreicht werden. Die Laufzeit des Algorithmus ist abhängig vom Aufwand zum Training eines Klassifikatorkandidaten g und linear in der Anzahl der Merkmalskombinationen, $\mathcal{O}(|\mathcal{F}|g)$.

Abb. 4.1.2 zeigt eine Visualisierung der erwarteten Fehloperationen $C(h)$ mit Gl. (4.1.3). Jeder farbige Punkt repräsentiert einen Klassifikator, dessen Farbe die erwarteten Fehloperationen darstellt. Der optimale Klassifikator ist durch einen schwarzen Punkt dargestellt. In Abb. 4.1.2(a) sind die Kandidaten in Kaskadenstufe 3 und Auflösungsstufe $\Delta r = \frac{1}{3}$ dargestellt. Abb. 4.1.2(b) zeigt die Kandidaten in Kaskadenstufe 12 und Auflösungsstufe $\Delta r = \frac{1}{5}$. Durch die höhere Auflösung existieren mehr Klassifikatorkandidaten und deshalb ist eine größere Menge dargestellt. In dieser Stufe wird ein Klassifikator mit guter Detektionsleistung und erhöhtem Berechnungsaufwand gewählt. Die Kosten für die Gesamtkaskade berechnen sich aus den akkumulierten Auftrittswahrscheinlichkeiten der Einzelklassifikatoren der Objektklasse und deren Berechnungskosten. Hierzu sei Gl. (2.3.21)



(a) (b)

Abbildung 4.1.2: Visualisierung der erwarteten Fehloperationen $C(h)$ (Gl. (4.1.3)) anhand einer Farbkodierung mit Auftrittswahrscheinlichkeit der Hintergrundklasse auf der X-Achse und den Berechnungskosten der Kandidatenklassifikatoren auf der Y-Achse.
(a): Kandidaten h bei Kaskadenstufe 3 in Auflösungsstufe $\Delta r = \frac{1}{3}$.
(b): Kandidaten h bei Kaskadenstufe 12 in Auflösungsstufe $\Delta r = \frac{1}{5}$.

wiederholt:

$$C(\omega_i|h_T(\underline{z}) = \omega_1) = C_0 + P_0(\omega_i)\, c(h_1) + \sum_{t=2}^{T} P(\omega_i|h_{t-1}(\underline{z}) = \omega_1)\, c(h_t)$$

Hierbei ist $P(\omega_i|h_{t-1}(\underline{z}) = \omega_1)$ die A-Posteriori-Wahrscheinlichkeit der Teilkaskade mit Klassifikatoren $h_1, \ldots, h_{t-1}$. Mit dieser Wahrscheinlichkeit treten die Kosten des Stufenklassifikators $c(h_t)$ auf. Die Konstante C_0 stellt die initialen Berechnungskosten dar. Abb. 4.1.3 zeigt die sinkende Fehlerwahrscheinlichkeit des Gesamtklassifikators durch Hinzunahme weiterer Stufenklassifikatoren, die farblich unterschiedlich gekennzeichnet sind, bis der Fehler nicht mehr weiter herabgesenkt werden kann. Die Fehloperationen steigen streng monoton.

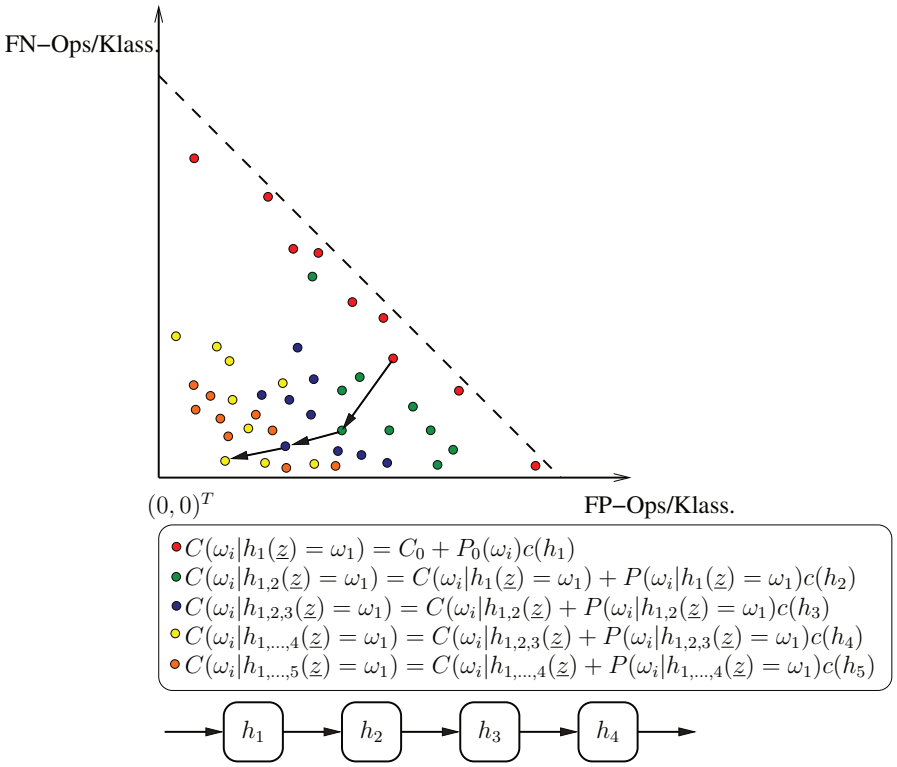


Abbildung 4.1.3: Entwicklung der Fehlerwahrscheinlichkeit des Kaskadenklassifikators durch Hinzunahme von Stufenklassifikatoren h_t, die farblich unterschiedlich gekennzeichnet sind. Die Anzahl der Fehloperationen steigt streng monoton.

4.2 Ressourcenoptimierte, sequentielle Merkmalsauswahl

Es hat sich jedoch gezeigt, dass auch mit den höherdimensionalen Maskenmerkmalen ohne einen zusätzlichen Mechanismus zu hohe Falsch-Positiv-Raten der Kaskade resultieren. Deshalb wird der kombinatorische Optimierungsalgorithmus 4.1 mit einer sequentiellen Vorwärtsauswahl von Merkmalen kombiniert (Sequential Forward Feature Selection (SFS) [118]). Die Vorwärtsauswahl bietet sich bei der Struktur des Kaskadenklassifikators an, da zu Beginn ein niedrigdimensionaler Merkmalsvektor bezüglich der Berechnungskosten von Vorteil ist und erst mit höheren Stufen sukzessive Merkma-

Algorithmus 4.2 Training sequentielle Vorwärtsauswahl SFS

1. **Gegeben**:
   (a) Kostengewichtung ν
   (b) Trainingsmuster $\mathcal{D}$
   (c) Merkmale $\mathcal{F}_{\Delta r}$
   (d) Fehlerschwelle $e_{\mathrm{SFS}}(\omega_0) = P_{\max}(h(\underline{z}) = \omega_1|\omega_0)$
   (e) $t_{\max}$, max. Anzahl an Iterationen
2. Initialisiere Merkmalsvektor $\underline{v}_1 = \underline{0}$.
3. **Iteriere** $1 \leq t \leq t_{\max}$:
   (a) Bestimme den optimalen Klassifikator $h_{opt,t}$ unter den Kandidaten mit Alg. (4.1) auf dem aktuellen Merkmalsvektor $\underline{v}_t$ und der Fehlerschwelle $e_{\mathrm{SFS}}(\omega_0)$
      i. Falls $P(h_{\mathrm{opt},t}(\underline{z}) = \omega_1|\omega_0) \leq e_{\mathrm{SFS}}(\omega_0)$: h_{opt} gefunden, stop.
      ii. Sonst gehe zum nächsten Schleifendurchgang.

le hinzugenommen werden. Bei der Rückwärtsauswahl (Sequential Backward Elimination SBE) wird mit einer großen Menge an Merkmalen begonnen. Merkmale werden entfernt bis ein spezielles Kriterium erfüllt ist. Eine ausführliche Beschreibung der Algorithmen zur Merkmalsauswahl ist in Artikel [94] gegeben. Dort erzielen die Methoden zur Vorwärtsauswahl nahezu identische Klassifikationsraten, wobei deutlich weniger Merkmale im Gegensatz zur Rückwärtsauswahl verwendet werden.

Alg. (4.2) ist ein Greedy Algorithmus und beschreibt das Training des optimalen Klassifikators durch sukzessives Hinzufügen aller Merkmalskombinationen an die bestehende Merkmalskombination des optimalen Klassifikators $h_{\mathrm{opt},t}$ im Zyklus t, indem in Schritt 3a Algorithmus 4.1 auf der aktuellen Merkmalskombination aufgerufen wird. Somit wird die Fehlerrate durch Hinzunahme der besten Merkmalskombination mit dem bestehenden Merkmalsvektor reduziert. Die Hinzunahme weiterer Merkmale lässt zugleich die mittlere Laufzeit des Klassifikators ansteigen. Allerdings wird das Aufstocken der Merkmale, abhängig vom spezifischen Klassifikationsproblem, im

Allgemeinen erst nach mehreren Stufen nötig. Falls sich die Fehlerrate durch Hinzunahme eines Merkmales verschlechtert, so wird derjenige Klassifikator $h_{\text{opt},t}$ mit minimalem Fehler gewählt. Die Parameter, die für Alg. (4.2) benötigt werden, sind in Tabelle 4.2.1 gelistet.

Parameter Alg. (4.2)			
Max. Fehler $e_{\text{SFS}}(\omega_0)$	Max. Iterationen $t_{\max}$	Merkmale $\mathcal{F}_{\Delta r}$	Kosten ν

Tabelle 4.2.1: Auflistung der Parameter von Alg. (4.2)

In Abb. 4.1(a) sind die geometrischen Ausprägungen der Merkmale für die Zyklen mit den Nummern 0 bis 3 durch farbige Rechtecke des Klassifikators h_{opt} für die Stufe $s = 20$ in der Auflösung $\Delta r = \frac{1}{5}$ dargestellt. Bei Abb. 4.1(b) wurden in Stufe $s = 26$ bei der höheren Auflösung $\Delta r = \frac{1}{6}$ Maskenmerkmale verwendet, da sonst die Merkmalsmenge zu hoch wird. In jedem Zyklus wird ein neues Maskenmerkmal mit einer spezifischen Geometrie hinzugenommen, sodass ein Großteil der Fläche des Detektionsfensters abgedeckt ist. In Abb. 4.1(c) sind die Gewichte des Merkmalsvektors mit den entsprechenden Rechteckgeometrien aus Abb. 4.1(a), $\underline{v}_t$ in jeder Zeile durch Balken dargestellt. Man erkennt, dass die Merkmale, die im letzten Zyklus $t = 3$ gewählt wurden, hohe Gewichte besitzen. Somit können auch Merkmale, die in späteren Zyklen gewählt werden, einen hohen Einfluss besitzen. Die Dimension des Vektors ist $\dim(\underline{v}_t) = 17$. In Abb. 4.1(d) sind die Gewichte bezüglich der Abb. 4.1(b) dargestellt. Durch die Maskenmerkmale und 4 Zyklen resultiert ein hochdimensionaler Merkmalsvektor mit $\dim(\underline{v}_t) = 151$. Das Maskenmerkmal, das im ersten Zyklus gewählt wurde, und sich auf den rechten Bereich des Detektionsfensters fokusiert, ist sinnvoll, da Speed-Limit-Schilder zwischen 20 und 70 mph in 5-er Schritten die Trainingsmenge der Objektklasse bilden und somit auf der rechten Seite des Ziffernpaares nur entweder die Ziffer 5 oder 0 auftritt.

4.3 Auflösungsspezifisches Kaskadentraining

Alg. (4.4) beschreibt das Bayes'sche Kaskadentraining mit Merkmalen aus einer bestimmten Auflösungsstufe $\mathcal{F}_{\Delta r}$. Das Prinzip ist, dass die A-priori-Wahrscheinlichkeiten für jede Stufe aus den A-posteriori-Wahrscheinlichkeiten der vorherigen Stufe angepasst werden (siehe Abb.

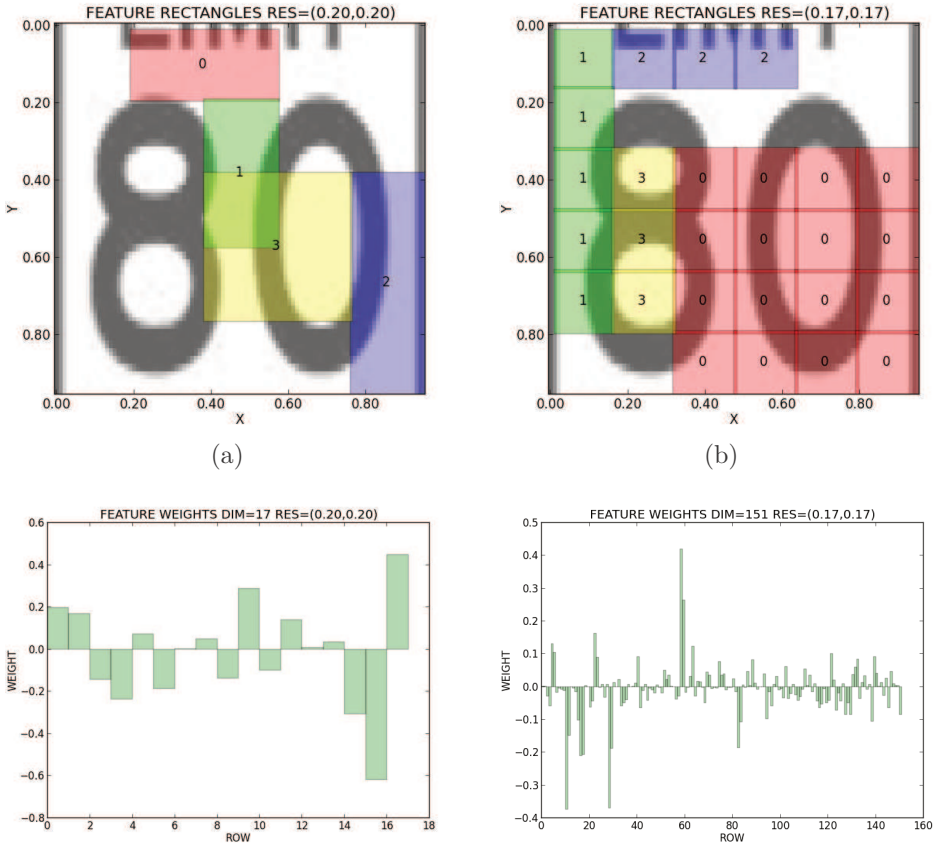


Abbildung 4.2.1: Geometrische Ausprägung der Merkmale bei SFS (Alg. (4.2))
(a): $\Delta r = \frac{1}{5}, t_{\max} = 4$, (b): $\Delta r = \frac{1}{6}, t_{\max} = 4$
Gewichte der entsprechenden Merkmalsvektoren.
(c): $\dim(\underline{v}_t) = 17$, (d): $\dim(\underline{v}_t) = 151$

2.3.4) und für die Bayes-Entscheidungsfunktion (Gl. 2.3.12):

$$h_s(\underline{x}) = \begin{cases} \omega_1 & \text{falls } g(\underline{x}) := P_s(\omega_1)P_s(\underline{x}|\omega_1) - P_s(\omega_0)P_s(\underline{x}|\omega_0) \geq 0 \\ \omega_0 & \text{sonst} \end{cases}$$

oder zum Bestimmen des Schwellwertes verwendet werden. Somit handelt es sich um eine Maximum A-Posteriori-Entscheidungsregel (MAP). Die Formel

Algorithmus 4.3 Rekursive Aktualisierung der A-priori-Wahrscheinlichkeiten

1. **Gegeben:** $P_s(\omega_i)$, $P_{\min}(\omega_i)$, $i \in \{0, 1\}$
2. Berechne $P_{s+1}(\omega_i|h(\underline{z}))$ (Gl. (4.3.1) mit Alg. (3.5)).
3. $P_{s+1}(\omega_i|h(\underline{z})) = \max(P_{s+1}(\omega_i|h(\underline{z}), P_{\min}(\omega_i))$.

zur Berechnung der neuen A-priori-Wahrscheinlichkeit für die Stufe $s + 1$ lautet:

$$P_{s+1}(\omega_i|h(\underline{z}) = \omega_j) = P_s(\omega_i)P_s(h(\underline{z}) = \omega_j \,|\, \omega_i) \;= P_s(\omega_i)\frac{\int_{H_j} P_s(x_s \,|\, \omega_i)\mathrm{d}x_s}{\int_{Y_i} P_s(x_s \,|\, \omega_i)\mathrm{d}x_s} \tag{4.3.1}$$

mit

$$H_j = \{x_s \,|\, h_s(\underline{z}) = \omega_j\}$$
$$Y_i = \{x_s \,|\, y(\underline{z}) = \omega_i\}$$

für $0 \leq i, j < 1$, wobei $x_s = \underline{w}_s^T \underline{z}$ die Projektionen der Zufallsvariablen in den Stufen entsprechen und H_j die Menge der geschätzen Entscheidungsregionen für die Klasse ω_j ist. Y_i entspricht hingegen der Menge der tatsächlichen Entscheidungsregionen für die Klasse ω_i. Der Bruch in Gl. (4.3.1) stellt somit die Wahrscheinlichkeiten der vier möglichen Entscheidungszustände (siehe Abb. 2.2(a) und 2.3(b)) für die Indexkombinationen $0 \leq i, j < 1$ dar. Außerdem wird die A-posteriori-Wahrscheinlichkeit des Gesamtklassifikators zur Kostengewichtung in Gl. (4.1.7) verwendet. Da die A-priori-Wahrscheinlichkeiten der Hintergrundklasse bei der rekursiven Berechnung mit Gl. (4.3.1) sehr schnell gegen Null konvergieren und damit keine Negativ-Entscheidung mehr möglich wäre, werden zusätzlich minimale Schwellen für die Entscheidungsfunktion verwendet. Die rekursive Aktualisierung der A-priori-Wahrscheinlichkeiten ist in Alg. (4.3) dargestellt.

Die genaue Bestimmung der repräsentativen Beispiele der Teilprobleme in Schritt 2a von Alg. (4.4) wird im folgenden Kapitel näher beschrieben. Das Standardvorgehen ist, dass alle positiven Muster als Objektklasse und Falsch-Positiv-Detektionen mit dem aktuellen Kaskadenklassifikator als Hintergrundklasse gewählt werden, da es die Entscheidungsfunktion des Kaskadenklassifikators (Gl. (2.3.8)) nahe legt. Abb. 4.3.1 zeigt das Prinzip, wo-

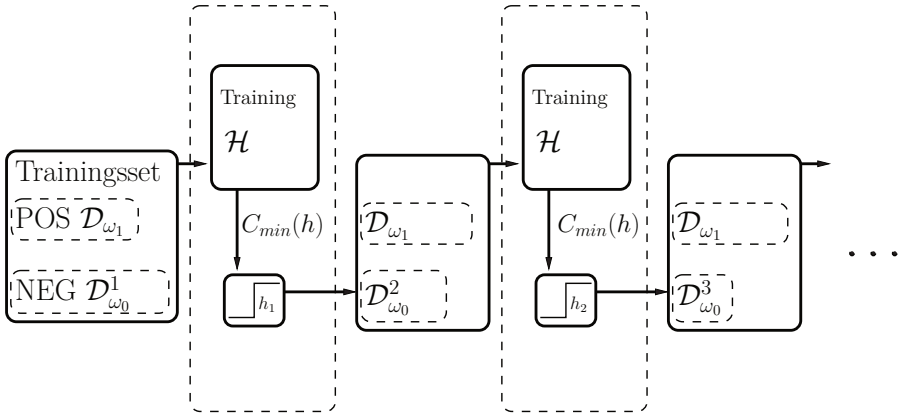


Abbildung 4.3.1: Auswahl der Trainingsbeispiele und Optimierung der Stufenklassifikatoren.

bei die Größe der Objektklasse identisch bleibt und als Negativbeispiele nur Falsch-Positive verwendet werden.

Die Gewichte und Entscheidungsfunktionen können dann mit den in Abschnitt 3.4 und 3.5 beschriebenen Algorithmen im Schritt 2b auf den stufenspezifischen Teilproblemen bestimmt werden. Als Abbruchkriterium des Algorithmus dient die Auftrittswahrscheinlichkeit der Hintergrundklasse. Falls diese den Schwellwert $e(\omega_0)$ übersteigt, ist die Trennfähigkeit der Klassen mit den Merkmalen aus der aktuellen Auflösungsstufe nicht mehr ausreichend und es muss in eine höhere Auflösungsstufe gewechselt werden. In Schritt 2c kann die Fehlerwahrscheinlichkeit der Objektklasse zusätzlich verwendet werden, falls kein Signifikanzniveau für die Stufenklassifikatoren eingestellt wird. Das Prinzip des auflösungsspezifischen Kaskadentrainings ist die hierarchische Aufteilung der Merkmale durch die beschriebenen Gitterstrukturen. Durch die Frequenzeigenschaften der Boxfilter sind die Merkmale mit geringer Auflösung örtlich stabiler bei Verschiebungen. Deshalb wird beim Training mit einer niedrigen Auflösung begonnen, und falls keine ausreichende Trenneigenschaft der Stufenklassifikatoren gegeben ist, wird in die nächsthöhere Auflösungsstufe gesprungen. Somit besteht der Kaskadenklassifikator aus Teilkaskaden mit steigenden Aufösungen der Merkmale, was in Abb. 4.3.2 zu sehen ist.

Man kann nun schlussfolgern, daß je wahrscheinlicher ein Objekt in den höheren Stufen vorliegt, desto feiner sind die Merkmale des Klassifikators.

Algorithmus 4.4 Bayes'sches Kaskadentraining

1. **Gegeben**:
   (a) Trainingsbeispiele $\mathcal{D}$
   (b) Merkmale $\mathcal{F}_{\Delta r}$
   (c) Fehlerwahrscheinlichkeit $e_{\text{bay}}(\omega_0)$ als Stop-Kriterium
   (d) A-priori-Wahrscheinlichkeiten $P_s(\omega_1)$ und $P_s(\omega_0)$, abhängig vom
   (e) Kaskadenklassifikator h_S
2. **Iteriere** über die Stufen s
   (a) Bestimme eine repräsentative Trainingsmenge $\mathcal{D}_s$ mit $h_{0,\ldots,s-1}$.
   (b) Trainiere den Stufenklassifikator h_s auf $\mathcal{D}_s$ und $\mathcal{F}_{\Delta r}$.
   (c) Falls $P(h_s(\underline{z}) = \omega_1|\omega_0) < e_{\text{bay}}(\omega_0)$:
      i. Update der Wahrscheinlichkeit $P_s(\omega_i) = P_{s-1}(\omega_i)P(h_s(\underline{z}) = \omega_1|\omega_i)$ (Gl. (4.3.1))
      ii. $h_S = h_{0,\ldots,s-1} \oplus h_s$
   (d) Sonst: Stop.

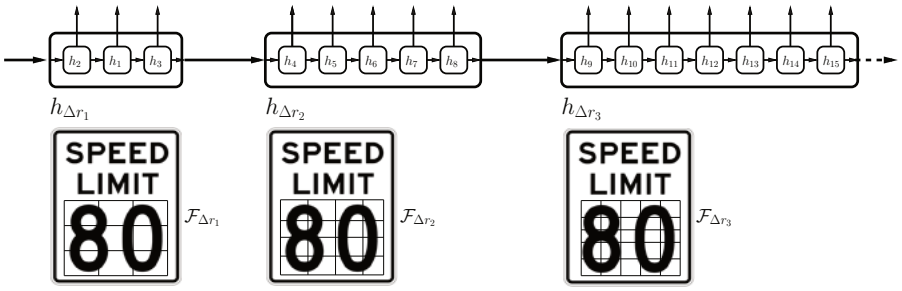


Abbildung 4.3.2: Klassifikator bestehend aus Teilkaskaden mit Merkmalen steigender Auflösung.

Dieser Mechanismus stellt auch eine Laufzeitoptimierung dar, da nur für Beispiele, die mit hoher Wahrscheinlichkeit zur Objektklasse gehören, ein höherer Berechnungsaufwand durch die feineren Auflösungen betrieben wird. Alg. (4.5) beschreibt das auflösunsspezifische Kaskadentraining. Als Einga-

Algorithmus 4.5 Auflösungsspezifisches Kaskadentraining

1. **Gegeben**:
   (a) Trainingsmenge an Bildern $\mathcal{B}$ mit Objektinformation
   (b) Auflösungen $\Delta r_1, \ldots, \Delta r_{\max}$ mit Merkmalen $\{\mathcal{F}_{\Delta r_1}, \ldots \mathcal{F}_{\Delta r_{\max}}\}$
   (c) Fehlerwahrscheinlichkeit $e_{\text{stop}}(\omega_0)$ als Stop-Kriterium
2. Initialisiere A-priori-Wahrscheinlichkeiten $P_0(\omega_i)$ (Gl. (2.3.13))
3. **Iteriere** über die Auflösungen $\Delta r_j \in \{\Delta r_1, \ldots, \Delta r_{\max}\}$
   (a) Generiere eine repräsentative Trainingsmenge $\mathcal{D}_{\Delta r_j}$, indem der Detektionsalgorithmus mit dem aktuellen Klassifikator h_S auf $\mathcal{B}$ aufgerufen wird.
   (b) Trainiere die Teilkaskade $h_{\Delta r_j}$ mit Alg. (4.4) auf $\mathcal{D}_{\Delta r_j}$ und $\mathcal{F}_{\Delta r_j}$ mit $e_{\text{stop}}(\omega_0)$ und $P_s(\omega_i)$
   (c) $h_S = h_{\Delta r_1, \ldots, \Delta r_{j-1}} \oplus h_{\Delta r_j}$
4. **Ausgabe**: Kaskadenklassifikator h_s

Trainingsparameter							
Kaskade H_C			**Alg.**	**Stufe** h_s			**Alg.**
$e_{\text{stop}}(\omega_0)$	$\mathcal{F}_{\Delta r}$	A-priori	4.5, 4.4	$e_{\text{SFS}}(\omega_0)$	$t_{\max}$	ν	4.2, 4.1

Tabelle 4.3.1: Parameter des auflösungsspezifischen Kaskadentrainings, Alg. (4.5) mit Bayes'schem Training der Unterkaskaden, Alg. (4.4) und sequentieller Vorwärtsauswahl zur Merkmalsberechnung, Alg. (4.2) sowie ressourcenoptimiertem Training der merkmalsbasierten Klassifikatoren, Alg. (4.1).

be dient ein Bilddatensatz $\mathcal{B}$ mit annotierter Objektinformation. Außerdem werden die Merkmale für die jeweilige Auflösungsstufe benötigt. In Schritt 3a wird dann aus dieser Menge ein Trainingsdatensatz für die Auflösungsstufe bestimmt, indem der Detektionsalgorithmus, der im nächsten Abschnitt erklärt wird, mit dem aktuellen Kaskadenklassifikator auf den Bildern aufgerufen wird. Die Hintergrundklasse bilden die Falsch-Positiven des Detektionsalgorithmus. Diese werden entweder aus den annotierten Bildern gewonnen, indem Detektionsfenster, die sich mit den Objektfenstern überschneiden, ver-

mieden werden, oder indem ein Bilddatensatz zur Verfügung gestellt wird, der keine Objekte enthält. In Tabelle 4.3.1 sind die Parameter des Alg. (4.4) in Kombination mit dem Algorithmus zur sequentiellen Vorwärtsauswahl der Merkmale gelistet.

4.4 Auflösungsspezifische Detektion

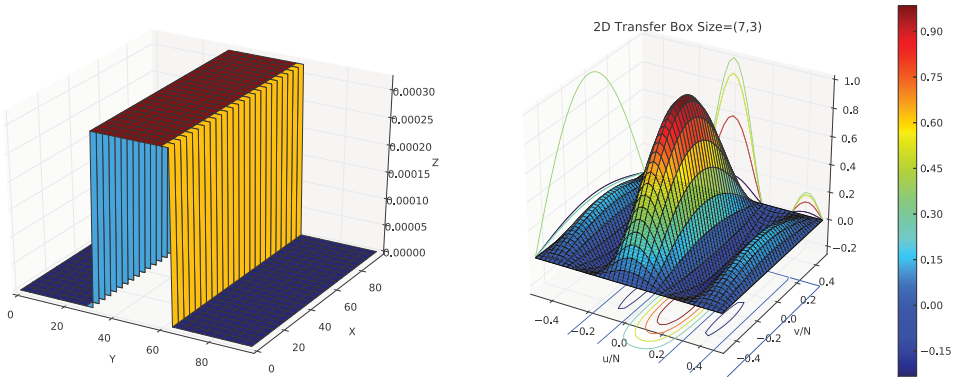
Um die auflösungsspezifische Kaskadenstruktur auch bei der Detektion nutzen zu können werden die Frequenzeigenschaften der Boxfilter genutzt. Eine exakte kanonische Skalenraumrepräsentation eines linearen Skalenraumes [74, 75] ist durch Gaussfilter gegeben [5]. Dennoch werden hier Boxfilter verwendet, da sie, wie in Abschnitt 3.1 beschrieben, in Kombination mit der Integralbilddarstellung eine von der Größe des Detektionsfensters unabhängige, konstante Berechnungskomplexität von vier Integralbildzugriffen besitzen und somit für laufzeitkritische Anwendungen geeignet sind.

Die Filtereigenschaften werden beim Training mitbelernt, indem konsistente Merkmalsrepräsentationen verwendet werden. Ein weiterer großer Vorteil ist, dass sich die Merkmale mittels beliebigen zweidimensionalen Rechteckgeometrien innerhalb der aktuellen Auflöung anwenden lassen. Dadurch können Frequenzeigenschaften von länglichen Objekten auch durch die Filtergeometrie mit konstantem Aufwand verstärkt werden (siehe Abb. 4.4.1).

Abb. 4.1(a) zeigt die Ortsdarstellung eines 2-D länglichen Boxfilters und die dazugehörige Modulationstransferfunktion anhand einer Farbkodierung. Für den Detektionsalgorithmus wird in der ersten Auflösungsstufe Δr_1 begonnen und eine Menge an Detektionsfenstern generiert, in der die entsprechenden Teilkaskaden aufgerufen werden. Die Detektionsfenster werden aus einer Menge an Basisbreiten $\mathcal{W} = \{w_0, \ldots, w_{\max}\}$ berechnet, aus denen sich die unterschiedlichen Fenstergrößen ergeben. Es wird von der Standardabtastung ausgegangen, indem ein Verschiebungsfaktor für die Detektionsfenster, abhängig von der aktuellen Auflösungsstufe und der entsprechenden Basisbreite $w_i \in \mathcal{W}$ berechnet wird:

$$\Delta s_i = \xi \, w_i \, \Delta r_1 \tag{4.4.1}$$

Für $\xi = 1$ entspricht obige Formel der Standardabtastung [65]. Mit $\xi < 1$ wird eine Überabtastung und mit $\xi > 1$ eine Unterabtastung erreicht. Aus der



(a) (b)

Abbildung 4.4.1:
(a): Ortsdarstellung eines 2-D länglichen Boxfilters.
(b): Zugehörige Modulationstransferfunktion.

Algorithmus 4.6 Auflösungsspezifische Depth-First-Detektion

1. **Gegeben:**
   (a) Bild I als geeignete Integralbilddarstellung
   (b) Kaskadenklassifikator $h_C = (h_{\Delta r_1}, \ldots, h_{\Delta r_{\max}})$
   (c) Basisbreiten $\mathcal{W}$
2. Detektionsfenster $\mathcal{R}$ (Gl. (4.4.2)), Ergebnismenge $\mathcal{E} = \emptyset$,
   Stack $\mathcal{S} = \{(d, \Delta r_1) | d \in \mathcal{R}\}$
3. **Solange** Elemente $(d, \Delta r_i) \in \mathcal{S}$ enthalten:
   (a) **Falls** $h_C(d) = \omega_1$: $\mathcal{E} \leftarrow \mathcal{E} \cup d$
   (b) **Falls** $(h_{\Delta r_1}, \ldots, h_{\Delta r_i})(d) = \omega_1$ **und** $h_{\Delta r_j}(d) = \omega_0$ **mit** $j > i$:
   Erweitere Stack $\mathcal{S}$ mit Nachbarn $(d_n, \Delta r_j)$ in einer Umgebung von d wobei $h_{\Delta r_j}(d) = \omega_0$, $j > i$
   (c) **Sonst**: Verwerfe d.

Objektklasse wird das Breite-zu-Höhe Verhältnis b der umschließenden Fenster berechnet. Es können somit in Schritt 3a im Trainingsalgorithmus (4.5)

auch Objekte unterschiedlicher Größe in das Training mit einfließen, ohne dass im Voraus auf Bildausschnitte identischer Größe skaliert werden muss, indem der Detektionsalgorithmus aufgerufen wird. Außerdem werden so skalenspezifische Hintergrundbeispiele der jeweiligen Auflösung ausgewählt, die für die Teilkaskade charakterisitische Frequenzeigenschaften aufweisen. Die Grundmenge an Detektionsfenstern für ein Bild I berechnet sich dann mittels:

$$\mathcal{R} = \{d \in I \,|\, d_x \bmod \Delta s_i = 0 \wedge w(d) = w_i \wedge h(d) = b\, w(d)\,,\, \forall w_i \in \mathcal{W}\} \quad (4.4.2)$$

wobei Δs_i mit Gl. (4.4.1) berechnet wird und $w(d)$ und $h(d)$ die Breite und Höhe des Detektionsfensters $d = (x_1, y_1, x_2, y_2)$ sind.

In Alg. (4.6) ist der rekursive, auflösungsspezifische Depth-First-Detektionsalgorithmus dargestellt, der in einem Bild I mit der auflösungsspezifischen Kaskade $h_C = (h_{\Delta r_1}, \ldots, h_{\Delta r_{\max}})$ und den Basisbreiten $\mathcal{W}$ eine Menge an Objekten $\mathcal{E}$ detektiert. Mittels Gl. (4.4.2) wird in Schritt 2 ein Hilfsstack $\mathcal{S}$, der die Detektionsfenster pontentieller Objektkandidaten und die zugehörige Auflösungsstufe enthält, befüllt. Falls ein Detektionsfenster d als Objekt detektiert wird, indem es alle Stufenklassifikatoren erfolgreich durchläuft, wird es als Objekt in die Menge $\mathcal{E}$ aufgenommen (Schritt 3a).

Falls ein Detektionsfenster d in einer Teilkaskade mit höherer Auflösung verworfen wird, werden in Schritt 3b Nachbarn in einer lokalen Umgebung, abhängig von der Auflösung Δr_j gesucht und in den Stack geladen. Das Fenster d hat die letzte Auflösungsstufe erfolgreich passiert, und deshalb kann ein Objekt in der aktuellen Auflösung vorliegen, das um einen Verschiebungsvektor innerhalb Auflösung Δr_j gegenüber des Fensters d verschoben ist.

Mit den Nachbarn d_n wird also eine Feinsuche innerhalb der neuen Auflösung durchgeführt um mögliche verschobene Objekte zu detektieren. Falls das aktuelle Detektionsfenster in einer Auflösungsstufe $\Delta r \leq \Delta r_i$ negiert wird, wird es verworfen. Es sei nochmals darauf hingewiesen, dass der Detektionsalgorithmus 4.6 bereits Teil des Trainingsalgorithmus 4.5 mit den bis zu diesem Zeitpunkt vorhandenen Stufenklassifikatoren in Schritt 3a ist, um eine repräsentative Trainingsmenge bezüglich der Auflösungen und Frequenzeigenschaften zu generieren. Das Prinzip der Detektion ist in Abb. 4.4.2 dargestellt. Aus Übersichtlichkeitsgründen sind die Zentren der Detektionsfenster als Punkte markiert und die Pfeile entsprechen den Klassifikationen für das Detektionsfenster d. Jede Auflösungsstufe Δr_i wird durch eine entsprechende Teilkaskade $h_{\Delta r_i}$ repräsentiert, die Merkmale $\mathcal{F}_{\Delta r_i}$ aus dieser Auflösung enthalten.

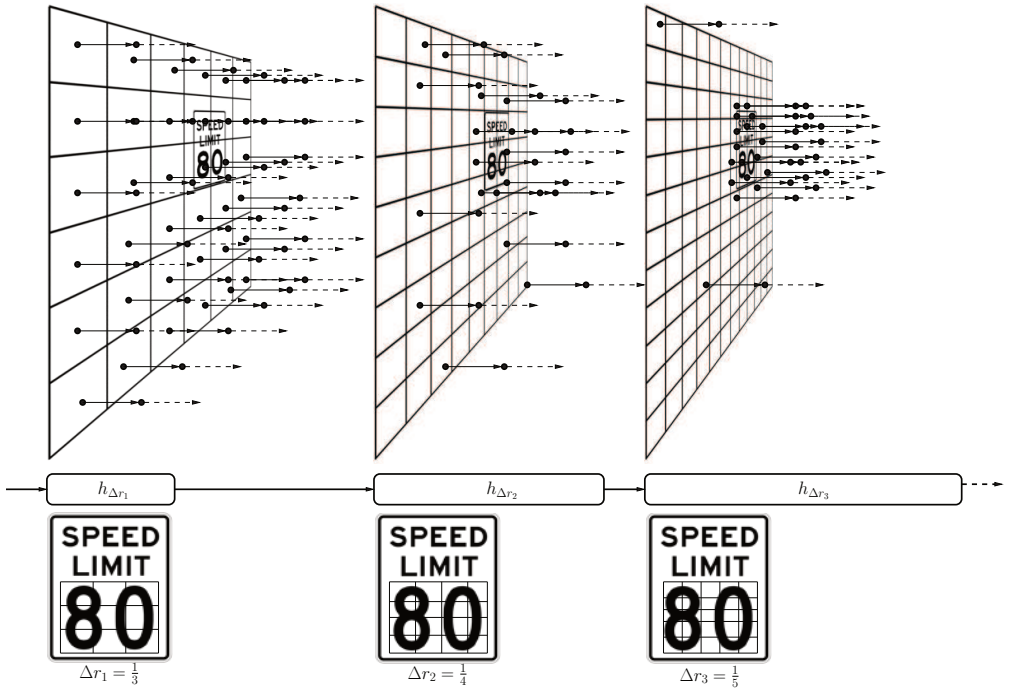


Abbildung 4.4.2: Hierarchische Detektion mittels auflösungsspezifischer Teilkaskaden $h_{\Delta r_i}$ im entsprechenden Gitter der Auflösung.

Für die Nachbarsuche in Schritt 3b wird eine von der Auflösung Δr_k abhängige Menge an Verschiebungsvektoren verwendet:

$$\mathcal{V}_{\Delta r_k}(x,y) = \{\underline{v} = (x+u, y+v)^T\} \tag{4.4.3}$$

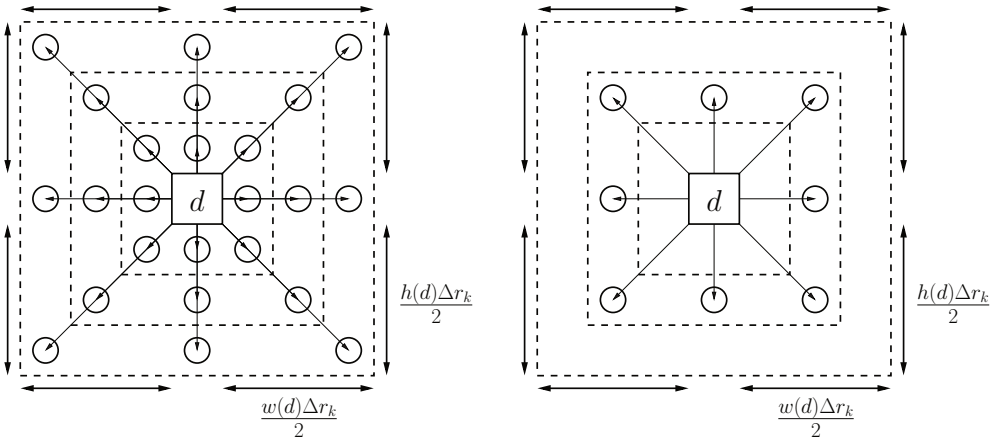
mit

$$u \in \{\lfloor -\frac{w(d)\Delta r_k}{2(k-i)} \rfloor, \ldots, \lceil +\frac{w(d)\Delta r_k}{2(k-i)} \rceil \,|\, 0 \leq i < k\}$$

$$v \in \{\lfloor -\frac{h(d)\Delta r_k}{2(k-i)} \rfloor, \ldots, \lceil +\frac{h(d)\Delta r_k}{2(k-i)} \rceil \,|\, 0 \leq i < k\}$$

Hierbei ist $w(d)$ die Breite und $h(d)$ die Höhe des aktuellen Detektionsfensters. Je höher die Auflösung Δ_k desto mehr 8-er-Nachbarschaften werden für die Verschiebungsvektoren verwendet.

Abb. 4.4.3(a) zeigt die Verschiebungsvektoren für die Auflösung $\Delta r_k = \frac{1}{3}$. Es sind drei 8-er-Nachbarschaften mit insgesamt 24 Vektoren. Der maximale Abstand wird aus der halben Auflösung $\frac{w(d)\Delta r_k}{2k}$ berechnet, wodurch mit



(a) (b)

Abbildung 4.4.3: (a): Verschiebungsvektoren für die auflösungsabhängige Berechnung der Nachbarn für die Feinsuche in Schritt 3b von Alg. (4.6) mit $\Delta r_k = \frac{1}{3}$. (b): Verschiebungsvektoren mit konstanter Anzahl, unabhängig der Auflösungsstufe.

den benachbarten Detektionsfenstern die Fläche lückenlos abgedeckt wird. In Abb. 4.4.3(b) sind die Verschiebungsvektoren für eine konstante Anzahl von acht Nachbarn, unabhängig von der Auflösungsstufe gezeigt. Alg. (4.7) be-

Algorithmus 4.7 Adaptive Feinsuche

1. **Gegeben:** Detektionsfenster und Auflösungsstufe $(d, \Delta r_k)$
2. Berechne Verschiebungsvektoren $\mathcal{V}_{\Delta r_k}(d)$ mit Gl. (4.4.3).
3. Berechne die n Nachbar-Detektionsfenster $\mathcal{N}$ mit d und $\mathcal{V}_{\Delta r_k}(d)$ abhängig von der aktuellen Auflösung. Die Nachbarn $\mathcal{N}$ können dann in Alg. (4.6) Schritt 3b dem Stack $\mathcal{S}$ hinzugefügt werden.

schreibt die adaptive Feinsuche mittels Gl. (4.4.3) zur Berechnung der Nachbarn, die dann in Alg. (4.6), Schritt 3b dem Stack $\mathcal{S}$ hinzugefügt werden. Alg. (4.8) beschreibt die auflösungsabhängige Nachbarsuche mit konstant acht Nachbarn nach Abb. 4.3(b).

Algorithmus 4.8 Konstante Feinsuche

1. **Gegeben:** Detektionsfenster und Auflösungsstufe $(d, \Delta r_k)$
2. Berechne acht Verschiebungsvektoren abhängig von der Auflösung Δr_k nach Abb. 4.3(b).
3. Berechne die acht Nachbar-Detektionsfenster, die in Alg. (4.6), Schritt 3b dem Stack $\mathcal{S}$ hinzugefügt werden.

4.5 US-Speed-Limit-Datensatz

Der **Trainingsdatensatz** der Klassifikatoren für US-Geschwindigkeitsbegrenzungen besteht aus 12930 Grauwertbildern von amerikanischen Geschwindigkeitsbegrenzungsschildern, die mit einem Testfahrzeug unter typischen Realweltbedingungen aufgenommen wurden. Bei der Aufnahme eines Realweltverkehrsschildes als isoliertes Objekt, das durch die Fahrzeugbewegung in unterschiedlichen Größen auftritt, wird, solange das Schild sichtbar ist, eine Sequenz aus einzelnen Bildern mit unterschiedlich skalierten Versionen des Schildes aufgenommen. Zur

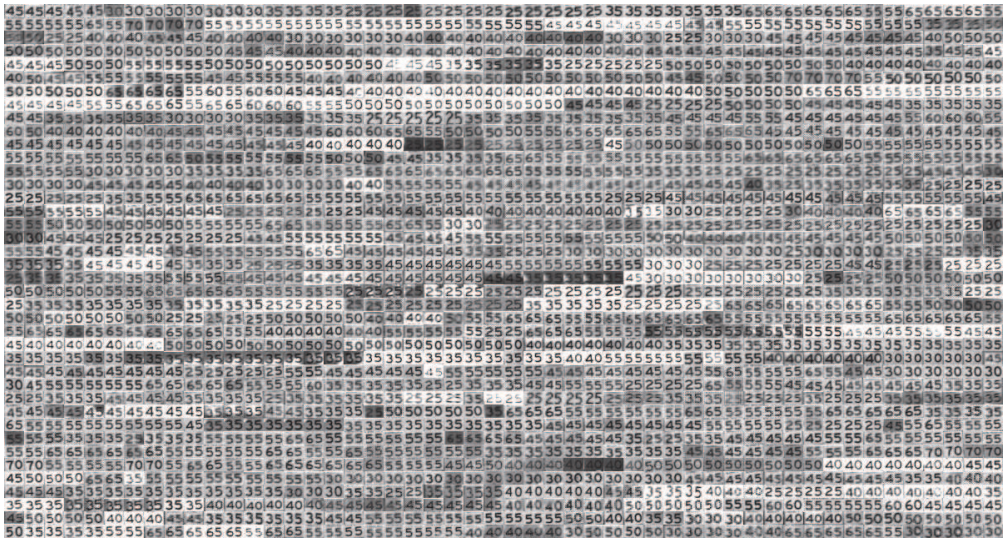


Abbildung 4.5.1: Zufällige Auswahl an 2500 Bildern aus dem Trainingsdatensatz. Die Bilder wurden auf eine Breite von 24 Pixel skaliert.

Darstellung sind in Abb. 4.5.1 eine zufällige Auswahl an 2500 Schildern, die auf die Breite von 24 Pixel skaliert wurden, dargestellt. Man erkennt sowohl stark überstrahlte Schilder, als auch sehr dunkle Schilder, die in der Dämmerung aufgenommen wurden. Die verschwommenen Schilder rühren von Bewegungsunschärfe durch höhere Fahrzeuggeschwindigkeiten her.

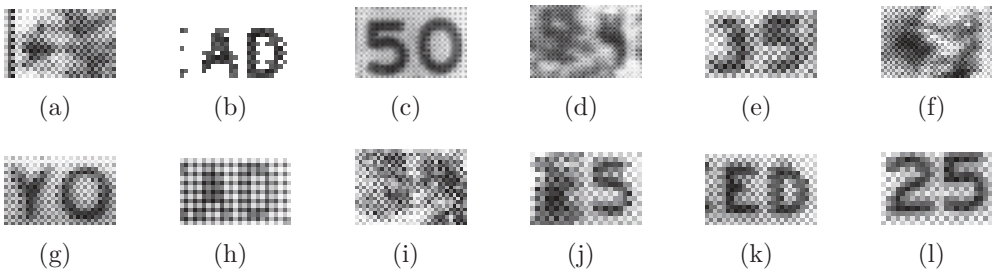
Der **Testdatensatz** besteht aus 6000 Bildern und enthält Realweltschilder, die nicht im Trainingsdatensatz enthalten sind. Die Bilder zeigen typische Effekte wie Schlagschatten, Unschärfe durch die Eigenbewegung des Fahrzeugs und Schachbrettstrukturen, die aus dem Bayer-Pattern resultieren. Die Geschwindigkeitsbegrenzungen reichen von 25 Mph in 5-er Schritten bis 70 Mph. Der Zwei-Ziffern-Block wird mit einem Rand bestehend aus 15 % der äußeren Strukturen der Ziffern als Trainingsdaten für die Detektoren verwendet. Typische Beispiele für die Objektklasse, die unter schwierigen Beleuchtungsverhältnissen aufgenommen wurden, sind in Abb. 4.5.2(a) dargestellt. Die Basisbreiten der Objekte reichen von 24 Pixel bis 60 Pixel. Die Bildgröße ist 752×480 Pixel und die Detektion erfolgt auf dem kompletten Bild. Die Negativbeispiele für die Hintergrundklasse werden aus den Bildern generiert, indem der Sliding-Window-Algorithmus in der jeweiligen Auflösungsstufe ausgeführt wird und alle Fenster, die sich nicht mit einem Objekt überschneiden verwendet werden. Alle Auswertungen in dieser Arbeit wurden auf diesem Datensatz vorgenommen. Es hat sich herausgestellt, dass die Negativbeispiele in hohen Kaskadenstufen sehr hohe strukturelle Ähnlichkeiten zu der Objektklasse aufweisen.

(a)

US Speed Limit Set										
Speed Limit	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**60**	**65**	**70**
Train Samples	1632	718	1960	1237	2261	1203	2541	119	1165	94
Test Samples	921	281	960	643	1158	514	1212	77	558	33

(b)

Abbildung 4.5.2:
(a): Typische Beispiele der Objektklasse unter schwierigen Beleuchtungsverhältnissen
(b): Verteilung der unterschiedlichen Geschwindigkeitsbegrenzungen



(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

Abbildung 4.5.3: Normalisierte Falsch-Positive der Stufe 113 von insgesamt 118. Die Bilder 4.3(c), 4.3(e) und 4.3(l) stammen von Bildern mit zwei Verkehrsschildern (links und rechts), wovon eines bei der Kennzeichnung übersehen wurde. Der Ausschnitt 4.3(k) stammt von den Buchstaben “E D” der Beschriftung “S P E E D”.

