Institut für Visualisierung und Interaktive Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3234

Interactive Learning

Rudolf Netzel

Studiengang:	Informatik
Prüfer:	Prof. Dr. D. Weiskopf
Betreuer:	Dipl.-Inf. B. Höferlin, Dipl.-Inf. M. Höferlin
begonnen am:	9. August 2011
beendet am:	8. Februar 2012
CR-Klassifikation:	I.2.6, H.5.2, I.2.1

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Algorithmen

1 Einleitung

Klassifikatoren kommen heutzutage in vielen Bereichen des Alltags zum Einsatz. Beispielsweise beim Frühstück. Hier schaltet sich der Wasserkocher ab, sobald die Wassertemperatur einen bestimmten Schwellwert überschritten hat. Die Automobilindustrie setzt sie in rechnergestützten Sicherheitssystemen ein und weist den Fahrer auf mögliche Gefahrensituationen hin. Beispielsweise auf einen zu geringen Abstand zu anderen Fahrzeugen oder vor Fahrzeugen im toten Winkel beim Spurwechsel.

Im Bereich der Informatik befasst sich das maschinelle Lernen mit dem Training von Klassifikatoren. Dies bedeutet, dass eine Menge von Merkmalen trainiert wird. Während des Trainings werden ihre Parameter automatisch adaptiert. Durch die Kombination der Merkmale sollten dann möglichst alle Elemente der Trainingsmenge korrekt klassifiziert werden. Für wenige Merkmale und eine kleine Menge an Beispielen könnte ein Mensch die optimalen Parameter bestimmen. Für eine größere Anzahl an Merkmalen und Beispielen ist dies jedoch keine triviale Angelegenheit.

Zu diesem Zweck ist eine Vielzahl an Algorithmen bzw. Trainingssysteme entstanden. Ihre Funktionsweisen beruhen jedoch auf den gleichen Prinzipien. Ein Klassifikator wird mit einer Menge an Beispielen trainiert. Dabei lernt er die Beispiele nicht auswendig, sondern sollte nach Beendigung des Trainings in der Lage sein zu verallgemeinern. Das heißt, der Klassifikator sollte Beispiele, die nicht in der Trainingsmenge enthalten sind, korrekt klassifizieren. Um diesen Generalisierungseffekt hervorzurufen, benötigen die Trainingsalgorithmen tausende Beispiele. Sowohl Positive als auch Negative. Dadurch ergeben sich zwei Nachteile. Zum einen nimmt das Erstellen der Trainingsmenge sehr viel Zeit in Anspruch, da jedem Beispiel manuell eine Objektklasse zugewiesen werden muss. Zum anderen ist das Training des Klassifikators sehr zeitintensiv. Dies liegt unter anderem an der großen Anzahl der benötigten Beispiele.

Um diesen Nachteilen entgegenzuwirken, können die Algorithmen bzw. Trainingssysteme um das Konzept des Active Learning erweitert werden. Dabei wird ein Klassifikator durch ein zyklisches Training verbessert. Eine Iteration besteht dabei aus zwei Teilen. Zum einen der Erweiterung der bestehenden Trainingsmenge. Zum anderen aus dem Training des Klassifikators anhand der erweiterten Menge. Beispiele werden der Menge hinzugefügt, nachdem ihnen durch den Benutzer eine Klasse zugewiesen wurde. Die Beispiele dafür werden durch das Active Learning ausgewählt. Dabei werden Beispiele mit einem hohen Informationsgehalt aus einer Menge an nicht klassifizierten Beispielen entnommen. Hat ein Beispiel einen hohen Informationsgehalt, so enthält es Aspekte, die durch den Klassifikator noch nicht richtig erkannt werden, welche jedoch für die Klassifikation relevant sind.

Durch das Active Learning sind folgende positive Effekte zu beobachten:

- Die Anzahl der Beispiele, die für das Training benötigt werden, wird verringert.
- Die Trainingsdauer wird reduziert.
- Der Aufwand der manuellen Zuweisung von Objektklassen nimmt ab.
- Die Generalisierungsfähigkeit nimmt zu.

Nachteilhaft daran ist, dass der Nutzer keinen Einfluss auf die Selektion der Beispiele hat. Zudem wird ihm nicht vermittelt, aus welchen Gründen das Beispiel ausgewählt wurde.