In Abb. 4.5.3 sind typische Falsch-Positive aus Stufe 113 von insgesamt 118 Stufen eines Kaskadenklassifikators dargestellt. Die Bilder 4.3(c), 4.3(e) und 4.3(l) stammen von Bildern mit zwei Verkehrsschildern (links und rechts) wovon eines bei der Kennzeichnung übersehen wurde. Das Beispiel 4.3(d) stammt von einem Baum, bei dem die Blätter die Form einer “25” annehmen.

4.6 Auswertungsalgorithmus auf den Testdaten

Eine aussagekräftige Auswertung kann bei einem Klassifikator nur durch die getrennte Betrachtung der Positiv- und Negativfehler getroffen werden, allerdings gibt es unterschiedliche Einheiten bei der Darstellung der Fehler. Eine weitverbreitete Darstellung sind die Fehler pro Detektionsfenster, wie zum Beispiel FPPW (False-Positives-Per-Window). Mit dieser Darstellung lässt sich das Detektionsverhalten auf dem Gesamtbild nur beurteilen, wenn die Gesamtzahl der Detektionsfenster bekannt sind. Eine geeignetere Darstellung zur Beurteilung von Detektoren ist eine flächenabhängige Darstellung, wie zum Beispiel FPPF (False-Positives-Per-Frame), also die Falsch-Positiven-Pro-Bild [34]. Eine Darstellung pro Pixel wäre zwar von der Bildgröße unabhängig, ist aber wieder schwerer interpretierbar und nicht gebräuchlich. Deshalb werden die ROC-Punkte durch die Detektionsrate der Objekte, die eine flächenmässige Überschneidung des realen Objektes von mehr als 50 %

erzielen auf der Y-Achse und der Falsch-Positiven pro Bild (FPPF) auf der X-Achse dargestellt. Da ein Schwerpunkt dieser Arbeit bei der Anwendung im Echtzeitbereich liegt, werden zusätzlich zu den ROC-Kurven die mittlere Laufzeit des Detektors in Millisekunden pro Bild auf der Y-Achse und die zugehörigen Falsch-Positiven pro Bild auf der X-Achse in zusätzlichen Grafiken dargestellt.

Im Folgenden wird der Auswertungsalgorithmus zur Generierung der ROC-Kurven (siehe Abb. 2.3.2) erklärt. Der Klassifikator wird auf jedem Bild des Testsets aufgerufen und das Erkennungsergebnis des Einzelbildes wird ermittelt. Zusätzlich wird die Laufzeit gemessen, die für die Detektion des Einzelbildes benötigt wird. Die einzelnen Erkennungsergebnisse pro Bild werden dann gemittelt und ergeben einen Punkt der ROC-Kurve. Um unterschiedliche Punkte zu generieren können prinzipiell beliebige Parameter eines Klassifikators variiert werden. Bei einem Kaskadenklassifikator bietet es sich allerdings an, sukzessive den letzten Stufenklassifikator zu deaktivieren, da die Klassifikationsleistung mit weiteren Stufen im Allgemeinen sinkt. Bei dem vorgestellten auflösungsabhängigen Verfahren kann es jedoch sein, dass die Erkennungsleistung beim Übergang in eine neue Auflösungsstufe wieder ansteigt. Da alle Kaskadenklassifikatoren eine ausreichende Anzahl an Stufenklassifikatoren besitzen, werden die Stufenklassifikatoren nicht verändert. Falls dies jedoch trotzdem nötig sein sollte, können ROC-Punkte der Stufenklassifikatoren mit Alg. (3.6) erzeugt werden, wobei die unterschiedlichen Signifikanzniveaus den ROC-Punkten entsprechen.

Alg. (4.9) beschreibt den Auswertungsmechanismus für die Detektionen pro Bild $\mathcal{A}$ eines Klassifikators, anhand der tatsächlichen Objektpositionen $\mathcal{G}$ (Schritt 1a). Der Algorithmus funktioniert auch mit einer Anzahl von mehreren Objekten pro Bild, wie es z. B. bei Schildern, die auf beiden Seiten des Straßenrandes aufgestellt sind, öfters vorkommt. Ein typisches Beispiel dazu ist in Abb. 4.6.1 dargestellt. Die diskreten Falsch-Positiven werden in Schritt 2b berechnet, indem für jedes detektierte Fenster a die maximale Schnittfläche $s_{\max}$ berechnet wird. Anhand dieser werden die diskreten Falsch-Positiven fp_d erhöht, falls die Fläche einen Minimalwert unterschreitet. Die reellen, flächenmäßigen Falsch-Positiven fp_r werden direkt mit dieser Schnittfläche berechnet.

Zur Berechnung der diskreten und reellen Falsch-Negativen wird für jedes tatsächliche Objektfenster die maximale Schnittfläche berechnet und daraus wie im ersten Schritt die Variablen aktualisiert. Die in dieser Arbeit erzeugten ROC-Kurven sind anhand der diskreten Werte erstellt, da diese Art und

Algorithmus 4.9 Ermittlung des Detektionsergebnisses pro Bild

1. **Gegeben:**
   (a) Tatsächliche Objektfenster $\mathcal{G}$
   (b) Detektionsergebnisse $\mathcal{A}$
2. **Für alle** $a \in \mathcal{A}$:
   (a) Berechne die maximale Schnittfläche $s_{\max} = \max\{a \cap g \,|\, g \in \mathcal{G}\}$ mit $s_{\max} \in [0, \ldots, 1]$.
   (b) **Falls** $s_{\max} < s_{\min}$: $\mathrm{fp}_d \leftarrow \mathrm{fp}_d + 1$
   (c) $\mathrm{fp}_r \leftarrow \mathrm{fp}_r + (1 - s_{\max})$, $\mathrm{tp}_r \leftarrow \mathrm{tp}_r + s_{\max}$
3. **Für alle** $g \in \mathcal{G}$:
   (a) Berechne die maximale Schnittfläche $s_{\max} = \max\{g \cap a \,|\, a \in \mathcal{A}\}$.
   (b) **Falls** $s_{\max} < s_{\min}$: $\mathrm{fn}_d \leftarrow \mathrm{fn}_d + 1$, **sonst** $\mathrm{tp}_d \leftarrow \mathrm{tp}_d + 1$
   (c) $\mathrm{fn}_r \leftarrow \mathrm{fn}_r + (1 - s_{\max})$
4. $\mathrm{tpr} = \mathrm{tp}/(\mathrm{tp} + \mathrm{fn})$, $\mathrm{fpr} = \mathrm{fp}/(\mathrm{fp} + \mathrm{tn})$

Weise in der Literatur als Pascal-Kriterium bekannt ist [93]. Durch Schritt 2c und 3c stehen zwar auch die normierten, flächenbezogenen Detektionsraten zur Verfügung, welche jedoch bei unterschiedlichen Auflösungen der Teilkaskaden durch die unterschiedlich präzise räumliche Positionierung der Detektionsfenster auflösungsabhängig sind. Die Schwelle für eine Detektion liegt beim Pascal-Kriterium bei $s_{\min} = 0.5$. Die bildbezogenen Ergebnisse werden über alle Bilder des Testsets gemittelt, woraus das Detektionsergebnis resultiert.

4.7 Visualisierung der Strukturen am Beispiel der US-Speed-Limits

Um das Verfahren an einem Beispiel der US-Geschwindigkeitsbegrenzungen zu erklären und die Struktur und Anzahl der Stufenklassifikatoren zu analy-

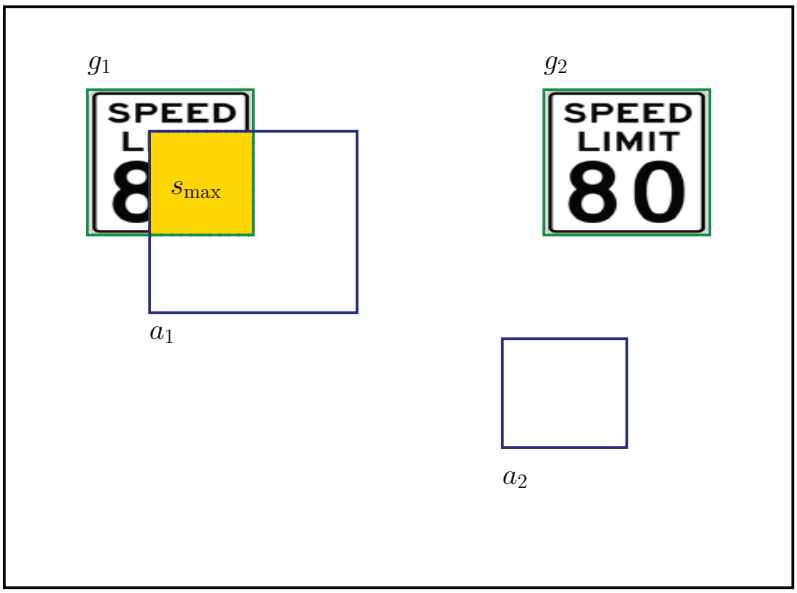


Abbildung 4.6.1: Eine typische Konfiguration mit zwei tatsächlichen Objektfenstern $g_{1,2}$ und zwei Detektionen $a_{1,2}$ zur Veranschaulichung von Alg. (4.9). Die gelbe Fläche ist die Schnittfläche $s_{\max}$, die zur Berechnung der Klassifikationsraten dient.

sieren werden im Folgenden eine Plausibilitätsanalyse und eine visuelle Überprüfung durchgeführt, sowie die wichtigsten Parameter besprochen. Sofern keine anderen Parameterkonfigurationen erwähnt werden, gilt diese Konfiguration für die experimentellen Versuche in den folgenden Abschnitten. Außerdem wird der hier beschriebene Kaskadenklassifikator mit einer AdaBoost-Kaskade und Haar-ähnlichen Merkmalen verglichen. Es wurde der in Abschnitt 4.5 beschriebene Datensatz verwendet. Das AdaBoost-Verfahren ist in Kapitel 6 genau beschrieben.

Trainingsparameter							
Kaskade H_C			**Alg.**	**Stufe** h_s			**Alg.**
$e_{\text{stop}}(\omega_0)$	$\mathcal{F}_{\Delta r}$	A-priori	4.5, 4.4	$e_{\text{SFS}}(\omega_0)$	$t_{\max}$	ν	4.2, 4.1
0.95	$\{\frac{1}{3}, \ldots, \frac{1}{12}\}$	update		0.6	8	Gl. (4.1.6)	

Tabelle 4.7.1: Trainingsparameter der zu analysierenden Kaskade

Die Trainingsparameter der zu analysierenden Kaskade sind in Tabelle

Parameter h_s				
Projektion	**Entscheidungsfunktion**			**Alg.**
$\underline{w}$	Funktion	Einträge b	Signifikanzniveau α	3.6, 3.4
Alg. (3.2)	$\max(P(\omega_i))$	256	0.05	

Tabelle 4.7.2: Parameter für die Entscheidungsfunktion

4.7.1 gelistet. Es wurde der auflösungsspezifische Trainingsalgorithmus (4.5) mit dem Bayes'schen Kaskadentraining (4.4) und der sequentiellen Vorwärtsauswahl zum Training der Projektionsachsen (4.1) verwendet. Die Schwellen der A-priori-Wahrscheinlichkeiten bei der Aktualisierung mit Alg. (4.3) sind $P_{\min}(\omega_1)) = 0.8$ und $P_{\min}(\omega_0)) = 10^{-8}$. Die Auflösungsstufen reichen von $\Delta r_{\frac{1}{3}}$ bis $\Delta r_{\frac{1}{12}}$ und zur Detektion wurde die adaptive Feinsuche (Alg. (4.7)) bei der auflösungsspezifischen Depth-First-Detektion, Alg. (4.6), verwendet.

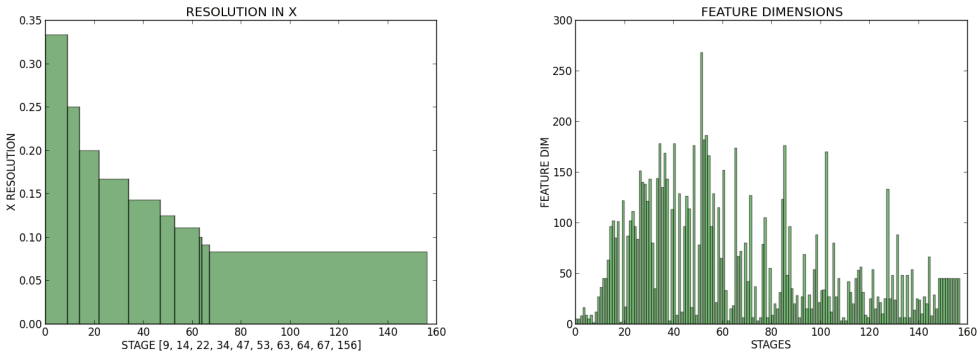
Die Basisbreiten und die Faktoren zur Berechnung der Detektionsfenster sind in Tabelle 4.7.3 gelistet. Die Merkmale sind die in Abschnitt 3.1.1

Basisbreiten und Verschiebungsfaktor (Alg. (4.6), Gl. (4.4.1))											
Basisbreiten $\mathcal{W}$											Verschiebung ξ
24	26	28	32	36	40	44	48	52	56	60	1

Tabelle 4.7.3: Basisbreiten und Faktor zur Berechnung der initialen Detektionsfenster mit Gl. (4.4.1).

beschriebenen Tensor- und Intensitätsmerkmale. In der Auflösungsstufe $\Delta r_{\frac{1}{3}}$ wurden alle 1-er und 2-er Kombinationen aus allen möglichen Rechtecken innerhalb des Gitters verwendet. In Auflösungsstufe $\Delta r_{\frac{1}{5}}$ wurden nur noch einzelne Basisrechtecke ohne Kombinationen verwendet und ab Stufe $\Delta r_{\frac{1}{6}}$ wurden nur noch feste Maskenmerkmalsgeometrien verwendet, wobei die größte Maske eine 4×4-Maske ist, da sonst der Berechnungsaufwand zu hoch wird. In Tabelle 4.7.2 sind die Parameter der Entscheidungsfunktion gelistet. Es wurden Frequenztabellen mit 256 Einträgen und einem Signifikanzniveau von $\alpha = 0.05$ verwendet.

Abb. 4.1(a) zeigt sowohl die Anzahl der Stufenklassifikatoren, als auch die entsprechenden Auflösungsstufen. Durch das Aktualisieren der A-priori-Wahrscheinlichkeiten der Stufenklassifikatoren kann eine sehr hohe Anzahl von über 150 Stufenklassifikatoren verwendet werden. In Abb. 4.1(b) sind die Dimensionen der Projektionsachsen der entsprechenden Stufenklassifika-



(a) (b)

Abbildung 4.7.1: (a): Anzahl der Stufenklassifikatoren (X-Achse) in den entsprechenden Auflösungsstufen (Y-Achse). (b): Dimensionen der Projektionsachsen der Stufenklassifikatoren (Y-Achse).

toren dargestellt. Bei den Übergängen in höhere Auflösungen enthalten die Projektionsachsen weniger Merkmale.

Abb. 4.7.2 zeigt die Visualisierung aller Komponenten des ersten Stufenklassifikators. Die Abbildung zeigt oben links die gewählte geometrische Merkmalsregion in Auflösungsstufe $\Delta r_{\frac{1}{3}}$ und oben rechts die daraus extrahierten Merkmale. Unten links sind die Gewichtungen der Merkmale als Balken auf der Y-Achse über den einzelnen Merkmalen auf der X-Achse dargestellt. Die Visualisierung der Entscheidungsfunktion ist unten rechts zu sehen, wobei die Differenzen der klassenbasierten Häufigkeiten, gewichtet mit den A-priori-Wahrscheinlichkeiten als Klassifikationsfunktion dient. Für den ersten Stufenklassifikator wurden die tensorbasierten Orientierungsmerkmale gewählt, da bei den Ziffernpaaren im ausgewählten Merkmalsbereich eine starke vertikale Orientierung der Strukturen erkennbar ist. Durch die zusätzlichen Polynome zweiten Grades wird die Klassifikationsleistung mit minimalem zusätzlichen Rechenaufwand erhöht.

Abb. 4.7.3 zeigt den Stufenklassifikator Nr. 36 der Kaskade. In der Auflösungsstufe $\Delta r_{\frac{1}{7}}$ wurden Maskenmerkmale verwendet. Die nummerierten Rechtecke der Masken oben links entsprechen den bei der sequentiellen Vorwärtsauswahl gewählten Masken. Oben rechts sind die entsprechenden Merkmale dargestellt, wobei sich auf der X-Achse die Rechtecke befinden. Es ist ersichtlich, dass für die 16 Rechtecke der ersten Maske sowohl Tensor- als

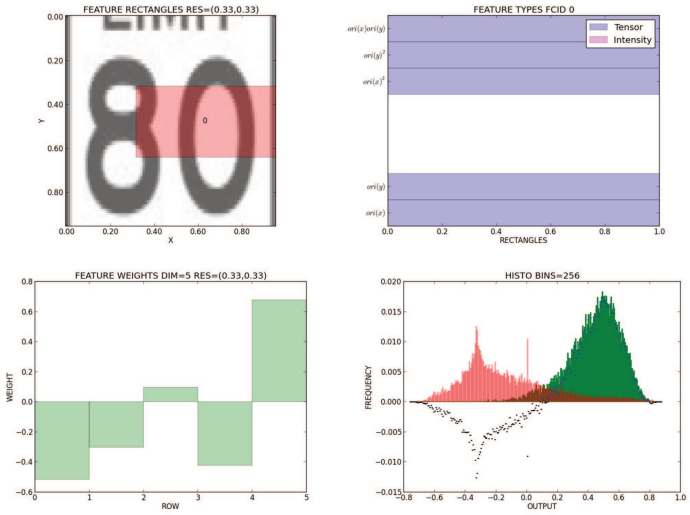


Abbildung 4.7.2: Analyse des ersten Stufenklassifikators.
Oben links: Merkmalsbereich, oben rechts: Tensormerkmale, unten links: Gewichtung der Merkmale, Unten rechts: Entscheidungsfunktion als Differenz der klassenspezifischen Frequenzen.

auch Intensitätsmerkmale gewählt wurden. Bei den restlichen Zyklen wurden entweder Tensor- oder Intensitätsmerkmale gewählt. Die entsprechenden Gewichte des 169-dimensionalen Projektionsvektors sind unten links dargestellt. Bei der Entscheidungsfunktion unten rechtes erkennt man im Gegensatz zu Abb. 4.7.2, dass die Entscheidungsfunktion durch die geringe Auftrittswahrscheinlichkeit der Hintergrundklasse in diesem Bereichen stark heruntergewichtet ist. Abb. 4.7.4 zeigt die Stufe 38 in Auflösungsstufe $\Delta r_{\frac{1}{7}}$, die lediglich aus Intensitätsmerkmalen besteht. Bei diesem Merkmal wird der helle Zwischenraum der dunklen Ziffern ausgenutzt.

Durch die Normierung der Merkmale nehmen dunkle Bereiche negative und helle Bereiche positive Werte an. Deshalb bewirkt die starke Negativgewichtung des ersten dunklen Bereichs und die positive Gewichtung (Abb. unten links) eine Projektion der Ziffernpaare auf hohe Werte und damit in den positiven Bereich der Verteilung (siehe Abb. unten rechts). Allerdings ist dort an den schwarzen Balken der Entscheidungsfunktion auch zu erkennen, dass ein zweiter negativer Klassifikationsbereich jenseits der maximalen positiven Verteilung existiert. Als Vergleich wird nun eine Viola-Jones Kas-

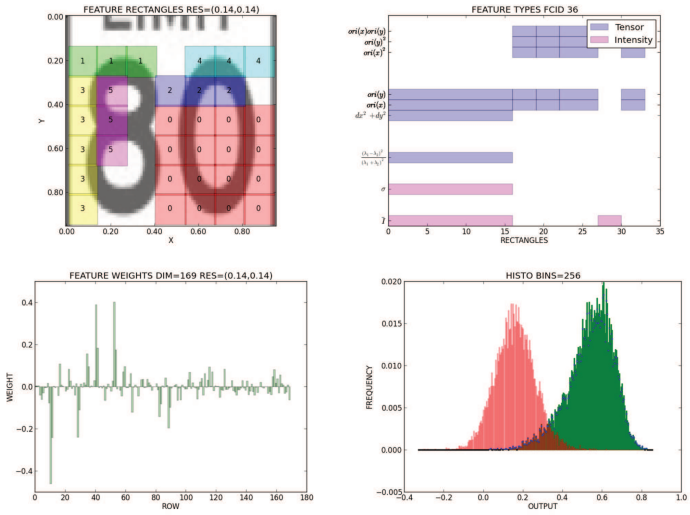


Abbildung 4.7.3: Analyse des Stufenklassifikators 36.
Oben links: Merkmalsbereich, Oben rechts: Merkmale, unten links: Gewichtung der Merkmale, unten rechts: Entscheidungsfunktion als Differenz der klassenspezifischen Frequenzen.

kade [115], die aus Haar-ähnlichen schwachen Klassifikatoren besteht, und die mit dem AdaBoost-Verfahren zu starken Stufenklassifikatoren kombiniert wird, analysiert. Tabelle 4.7.4 zeigt die Parametereinstellungen des

Trainingsparameter							
Kaskade H_C			**Alg.**	**Stufe** h_s			**Alg.**
$e_{\text{stop}}(\omega_0)$	$\mathcal{F}_{\Delta r}$	A-priori	4.5, 4.4	$e_{\text{ada}}(\omega_0)$	$t_{\max}$	ν	6.1
0.99	$\{\frac{1}{12}\}$	const		0.25	162	-	

Tabelle 4.7.4: Trainingsparameter der zu analysierenden Kaskade

Parameter				
Projektion	**Entscheidungsfunktion**			**Alg.**
$\underline{w}$	Funktion	Einträge b	Signifikanzniveau α	3.4, 3.6, 3.9
Alg. (6.1)	$\Theta(x_s - \theta)$	1024	0.025	

Tabelle 4.7.5: Parameter für die Entscheidungsfunktion

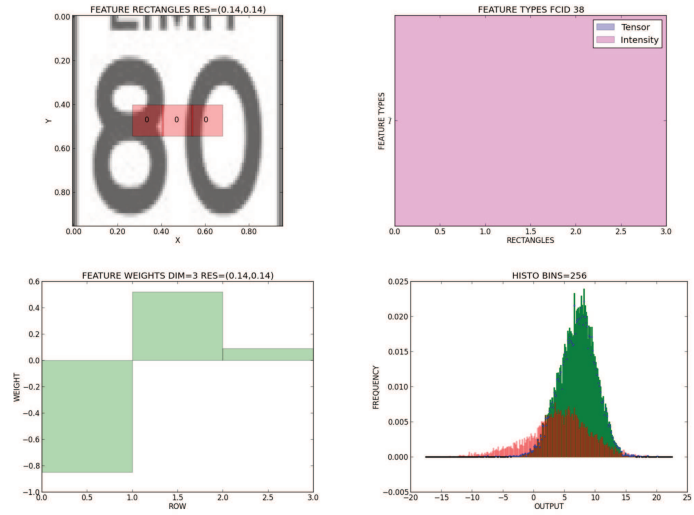


Abbildung 4.7.4: Analyse des Stufenklassifikators 38.
Oben links: Merkmalsbereich, oben rechts: Merkmale, unten links: Gewichtung der Merkmale, unten rechts: Entscheidungsfunktion als Differenz der klassenspezifischen Frequenzen

Trainings. Es konnten lediglich in der höchsten Auflösungsstufe $\Delta r_{\frac{1}{12}}$ funktionsfähige Klassifikatoren erzeugt werden. Da keine Aktualisierung der A-priori-Wahrscheinlichkeiten bei diesem Algorithmus verwendet werden kann, wird das Signifikanzniveau auf 0.025 eingestellt, sonst wäre die Detektionsleistung nicht ausreichend.

Abb. 4.7.5 zeigt die Anzahl der schwachen Klassifikatoren in den entsprechenden Kaskadenstufen. Der erste schwache Klassifikator, der beim Boosting gewählt wird, ist in Abb. 4.7.6 dargestellt. Das erste Haar-ähnliche Merkmal sitzt unten links auf der rechten Ziffer. Da bei den Haar-ähnlichen Merkmalen nur Intensitäten verwendet werden, ist der Bereich der Merkmalstypen in der Abbildung oben rechts vollständig ausgefüllt. Die Gewichte sind bei den Haar-Merkmalen fest und dienen zur Berechnung der Differenz der einzelnen Rechteckbereiche. Deshalb sind die Gewichte in der Abbildung unten links abwechselnd $+1$ und -1. Der Schwellwert der Entscheidungsfunktion wird durch Alg. (3.9) bestimmt und wird an die Stelle des minimalen Fehlers gesetzt. Der zweite schwache Klassifikator ist in Abb. 4.7.7 dargestellt und befindet sich oben auf der rechten Ziffer. Die Kombination aus diesen

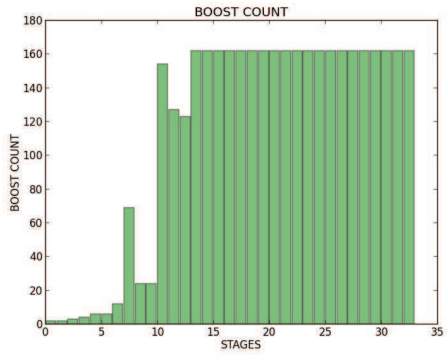


Abbildung 4.7.5: Anzahl der schwachen Klassifikatoren (Y-Achse) in den entsprechenden Stufen (X-Achse).

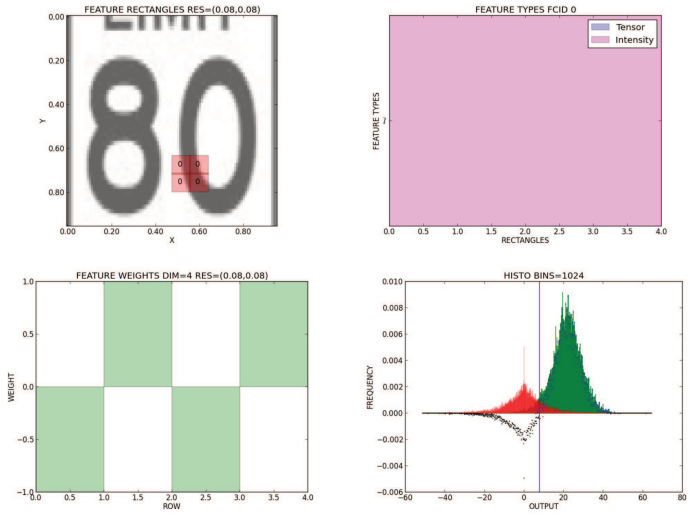


Abbildung 4.7.6: Analyse des ersten weak learners des ersten AdaBoost-Klassifikators.

Merkmalen ermöglicht eine Falsch-Positiv-Rate von weniger als 0.25 %. Im Gegensatz zu dem sehr breiten Merkmal in Auflösungsstufe $\Delta r_{\frac{1}{3}}$ sind die Haar-Merkmale durch die höhere Auflösung, die allerdings bei dieser Art von Merkmalen wichtig ist, deutlich kleiner. Die Abtastung bei der Detektion durch die hohe Auflösung ist deshalb rechenaufwändig, da eine hohe Anzahl

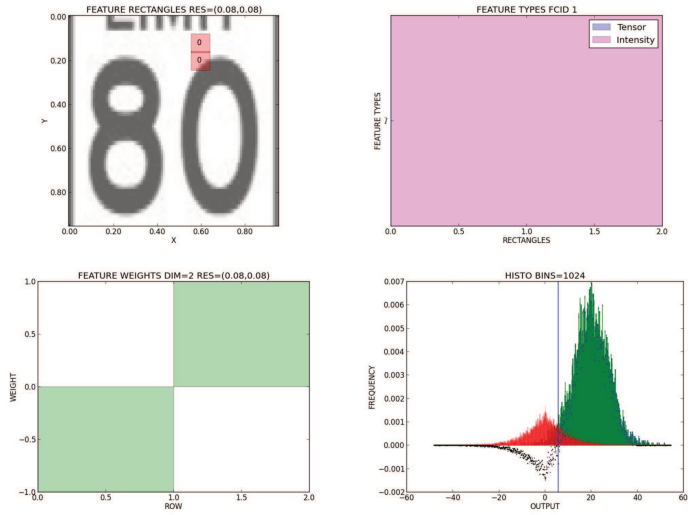


Abbildung 4.7.7: Analyse des zweiten weak learners des ersten AdaBoost-Klassifikators.

an Detektionsfenstern überprüft werden muss.

4.8 Experimentelle Auswertung

In den folgenden Versuchen werden unterschiedliche Parameter und Funktionen der eingeführten Algorithmen getestet. Da bei dieser Arbeit die Echtzeitfähigkeit der Klassifikatoren ein wichtiger Punkt ist, werden zur Beurteilung der Klassifikatoren zusätzlich zu den Klassifikationsraten die mittleren Laufzeiten der Klassifikatoren gemessen und über den Falsch-Positiven pro Bild dargestellt. Alle Klassifikatoren wurden mit dem in Abschnitt 4.5 beschriebenen Trainingsdatensatz erstellt und mit dem dort ebenfalls beschriebenen separaten Testdatensatz evaluiert, wobei ein aktueller Rechner mit einem 3.3 GHz Prozessor (Intel i7 - Architektur) verwendet wurde, auf dem ein spezifischer Klassifikator mit einem Prozessor trainiert und mit einem Prozessor evaluiert wurde. Um die Verfahren zu beschleunigen wurde parallel auf unterschiedlichen Prozessoren trainiert und evaluiert.

Der Testdatensatz enthält keine Realweltschild-Objekte aus dem Trainingsdatensatz. Die Einheit der dargestellten Falsch-Positiven bei den ROC-Kurven sind Falsch-Positive pro Bild, was nicht mit den Falsch-Positiven

pro Detektionsfenster verwechselt werden sollte, die auch in einigen Artikeln in der Literatur verwendet werden. Alle Algorithmen wurden in dieser Arbeit vollständig, ohne Abhängigkeiten von externen Bibliotheken, implementiert. Dafür existieren mehrere Gründe. Aufgrund der großen Anzahl an Parametern und unterschiedlichen Merkmalen ist es nur durch eine genaue Beobachtung und Analyse einer geringen Anzahl an isolierten Parametern möglich, das Verhalten der Klassifikatoren zu beurteilen und Rückschlüsse zu ziehen, da sich insbesondere bei Kaskadenklassifikatoren der Fehler in den Stufenklassifikatoren potenzieren kann. Dies kann nur durch eine modulare Softwarestruktur erfolgen, bei der einzelne Parameter isoliert variiert, sowie unterschiedliche Teilalgorithmen spezifiziert werden können. Es existieren keine frei verfügbaren Bibliotheken, die eine derartige Integration der hier beschriebenen Verfahren ermöglichen.

Durch das Fehlerverhalten der Kaskade ist es ebenfalls unumgänglich, detaillierte visuelle Überprüfungen durchzuführen um das Verhalten der Stufenklassifikatoren zu ermitteln. Da die Klassifikatoren im industriellen Umfeld in hohen Stückzahlen Anwendung finden, müssen in extensiver Weise Parameter getestet werden um energieeffiziente Varianten zu identifizieren. Bei allen Versuchen in dieser Arbeit wurden, bis auf die in den Versuchen erwähnten Unterschiede, identische Algorithmen und Parameter verwendet um das Verhalten bestmöglich interpretieren zu können.

4.8.1 Trainingszeiten

Der Umfang der zu testenden Parameter wird allerdings durch die langen Trainingszeiten begrenzt. Für den folgenden Versuch mit einer mittleren Auflösung und dem konfidenzbasierten Gradientenboosting als Trainingsverfahren der Stufenklassifikatoren ergaben sich Trainingszeiten von bis zu vier Wochen. Mit dem auflösungsspezifischen Kaskadentraining konnten die Trainingszeiten auf zwei Tage reduziert werden. Allerdings wurden für die teilüberwachten Trainingsverfahren in Kapitel 5 und die Unterteilungsbäume aus Kapitel 6 bis zu sechs Wochen benötigt.

4.8.2 Merkmale

Bei der ersten Auswertungsreihe werden die Auswirkungen unterschiedlicher Merkmale und Merkmalskombinationen getestet. Um Seiteneffekte zu minimieren und die Auswirkungen der unterschiedlichen Merkmalstypen auf

die Laufzeit zu verdeutlichen wird nur eine einzige, mittlere Auflösung der Merkmale von $\Delta r_{\frac{1}{5}}$ verwendet. Da sich gezeigt hat, dass die Haar-ähnlichen Merkmale bei einer deutlich feineren Auflösung wirksam sind, werden diese zusätzlich mit der Auflösung $\Delta r_{\frac{1}{12}}$ trainiert.

Das Training der Stufenklassifikatoren in der mittleren Auflösung $\Delta r_{\frac{1}{5}}$ mit der sequentiellen Vorwärtsauswahl würde allerdings zu relativ hohen Falsch-Positiv-Raten führen. Aus diesem Grund werden die Stufenklassifikatoren mit dem in dieser Arbeit erweiterten, konfidenzbasierten Gradientenboosting-Verfahren erstellt (siehe Alg. (6.2)). Solange die Anzahl der Falsch-Positiven gesenkt werden kann, wird die Kaskade mit neuen Stufen erweitert. Für eine detaillierte Beschreibung und Diskussion des Verfahrens sei auf Kapitel 6, Abschnitt 6.2 verwiesen. Zum Vergleich wird mit identischen Haar-ähnlichen Merkmalen eine Kaskade mit dem AdaBoost-Verfahren (siehe Alg. (6.1)), das ebenfalls in Kapitel 6 beschrieben ist und auf schwellwertbasierten Entscheidungen basiert, erzeugt. Für die Entscheidungsfunktionen beim Gradientenboosting werden sowohl für die Basisklassifikatoren als auch für den Ensembleklassifikator 32 Einträge in den Frequenztabellen verwendet und das Signifikanzniveau von $\alpha = 0.05$ verwendet (siehe Alg. (3.6)). Die maximale Anzahl der Basisklassifikatoren wurde auf $t_{\max} = 256$ beschränkt, die Fehlerschwelle für die Hinzunahme weiterer Basisklassifikatoren auf $e_{\mathrm{gb}}(\omega_0) = 0.5$ gesetzt, und die stochastische Komponente wurde auf $\beta = 1$ gesetzt (siehe Alg. (6.2)). Bis auf eine Kaskade, deren Basisklassifikatoren und Ensembleklassifikatoren mit der multiplen linearen Regression (siehe Alg. (3.3)) trainiert wurden, wurden alle anderen Kaskaden mit der Fisher's linearen Diskriminanten (siehe Alg. (3.2)) trainiert. Zur kostenoptimierten Selektion der Klassifikatoren wurde die Gewichtungsfunktion Gl. (4.1.5) in Verbindung mit Gl. (4.1.3) verwendet. Außer bei den Haar-ähnlichen Merkmalen werden einzelne Rechtecke mit allen möglichen Formen verwendet (siehe Tabelle 3.1.1).

Tabelle 4.8.1 zeigt die verwendeten Merkmale und die unterschiedlichen Trainingsparameter. In der ersten Spalte wird eine Identifikationsnummer zugeordnet, mit der sich die entsprechenden ROC-Kurven aus Abb. 4.8.1 ermitteln lassen. Die Kaskaden in den letzten zwei Reihen haben vergleichsweise hohe Falsch-Positiv-Detektionen, deshalb werden keine ROC-Kurven dazu gezeigt. Tabelle 4.8.2 zeigt die Tensormerkmale, die in den entsprechenden Klassifikatoren verwendet wurden. Die Trainingsparameter und -algorithmen von F11 sind identisch mit M01, mit dem einzigen Unterschied, dass F11 in der Auflösung $\mathcal{F}_{\Delta r} = 1/5$ trainiert wurde. Die minimalen Falsch-Positiven

Parameter						
ROC	**Merkmale**			**Training-Alg.**		
ID	$\mathcal{F}_{\Delta r}$	Typ	Norm.	Stufen	Stufe	Basis
M01	1/12	Haar	Gl. (3.3.3)	54	(6.2)	(3.2)
M02	1/12	Int., Tens.	Gl. (3.3.3,3.3.7)	38	(6.2)	(3.2)
M03	1/7	Int., Tens.	Gl. (3.3.3,3.3.7)	32	(6.2)	(3.2)
M04	1/12	Haar	Gl. (3.3.3)	33	(6.1)	(3.2)
M05	1/5	Int., Tens.	Gl. (3.3.3,3.3.7)	36	(6.2)	(3.2)
M06	1/5	HOGC	$\|\underline{x}\| = 1$	33	(6.2)	(3.2)
M07	1/5	Tens.	Gl. (3.3.7)	35	(6.2)	(3.2)
M08	1/5	Tens.	Gl. (3.3.7)	184	(6.2)	(3.2),Gl. (6.2.9)
M09	1/5	Tens.	Gl. (3.3.7)	77	(6.2)	(3.3)
M10	1/5	Tens.	-	110	(6.2)	(3.2)
Ohne ROC-Kurven						
F09	1/5	Int.	Gl. (3.3.3)	100	(6.2)	(3.2)
F10	1/5	Tens.	Gl. (3.3.11)	98	(6.2)	(3.2)
F11	1/5	Haar	Gl. (3.3.3)	67	(6.2)	(3.2)

Tabelle 4.8.1: Konfigurationen mit unterschiedlichen Merkmalen und Trainingsmechanismen. Die erste Spalte beinhaltet Identifikationsnummern für die entsprechenden ROC-Kurven in Abb. 4.8.1.

Tensormerkmale		
ID	M07, M08, M09, M10, F10 (siehe Tabelle 4.8.1)	
Dim	$\underline{x}^T$	**Formeln**
2	t_4, t_5	o_x, o_y
2	t_{11}, t_{16}	$\overline{i_x^2} + \overline{i_y^2}$, $(\overline{i_x^2} - \overline{i_y^2})^2 - 4\overline{i_{xy}}^2$
3	t_1, t_2, t_3	$\overline{i_x^2}, \overline{i_y^2}, \overline{i_{xy}}$
4	t_4, t_5, t_9, t_{11}	$o_x, o_y, t_e, \overline{i_x^2} + \overline{i_y^2}$
5	t_4, t_5, t_6, t_7, t_8	$o_x, o_y, o_x^2, o_y^2, o_x o_y$
6	$t_1, t_2, t_3, t_{17}, t_{18}, t_{19}$	$\overline{i_x^2}, \overline{i_y^2}, \overline{i_{xy}}, \overline{i_x}, \overline{i_y}, \sqrt{\overline{i_{xy}}}$
10	$t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{15}, t_{17}, t_{18}$	$o_y, o_y, o_x^2, o_y^2, o_x o_y, t_e, t_c, t_\alpha, \overline{i_x}, \overline{i_y}$

Tabelle 4.8.2: Tensormerkmale, die in den Klassifikatoren M07, M08, M09, M10 und F10 verwendet wurden und sich mit den Gleichungen (3.1.14), (3.1.15), (3.1.16), (3.1.18) und (3.1.19) berechnen.

Intensitätsmerkmale		
1	l_1	$\bar{i}$
1	l_3	$\overline{i^2} - (\bar{i})^2$
2	l_1, l_2	$\bar{i}, \overline{i^2}$
2	l_1, l_3	$\bar{i}, \overline{i^2} - (\bar{i})^2$
3	l_1, l_2, l_3	$\bar{i}, \overline{i^2}, \overline{i^2} - (\bar{i})^2$
4	l_1, l_2, l_3, l_4	$\bar{i}, \overline{i^2}, \overline{i^2} - (\bar{i})^2, \sqrt{\overline{i^2} - (\bar{i})^2}$

Tabelle 4.8.3: Intensitätsmerkmale, die in Kaskade F09 verwendet wurden (siehe Tabelle 4.8.1).

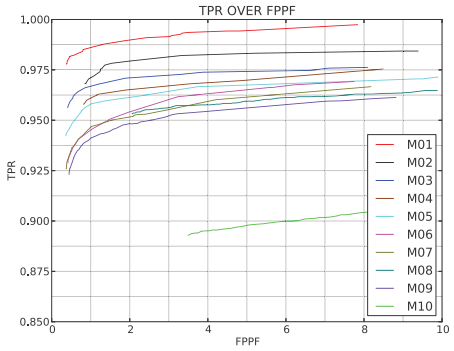
liegen bei 136 Falschdetektionen pro Bild, bei einer Detektionsrate von 97.8 % und einer mittleren Detekionszeit von 3050 ms pro Bild. Dieser Klassifikator ist somit nicht verwertbar. Im Vergleich mit M01 wird allerdings ersichtlich, dass die Haar-ähnlichen Merkmale ab einer bestimmten Auflösung wirksam sind und sich dann hohe Klassifikationsraten erzielen lassen. Die geringe Separationsfähigkeit der Merkmale bei niedriger Auflösung führten dazu, dass sehr viele Basisklassifikatoren beim Boostingprozess zu den Stufenklassifikatoren hinzugefügt wurden, was die hohe mittlere Laufzeit erklärt. Die Haar-ähnlichen Merkmale lassen sich also erst ab einer gewissen Mindestauflösung effektiv verwenden. Für die angestrebte Echtzeitfähigkeit ergibt sich allerdings das Problem, dass bei höheren Auflösungen auch eine höhere Anzahl an initialen Detektionsfenstern überprüft werden muss. Ähnlich verhält es sich mit Kaskade F10, die bis auf die Normierungsmethode identisch mit Kaskade M07 ist. Durch die unterschiedliche Normierung wurden 53 Falsch-Positive pro Bild im Minimum erreicht, bei einer Detektionsrate von 98.6 % und einer mittleren Detektionszeit von 5500 ms.

Diese Art der Normierung lässt sich also für einen laufzeiteffizienten Klassifikator nicht verwenden. Kaskade F09 ist mit intensitätsbasierten Merkmalen mit identischer Normierung, die bei den Haar-ähnlichen Merkmalen verwendet wird, trainiert, allerdings werden die Intensitätsmerkmale innerhalb beliebiger Rechtecke extrahiert. Die minimalen Falsch-Positiven von 17 Fehldetektionen pro Bild werden bei einer Detektionsrate von 93.6 % und einer mittleren Laufzeit von 2300 ms erreicht. Diese Konfiguration ist den anderen ebenfalls unterlegen, da die Intensitätsmerkmale auch bei Extraktion innerlhalb beliebiger Rechtecke eine schwächere Trennfähigkeit besitzen. Alle weiteren ROC-Kurven sind in Abb. 4.8.1 gezeigt, wobei die Detektionsraten

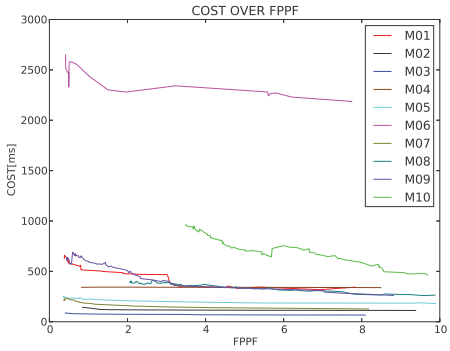
Tensor- und Intensitätsmerkmale		
ID	M05 (siehe Tabelle 4.8.1)	
Dim	$\underline{x}^T$	**Formeln**
2	t_4, t_5	o_x, o_y
2	t_{11}, t_{16}	$\overline{i_x^2} + \overline{i_y^2}$, $(\overline{i_x^2} - \overline{i_y^2})^2 - 4\overline{i_{xy}}^2$
5	t_4, t_5, t_6, t_7, t_8	$o_x, o_y, o_x^2, o_y^2, o_x o_y$
10	$t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{15}, t_{17}, t_{18}$	$o_y, o_y, o_x^2, o_y^2, o_x o_y, t_e, t_c, t_\alpha, \overline{i_x}, \overline{i_y}$
1	l_1	$\overline{i}$
1	l_3	$\overline{i^2} - (\overline{i})^2$
2	l_1, l_2	$\overline{i}, \overline{i^2}$
2	l_1, l_3	$\overline{i}, \overline{i^2} - (\overline{i})^2$
3	l_1, l_2, l_3	$\overline{i}, \overline{i^2}, \overline{i^2} - (\overline{i})^2$
4	l_1, l_2, l_3, l_4	$\overline{i}, \overline{i^2}, \overline{i^2} - (\overline{i})^2, \sqrt{\overline{i^2} - (\overline{i})^2}$
2	l_1, t_9	$\overline{i}, t_e$
2	l_1, t_{10}	$\overline{i}, t_c$
2	l_1, t_{11}	$\overline{i}, \overline{i_x^2} + \overline{i_y^2}$
2	l_1, t_{15}	$\overline{i}, t_\alpha$
2	l_1, t_{16}	$\overline{i}, (\overline{i_x^2} - \overline{i_y^2})^2 - 4\overline{i_{xy}}^2$
3	l_1, l_4, t_9	$\overline{i}, \sqrt{\overline{i^2} - (\overline{i})^2}, t_e$
4	l_1, l_4, t_4, t_5	$\overline{i}, \sqrt{\overline{i^2} - (\overline{i})^2}, o_x, o_y$
6	$l_1, l_4, t_4, t_5, t_9, t_{11}$	$\overline{i}, \sqrt{\overline{i^2} - (\overline{i})^2}, o_x, o_y, t_e, \overline{i_x^2} + \overline{i_y^2}$
3	l_1, l_2, t_{10}	$\overline{i}, \overline{i^2}, t_c$
3	l_1, l_2, t_9	$\overline{i}, \overline{i^2}, t_e$
3	l_1, l_2, t_{16}	$\overline{i}, \overline{i^2}, (\overline{i_x^2} - \overline{i_y^2})^2 - 4\overline{i_{xy}}^2$
4	l_1, l_2, t_4, t_5	$\overline{i}, \overline{i^2}, o_x, o_y$
8	$l_1, l_2, t_1, t_2, t_4, t_5, t_9, t_{11}$	$\overline{i}, \overline{i^2}, \overline{i_x^2}, \overline{i_y^2}, o_x, o_y, t_e, \overline{i_x^2} + \overline{i_y^2}$

Tabelle 4.8.4: Gemischte Intensität- und Tensormerkmale, die für die Kaskade M05 verwendet wurden (siehe Tabelle 4.8.1).

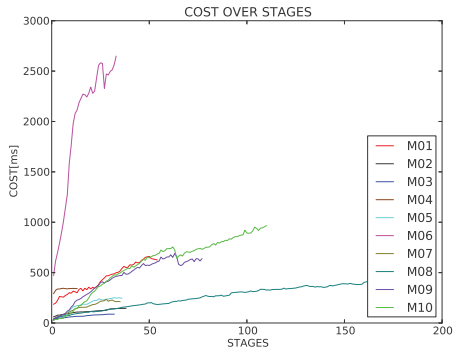
in Abb. 4.1(a) und die entsprechenden mittleren Laufzeiten bei gegebenen Falsch-Positiven pro Bild in Abb. 4.1(b) zu sehen sind. Aus diesen beiden Abbildungen lassen sich die Klassifikatoren im Bezug auf Klassifikationsleistung und Laufzeiteffizienz beurteilen. In Abb. 4.1(c) sind die mittleren Laufzeiten der Klassifikatoren mit zunehmender Anzahl an Stufen dargestellt, wobei sich



(a)



(b)



(c)

Abbildung 4.8.1: ROC-Kurven der unterschiedlichen Klassifikatoren aus Tabelle 4.8.1 (a): Detektionsraten bei entsprechenden Falsch-Positiven pro Bild (b): Mittlere Laufzeiten pro Bild bei entsprechenden Falsch-Positiven pro Bild (c): Mittlere Laufzeiten mit zunehmender Anzahl an Stufenklassifikatoren

die initialen Kosten für die erste Stufe ablesen lassen. Generell sind die Kosten für den ersten Stufenklassifikator bei der mittleren Auflösung bedeutend geringer. Eine Ausnahme bilden die HOG-Merkmale (Id M06), die auch mit mittlerer Auflösung die höchsten initialen Kosten besitzen. Darauf folgen die Haar-ähnlichen Merkmale in der Auflösung $\mathcal{F}_{\Delta r} = 1/12$, wobei das erweiterte Gradientenboosting mit den aktualisierten A-priori-Wahrscheinlichkeiten der Bayes-Entscheidungsfunktionen niedrigere Kosten in der ersten Stufe verursacht, jedoch steigen die Kosten mit zunehmenden Stufen bei nahezu allen

Kaskaden, bedingt durch die wachsende Anzahl an Basisklassifikatoren in den entsprechenden Stufen, stark an.

Kaskade M10 extrahiert die Merkmale ohne Normierung und ist sonst identisch mit M07. Beim Vergleich der Kurven wird ersichtlich, dass diese Methode sowohl eine schlechtere Detektionsleistung als auch eine höhere mittlere Laufzeit mit größeren Schwankungen erzielt. Durch die gestufte Klassifikation hat die Art der Normierung einen großen Einfluss auf den Gesamtklassifikator. Ohne Normierung bleibt eine höhere Variabilität in den extrahierten Merkmalen erhalten, was für die wiederholte Klassifikation in der Kaskade von Nachteil ist. Der Detektor M09 ist bis auf die Optimierungsmethode der Basisklassifikatoren und des Ensembleklassifikators, identisch mit M07, wobei die multiple lineare Regression im Gegensatz zur Fischer LDA verwendet wurde. Es resultiert eine geringfügig schlechtere Detektionsleistung bei doppelten gemittelten Laufzeitkosten pro Bild. Aus diesem Grund wird für alle weiteren Versuche die Fisher LDA verwendet. M08 ist bis auf eine quadratische Erweiterung der Merkmale, die in Kapitel 6, Abschnitt 6.2.1 erklärt ist und einer maximalen Anzahl von lediglich 32 Basisklassifikatoren im Gegensatz zu 256, identisch mit M07. Die Detektionsleistungen sind ähnlich, allerdings hat sich die erhoffte Reduktion der Laufzeit nicht eingestellt, die mittleren Laufzeiten sind deutlich höher. Kaskade M06 wurde mit HOG-ähnlichen Merkmalen, die innerhalb den beliebigen Rechtecken extrahiert wurden, trainiert. Es wurden zwei Varianten verwendet, wobei das Rechteck in einer Variante in 8x8 Unterbereiche und in einer weiteren in 4x4 Unterbereiche unterteilt wird, in denen die Kantenrichtungen ermittelt und in ein 9-dimensionales Histogramm akkumuliert werden. Der Optimierungsalgorithmus wählt, wie bei den anderen Merkmalen, aus diesen zwei Varianten und den unterschiedlichen Rechteckformen die entsprechenden Merkmale beim Gradientenboosting aus. Wie erwartet erzielen diese Merkmale für eine angestrebte Echtzeitfähigkeit zu hohe mittlere Laufzeiten. Die ursprünglichen HOG-Merkmale [31] beinhalten noch eine überlappende Konfiguration von mehreren HOG-Zellen, was zur Stabilität beiträgt aber gleichzeitig einen noch höheren Rechenaufwand bedeutet. Die Konfiguration mit Tensor- und Intensitätsmerkmalen M03 erzielt bei der Auflösung von $\mathcal{F}_{\Delta r} = 1/7$ eine der besten Detektionsergebnisse mit 96.7%@1FPPF und erzeugt gleichzeitig die geringsten Berechnungskosten mit 76ms@1FPPF. Eine Erhöhung der Auflösung auf $\mathcal{F}_{\Delta r} = 1/12$ bei M02 hat eine Erhöhung der Detektionsraten auf 97.2%@1FPPF zur Folge, allerdings mit einer höheren mittleren Laufzeit von 135ms@1FPPF. M01 zeigt die Überlegenheit des erweiterten, konfidenzba-

sierten Gradientenboostings im Gegensatz zu AdaBoost (M04), mit dem eine Detektionsrate von 96.2 % bei einem Falsch-Positiven erreicht wird. Mit M01 werden außerordentlich hohe Detektionsraten von 98.6 % und 500 ms mittlerer Laufzeit bei einem Falsch-Positiven pro Bild erzielt. Allerdings ist eine Echtzeitanwendung durch die hohe Anzahl an Detektionsfenstern bei M01 und M04 durch die hohe Auflösung von $\mathcal{F}_{\Delta r} = 1/12$ nicht möglich. Außerdem ist hervorzuheben, dass mit der mittleren Merkmalsauflösung und den kombinierten Merkmalen identische minimale Falsch-Positiv-Detektionen erzeugt werden, wie mit der hohen Auflösung. Das gute Ergebnis der im Rahmen dieser Arbeit entwickelten kombinierten Intensitäts- und Tensormerkmale mit einer Detektionsrate von 96.7% bei einem Falsch-Positiven pro Bild und einer mittleren Laufzeit von 76 ms wird in den folgenden Versuchen weiterverfolgt. Die Idee ist es, durch eine noch geringere Auflösung und durch, auf Auflösungshierarchien basierenden Detektionsmechanismen, die den Übergang zu höheren Auflösungen ermöglichen, Echtzeitfähigkeit zu erreichen.

4.8.3 Ressourcenoptimierung

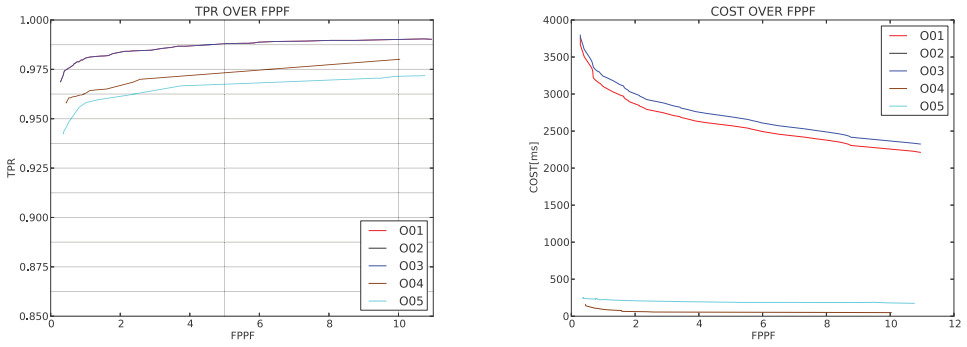
Bei diesem Versuch sollen die Auswirkungen der unterschiedlichen Gewichtungsfunktionen von Gl. 4.1.3, die zur Auswahl des optimalen Klassifikators in Alg. (4.1), Schritt 3c genutzt wird, getestet werden. Wie in der vorangehenden Auswertung werden in diesem Versuch die Klassifikatoren mit einer Auflösung von $\mathcal{F}_{\Delta r} = 1/5$ trainiert um Laufzeiteffekte, die aus den Auflösungshierarchien resultieren, zu vermeiden. Die Konfigurationen der Klassifikatoren sind identisch mit Abschnitt 4.8.2, wobei die gemischten Tensor- und Identitätsmerkmale aus Tabelle 4.8.3 verwendet wurden, da diese das beste Verhältnis aus Klassifikationsleistung und mittlerer Laufzeit erzielten. Um eine größere Vielfalt in der Komplexität der Kandidatenklassifikatoren zu erzeugen, die mit den unterschiedlichen Gewichtungsfunktionen zur Auswahl stehen, werden im Gegensatz zum obigen Versuch, bei dem nur einzelne Merkmalsrechtecke zur Extraktion verwendet wurden, zusätzlich zu den einzelnen Rechtecken in allen möglichen Formen (siehe Abb. 3.11(a)) Maskenmerkmale verwendet. Dies geschieht, weil es bei dieser Auflösung zu rechenintensiv wäre, alle zweistelligen Kombinationen der Rechtecke zu verwenden. Laut Tabelle 3.1.1 ist die Anzahl der einstelligen Kombinationen 225, im Gegensatz dazu existieren bereits 25200 zweistellige Kombinationen. Mit den 23 unterschiedlichen Merkmalstypen würden daraus 579600 Kandidatenklassifikatoren entstehen.

$$
\begin{array}{cccc}
5\times 5 & 4\times 4 & 3\times 3 & 2\times 2 \\
\begin{pmatrix} \underline{e} & \underline{e} & \underline{e} & \underline{e} & \underline{e} \\ \underline{e} & \underline{e} & \underline{e} & \underline{e} & \underline{e} \\ \underline{e} & \underline{e} & \underline{e} & \underline{e} & \underline{e} \\ \underline{e} & \underline{e} & \underline{e} & \underline{e} & \underline{e} \\ \underline{e} & \underline{e} & \underline{e} & \underline{e} & \underline{e} \end{pmatrix} &
\begin{pmatrix} \underline{e} & \underline{e} & \underline{e} & \underline{e} \\ \underline{e} & \underline{e} & \underline{e} & \underline{e} \\ \underline{e} & \underline{e} & \underline{e} & \underline{e} \\ \underline{e} & \underline{e} & \underline{e} & \underline{e} \end{pmatrix} &
\begin{pmatrix} \underline{e} & \underline{e} & \underline{e} \\ \underline{e} & \underline{e} & \underline{e} \\ \underline{e} & \underline{e} & \underline{e} \end{pmatrix} &
\begin{pmatrix} \underline{e} & \underline{e} \\ \underline{e} & \underline{e} \end{pmatrix}
\end{array}
$$

Abbildung 4.8.2: Zusätzlich verwendete Maskenmerkmale, aus denen die Merkmalsvektoren $\underline{e}$ aus den entsprechenden unterteilten Regionen extrahiert und zu einem großen Merkmalsvektor verkettet werden.

Ressourcenoptimierung		
ID	**Gewichtungsfunktion**	**Beschreibung**
O01	$C(h) = e(h)c(h)^{P(\omega_1\|h(\underline{z})=\omega_0)}$	FPR als Exponent der Kosten (Gl.(4.1.7))
O02	$C(h) = c(h)$	Nur Berechnungskosten
O03	$C(h) = e(h)$	Nur Falsch-Positiv-Rate (Gl.(4.1.4))
O04	$C(h) = e(h)c(h)$	“Wahrscheinliche Operationen” (Gl.(4.1.5))
M05	$C(h) = e(h)c(h)$	Nur einstellige Rechteckkomb. (Tab. 4.8.1)

Tabelle 4.8.5: Unterschiedliche Gewichtungsfunktionen in Gl. (4.1.3) und die entsprechenden ROC-Ids aus Abb. 4.8.3.



(a) (b)

Abbildung 4.8.3: ROC-Kurven für die unterschiedlichen Gewichtungsfunktionen aus Tabelle 4.8.5. (a): Detektionsraten über den Falsch-Positiven pro Bild. (b): Mittlere Laufzeiten über den Falsch-Positiven pro Bild.

Abb. 4.8.2 zeigt die zusätzlich zu den einfachen Rechtecken verwendeten Maskenmerkmale mit festen Konfigurationen. In Tabelle 4.8.5 sind die unterschiedlichen Gewichtungsfunktionen für die Auswahl der Kandidatenklassifikatoren mit Alg. (4.1) gelistet. Bis auf den Klassifikator M05 wurden die zusätzlichen Maskenkonfigurationen zur Extraktion der Merkmale verwendet.

Die erreichten minimalen Detektionsraten und die entsprechenden mittleren Detektionszeiten pro Bild sind in Tabelle 4.8.6 zu sehen. Da relativ

ID	**Stufen**	**min. TPR@FPPF**	**Det.-Zeit pro Bild**
O01	83	98%@0.68	5503 ms
O02	73	98%@17	2154 ms
O03	83	98%@0.68	5527 ms
O04	23	96%@0.45	279 ms
M05	36	94%@0.37	246 ms

Tabelle 4.8.6: Minimale Detektionsraten und entsprechende mittlere Laufzeiten pro Bild in ms. Die Gewichtungsfunktionen sind aus Tabelle 4.8.5 zu entnehmen.

geringe Falsch-Positiv-Raten der einzelnen Kandidatenklassifikatoren auftreten, die in die Kostengewichtung einfließen (siehe Tabelle 4.8.5), sind keine Unterschiede in den Klassifikatoren O01 und O03 zu erkennen. Die Detektionsraten fallen nicht unter 98 %, allerdings sind die mittleren Laufzeiten von über 5 Sekunden für eine Echtzeitanwendung unbrauchbar. Bei O02 und einer Gewichtung, die nur auf den Berechnungskosten basiert, endet das Training bei einer hohen Anzahl an Falsch-Positiven von 17. Dieser Effekt lässt sich damit erklären, dass immer nur die kostengünstigen Klassifikatoren gewählt werden, die jedoch kaum zur Reduzierung der Falsch-Positiven beitragen. Die mittlere Laufzeit ist zwar besser, allerdings mit über 2 Sekunden immer noch für eine Echtzeitanwendung unbrauchbar. Mit der Gewichtung nach Gl. (4.1.5) deren Einheit mit “wahrscheinlich auftretenden Operationen” interpretiert werden kann, sinken die mittleren Laufzeiten um den Faktor 20, bei leichtem Absinken der Detektionsrate auf 96% mit den Maskenmerkmalen (O04) und 94% ohne (M05) auf 280 bzw. 250 ms. Der Unterschied in den Detektionsraten ist bei diesem Versuchen durch das Gradientenboostingverfahren, das für die Stufenklassifikatoren verwendet wurde, zwischen O04 und M05 nicht stark ausgeprägt. Allerdings ist bei einfacheren Trainingsverfah-

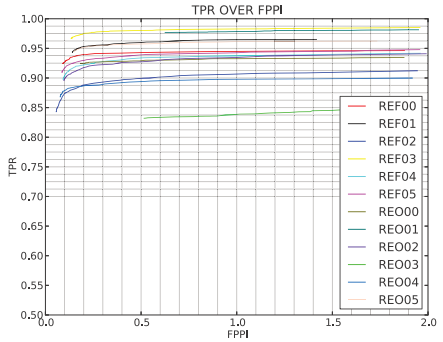
ren für die Stufenklassifikatoren die Verwendung von komplexeren Merkmalen, wie die Maskenmerkmale, die einzige Möglichkeit, die Falsch-Positiven so stark zu reduzieren, dass der Klassifikator tatsächlich unter Realweltbedingungen verwendet werden kann. Mit den mächtigeren Maskenmerkmalen benötigt O04 insgesamt 23 Stufen, im Gegensatz zu M05 mit 36 Stufen. Obwohl durch die Boostingverfahren sehr hohe Detektionsraten erzielt werden können, entstehen mit einer hohen Anzahl von Basisklassifikatoren, die wiederum nötig sind um die Falsch-Positiven zu reduzieren, hohe mittlere Laufzeiten. In den folgenden Versuchen wird gezeigt, wie sich die Laufzeiten weiter senken lassen, indem die Stufenklassifikatoren ohne Boosting-Verfahren, sondern mit der ressourcenoptimierten Gewichtung des Klassifikators O04 aus Tabelle 4.8.5 und den zusätzlichen Maskenmerkmalen mit einer sequentiellen Merkmalsauswahl trainiert werden.

4.8.4 Auflösungen

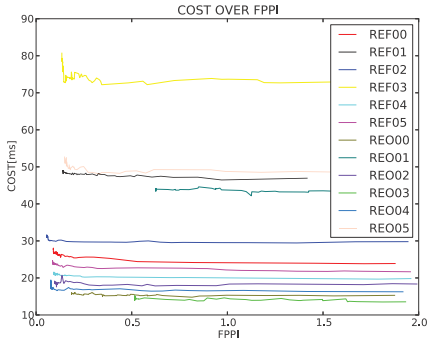
Bei dieser Auswertungsreihe sollen die Auswirkungen der Auflösungen auf die Klassifikatoren getestet werden. Um die mittleren Laufzeiten weiter zu reduzieren werden die Stufenklassifikatoren nicht wie im obigen Versuch mit dem Gradientenboosting, sondern mit der sequentiellen Vorwärtsauswahl der Merkmale (siehe Alg. (4.2)) trainiert.

Die allgemeinen Parameter des Trainings sind identisch mit Tabelle 4.7.1 und 4.7.2, wobei bei den schwellwertbasierten Entscheidungsfunktionen Alg. (3.9) verwendet wurde und bei den Bayes-Entscheidungsfunktionen kein Signifikanzniveau eingestellt wurde. Als Merkmale wurden, wie im vorhergehenden Abschnitt beschrieben, eine Kombination aus den Tensor- und Intensitätsmerkmalen verwendet. Die Detektion wird auf dem ganzen Bild ausgeführt und als Detektionsalgorithmus wird Alg. (4.6) mit der adaptiven Feinsuche (siehe Alg. (4.7)) und dem Standardsampling, also $\xi = 1$ in Gl. (4.4.1), verwendet.

Abb. 4.8.4 zeigt die Detektionsraten als ROC-Kurve (Abb. 4.8.4(a)) und die entsprechenden mittleren Laufzeiten (Abb. 4.8.4(b)). Die verwendeten Parameter sind in Abb. 4.8.4(c) gelistet. In Abb. 4.8.4(d) sind die Klassifikatoren nach absteigenden Detektionsraten bei einer Falsch-Positiv-Detektion und entsprechenden mittleren Laufzeiten sortiert. Zusätzlich sind die Detektionsraten und Laufzeiten der AdaBoost-Kaskade (Nummer VJH_0 mit einer Auflösungsstufe $\Delta r_{\frac{1}{12}}$ und VJH_1 mit zwei Auflösungsstufen $\Delta r_{\frac{1}{6}}$ und $\Delta r_{\frac{1}{12}}$) mit den Konfigurationen in Tabelle 4.7.4 und 4.7.5 gelistet.



(a)



(b)

Parameter				
			Entscheidung	
ID	Auflösungen $\mathcal{F}_{\Delta r}$	Stufen	Type	Einträge
REF00	$1/4 \ldots 1/6, 1/12$	143	$\max(P_s(\omega_i))$	256
REF01	$1/5, 1/6, 1/12$	143	$\max(P_s(\omega_i))$	256
REF02	$1/2 \ldots 1/6, 1/12$	124	$\max(P_s(\omega_i))$	256
REF03	$1/6, 1/12$	168	$\max(P_s(\omega_i))$	256
REF04	$1/3 \ldots 1/6, 1/12$	138	$\max(P_s(\omega_i))$	256
REF05	$1/3 \ldots 1/12$	118	$\max(P_s(\omega_i))$	256
REO00	$1/4 \ldots 1/12$	113	$\Theta(x_s - \theta_s)$	1024
REO01	$1/6 \ldots 1/12$	101	$\Theta(x_s - \theta_s)$	1024
REO02	$1/3 \ldots 1/12$	112	$\max(P_s(\omega_i))$	256
REO03	$1/3 \ldots 1/12$	82	$\Theta(x_s - \theta_s)$	1024
REO04	$1/4 \ldots 1/12$	145	$\max(P_s(\omega_i))$	256
REO05	$1/6 \ldots 1/12$	141	$\max(P_s(\omega_i))$	256

(c)

Eval					Parameter
TPR@1	Cost@1	TPR_{min}	FPPF_{min}	Cost	ID
98.2 %	74 ms	96.6 %	0.134	81 ms	REF03
97.8 %	44 ms	97.6 %	0.625	44 ms	REO01
96.4 %	46 ms	94.3 %	0.140	49 ms	REF01
96.2 %	343 ms	95.8 %	0.822	351 ms	VJH_0
96.1 %	40 ms	94.4 %	0.149	53 ms	REO05
95.4 %	1115 ms	89.6 %	0.101	1121 ms	VJH_1
94.5 %	24 ms	92.4 %	0.090	28 ms	REF00
94.2 %	22 ms	90.9 %	0.085	25 ms	REF05
93.7 %	20 ms	89.7 %	0.092	22 ms	REF04
93.4 %	18 ms	89.5 %	0.096	19 ms	REO02
93.2 %	15 ms	92.4 %	0.185	16 ms	REO00
90.7 %	30 ms	84.3 %	0.056	31 ms	REF02
89.8 %	17 ms	86.8 %	0.077	19 ms	REO04
83.8 %	14 ms	83.2 %	0.515	15 ms	REO03

(d)

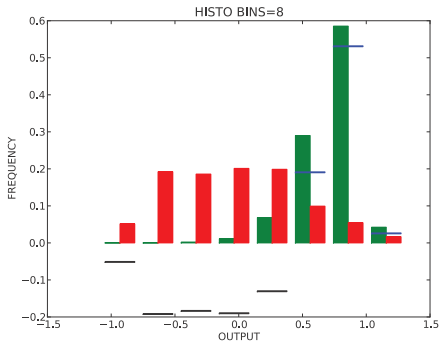
Abbildung 4.8.4: Auswertung der Klassifikatoren mit unterschiedlichen Auflösungen (a): ROC-Kurven über den Falsch-Positiven pro Bild (b): Mittlere Laufzeiten der Detektionen über den Falsch-Positiven pro Bild (c): Verwendete Parameter mit unterschiedlichen Auflösungen und Klassifikationsfunktionen (d): Nach Detektionsraten absteigend sortierte Klassifikatoren bei einer Falsch-Positiv-Detektion pro Bild

Dieser Versuch zeigt, dass aus feineren Auflösungen bessere Detektionsraten resultieren, allerdings bei erhöhtem Rechenaufwand. Schon ab einer Auflösung von $\Delta r_{\frac{1}{3}}$ lassen sich effiziente Klassifikatoren erzeugen, deren mittlere Laufzeiten unter 25 ms pro Bild liegen. Beim direkten Vergleich zwischen Bayes- und Schwellwertentscheidungen erzielen die Bayes-Entscheidungen mit Frequenztabellen mit 256 Einträgen eine bessere Detektionsleistung bei identischen Auflösungen. Zusätzlich lassen sich mit den Bayes-Entscheidungsfunktionen fusionierte Konfidenzen der Stufenklassifika-

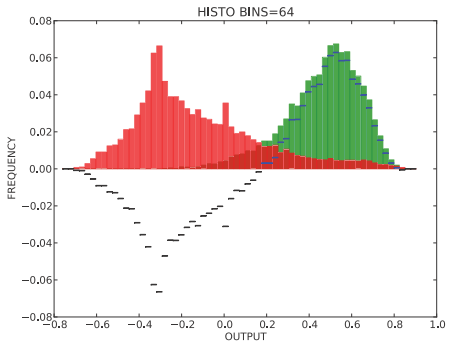
toren berechnen. Ein guter Kompromiss zwischen Detektionsleistung und Laufzeit wird bei Auflösungen mit kleinen Sprüngen, wie z.B. Klassifikator REF05 erzielt. Bei den AdaBoost-Kaskaden konnten erst ab einer Auflösung von $\Delta r_{\frac{1}{6}}$ funktionsfähige Klassifikatoren erzeugt werden. Eine Erklärung dafür ist das Prinzip zur Neugewichtung der Muster, das bei AdaBoost nur von einem einzelnen schwachen Klassifikator abhängt, der bei einer zu geringen Auflösung keine ausreichende Klassifikationsleistung besitzt und somit die Neugewichtung keinen ausreichenden Einfluß erzielt.

4.8.5 Frequenztabellen und Detektionsalgorithmen

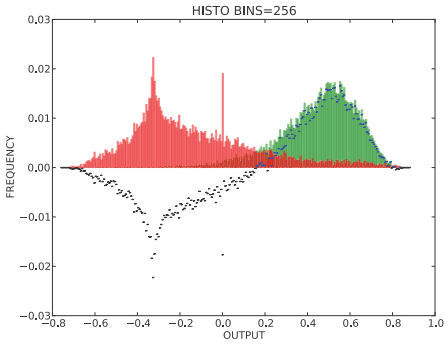
In diesem Versuch sollen die Auswirkungen der Anzahl der Einträge in den Frequenztabellen auf die Detektionsleistung und mittlere Laufzeiten der Klassifikatoren untersucht werden. Je mehr Einträge verwendet werden, desto mehr nicht zusammenhängende Entscheidungsregionen können erzeugt werden. Allerdings können dann auch Einträge entstehen, die zu wenig Beispiele enthalten und bei Testdaten schlechter generalisieren. Abb. 4.8.5 zeigt die Frequenztabellen der Entscheidungsfunktionen mit unterschiedlicher Anzahl an Einträgen der ersten Stufen der Kaskaden. Grüne Einträge stammen von der Objektklasse und rote Einträge entsprechen der Hintergrundklasse. Die Entscheidungsfunktion ist durch blaue Balken für eine Positiventscheidung und durch schwarze Balken für eine Negativentscheidung, d.h. für Werte kleiner Null, gekennzeichnet. Bei Abb. 4.8.5(a) fällt auf, dass die Hintergrundklasse eine sehr flache Verteilung mit mehreren Maxima aufweist. Je mehr Einträge verwendet werden, desto besser sind die Maxima der Verteilungen zu erkennen. Bei Abb. 4.8.5(c) und 4.8.5(d) ist eine zunehmende Streuung der Werte der Entscheidungsfunktion zwischen benachbarten Einträgen zu sehen. Bei diesem Versuch werden außerdem die unterschiedlichen Detektionsalgorithmen getestet, wobei beim rekursiven Depth-First-Detektionsalgorithmus Alg. (4.6) eine konstante Anzahl an acht Nachbarn verwendet wird (siehe Abb. 4.3(b)). Die entsprechenden Parameter des Versuchs sind in Tabelle 4.6(a) gelistet. Außerdem wird die adaptive Feinsuche verwendet (siehe Alg. (4.7)), wobei abhängig von der aktuellen Auflösungsstufe eine wachsende Anzahl an Nachbarn überprüft wird (siehe Gl. (4.4.3)), wie es in Abb. 4.3(a) dargestellt ist. Es sind die Auflösungsstufen $\{\Delta r_{\frac{1}{3}}, \ldots, \Delta r_{\frac{1}{6}}, \Delta r_{\frac{1}{12}}\}$ verwendet worden. Bei einer Fehldetektion in Auflösungsstufe $\Delta r_{\frac{1}{3}}$ werden somit acht Nachbarn überprüft, in Auflösungsstufe $\Delta r_{\frac{1}{4}}$ werden 16 Nachbarn über-



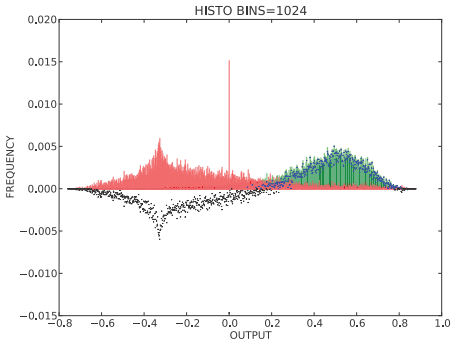
(a) 8 Einträge



(b) 64 Einträge



(c) 256 Einträge



(d) 1024 Einträge

Abbildung 4.8.5: Frequenztabellen der ersten Stufenklassifikatoren mit unterschiedlicher Anzahl an Einträgen. Die Spitze in Abb. (c) und (d) rührt von einer Abbildung von Detektionsfenstern mit konstanten Werte auf den Wert 0, wie z.B. Regionen im Himmel oder auf der Straße.

prüft, usw . Die Ergebnisse der entsprechenden Versuche mit der konstanten Feinsuche aus Tabelle 4.6(a) sind in Abb. 4.8.7(a) für die ROC-Kurve der Detektionsraten und in Abb. 4.8.7(c) abgebildet. Die Ergebnisse der Versuche mit der adaptiven Feinsuche aus Tabelle 4.6(b) sind in Abb. 4.8.7(b) und Abb. 4.8.7(d) dargestellt.

Die Falsch-Positiv-Raten pro Bild wurden von 0 bis 20 auf der X-Achse gewählt, damit die Entwicklung der Raten bei der konstanten Feinsuche erkennbar ist. Diese fallen im Gegensatz zum adaptiven Sampling stark ab. Die Kaskade mit adaptiver Feinsuche und 256 Einträgen (blaue Kurve in

Trainingsparameter			
ID	Funktion	Einträge	Detektion
CB01	$\max(P_s(\omega_i))$	1024	Alg. (4.6), (4.8)
CB03	$\max(P_s(\omega_i))$	256	Alg. (4.6), (4.8)
CB04	$\max(P_s(\omega_i))$	64	Alg. (4.6), (4.8)
CB05	$\max(P_s(\omega_i))$	32	Alg. (4.6), (4.8)
CB06	$\max(P_s(\omega_i))$	8	Alg. (4.6), (4.8)
CTHR	$\Theta(x_s - \theta)$	1024	Alg. (4.6), (4.8)

(a)

Trainingsparameter			
ID	Funktion	Einträge FT	Detektion
AB01	$\max(P_s(\omega_i))$	1024	Alg. (4.6), (4.7)
AB02	$\max(P_s(\omega_i))$	512	Alg. (4.6), (4.7)
AB03	$\max(P_s(\omega_i))$	256	Alg. (4.6), (4.7)
AB04	$\max(P_s(\omega_i))$	64	Alg. (4.6), (4.7)
AB05	$\max(P_s(\omega_i))$	32	Alg. (4.6), (4.7)
AB06	$\max(P_s(\omega_i))$	8	Alg. (4.6), (4.7)
ATHR	$\Theta(x_s - \theta)$	1024	Alg. (4.6), (4.7)

(b)

Abbildung 4.8.6: Parameter mit unterschiedlichen Entscheidungsfunktionen und Detektionsalgorithmen. Bei den Funktionen bedeutet $\max(P_s(\omega_i))$ die MAP-Entscheidung, die mit einer Frequenztabelle mit N Einträgen in der Frequenztabelle (siehe Spalte "Einträge FT") modelliert wird. Die Anzahl der Einträge sind für alle Stufen identisch. Die Formel $\Theta(x_s - \theta)$ bedeutet eine Schwellwertentscheidung in der Stufe s. Die Frequenztabelle wird zur Berechnung des Schwellwertes verwendet.

Abb. 4.8.7(b)) erzielt im Gegensatz zur identischen Kaskade mit konstantem Sampling (schwarze Kurve in Abb. 4.8.7(a)) eine um acht Prozent bessere Detektionsrate bei fünf Falsch-Positiven pro Bild. Vergleicht man die Anzahl der Einträge der Kaskaden, so fällt auf, dass bei beiden Detektionsalgorithmen die Frequenztabellen mit 256 Einträgen die besten Detektionsraten erzielen. Die Klassifikatoren mit 1024 Einträgen (rote Kurven) erzielen schlechtere Detektionsraten, wobei sehr effektive mittlere Laufzeiten erreicht werden. Ein guter Kompromiss wird mit der adaptiven Feinsuche und 256 bzw. 512 Einträgen erreicht.

4.8.6 Optimierte Merkmale mit sequentieller Vorwärtsauswahl

Bei diesem Versuch wurde eine Optimierung der Merkmale durch Programmiermethoden wie z.B. Loop-Unrolling vorgenommen. Außerdem wurden die Merkmalskombinationen, die oft ausgewählt werden mit einem einzelnen Funktionsaufruf extrahiert, da bei der ursprünglichen flexiblen Version, in der alle möglichen Kombinationen von Merkmalen kombiniert werden können, jedes Merkmal mit einem Funktionspointeraufruf extrahiert wird. Die Resultate sollen zeigen, wie stark sich die Optimierung auf die mittleren Laufzeiten und auf die Detektionsleistung auswirkt. Da die Merkmale mit der

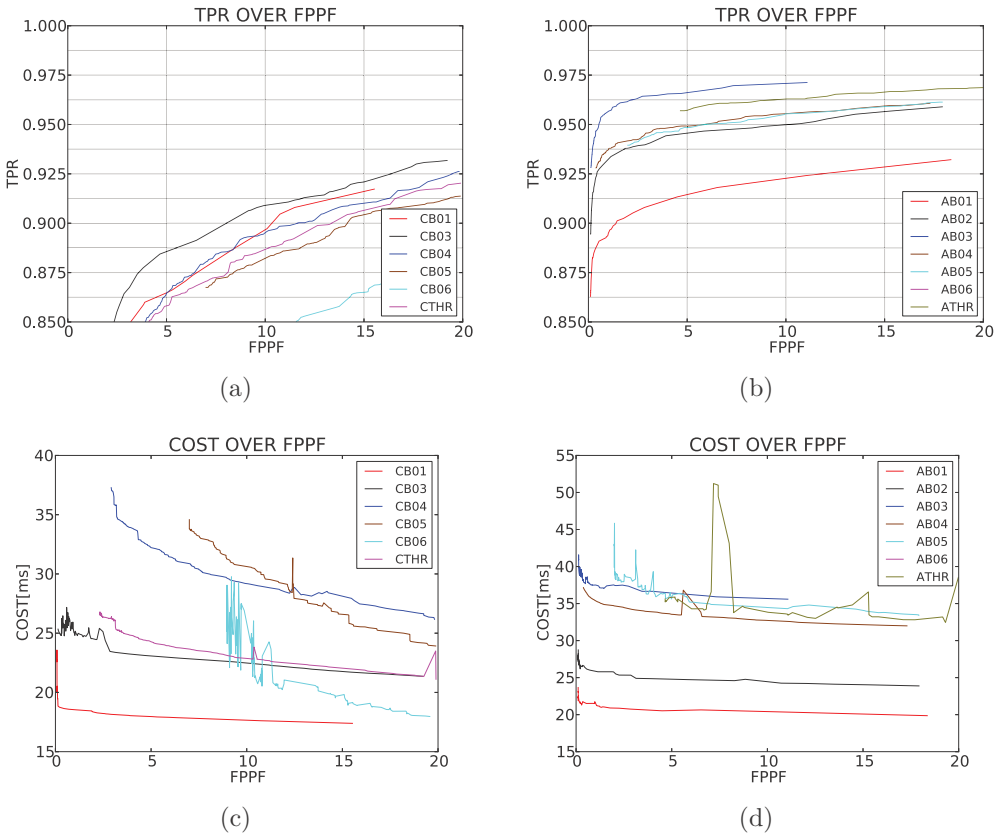


Abbildung 4.8.7: Ergebnisse der Versuche mit konstanter und adaptiver Feinsuche und unterschiedlicher Anzahl an Einträgen in den Frequenztabellen (a):ROC der Detektionsraten aus Tabelle 4.6(a) (b): ROC der Detektionsraten aus Tabelle 4.6(b) (c): ROC der mittleren Laufzeiten aus Tabelle 4.6(a) (d): ROC der mittleren Laufzeiten aus Tabelle 4.6(b)

sequentiellen Vorwärtsauswahl (siehe Alg. (4.2)) ausgewählt werden, werden außerdem unterschiedliche Fehlerschwellen $e_{\mathrm{SFS}}(\omega_0)$ getestet die als maximale Auftrittswahrscheinlichkeiten der Hintergrundklasse zum Abbruchkriterium der sequentiellen Hinzunahme der Merkmale dienen. Als maximale Anzahl der Iterationen wurde $t_{\max} = 8$ gewählt. Alle Klassifikatoren werden mit den Bayes-Entscheidungsfunktionen, die mittels Frequenztabellen mit 256 Einträgen modeliert werden, trainiert. Die Kaskade wird mit auflösungsspezifischen Merkmalen mittels Alg. (4.5), bestehend aus einer Kombination aus

Intensitäts- und Tensormerkmalen, trainiert, wobei das Abbruchkriterium innerhalb einer Auflösungsstufe von $e_{\text{stop}}(\omega_0) = 0.95$ gewählt wurde. Es wurden Auflösungsstufen von $\mathcal{F}_{\Delta r} = \{\Delta r_{\frac{1}{3}}, \ldots, \Delta r_{\frac{1}{12}}\}$ verwendet. Abb. 4.8.8 zeigt die

Trainingsparameter		
ID	$e_{\text{SFS}}(\omega_0)$	**Stufen**
SFT00	0.3	143
SFT01	0.4	123
SFT02	0.8	133
SFT03	0.9	122
SFT04	0.6	120
SFT05	0.7	127
SFT06	1.0	177
SFT07	0.5	115

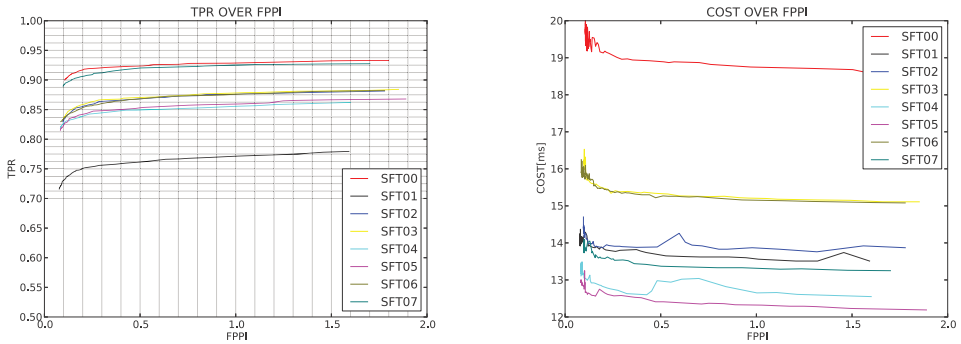
(a)

Evaluierung				**Parameter**
TPR@1	Cost@1	TPR_{min}	FPPI_{min}	ID
0.928	19 ms	0.900	0.106	SFT00
0.925	13 ms	0.889	0.097	SFT07
0.878	15 ms	0.836	0.099	SFT03
0.876	15 ms	0.829	0.084	SFT06
0.875	14 ms	0.829	0.095	SFT02
0.859	12 ms	0.814	0.083	SFT05
0.855	13 ms	0.818	0.083	SFT04
0.771	13 ms	0.715	0.077	SFT01

(b)

Abbildung 4.8.8: Optimierte Merkmale mit dem SFS-Auswahlverfahren
(a): Parameter des Alg. (4.2)
(b): Absteigend sortierte Detektionsraten der Klassifikatoren mit den Parameterkonfigurationen aus Tabelle (a)

Tabellen mit den Parameterkonfigurationen (Tab. 4.8.8(a)) und die entsprechenden absteigend sortierten Detektionsraten (Tab. 4.8.8(b)). Die ROC-Kurven der Detektionsraten und mittleren Laufzeiten sind in Abb. 4.8.9 dargestellt. Bei Betrachtung der Detektionsraten und Laufzeiten fällt auf, dass die Kombination SFT07 bei identischer Detektionsleistung eine deutlich kürzere mittlere Laufzeit erzielt, wobei bei einer Detektionsleistung von 92.5 % eine mittlere Laufzeit von 13 ms erreicht wird. Dies entspricht einer Aktualisierungsrate von über 75 FPS. Da die Detektionsraten einzelbildbasiert sind, ist die Detektionsrate für Bildsequenzen, wie sie im mobilen Fahrbetrieb vorkommen ausreichend. Ein reales Schildobjekt wird bei einer Sequenz von zehn Bildern neun mal korrekt erkannt. Insgesamt sind die Laufzeiten kürzer als im vorangehenden Versuch. Die Detektionsraten sind geringfügig schlechter, da mit den optimierten Merkmalen nicht alle Merkmalskombinationen implementiert wurden. Allerdings sollte die Tendenz erkennbar sein, dass bei kleineren Fehlerschwellen $e_{\text{SFS}}(\omega_0)$ bessere Detektionsraten und längere mittlere Laufzeiten resultieren, da dann mehr Merkmale durch die sequentielle Auswahl hinzugefügt werden. Diese Tendenz ist allerdings nicht erkennbar. Ein Grund dafür ist, dass der Trainingsalgorithmus zur Bestimmung der Frequenztabellen Alg. (3.4) zwar mit Aktualisierung der A-priori-



(a) (b)

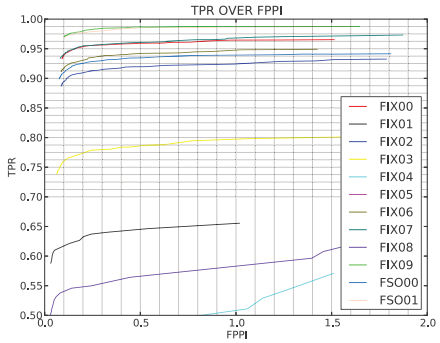
Abbildung 4.8.9: ROC-Kurven mit Detektionsraten (a) und gemittelten Laufzeiten (b) pro Bild. Die entsprechenden Konfigurationen sind aus Tab. 4.8(a) zu entnehmen.

Wahrscheinlichkeiten, aber ohne eine Normierung auf ein Signifikanzniveau, gewählt wurde, wodurch sich die unterschiedlichen Klassifikationsleistungen der Stufenklassifikatoren stark auf die Gesamtleistung auswirken. Deshalb werden im nächsten Versuch die Auswirkungen bei Normierung bzw. Skalierung der Stufenklassifikatoren auf ein bestimmtes Signifikanzniveau analysiert.

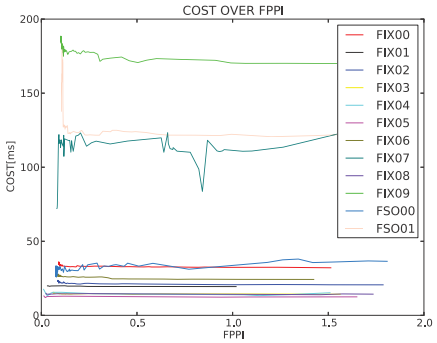
4.8.7 Signifikanzniveaus und Aktualisierung der A-priori-Wahrscheinlichkeiten

In diesem Versuch sollen die Auswirkungen der Aktualisierung der A-priori-Wahrscheinlichkeiten der Stufenklassifikationsfunktionen im Gegensatz zur Verwendung von konstanten A-priori-Wahrscheinlichkeiten in Alg. (4.4), Schritt 2(c)i, getestet werden. Für konstante A-priori-Wahrscheinlichkeiten werden anstatt der Aktualisierung konstante Werte verwendet. Außerdem werden in diesem Versuch unterschiedliche Signifikanzniveaus der Stufenklassifikatoren getestet, indem jeder Klassifikatorkandidat, der in Alg. (4.1) zum Auswahlprozess zur Verfügung steht, mit Alg. (3.6) auf ein vordefiniertes Signifikanzniveau skaliert wird. Wird ein Stufenklassifikator z. B. auf ein Signifikanzniveau von α eingestellt, so ist die A-priori-Wahrscheinlichkeit der Objektklasse nach der Klassifikation $P(\omega_1) = 1 - \alpha$. Außerdem werden zur Ge-

nierung der Muster für die Stufenklassifikatoren in Alg. (4.4), Schritt 2a als Objektklasse entweder alle Positiv-Beispiele oder nur die korrekt klassifizierten Positiven verwendet. Die allgemeinen Parameter des Trainings sind identisch mit Tabelle 4.7.1. Es wurden dieselben nicht optimierten Merkmale, wie in Abschnitt 4.7 und 4.8.5 beschrieben, verwendet. Die Auflösungsstufen sind $\{\Delta r_{\frac{1}{3}}, \ldots, \Delta r_{\frac{1}{12}}\}$. In Abb. 4.8.10 sind die ROC-Kurven der Detektionsraten



(a)



(b)

Parameter						
ID	Stufen	Sign. α	$P(\omega_1)$	A-priori update	$\mathcal{D}_1$	$\mathcal{D}_0$
FIX00	157	0.05	0.95	✓	TP,FN	FP
FIX01	57	0.01	0.99	-	TP,FN	FP
FIX02	121	0.1	0.90	✓	TP,FN	FP
FIX03	97	0.2	0.80	✓	TP,FN	FP
FIX04	23	0.05	0.95	-	TP,FN	FP
FIX05	15	0.1	0.90	-	TP,FN	FP
FIX06	117	-	-	✓	TP,FN	FP
FIX07	105	0.001	0.999	-	TP,FN	FP
FIX08	34	0.025	0.975	-	TP,FN	FP
FIX09	172	0.001	0.999	✓	TP,FN	FP
FSO00	144	0.05	0.95	✓	TP	FP
FSO01	163	0.001	0.999	✓	TP	FP

(c)

Eval					Parameter
TPR@1	Cost@1	TPR_{min}	FPPI_{min}	Cost	ID
0.988	170 ms	0.971	0.102	186 ms	FIX09
0.986	122 ms	0.968	0.104	156 ms	FSO01
0.968	65 ms	0.934	0.081	72 ms	FIX07
0.964	32 ms	0.934	0.091	34 ms	FIX00
0.962	343 ms	0.958	0.822	351 ms	VJH_0
0.954	1115 ms	0.896	0.101	1121 ms	VJH_1
0.948	24 ms	0.912	0.086	27 ms	FIX06
0.939	30 ms	0.899	0.076	31 ms	FSO00
0.924	21 ms	0.887	0.086	23 ms	FIX02
0.798	14 ms	0.738	0.063	15 ms	FIX03
0.655	19 ms	0.588	0.031	20 ms	FIX01
0.596	14 ms	0.494	0.024	15 ms	FIX08
0.511	14 ms	0.399	0.012	17 ms	FIX04
0.431	12 ms	0.273	0.013	13 ms	FIX05

(d)

Abbildung 4.8.10: Auswertung der Klassifikatoren wobei die A-priori-Wahrscheinlichkeiten entweder konstant gelassen oder aktualisiert wurden. (a): ROC-Kurven über den Falsch-Positiven pro Bild (b): Mittlere Laufzeiten der Detektionen über den Falsch-Positiven pro Bild (c): Verwendete Parameter (d): Nach Detektionsraten absteigend sortierte Klassifikatoren bei einer Falsch-Positiv-Detektion pro Bild

(Abb. 4.8.10(a)) und mittleren Laufzeiten (Abb. 4.8.10(b)) abgebildet. Die Parameter sind in Tabelle 4.8.10(c) und die nach Detektionsraten absteigend

sortierten Resultate bei einem Falsch-Positiven in Tabelle 4.8.10(d) gelistet. Wird das Verfahren zur Aktualisierung der A-priori-Wahrscheinlichkeiten in Kombination mit niedrigen Signifikanzniveaus angewendet, resultieren sehr hohe Detektionsraten (siehe FIX09 und FSO01). Allerdings entsteht aus der Konfiguration ohne A-priori-Update des Klassifikators FIX07 ebenfalls eine guter Kompromiss aus Detektionsleistung und Laufzeit. Für höhere Signifikanzniveaus ist die Detektionsleistung ohne A-priori-Aktualisierung bei einer höheren Anzahl an Stufen jedoch zu gering. Die Konfiguration FIX04 hat z.B. eine Detektionsrate von lediglich 50 % bei einer Falsch-Positiv-Detektion. Bei identischem Signifikanzniveau mit A-priori-Aktualisierung (Konfiguration FIX00) resultiert ein sehr effizienter Klassifikator mit durchschnittlichen 32 ms Detektionszeit pro Bild und einer Detektionsleistung von 96.4 %. Die Verwendung von Bayes-Entscheidungsfunktionen mit Aktualisierung der A-priori-Wahrscheinlichkeiten erzielt somit ebenfalls ressourcenoptimierte Klassifikatoren.

Vergleicht man diese Konfiguration mit FSO00, bei der nur die korrekt klassifizierten Beispiele (ohne Falsch-Negative) als Objektmuster für die Stufenklassifikatoren verwendet wurden, fällt auf, dass es günstiger ist, die Falsch-Negativ-Beispiele mitzuführen, da man eine bessere Detektionsrate bei identischer Laufzeit erhält. Obwohl die Falsch-Negativen auf dem Trainingsdatensatz in den entsprechenden Stufen nicht vorkommen, bringen sie einen Vorteil bei “ungesehenen” Daten im Testset.

4.8.8 Vergleich zu AdaBoost mit mehreren Auflösungsstufen

Bei diesem abschließenden Versuch wird auf die Auflösungseigenschaften und Trainingsparameter der AdaBoost-Kaskade mit den Haar-Merkmalen eingegangen, und die Klassifikationsleistungen werden mit der Bayes-Kaskade (Id FIX00), die im vorangehenden Abschnitt 4.8.7 beschrieben wurde, verglichen.