Bisher wird das Potenzial des Benutzers nicht vollständig genutzt. Seine Aufgabe besteht lediglich darin, Beispielen eine Objektklasse zuzuordnen. In dieser Arbeit soll daher der Übergang zum Interactive Learning realisiert werden. Dabei wird die Grundidee des Active Learning um eine interaktive Schnittstelle für Aktionsmöglichkeiten durch den Benutzer erweitert. Er sollte mit diesen in der Lage sein Expertenwissen in den Trainingsprozess einfließen zu lassen, um die Güte des Klassifikators zu verbessern und das Training zu beschleunigen. Das Expertenwissen setzt sich aus allgemeinem Vorwissen und dem Verständnis über das Verhalten des Klassifikators, während des Trainings, zusammen. Daher muss ein System, dass das Interactive Learning ermöglicht, folgende Anforderungen erfüllen:

- Es muss Wissen über das Verhalten des Klassifikators vermitteln.
- Trainingsmengen müssen schnell und einfach gebildet werden können.
- Das Auffinden von Beispielen, die für das Training relevant sind, muss möglich sein.
- Möglichkeiten zur Adaption des Klassifikators müssen vorhanden sein.

Gliederung

Diese Arbeit besteht aus fünf weiteren Kapiteln und ist wie folgt gegliedert:

Kapitel 2 – Grundlagen beschreibt Verfahren, Methoden und Algorithmen, welche im Verlauf dieser Arbeit zum Einsatz kommen.

Kapitel 3 - Themenbezogen Arbeiten erörtert einige Rahmenwerke und Methoden, die Konzepte des Active Learning nutzen oder dessen Ziele umsetzten.

Kapitel 4 – Systembeschreibung erläutert die Struktur des Systems, sowie die Funktion der entstandenen Komponenten und deren Verknüpfung.

Kapitel 5 – Anwendungsbeispiel beschreibt mehrere Iterationsschritte beim Training eines Kaskadenklassifikators. Dabei kommen die vorgestellten Komponenten und deren Funktionen zum Einsatz.

Kapitel 6 - Zusammenfassung fasst die Ergebnisse dieser Arbeit zusammen und zeigt Möglichkeiten zur Verbesserung und Erweiterung des Systems auf.

2 Grundlagen

In diesem Kapitel werden die Grundlagen der Verfahren, Methoden und Algorithmen vorgestellt, welche im Verlauf dieser Arbeit zum Einsatz gekommen sind. Zuerst wird das Konzept des Active Learning erläutert, gefolgt von einem Überblick einiger Lernalgorithmen. Zuletzt wird die Funktionsweise des Algorithmus beschrieben, der in dieser Arbeit verwendet wurde, um Bilder nach ihrer Ähnlichkeit anzuordnen.

2.1 Active Learning

In dem Bericht „Active Learning Literature Survey“ [Set09] von Burr Settles werden grundlegende Elemente, Aufgaben und weiterführende Arbeiten in Bezug zu Active Learning beschrieben und analysiert.

Im Folgenden werden einige dort beschriebene Schlüsselaspekte näher dargestellt.

2.1.1 Definition und Ziele

Der Begriff Active Learning beschreibt das Vorgehen beim Training eines Modells sowie die dafür verwendeten Methoden. Bestehende Verfahren können um dieses Konzept erweitert werden. Das Ziel dabei ist, die Trainingszeit als auch die benötigte Menge an Trainingsbeispielen zu reduzieren. Dabei sollte die Güte eines Modells, das mit dieser Methode trainiert wurde, mit der Güte eines auf konventionelle Art und Weise entstandenen Modells vergleichbar sein.

2.1.2 Ablauf

Um diese Ziele zu erreichen, ist die zugrunde liegende Idee sehr einfach zu formulieren. Das Modell wird in jedem Trainingsschritt, nur mithilfe von Beispielen trainiert, mit denen eine Verbesserung des Modells zu erwarten ist. Dadurch ist es erforderlich Trainingsbeispiele zu selektieren, welche das Modell noch nicht richtig erkennt. Beispiele, die bereits dem Modell entsprechen, werden beim Active Learning somit nicht berücksichtigt.