Beim Training der AdaBoost-Kaskaden mit unterschiedlichen Anfangsauflösungen konnten im Vergleich zu den Bayes'schen Kaskaden erst ab einer Auflösung von $\Delta r_{\frac{1}{6}}$ Kaskaden erzeugt werden, die von der Klassifikationsleistung als Detektoren verwendbar sind. Zusätzlich zu den Grundeinstellungen, die in Tabelle 4.7.4 und 4.7.5 gelistet sind, mussten beim Training des Schwellwertes die Stufenklassifikatoren auf ein Signifikanzniveau von $\alpha = 0.01$ gebracht werden, um funktionsfähige Kaskaden mit einer höhe-

ren Anzahl an Stufenklassifikatoren zu erzeugen, die entsprechend niedrige Falsch-Positiv-Detektionen generieren. Tabelle 4.8.7 zeigt die unterschiedli-

Parameter						
					Entscheidung	
ID	Auflösungen $\mathcal{F}_{\Delta r}$	Stufen	A-priori	α	Typ	Einträge
FIX00	$1/3, \ldots, 1/12$	157	✓	0.05	$P_s(\omega_i)$	256
AV01	$1/12$	13	-	0.01	$\Theta(x_s - \theta_s)$	1024
AV02	$1/6, 1/12$	90	-	0.01	$\Theta(x_s - \theta_s)$	1024
AV03	$1/6, \ldots, 1/12$	43	-	0.01	$\Theta(x_s - \theta_s)$	1024

Tabelle 4.8.7: AdaBoost- und Bayes-Kaskaden mit unterschiedlichen Auflösungen

chen Konfigurationen der Klassifikatoren. Abb. 4.8.11 zeigt die unterschiedlichen ROC-Kurven der Klassifikatoren. Betrachtet man die Detektionsrate über den Falsch-Positiven pro Bild auf der X-Achse (Abb. 4.8.11(a)), so fällt auf, dass der Klassifikator AV03 schon vorher in die Sättigung geraten ist und eine höhere Falsch-Positiv-Rate besitzt. Deshalb ist in Abb. 4.8.11(b) die Detektionsrate über der Falsch-Positiv-Rate dargestellt. Dieser Vergleich dient außerdem zur Veranschaulichung des Unterschiedes zwischen den Einheiten Falsch-Positive pro Bild (Abb. 4.8.11(a)) und Falsch-Positiv-Rate, welche die Einheit Falsch-Positive pro Detektionsfenster besitzt (Abb. 4.8.11(b)). Die Zahlenwerte unterscheiden sich um mehrere Größenordnungen. Durch die Verwendung der auflösungsspezifischen Depth-First-Detektion wird die Anzahl der zu überprüfenden Rechtecke im Voraus stark reduziert. Nur bei einer wahrscheinlichen Zugehörigkeit zur Objektklasse werden weitere verschobene Detektionsfenster in den Nachbarschaftsregionen überprüft, die dann als "komplexere" Falsch-Positive in die Berechnung der Falsch-Positiv-Rate miteinfließen. Daraus resultiert eine im Vergleich höhere Falsch-Positiv-Rate, und die ROC-Kurve des Klassifikators FIX00 liegt unterhalb der anderen Kurven in Abb. 4.8.11(b). Deshalb ist die Darstellung über die Falsch-Positiv-Detektionen pro Bild mit dem auflösungsabhängigen Detektionsalgorithmus geeigneter, darüber hinaus lässt sich das Verhalten des Detektors im Anwendungsfall besser veranschaulichen. In Abb. 4.8.11(c) sind die Auflösungen der Stufenklassifikatoren dargestellt und Abb. 4.8.11(d) zeigt die mittleren Laufzeiten pro Bild, wobei sukzessive der letzte Stufenklassifikator entfernt wurde. Im Vergleich zur Bayes-Kaskade FIX00 wird in der Auflösungsstufe $\Delta r_{\frac{1}{12}}$ eine

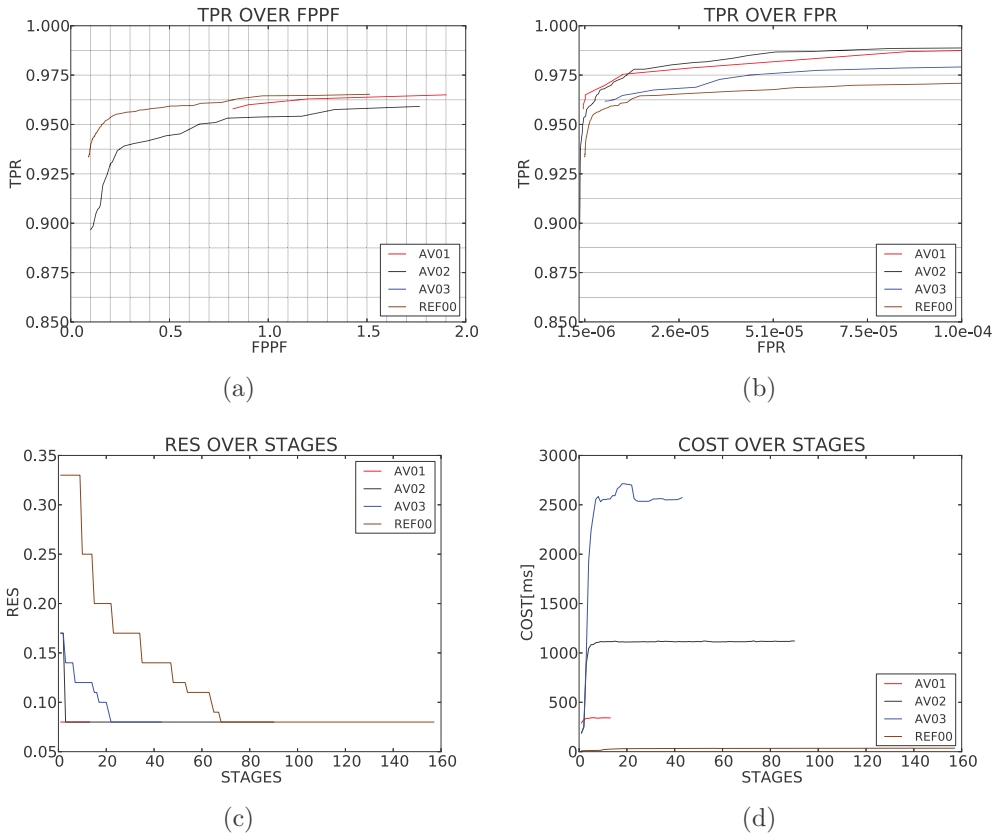


Abbildung 4.8.11: ROC-Kurven der Klassifikatoren aus Tabelle 4.8.7 (a): Detektionsrate über den Falsch-Positiven pro Bild (b): Detektionsrate über der Falsch-Positiv-Rate (c): Auflösungen der Stufenklassifikatoren (d): Gemittelte Laufzeiten pro Bild für die entsprechenden Stufen

initiale Menge an Detektionsfenstern von $|\mathcal{R}| = 738384$ (Gl. (4.4.2)) generiert. Dadurch wird der Berechnungsaufwand des Klassifikators AV01 auch bei einer geringen Anzahl von schwachen Klassifikatoren (es sind zwei Weak-Learner in der ersten Stufe) beachtlich, wodurch die mittlere Laufzeit von 300 ms pro Bild entsteht. Die Bayes-Kaskade FIX00 benötigt bei einer Anfangsauflösung von $\Delta r_{\frac{1}{3}}$ lediglich eine Anzahl von $|\mathcal{R}| = 50719$ Detektionsfenster und erzielt eine mittlere Laufzeit von 34 ms, was einer Laufzeitverbesserung um einem Faktor von nahezu zehn entspricht.

Bei Betrachtung der AdaBoost-Kaskaden mit Haar-Merkmalen in gerin-

geren Anfangsauflösungen fällt auf, dass es keine Verbesserung der Detektionsleistung gibt. Zusätzlich steigen die mittleren Berechnungskosten in starkem Maße an. Bei der Kaskade AV02, die mit der Auflösung $\Delta r_{\frac{1}{6}}$ startet, ist die Klassifikationsleistung geringfügig schlechter und die Berechnungskosten steigen auf ca. eine Sekunde pro Bild. Bei der Kaskade AV03, die mit den Auflösungen $\Delta r_{\frac{1}{6}}, \Delta r_{\frac{1}{7}}, \ldots, \Delta r_{\frac{1}{12}}$ konfiguriert wurde, tritt eine Sättigung des Klassifikators bei 4 FPPF auf und die mittleren Kosten steigen auf ca. 2,5 Sekunden pro Bild. Dieses Verhalten kann damit erklärt werden, dass die Neugewichtung der Beispiele beim AdaBoost-Verfahren anhand des im aktuellen Boostingschrittes ermittelten schwachen Klassifikators vorgenommen werden. Die genaue Beschreibung des Verfahrens und die unterschiedlichen Gewichtungsverfahren der Muster bei AdaBoost und dem Gradientenboosting wird in Kapitel 6 erklärt. Durch die niedrige Auflösung der Haar-Merkmale hat diese Neugewichtung einen zu schwachen Effekt um einen ausreichenden Performancegewinn zu erzielen, sodass das Boostingverfahren relativ schnell in eine Sättigung gerät und abbricht, wodurch eine geringere Anzahl an schwachen Klassifikatoren in den Stufenklassifikatoren enthalten ist. In Abb. 4.8.12 ist die Anzahl der schwachen Klassifikatoren (Abb. 4.8.12(a) und 4.8.12(c)) in den jeweiligen Stufenklassifikatoren der Kaskaden, sowie die entsprechenden Auflösungen (Abb. 4.8.12(b) und 4.8.12(d)) der Kaskaden AV02 und AV03 gelistet. Es sind deutlich weniger Klassifikatoren enthalten als in der Kaskade AV01, die mit Auflösung $\Delta r_{\frac{1}{12}}$ beginnt (siehe Abb. 4.7.5).

Für die Auflösungsstufen bedeutet dies, dass bei einer relativ hohen Auftrittswahrscheinlichkeit von Negativbeispielen in eine neue Auflösungsstufe gewechselt wird. So wird z.B. bei beiden Kaskaden AV02 und AV03 nach zwei Stufenklassifikatoren in die nächst höhere Auflösungsstufe gewechselt (siehe Abb. 4.8.12(b) und 4.8.12(d)), da die Auftrittswahrscheinlichkeit der Negativbeispiele nicht weiter reduziert werden konnte und durch die Vervielfachung der Negativbeispiele in den Nachbarschaftstests der Depth-First-Detektion (Alg. (4.6), Schritt 3b) die Berechnungskosten stark ansteigen. Dieser Effekt verstärkt sich mit weiteren Auflösungsstufen bei Kaskade AV03. Deshalb sind die mittleren Laufzeiten bei der feineren Aufteilung der Auflösungsstufen in Kaskade AV03 nochmals deutlich höher als bei Kaskade AV02 mit insgesamt zwei Auflösungsstufen.

Die Klassifikationsleistungen der Stufenklassifikatoren der Bayes-Kaskade REF00 können durch die Extraktion von mehrdimensionalen Merkmalsvek-

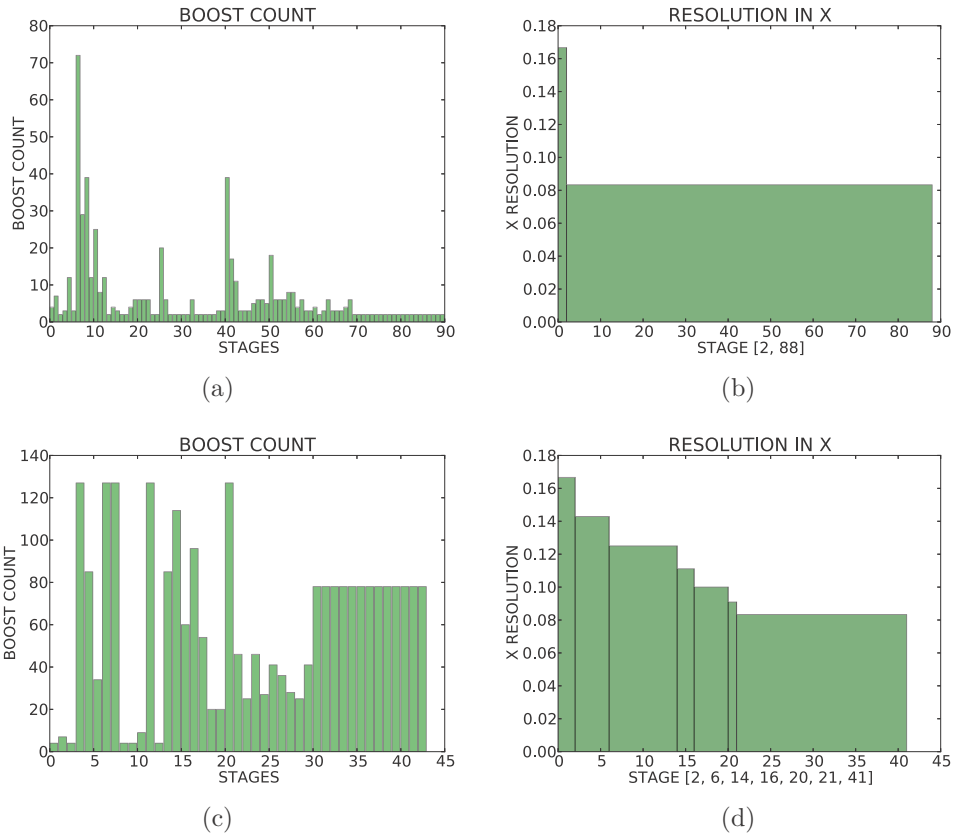


Abbildung 4.8.12: Anzahl schwacher Klassifikatoren in den entsprechenden Stufen der Kaskaden AV02 (siehe (a)) und AV03 (siehe (c)), sowie die unterschiedlichen Auflösungen für AV02 (siehe (b)) und AV03 (siehe (d)) mit den Konfigurationen aus Tabelle 4.8.7

toren und der sequentiellen Vorwärtsauswahl im Gegensatz auf ein hohes Maß gesteigert werden, dass die Auftrittswahrscheinlichkeiten der Negativbeispiele so gering sind, dass die Nachbarschaftstest nicht ins Gewicht fallen und effiziente mittlere Laufzeiten von 34 ms entstehen.

4.9 Zusammenfassung

In diesem Kapitel wurde ein neues Verfahren zur adaptiven, kombinatorischen Ressourcenoptimierung vorgestellt. Mit dem Verfahren lässt sich der Kaskadenklassifikator im Bezug auf den erwarteten Energieverbrauch optimieren, indem aus einer Menge an Kandiatenklassifikatoren mit unterschiedlichen Klassifikations- sowie Laufzeiteigenschaften derjenige Stufenklassifikator gewählt wird, der die erwarteten Fehloperationen minimiert. Dazu wird jedem Klassifikator ein Kostenwert zugeordnet, der den Elementaroperationen für eine Klassifikation entspricht. Zusätzlich wurde eine Gewichtung eingeführt, sodass in höheren Stufen der Klassifikationsfehler stärker gewichtet wird, da die Auftrittswahrscheinlichkeit der Objektklasse und die Ähnlichkeit zur Hintergrundklasse der Muster steigen. Da die Klassifikationsleistung der Kandidatenklassifikatoren in höheren Stufen nicht ausreichend ist um ausreichend niedrige Falsch-Positiv Detektionen zu erreichen, wurde das sequentielle Merkmalsselektionsverfahren (SFS) [118] mit der kombinatorischen Ressourcenoptimierung verknüpft. Mit den im vorigen Kapitel eingeführten gitterbasierten Merkmalen wurden auflösungsspezifische Trainings- und Detektionsverfahren mit einer von der Auflösung abhängigen lokalen Nachbarschaftssuche eingeführt. Der Unterschied dieser Verfahren, die im Rahmen der vorliegenden Arbeit entwickelt wurden, zu den in der Literatur gängigen Pyramidenverfahren ist, dass die Auflösungseigenschaft direkt durch die geometrische Form der Merkmale erreicht wird. Somit muss zum einen keine Bildpyramide erstellt werden, und zum anderen kann die Aufteilung der unterschiedlichen Auflösungen in feineren Schritten als mit Halbierungsschritten erfolgen. Der US-Verkehrszeichendatensatz, der bei allen Versuchen in dieser Arbeit verwendet wird, wurde ausführlich erklärt, sowie das Auswertungsverfahren, mit dem die ROC-Kurven erstellt wurden. Um das Verfahren zu veranschaulichen wurde eine genaue Analyse und Visualisierung aller wesentlichen Komponenten einiger wichtiger Stufenklassifikatoren durchgeführt.

Bei der experimentellen Auswertung wurden die unterschiedlichen Merkmale und Trainings- sowie Detektionsalgorithmen, die in Kapitel 3 und in diesem Kapitel beschrieben sind, mit ROC-Kurven und zusätzlichen Kurven mit den mittleren Laufzeiten, ausgewertet und verglichen. Bei dem ersten Versuch 4.8.2 wurden die unterschiedlichen Merkmale in einer mittleren Auflösung getestet, wobei das erweiterte Gradientenboosting aus Kapitel 6 verwendet wurde um die Falsch-Positiv-Detektionen zu minimieren. Da die Haar-

ähnlichen Merkmale erst ab einer höheren Merkmalsauflösung funktionsfähige Trenneigenschaften besitzen, wurde zusätzlich eine AdaBoost-Kaskade und eine mit dem im Rahmen dieser Arbeit erweiterten Gradientenboosting trainierten Kaskade mit den Haar-ähnlichen Merkmalen trainiert. Die Kaskade mit dem erweiterten Gradientenboosting erzielte eine Detektionsrate von 98.6%@1FPPF im Gegensatz zur AdaBoost-Kaskade mit 96.2%@1FPPF mit identischen Merkmalen. Allerdings entstanden relativ hohen Berechnungskosten. Die modifizierten HOG-Merkmale erzielten, wie erwartet, die längsten mittleren Laufzeiten von über 2500 ms pro Bild. Mit den in dieser Arbeit eingeführten kombinierten Intensitäts- und Tensormerkmalen wurden die besten mittleren Laufzeiten von 76 ms pro Bild bei einer Detektionsrate von 96.7%@1FPPF erreicht. Da mit den Intensitäts- und Tensormerkmalen das beste Verhältnis aus Detektionsleistung und mittlerer Laufzeit erzielt wurde, wurden diese Merkmale als Resultat aus diesem Versuch in den weiteren Experimenten verwendet.

Beim nächsten Versuch 4.8.3 wurden unterschiedliche Funktionen zur Ressourcenoptimierung mit den kombinierten Intensitäts- und Tensormerkmalen in einer mittleren Auflösungsstufe untersucht. Mit der Kostenfunktion, die nur die Fehlklassifikationen berücksichtigt, wurde eine Detektionsrate von 98%@0.68FPPF bei 5503 ms pro Bild erreicht. Bie der Kostenfunktion mit zusätzlicher Berücksichtigung der Berechnungskosten und somit den erwarteten Fehloperationen wurde eine geringfügig niedrigere Detektionsrate von 96%@0.45FPPF erzielt, allerdings bei einer mittleren Laufzeit von 279 ms pro Bild und damit einem Unterschied in den Laufzeiten von Faktor 20.

Da mit den mittleren und hohen statischen Auflösungen noch keine echtzeitfähigen Klassifikatoren generiert werden konnten, wurden im darauffolgenden Versuch 4.8.4 die Algorithmen mit unterschiedlichen Unterteilungen und steigenden Auflösungen in den Merkmalen getestet. Dazu wurden im Training Unterkaskaden mit Merkmalen aus einer bestimmten Auflösungung traininert, indem die Basisrechtecke in ihrer Größe beschränkt werden. Beim Detektionsalgorithmus wird ein in dieser Arbeit entwickelter rekursiver Depth-First-Algorithmus zur auflösungsabhängigen Feinsuche mit den Unterkaskaden verwendet. Die Stufenklassifikatoren wurden nicht mit einem Boosting-Verfahren wie AdaBoost oder Gradientenboosting trainiert, sondern mit der sequentiellen Merkmalsselektion (SFS). Zum einen hat sich bei diesem Versuch gezeigt, dass bei feineren Anfangsauflösungen bessere Detektionsergebnisse erzielen lassen. So wird zum Beispiel mit zwei unterschiedlichen Auflösungen eine Kaskade mit 98.2%@1FPPF und 74 ms mitt-

lere Laufzeit pro Bild erreicht, wodurch die Energiekosten bei gleicher Detektionsleistung wie bei den anderen Versuchen um den Faktor $3\frac{3}{4}$ reduziert werden konnten. Zum anderen wurde ersichtlich, dass eine feine Unterteilung mit vielen Auflösungsstufen sehr effiziente Detektoren erzeugt. So erzielen die Klassifikatoren mit den im Rahmen dieser Arbeit entwickelten Bayes-Entscheidungsfunktionen für den Kaskadenklassifikator Detektionsraten von 96.1%@1FPPF bei 40 ms (Id REO05) und 94.2%@1FPPF bei 22 ms mittlere Laufzeit pro Bild (Id REF05). Im Vergleich zu der AdaBoost-Kaskade mit Haar-ähnlichen Merkmalen wird so ein Laufzeitgewinn von Faktor neun erzielt. Diese Klassifikatoren befinden sich bereits im Echtzeitbereich. Es hat sich ausserdem gezeigt, dass mit den Haar-ähnlichen Merkmalen erst ab einer höheren Auflösung funktionsfähige Kaskaden erzeugt werden können, die dadurch allerdings höhere Energiekosten verursachen.

Beim nächsten Versuch 4.8.5 wurden zum einen die Auswirkungen der Anzahl der Einträge für die Bayes'schen Entscheidungsfunktionen und zum anderen die Depth-First-Detektionsalgorithmen getestet. Als Ergebnis für die Entscheidungsfunktionen wurde ersichtlich, dass bei mehr Einträgen die mittlere Laufzeit sinkt, aber auch die Klassifikationsraten schlechter werden. Ein guter Kompromiss wurde mit 256 Einträgen erreicht. Bei den im Rahmen dieser Arbeit entwickelten Detektionsalgorithmen hat sich gezeigt, dass mit der auflösungsabhängigen, adaptiven Nachbarsuche mit dem Depth-First-Algorithmus um bis zu 10 % bessere Detektionsraten erzielt werden können.

Im Versuch 4.8.6 wurde die Auswirkung einer softwaretechnischen Optimierung der Merkmale untersucht. Es konnten damit zwar sehr schnelle Klassifikatoren erzeugt werden, deren mittlere Laufzeiten unter 20 ms liegen. Allerdings wäre es zu aufwändig gewesen alle Merkmalstypen und -kombinationen zu implementieren, wodurch die Detektionsrate mit 92.8% schlechter wurde. Bei diesem Versuch wurde auch ersichtlich, dass manche Stufenklassifikatoren einen schlechten Einfluss auf die Gesamtdetektionsleistung haben, falls die Einzelklassifikationsraten zu gering sind.

Deshalb wurde im Versuch 4.8.7 der Algorithmus zur Skalierung der Stufenklassifikatoren auf ein definiertes Signifikanzniveau, der im Rahmen dieser Arbeit entwickelt wurde, getestet. Das Signifikanzniveau von 5 % in Verbindung mit dem Algorithmus zur Aktualisierung der A-priori-Wahrscheinlichkeiten erzeugt Klassifikatoren mit hoher Detektionsrate und effizienten mittleren Laufzeiten, von z.B. 96.4%@1FPPF bei 32 ms pro Bild. Außerdem wurde bei diesem Versuch gezeigt, dass es für die Detektionsrate günstiger ist, alle Objektbeispiele zum Training der Stufenklassifikatoren zu

verwenden, inklusive der Falsch-Negativen.

Beim letzten Versuch 4.8.8 wurden die AdaBoost-Klassifikatoren mit Haar-ähnlichen Merkmalen mit unterschiedlichen Auflösungen analysiert und mit der Kaskade mit Id FIX00 aus dem letzten Versuch verglichen. Mit letzterer ergaben sich im Vergleich zu den AdaBoost Klassifikatoren sowohl bessere Detektionsraten als auch ein Laufzeitgewinn um einen Faktor zehn.

Kapitel 5

Konfidenzberechnung zur Klassifikatorfusion und semi-überwachtem Lernen

Bei den Standardverfahren zur Kaskadenklassifikation mit geboosteten Stufenklassifikatoren werden schwellwertbasierte Klassifikationsfunktionen verwendet, die auf die Objekt- und Hintergrundklasse abbilden. Es liegt somit kein kontinuierliches Sicherheitsmaß vor, welches die unterschiedlichen Entscheidungen der Stufenklassifikatoren, sowie des Kaskadenklassifikators berücksichtigt.

In diesem Kapitel werden Verfahren zur Konfidenzwertberechnung vorgestellt, die im Rahmen dieser Arbeit entwickelt wurden und zum Teil in dem Artikel [108]* präsentiert wurden. Diese Verfahren erlauben eine genaue Differenzierung beliebiger Beispiele derselben Klassenzugehörigkeit im Sinne einer Sicherheit der Entscheidung der Stufenklassifikatoren. Außerdem werden Berechnungsverfahren zur Fusion der Konfidenzwerte zwischen den Stufenklassifikatoren vorgestellt, woraus Konfidenzmaße für den Kaskadenklassifikator resultieren. In der Literatur werden moderate Kaskadentiefen von drei bis fünf Kaskadenstufen [91] oder 32 Stufen [116] verwendet, wobei das hier vorgestellte Verfahren eine Fusion der Konfidenzwerte von 200 Stufen mit einem geringen Klassifikationsfehler und effizienter Laufzeit ermöglicht [108]*.

Mit den fusionierten Konfidenzen ist es weiterhin möglich ein gewichtetes Detektionsfenster aus multiplen Detektionen zu berechnen. Außerdem wird ein konfidenzbasiertes Auswahlverfahren der Muster für die Stufenklassifikatoren vorgestellt, welches in das Kaskadentraining integriert ist. Die Entscheidungsfunktionen werden geeignet gewichtet, sodass vermehrt “interessante” Beispiele mit geringer Konfidenz und somit größerer Klassenähnlichkeit gewählt werden und dementsprechend das Training des Klassifikators, im Gegensatz zur Ausführung, auf komplexeren Teilproblemen der Stufen vollzogen wird. Dadurch kann der Klassifikationsfehler des Zielklassifikators drastisch reduziert werden, jedoch mit einer erhöhten Berechnungskomplexität. Auf der anderen Seite kann sich der Klassifikator auf “einfachere Beispiele” in den Stufen fokusieren, sodass schnellere Laufzeiten und ein größerer Klassifikationsfehler resultieren. Die Priorisierung dieser Optimierungskriterien lässt sich mit der vorgestellten Gewichtungsfunktion steuern. Letztendlich wird ein neues, im Rahmen dieser Arbeit entwickeltes, teilüberwachtes Verfahren zum Training von Kaskadenklassifikatoren vorgestellt, bei dem der Klassifikator die Möglichkeit besitzt für die Stufenprobleme Beispiele der Hintergrundklasse in die Objektklasse zu “migrieren” und als Objekte zu verwenden, falls eine hohe positive Konfidenz existiert und zugleich interessante Negativbeispiele mit niedriger Konfidenz aus der Hintergrundklasse auswählen kann.

Das semi-überwachte Lernen (SSL) [12, 26, 51, 124, 127]) ist insbesondere bei Detektoren interessant, bei denen eine überproportional große Menge an Negativbeispielen existiert, die ebenfalls Beispiele aus der Objektklasse, die durch menschliche Fehler beim Annotieren der Beispiele oder strukturellen Ähnlichkeiten zu anderen Objektklassen entstehen, enthalten kann, wie es hier der Fall ist. Aus dieser Menge können mit dem semi-überwachten Lernen potentielle Objektkandidaten ermittelt werden, die in die Objektklasse migriert werden. Das zeitgleiche Extrahieren "interessanter" Negativbeispiele aus der umfangreichen Hintergrundklasse entspricht den Methoden des aktiven Lernen [111, 82, 61, 27, 102, 78], bei denen der Klassifikator für sich "interessante" Beispiele bevorzugt bei denen eine größere Unsicherheit existiert und aktiv aus einer großen Menge auswählt, da in diesen Beispielen mehr "neues Wissen" enthalten ist und damit ein größerer Lernerfolg zu erwarten ist, als mit Beispielen auf denen sich der Klassifikator sicher ist. Das hier vorgestellte Verfahren ist ein neuer Ansatz zum integrierten, teilüberwachten Lernen mit Kaskadenklassifikatoren, da es kein Wrapper-Ansatz ist, sondern innerhalb der Kaskadenstruktur angewendet wird.

Die in der Literatur beschriebenen Methoden des teilüberwachten Lernens beeinhalten Self-training und Co-training [129], Generative model based training [33], Graph based training [14, 110], Training mittels sogenannter transductive support vector machines [11] und semi-supervised linear discriminant analysis [23]. Teilüberwachte Verfahren mit geboosteten Klassifikatoren werden in den Arbeiten [80, 66, 117, 72] beschrieben, wobei in dem Artikel [72] das Semiboost-Verfahren [80] erweitert wird, indem visuelle Ähnlichkeiten zwischen Bildausschnitten belernt werden.

Teile aus den im Folgenden beschriebenen Verfahren wurden in der Veröffentlichung [108]* vorgestellt.

5.1 Konfidenzwertberechnung und Fusion innerhalb kaskadierter Systeme

Für die Berechnung des Konfidenzwertes benötigt man ein geeignetes Modell. Deshalb werden abhängig von der Entscheidungsfunktion des Stufenklassifikators drei Modelle für die fusionierte Konfidenzwertberechnung der

Stufenklassifikatoren vorgestellt [108]*:

$$\gamma_1(h_c^T, \underline{z}) = \frac{1}{T} \sum_{s=1\,,\, h_s(\underline{z})=\omega_1}^{T} |p_s(x(\underline{z})|\omega_1)P_s(\omega_1) - p_s(x(\underline{z})|\omega_0)P_s(\omega_0)| \quad (5.1.1)$$

$$\gamma_2(h_c^T, \underline{z}) = \frac{1}{T} \sum_{s=1, h_s(\underline{z})=\omega_1}^{T} \frac{|p_s(x(\underline{z})|\omega_1)P_s(\omega_1) - p_s(x(\underline{z})|\omega_0)P_s(\omega_0)|}{p_s(x(\underline{z})|\omega_1)P_s(\omega_1) + p_s(x(\underline{z})|\omega_0)P_s(\omega_0)} \quad (5.1.2)$$

$$\gamma_3(h_c^T, \underline{z}) = \frac{1}{T} \sum_{s=1, h_s(\underline{z})=\omega_1}^{T} |x_s(\underline{z}) - \theta_s| \quad (5.1.3)$$

wobei $x(\underline{z}) = \underline{w}_s^T \underline{z}$ die Projektionen der Stufenklassifikatoren sind. Laut Schürmann [101] ist die "bestmögliche Konfidenz" die Bayes'sche. Deshalb liegt es nahe, die Konfidenzen in den Gleichungen (5.1.1),(5.1.2) als Differenzen der mit Algorithmus (3.4) und (3.6) geschätzten Wahrscheinlichkeitsdichten zu berechnen, die zur Evaluierung der Entscheidung berechnet werden. Im Detektionsalgorithmus (4.6) müssen diese dann bei der Auswertung der Stufenklassifikatoren nur noch summiert und gemittelt werden. Da diese Berechnung als Look-Up-Operation implementiert werden kann, wird kein zusätzlicher Berechnungsaufwand benötigt. Für Einträge mit hohen Frequenzen einer Klasse besitzt der Klassifikator eine hohe Konfidenz, da eine große Menge an Evidenzen gesammelt wurden. Falls die Frequenzen beider Klassen hoch sind, gibt es einen Konflikt bei der Entscheidung und die Konfidenz ist niedrig.

Zusätzlich werden ebenfalls die A-priori-Wahrscheinlichkeiten berücksichtigt, wobei eine stärkere Gewichtung der Objektklasse in hohen Stufen durchgeführt wird, da die Auftrittswahrscheinlichkeit dieser Klasse mit jeder Stufe im Allgemeinen steigt. Durch die Mittelung über alle Stufen ist die Konfidenzberechnung unabhängig von der Gesamtzahl der Stufen. Gl. (5.1.2) unterscheidet sich von Gl. (5.1.1) durch eine zusätzliche Normierung wodurch geringe Frequenzen hochgewichtet werden, die Entscheidungsgrenzen sind identisch, was in Abb. 5.1.1 anhand einer Verteilung mit A-priori-Wahrscheinlichkeiten von $P(\omega_1) = 1$ und $P(\omega_0) = 0.6$ dargestellt ist.

Für den Fall der standardmäßig verwendeten schwellwertbasierten Entscheidungsfunktionen wird Gl. (5.1.3) verwendet, wobei der gemittelte Abstand zu den Entscheidungsgrenzen berechnet wird. Abb. 5.1.2 zeigt die Konfidenzwerte der einzelnen Stufenklassifikatoren, indem die Differenz der Frequenzen der Klassen gebildet wird, in 5.2(a) und 5.2(b) dargestellt durch

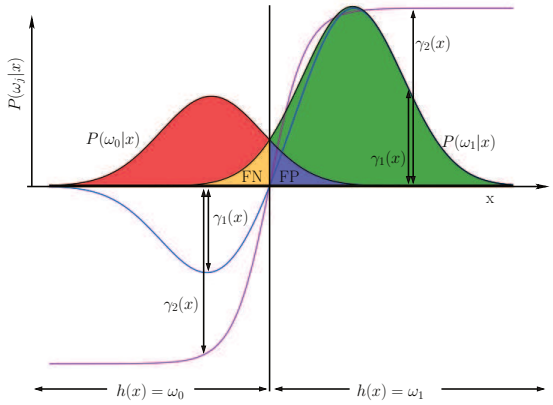


Abbildung 5.1.1: Unterschied der Konfidenzwerte γ_1 (Gl. (5.1.1)) als blaue Linie eingezeichnet und γ_2 (Gl.(5.1.2)) als violette Linie eingezeichnet. Die Entscheidungsgrenzen sind identisch, die A-priori-Wahrscheinlichkeiten sind $P(\omega_1) = 1$ und $P(\omega_0) = 0.6$.

blaue Balken für Positiv-Entscheidungen und schwarze Balken für Negativ-Entscheidungen. Es ist zu erkennen, dass die Konfidenzwerte für die Stufe $s = 79$ in Abb. 5.2(b) durch die A-priori-Wahrscheinlichkeiten für Negativ-Entscheidungen nahe bei Null liegen, da die Auftrittswahrscheinlichkeit der Hintergrundklasse in den hohen Stufen gering ist. Im Gegensatz zu den schwellwertbasierten Entscheidungsfunktionen 5.2(c) und 5.2(d) existieren bereits in Stufe $s = 1$ zwei nicht zusammenhängende negative Entscheidungsregionen, davon eine kleine Region zwischen 0.8 und 0.9. In Abb. 5.2(c) und 5.2(d) wird eine höhere Anzahl von 1024 Einträgen verwendet, da der Schwellwert integrativ (Alg. (3.8)) berechnet wird und somit genauer bestimmt werden kann. In Abb. 5.2(d) ist der Schwellwert ebenfalls durch die A-priori-Wahrscheinlichkeiten in Stufe $s = 65$ nach links verschoben.

Bei den Frequenztabellen kann mit einer höheren Anzahl an Einträgen eine schnellere Laufzeit bei geringerer Generalisierungsfähigkeit und mit weniger Einträgen ein Kaskadenklassifikator mit besserer Generalisierungfähigkeit und damit geringerem Klassifikationsfehler bei höheren Laufzeiten erreicht werden. Der Erwartungswertoperator aus Gl. (2.3.14) wird durch die Mittelung der Evidenzen pro Eintrag modelliert, wobei die Gesamtzahl der Einträge Einfluss auf den Grad der Mittelung hat. Je weniger Beispiele zur

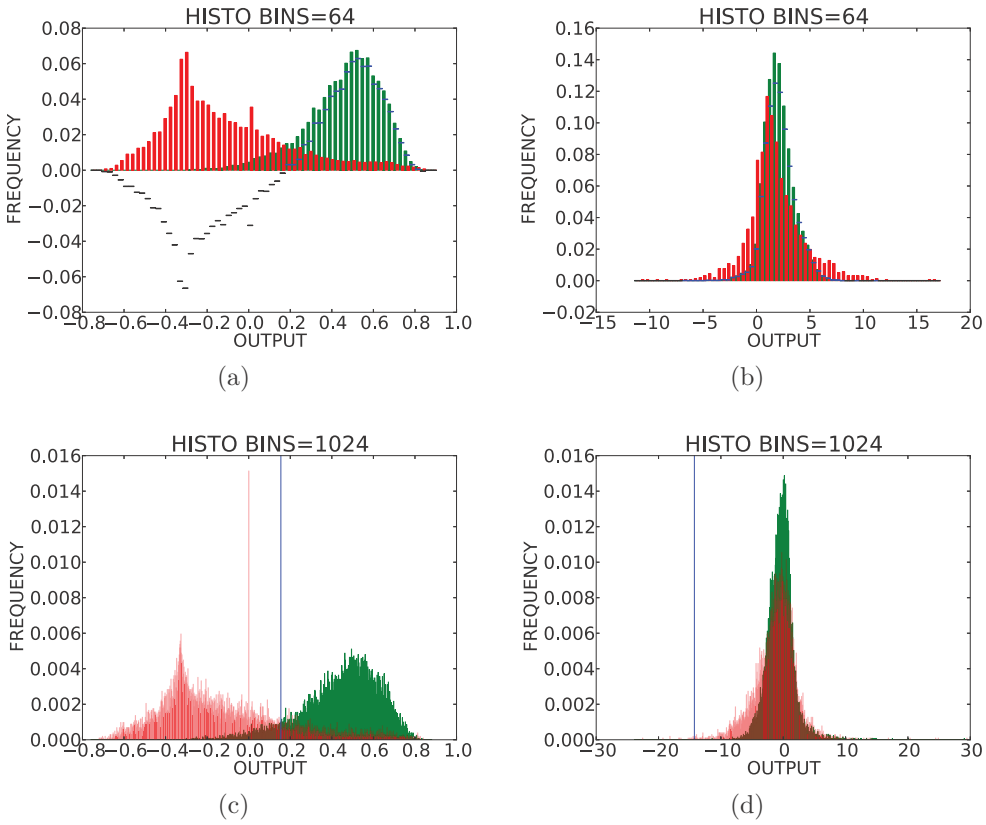


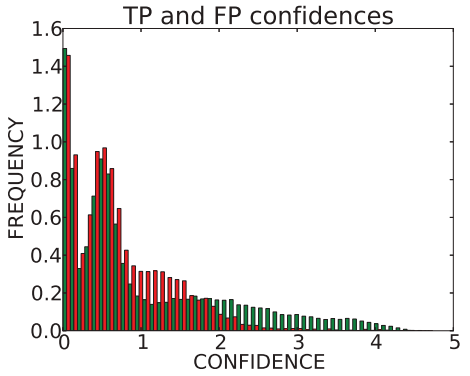
Abbildung 5.1.2: Erste Reihe: Konfidenzwerte entsprechen den Differenzen der Frequenzen. Hier durch blaue Balken für positive Entscheidungen (Objekte) und schwarzen Balken für negative Entscheidungen dargestellt.
Zweite Reihe: Schwellwertbasierte Entscheidungsfunktionen. Links der blauen Linie (Schwellwert) entspricht einer Negativentscheidung und rechts der blauen Linie einer Positiventscheidung. Die Konfidenz ist der Abstand des entsprechenden Eintrages zur blauen Linie. (a), (c): Stufe $s = 1$, (b): $s = 79$. (d): $s = 65$.

Akkumulation der entsprechenden Einträge beteiligt sind, desto fehlerhafter werden die auf den Einträgen basierenden Entscheidungen mangels repräsentativer Statistik.

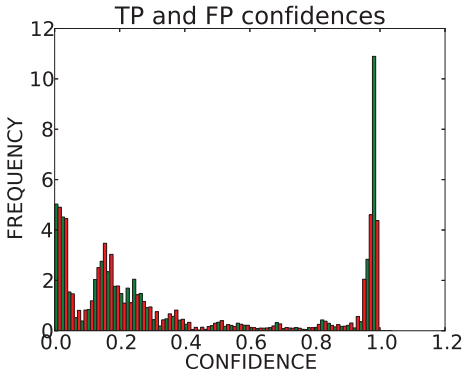
5.2 Experimentelle Auswertung der Konfidenzberechnung

Für die Auswertung der Konfidenzberechnung werden zwei unterschiedliche Versuche und eine visuelle Überprüfung durchgeführt. Wie in den anderen Versuchen wurde die Evaluierung auf dem in Kapitel 4, Abschnitt 4.5 beschriebenen Datensatz durchgeführt, wobei ein aktueller Rechner mit einem 3.3 GHz Prozessor (Intel i7 - Architektur) verwendet wurde, auf dem ein spezifischer Klassifikator mit einem Prozessor trainiert und mit einem Prozessor evaluiert wurde. Die Klassifikatoren werden mit dem Trainingsdatensatz generiert und auf dem separaten Testdatensatz, der keine identischen Realweltschilder enthält, ausgewertet. Die Einheit der dargestellten Falsch-Positiven bei den ROC-Kurven sind Falsch-Positive pro Bild, was nicht mit den Falsch-Positiven pro Detektionsfenster verwechselt werden sollte. Die erste Auswertungsmethode nutzt die statistische Tatsache, dass der Klassifikator bei Richtig-Positiven eine höhere Konfidenz als bei Falsch-Positiven aufweisen muss. Dieser Effekt wird in höheren Stufen abgeschwächt, ist aber auch vorhanden. Die Konfidenzwerte werden dann über ihrem kompletten Wertebereich auf der X-Achse in einer Frequenztabelle mit je einem Kanal für die TPs und FPs akkumuliert. Es sollten sich dann starke Anhäufungen der FPs bei geringen und der TPs bei hohen Konfidenzwerten ergeben. Je klarer die Trennung beider Verteilungen, desto besser ist das Konfidenzmaß. Für die Auswertung dieser Statistik wurden von 6000 FPs und TPs die unterschiedlichen Konfidenzwerte mittels Gleichungen (5.1.1),(5.1.2) und (5.1.3) mit unterschiedlichen Kaskadenklassifikatoren berechnet und akkumuliert. Abb. 5.2.1 zeigt die Auswertung der akkumulierten Konfidenzen. Die beste Trennung wird mittels Gl. (5.1.1) erreicht, was in Abb. 5.2.1(a) zu sehen ist. Bei den normierten Konfidenzen mit Gl. (5.1.2) in Abb. 5.2.1(b) ist keine eindeutige Trennung zu erkennen. Mit den schwellwertbasierten Entscheidungen anhand von Gl. (5.1.3) ist die Verteilung der TPs bei Konfidenzwerten um den X-Achsenbereich 7 leicht erhöht, die beiden Verteilungen weisen dennoch eine starke Überlappung auf.

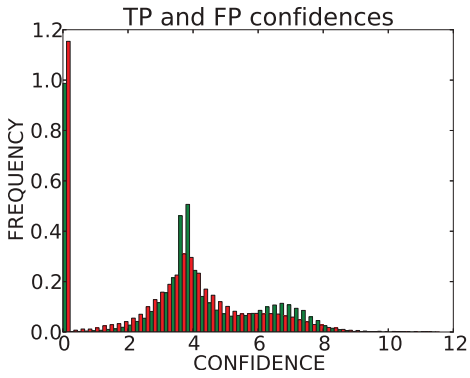
Die zweite Auswertungsmethode basiert auf einem statistisch exakten Maß der Konfidenzwerte [29], indem für eine Anzahl von K Klassifikatoren, die auf disjunkten Mengen trainiert wurden, die entsprechenden Entschei-



(a) Gl. (5.1.1)



(b) Gl. (5.1.2)



(c) Gl. (5.1.3)

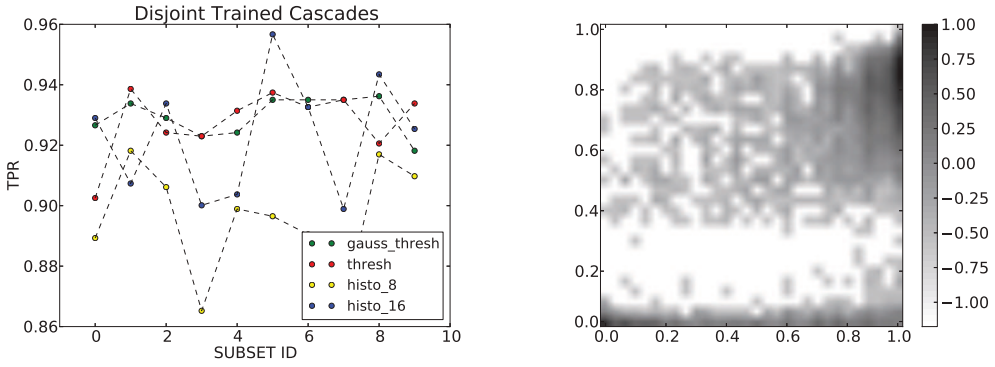
Abbildung 5.2.1: Auswertung der unterschiedlichen Konfidenzberechnungen. Die X-Achse repräsentiert die Konfidenzwerte in den entsprechenden Bereichen. Auf der Y-Achse sind die TPs durch grüne Balken und die FPs durch rote Balken gekennzeichnet.

dungen eines Musters gemittelt werden:

$$\gamma_4(h_1, \ldots, h_K, \underline{z}) = \frac{1}{K} \sum_{k=1}^{K} h_k(\underline{z}) \tag{5.2.1}$$

wobei $h_k(\underline{z} \in \mathcal{D}_{\omega_1}) = 1$ für die Objektklasse und $h_k(\underline{z} \in \mathcal{D}_{\omega_0}) = 0$. Somit ergeben sich Konfidenzwerte $\gamma_4 \in [0, \ldots, 1]$. Aus 13000 Bildern mit Objekten wurden zehn Klassifikatoren auf den disjunkten Mengen von 1300 Bildern trainiert. Um die Anzahl an unterschiedlichen Klassifikatoren zu erhöhen

wurden pro Trainingsmenge vier Klassifikatoren mit unterschiedlichen Parametern trainiert. Zur Auswertung wurden die Konfidenzwerte γ_4 (siehe Gl. (5.2.1)) aus den 40 Klassifikatoren mit einem Testset aus 6000 Bildern, das unterschiedliche Realweltschilder enthält, ermittelt und mit dem Konfidenzwert γ_1 aus Gl. (5.1.1) in einem zweidimensionalen Diagramm korreliert, was in Abb. 5.2.2(b) gezeigt wird. Die Detektionsraten der 40 unter-



(a) (b)

Abbildung 5.2.2: (a): Detektionsraten der 40 Kaskaden, die auf zehn disjunkten Trainingsmengen aus jeweils 1300 Bildern mit Objekten trainiert wurden. Jeder Punkt entspricht einem Klassifikator. Auf der Y-Achse sind die Trainingsmengen gekennzeichnet. (b): Korrelation der statistischen Konfidenz (Gl. 5.2.1) auf der X-Achse mit der Konfidenz nach Gl. (5.1.1) auf der Y-Achse und einer logarithmischen Skalierung.

schiedlichen Klassifikatoren werden in Abb. 5.2.2(a) gezeigt. Jeder Punkt entspricht der Detektionsrate eines Klassifikators. Es wurden jeweils Klassifikatoren mit Bayes-Entscheidungsfunktionen mit acht bzw. 16 Einträgen und zwei schwellwertbasierten Entscheidungsfunktionen trainiert. Bei den korrelierten Konfidenzwerten ergeben sich zwei Cluster. Ein Cluster bei den Negativ-Entscheidungen am Ursprung $(0, 0)$ und ein Cluster für die Postitiv-Entscheidungen mit Mittelpunkt $(1, 0.6)$.

Die letzte Auswertung basiert auf einer Visualisierung der Konfidenzwerte von multiplen Detektionen. Jedes Detektionsfenster wird dabei als rotes Rechteck dargestellt und der entsprechende Konfidenzwert als vertikaler Balken. Die Balken sind in absteigender Reihenfolge von links nach rechts dargestellt. Das Detektionsfenster mit der höchsten Konfidenz ist als grü-

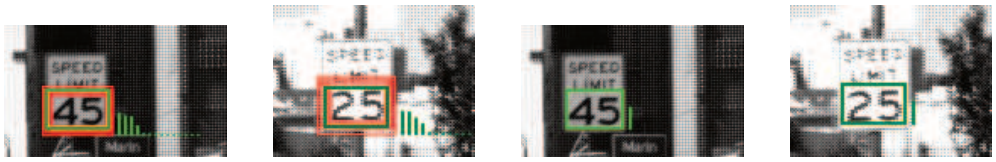
Algorithmus 5.1 Konfidenzbasierter Fusions-Algorithmus

1. **Gegeben:** Liste an Detektionsfenstern mit Konfidenzwerten, sortiert nach den Überschneidungsflächen, $\mathcal{P} = \{(r_1, \gamma_1), \ldots, (r_N, \gamma_N)\}$
2. Initialisiere Menge an fusionierten Detektionen $\mathcal{M} = \emptyset$.
3. **Iteriere** über $(r_j, \gamma_j) \in \mathcal{P}$:
   (a) $t = 0$
   (b) **Iteriere** über $(r_i, \gamma_i) \in \mathcal{M}$:
      i. Berechne Überschneidung o von r_j und r_i.
      ii. Falls $o > \theta$:
         A. $r_i \leftarrow \frac{\gamma_j}{\gamma_j+\gamma_i} r_j + \frac{\gamma_i}{\gamma_j+\gamma_i} r_i$
         B. $\gamma_i \leftarrow \gamma_i + \gamma_j$
         C. $t \leftarrow t + 1$.
   (c) **Falls** $t = 0$: $\mathcal{M} \leftarrow \mathcal{M} \cup r_j$.
4. **Erg:** Menge an fusionierten Detektionsfenstern $\mathcal{M}$

nes Rechteck dargestellt, was in Abb. 5.2.3(a) und 5.2.3(b) zu sehen ist. Mit dieser Visualisierung wurde ein Film aus den 6000 Bildern des Testsets angefertigt und evaluiert. Das Detektionsfenster mit dem höchsten Konfidenzwert ist dabei praktisch immer das zentrale Fenster, das die Ziffern bestmöglich umschließt.

Mit den Konfidenzwerten der Detektionsfenster wurde außerdem ein Fusions-Algorithmus entwickelt, der die multiplen Detektionen zu einer Detektion vereint, indem mit den Konfidenzwerten gewichtete Linearkombinationen der Rechteckkoordinaten gebildet werden, was in Abb. 5.2.3(c) und 5.2.3(d) zu sehen ist. Daraus resultiert eine exakte Position des gewichteten Detektionsfensters. Die Berechnung der Menge an fusionierten Detektionen erfolgt mit Alg. (5.1). Um korrekte Fusionen zu erzeugen, muss die Eingabeliste an Detektionsfenstern nach den Überschneidungsflächen sortiert werden. Das bedeutet, das Fentser, das die meisten Überschneidungen bezüglich der Fläche mit allen anderen Fenstern erzielt, kommt zuerst. Falls in Schritt 3(b)ii die flächenmäßige Überschneidung des neuen Fensters r_j mit einem fusionierten Fenster r_i einen Schwellwert θ überschreitet, werden die Koordi-

naten (x_1, y_1, x_2, y_2) des fusionierten Fensters in Schritt 3(b)iiA mit der neuen Konfidenz γ_j angeglichen und die fusionierte Konfidenz wird summiert. Der Algorithmus findet eine beliebige Anzahl an Clustern und fusioniert diese. Die Laufzeit für das Sortieren der Eingabeliste von Alg. (5.1) ist quadratisch. Für die restlichen Berechnungen beträgt die Laufzeit $\mathcal{O}(lN)$, wobei l die Anzahl der fusionierten Detektionen ist, die in Schleife 3b überprüft werden müssen und damit von der Ausgabe des Algorithmus abhängt. N ist die Länge der äußeren Schleife in Schritt 3.



(a) (b) (c) (d)

Abbildung 5.2.3:
(a), (b): Detektionsfenster mit entsprechenden Konfidenzen in absteigender Reihenfolge sortiert. Das Fenster mit höchster Konfidenz ist farblich grün, alle anderen sind rot.
(c), (d): Mittels Konfidenzen gewichtete Linearkombinationen der Detektionsfenster (Alg. (5.1)).

(a) (b)

Abbildung 5.2.4: Typische Szenen mit Falsch-Positiv-Detektionen. Die Falsch-Positiven haben eine geringere Konfidenz als die maximale Konfidenz der Objekte. (a): Buchstaben “ED” aus “SPEED” als Falsch-Positiv-Detektion mit geringer Konfidenz. (b): Buchstaben “as” einer Beschriftung eines LKWs mit geringer Konfidenz.

Abb. 5.2.4 zeigt typische Beispiele von Falsch-Positiv-Detektionen mit geringen Konfidenzen.

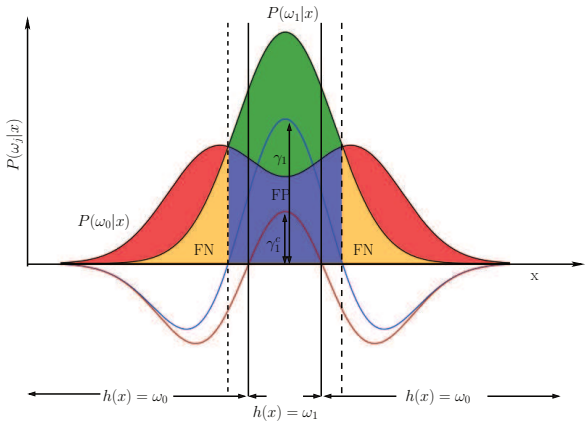
5.3 Konfidenzbasierte, integrierte Musterauswahl zur Optimierung der Klassifikatoreigenschaften

Die Methode zur konfidenzbasierten Musterauswahl bezieht sich auf Alg. (4.5) Schritt 3a, wobei repräsentative Muster für die entsprechende Stufe anhand des bestehenden Kaskadenklassifikators ausgewählt werden. Das Standardverfahren ist Falsch-Positive als negative Muster zu verwenden [115]. Das vorgestellte konfidenzbasierte Verfahren bietet den Vorteil zur Steuerung zwischen der Klassifikationsleistung und Laufzeit des Zielklassifikators, da sich das parametergesteuerte Auswahlkriterium auf diese Eigenschaften auswirkt. Das Auswahlkriterium ermöglicht die Fokusierung auf "schwierige" Beispiele, bei denen der bestehende Klassifikator unsicher ist. Somit wird auf komplexeren Beispielen trainiert und es werden im Vergleich mehr Merkmale zur Reduktion des Fehlers für die Stufenklassifikatoren verwendet. Bei der Ausführung sind die entsprechenden Stufenprobleme einfacher und der Klassifikationsfehler auf dem Testset sinkt bei gleichzeitig erhöhter Berechnungskomplexität, der Klassifikator besitzt also eine bessere Generalisierungsfähigkeit. Umgekehrt können auch "einfache" Beispiele ausgewählt werden, wodurch die mittlere Laufzeit des Klassifikators verkürzt wird und gleichzeitig der Generalisierungsfehler steigt.

Der komplexitätsgesteuerte Auswahlmechanismus basiert auf einer zusätzlichen Korrektur-Wahrscheinlichkeit ξ_j, mit der eine temporäre Modifikation der Entscheidungsregionen in jeder Stufe erfolgt, indem die A-priori-Wahrscheinlichkeiten bei den Entscheidungsfunktionen (Gl. (2.3.12)) multipliziert werden:

$$P_s^c(\omega_j) = \xi_j P_s(\omega_j) \tag{5.3.1}$$

Die Korrektur-Wahrscheinlichkeiten müssen allerdings in Alg. (4.5) Schritt 3a nach dem Schema der Tabelle in Abb. 5.3.1(b) für jede Klasse getrennt gesetzt werden, damit sich die Entscheidungsgrenzen korrekt verschieben. Abb. 5.3.1(a) zeigt den Effekt der Modifikation der positiven Wahrscheinlichkeit $P_s^c(\omega_1) = 0.6 P_s(\omega_1)$. Die Entscheidungsgrenze verschiebt sich damit von der



(a)

Prior modification			
	Alg. (4.5)	Eqs.(2.3.12),(5.3.1):	
Class	**Samples**	$P_s^c(\omega_1)$	$P_s^c(\omega_0)$
ω_0	$\mathcal{D}_s(\omega_0)$	$\xi_1 < 1$	$\xi_0 = 1$
ω_1	$\mathcal{D}_s(\omega_1)$	$\xi_1 = 1$	$\xi_0 < 1$

(b)

Abbildung 5.3.1: (a): Modifikation der positiven Wahrscheinlichkeiten $P_s^c(\omega_1) = 0.6P_s(\omega_1)$ hat den Effekt, dass mehr "nicht konfidente" negative Beispiele in der Nähe des Maximums der positiven Verteilung gewählt werden und "konfidente" Falsch-Positiv-Beispiele werden verworfen. (b): Anwendung der klassenspezifischen Gewichte auf die Entscheidungsfunktionen der Stufen $1 \leq s \leq T-1$.

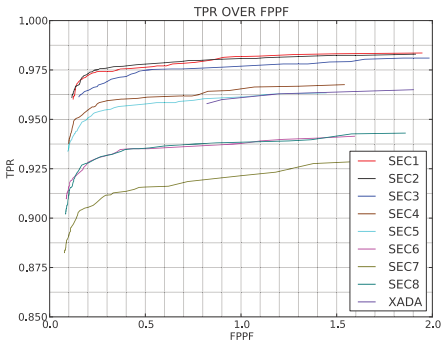
gestrichelten Linie weiter in die Mitte, was zur Folge hat, dass "nicht konfidente" negative Beispiele (FPs) gewählt werden, die näher am Maximum der positiven Verteilung liegen.

5.4 Experimentelle Auswertung zur konfidenzbasierten Musterauswahl

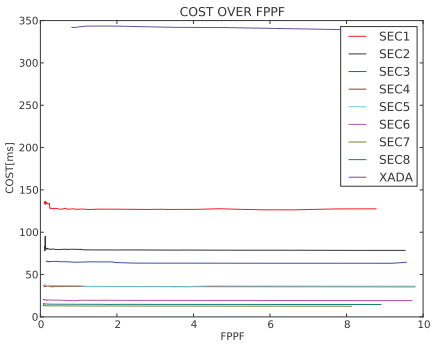
Das Verfahren zur konfidenzbasierten Musterauswahl mittels Gl. (5.3.1) und Tabelle 5.3.1(b) wurde mit 8 unterschiedlichen Parameterkombinationen getestet und wie bei allen anderen Versuchen auf dem Datensatz der in Kapitel 4, Abschnitt 4.5 beschrieben ist, ausgewertet, wobei ein aktueller Rechner mit einem 3.3 GHz Prozessor (Intel i7 - Architektur) verwendet wurde auf dem ein spezifischer Klassifikator mit einem Prozessor trainiert und mit einem Prozessor evaluiert wurde. Die Einheit der dargestellten Falsch-Positiven bei den ROC-Kurven sind Falsch-Positive pro Bild, was nicht mit den Falsch-

Positiven pro Detektionsfenster verwechselt werden sollte. Die Klassifikatoren werden mit dem Trainingsdatensatz generiert und auf dem separaten Testdatensatz, der keine Realweltschilder aus dem Trainingsdatensatz enthält, ausgewertet.

Neben den Gewichtungen für die modifizierten Entscheidungsgrenzen wurden ebenfalls unterschiedliche Signifikanzniveaus mit Alg. (3.6) getestet. Zum Vergleich wurde ein Kaskadenklassifikator mit dem klassischen AdaBoost-Verfahren für die Stufenklassifikatoren trainiert, die wiederum aus schwachen Klassifikatoren mit Haar-ähnlichen Merkmalen zusammengesetzt sind [116, 115]. Da die Anzahl der Objektklasse im Gegensatz zur Hintergrundklasse gering ist, wurden alle positiven Beispiele für die Stufenklassifikatoren verwendet, was gleichbedeutend ist mit der Parametereinstellung $\xi_0 = 0$ in Gl. (5.3.1). Abb. 5.4.1(a) zeigt die ROC-Kurven der Detektionsraten



(a) TPR über FP pro Bild



(b) Mittlere Laufzeit pro Bild

Konfidenzbasierte Musterauswahl: Klassifikatorkonfigurationen									
ID	**SEC1**	**SEC2**	**SEC3**	**SEC4**	**SEC5**	**SEC6**	**SEC7**	**SEC8**	**XADA**
Mod. Wahrsch. ξ_1	0.75	0.75	0.9	0.9	1	1	1.1	1.1	-
Signifikanzniveau α	0.025	0.05	0.025	0.05	0.025	0.05	0.025	0.05	-

(c) Klassifikatorkonfigurationen

Abbildung 5.4.1: Evaluierung der konfidenzbasierten Musterauswahl (a): ROC-Kurven über den Falsch-Positiven pro Bild (b): Mittlere Laufzeit pro Bild in Millisekunden (c): Klassifikatorkonfigurationen

über den Falsch-Positiven pro Bild und in Abb. 5.4.1(b) sind die entsprechenden mittleren Laufzeiten über den Falsch-Positiven pro Bild eingezeichnet. Die Konfigurationen der Klassifikatoren sind mittels der Legenden in Tabelle 5.4.1(c) dargestellt. Die Größe der Bilder beträgt 752×480 Pixel, wobei

das komplette Bild abgesucht wird (full scan). Es fällt auf, dass je niedriger die Gewichtung der A-priori-Wahrscheinlichkeiten ist, desto besser ist die Klassifikationsleistung und desto höher sind die mittleren Laufzeiten. Durch die konfidenzbasierte Musterauswahl sind Verbesserungen der Klassifikationsraten von bis zu 5% zu erkennen. Außerdem fällt auf, dass die Klassifikationsraten zwischen den Signifikanzniveaus von 2.5% und 5% ähnlich sind, jedoch bei deutlich besseren Laufzeiten mit dem 5%-Niveau. Die Klassifikatoren SEC1 bis SEC5 schlagen die AdaBoost-Kaskade (XADA) sowohl in der Klassifikationsleistung als auch in der mittleren Laufzeit, die bei 0.8 Falsch-Positiv-Detektionen pro Bild in die Sättigung gerät. Mit der Konfiguration SEC5 wird eine Verbesserung der Laufzeit um einen Faktor von 9 erreicht, bei vergleichbarer Detektionsrate. Der schnellste Klassifikator (SEC8) erreicht Detektionsraten von 93% bei 0.2 Falsch-Positiven pro Bild mit einer mittleren Laufzeit von 15 ms oder 66 FPS mit einem 3.3 GHz Prozessorkern. Die Klassifikatoren mit stärkeren Gewichten erreichen Detektionsraten von 97% bei 0.2 Falsch-Positiv-Detektionen pro Bild und mittleren Laufzeiten von ca. 80 ms.

5.5 Semi-überwachte, integrierte Kaskadendetektion und Datenexploration

Semi-überwachte Lernalgorithmen nutzen sowohl Beispiele, die von einem Menschen zu Klassen zugeordnet wurden, als auch nicht spezifizierte Beispiele. Da die Zuordnung durch einen menschlichen Experten im Gegensatz zur automatisierten Zuordnung im Allgemeinen viel Zeitaufwand benötigt sind diese semi-überwachten Lernverfahren (SSL) [12, 26, 51, 124, 127] sehr interessant. Erwähnenswerte Techniken des SSL sind das Self-training und Co-training [129], Verfahren basierend auf generativen Modellen [33], auf Graphen basierende Verfahren [14, 110], Verfahren die Support Vector Machines verwenden [11] und semi-überwachte lineare Diskriminanzanalyse [23].

Für Detektionsaufgaben bei denen sich die Mengen der Klassen um mehrere Größenordnungen unterscheiden und die wesentliche größere Hintergrundklasse nahezu alle möglichen Objekte, darunter auch die der Objektklasse enthalten kann, sind die SSL-Verfahren attraktiv. Die Meisten SSL-Verfahren zur Detektion verwenden geboostete Klassifikatoren [80, 66, 117, 72] mit diskreten Entscheidungsfunktionen, die zusätzlich ein Ähnlich-

keitsmaß verwenden und als Wrapperverfahren für das Kaskadentraining spezifiert sind. Leistner et al. [72] erweitern das Semiboost-Verfahren [80], indem das Ähnlichkeitsmaß gelernt wird. Das hier vorgestellte Verfahren zur semi-überwachten Kaskadendetektion unterscheidet sich im Wesentlichen von den bestehenden Verfahren, indem das Ähnlichkeitsmaß direkt über die Bayes'sche Konfidenz der Entscheidungsfunktion verwendet wird, was ebenfalls "gelernt" wird und die Konfidenzen aller Stufenklassifikatoren fusioniert. Dadurch verändert sich die Konfidenz mit jeder Stufe, und die semiüberwachte Auswahl der Beispiele findet für jeden neuen Stufenklassifikator statt, es handelt sich also um ein in das Kaskadentraining *integriertes* Verfahren.

Es wird angenommen, dass die Hintergrundklasse alle möglichen Objekte enthält, worunter sich auch nicht spezifizierte Objekte der Objektklasse befinden. Die grundlegende Idee ist deshalb, dass sich der Klassifikator zu einem bestimmten Maß "selbst weiterbilden" kann, indem er Hintergrundbeispiele, bei denen er sich sicher ist, dass sie zur Objektklasse gehören, selbst umdefinieren darf und von der Hintegrundklasse in die Objektklasse verschieben kann. Diese Beispiele werden im Folgenden *migrierte Objekte* genannt und der Vorgang nennt sich *Migration*:

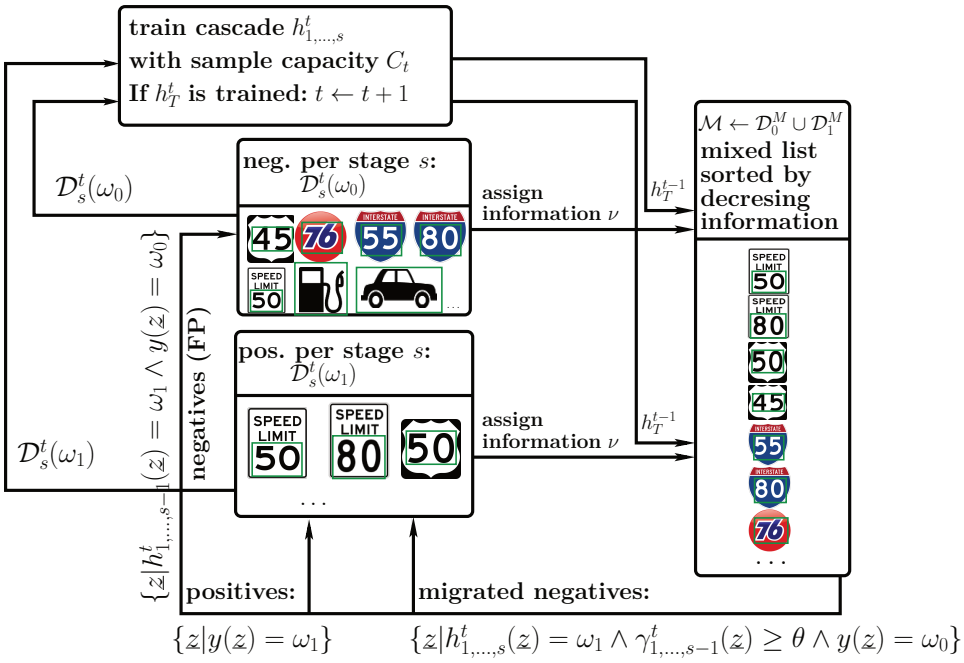
$$\mathcal{D}_s^t(\omega_1) \leftarrow \begin{cases} \mathcal{D}_s^t(\omega_1) \cup \underline{z} & \text{falls } \gamma(\underline{z}) \geq \gamma_{min} + \theta(\gamma_{max} - \gamma_{min}) \text{ } \textit{(Migration)} \\ \mathcal{D}_s^t(\omega_1) \cup \underline{z} & \text{falls } y(\underline{z}) = \omega_1 \\ \mathcal{D}_s^t(\omega_1) \cup \emptyset & \text{sonst} \end{cases} \tag{5.5.1}$$

wobei $\mathcal{D}_s^t(\omega_1)$ die positive Trainingsmenge für den Stufenklassifikator h_s im Evolutionsschritt t entspricht und $\underline{z}$ aus der Objekt- und Hintergrundklasse stammen kann. Somit findet bei der Migration ebenfalls ein *Explorationsschritt* statt, bei dem unbekannte Muster in die Objektlasse mit aufgenommen werden können. Die Konfidenzen γ werden mit den Gleichungen (5.1.1) und (5.1.2) berechnet. In jedem Evolutionsschritt wird ein Kaskadenklassifikator mit erhöhter Kapazität für die Objektklasse trainiert. Um interessante Beispiele zu identifizieren wird eine gemischte Liste verwendet, die aus positiven und negativen Beispielen besteht und jedem Beispiel wird ein Informationswert zugewiesen.

Algorithmus 5.2 Semi-überwachtes, integriertes Kaskadentraining

1. **Gegeben:** Beispiele $\mathcal{D}(\omega_i)$ für $i \in \{0, 1\}$ und Kapazität κ
2. Trainiere erste Kaskade h_1^1 mit $\mathcal{D}_1^1$
3. Iteriere über Kaskaden $t \in 2, \ldots, T$:
   (a) Für jeden Stufenklassifikator h_t^s:
      i. Initialisiere gemischte Liste $\mathcal{M} \leftarrow \emptyset$
      ii. Iteriere über Muster $\underline{z} \in \mathcal{D}(\omega_i)$:
         A. Berechne Information $\nu(\underline{z})$ mit h_{t-1}^C
         B. $\mathcal{M} \leftarrow \mathcal{M} \cup \underline{z}$
      iii. Sortiere $\mathcal{M}$ nach absteigender Information $\nu(\underline{z})$ mit $\underline{z} \in \mathcal{M}$
      iv. $\nu_{\max} = \arg\max_{\nu(\underline{z})}(\mathcal{M})$, $\nu_{\min} = \arg\min_{\nu(\underline{z})}(\mathcal{M})$
      v. Iteriere über Muster $\underline{z} \in \mathcal{M}$:
         A. Erweitere $\mathcal{D}_s^t(\omega_1)$ mit Gl. (5.5.1) und $h_t(h_1, \ldots, h_{s-1})$ mit beschränkter Kapazität κ
         B. Erweitere $\mathcal{D}_s^t(\omega_0)$ mit zufälliger Auswahl von FPs mit beschränkter Kapazität κ
      vi. Trainiere Stufe h_t^s mit $\mathcal{D}_s^t(\omega_i)$ für $i \in \{0, 1\}$
   (b) **Erg:** Kaskade $h_t^C = (h_t^1, \ldots, h_t^N)$
   (c) Erhöhe Kapazität κ für nächsten Zyklus t

Alg. (5.2) zeigt das semi-überwachte, integrierte Kaskadentraining. Die Schleife in Schritt 3a iteriert über die Kaskadenklassifikatoren mit erhöhter Kapazität (Schritt 3c). Der Informationswert wird in Schritt 3(a)iiA berechnet und die gemischte Liste $\mathcal{M}$, bestehend aus Objekt- und Hintergrundbeispielen wird aufgebaut. $\mathcal{M}$ dient als Rangliste der "interessanten Beispiele", indem absteigend nach dem Informationswert in Schritt 3(a)iii sortiert wird. In der Schleife 3(a)v wird über die Elemente der Liste iteriert und die Beispiele nach Gl. (5.5.1) zur Objektklasse hinzugefügt. Somit können auch Positivbeispiele, die weiter unten in der Liste vorkommen, durch die beschränkte Kapazität der Objektklasse verworfen werden, indem migrierte Hintergrundbeispiele mit höherer Konfidenz vorgezogen werden.



(a)

Semi-überwachte Konfigurationen		
ID	Information	Konfidenz
Bayes	$\nu_1 = 1/(1+\gamma_1)$	γ_1
Bayes norm.	$\nu_2 = 1/(1+\gamma_2)$	γ_2
Inverse	$\nu_3 = \gamma_1$	γ_1
Random	$\nu_4 = \text{rand}[0,\dots,1]$	$\text{rand}(0,\dots,1)$

(b)

Abbildung 5.5.1: (a): Graph des semi-überwachten, integrierten Kaskadentrainings. Jede Schleife t generiert Klassifikatoren h^t_T mit erhöhter Kapazität der Objektklasse. Als Positive werden für jedes Stufenproblem sowohl wahre Positive mit hoher Konfidenz verwendet, als auch Negative mit hoher Konfidenz. Um die positiven und migrierten Muster zu bestimmen wird eine gemschichte Liste $\mathcal{M}$ verwendet, indem jedem Beispiel ein Informationswert ν zugeordnet wird und die Liste nach diesem Wert absteigend sortiert wird. Falls der Konfidenzwert die Schwelle θ übersteigt, wird das Negativbeispiel zum migrierten Positivbeispiel. (b): Kombinationen der unterschiedlichen Konfidenzwertberechnung γ und Informationswerte ν zum semi-überwachten Training.

Die wesentlichen Mechanismen sind zum einen die Identifikation "interessanter" Beispiele über die Rangliste der Informationswerte und zum anderen die klassifikatorbasierte Migration aus der Hintergrundklasse in die Objektklasse mittels Gl. (5.5.1). Abb. 5.5.1(a) zeigt die Funktionsweise des Algorithmus (5.2). Durch die gemischte Liste $\mathcal{M}$ können nicht identifizierte Objekte aus der Hintergrundklasse, die starke Ähnlichkeiten mit der Objektklasse aufweisen, wie zum Beispiel die unterschiedlichen Ziffernkombinationen, die aus Interstate-Highway-Tafeln oder aus Werbetafeln stammen, in die Objektklasse migrieren. Da der Detektor mit den zweistelligen Ziffernkombinationen belernt wird, sind andere Ziffernkombinationen, die detektiert werden, hilfreich. Objekte aus der Objektklasse, die starke Abweichungen aufweisen, falls ein Muster zum Beispiel falsch ausgeschnitten wurde, können verworfen werden. Die unterschiedlichen Kombinationen zur Berechnung des Informationswertes ν und der Konfidenzen γ sind in Tabelle 5.5.1(b) aufgelistet.

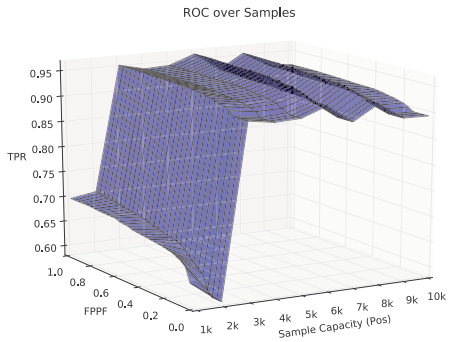
5.6 Experimentelle Auswertung der semi-überwachten, integrierten Kaskadendetektion

Wie in den anderen Versuchen wurde die Evaluierung auf dem in Kapitel 4, Abschnitt 4.5 beschriebenen Datensatz durchgeführt. Die Klassifikatoren werden mit dem Trainingsdatensatz generiert und auf dem separaten Testdatensatz, der keine identischen Realweltschilder enthält, ausgewertet. Um die Funktionsweise und die Eigenschaften der semi-überwachten Kaskadendetektion zu erforschen wurde bei Alg. (5.2) mit einer Kapazität von 1000 positiven und negativen Beispielen begonnen. Nach jedem Evolutionszyklus wird die Kapazität in Schritt 3c um 1000 erhöht. Da die Trainingszeiten mehrere Wochen überschreiten können, wurde die maximale Anzahl an Evolutionszyklen auf $T = 10$ gesetzt, wodurch bei jedem Durchlauf zehn Kaskadenklassifikatoren generiert werden. Der letzte Klassifikator wird mit 10000 Positiv- und Negativbeispielen belernt. Insgesamt wurden mehr als 100 Kaskadenklassifikatoren trainiert, woraus die Klassifikatoren mit den aussagekräftigsten Ergebnissen gewählt wurden. Um die Trainingszeiten zu reduzieren wurde eine eingeschränkte Merkmalsmenge in den Auflösungsstufen verwendet. Durch die unterschiedlichen Kapazitäten bei den Beispielen der einzelnen Klassi-

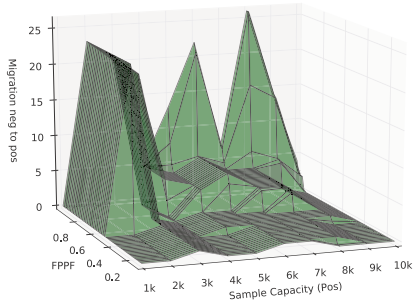
fikatoren wurde die Anzahl der Einträge bei den Frequenztabellen für die Entscheidungsfunktionen anhand der Kapazität κ berechnet:

$$e = 0.02\kappa \tag{5.6.1}$$

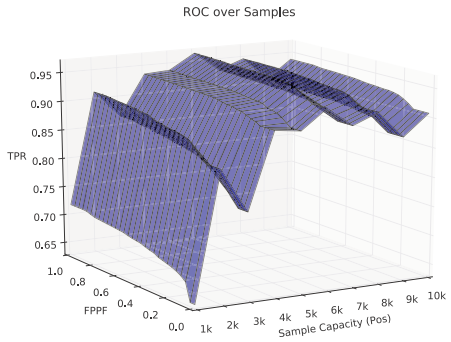
wobei e die Anzahl der Einträge darstellt. Die Häufigkeiten der Klassenbeispiele für einen Eintrag sind somit höher bei einer geringen Kapazität.



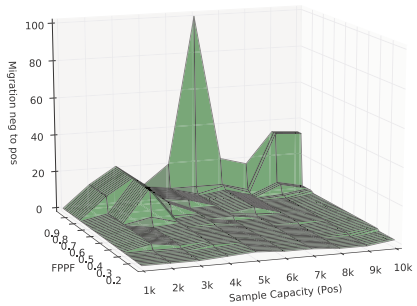
(a)a $\theta = 0.85$ (Gl. (5.5.1))



(a)b $\theta = 0.85$ (Gl. (5.5.1))



(b)a $\theta = 0.95$ (Gl. (5.5.1))

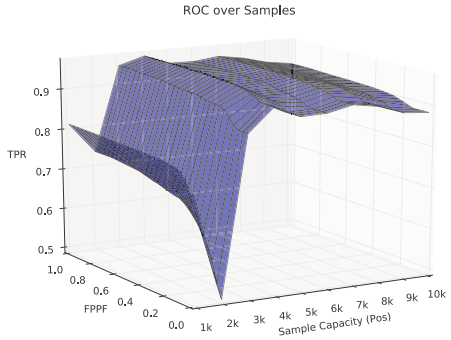


(b)b $\theta = 0.95$ (Gl. (5.5.1))

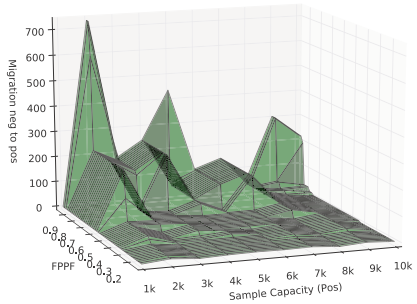
Abbildung 5.6.1: Konfidenz in Gl. (5.5.1): γ_1 (Gl. (5.1.1)), Information $\nu_1 = \frac{1}{1+\gamma_1}$ (Alg. 5.2, Schritt 3(a)iiA)

Die blau eingefärbten Abbildungen auf den linken Seiten der folgenden Auswertungen, wie z.B. Abb. 5.6.1, zeigen dreidimensionale ROC-Kurven auf dem Testset mit unbekannten Realweltbeispielen, indem die Detektionsraten auf der vertikalen Achse und die Falsch-Positiven pro Bild auf der

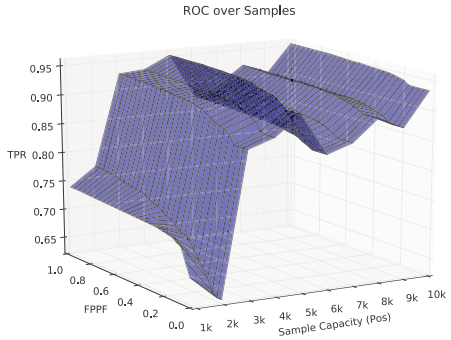
Achse in die Bildebene eingezeichnet sind. Die Evolutionszyklen sind auf der horizontalen Achse aufgetragen, wobei die Beschriftung 1K bis 10K den Kapazitäten der Trainingsmengen der Klassifikatoren entspricht. Die grün markierten Abbildungen auf den rechten Seiten der folgenden Auswertungen zeigen auf der vertikalen Achse die Mengen der migrierten Hintegrundbeispiele, die in die Objektklasse aufgenommen wurden und mit Beispielen niedriger Konfidenz ausgetauscht wurden. Jede Reihe gehört zum gleichen semi-überwachten Training und zeigt die Detektionsraten beim entsprechenden Zuwachs der migrierten Beispiele. Auf den beiden anderen Achsen sind ebenfalls die Falsch-Positiven pro Bild und die Kapazitäten aufgetragen. Die Anzahl der Migrierten ist somit für einen spezifischen Klassifikator aufgetragen. Für fallende Falsch-Positiv-Raten gehören die Detektionsraten und Migrationszahlen zu einem Kaskadenklassifikator mit steigender Anzahl an Stufen. Somit wird ein Kaskadenklassifikator mit den kumulierten Migrierten trainiert, wobei bei steigender Anzahl an Stufen der Zuwachs an migrierten Mustern sinkt. Abb. 5.6.1 zeigt die Auswertung des Trainings mit der Konfidenzberechnung nach Gl. (5.5.1) über die nicht normierten fusionierten Wahrscheinlichkeitsdichten. Die Information wird über die inversen Konfidenzen berechnet. Muster mit großer Unsicherheit des Klassifikators werden somit ausgewählt. Als Schwelle für die migrierten Negativbeispiele wird in den Abb. 5.6.1(a)a und 5.6.1(a)b $\theta = 0.85$ und in den Abb. 5.6.1(b)a und 5.1(b)b $\theta = 0.95$ verwendet. Bei Abb. 5.6.1(a)a fällt der steile Anstieg der Detektionsperformance zwischen den Kapzitäten 2K und 3K auf. Dieser Anstieg ist bei Abb. 5.6.1(b)a weniger stark ausgeprägt und es ist ein Einbruch bei 3K erkennbar. Außerdem schwanken die Detektionsraten zwischen den Klassifikatoren mit unterschiedlichen Kapazitäten stärker.



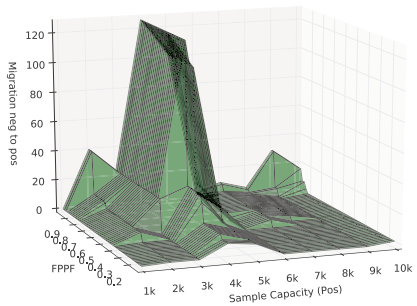
(a)a $\theta = 0.85$ (Gl. (5.5.1))



(a)b $\theta = 0.85$ (Gl. (5.5.1))



(b)a $\theta = 0.95$ (Gl. (5.5.1))

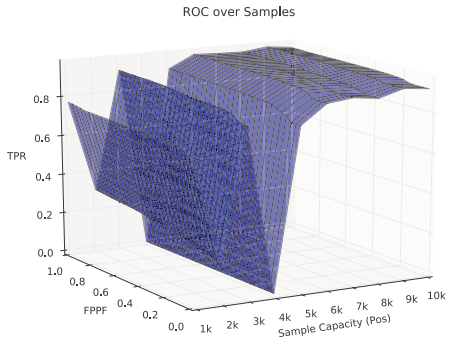


(b)b $\theta = 0.95$ (Gl. (5.5.1))

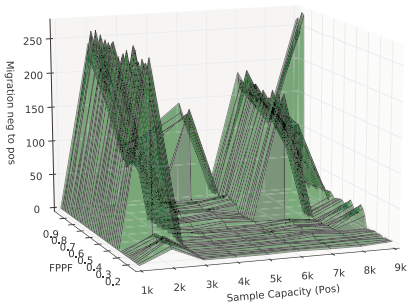
Abbildung 5.6.2: Konfidenz in Gl. (5.5.1): γ_2 (Gl. (5.1.2)), Information $\nu_2 = \frac{1}{1+\gamma_2}$ (Alg. 5.2, Schritt 3(a)iiA)

In Abb. 5.6.2 wurde die Berechnung der Konfidenzen mit den normierten Wahrscheinlichkeitsdichten mittels Gl. (5.1.2) durchgeführt. Als Informationswert wird ebenfalls die inverse Konfidenz verwendet. In den Abb. 5.6.2(a)b ist zu erkennen, dass die hohe Anzahl an Migrationen bei einer Kapzität von 2K zu einer deutlichen Verschlechterung der Klassifikationsleistung in Abb. 5.6.2(a)a führt, da die Statistik für die Konfidenzwertberechnung zu gering ist. Außerdem führt die Normierung der Konfidenzen über die Wahrscheinlichkeitsdichten zu dem Effekt, dass Einträge der Frequenztabelle mit geringer Evidenz einer Klasse ohne Gegenbeispiele zur höchsten Konfidenz von 100 % hochgewichtet werden. Dieser Effekt sorgt unter anderem dafür, dass mehr Beispiele migriert werden. In Abb. 5.6.2(b)a schwanken die Detekti-

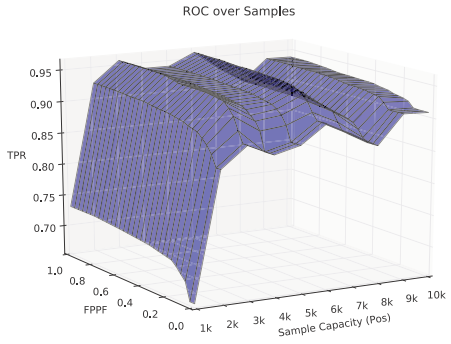
onsraten zwischen den Kapazitäten, ähnlich wie in Abb. 5.6.1, stärker als in Abb. 5.6.2(a)*a*.



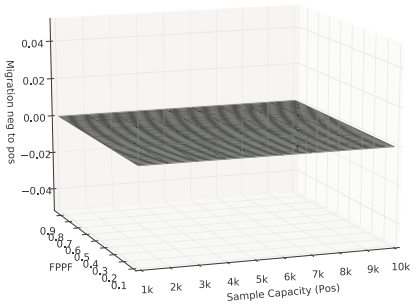
(a)*a* $\theta = 0.9$ (Gl. (5.5.1))



(a)*b* $\theta = 0.9$ (Gl. (5.5.1))



(b)*a* $\theta = 1$ (Gl. (5.5.1))

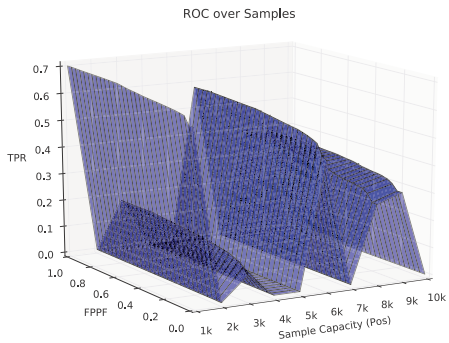


(b)*b* $\theta = 1$ (Gl. (5.5.1))

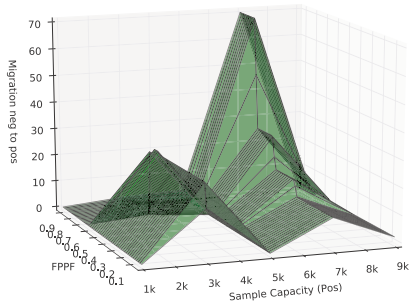
Abbildung 5.6.3: Konfidenz in Gl. (5.5.1) $\gamma = \text{rand}[0, \ldots, 1]$, Information $\nu_4 = \text{rand}[0, \ldots, 1]$

In Abb. 5.6.3 wurden die Konfidenzen und die Informationswerte mit zufälligen Werten zwischen Null und Eins bestimmt. Es ist in Abb. 5.6.3(a)*b* zu erkennen, dass die Klassifikatoren mit geringen Kapazitäten völlig instabil sind. Bei höheren Kapazitäten sind die Detektionsraten im Vergleich schlechter als die semi-überwachten Detektoren mit den vorgestellten Konfidenzen. In Abb. 5.6.3(a)*b* lassen sich wesentlich größere Schwankungen der Migrationen erkennen. In Abb. 5.6.3(b)*a* werden durch $\theta = 1$ die Migrationen vermieden (Abb. 5.6.3(b)*b*). Somit sind diese Klassifikatoren vollständig

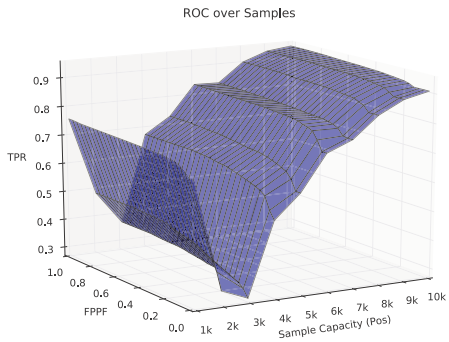
überwacht trainiert und dienen als Vergleich. Obwohl die Detektionsraten bei höheren Falsch-Positiv-Raten hoch sind, ist der Vorteil der semi-überwachten Detektoren deutlich erkennbar: Je niedriger die Falsch-Positiv-Raten, desto stärker sinkt auch die Klassifikationsleistung in Abb. 5.6.3(b)a, was durch die Neigung der Fläche ersichtlich ist. Im Gegensatz dazu ist bei den Abb. 5.6.1(a)a und 5.6.2(a)a dieser Effekt kaum erkennbar.



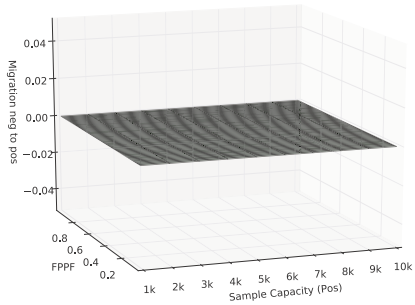
(a)a $\theta = 0.995$ (Gl. (5.5.1))



(a)b $\theta = 0.995$ (Gl. (5.5.1))



(b)a $\theta = 1$ (Gl. (5.5.1))



(b)b $\theta = 1$ (Gl. (5.5.1))

Abbildung 5.6.4: Konfidenz in Gl. (5.5.1): γ_1 (Gl. (5.1.1)), Information $\nu_3 = \gamma_1$ (Alg. 5.2, Schritt 3(a)iiA)

In Abb. 5.6.4 wurde der Informationswert über die Konfidenzen berechnet um die Auswirkungen auf die Klassifikatoren zu testen. Es werden also immer die Beispiele gewählt, bei denen sich der Klassifikator sicher ist. In Abb. 5.6.4(a)a und 5.6.4(a)b werden zusätzlich Beispiele migriert. Die Klassi-

fikatoren werden durch die Migration absolut instabil. Dieser Effekt lässt sich durch die zu schwach repräsentative Statisitik im Trainingsset erklären. Für den Fall, dass keine Migrationen stattfinden (Abb. 5.6.4(b)a und 5.6.4(b)b) lässt sich erkennen, dass die Detektionsraten erst bei großen Kapazitäten, bei denen der Effekt der Musterauswahl durch den Informationswert sich verringert, besser werden. Bei Betrachtung der Laufzeiten der Klassifikatoren erkennt man, dass die Klassifikatoren mit den schlechten Detektionsraten auch sehr geringe Laufzeiten von wenigen Millisekunden aufweisen. Die Klassifikatoren sind allerdings durch die niedrigen Detektionsraten nicht verwendbar.

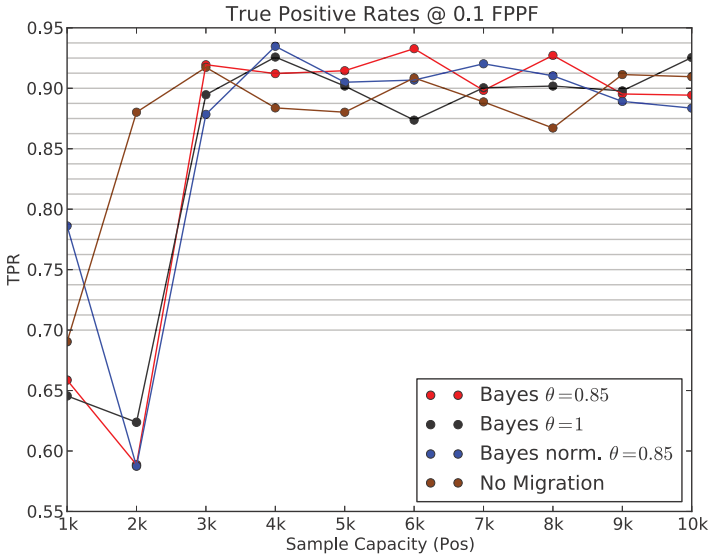
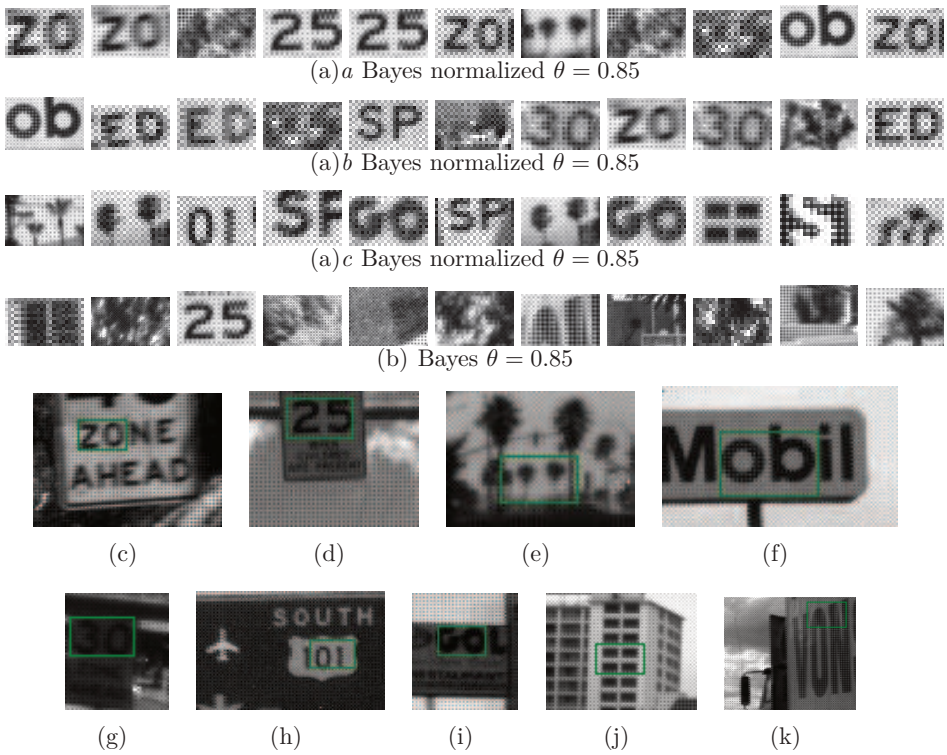


Abbildung 5.6.5: Gegenüberstellung der Detektionsraten der semi-überwachten Detektoren bei 0.1 Falsch-Positiven pro Bild.

In Abb. 5.6.5 sind die Detektionsraten der semi-überwachten Klassifikatoren mit den besten Detektionsergebnissen im Vergleich zum vollständig überwachten Fall (braune Kurve) bei 0.1 Falsch-Positiven pro Bild über den Kapazitäten dargestellt. Ab einer Kapazität von 4K sind die Klassifikationsraten der semi-überwachten Klassifikatoren den überwachten Klassifikatoren deutlich überlegen. Der Klassifikator mit maximaler Klassifikationsleistung ist der semi-überwachte Klassifikator bei einer Kapazität von 4000 Mustern,

also weniger als die Hälfte der maximalen Anzahl an Mustern und der Konfidenzberechnung mit den normierten Wahrscheinlichkeitsdichten (Gl. (5.1.2)). Die durchschnittliche Laufzeit dieses Klassifikators beträgt 25 Bilder pro Sekunde (Echtzeit). Die schwache Klassifikationsleistung der semi-überwachten Fälle bei der Kapazität 2K lässt sich dadurch erklären, dass die Menge der Muster zur Berechnung der Konfidenzen, die auf den statistischen Häufigkeiten der Beispielen basiert, zu gering ist. Generell lässt sich die Klassifikationsleistung noch verbessern, indem weniger Einträge für die Frequenztabellen verwendet werden, was allerdings längere mittlere Laufzeiten zur Folge hat und mit der Echtzeitanforderung kollidiert. Um die Ähnlichkeiten der migrierten Beispiele mit der Objektlasse zu überprüfen wurden die migrierten Hintergrundbeispiele in absteigender Reihenfolge der Konfidenz sortiert und inspiziert.



(a)a Bayes normalized $\theta = 0.85$

(a)b Bayes normalized $\theta = 0.85$

(a)c Bayes normalized $\theta = 0.85$

(b) Bayes $\theta = 0.85$

(c) (d) (e) (f)

(g) (h) (i) (j) (k)

Abbildung 5.6.6: Migrierte Hintergrundbeispiele, die in die Objektklasse aufgenommen wurden, sortiert nach absteigenden Konfidenzen (a)a – (a)c: Norm. Bayes Konfidenz (siehe Tabelle 5.1(b)), (b): Bayes Konfidenz (c) – (k): Kontext einiger Beispiele

Abb. 5.6.6 zeigt die ersten migrierten Hintergrundbeispiele, die in die Objektklasse aufgenommen wurden, mit den höchsten Konfidenzwerten in absteigender Reihenfolge. In Abb. 5.6(c) - 5.6(k) sind die Kontexte einiger Bilder dargestellt. Die Bilder weisen eine starke Ähnlichkeit mit der Beispielklasse auf und sind somit geeignete Beispiele. Das erste migrierte Beispiel in Abb. 5.6(a)a stammt von einem “40 Zone Ahead”-Schild (Abb. 5.6(c)). Bereits das vierte Beispiel in Abb. 5.6(a)a stammt von einem Speed-Limit-Schild, das bei der Annotation der Schilder übersehen wurde. Ein Großteil der migrierten Beispiele sind Buchstaben- und Ziffernpaare, die von Werbetafeln (sieh Abb. 5.6(f) und 5.6(i)) oder von der Beschriftung innerhalb der Schilder (siehe erstes und zweites Bild in Abb. 5.6(a)a und zweites und drittes Bild in Abb. 5.6(a)b), stammen. Selbst Beispiele die keine Ziffernpaare

repräsentieren, wie z.B. die zwei Palmen (siebtes Beispiel in Abb. 5.6(a)*a* und 5.6(e)), weisen strukturelle Ähnlichkeiten mit den Objekten auf.

5.7 Zusammenfassung

In diesem Kapitel wurden im Abschnitt 5.1 unterschiedliche Konfidenzwertberechnungsmethoden zur Fusion der Konfidenzen der einzelnen Stufenklassifikatoren eines Kaskadenklassifikators präsentiert, die im Rahmen dieser Arbeit entwickelt wurden. Die unterschiedlichen Berechnungsmethoden werden in der experimentellen Auswertung in Abschnitt 5.2 in zwei unterschiedlichen Versuchen und mit einer visuellen Überprüfung untersucht. Beim ersten Versuch erzielten die nichtnormierten wahrscheinlichkeitsbasierten Konfidenzen das beste Resultat. Beim zweiten Versuch wurden die nichtnormierten wahrscheinlichkeitsbasierten Konfidenzen mit einem statistisch exakten Konfidenzmaß korreliert. Bei der Auswertung war jeweils für die Negativ- und für die Positiventscheidung ein Cluster zu erkennen. Bei der letzten visuellen Auswertung wurden die multiplen Detektionen eines Objektes im Bild absteigend mit dem Konfidenzwert sortiert und das Detektionsfenster mit maximaler Konfidenz farblich hervorgehoben. Die visuelle Überprüfung der Videosequenzen auf dem Testset hat gezeigt, dass das Detektionsfenster mit maximaler Konfidenz nahezu immer das zentrierte Detektionsfenster mit minimalem Abstand zu den im Trainingsdatensatz annotierten Objekten war. Außerdem wurde der konfidenzbasierte Fusionsalgorithmus getestet und es konnte gezeigt werden, dass eine stabile Klassifikation mit exakter Positionierung des fusionierten Detektionsfenster resultiert.