Wie bereits angedeutet ist diese Art des Trainings ein iterativer Prozess. Zu Beginn existieren zwei Mengen, L und U, an Beispielen. Allen Elementen aus L ist dabei eine Objektklasse zugeordnet, während dies bei denen aus U nicht der Fall ist. Das zugrunde liegende Modell wird dann mit den Beispielen aus L trainiert. Hiernach erfolgt die Selektion der Elemente

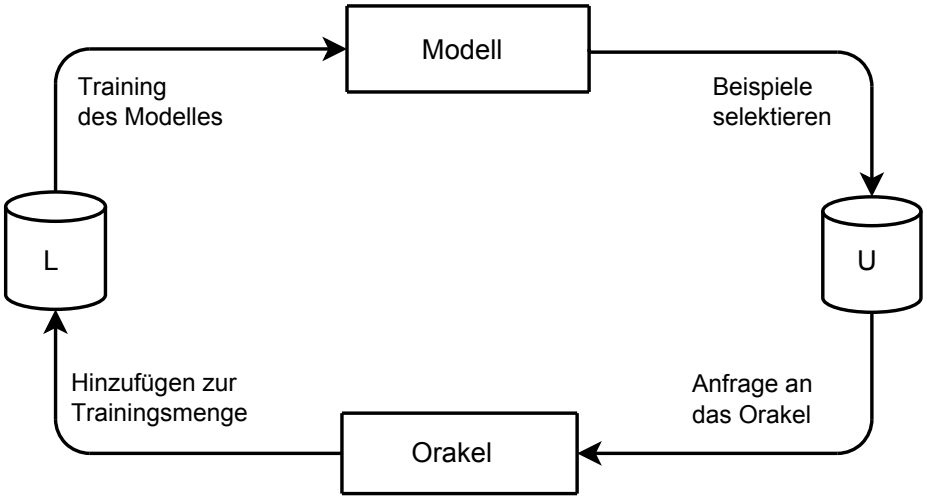


Abbildung 2.1: Abgebildet ist der Ablauf des Active Learning. *L* entspricht der Menge an Beispielen, die einer Klasse zugeordnet wurden. *U* enthält die Elemente, die noch keine Klasse besitzen.

aus *U*, durch die, nach einem erneuten Training, eine Verbesserung des Modells zu erwarten ist. Jedem Element wird durch ein Orakel eine Objektklasse zugewiesen. In der Literatur wird dieser Schritt mit einer sogenannten query-function durchgeführt. Bei dem Orakel handelt es sich entweder um einen Menschen oder ein anderes System, das in der Lage ist den Objekten eine Klasse zuzuweisen. Anschließend werden die Elemente zu *L* hinzugefügt und das Modell erneut trainiert. Dieser Ablauf wiederholt sich nun solange, bis das Lernziel erfüllt ist. Abbildung 2.1 stellt dies dar. Dabei wird ein weiterer positiver Effekt des Active Learning deutlich. Bei diesem handelt es sich um die Zeitersparnis bei der Zuweisung der Objektklasse für jedes Beispiel. Zu Beginn ist hier ein verhältnismäßig kleiner Pool *L* nötig. Bei herkömmlichen Trainingsmethoden benötigt man tausende von initialen Trainingsbeispielen. Anhand des Trainings zur Spracherkennung wird diese Zeitersparnis verdeutlicht. Eine Minute an Sprache bedarf auf Wortebene das Zehnfache an Zuweisungszeit (zehn Minuten) und auf Ebene von Phonemen 100 mal so lange (100 Minuten).

2.1.3 Query- Funktionsarten

Nachdem Ziele, Ideen und der grundsätzliche Ablauf beschrieben wurden, ist es an der Zeit mögliche Query- Funktionen näher zu erläutern. Sie lassen sich in drei Arten einteilen:

- Membership Query Synthesis
- Stream-Based Selective Sampling
- Pool-Based Sampling

2.1.3.1 Membership Query Synthesis

Bei der Membership Query Synthesis ist es zum einem möglich ein Element aus dem Raum aller Elemente, dem Instanzraum, auszuwählen. Dies geschieht anhand definierter Eigenschaften. Um seine Klassenzugehörigkeit zu ermitteln, wird ein Orakel genutzt. Zum anderen kann anhand eines Modells ein neues Element erzeugt werden, welches dann an das Orakel übergeben wird. Dieses Element kann abstrakt sein und muss daher nicht notwendigerweise für den Menschen sichtbare Strukturen enthalten. Auf dieses Problem stießen K. Lang und E.Baum in ihrer Arbeit „Query learning can work poorly when a human oracle is used“ [BL92].