Im Abschnitt 5.3 wurde ein im Rahmen dieser Arbeit entwickeltes Verfahren vorgestellt, das eine konfidenzbasierte, integrierte Musterauswahl ermöglicht. Mit diesem Verfahren werden für die entsprechende Klasse diejenigen Muster mit geringer Konfidenz vermehrt zum Training der Stufenklassifikatoren ausgewählt, indem die A-priori-Wahrscheinlichkeiten modifiziert werden. Somit wird im Vergleich zur Ausführung des Klassifikators beim Training auf "härteren Problemen" trainiert. Dieser Effekt lässt sich mit einer Gewichtung fein justieren. Mit diesem Verfahren kann der Kaskadenklassifikator ebenfalls zwischen den Eigenschaften Detektionsleistung und Laufzeitperformance eingestellt werden. Im Abschnitt 5.4 wurde das Verfahren mit unterschiedlichen Parametern getestet. Mit dem Verfahren konnten sich die Detektionsraten um bis zu 5 % steigern lassen. Allerdings steigen ja nach Konfiguration auch

die mittleren Laufzeiten der Klassifikatoren an. So wurden mit dem Verfahren zum Beispiel 98.2%@1FPPF bei 127 ms mit einem Signifikanzniveau von 5 % und einem Gewichtungsfaktor von 0.75 erreicht. Bei einem Faktor von 0.9 und einem Signifikanzniveau von 2.5 % wurden 97.7%@1FPPF bei 65 ms mittlerer Detektionszeit pro Bild erreicht. Der schnellste Klassifikator erzielte eine Detektionsrate von 93.8%@1FPPF bei 15 ms Detektionszeit pro Bild, was einer Aktualisierungsrate von 67 FPS entspricht.

Im Abschnitt 5.5 wurde ein neues, im Rahmen dieser Arbeit entwickeltes teilüberwachtes, integriertes Lernverfahren für Kaskadenklassifikatoren vorgestellt, bei dem der Klassifikator die Möglichkeit besitzt Beispiele aus der Hintergrundklasse, die mit hoher Wahrscheinlichkei zur Objektklasse gehören in die Objektklasse zu migrieren und gleichzeitig interessante Negativbeispiele für die Stufenklassifikatoren auswählt. Das Migrieren der interessanten Beispiele in die Objektklasse ist vor allem bei Vollbilddetektoren sinnvoll, da zum einen oft Objekte in der Hintergrundklasse vorkommen, die vom Mensch übersehen wurden und zum anderen auch viele Objekte in der Hintergrundklasse existieren, die dem tatsächlichen Objekten sehr ähnlich, und somit beim Lernen als Objekte hilfreich sind. Im Abschnitt 5.6 wird das semi-überwachte Verfahren ausgewertet, indem dreidimensionale ROC-Kurven gezeigt werden und zusätzlich die migrierten Muster über den Falsch-Positiven und den Kapazitäten. Ausserdem wurden die Detektionsraten für die minimalen Falsch-Positiven von 0.1 FPPF pro Bild gegenübergestellt. Die semi-überwachten Klassifikatoren erzielten mit mittelgroßen Objektmengen die besten Detektionsraten im Vergleich zu den überwachten Klassifikatoren. Bei inverser Sortierung der Muster konnten allerdings keine funktionsfähigen Klassifikatoren erzeugt werden. Auch die Wahl der Paramter hatte großen Einfluss auf die Performance der Klassifikatoren.

Die migrierten Beispiele der Klassifikatoren wurden absteigend nach den Konfidenzwerten sortiert und in einer Abbildung dargestellt, wobei gezeigt werden konnte, dass alle migrierten Hintergrundbeispiele starke Ähnlichkeiten zur Objektklasse aufweisen. Bereits das vierte migrierte Muster rührte von einem Verkehrsschild, das beim Annotieren der Bilder übersehen wurde.

Kapitel 6

Ensemble- und Aggregationsverfahren

In diesem Kapitel werden Ensemble- und Aggregationsverfahren vorgestellt, die im Gegensatz zu Kapitel 3 nicht auf Merkmalen basieren, sondern deren Basiseinheiten beliebige Klassifikatoren sind. Die grundlegende Idee bei diesen Verfahren ist, den Klassifikationsfehler eines einzelnen Klassifikators zu verbessern, indem mehrere Klassifikatoren trainiert und zu einem Ensembleklassifikator mit geringerem Fehler vereint werden. Lehrbücher die sich eingehend mit dieser Art von Verfahren beschäftigen ist zum Beispiel bei Rokach [99] und Kuncheva [70] zu finden. Der Kaskadenklassifikator ist ein Beispiel eines Ensembleklassifikators.

Zum einen wird in diesem Kapitel ein neues Verfahren entwickelt, bei dem baumartige Ensembleklassifikatoren generiert werden, die mit steigender Komplexität der Klassifikationsaufgaben eine steigende Vernetzung und Anzahl an Split- und Blattklassifikatoren aufweisen. Die Anzahl der an der Entscheidung beteiligten Basisklassifikatoren läßt sich mit den Konfidenzwerten regulieren, indem der Verzweigungsgrad innerhalb des Baumes reguliert wird. Damit lassen sich die Eigenschaften des Klassifikators zwischen mittlerer Laufzeit und Klassifikationsfehler bei der Ausführung einstellen. Zum anderen wird ein neues Gradientenboosting-Verfahren vorgestellt. Im Gegensatz zum klassischen Gradientenboosting wird die Mustergewichtung direkt mit den Konfidenzwerten vorgenommen, welche die Wahrscheinlichkeitsdichten repräsentieren. Die Gewichtung der Basisklassifikatoren wird nicht mit einer Liniensuche, sondern mit einem expliziten multidimensionalen Optimierungsverfahren durchgeführt, wodurch das Verfahren zu den Stackingverfahren zuzuordnen ist. Bei beiden vorgestellten Verfahren werden die Basisklassifikatoren mit dem kombinatorischen Ressourcenoptimierungsverfahren aus Kapitel 4 trainiert. Beide Verfahren erzielen im Vergleich zu AdaBoost bessere Klassifikationsleistungen bei geringeren mittleren Laufzeiten.

Im Folgenden werden *Basisklassifikatoren* als diejenigen Klassifikatoren definiert, aus denen der Ensembleklassifikator aufgebaut ist. Die Ensembleklassifikatoren selbst sind wiederum Stufenklassifikatoren des Kaskadenklassifikators.

Die Grundvoraussetzung für einen Ensembleklassifikator sind eine Menge an Basisklassifikatoren oder "Modellen", die sich voneinander unterscheiden, indem sie unterschiedliche Eigenschaften der Trainingsmuster hervorheben. Der Ansatz wird oft auch als *Model Averaging* bezeichnet. Die Art und Weise, wie die unterschiedlichen Basisklassifikatoren generiert werden und wie ihre Entscheidungen kombiniert werden, wird durch die Ensembleverfahren beschrieben. Abb. 6.0.1 zeigt einen Ensembleklassifikator H, der aus einzelnen,

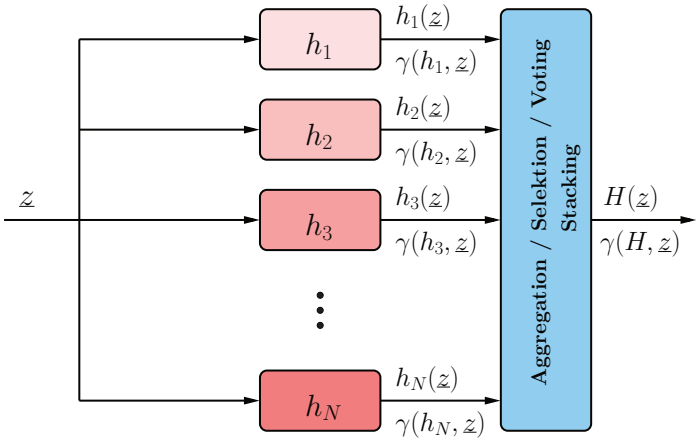


Abbildung 6.0.1: Ensembleklassifikator H, der aus einzelnen, unterschiedlichen Klassifikatoren h_i besteht. Jeder Klassifikator verfügt zusätzlich über einen Konfidenzwert $\gamma(h_i, \underline{z})$ für ein Muster $\underline{z}$. Die Konfidenz und Klassenentscheidung von H wird entweder durch Akkumulation, Selektion, Voting oder Stacking bestimmt.

unterschiedlichen Basisklassifikatoren h_i besteht. Jeder Klassifikator verfügt zusätzlich über einen Konfidenzwert $\gamma(h_i, \underline{z})$. Je nachdem welches Verfahren und welche Klassifikatorstruktur für das Ensembleverfahren verwendet wird, wird die Konfidenz und die Klassenentscheidung von H entweder durch Akkumulation, Selektion, Voting oder Stacking bestimmt.

Die strukturellen und parametrischen Unterschiede in den Basisklassifikatoren können zum einen durch eine bestimmte Auswahl der Merkmale erfolgen, dies geschieht jedoch hauptsächlich durch die Verfahren zur Merkmalsauswahl. Eine andere Möglichkeit, die spezifischen Klassifikatoren zu generieren, ist eine Neugewichtung oder eine Teilauswahl der Muster. Zu diesen Verfahren gehören die *Bagging-* und *Boostingverfahren*. Eine Einführung, sowie Vergleiche der Verfahren auf unterschiedlichen Datensätzen sind in den Arbeiten [56, 10, 58, 48] zu finden. Beim Bagging [15] bzw. Bootstrap Aggregating werden aus einer Mustermenge mehrere unterschiedliche kleinere Mengen (Bootstrap Samples) durch "Ziehen mit Zurücklegen" generiert. Auf jeder dieser Untermengen werden dann Klassifikatoren trainiert und die Klasse mit den meisten Stimmen der Einzelklassifikatoren wird gewählt. Die Klassifikationsleistung kann sich verbessern, falls die Basisklassifikatoren gut klassifizieren und nicht korreliert sind. Bei stabilen Basisklassifikatoren kann

sich die Performance allerdings auch verschlechtern [15]. Die Idee des Boostings wurde 1990 von Schapire vorgestellt [100], wobei die Klassifikationsleistung eines schwachen Lernverfahrens durch Boosting verstärkt wird, indem eine Neugewichtung der Trainingsmuster vorgenommen wird. Beim Boosting werden die Basisklassifikatoren sequentiell generiert, indem gewichtete Muster verwendet werden, die nach jedem Trainingsschritt, abhängig von einem Basisklassifikator oder von dem Ensembleklassifikator, neugewichtet werden. Laut Breiman lässt sich sowohl durch Boosting als auch Bagging nur die Klassifikationsleistung "instabiler" Lernverfahren verbessern [48, 17]. Bauer und Kohavi [10] kommen in einer empirischen Studie zu dem Resultat, dass Boosting im Vergleich zu Bagging auf unterschiedlichen Problemstellungen durchschnittlich bessere Ergebnisse erzielt, dies jedoch nicht immer der Fall sei.

Ein weiteres Verfahren, das im Bezug auf den Ensembleklassifikator erwähnt werden muss, ist das *Stacking*, das auch mit Stacked Generalization [123] benannt wird. Beim Stacking werden zuerst unterschiedliche Basisalgorithmen mit den vorhandenen Daten trainiert und dann werden die Ausgaben der Algorithmen als Eingabe für einen Trainingsalgorithmus verwendet, der die Schätzungen der unterschiedlichen Basisalgorithmen zu einer kombinierten Ausgabe bzw. Klassifikation zusammenführt. Dadurch, dass prinzipiell beliebige Algorithmen zur Kombination der Basisalgorithmen verwendet werden können, ist es theoretisch möglich, alle beliebigen Ensembleverfahren durch das Stacking mit geeigneten Basis- und Ensemblealgorithmen zu repräsentieren. In der Praxis wird jedoch häufig eine einschichtige logistische Regression als Ensemblealgorithmus verwendet. Das Stacking erzielt bessere Klassifikationsleistungen als jeder einzelne Basisalgorithmus [123] und wurde zum Beispiel auf überwachte Lernaufgaben [16] mittels Regression, sowie auf unüberwachte Lernaufgaben [105] eingesetzt. Clarke [25] beschreibt, dass mit Stacking deutlich bessere Klassifikationsleistungen erzielt werden, als mit Bayes'scher Modellmittelung (Bayesian model-averaging). Abb. 6.0.2 zeigt die Unterschiede beim Training der Basisklassifikatoren. Abb. 6.0.2(a) beschreibt das Bagging-Verfahren, bei dem Boostrap Mustermengen gebildet werden, die durch Ziehen mit Zurücklegen aus einer Merkmalsmenge $\mathcal{D}$ generiert werden. Die einzelnen Boostrap Mengen sind somit kleiner und die Basisklassifikatoren haben keinen Einfluss auf die Wahl der Muster. Abb. 6.0.2(b) zeigt ein sequentielles Training der Basisklassifikatoren. Der jeweils zuletzt trainierte Klassifikator nimmt eine Neugewichtung der Beispiele vor und generiert somit einen veränderten Trainingsdatensatz, mit

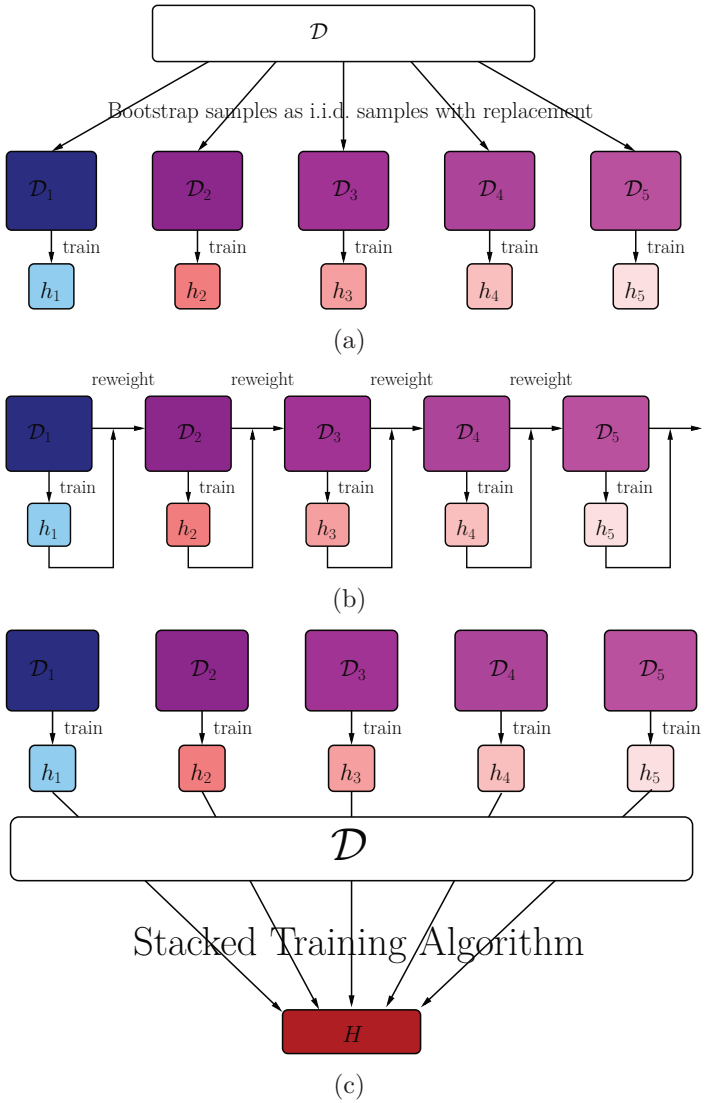


Abbildung 6.0.2: (a): Beim Bagging-Verfahren werden aus einer Mustermenge durch Ziehen mit Zurücklegen sogenannte Bootstrap Samples für das Training der Basisklassifikatoren generiert. (b): Iteratives Training mit Boosting durch Neugewichtung der Beispiele anhand des zuletzt trainierten Klassifikators. (c): Training des Ensembleklassifikators mit Stacking. Der Ensembleklassifikator H wird mit den Ausgaben der Basisklassifikatoren traininert.

dem der nächste Klassifikator trainiert wird. Dieses Schema wird z.B. bei AdaBoost, Real AdaBoost, Gentle AdaBoost und LogitBoost [58] verwendet. Abb. 6.0.2(c) stellt das Stacking dar. Der Ensembleklassifikator H wird mit einem Trainingsverfahren trainiert, das die Ausgaben der Basisklassifikatoren h_i als Eingabe verwendet. Da sich bei stabilen Basisklassifikatoren die Klassifikationsperformance beim Bagging verschlechtern kann [15] und aufgrund der Tatsache, dass in dieser Arbeit die Algorithmen auf die jeweiligen Stufenprobleme angepasst sein müssen und im voraus nicht bekannt ist, wie viele unterschiedliche Bootstrap Samples benötigt werden um den Klassifikationsfehler zu reduzieren, wird das Bagging-Verfahren in dieser Arbeit nicht weiter verfolgt. In Abschnitt 6.1 wird auf die unterschiedlichen Boostingverfahren in der Literatur eingegangen und das AdaBoost-Verfahren [38] beschrieben. Außer dem AdaBoost-Verfahren wurde das sogenannte Gradientenboosting [43, 42] verwendet, auf die kaskadierte Klassifikatorstruktur und Basisiklassifikatoren angepasst und weiterentwickelt, indem ein konfidenzbasiertes Gewichtungsschema und eine multidimensionale Optimierung der Basisklassifikatoren vorgenommen wird. Bei dem hier entwickelten konfidenzbasierten Gradientenboosting werden die Lernbeispiele mit der Konfidenzfunktion des Ensembleklassifikators absolut gewichtet und sind mit beliebigen Basisklassifikatoren anwendbar. Ein weiterer Vorteil ist, dass sowohl die Basisklassifikatoren als auch der Ensembleklassifikator mit den in Kapitel 3 beschriebenen Methoden auf ein definiertes Signifikanzniveau eingestellt werden können, indem die Falsch-Negativ-Fehler getrennt von den Falsch-Positiv-Fehlern betrachtet werden. Eine ausführliche Beschreibung des Verfahrens und die grundlegenden Unterschiede zu AdaBoost sind in Abschnitt 6.2 beschrieben. Zusätzlich wurde ein neues Verfahren entwickelt und ebenfalls auf die Kaskadenstruktur angepasst, das eine adaptive Unterteilung der Mustermenge, abhängig von der Komplexität des Teilproblems, anhand eines Baumes vornimmt, und somit eine effektive, ressourcenoptimierte Klassifikation ermöglicht. Außerdem bietet diese Klassifikatorstruktur einen Mechanismus zum Anpassen der Klassifikatoreigenschaften zwischen Detektionsleistung und mittlerer Laufzeit bei der Anwendungsphase des Klassifikators. Das Verfahren wird in Abschnitt 6.3 beschrieben.

Algorithmus 6.1 AdaBoost [40]

1. **Gegeben:** $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, wobei $x_i \in X$ und $y_i \in \{-1, +1\}$
2. Initialisiere $D_1(i) = 1/m$.
3. **Für** $t = 1, \ldots, T$:
   (a) Trainiere die schwachen Klassifikatoren mit der Verteilung D_t
   (b) Bestimme den besten schwachen Klassifikator $h_t : X \to \{-1, +1\}$ mit dem geringsten Fehler $\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$, also die mit D_t gewichteten Fehlklassifikationen
   (c) Wähle $\alpha_t = \ln(\frac{1-\epsilon_t}{\epsilon_t})$
   (d) Update:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{falls } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{falls } h_t(x_i) \neq y_i \end{cases}$$
$$= \frac{D_t(i) \exp(\alpha_t y_i h_t(x_i))}{Z_t}$$

   wobei Z_t eine Normierungskonstante ist
4. **Ausgabe:** Ensembleklassifikator $H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$

6.1 AdaBoost

In diesem Abschnitt wird das AdaBoost-Verfahren [38] näher erklärt, da es als Referenzverfahren in dieser Arbeit dient. Außerdem werden einige der bestehenden unterschiedlichen Boostingverfahren aufgezählt und die wesentlichen Unterschiede zu AdaBoost erklärt. In Alg. (6.1) ist das AdaBoost-Verfahren beschrieben, wobei in jeder Iteration t ein Trainingsalgorithmus einen schwachen Klassifikator als Basisklassifikator trainiert. Eine der Hauptideen dabei ist, dass für jedes Trainingsbeispiel ein vom Klassifikationsfehler abhängiges Gewicht mitgeführt wird. Der Fehler des schwachen Klassifikators wird anhand der Fehlklassifikationen auf den gewichteten Beispielen bestimmt (Schritt 3b) und ein logarithmischer, inverser Gewichtungsfaktor α_t berechnet. Anhand des schwachen Klassifikators werden dann in Schritt 3d die

Gewichte adaptiert, indem korrekt klassifizierte Beispiele heruntergewichtet und falsch klassifizierte Beispiele hochgewichtet werden. Korrekt klassifizierte Beispiele mit geringem Fehler werden dabei stark heruntergewichtet und diejenigen mit größerem Fehler schwächer heruntergewichtet. Andererseits werden falsch klassifizierte Beispiele mit geringem Fehler stark hochgewichtet, diejenigen mit großem Fehler dahingegen schwächer hochgewichtet. Die Iteration wird abgebrochen, falls eine maximale Anzahl T erreicht ist oder, falls der Fehler des schwachen Klassifikators $\epsilon_t > \frac{1}{2}$. Die Entscheidungsfunktion des Ensembleklassifikators berechnet sich durch Mehrheitsentscheidung der gewichteten Basisklassifikatoren.

Ausgehend von dem AdaBoost-Verfahren wurde eine Vielzahl an unterschiedlichen Boosting-Verfahren entwickelt. Die Unterschiede sind meistens in der Art der Basisklassifikatoren und der Neugewichtung zu finden. Bei Real AdaBoost [41] werden mit der gewichteten Lernstichprobe die Zuordnungswahrscheinlichkeiten mit dem Basisklassifikator bestimmt, woraus ein reellwertiger Klassifizierer entwickelt wird, mit welchem die Gewichte angepasst werden. Bei Gentle AdaBoost [41] wird der reellwertige Klassifizierer in Schritt 3a als Regressionsfunktion mittels der Methode der kleinsten Quadrate trainiert. Beim WaldBoost-Verfahren [106] wird ein Ensembleklassifikator berechnet, indem mit dem Sequential Probability Ratio Test (SPRT) für jeden Basisklassifikator zwei Schwellwerte berechnet werden, mit denen beim Training Beispiele aussortiert werden und die bei der Ausführung zur Klassifikation des Ensembleklassifikators genutzt werden. Die Wahrscheinlichkeiten, die für den SPRT benötigt werden, werden mit dem AdaBoost-Verfahren geschätzt, das in das Training integriert ist. Die relevanten Funktionen werden ebenfalls mit dem AdaBoost-Verfahren ermittelt. In der Literatur existieren noch weitere Boostingverfahren, die allerdings im Rahmen dieser Arbeit nicht alle aufgezählt werden können.

6.2 Konfidenzbasiertes Gradientenboosting mit multidimensionaler Optimierung

Gradientenboosting wurde erstmals von Friedman [43] vorgestellt und ist ein Verfahren, das sowohl als Regressionsverfahren als auch als Klassifikationsverfahren eingesetzt werden kann. In einer weiteren Arbeit [42] wird der Algorithmus um eine stochastische Komponente erweitert, wodurch sich die

Klassifikationsleistung steigern lässt. Das Verfahren basiert, wie andere Gradientenverfahren, auf einer zu minimierenden Zielfunktion L. Im Bezug auf ein Ensembleklassifikator H lautet die Zielfunktion:

$$H_{\text{opt}} = \arg\min E_{\underline{z}}[E_y(L(y, H(\underline{z})))|\underline{z}] \tag{6.2.1}$$

wobei E dem erwarteten Schätzfehler [56] entspricht. Das Ensemble wird mit einem Klassifikator initiiert, und in einer Schleife werden dann iterativ die sogenannten *Pseudo-Residuen* mittels partieller Ableitung auf dem aktuellen Ensembleklassifikator berechnet:

$$r_{i,t} = -\left[\frac{\partial L(y_i, H(\underline{z}_i))}{\partial H(\underline{z}_i)}\right]_{H(\underline{z})=H_{t-1}(\underline{z})} \tag{6.2.2}$$

wobei t der Iteration entspricht und $(\underline{z}_i, y_i) \in \mathcal{D}$ die Muster mit entsprechender Klassenzugehörigkeit sind. Die Basisklassifikatoren $h_t(\underline{z})$ werden dann auf den Pseudo-Residuen trainiert und die Gewichte w_t werden mittels einem eindimensionalen Optimierungsproblem bzw. einer Liniensuche bestimmt. Der Ensembleklassifikator wird am Ende jeder Iteration aktualisiert:

$$H_m(\underline{z}) = H_{t-1}(\underline{z}) \oplus w_t h_t(\underline{z}) \tag{6.2.3}$$

Im Folgenden ist das in dieser Arbeit entwickelte konfidenzbasierte Gradientenboosting mit multidimensionaler Optimierung beschrieben. Im gegensatz zum einfachen Gradientenboosting, bei dem eine eindimensionale Optimierung für den Ensemleklassifikator verwendet wird, wird hier eine multidimensionale Optimierung verwendet. Somit zählt das hier beschriebene konfidenzbasierte, erweiterte Gradientenboosting zur Klasse der *Stackingalgorithmen*. Um das Gradientenboosting auf die hier verwendeten Klassifikatoren und Entscheidungsfunktionen anzuwenden, wird die reellwertige Klassifikationsfunktion des Ensembleklassifikators mit der normierten Konfidenz als Klassifikationsfunktion ohne Betrag gesetzt:

$$H_T(\underline{z}) = \gamma_{\text{gb}}(\langle \underline{w}, \underline{h}(\underline{z})\rangle_T) = \frac{P(\langle \underline{w}, \underline{h}(\underline{z})\rangle_T \,|\, \omega_1) - P(\langle \underline{w}, \underline{h}(\underline{z})\rangle_T \,|\, \omega_0)}{P(\langle \underline{w}, \underline{h}(\underline{z})\rangle_T \,|\, \omega_1) + P(\langle \underline{w}, \underline{h}(\underline{z})\rangle_T \,|\, \omega_0)} \tag{6.2.4}$$

wobei $\langle \underline{w}, \underline{h}(\underline{z})\rangle_T = \sum_{t=1}^{T} w_t h_t(\underline{z})$ die Projektion der Basisklassifikatoren mit den Gewichten w_t ist. Da die Konfidenz zur Berechnung der Pseudo-Residuen innerhalb einer Kaskadenstufe verwendet wird, müssen keine Apriori-Wahrscheinlichkeiten für die Entscheidungsfunktionen der Basisklassifikatoren berücksichtigt werden. Die Wahrscheinlichkeitsdichten lassen sich

mit den Frequenztabellen approximieren. Durch die Normierung ist der reelle Wertebereich zwischen $H(\underline{z}) \in [-1, \ldots, +1]$ und die Zielwerte sind $y(\underline{z} \in \mathcal{D}_1) = +1$ und $y(\underline{z} \in \mathcal{D}_0) = -1$. Die Zielfunktion als quadratische Fehlerfunktion lautet dann:

$$L(y, H(\underline{z})) = \frac{1}{2}(y - H(\underline{z}))^2 \tag{6.2.5}$$

Und die Ableitung berechnet sich mit:

$$-\frac{\partial L(y, H(\underline{z}))}{\partial H(\underline{z})} = y - H(\underline{z}) \tag{6.2.6}$$

Da die Pseudo-Residuen als Gewichte der Trainingsmuster verwendet werden, wird Gl. (6.2.6) mit dem Absolutbetrag verwendet:

$$r_{i,t} = |y_i - H_{t-1}(\underline{z}_i)| \tag{6.2.7}$$

wobei H_{t-1} mit Gl. (6.2.4) berechnet wird. Somit ist das Gewicht der Muster proportional zum Abstand der Approximation des Ensembleklassifikators zum Zielwert der Muster. Maximale Gewichte werden von fehlklassifizierten Mustern angenommen. Der Unterschied zu den anderen Boostingverfahren ist, dass für die Neugewichtung der Beispiele alle bisher trainierten Basisklassifikatoren beteiligt sind. Die Gewichte werden dadurch besser an den Ensembleklassifikator angepasst, da sie abhängig von diesem berechnet werden. Außerdem wird zur Neugewichtung nicht nur eine Konstante für alle Muster verwendet, die entweder das Gewicht nach oben oder unten aktualisiert (AdaBoost-Verfahren), sondern die Gewichte (bzw. Pseudo-Residuen) werden für jedes Muster $\underline{z}$ direkt berechnet und sind von der Modellierung der Wahrscheinlichkeiten in Gl. (6.2.4) abhängig. Abb. 6.2.1 zeigt das sequentielle Trainingsverfahren. Die unterschiedlichen Farben stellen die unterschiedlich gewichteten Trainingsmuster in den Zyklen dar, die anhand der Konfidenz des Ensembleklassifikators bestimmt werden. Werden zur Modellierung der Konfidenzen Frequenztabellen eingesetzt, ist die Anzahl unterschiedlicher Konfidenzwerte abhängig von der Anzahl der Einträge.

Im Gegensatz zu dem in der Literatur beschriebenen Gradientenboosting werden die Gewichte des Ensembleklassifikators $\underline{w}$ in Gl. (6.2.4) nicht mit einem eindimensionalen Optimierungsverfahren, sondern über eine mehrdimensionale Minimierung des Quadratischen Fehlers mit den bereits beschriebenen Verfahren bestimmt, was in der Abbildung 6.2.1 unten links dargestellt

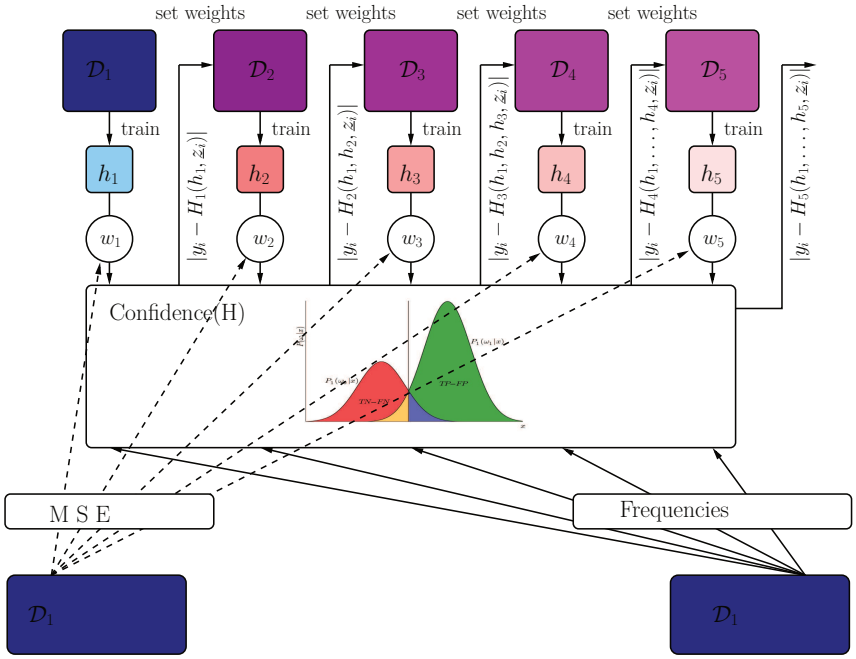


Abbildung 6.2.1: Iteratives Training beim konfidenzbasierten Gradientenboosting. Die Gewichte werden hier anhand der Konfidenz (Gl. (6.2.4)) direkt berechnet und hängen von allen bisher trainierten Basisklassifikatoren ab. Jeder Basisklassifikator wird linear gewichtet, wobei die Gewichte über ein Verfahren zur Minimierung des Quadratischen Fehlers mit gleichgewichteten Mustern bestimmt werden.

ist. Die Optimierung wird dabei auf gleichgewichteten Mustern ausgeführt. Mit diesen Mustern werden auch die Wahrscheinlichkeiten für die Entscheidungsfunktion des Ensembleklassifikators geschätzt (siehe Abb. 6.2.1 unten rechts). Nach jedem neuen Basisklassifikator wird der Schritt zur Optimierung der Gewichte und zum Schätzen der Wahrscheinlichkeiten durchgeführt, und somit ändern sich mit jedem neuen Basisklassifikator die Gewichte und die Entscheidungsfunktion bzw. die Konfidenzen des Ensembleklassifikators. Da der Ensembleklassifikator als Stufenklassifikator innerhalb der Kaskade eingesetzt wird, wird die Bayes-Entscheidungsfunktion mit den in dieser Stufe auftretenden A-priori-Klassenwahrscheinlichkeiten berechnet:

$$H_{\text{res}}(\underline{z}) = \frac{P(\langle \underline{w}, \underline{h}(\underline{z}) \rangle \,|\, \omega_1) P(\omega_1) - P(\langle \underline{w}, \underline{h}(\underline{z}) \rangle \,|\, \omega_0) P(\omega_0)}{P(\langle \underline{w}, \underline{h}(\underline{z}) \rangle \,|\, \omega_1) P(\omega_1) + P(\langle \underline{w}, \underline{h}(\underline{z}) \rangle \,|\, \omega_0) P(\omega_0)} \tag{6.2.8}$$

Die Entscheidungsfunktionen der Basisklassifikatoren werden mit identischen A-priori-Wahrscheinlichkeiten berechnet. Das erweiterte konfidenzbasierte Gradientenboosting ist in Alg. (6.2) dargestellt. Durch das Training des Ensembleklassifikators in Schritt 3e auf der gleichgewichteten Trainingsmenge wird vermieden, dass verrauschte Beispiele ein höheres Gewicht erhalten, wie es bei den Basisklassifikatoren der Fall sein kann.

Algorithmus 6.2 Konfidenzbasiertes Gradientenboosting

1. **Gegeben:**
   (a) Trainingsmuster $\mathcal{D}$
   (b) Fehlerschwelle $e_{\mathrm{gb}}(\omega_0)$, Größe der Untermenge $\beta \leq 1$
   (c) $t_{\max}$, max. Anzahl an Basisklassifikatoren
2. Initialisiere H_0 als konstante Funktion.
3. **Iteriere** $t = 1, \ldots, t_{\max}$:
   (a) Bestimme die Gewichte $r_{t,i}$ der Muster $(r_{t,i}, \underline{z}_i, y_i) \in \mathcal{D}_t$ anhand der Konfidenz des Ensembles H_{t-1} durch Gl.(6.2.7) und normiere sie.
   (b) Bestimme eine zufällige Teilmenge $\mathcal{D}'_t$ aus $\mathcal{D}_t$ im Verhältnis $\beta \leq 1$.
   (c) Trainiere Basisklassifikator h_t mit $\mathcal{D}'_t$.
   (d) Aktualisiere $H_t = H_{t-1} \oplus h_t$.
   (e) Trainiere Ensembleklassifikator H_t, indem die Gewichte w_i mit $1 \leq i \leq t$ durch die MSE Optimierung und die Entscheidungsfunktion γ mit A-priori-Wahrscheinlichkeiten mit den gleichgewichteten Mustern $\mathcal{D}_1$ bestimmt werden.
      i. Falls $P(H_t(\underline{z}) = \omega_1 | \omega_0) \leq e_{\mathrm{gb}}(\omega_0)$: stop.
      ii. Sonst gehe zum nächsten Schleifendurchgang.
4. **Ausgabe:**
   $H_{\mathrm{res}}(\underline{z}) = \frac{P(\langle \underline{w}, \underline{h}(\underline{z}) \rangle \,|\, \omega_1) P(\omega_1) - P(\langle \underline{w}, \underline{h}(\underline{z}) \rangle \,|\, \omega_0) P(\omega_0)}{P(\langle \underline{w}, \underline{h}(\underline{z}) \rangle \,|\, \omega_1) P(\omega_1) + P(\langle \underline{w}, \underline{h}(\underline{z}) \rangle \,|\, \omega_0) P(\omega_0)}$,
   wobei $\underline{w}$ der Gewichtsvektor und $\underline{h}(\underline{z})$ der Vektor aus den Konfidenzen der Basisklassifikatoren ist.

Die Beschränkung auf eine Untermenge beim Training der Basisklassifikatoren in Schritt 3b bewirkt, ähnlich wie beim Bagging, eine höhere Diversität der Basisklassifikatoren, da sich die Trainingsmengen stärker unterscheiden.

6.2.1 Quadratisches, konfidenzbasiertes Gradientenboosting

In dem bisher beschriebenen Verfahren wurde für jeden Basisklassifikator ein Gewicht mit dem MSE-Verfahren bestimmt (siehe Abb. 6.2.1). Damit tragen die unterschiedlichen Basisklassifikatoren unterschiedlich stark zum Gesamtergebnis bei. Um den Fehler des Ensembleklassifikators mit geringem Rechenaufwand weiter zu reduzieren liegt es nahe, den linearen Merkmalsvektor, bestehend aus den Konfidenzen der Basisklassifikatoren, mit gemischt quadratischen Termen zu ergänzen. Da die Konfidenzberechnungen der Basisklassifikatoren, die wesentlich mehr Rechenoperationen benötigen, weiterhin nur linear in ihrer Anzahl einfließen, indem ihre Konfidenzen temporär gespeichert werden, kann die quadratische Erweiterung sehr effizient berechnet werden. Die Dimension des Merkmalsvektors bei T Basisiklassifikatoren ist dann $T + T(T+1)/2$:

$$\underline{x} = (h_1, \ldots, h_T, h_1h_1, h_1h_2, \ldots, h_1h_T, h_2h_3, \ldots, h_2h_T, \ldots, h_Th_T)^T \quad (6.2.9)$$

wobei hier $h_i(\underline{z})$ der Konfidenz des i. Basisiklassifikators entspricht. Die gemischtquadratischen Terme sind identisch mit den Einträgen der Kovarianzmatrix der Konfidenzen. Durch die gemischtquadratischen Terme in Gl. (6.2.9) werden zusätzlich zu den Konfidenzen der Basisklassifikatoren die statistischen Abhängigkeiten zwischen den einzelnen Klassifikatoren berücksichtigt. Die Gefahr des Übertrainierens des Ensembleklassifikators kann durch eine an den Datensatz angepasste Anzahl T minimiert werden. So ist zum Beispiel die Dimension des Merkmalsvektors bei $T = 64$ bereits 2144.

6.3 Ressourcenoptimierte, konfidenzbasierte Unterteilungsbäume

Baumartige Klassifikatorstrukturen finden vielseitige Anwendung bei Klassifikationsaufgaben. Diese sogenannten Entscheidungsbäume bestehen aus einem Wurzelknoten und beliebig vielen inneren Knoten, sowie mindestens zwei

Blättern. Die inneren Knoten repräsentieren logische Regeln, indem typische Merkmale getestet werden, wobei in den Blättern eine Klassenzuordnung erfolgt. Der Begriff CART [56] (Classification And Regression Trees) beinhaltet sowohl Klassifikations- als auch Regressionsbäume und wurde von Breiman et al. eingeführt [19]. Die bekanntesten Algorithmen zur Erzeugung von Entscheidungsbäumen, die zur Familie der CARTs gehören, sind der ID3 und C4.5 Algorithmus [97, 98]. Das Ziel beim Erstellen dieser Bäume ist die Unterteilung des Merkmalsraumes in Regionen, die eine hohe Homogenität der Datenpunkte enthalten, indem Merkmale aus nur einer Klasse enthalten sind. Als Unterteilungskriterium wird deshalb das Merkmal gewählt, das den höchsten Informationsgewinn erzielt, der über die maximale Reduktion der Entropie berechnet wird und somit eine Unterteilung in möglichst homogene Bereiche ermöglicht. Es wird zwischen univariaten Entscheidungsbäumen, bei denen ein Merkmal in jedem Knoten getestet wird, und multivariaten Entscheidungsbäumen [21], bei denen eine gewichtete Kombination aus mehreren Merkmalen verwendet wird, unterschieden. Laut Bordley [21] kann die Klassifikationsleistung durch Verwendung von multivariaten Tests in den Knoten verbessert werden.

Eine weitere Verbesserung der Klassifikationsleistung von Entscheidungsbäumen wird durch die sogenannten Entscheidungswälder (Decision Forests) ermöglicht, indem über die Einzelentscheidungen der Bäume gemittelt wird. Durch Randomisierung der Entscheidungswälder bezüglich der Trainingsmuster und der Merkmale sind die sogenannten Random Forests entstanden [59, 18]. Eine schnelle Klassifikation kann zwar durch Verwendung eines Multiprozessorsystems erreicht werden, indem auf jedem Prozessor ein Baum ausgewertet wird, der Gesamtenergiebedarf wird dadurch jedoch nicht reduziert. Die benötigten Operationen für eine Klassifikation steigen mit der Zahl der Bäume.

Die Erzeugung von Entscheidungswäldern mit Bäumen der Tiefe eins, den sogenannten Decision Stumps, die auf die Kaskadenstruktur abgestimmt ist, wurde in Abschnitt 6.2 durch das Gradientenboosting mit Randomisierung aufgegriffen. Der auf diese Weise trainierte Ensembleklassifikator ist ein Random Forest, der aus Bäumen der Tiefe eins (Stümpfen, vgl. englisch "stumps") besteht, die aus einer randomisierten Untermenge an Mustern trainiert wurden. Der Vorteil ist, dass bei einfacheren Problemen in niedrigen Stufen nur wenige Basisklassifikatoren verwendet werden und somit der Energieaufwand reduziert werden kann.

Die baumartigen Klassifikatoren sollen als Stufenklassifikatoren verwen-

det werden, deshalb werden im Folgenden die Anforderungen an den Algorithmus zur Generierung der baumartigen Klassifikatoren formuliert. Es soll, wie bei der sequentiellen Merkmalsauswahl, eine wahrscheinlichkeitsbasierte Steuerung der Komplexität der Bäume erfolgen um nur bei schwierigen Teilproblemen komplexe Bäume zu erzeugen und somit die mittlere Laufzeit niedrig zu halten. Außerdem sollen die bereits eingeführten Mechanismen zur Konfidenzberechnung und A-priori-gewichteten Bayes'schen Entscheidungsfunktionen sowie die Ressourcenoptimierung der Einzelklassifikatoren integriert werden. Der Algorithmus soll als Ensembleverfahren entworfen werden, das mit beliebigen Basisklassifikatoren verwendet werden kann. Somit resultieren mit den bereits eingeführten Trainingsverfahren der merkmalsbasierten Klassifikatoren *Hybridverfahren* ohne zusätzlichen Aufwand. Um der Echtzeitanforderung des Klassifikators gerecht zu werden soll außerdem die Möglichkeit der Regulierung der Klassifikatoreigenschaften zwischen mittlerer Laufzeit und Klassifikationsgüte bei der Ausführung bestehen. Somit kann im mobilen Einsatz je nach verfügbaren Rechenressourcen der Klassifikator zwischen energiesparend und effizient bezüglich den mittleren Laufzeiten und effizient bezüglich der Klassifikationsleistung adaptiv angepasst werden. Da die Basisklassifikatoren bei wenig Daten instabil werden, soll ein von der Trainingsmenge abhängiges Abbruchkriterium verwendet werden.

Da die Standardalgorithmen diese Anforderungen nicht erfüllen wird ein neuer Algorithmus entworfen, der im weiteren Verlauf beschrieben ist und obige Anforderungen erfüllt.

Abb. 6.3.1 zeigt den grundlegenden Aufbau des Baumes. Die inneren Knoten sind Splitknoten, die durch Klassifikatoren realisiert werden und die positiven Beispiele bestmöglich trennen. Sie zeigen entweder auf weitere Splitknoten oder auf Blattknoten. Beim Training werden die Trainingsbeispiele ebenfalls mit den Splitklassifikatoren aufgespalten, indem sie an unterschiedliche Unterklassifikatoren weitergereicht werden. Die Blattklassifikatoren treffen, im Gegensatz zu den klassischen Entscheidungsbäumen, eine Klassenentscheidung für die Objekt- oder Hintergrundklasse. Somit können als Blattknoten beliebige Klassifikatoren verwendet und insbesondere die Bayes'schen Entscheidungsfunktionen verwendet werden. Die Klassifikatoren "kommunizieren" mithilfe der Wahrscheinlichkeiten und somit auch der Konfidenzen untereinander. Bei niedrigen Konfidenzen in den Splitknoten können deshalb vergabelte Verweise auf mehrere Kinderknoten erfolgen und somit den Klassifikationsfehler reduzieren. Je nachdem mit welcher Regel gesplittet wird, kann damit zwischen einem einzelnen Pfad und mehreren Pfaden

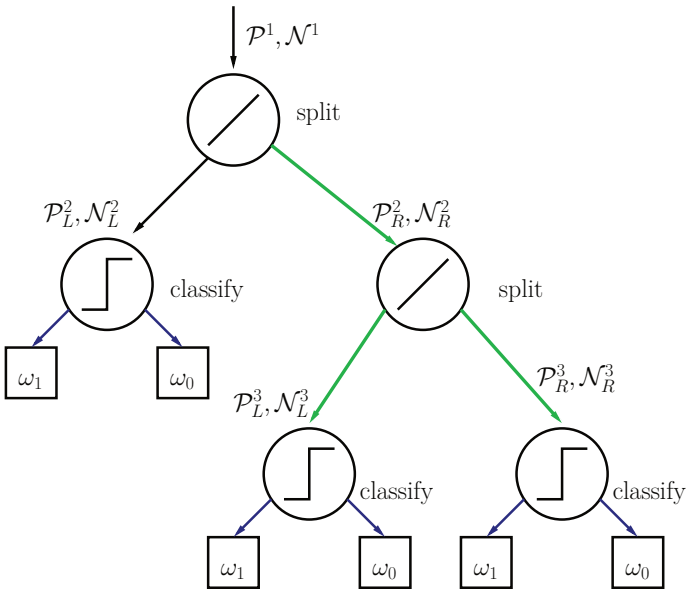


Abbildung 6.3.1: Aufbau der ressourcenoptimierten Unterteilungsbäume mittels konfidenzbasierten Entscheidungspfaden. Innere Knoten des Baumes bilden die Split-Klassifikatoren, die entweder auf den linken, den rechten oder beide Kinderknoten verweisen und mit den Beispielen $\mathcal{P}^i_{\{R,L\}}$ (Objekte) und $\mathcal{N}^i_{\{R,L\}}$ (Hintergrund) trainiert werden. Die Blatt-Klassifikatoren treffen die Entscheidung. Für unsichere Entscheidungen der Split-Klassifikatoren werden die Pfade aufgeteilt und die Entscheidung der Blatt-Klassifikatoren wird fusioniert.

gewählt werden. Falls mehrere Blattklassifikatoren an der Entscheidung beteiligt sind, muss die Entscheidung fusioniert werden. Mit diesem Mechanismus lässt sich, abhängig vom Grad der Verzweigung der Pfade, zwischen der Klassifikationsleistung und mittleren Laufzeit optimieren.

In Alg. (6.3) ist der Trainingsalgorithmus zur Erzeugung des Baumes $\mathcal{T}$ in einer spezifischen Stufe beschrieben. Das Training des Blattklassifikators in Schritt 4a kann mit einem beliebigen Trainingsverfahren erfolgen. In Schritt 4b wird der Klassifikationsfehler überprüft. Bei geringem Fehler wird der Klassifikator als Blattklassifikator verwendet. Die Indizes der Knoten im Baum werden dabei mitgeführt, so dass die Position innerhalb des Baumes, an der der Klassifikator hinzugefügt wird, mit den repräsentiven Mustermengen und Splitpfaden konsistent ist. Somit entstehen in unteren Stufen der

Algorithmus 6.3 Training ressourcenoptimierter Unterteilungsbäume

1. **Gegeben:**
   (a) Für das Teilproblem repräsentative Trainingsmenge $\mathcal{D}^0$
   (b) Min. Anzahl an Beispielen m
   (c) Fehlerschranke $e_{\mathrm{UB}}(\omega_0)$
2. Trainingsmenge $\mathcal{D}^0 = \{\mathcal{D}^0_{\omega_1}, \mathcal{D}^0_{\omega_0}\}$ mit Objekt- und Hintergrundbeispielen.
3. Initialisiere Baumklassifikator $\mathcal{T} = \emptyset$, initialisiere Stack $\mathcal{S} = \{(\mathcal{D}^0, k_0)\}$, wobei k_0 der Knotenindex der Wurzel ist.
4. **Solange** der Stack Elemente enthält $(\mathcal{D}^t, k^t) \in \mathcal{S}$:
   (a) Trainiere Blattklassifikator $h_{\mathrm{opt},t}$ mit $\mathcal{D}^t$.
   (b) **Falls** $P(h_{\mathrm{opt},t}(\underline{z}) = \omega_1|\omega_0) \leq e_{\mathrm{UB}}(\omega_0)$ **oder** $|\mathcal{D}| < m$:
   Füge $h_{\mathrm{opt},t}$ als Blattklassifikator dem Baum $\mathcal{T}$ am Knoten k^t hinzu.
   (c) **Sonst**:
      i. Trainiere optimalen Splitklassifikator $h_{\mathrm{split},t}$ auf $\mathcal{D}^t_{\omega_1}$ mittels K-Means für jede Merkmalskombination $\mathcal{F}_{\Delta r}$ als unüberwachtes Trainingsverfahren und einem Splitkriterium s.
      ii. Füge $h_{\mathrm{split},t}$ als innerer Knoten dem Baum $\mathcal{T}$ am Knoten k^t hinzu.
      iii. Unterteile die Beispiele $\mathcal{D}^t$ mittels $h_{\mathrm{split},t}$ durch Klassifikation in zwei disjunkte Mengen $\mathcal{D}^t = (\mathcal{D}^t_{\mathrm{L}}, \mathcal{D}^t_{\mathrm{R}})$.
      iv. Berechne die neuen Knotenindizes $k^t_{\mathrm{L}}, k^t_{\mathrm{R}}$.
      v. $\mathcal{S} \leftarrow \mathcal{S} \cup (\mathcal{D}^t_{\mathrm{L}}, k^t_{\mathrm{L}}) \cup (\mathcal{D}^t_{\mathrm{R}}, k^t_{\mathrm{R}})$
5. Iteriere über alle Blattklassifikatoren und berechne die Entscheidungsfunktionen mit allen Beispielen $\mathcal{D}^0$.
6. **Ausgabe**: Unterteilungsbaum $\mathcal{T}$.

Kaskade einfache Bäume oder nur Einzelklassifikatoren. Ist die Menge an Beispielen zu gering, wird ebenfalls abgebrochen um ein Übertrainieren der Klassifikatoren zu vermeiden. Außerdem wird eine maximal erlaubte Merkmalsdimension, abhängig von der Anzahl der Trainingsbeispiele verwendet, wodurch ebenfalls Overfitting-Effekte reduziert werden. In Schritt 4(c)i wird der optimale Splitklassifikator bestimmt, indem für jede Merkmalskombination eine unüberwachtes K-Means-Verfahren [35] mit $K = 2$ auf den positiven Beispielen $\mathcal{D}^t_{\omega_1}$ vollzogen wird. Jeder Merkmalsvektor erhält eine Klassenzuweisung. Somit können mit der FLDA die Gewichte des Splitklassifikators berechnet werden. Der berechnete Richtungsvektor der FLDA entspricht einer Trennung mit einer beliebig orientierten Trennebene und somit einem multivariaten Test.

Algorithmus 6.4 K-Means

1. **Gegeben:** Menge an Merkmalsvektoren ohne Klassenzuordnung
2. Initialisiere Mittelwerte innerhalb der Vektoren.
3. **Solange** die Anzahl der veränderten Klassenzugehörigkeiten über einer Schwelle liegt:
   (a) Weise jedem Vektor die Klasse des Klassenmittelwertes der Vektoren mit minimalem Abstand zu.
   (b) Berechne neue Mittelwerte für die zugewiesenen Klassen der Vektoren.
4. **Ergebnis:** Merkmalsvektoren mit Klassenzuordnungen und deren Mittelwerte.

Alg. (6.4) zeigt die prinzipielle Funktionsweise des K-Means-Algorithmus. Für jede Merkmalskombination werden damit die Klassenzuordnungen berechnet, mit denen dann mit dem FLDA-Verfahren (Alg. (3.2)) die Split-Ebene berechnet werden kann. Die Entscheidungsfunktion besteht aus einer Frequenztabelle, die ebenfalls aus den zugewiesenen Beispielen bestimmt wird. Da keine Klassenentscheidung bei den Splitklassifikatoren getroffen wird, werden die A-priori-Wahrscheinlichkeiten gleichgesetzt. Somit können aus der Frequenztabelle Konfidenzen und Regeln für die Verzweigung der Pfade bestimmt werden. Um aus den unterschiedlichen Merkmalskombinationen

zum optimalen Splitklassifikator zu gelangen wird ein Optimierungskriterium eingeführt:

$$f_{\text{split}}(h) = (1 + \frac{||\mathcal{D}^t_{\omega_1}| - |\mathcal{D}^t_{\omega_0}||}{\max(|\mathcal{D}^t_{\omega_1}|, |\mathcal{D}^t_{\omega_0}|)})(1 + e(\omega_1) + e(\omega_0)) \tag{6.3.1}$$

Es werden damit Merkmale mit einer ausgewogenen Klassenverteilung, also einer hohen Entropie, und geringen Klassifikationsfehlern auf den durch K-Means zugewiesenen Klassen bevorzugt. Je kleiner die Klassifikationsfehler sind, desto sauberer ist die Trennung der Klassen. Um den optimalen Splitklassifikator zu ermitteln wird das Minimum aus allen trainierten Klassifikatoren $h \in \mathcal{H}$ gesucht:

$$h_{\text{split}} = \underset{f_{\text{split}}(h)}{\arg\min}\ \mathcal{H} \tag{6.3.2}$$

Mit dem Splitklassifikator werden dann die Beispiele aufgeteilt und in Schritt 4(c)v auf den Stack gelegt. Es hat sich gezeigt, dass sich die Klassifikationsleistung steigern lässt, indem die Frequenztabellen der Entscheidungsfunktionen der Blattklassifikatoren in Schritt 5 mit allen Beispielen $\mathcal{D}^0$ erneut berechnet werden.

Um bei der Ausführung des Baumes, abhängig von der Konfidenz, die unterschiedlichen Pfade zu den Blattklassifikatoren zu berechnen, wird bei jedem Splitklassifikator der Konfidenzwert für das Muster überprüft und in Abhängigkeit dieses Konfidenzwertes entweder nur der linke, nur der rechte oder beide Pfade zu den Kindknoten weiterverfolgt. Wie beim Training wird dazu ein Stack eingesetzt.

Alg. (6.5) zeigt den Algorithmus zur Klassifikation eines Musterbeispiels. Der Verzweigungsfaktor γ_θ ist dabei eine Konfidenzschwelle. Für Konfidenzwerte der Splitklassifikatoren, die diese Schwelle unterschreiten wird in Schritt 3(c)i der rechte und linke Teilpfad weiterverfolgt. Mit diesem Faktor lässt sich also der Verzweigungsgrad in die Teilbäume bestimmen und somit eine Regulierung zwischen Klassifikationsperformance und mittlerer Laufzeit erreichen. Die Konfidenzberechnung in Schritt 3a kann mit den eingeführten Modellen zur Berechnung der Konfidenz erfolgen, allerdings ist es numerisch stabiler, die Konfidenz für Splitklassifikatoren ohne die A-priori-Wahrscheinlichkeiten zu berechnen, da sie für den Splitvergleich nicht benötigt wird. Für die Blattklassifikatoren werden die A-priori-Wahrscheinlichkeiten verwendet. Die normierte Konfidenz bietet den Vorteil,

Algorithmus 6.5 Klassifikation mit konfidenzbasierten Pfaden

1. **Gegeben:** Baumklassifikator $\mathcal{T}$, zu klassifizierendes Beispiel $\underline{z}$, Verzweigungsfaktor γ_θ
2. Initialisiere Stack $\mathcal{S} = h_0$, wobei h_0 der Wurzelklassifikator ist.
3. **Solange** der Stack Elemente enthält $h_i \in \mathcal{S}$:
   (a) Berechne Konfidenz $\gamma(h_i)$ mit dem Beispiel $\underline{z}$, für Splitklassifikator ohne A-priori-Wahrscheinlichkeiten.
   (b) Falls h_i Blattklassifikator ist:
   Berechne Blattklassifikator $h_{\max}$ mit maximaler Konfidenz $\gamma_{\max}$.
   (c) Falls h_i Splitklassifikator ist, berechne Konfidenz $\gamma(h_i, \underline{z})$:
      i. **Falls** $\gamma(h_i) < \gamma_\theta$:
      Verfolge linken und rechten Teilbaum
      $\mathcal{S} \leftarrow \mathcal{S} \cup h_{i,L} \cup h_{i,R}$
      ii. **Sonst**, falls $h_i(\underline{z}) = L$:
      Verfolge linken Teilbaum $\mathcal{S} \leftarrow \mathcal{S} \cup h_{i,L}$
      iii. **Sonst**, falls $h_i(\underline{z}) = R$:
      Verfolge rechten Teilbaum $\mathcal{S} \leftarrow \mathcal{S} \cup h_{i,R}$
4. **Ausgabe**: Klassenzuweisung $h_{\max}(\underline{z})$ und Konfidenz $\gamma_{\max}$

dass der Wertebereich im Intervall $[0, \ldots, 1]$ liegt. Abhängig vom Konfidenzwert werden dann entweder beide Teilbäume in Schritt 3(c)i weiterverfolgt oder abhängig vom Klassifikationsergebnis nur der linke oder rechte Teilbaum. Der Klassifikator mit maximaler Konfidenz entscheidet über die Klassenzugehörigkeit, indem in Schritt 3b ein Index auf diesen Klassifikator berechnet wird. Die Konfidenzberechnung in den Blattklassifikatoren wird mit den A-priori-Wahrscheinlichkeiten berechnet. Somit entscheidet unter den ausgewählten Blattklassifikatoren derjenige mit maximaler Konfidenz über die Klassenzugehörigkeit und den entsprechenden Konfidenzwert. Zum Vergleich wurde noch ein Klassifikationsalgorithmus getestet, der die Klassenzugehörigkeit als Mehrheitsentscheidung der Blattklassifikatoren berechnet und in Alg. (6.6) dargestellt ist.

Im Folgenden werden die Unterteilungsbäume als Stufenklassifikatoren

Algorithmus 6.6 Klassifikation mit Mehrheitsentscheid

1. **Gegeben:** Baumklassifikator $\mathcal{T}$, zu klassifizierendes Beispiel $\underline{z}$
2. **Ausgabe**: Klassenzuweisung $h_{\text{mehr}}(\underline{z})$ und Konfidenz γ_{mehr} als Mehrheitsentscheidung aller Blattklassifikatoren

einer trainierten Kaskade und deren Struktur untersucht. Als Stopkriterium für die Splitklassifikatoren wurde eine minimale Anzahl von 200 Negativbeispielen und 1000 Positivbeispielen gewählt. Um auch mit wenigen Beispielen die Einträge der Frequenztabellen zu berechnen wurde die Anzahl der Einträge e der Frequenztabellen mit

$$e = \frac{1}{160} \min(|\mathcal{D}_1|, |\mathcal{D}_0|) \tag{6.3.3}$$

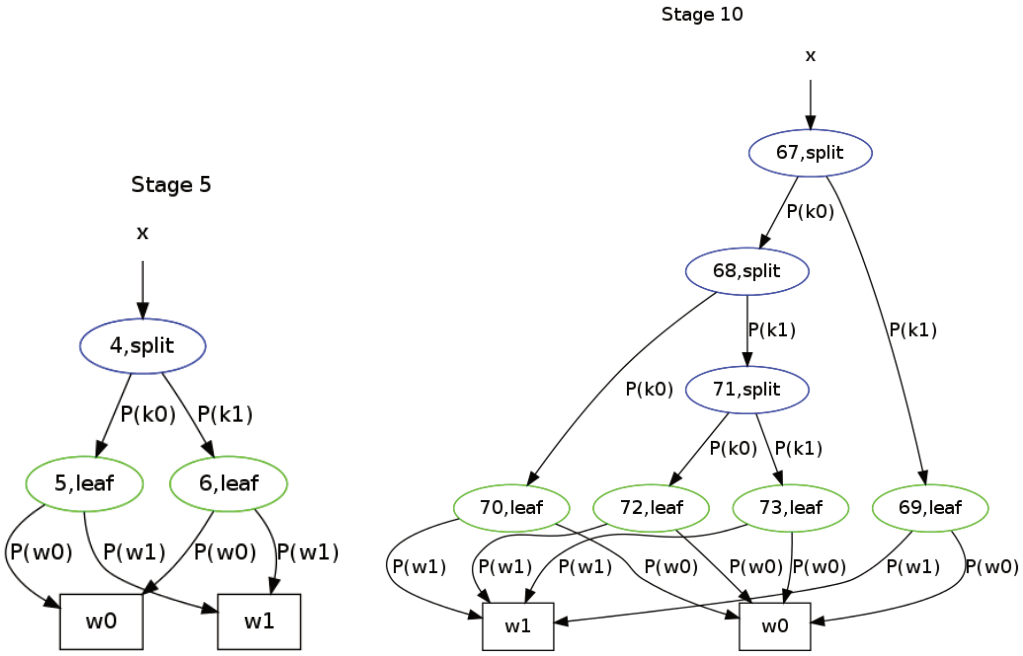
berechnet. Die Blattklassifikatoren wurden mit Alg. (4.2) zur sequentiellen Vorwärtsauswahl der Merkmale trainiert mit einer maximalen Anzahl von $t_{\max} = 8$ Iterationen. Um eine Überanpassung der Klassifikatoren zu vermeiden wurde die maximal mögliche Dimension des Merkmalsvektors durch

$$d_{\max} = 0.1 \min(|\mathcal{D}_1|, |\mathcal{D}_0|) \tag{6.3.4}$$

beschränkt. Die auflösungsspezifische Kaskade wurde mit den Auflösungsstufen $(r_{\Delta\frac{1}{3}}, r_{\Delta\frac{1}{4}}, r_{\Delta\frac{1}{5}}, r_{\Delta\frac{1}{6}}, r_{\Delta\frac{1}{12}})$ und einer Auftrittwahrscheinlichkeit von $e(\omega_0) = 0.8$ als Stop-Kriterium trainiert.

Abb. 6.3.2 zeigt eine Visualisierung der Stufenklassifikatoren. Die Splitklassifikatoren sind blau und die Blattklassifikatoren grün markiert. Die Klassenentscheidungen sind durch die rechteckigen Kästen dargestellt, wobei w1 der Objektklasse und w0 der Hintergrundklasse entspricht. Der erste Baum mit einem Splitklassifikator taucht in Stufe 5, Auflösung $r_{\Delta\frac{1}{3}}$ auf (siehe Abb. 6.3.2(a)). Die Baumstrukturen weisen innerhalb einer Auflösungsstufe mit steigenden Stufen eine höhere Anzahl an Verzweigungen, sowie Split- und Knotenklassifikatoren auf (siehe Abb. 6.3.2(b)).

Abb. 6.3.6 zeigt einen Baum, der aus einer großen Anzahl an Split- und Blattknoten besteht und in Stufe 15 entstanden ist. Mittels Alg. (6.5) werden bei der Klassifikation mehrere Pfade hinuntergelaufen, die grün eingefärbt sind. Unter den beteiligten Blattklassifikatoren (magenta) wird derjenige mit maximaler Konfidenz zur Klassenzuweisung verwendet (rot). Um die Merkmalsauswahl und die Klassifikationsfunktionen für einen Splitklassifikator zu



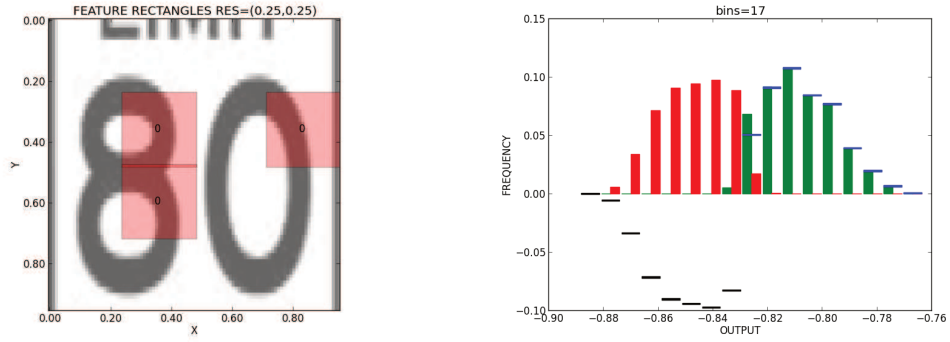
(a) Stufe 5, $r_{\Delta\frac{1}{3}}$

(b) Stufe 10, $r_{\Delta\frac{1}{3}}$

Abbildung 6.3.2: Visualisierung der Baumklassifikatoren in den jeweiligen Stufen. Blaue Knoten entsprechen Splitklassifikatoren und grüne Knoten den Blattklassifikatoren. Objektklasse: w1, Hintergrundklasse: w0.
(a): Stufe 5, $r_{\Delta\frac{1}{3}}$, (b): Stufe 10, $r_{\Delta\frac{1}{3}}$

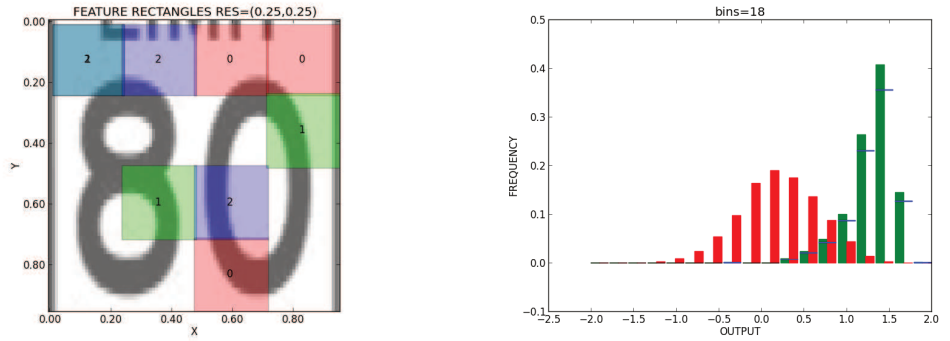
überprüfen sind in Abb. 6.3.3 die geomtrische Anordnung der Merkmale und die Merkmalstypen (Abb. 6.3.3(a)), sowie die Entscheidungsfunktion (Abb. 6.3.3(b)) dargestellt. Die Merkmale wurden mit dem K-Meansverfahren und dem Optimierungskriterium mit Gl. (6.3.2) bestimmt. Das Rechteck auf der rechten Seite lässt sich dadurch erklären, dass die rechte Ziffer entweder eine Null oder eine Fünf ist und die Unterschiede an dieser Stelle stark ausgeprägt sind.

Für die Entscheidungsfunktion wurden die Häufigkeiten akkumuliert und die A-priori-Wahrscheinlichkeiten auf identische Werte gesetzt. Abb. 6.3.4 zeigt den linken Kindklassifikator von Klassifikator 71. Die Merkmale wurden mit der sequentiellen Vorwärtsauswahl und dem FLDA-Verfahren (vgl. Kapitel 3, Abschnitt 3.4.1) in drei Iterationsschritten bestimmt und beste-



(a) (b)

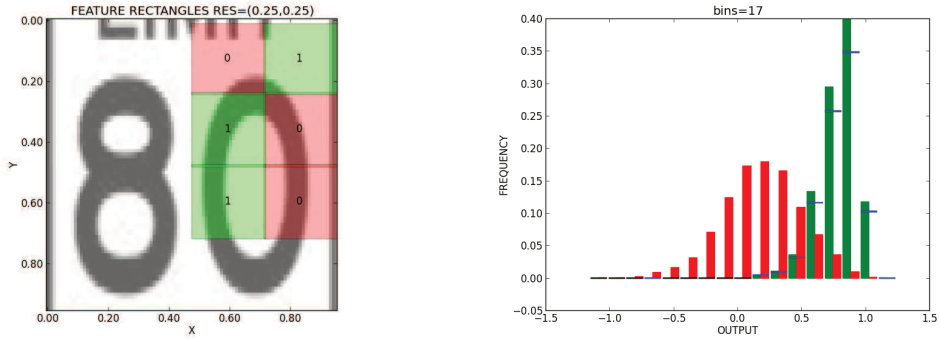
Abbildung 6.3.3: Splitklassifikator 71 aus Abb. 6.3.2(b)
(a): Merkmale $(t_e(r_i), \sqrt{\overline{i_x^2}(r_i) + \overline{i_y^2}(r_i)})^T$ für jedes Rechteck r_i
(b): Entscheidungsfunktion für Splits mit ident. A-priori-Wahrscheinlichkeiten



(a) (b)

Abbildung 6.3.4: Blattklassifikator 72 aus Abb. 6.3.2(b)
(a): Merkmale $(o_x(r_i), o_y(r_i)^T)$ für $0 \leq i \leq 1$ und $(\bar{i}(r_i), \overline{i^2}(r_i), \overline{i^2}(r_i) - (\bar{i})^2(r_i))^T$ für $i = 2$ (b): Entscheidungsfunktion.

hen für die ersten 6 Rechtecke aus den tensorbasierten Orientierungsmerkmalen. Für die letzten drei Rechtecke wurden intensitätsbasierte Merkmale gewählt. Bei der Entscheidungsfunktion in Abb. 6.3.4(b) werden die A-priori-Wahrscheinlichkeiten berücksichtigt.



(a) (b)

Abbildung 6.3.5: Blattklassifikator 73 aus Abb. 6.3.2(b)
(a): Merkmale $(o_x(r_i), o_y(r_i), \bar{i}(r_i), \overline{i^2}(r_i))^T$ für $i = 0$
und $(o_x(r_i), o_y(r_i), o_x^2(r_i), o_y^2(r_i), o_x(r_i)o_y(r_i))^T$ für $i = 1$
(b): Entscheidungsfunktion

6.4 Experimentelle Evaluierung

Wie in allen anderen Versuchen wurde die Evaluierung auf dem in Kapitel 4, Abschnitt 4.5 beschriebenen Datensatz durchgeführt, wobei ein aktueller Rechner mit einem 3.3 GHz Prozessor (Intel i7 - Architektur) verwendet wurde, auf dem ein spezifischer Klassifikator mit einem Prozessor trainiert und mit einem Prozessor evaluiert wurde. Die Klassifikatoren werden mit dem Trainingsdatensatz generiert und auf dem separaten Testdatensatz, der keine identischen Realweltschilder enthält, ausgewertet. Die Einheit der dargestellten Falsch-Positiven bei den ROC-Kurven sind Falsch-Positive pro Bild, was nicht mit den Falsch-Positiven pro Detektionsfenster verwechselt werden sollte. Im ersten Teil dieser Auswertung sind die Versuche und Auswertungen zu den Unterteilungsbäumen beschrieben. Im darauffolgenden Teil sind die Versuche zum konfidenzbasierten Gradientenboosting beschrieben. Im letzten Teil folgen dann die verwendeten Softwaredesignkriterien und Verfahren.

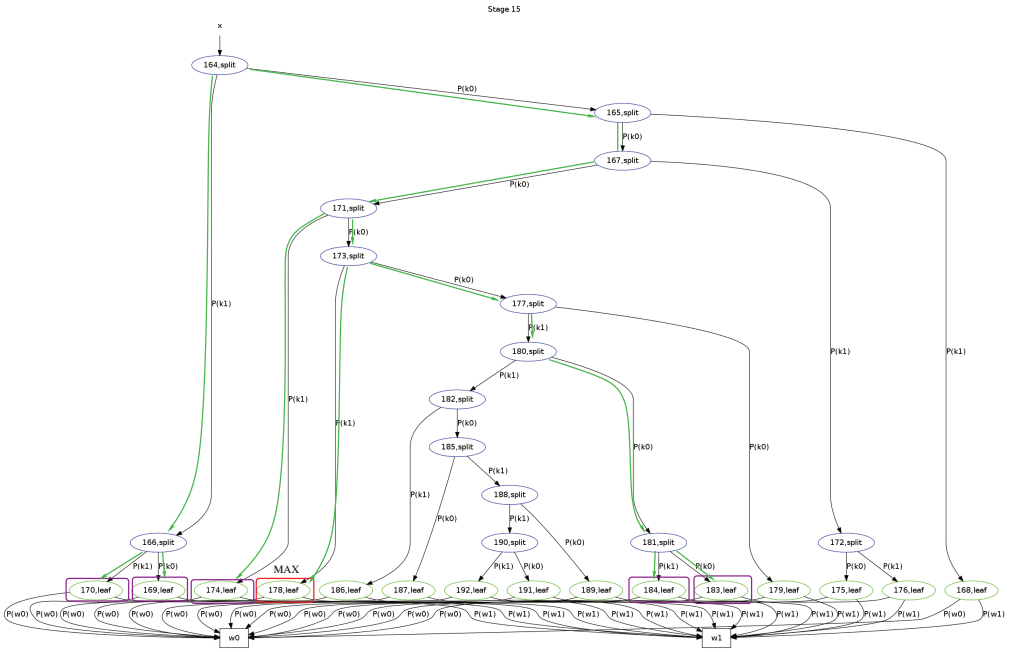
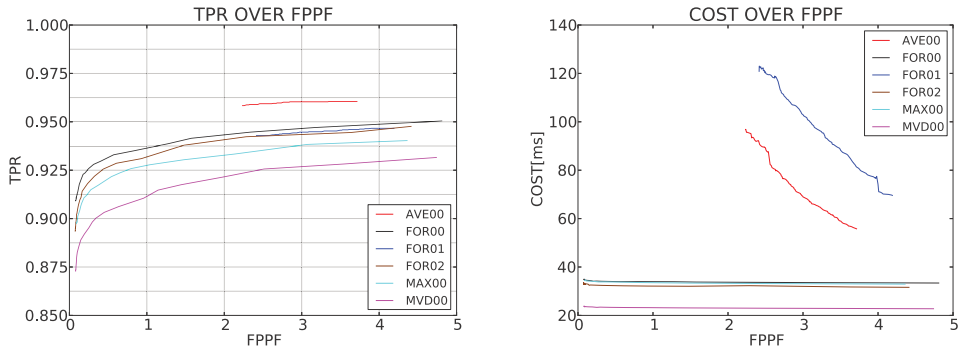


Abbildung 6.3.6: Stufenklassifikator 15, $r_{\Delta\frac{1}{4}}$ und zusätzliche farbliche Markierung der beteiligten Blattklassifikatoren (magenta) und dem Klassifikator mit maximaler Konfidenz (rot).

6.4.1 Ressourcenoptimierte, konfidenzbasierte Unterteilungsbäume

Bei diesen Versuchen sollen die Auswirkungen der unterschiedlichen Parameter der Entscheidungsfunktionen in den Baumklassifikatoren durch Alg. (6.5) untersucht werden. Dazu werden unterschiedliche Verzweigungsfaktoren γ_θ getestet. Außerdem werden bei zwei Entscheidungsfunktion alle Blattklassifikatoren ausgewertet, bei denen zum einen der Klassifikator mit maximaler Konfidenz (“winner takes all”) entscheidet und zum anderen die gemittelte Entscheidung aller Blattklassifikatoren verwendet wird (“Mehrheitsentscheid”). Bei einer weiteren Variante wird ohne Verzweigungen nur ein Pfad verfolgt, wobei der einzelne zugeordnete Blattklassifikator entscheidet. Bei den Versuchen wurden die Blattklassifikatoren mit der sequentiellen Vorwärtsauswahl (Alg. (4.2)) und deren Entscheidungsfunktionen durch Akkumulation der Häufigkeiten (Alg. (3.7)) trainiert. Für die Bestimmung der



(a) (b)

Abbildung 6.4.1: Auswertung der Versuche zu den Unterteilungsbäumen (a): ROC-Kurven über den Falsch-Positiven pro Bild (b): Mittlere Laufzeiten in ms über den Falsch-Positiven pro Bild. Die Parameter sind in Tabelle 6.4.1 beschrieben.

Anzahl der Einträge in den Frequenztabellen wurde Gl. (6.3.3) verwendet, da die Anzahl der Muster in den Split- und Blattklassifikatoren abhängig von der Tiefe des Baumes starken Schwankungen unterliegt. Die Merkmale sind identisch mit den Merkmalen aus den Versuchen in Abschnitt 4.8.7. In

Parameter		
ID	γ_θ	**Beschreibung**
FOR00	0.25	Verzweigungsgrad Alg. (6.5)
FOR01	0.5	Verzweigungsgrad Alg. (6.5)
FOR02	0.75	Verzweigungsgrad Alg. (6.5)
MVD00	-	Einzelner Pfad
AVE00	-	Gemittelte Entscheidung aller Blätter
MAX00	-	Max. Konfidenz aller Blätter

Tabelle 6.4.1: Parameter der Entscheidungsfunktionen zum Versuch Unterteilungsbäume

Tabelle 6.4.1 sind die entsprechenden Parameterkonfigurationen gelistet.

Abb. 6.4.1 zeigt die Auswertung der Versuche, wobei die ROC-Kurven der Detektionsraten über den Falsch-Positiven pro Bild in Abb. 6.4.1(a) und die mittleren Laufzeiten in Millisekunden über den Falsch-Positiven pro Bild

in Abb. 6.4.1(b) zu sehen sind.

Wie erwartet benötigen die Klassifikatoren bei denen alle Blattklassifikatoren ausgewertet werden (vgl. AVE00 und MAX00) wesentlich mehr Operationen als die anderen Klassifikatoren. Während sich die mittleren Laufzeiten bei fallenden Falsch-Positiven pro Bild ausgehend von 5 FPPI kaum nach oben verändern steigen die Kosten von AVE00 und MAX00 fast linear an. Das Training beider Klassifikatoren ist bei 2 FPPI in eine Sättigung geraten. Beim Vergleich der Beiden hat MAX00 sowohl eine deutlich bessere Detektionsrate als auch niedrigere Berechnungskosten. Dieser Effekt lässt sich mit der Art und Weise des Trainings erklären. Durch das K-Means-Training der Splitklassifikatoren häufen sich in den Blattklassifikatoren ähnliche Muster an, die von den Splitklassifikatoren durch spezifische Merkmale "gefiltert" werden. Somit repräsentiert ein Blattklassifikator einen bestimmten Merkmalsbereich. Das einzelne zu klassifizierende Muster fällt zwangsläufig in einen solchen begrenzten Bereich, abhängig von der Anzahl der Blattklassifikatoren. Durch die Klassenentscheidung des Maximum Konfidenz-Klassifikators aller Blätter bei MAX00 wird dieses Muster besser repräsentiert, als bei der gemittelten Entscheidung von AVE00. Wobei auch über Entscheidungen gemittelt wird, die im Merkmalsraum weiter entfernt sind und dieses Muster kaum repräsentieren.

Die Klassifikation mit einem einzelnen Pfad mit MVD00 erzielt, was ebenfalls zu erwarten war, schnellere Laufzeiten wobei die Detektionsleistung im Vergleich niedriger ist. Die Klassifikatoren mit den unterschiedlichen Verzweigungsgraden γ_θ, FOR00 bis FOR02 erzielen Detektionsleistungen und mittlere Laufzeiten zwischen den Extremen MAX00 und MVD00.

6.4.2 Konfidenzbasierte Musterauswahl bei den Unterteilungsbäumen

Im letzten Versuch hat sich gezeigt, dass die mittleren Laufzeiten der Unterteilungsbäume im Echtzeitbereich liegen, jedoch befinden sich die Klassifikationsleistungen leicht unter den bisher betrachteten. Deshalb wurde das Verfahren zur konfidenzbasierten, integrierten Musterauswahl mit der Korrektur der A-priori-Wahrscheinlichkeiten durch Gl. (5.3.1) mit den Unterteilungsbäumen (Alg. (6.3)), das in Abschnitt 5.3 beschrieben ist, kombiniert. Die Blattklassifikatoren wurden mit der sequentiellen Vorwärtsauswahl (Alg. (4.2)) mit maximal acht Zyklen trainiert. Die Fehlerschwelle zur Erweiterung

der Merkmale wurde auf $e_{\mathrm{SFS}}(\omega_0) = 0.7$ gesetzt. Außerdem wurden im Gegensatz zum obigen Versuch die Blattklassifikatoren auf feste Signifikanzniveaus mittels Alg. (3.6) von $\alpha = 0.05$ skaliert. Für die konfidenzbasierte Musterauswahl sind die Entscheidungsfunktionen mit Frequenztabellen zwingend notwendig. Die Anzahl der Einträge wird wie oben beschrieben abhängig von der Anzahl der Trainingsmuster mit Gl. (6.3.3) bestimmt. Darüber hinaus wurde zusätzlich ein Unterteilungsbaum mit einer schwellwertbasierten Entscheidungsfunktion trainiert.

Die Merkmale und Auflösungsstufen sind identisch mit den Merkmalen wie sie in Abschnitt 4.8.7 beschrieben sind und damit identisch zu den Merkmalen im obigen Versuch. Die Fehlerschwelle der Stufenklassifikatoren bei der in eine neue Auflösungsstufe gesprungen wird (Alg. (4.5) ist $e_{\mathrm{stop}}(\omega_0) = 0.8$. Die minimale Anzahl an Beispielen, die in Alg. (6.3) bei Unterschreitung zur Beendigung der Verzweigungen und zum Training der Blattklassifikatoren führt, wurde auf $m = 1000$ gesetzt. Da diese Menge im Vergleich zu den über 10000 Verfügbaren Mustern in der aktuellen Stufe sehr gering ist, wurde bei diesem Versuch außerdem eine Variante getestet, bei der die Entscheidungsfunktionen der Blattklassifikatoren nach dem Ende des Baumtrainings mit allen Trainingsmustern in der aktuellen Stufe neu berechnet werden. Tabelle 6.4.2 listet die unterschiedlichen Parameter des Versuchs. Bei den mit „-“ markierten Kästchen in der Spalte des Parameters ξ wurde keine konfidenzbasierte A-priori-Modifikation zur Musterauswahl verwendet (Gl. (5.3.1)). Ebenso wurden in den mit „-“ markierten Kästchen des Parameters γ_θ (Alg. (6.5)) keine Verzweigungen durchgeführt, sodass nur ein einzelner Blattklassifikator die Klassenzugehörigkeit entscheidet. In der letzten Spalte wurden die Frequenztabellen der Blattklassifikatoren mit allen verfügbaren Trainingsmustern der jeweiligen Stufe trainiert, falls die Kästchen mit „✓“ markiert sind. Abb. 6.4.2 zeigt die ROC-Kurven zu den Konfigurationen aus Tabelle 6.4.2. Es zeigt sich, dass durch die konfidenzbasierte Musterauswahl bei den Klassifikatoren FB00, FB01 und FB02 eine Verbesserung der Klassifikationsrate bei den Unterteilungsbäumen von durchschnittlich 5% auf ca. 97,5%@1FPPF, erreicht wird (siehe Abb. 6.4.2(a)). Es werden dafür deutlich längere mittlere Laufzeiten benötigt, wobei mit dem Verzweigungsfaktor von $\gamma_\theta = 0.5$ bei FB02 mit nahezu identischer Detektionsrate eine Halbierung der mittleren Laufzeiten auftritt (siehe Abb. 6.4.2(b)). Abb. 6.4.2(c) zeigt die Aulösungen in den entsprechenden Stufen. Es fällt auf, dass die Gesamtzahl der Stufenklassifikatoren bzw. Unterteilungsbäume zwischen 35 und 45 liegt und damit deutlich geringer ist, als bei den Kaskanden mit linearen

Parameter					
ID	γ_θ	ξ	$t_{\max}$	Entscheidung	Alle Muster
FB00	0.75	0.75	8	$\max(P_s(\omega_i))$	✓
FB01	0.25	0.75	8	$\max(P_s(\omega_i))$	✓
FB02	0.5	0.75	8	$\max(P_s(\omega_i))$	✓
FB03	0.5	-	8	$\max(P_s(\omega_i))$	✓
FB04	0.25	-	8	$\max(P_s(\omega_i))$	✓
FB05	0.25	-	8	$\max(P_s(\omega_i))$	-
FB06	0.5	-	8	$\max(P_s(\omega_i))$	-
FB07	0.75	-	8	$\max(P_s(\omega_i))$	-
FB08	0.75	-	8	$\max(P_s(\omega_i))$	✓
SP00	-	-	1	$\Theta(x_s - \theta)$	-
SP01	-	-	8	$\max(P_s(\omega_i))$	✓
SP02	-	-	8	$\max(P_s(\omega_i))$	-

Tabelle 6.4.2: Unterschiedliche Parameter des Versuchs zu den Unterteilungsbäumen. Der Parameter γ_θ ist Alg. (6.5) zuzuordnen, ξ gehört zu Gl. (5.3.1) und $t_{\max}$ stammt von Alg. (4.2). Die letzte Spalte bedeutet, dass alle Muster des aktuellen Stufenproblems zur Berechnung der Entscheidungsfunktionen in den Blättern verwendet werden.

Stufenklassifikatoren, die bis zu 200 Stufen enthalten. Dieser Effekt resultiert aus der Unterteilung des Stufenproblems in homogenere, "einfachere" Teilprobleme. Abb. 6.4.2(d) zeigt die mittleren Laufzeiten über den Stufen der Kaskadenklassifikatoren. Bei FB00 tritt ein großer Anstieg der Kosten bei dem Wechsel in die zweite Auflösungsstufe statt.

Bei Betrachtung der restlichen Klassifikatoren lässt sich feststellen, dass die Berechnung der Entscheidungsfunktionen mit allen Beispielen einen leichten Anstieg der Klassifikationsleistung bei ähnlichen mittleren Berechnungskosten zur Folge hat. Dabei erzielen die Klassifikatoren FB03 mittlere Laufzeiten im Echtzeitbereich von 18ms@1FPPF und FB04 von 22ms@1FPPF mit Klassifikationsraten von 92.5%@1FPPF. Durch Variation von ξ lassen sich die Klassifikatoren graduell zwischen Klassifikationsleistung und mittlerer Laufzeit variieren. Die ROC-Kurven der Klassifikatoren mit einzelnen Entscheidungspfaden sind in Abb. 6.4.3 dargestellt. SP00 erzielt bessere Detektionsraten bei ähnlichen mittleren Laufzeiten (Abb. 6.4.3(a) und 6.4.3(b)). Die Entscheidungsfunktionen der Split- und Blattklassifikatoren sind schwell-

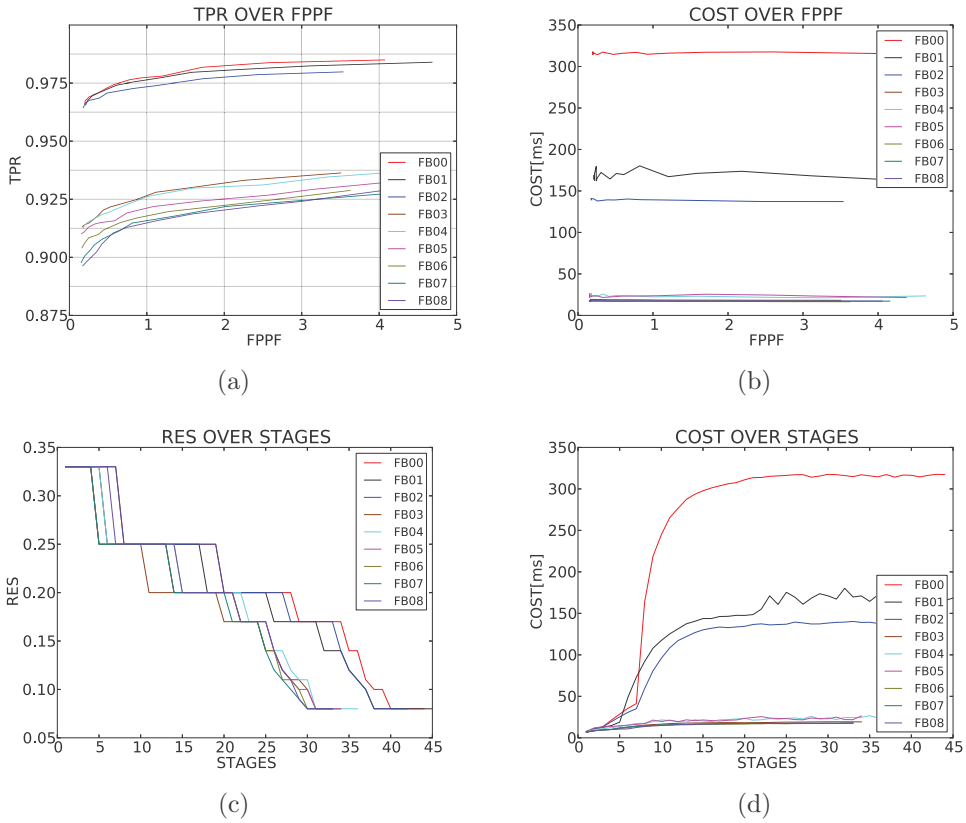


Abbildung 6.4.2: ROC-Kurven der Unterteilungsbäume (Alg. (6.3)) mit konfidenzbasierter Musterauswahl (Gl. (5.3.1)). (a): Detektionsraten über den Falsch-Positiven pro Bild. (b): Mittlere Laufzeiten pro Bild über den Falsch-Positiven. (c): Auflösungen in den Stufen. (d) mittlere Laufzeiten für die entsprechenden Stufen.

wertbasiert. Bei der relativ niedrigen Anzahl der Trainingsmuster in den Blattklassifikatoren sind die schwellwertbasierten Entscheidungen stabiler. Die Nachteile sind jedoch zum einen, dass durch die fehlende Konfidenz keine multiplen Verzweigungspfade erzeugt werden können und sich zum anderen die konfidenzbasierte Musterauswahl zur Steigerung der Klassifikationsleistung nicht anwenden lässt.

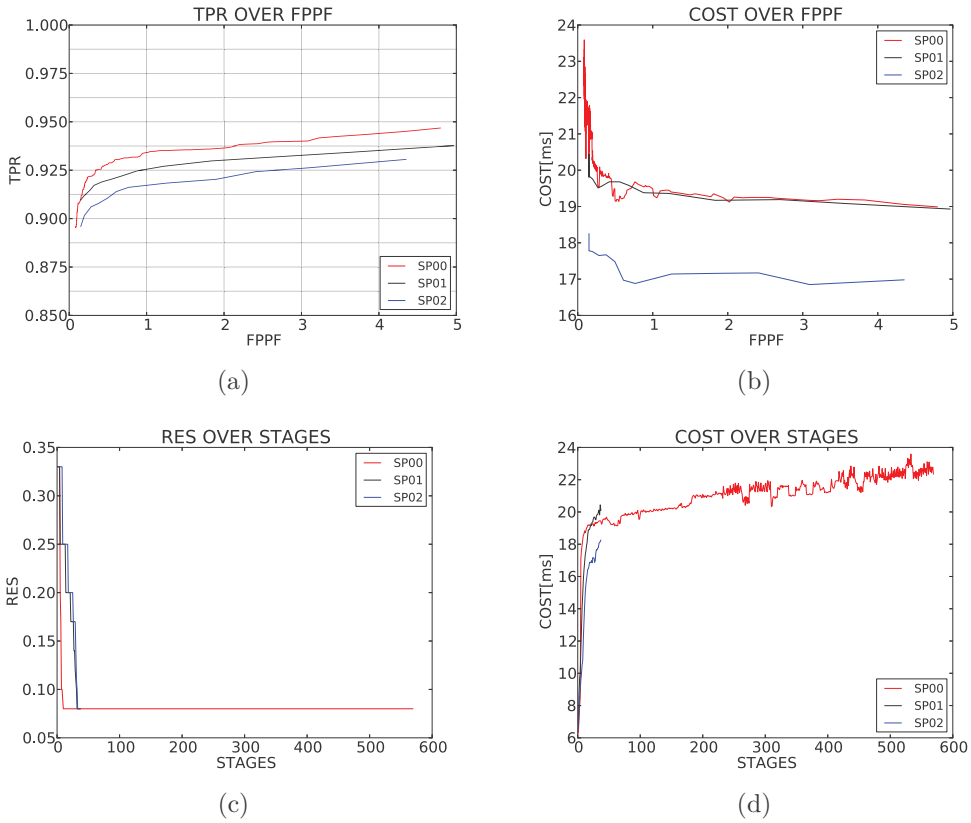


Abbildung 6.4.3: ROC-Kurven der Unterteilungsbäume (Alg. (6.3)) mit einfachen Entscheidungspfaden.

6.4.3 Konfidenzbasiertes Gradientenboosting

Bei diesem Versuch wird das Gradientenboosting (Alg. (6.2)) auf dem US-Speed-Limit Datensatz getestet. Jeder Stufenklassifikator besteht aus mehreren geboosteten Basisklassifikatoren. Um die Berechnungseffizienz zu erhöhen wird, falls ein einzelner Basisklassifikator die Schwelle $e_{\mathrm{gb}}(\omega_0)$ unterschreitet, nur dieser Klassifikator als Stufenklassifikator gespeichert. Um möglichst laufzeiteffiziente Klassifikatoren zu erzeugen wird die maximale Anzahl an Basisklassifikatoren $t_{\max}$ auf entweder 8 oder 16 gesetzt, denn in der Versuchsreihe zu den Merkmalen wurde das Gradientenboosting mit maximal 256 Basisklassifikatoren und einer mittleren Auflösung angewendet. Dort wurde ersichtlich, dass die hohe Anzahl an Basisklassifikatoren

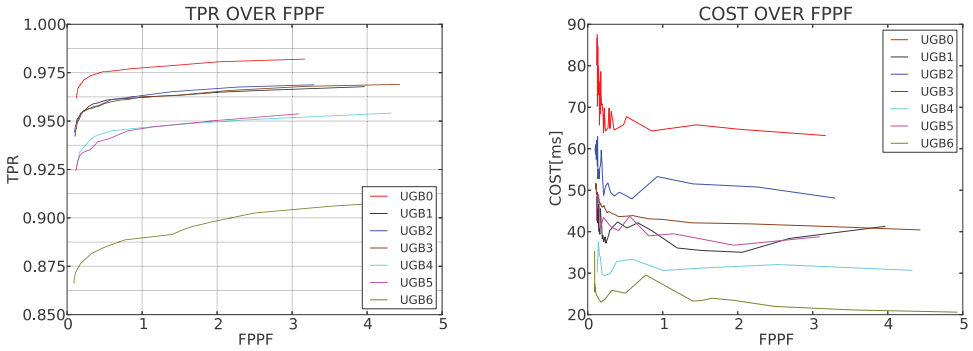
Parameter						
ID	Eintäge FT	$t_{\max}$	$e_{\text{gb}}(\omega_0)$	$e_{\text{stop}}(\omega_0)$	ξ	Stufen
UGB0	64	16	0.6	0.7	0.9	57
UGB1	64	8	0.75	0.8	1	65
UGB2	64	16	0.6	0.7	1	52
UGB3	64	8	0.6	0.7	1	76
UGB4	256	8	0.6	0.7	1	32
UGB5	256	16	0.6	0.7	0.9	34
UGB6	256	16	0.6	0.7	1	34

Tabelle 6.4.3: Parameter des Gradientenboosting-Versuchs mit US-Verkehrszeichen. Die Entscheidungsfunktionen $\max(P_s(\omega_i))$ der Basis- und des Ensembleklassifikators enthalten die identische Anzahl an Einträgen (siehe Spalte Einträge FT). $t_{\max}$ ist die maximale Anzahl an Basisklassifikatoren aus Alg. (6.2). $e_{\text{gb}}(\omega_0)$ ist das Stop-Kriterium für das Boosting neuer Klassifikatoren. $e_{\text{stop}}(\omega_0)$ ist die Auftrittswahrscheinlichkeit der Hintergrundklasse und das Kriterium zum Wechsel in eine neue Auflösungsstufe nach Alg. (4.5). Der Paramter ξ steuert die integrierte, konfidenzbasierte Musterauswahl nach Gl. (5.3.1).

einen starken Einfluss auf die mittlere Laufzeit hat. Die Auflösungsstufen mit den entsprechenden Merkmalen sind mit den Merkmalen aus den vorherigen Versuchen identisch: $\mathcal{F}_{\Delta r} = \{F_{\Delta\frac{1}{3}}, \ldots, F_{\Delta\frac{1}{12}}\}$. Um höhere Detektionsraten zu erzielen wird die Schwelle der maximalen Auftrittswahrscheinlichkeiten $e_{\text{stop}}(\omega_0)$ für den Wechsel in eine neue Auflösungsstufe auf 0.7 bzw. 0.8 gesetzt. Die Schwelle für die letzte Auflösungsstufe ist bei allen Klassifikatoren $e_{\text{stop}}(\omega_0) = 0.999$. Die Konfidenzen werden, wie in Abschnitt 6.2 beschrieben, mit den Frequenztabellen berechnet. Bei der sequentiellen Merkmalsauswahl in Versuch 4.8.5 erzielten die Tabellen mit 256 Einträgen die besten Ergebnisse. Deshalb werden Entscheidungsfunktionen mit 256 Einträgen und 64 Einträgen getestet, denn bei weniger Einträgen sollten die Entscheidungen weniger fehleranfällig sein. Die Basisklassifikatoren und der Ensembleklassifikator besitzen eine identische Anzahl an Einträgen. Bei Versuch 5.4 zur konfidenzbasierten Musterauswahl wurden sehr gute Detektionsergebnisse erzielt. Deshalb ist der Korrekturprior aus Gl. (5.3.1) bei zwei Klassifikatoren auf 0.9 gesetzt.

Tabelle 6.4.3 zeigt die unterschiedlichen Parameter der Kaskaden, die mit

der ID-Nummer in der ersten Spalte identifiziert werden.



(a) (b)

Abbildung 6.4.4: ROC-Kurven des Versuchs Gradientenboosting mit den Parameterkonfigurationen aus Tabelle 6.4.3 (a): Detektionsraten über den Falsch-Positiven pro Bild (b): Mittlere Laufzeiten über den Falsch-Positiven pro Bild in Millisekunden

Die entsprechenden ROC-Kurven sind in Abb. 6.4.4 gezeigt, wobei links die Detektionsraten über den Falsch-Positiven pro Bild (Abb. 6.4.4(a)) und rechts die mittleren Laufzeiten in Millisekunden über den Falsch-Positiven pro Bild (Abb. 6.4.4(b)) gezeigt sind.

Die Kaskade UGB0 erreicht die höchsten Detektionsraten von 98%@1FPPF bei einer durchschnittlichen Laufzeit von 65 ms pro Bild. Bei diesem Arbeitspunkt findet auch der Übergang in die letzte Auflösungsstufe statt. Der minimale ROC-Punkt befindet sich bei 96%@0.125FPPF bei einer erhöhten mittleren Laufzeit von 87 ms. Die Klassifikatoren UGB1 bis UGB3 erzielen ähnliche Detektionsraten von 96%@1FPPF, wobei sich die Laufzeiten mit UGB1: 38ms@1FPPF, UGB2: 52ms@1FPPF und UGB3: 43ms@1FPPF unterscheiden. Bei UGB1 wurde die Schwelle für das Hinzufügen neuer Klassifikatoren auf $e_{\mathrm{gb}}(\omega_0) = 0.75$ erhöht, was zur Folge hat, dass der Ensembleklassifikator erst bei niedrigeren Falsch-Positiv-Raten anwächst, wodurch die verbesserte Laufzeit und die maximale Anzahl von 65 Stufen erzielt wird.

Die Detektionsraten der Klassifikatoren UGB4 bis UGB6 liegen wiederum deutlich unterhalb der bisher betrachteten Klassifikatoren, was mit der Anzahl der Einträge in den Frequenztabellen von 256 zu erklären ist. Durch die Modifikation der Gewichte der Beispiele in jeder Iteration beim Boosting

ist die effektive Anzahl der Trainingsbeispiele wesentlich geringer und damit auch die Evidenz bzw. Information pro Eintrag. Mit jedem neuen Basisklassifikator wird dieser Effekt noch verstärkt, da die Anzahl der konfidenten Beispiele steigt und die Anzahl der nicht konfidenten Beispiele, die hochgewichtet werden, sinkt. Genau dieser Effekt sorgt dafür, dass UGB4 mit einer Beschränkung von maximal acht Basisklassifikatoren bessere Detektionsraten als UGB6 mit einer maximalen Anzahl von 16 Basisklassifikatoren erzielt.

Somit wirkt sich die Anzahl der Basisklassifikatoren durch die Veränderung der Gewichte auf die Güte der Basisklassifikatoren und damit auch auf den Ensembleklassifikator aus. Es kann sehr schnell eine Überanpassung durch die veränderte Statistik der Beispiele entstehen. Bei den Unterteilungsbäumen konnte eine Überanpassung durch eine Beschränkung der Dimension der Merkmalsvektoren und eine lineare Berechnung der Anzahl der Einträge der Frequenztabellen abhängig von der Gesamtzahl der Trainingsmuster verhindert werden.

Eine Möglichkeit, die Überanpassung der Frequenztabellen zu verhindern ist das in Kapitel 3 vorgestellte Verfahren zur lokalen Akkumulation der Frequenztabellen.

Die in Kapitel 4, Abschnitt 4.8.2 durchgeführten Versuche zu den Merkmalen sind ebenfalls mit dem konfidenzbasierten Gradientenboosting durchgeführt worden, wobei auch die quadratische Variante, die in Abschnitt 6.2.1 beschrieben ist, getestet wurde. Außerdem wurde dort auch eine AdaBoost-Kaskade mit Haar-ähnlichen Merkmalen mit einer Gradientenboosting-Kaskade mit identischen Merkmalen verglichen, bei der die Gradientenboosting-Kaskade eine deutlich höhere Detektionsrate bei ähnlichen mittleren Laufzeiten erzielt. Eine genaue Beschreibung der Versuche ist in Abschnitt 4.8.2 zu finden.

6.5 Softwaredesign

Mit den im folgenden beschriebenen Methoden konnten sowohl echtzeitfähige Klassifikatoren für den industriellen Realwelteinsatz, als auch flexible Softwaremodule mit minimalen Abhängigkeiten erstellt werden, welche die Entwicklung der Verfahren unter wissenschaftlichen Entwurfs-, Implementierungs- und Testkriterien ermöglichten.

Die projektbedingte Anforderung an eine echtzeitfähige, stabile Softwa-

re war zum einen eine effiziente Programmiersprache und zum anderen das Vermeiden externer Bibliotheken. Zur Implementierung der unterschiedlichen Algorithmen wurden deshalb objektorientierte und allgemeingültige Softwaredesignkriterien auf eine Ansi-C Implementierung angewandt um eine hohe Flexibilität und gleichzeitig hohe Effizienz zu erreichen. Es wurden ausschließlich Klassen verwendet, bei denen das Information-Hiding durch Opaque-Zeiger realisiert wurde (Bridge Design Patterns). Polymorphe Klassenfunktionalität [24] ist durch Factory-Design-Patterns [68] realisiert, die für die abstrakte Klassifikator-Klasse, die Trainingsklasse, die Entscheidungsklasse und die Merkmalsklasse verwendet wurde, indem die unterschiedlichen Objekte mit nahezu identischen Funktionsinterfaces in die polymorphe Klasse eingebettet wurden. Die Funktionen werden über Funktionszeigertabellen verwaltet, indem für jede Funktion eine Plugin-Funktion verwendet wird, in welcher die datentypspezifischen Funktionen aufgerufen werden. Somit ist der Verwaltungsaufwand für einen Funktionsaufruf der polymorphen Funktion mit einem indizierten Sprung in die Funktionszeigertabelle sehr gering und unabhängig von der Anzahl der implementierten Datentypen. Bei der Erweiterung der polymorphen Klasse muss lediglich der neue Datentyp (z.B. der spezifische neue Klassifikator) als Attribut in eine Vereinigungsstruktur aufgenommen werden und die Funktionszeigertabellen im "versteckten" Implementationsteil erweitert werden. Dadurch ändert sich das Interface der polymorphen Klasse nicht und die Funktionalität des bestehenden Codes bleibt unverändert. Da die Interfaces der spezifischen Klassifikatoren nicht vollständig identisch sind, werden Funktionspointer der spezifischen Klasse, für die keine Funktion existiert, mit Null-Zeigern versehen, welche im abgesicherten Modus zur Laufzeit geprüft werden. Wird also versehentlich eine Funktion der abstrakten Klasse aufgerufen, die von der spezifischen Klasse nicht bedient werden kann, so wird das Programm mit der entsprechenden Fehlermeldung abgebrochen. Ein weiterer Vorteil ist, dass für dieses Verfahren keine void-Zeiger für die spezifischen Klassen verwendet werden und damit die Typüberprüfung durch den Compiler gewährleistet ist, da keine fehleranfälligen Typkonvertierungen durchgeführt werden müssen (Kompilationszeit-Polymorphie, statisches Binden). Ein Nachteil des Verfahrens ist der erhöhte Verwaltungsaufwand durch das Erstellen der Funktionszeigertabellen. Der größte Vorteil ist die enorme Flexibilität der Software. Der poylmorphe Klassifikator kann beliebig viele unterschiedliche Datentypen, welche durch die individuellen Klassifikatorentypen und Funktionen spezifiziert sind, annehmen. Damit lässt sich die Software erweitern, ohne dass die bestehende Funktio-

nalität beeinträchtigt wird. Für die Trainingsverfahren werden ebenfalls die polymorphen Klassifikatoren verwendet und somit lassen sich die Verfahren und Klassifikatoren softwareseitig beliebig kombinieren, woraus neue kombinierte Klassifikatorstrukturen und Trainingsverfahren entstehen. Für die rekursiven Verfahren, wie z.B. das Training und die Klassifikation mit den Unterteilungsbäumen (Alg. 6.3 und Alg. 6.5), aber auch für die auflösungsspezifische Depth-First-Detektion (Alg. 4.6) wird eine generische Stack-Klasse verwendet. Außerdem wurde eine umfangreiche Mathematik-Bibliothek erstellt, die unter anderem unterschiedliche numerische Zerlegungsverfahren zur Matrizenberechnung enthält. Um eine hohe Anzahl an Mustern der Hintergrundklasse effizient im Speicher zu halten sind die Muster als Strukturen mit der Klassenzugehörigkeit, dem umschließenden Bereich (Detektionsfenster) und einem Zeiger auf das zugehörige Integralbild sowie dem Bildnamen implementiert. Für den Zielklassifikator werden 10^7 Hintergrundmuster, die auf ca. 300 Integralbilder zeigen, verwendet. Dadurch steigt zwar der Speicherbedarf, dafür wird das Training beschleunigt, denn das Generieren der Hintergrundbeispiele nach jedem neuen Stufenklassifikator aus den Bildern ist zeitaufwändig. Die Liste der Bilder wird nach dem Einlesen permutiert, um eine bessere Verteilung der Einzelobjekte zu gewährleisten, da die Bilder als Sequenzen von identischen Verkehrszeichen von ca. 12 Bildern pro Realweltschild vorliegen.

Die Spezifikation der entsprechenden Trainings- und Klassifikatortypen wird durch Konfigurationsdateien bestimmt. Somit können unterschiedliche Parameter getestet werden, indem mehrere Trainingsdurchläufe parallel gestartet werden.

Abb. 6.5.1 zeigt die Realisierung der abstrakten Klassifikator-Klasse durch das Factory Design Pattern in der oberen Reihe. Die unterschiedlichen spezifischen Klassifikatoren sind in der zweiten Reihe dargestellt. Der Kaskadenklassifikator besitzt beispielsweise eine “detect”-Funktion, die bei den anderen Klassifikatoren nicht sinnvoll wäre. Beim Aufruf der abstrakten Funktionen werden die Funktionszeiger getestet und mit entsprechender Fehlermeldung abgebrochen. Ein späteres Nachrüsten der fehlenden Funktionen ist jederzeit möglich, ohne dass das Interface verändert werden muss. Der Kaskadenklassifikator, der Baumklassifikator und der Gradientenboosting-Klassifikator bestehen intern wiederum aus einer Liste von abstrakten Klassifikatoren und können somit jeden Klassifikatortyp als Stufenklassifikator bzw. Basisklassifikator enthalten.

Durch die Auftrennung in Training und Ausführungsanwendung muss je-

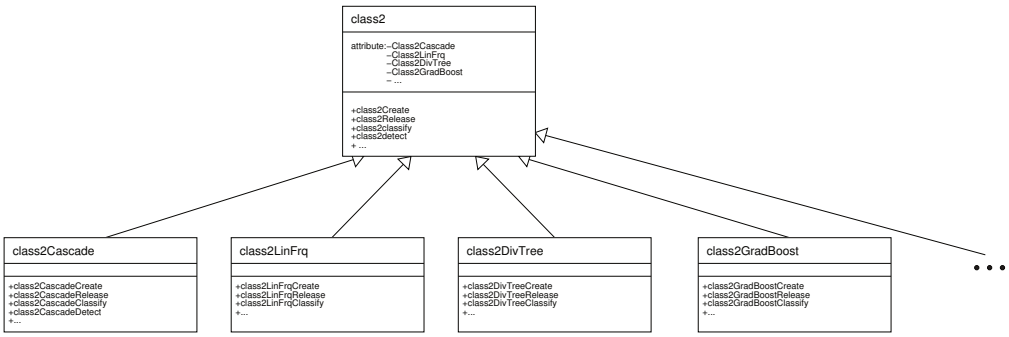


Abbildung 6.5.1: Factory Design Pattern der polymorphen Klassifikator-Klasse. Die spezifischen Klassen müssen nicht alle Funktionen der abstrakten Klasse bereitstellen.

der Klassifikator konsistent in einer Datei abgespeichert und später wieder eingelesen werden können. Es existieren also in Abb. 6.5.1 zusätzlich noch Lese- und Schreibfunktionen für jeden Klassifikator. Der Codeumfang der dadurch entstanden ist, sind 90 000 Zeilen Code, die auf 158 unterschiedliche Klassen aufgeteilt sind. Die Anzahl an vollständig trainierten und ausgewerteten Klassifikatoren beträgt 1050. Für die Erzeugung der Grafiken und ROC-Kurven wurde die Python-Bibliothek Matplotlib verwendet.

6.6 Zusammenfassung

In diesem Kapitel wurden unterschiedliche Ensembleverfahren beschrieben. In der Einleitung wurden die unterschiedlichen Verfahren in ihrer Funktionsweise beschrieben sowie deren Unterschiede erklärt. In Abschnitt 6.1 wurde das AdaBoost-Verfahren genauer beschrieben, sowie die Unterschiede der unterschiedlichen Boostingverfahren charakterisiert. Im nächsten Abschnitt 6.2 wurde das Gradientenboosting-Verfahren beschrieben und die im Rahmen dieser Arbeit entwickelten Erweiterungen des Verfahrens beschrieben. Die Konfidenzen des Ensembleklassifikators werden verwendet um die Gewichte der Muster nach jeder Iteration zu ermitteln, wobei schwach konfidente Muster höhere Gewichte erhalten und stark konfidente Muster niedrigere Gewichte. Mit den gewichteten Mustern wird dann der nächste Basisklassifikator trainiert. Für das Ensemble mit dem neuen Klassifikator werden die Gewichte der Klassifikatoren mit einer multidimensionalen Optimierungs-

methode wie z. B. der Fisher LDA bestimmt. Das Verfahren wird mit einer quadratischen Erweiterung des Merkmalvektors der Basisklassifikatoren erweitert.

Das im Abschnitt 6.3 beschriebene Verfahren ist ein neues Verfahren, das im Rahmen dieser Arbeit konzipiert wurde. Der Ensembleklassifikator ist ein Unterteilungsbaum, der mit sogenannten Splitklassifikatoren eine Unterteilung des Merkmalraumes ermöglicht, wobei ein zu klassifizierendes Muster die Splitklassifikatoren in sogenannten Entscheidungspfaden passiert und in einem Blattklassifikator endet. Der Verzweigungsgrad der Entscheidungspfade lässt sich mit einer Konfidenzschwelle einstellen. Falls die Konfidenz eines Splitklassifikators eine Schwelle unterschreitet, werden beide Entscheidungspfade weiterverfolgt. Da, je nach Verzweigungsgrad, mehrere unterschiedliche Blattklassifikatoren an der Entscheidung des Unterteilungsbaumes beteiligt sind, wurden unterschiedliche Verfahren zur Fusion vorgeschlagen. Das Training der Splitklassifikatoren erfolgt mit einem unüberwachten K-Means-Verfahren. Der optimale Splitklassifikator wurde aus einer Kandidatenmenge mit einem speziellen Optimierungskriterium ermittelt. Die Blattklassifikatoren und deren Trainingsverfahren können prinzipiell beliebig gewählt werden, allerdings wurden die ressourcenoptimierten Trainingsalgorithmen mit den Bayes'schen Entscheidungsfunktionen gewählt, da die Signifikanzniveaus und die aktualisierten A-priori-Wahrscheinlichkeiten wichtige Eigenschaften für eine hohe Klassifikationsleistung des Kaskadenklassifikators besitzen. Mit einer visuellen Strukturanalyse der Merkmale und der Entscheidungsfunktionen wurde die Funktionsweise der Split- und Blattklassifikatoren sowie deren Fusion erläutert.

Im Abschnitt 6.4.1 wurden die unterschiedlichen Entscheidungsmechanismen der Unterteilungsbäume evaluiert, wobei die Entscheidungsstrategie mit maximaler Konfidenz der Blattklassifikatoren das beste Resultat erzielte. Die mittleren Laufzeiten der Klassifikatoren lagen alle im Echtzeitbereich, einige unter 20 ms. Allerdings lagen die Klassifikationsleistungen leicht unter den bisher ermittelten. Der Grund dafür ist die reduzierte Anzahl der Muster, welche für die entsprechenden Blattklassifikatoren verwendet werden. Erst mit einer starken Dezimierung der Anzahl der Einträge in den Frequenztabellen konnten funktionsfähige Klassifikatoren erzeugt werden. Trotzdem waren die Klassifikationsraten leicht unter den leistungsstärksten alternativen Klassifikatoren. Deshalb wurde das Verfahren zur konfidenzbasierten, integrierten Musterauswahl mit der Korrektur der A-priori-Wahrscheinlichkeiten mit den Unterteilungsbäumen in Abschnitt 6.4.2 kombiniert und evaluiert.

Mit diesem Verfahren konnten die Klassifikationsraten um durchschnittlich 5% auf 97.5%@1FPPF gesteigert werden, allerdings mit einem erhöhten Rechenaufwand. Die Detektionsrate liegt mit diesem Verfahren über der mit dem AdaBoost-Verfahren erzeugten Kaskade bei gleichzeitig geringerer mittlerer Laufzeit.

In Abschnitt 6.4.3 wurde das konfidenzbasierte Gradientenboosting evaluiert. Da sich eine hohe Anzahl an Basisklassifikatoren in der Versuchsreihe mit den Merkmalen stark auf die mittlere Laufzeit ausgewirkt haben, wurde eine geringe maximale Anzahl von 8 bzw. 16 Basisklassifikatoren verwendet. Ausserdem wurde die konfidenzbasierte Musterauswahl mit der Gewichtung 0.9 genutzt. Mit dieser Kombination wurde eines der besten Resultate aus allen Versuchen mit einer Detektionsrate von 98%@1FPPF bei einer durchschnittlichen Laufzeit von 65 ms pro Bild erreicht. Die minimale Detektionsrate beträgt 96%@0.125FPPF und ist somit höher als die Detektionsraten der anderen Klassifikatoren in diesem Falsch-Positiv-Bereich. Aus den Versuchen wurde weiterhin ersichtlich, dass eine höhere Anzahl an Einträgen in den Frequenztabellen sehr schnell zu einer Verschlechterung der Detektionsrate des Kaskadenklassifikators führt, denn durch die inhomogene Gewichtung der Muster zum Training der Basisklassifikatoren resultieren zu schwach repräsentative Statistiken in den Entscheidungsfunktionen, die wiederum eine schlechte Generalisierungsfähigkeit des Ensembleklassifikators zur Folge haben.

In Abschnitt 6.5 wurden abschliessend die wichtigsten Aspekte und Konzepte des Softwaredesigns erläutert, mit denen es erst möglich war, die Software zu erweitern ohne die bestehenden Teile zu beeinflussen und damit wissenschaftlich zu arbeiten, indem neue Algorithmen implementiert und getestet wurden. Gleichzeitig konnten echtzeitfähige Klassifikatoren für den industriellen Realwelteinsatz erzeugt werden.

Kapitel 7

Ausblick

7.1 Gesamtzusammenfassung

Da in den entsprechenden Kapiteln schon jeweils eine Zusammenfassung verfasst wurde, wird hier eine kurze Gesamtzusammenfassung gegeben.

Im Kapitel 1 wurde die Funktionsweise der kamerabasierten Verkehrszeichenerkennung im Fahrzeug und die Anwendung auf die amerikanischen Geschwindigkeitsbegrenzungen beschrieben. In Kapitel 2 wurde der aktuelle Stand der Technik beschrieben und die Schwerpunkte dieser Arbeit herausgestellt. Außerdem wurde der für die in dieser Arbeit beschriebenen Verfahren entwickelte Bayes'sche Ansatz zur Kaskadenklassifikation beschrieben und mathematisch erläutert. Am Ende des Kapitels stand ein Überblick über die weiteren Kapitel. In Kapitel 3 wurde die merkmalsbasierte Klassifikation aus Bildinformationen einzelner Klassifikatoren beschrieben, indem die Berechnung unterschiedlicher Merkmalstypen sowie geometrischer Merkmalsbereiche erklärt wurde. Daraufhin folgten die Merkmalstransformation und die Verfahren zur Klassenzuordnung. Die entwickelten Algorithmen basieren auf Frequenztabellen. Mit dem vorgestellten Verfahren lässt sich ein definiertes Signifikanzniveau in den Entscheidungen auf dem Trainingsdatensatz der Stufenklassifikatoren einstellen, was insbesondere bei kaskadierten Klassifikationssystemen eine wichtige Methode darstellt um den Klassifikationsfehler mit einer höheren Anzahl an Stufenklassifikatoren gering zu halten. In Kapitel 4 wurde das statistische Verfahren zur Ressourcenoptimierung beschrieben, sowie das auflösungsspezifische Kaskadentraining und die auflösungsspezifische Detektion. Mit diesem Optimierungsverfahren kön-

nen in ihrer Berechnungskomplexität unterschiedliche Klassifikatoren durch ihre "erwarteten Fehloperationen" verglichen werden. Durch eine zusätzliche Sortierung lassen sich somit für die Kaskadenklassifikatoren zu Beginn effiziente Stufenklassifikatoren bezüglich ihrer Berechnungskomplexität wählen und mit steigender Anzahl an Stufen eine graduelle Verschiebung der Optimierungsstrategie auf Effizienz bezüglich der Klassifikationsleistung erreichen. Mit dieser Strategie lassen sich echtzeitfähige Detektoren berechnen, die eine drastisch reduzierte mittlere Laufzeit bei identischen Detektionsraten im Vergleich zu den Optimierungsstrategien die nur auf dem Klassifikationsfehler beruhen, sowie den AdaBoost-Kaskaden, aufweisen. Außerdem hat das hier neu vorgestellte Verfahren zum auflösungsspezifischen Kaskadentraining und zur entsprechenden Kaskadendetektion, das auf den Filtereigenschaften der Merkmale beruht, wodurch mehrere Vorteile im Vergleich zu Laplace-Pyramiden entstehen, eine weitere Reduktion der mittleren Laufzeiten bei der Ausführung, sowie beim Training der Klassifikatoren, ermöglicht. Hiermit konnte die Vollbildsuche auf mittlere Laufzeiten im Bereich von 20 ms gedrückt werden. Dies konnte in umfangreichen experimentellen Versuchen gezeigt werden. Somit wurden im Rahmen dieser Arbeit energieeffiziente Klassifikatoren erzeugt, wodurch sich der Rechenaufwand in mobilen Geräten stark reduzieren läßt. Prinzipiell lässt sich das entwickelte Optimierungsverfahren auf beliebige Eigenschaften einer Funktion anwenden, für die ein Wahrscheinlichkeitsmodell aufgestellt werden kann.

Der Datensatz, auf dem alle Experimente durchgeführt wurden, sowie der Algorithmus zu den Auswertungen wurden ausführlich beschrieben. Daraufhin wurden zahlreiche Versuche zu den unterschiedlichen Algorithmen und Parametern diskutiert. Die Verfahren des Kapitels und die wichtigsten Ergebnisse der Versuche wurden abschließend zusammengefasst. In Kapitel 5 wurden Verfahren zur Konfidenzwertberechnung beschrieben, die es ermöglichen die Einzelkonfidenzen der Stufenklassifikatoren zu fusionieren. Außerdem wurde ein neues, im Rahmen dieser Arbeit entwickeltes Verfahren zur integrierten, konfidenzbasierten Musterauswahl innerhalb des Kaskadentrainings beschrieben. Mit diesem Verfahren konnten die Detektionsraten um 5 % im Mittel gesteigert werden. Ein neues, in das Kaskadentraining integriertes, teilüberwachtes Kaskadentraining, das gleichzeitig eine konfidenzbasierte Musterauswahl ermöglicht, wurde ebenfalls präsentiert. Bei diesem Verfahren besitzt der Klassifikator die Möglichkeit Hintergrundbeispiele, bei denen eine sehr hohe Konfidenz zur Objektklasse erreicht wird, in die Objektklasse zu übernehmen und als Objekte mitzutrainieren. Es konnte gezeigt werden,

dass bereits mit der Hälfte der verfügbaren Beispiele die besten Klassifikationsraten erzielt wurden und die vollständig überwachten Klassifikatoren übertroffen wurden. Insbesondere in den höheren Stufen konnte ein Absinken der Klassifikationsrate mit dem Verfahren vermieden werden. Auch die mit dem teilüberwachten Trainingsverfahren erstellten Klassifikatoren lagen bei der Ausführung im Echtzeitbereich.

Abschließend wurden die Verfahren und die wichtigsten Ergebnisse der Experimente zusammengefasst. In diesem Kapitel wurden Ensemble- und Aggregationsverfahren vorgestellt, die im Gegensatz zu Kapitel 3 nicht auf Merkmalen basieren, sondern deren Basiseinheiten beliebige Klassifikatoren sind. Zu Beginn wurde das AdaBoost-Verfahren beschrieben, sowie ähnliche Boosting-Verfahren und deren Unterschiede. Dann wurde das konfidenzbasierte Gradientenboosting mit multidimensionaler Optimierung beschrieben. Mit diesem Verfahren konnten bei hohen Detektionsraten von 98 % die besten mittleren Laufzeiten erzielt werden. Als nächstes wurden die ressourcenoptimierten, konfidenzbasierten Unterteilungsbäume beschrieben. Dieses Verfahren besitzt den Vorteil, dass bei der Ausführung die Klassifikationseigenschaften zwischen energiesparender, schneller Klassifikation mit größerem Klassifikationsfehler und langsamer Klassifikation mit geringem Fehler adaptiv angepasst werden können. Die Verfahren wurden in einer experimentellen Evaluierung eingehend getestet und die Ergebnisse diskutiert. Abschließend wurde die Softwarearchitektur beschrieben, mit der es möglich war echtzeitfähige Klassifikatoren für den industriellen Einsatz zu erzeugen und gleichzeitig durch die hochgradig modulare Struktur die unterschiedlichen Teilalgorithmen und Verfahren zu testen und wissenschaftlich zu beurteilen.

7.2 Ausblick

Die projektbedingte Hauptanwendung bei dieser Arbeit war die Detektion amerikanischer Geschwindigkeitsbegrenzungen für die Echtzeitanwendung in mobilen Systemen. Es wurden jedoch ausschließlich vom Objekttyp unabhängige, allgemein anwendbare Verfahren verwendet. Deshalb wäre es interessant, das Verfahren auf weitere Detektionsaufgaben anzuwenden um die Energieeffizienz zu steigern.

Um die amerikanischen Verkehrsschilder vollständig zu verstehen müsste ein lesendes System entwickelt werden, das alle Wörter auf dem Schild erkennt, deren Bedeutung dann nachgeschlagen werden kann. Mit einem sol-

chen System könnten auch Straßenschilder erkannt werden und in das Navigationssystem integriert werden. Ausserdem könnten relevante Informationen wie die Entfernungen zur nächsten Tank- und Raststätte mit angezeigt werden. Aus Werbetafeln könnten Informationen über naheliegende Restaurants oder Werkstätten extrahiert und dem Fahrer bei Bedarf zur Verfügung gestellt werden.

Für die Fahrerassistenzsysteme exisiteren weitere zahllose Anwendungsbeispiele. Aus dem Fahrzeug schauende Systeme erkennen unterschiedliche Objekte wie z.B. Fußgänger, Radfahrer, Motorradfahrer, Automobilrückfronten und Fahrbahnmarkierungen [121, 81]. Zur Vermeidung und Verminderung von Wildunfällen könnten unterschiedliche Wildtiere erkannt werden. Ein weiteres interessantes Anwendungsgebiet sind Kreuzungen. Dort müssen unterschiedliche Ampeln und Abbiegespuren sowie die Rückfronten des vorrausfahrenden Fahrzeugs erkannt werden. Auch in das Fahrzeug schauende Systeme können in Betracht gezogen werden. So kann durch eine Erkennung der Kopfausrichtung sowie der Blickrichtung des Fahrers ermittelt werden, ob eine Gefahrensituation erkannt wird und ob der Fahrer abgelenkt ist. Auch Überholvorgänge mit Schulterblick könnten somit analysiert werden und bei Fehlverhalten entsprechend gewarnt werden.

Für die teilüberwachten Verfahren wäre es weiterhin interessant, eine Variante mit dem konfidenzbasierten Gradientenboosting zu testen, da mit diesem Verfahren sehr gute Klassifikationsergebnisse erzielt wurden und es sich anbietet, bei der konfidenzbasierten Gewichtung der Beispiele auch eine Migration aus der Hintergrundklasse in die Objektklasse zu erlauben. Dies könnte auch für den umgekehrten Fall durchgeführt werden, wobei Muster aus der Objektklasse mit geringer Konfidenz entweder verworfen werden oder in die Hintergrundklasse abwandern.

Eine weitere interessante Aufgabe im Rahmen des teilüberwachten Lernen ist die Adaption eines Kaskadenklassifikators auf eine veränderte Detektionsaufgabe. So könnte zum Beispiel erforscht werden, ob und mit welchen Verfahren es möglich ist, einen auf eine spezifische Aufgabe trainierten Klassifikator auf eine umfassendere Aufgabe zu erweitern und damit zu generalisieren.

Literaturverzeichnis

[1] Convention on road signs and signals of 1968. United Nations Economic Commission for Europe Geneva, Switzerland (1968)

[2] California manual on uniform traffic control devices for streets and highways. State of California, Dept. Transp, Sacramento, CA (2006)

[3] Alefs, B., Eschemann, G., Ramoser, H., Beleznai, C.: Road sign detection from edge orientation histograms. In: Intelligent Vehicles Symposium, 2007 IEEE. p. 993–998 (2007)

[4] Alessandro, S.J.K., Magnani, A., Boyd, S.P.: Robust fisher discriminant analysis. In: Advances in Neural Information Processing Systems. pp. 659–666 (2006)

[5] Babaud, J., Witkin, A.P., Baudin, M., Duda, R.O.: Uniqueness of the gaussian kernel for scale-space filtering. IEEE Trans. Pattern Anal. Mach. Intell. 8(1), 26–33 (1986)

[6] Bahlmann, C., Zhu, Y., Ramesh, V., Pellkofer, M., Koehler, T.: A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In: IEEE Intelligent Vehicles Symposium (IV 2005) (2005)

[7] Ballard, D.H.: Readings in computer vision: issues, problems, principles, and paradigms. chap. Generalizing the hough transform to detect arbitrary shapes, pp. 714–725. Morgan Kaufmann Publishers Inc. (1987)

[8] Barnes, N., Zelinsky, A.: Real-time radial symmetry for speed sign detection. pp. 566–571 (2004)

[9] Baró, X., Escalera, S., Vitrià, J., Pujol, O., Radeva, P.: Traffic sign recognition using evolutionary adaboost detection and forest-ECOC classification. IEEE Transactions on Intelligent Transportation Systems 10(1), 113–126 (2009)

[10] Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Mach. Learn. 36(1-2), 105–139 (Jul 1999)

[11] Bennett, K.P., Demiriz, A.: Semi-supervised support vector machines. In: Advances in Neural Information Processing Systems. pp. 368–374 (1998)

[12] Bennett, K.P., Demiriz, A., Maclin, R.: Exploiting unlabeled data in ensemble methods. In: Proceedings of the eighth ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining. pp. 289–296. KDD '02 (2002)

[13] Bigün, J., Granlund, G.H.: Optimal orientation detection of linear symmetry. In: IEEE First International Conference on Computer Vision. pp. 433–438 (1987)

[14] Blum, A., Chawla, S.: Learning from labeled and unlabeled data using graph mincuts. In: Proceedings of the Eighteenth Int. Conference on Machine Learning. pp. 19–26. ICML '01 (2001)

[15] Breiman, L.: Bagging predictors. In: Machine Learning. pp. 123–140 (1996)

[16] Breiman, L.: Stacked regressions. Machine Learning 24(1), 49–64 (1996)

[17] Breiman, L.: Arcing classifiers (1998)

[18] Breiman, L.: Random forests. Mach. Learn. 45(1), 5–32 (Oct 2001)

[19] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees (1984)

[20] Brkic, K.: An overview of traffic sign detection methods

[21] Brodley, C.E., Utgoff, P.E.: Multivariate decision trees. Machine Learning 19(1), 45–77 (Apr 1995)

[22] Burger, W., Burge, M.J.: Digitale Bildverarbeitung : Eine Einführung mit Java und ImageJ. Springer-Verlag (2005)

[23] Cai, D., He, X., Han, J.: Semi-supervised discriminant analysis. In: in Proc. of the IEEE Int'l Conf. on Comp. Vision (ICCV) (2007)

[24] Cardelli, L., Wegner, P.: On understanding types, data abstraction, and polymorphism. ACM Comput. Surv. 17(4), 471–523 (Dec 1985)

[25] Clarke, B.: Comparing bayes model averaging and stacking when model approximation error cannot be ignored. J. Mach. Learn. Res. 4, 683–712 (Dec 2003)

[26] Cohen, I., Cozman, F.G., Sebe, N., Cirelo, M.C., Huang, T.S.: Semi-supervised learning of classifiers: Theory, algorithms and their application to human-computer interaction. IEEE Transactions on Pattern Analysis and Machine Intelligence 26, 1553–1567 (2004)

[27] Cohn, D.A.: Neural network exploration using optimal experiment design. In: Neural Networks. pp. 679–686. Morgan Kaufmann (1994)

[28] Creusen, I.M., Wijnhoven, R.G.J., Herbschleb, E., de With, P.H.N.: Color exploitation in HOG-based traffic sign detection. In: ICIP. pp. 2669–2672. IEEE (2010)

[29] Cui, T., Grumpe, A., Hillebrand, M., Kreßel, U., Kummert, F., Wöhler, C.: Analytically tractable sample-specific confidence measures for semi-supervised learning. Proc. Workshop Computational Intelligence pp. 171–186 (2011)

[30] Cun, L., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Handwritten digit recognition with a back-propagation network. In: Advances in Neural Information Processing Systems. pp. 396–404. Morgan Kaufmann (1990)

[31] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: In CVPR. pp. 886–893 (2005)

[32] Dalal, N., Triggs, B., Schmid, C.: Human detection using oriented histograms of flow and appearance. In: Proceedings of the 9th European conference on Computer Vision - Volume Part II. pp. 428–441. ECCV'06, Springer-Verlag (2006)

[33] Dillon, J.V., Balasubramanian, K., Lebanon, G.: Asymptotic analysis of generative semi-supervised learning. CoRR (2010)

[34] Dollar, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: A benchmark. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009. CVPR. pp. 304–311 (Jun 2009)

[35] Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley (2001)

[36] Enzweiler, M., Gavrila, D.M.: Monocular pedestrian detection: Survey and experiments. IEEE Transactions on Pattern Analysis and Machine Intelligence 31, 2179–2195 (2009)

[37] Fisher, R.A.: The statistical utilization of multiple measurements. Annals of Eugenics pp. 376–386 (1938)

[38] Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Computational Learning Theory: Eurocolt 95 pp. 23–37 (1995)

[39] Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Proceedings of the Second European Conference on Computational Learning Theory. pp. 23–37 (1995)

[40] Freund, Y., Schapire, R.E.: A short introduction to boosting. In: In Proceedings of the Sixteenth Int. Joint Conference on Artificial Intelligence. pp. 1401–1406 (1999)

[41] Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Annals of Statistics 28, 2000 (1998)

[42] Friedman, J.H.: Stochastic gradient boosting. Computational Statistics and Data Analysis 38, 367–378 (1999)

[43] Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics 29, 1189–1232 (2000)

[44] Fu, M.Y., Huang, Y.S.: A survey of traffic sign recognition. ICWAPR pp. 119 – 124 (July 2010)

[45] Fukushima, K., Wake, N.: Handwritten alphanumeric character recognition by the neocognitron. IEEE Transactions on Neural Networks 2(3), 355–365 (May 1991)

[46] Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36, 193–202 (1980)

[47] Fukushima, K., Yoshimoto, K.: Self-organization of shift-invariant receptive fields. In: IWANN (1). Lecture Notes in Computer Science, vol. 1606, pp. 806–815. Springer (1999)

[48] Gama, J.a., Brazdil, P.: Cascade generalization. Mach. Learn. 41(3), 315–343 (Dec 2000)

[49] Gao, X., Podladchikova, L., Shaposhnikov, D., Hong, K., Shevtsova, N.: Recognition of traffic signs based on their colour and shape features extracted using human vision models. Journal of Visual Communication and Image Representation 17(4), 675 – 685 (2006)

[50] Garrido, M.Á.G., Ocaña, M., Llorca, D.F., Sotelo, M.Á., Llamazares, Á., Arroyo, E.: Robust traffic signs detection by means of vision and V2I communications. In: 14th International IEEE Annual Conference on Intelligent Transportation Systems (IEEE ITSC 2011) (oct 2011)

[51] Goldman, S., Zhou, Y.: Enhancing supervised learning with unlabeled data. In: In Proceedings of the 17th Int. Conference on Machine Learning. pp. 327–334 (2000)

[52] Gómez-Moreno, H., Maldonado-Bascón, S., Gil-Jiménez, P., Lafuente-Arroyo, S.: Goal evaluation of segmentation algorithms for traffic sign recognition. Trans. Intell. Transport. Sys. 11(4), 917–930 (Dec 2010)

[53] González, Á., Garrido, M.Á.G., Llorca, D.F., Gavilán, M., Fernández, J.P., Alcantarilla, P.F., Parra, I., Herranz, F., Bergasa, L.M., Sotelo, M.Á., Revenga, P.: Automatic traffic signs and panels inspection system using computer vision. Intelligent Transportation Systems, IEEE Transactions on 12(2), 485 –499 (june 2011)

[54] Haar, A.: Zur Theorie der orthogonalen Funktionensysteme. Mathematische Annalen 69(3), 331–371 (Sep 1910)

[55] Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings of the 4th Alvey Vision Conference. pp. 147–151 (1988)

[56] Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning. Springer (2003)

[57] Haußecker, H., Jähne, B.: A tensor approach for local structure analysis in multi-dimensional images. In: Interdisciplinary Center for Scientific Computing, University of Heidelberg (1998). (1998)

[58] Hechenbichler, K.: Ensemble-Techniken und ordinale Klassifikation (2005)

[59] Ho, T.K.: Random decision forests. In: Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1. ICDAR '95, IEEE Computer Society (1995)

[60] Houben, S.: A single target voting scheme for traffic sign detection. In: Intelligent Vehicles Symposium. pp. 124–129. IEEE (2011)

[61] Huang, J., Ertekin, S., Song, Y., Zha, H., Giles, C.L.: Efficient multiclass boosting classification with active learning. In: SDM. SIAM (2007)

[62] IBM: Blue gene (2013), `http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/bluegene/`, [Online; accessed 1-October-2013]

[63] Illingworth, J., Kittler, J.: A survey of the hough transform. Computer Vision, Graphics, and Image Processing 44(1), 87 – 116 (1988)

[64] Jähne, B.: Spatio-Temporal Image Processing: Theory and Scientific Applications. Springer-Verlag New York, Inc. (1993)

[65] Jähne, B.: Digitale Bildverarbeitung. Springer (2001)

[66] Jain, V., Learned-Miller, E.: Online domain adaptation of a pre-trained cascade of classifiers. In: Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 577–584 (2011)

[67] Keller, C.G., Sprunk, C., Bahlmann, C., Giebel, J., Baratoff, G.: Real-time recognition of U.S. speed signs. In: IEEE Intelligent Vehicles Symposium. pp. 518–523 (June 2008)

[68] Kerji, V.K.: Abstract factory and singleton design patterns to create decorator pattern objects in web application 1(5) (2011)

[69] Koch, K.R.: Einführung in die Bayes-Statistik. Springer (2000)

[70] Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience (2004)

[71] Lafuente-Arroyo, S., Salcedo-Sanz, S., Maldonado-Bascón, S., Portilla-Figueras, J.A., López-Sastre, R.J.: A decision support system for the automatic management of keep-clear signs based on support vector machines and geographic information systems. Expert Syst. Appl. 37(1), 767–773 (2010)

[72] Leistner, C., Grabner, H., Bischof, H.: Semi-supervised boosting using visual similarity learning. In: IN: PROC. CVPR (2008)

[73] Lienhart, R., Kuranov, E., Pisarevsky, V.: Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In: In DAGM 25th Pattern Recognition Symposium. pp. 297–304 (2003)

[74] Lindeberg, T.: Scale-space theory: A basic tool for analysing structures at different scales. Journal of Applied Statistics pp. 224–270 (1994)

[75] Lindeberg, T.: Scale-Space Theory in Computer Vision. Kluwer Academic Publishers (1994)

[76] Loy, G., Zelinsky, A.: Fast radial symmetry for detecting points of interest. IEEE Trans. Pattern Anal. Mach. Intell. 25(8), 959–973 (2003)

[77] Loy, G.B., Barnes, N.M.: Fast shape-based road sign detection for a driver assistance system. In: IROS. pp. 70–75. IEEE (2004)

[78] MacKay, D.J.: Information-based objective functions for active data selection. Neural Computation 4, 590–604 (1992)

[79] Maldonado-Bascón, S., Lafuente-Arroyo, S., Gil-Jiménez, P., Gómez-Moreno, H., López-Ferreras, F.: Road-sign detection and recognition based on support vector machines. IEEE Transactions on Intelligent Transportation Systems 8(2), 264–278 (2007)

[80] Mallapragada, P.K., Jin, R., Jain, A.K., Liu, Y.: SemiBoost: Boosting for semi-supervised learning. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(11), 2000–2014 (2009)

[81] Maurer, M., Stiller, C. (eds.): Fahrerassistenzsysteme mit maschineller Wahrnehmung. Springer (2005)

[82] McCallum, A., Nigam, K.: Employing EM and pool-based active learning for text classification. In: Proceedings of the Fifteenth International Conference on Machine Learning. pp. 350–358. ICML '98 (1998)

[83] Mccane, B., Novins, K., Albert, M.: Optimizing cascade classifiers (2004)

[84] Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. Int. J. Comput. Vision 60(1), 63–86 (2004)

[85] Mogelmose, A., Trivedi, M.M., Moeslund, T.B.: Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. IEEE Transactions on Intelligent Transportation Systems 13(4), 1484–1497 (2012)

[86] Moutarde, F., Bargeton, A., Herbin, A., Chanussot, L.: Robust on-vehicle real-time visual detection of american and european speed limit signs, with a modular traffic signs recognition system. Proc. IEEE Intell. Veh. Symp. p. 1122–1126 (2007)

[87] Nguwi, Y.Y., Kouzani, A.Z.: Detection and classification of road signs in natural environments. Neural Comput. Appl. 17(3), 265–289 (Apr 2008)

[88] Nico, T., Kai-Uwe, P., Peter, H., Günter, H.: Hough transformation for image processing in eye tracking recording. Biomed Tech (Berl) 47 Suppl 1 Pt 2, 636–8 (2002)

[89] North, D.O.: An analysis of the factors which determine signal/noise discrimination in pulsed-carrier systems. Proceedings of the IEEE 51(7), 1016–1027 (1963)

[90] Ohm, J.R., Lüke, H.D.: Signalübertragung Grundlagen der digitalen und analogen Nachrichtenübertragungssysteme. Springer, Berlin; Heidelberg; New York (2007)

[91] Overett, G., Petersson, L.: Large scale sign detection using HOG feature variants. Intelligent Vehicles Symposium (IEEE IV) pp. 1–6 (2011)

[92] Pettersson, N., Petersson, L., Andersson, L.: The histogram feature – a resource-efficient weak classifier. In: Intelligent Vehicles Symposium, 2008 IEEE (2008)

[93] Ponce, J., Berg, T.L., Everingham, M., Forsyth, D.A., Hebert, M., Lazebnik, S., Marszałek, M., Schmid, C., Russell, B.C., Torralba, A., Williams, C.K.I., Zhang, J., Zisserman, A.: Dataset issues in object recognition. In: Ponce, J., Hebert, M., Schmid, C., Zisserman, A. (eds.) Toward Category-Level Object Recognition, LNCS, vol. 4170, pp. 29–48. Springer (2006)

[94] Pratama, S.F.: Computationally inexpensive sequential forward floating selection for acquiring significant features for authorship invariantceness in writer identification. International Journal of New Computer Architectures and their Applications (IJNCAA) 3(1) (2011)

[95] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes 3rd Edition: The Art of Scientific Computing. Cambridge University Press, New York, NY, USA, 3 edn. (2007)

[96] Prisacariu, V., Timofte, R., Zimmermann, K., Reid, I., Gool, L.V.: Integrating object detection with 3D tracking towards a better driver assistance system. In: Pattern Recognition (ICPR), 2010 20th International Conference on. pp. 3344 –3347 (aug 2010)

[97] Quinlan, J.R.: Induction of decision trees. Mach. Learn. 1(1), 81–106 (Mar 1986)

[98] Quinlan, J.R.: C4.5: programs for machine learning (1993)

[99] Rokach, L.: Pattern Classification Using Ensemble Methods. World Scientific Publishing Co., Inc., River Edge, NJ, USA (2010)

[100] Schapire, R.E.: The strength of weak learnability. Mach. Learn. 5(2), 197–227 (Jul 1990)

[101] Schürmann, J.: Pattern Classification, A Unified View of Statistical and Neural Approaches. Wiley (1996)

[102] Settles, B.: Active learning literature survey. Tech. rep. (2010)

[103] Shi, J., Tomasi, C.: Good features to track. In: 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94). pp. 593–600. IEEE (1994)

[104] Smith, S.M., Brady, J.: SUSAN - A new approach to low level image processing. International Journal of Computer Vision 23, 45–78 (1995)

[105] Smyth, P., Wolpert, D.: Linearly combining density estimators via stacking. In: Machine Learning (1999)

[106] Sochman, J., Matas, J.: WaldBoost - Learning for time constrained sequential detection. In: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02. pp. 150–156. CVPR '05, IEEE Computer Society, Washington, DC, USA (2005)

[107] Staudenmaier, A., Klauck, U., Kreßel, U., Lindner, F., Wöhler, C.: A real-time object detection framework using resource optimized cascaded perceptron classifiers and its application to US speed limits. In: F. Hoffmann, E. Hüllermeier: 20. Workshop Computational Intelligence. pp. 204–219 (2010)

[108] Staudenmaier, A., Klauck, U., Kreßel, U., Lindner, F., Wöhler, C.: Confidence measurements for adaptive bayes decision classifier cascades and their application to US speed limit detection. In: A. Pinz et al. (Eds.): DAGM/OAGM 2012, LNCS 7476. pp. 478–487 (2012)

[109] Staudenmaier, A., Klauck, U., Lindner, F., Kreßel, U., Wöhler, C.: Resource optimized cascaded perceptron classifiers using structure tensor features for US speed limit detection. 8th Int. Workshop on Intelligent Transportation (WIT 2011) (2011)

[110] Subramanya, A., Petrov, S., Pereira, F.C.N.: Efficient graph-based semi-supervised learning of structured tagging models. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 167–176 (2010)

[111] Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. J. Mach. Learn. Res. 2, 45–66 (Mar 2002)

[112] Toth, N.: On classification confidence and ranking using decision trees. Intelligent Engineering Systems pp. 133–138 (2007)

[113] Tsatsanis, M.K., Giannakis, G.B.: Object and texture classification using higher order statistics. IEEE Trans. Pattern Anal. Mach. Intell. 14(7), 733–750 (1992)

[114] Viola, P., Jones, M.: Robust real-time object detection. International Journal of Computer Vision 57(2), 137–154 (2002)

[115] Viola, P., Jones, M.: Robust real-time face detection. Int. Journal of Computer Vision 57, 137–154 (2004)

[116] Viola, P.A., Jones, M.J.: Rapid object detection using a boosted cascade of simple features. In: CVPR (1)'01. pp. 511–518 (2001)

[117] Wang, F., Yuan, C., Xu, X., van Beek, P.: Supervised and semi-supervised online boosting tree for industrial machine vision application. In: Proceedings of the Fifth Int. Workshop on Knowledge Discovery from Sensor Data. pp. 43–51 (2011)

[118] Whitney, A.W.: A direct method of nonparametric measurement selection. IEEE Trans. Comput. 20(9), 1100–1103 (1971)

[119] Wikipedia: Fahrerassistenzsystem (2013), `http://de.wikipedia.org/wiki/Fahrerassistenzsystem`, [Online; accessed 1-October-2013]

[120] Wikipedia: Film noir — Wikipedia, the free encyclopedia (2013), `http://en.wikipedia.org/wiki/File:BigComboTrailer.jpg`, [Online; accessed 22-July-2013]

[121] Winner, H.: Handbuch Fahrerassistenzsysteme Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort. Vieweg + Teubner (2012)

[122] Witting, H.: Mathematische Statistik: Parametrische Verfahren bei festem Stichprobenumfang. Teubner B.G. GmbH (1985)

[123] Wolpert, D.H.: Stacked generalization. Neural Networks 5, 241–259 (1992)

[124] Wu, Y., Huang, T.S., Toyama, K.: Self-supervised learning for object recognition based on kernel discriminant-em algorithm. In: on Kernel Discriminant-EM Algorithm. ICCV. pp. 275–280 (2001)

[125] Wöhler, C., Anlauf, J.K.: A time delay neural network algorithm for estimating image-pattern shape and motion. Image Vision Comput. 17(3-4), 281–294 (1999)

[126] Xie, Y., Liu, L.F., Li, C.H., Qu, Y.Y.: Unifying visual saliency with HOG feature learning for traffic sign detection. In: Intelligent Vehicles Symposium, 2009 IEEE. p. 24–29 (2009)

[127] Zhang, T., Oles, F.J.: A probability analysis on the value of unlabeled data for classification problems. In: 17th Int. Conference on Machine Learning (2000)

[128] Zhu, Q., Avidan, S., chen Yeh, M., ting Cheng, K.: Fast human detection using a cascade of histograms of oriented gradients. In: In CVPR06. pp. 1491–1498 (2006)

[129] Zhu, X., Goldberg, A.B.: Introduction to Semi-Supervised Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers (2009)