2.1.3.2 Stream-Based Selective Sampling

Das Stream-Based Selective Sampling entnimmt seine Elemente sequenziell aus dem Instanzraum und entscheidet dann individuell, ob sie dem Orakel überreicht werden oder nicht. Die Entscheidung wird in der Regel anhand des Informationsgehaltes getroffen. Beispielsweise könnte anhand der Modellparameter eine Region der Unsicherheit errechnet werden. Falls ein Element darin liegt, ist der Informationsgehalt dieses Beispiels hoch. Diese Methode eignet sich dann besonders gut, falls der zur Verfügung stehender Speicher oder die Rechenleistung stark limitiert ist.

2.1.3.3 Pool-Based Sampling

Zuletzt folgt das Pool-Based Sampling. Hierbei wird aus dem Instanzraum eine Menge an Elementen entnommen und jeweils ihr Informationsgehalt berechnet. Die Besten unter ihnen werden dann an das Orakel übergeben.

2.1.4 Selektionsstrategien

Im Folgenden werden einige Methoden vorgestellt, um den Informationsgehalt eines Elements zu bestimmen.

2.1.4.1 Uncertainty Sampling

Das Uncertainty Sampling ist wohl die einfachste Methode. Hierbei muss lediglich berechnet werden, wie unsicher sich das aktuelle Modell über die Klassenzugehörigkeit eines Objektes ist. Das mit der höchsten Unsicherheit wird genutzt, um das Modell zu trainieren.

Für ein probabilistisches Modell und einer binären Klassifikation ist dies verhältnismäßig einfach. Gesucht ist ein Beispiel, das durch das Modell als positiv erkannt wird und dessen Abstand zu 0.5 minimal ist. Unter der Annahme, dass die Klassenzugehörigkeit durch

einen Wert zwischen null und eins dargestellt wird, entspricht 0.5 der größtmöglichen Unsicherheit.

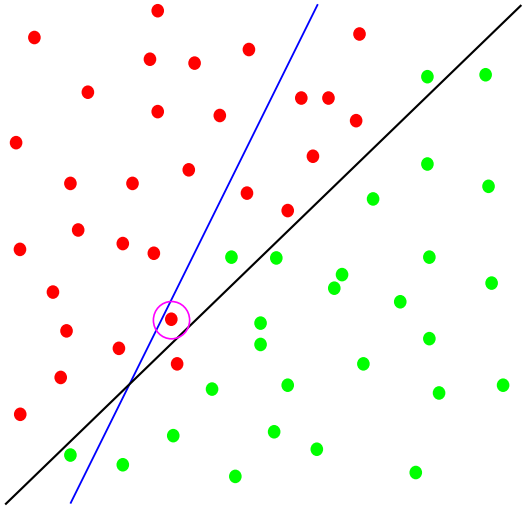


Abbildung 2.2: Trenngerade vor (blau) und nach dem Training (schwarz). Rechts der Trenngeraden sollten sich Elemente befinden, die zur Klasse gehören. Links davon die nicht dazugehören. Zu sehen ist eine mögliche Veränderung des Klassifikators nach Auswahl des pink umkreisten Beispiels. Dieses Beispiel wird hier als positiv erkannt und hat den kleinsten Abstand zu 0.5.

Bei mehr als zwei Klassen lässt sich das Objekt mit der geringsten Detektionssicherheit $x_{least\ confident}$ wie folgt berechnen:

$$x_{least\ confident} = \underset{x}{\text{argmax}}\ (1 - P_\theta(\hat{y}|x))$$

$$\hat{y} = \text{argmax}_y\ P_\theta(y|x)$$

Dabei entspricht $\hat{y}$ der Klasse zu der x mithilfe der Verteilung P_θ am wahrscheinlichsten zugeordnet wird. Die Verteilung wird anhand des Modells θ erstellt. $P_\theta(y|x)$ steht für die Wahrscheinlichkeit, dass x zur Klasse y gehört. θ entspricht dabei dem verwendeten Modell. Somit ist $x_{least\ confident}$ das Element mit der größten Wahrscheinlichkeit $1 - P_\theta(\hat{y}|x)$, das dem Element mit der größten Unsicherheit entspricht. Der Nachteil hierbei ist, dass nur die Klasse berücksichtigt wird, zu der x am besten passt. Die gesamte Verteilung wird nicht ausgewertet. Dieser Missstand wird durch das Miteinbeziehen der zweitbesten Klassenzugehörigkeit $\hat{y}_2$ wie folgt verbessert:

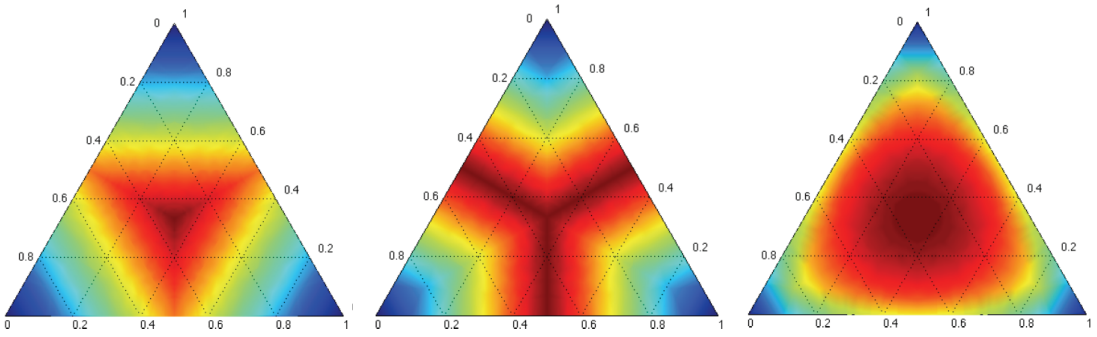
$$x_{margin} = \underset{x}{\text{argmin}}\ (P_\theta(\hat{y}|x) - P_\theta(\hat{y}_2|x))$$

Über das Minimum der Wahrscheinlichkeitsdifferenz (margin) für alle x, erhält man das Element, das zwei Klassen zugeordnet werden könnte. Aber auch hier wird nicht die gesamte Wahrscheinlichkeitsverteilung der Klassen berücksichtigt.

Unter Verwendung der Entropie zur Berechnung des unsichersten Elements ergibt sich folgender Ansatz, der das Problem der vorherigen Berechnungsmethoden löst.

$$x_H = \underset{x}{\operatorname{argmax}}(-\sum_i P_\theta(y_i|x) * \log(P_\theta(y_i|x)))$$

Durch die Entropie erhält man den Informationsgehalt eines Objektes bezogen auf die Wahrscheinlichkeitsverteilung der Klassen. Je gleichmäßiger die Wahrscheinlichkeitsverteilung für die Klassenzugehörigkeit eines Objektes x dabei sind, desto höher ist der Informationsgehalt. Dieses Element gehört dann theoretisch zu jeder Klasse. Durch das Training mit diesem könnte man daher den größten Nutzen ziehen. In Abbildung 2.3 sind die Regionen der Unsicherheit für die vorgestellten Berechnungsarten dargestellt.



(a) least confident (b) margin (c) entropy

Abbildung 2.3: Dargestellt sind die Regionen, die bei Verwendung von drei Klassen, als unsicher eingestuft werden. Rot bedeutet hohe Unsicherheit bzw. hohe Informationsdichte. Blau hohe Sicherheit und geringe Informationsdichte. Beispiele aus roten Bereichen werden bevorzugt für das Training verwendet [Set09].

2.1.4.2 Query-By-Committee

Bei diesem Ansatz wird zunächst ein Wahlkomitee $C = \theta^1 \ \ldots \ \theta^k$ an Modellen gebildet. Jedes Modell wird auf der Menge der klassifizierten Beispiele L trainiert. Um zu entscheiden, aus welchem der größte Nutzen gezogen werden kann, steht jedem $c^i \in C$, für jedes Beispiel eine Stimme zu. Das Objekt, bei dem die größte Uneinigkeit entsteht, wird für das Training verwendet.

Um die Uneinigkeit zu messen, gibt es zwei nennenswerte Funktionen. Zum einen die Entropie der Wahl:

$$x_{Hv} = \underset{x}{\operatorname{argmax}} - \sum_i \frac{V(y_i)}{|C|} * \log(\frac{V(y_i)}{|C|})$$

$$V(y_i) = \textit{Anzahl der Stimmen für die Klasse } y_i$$

Zum anderen die durchschnittliche Kullback-Leibler Divergenz, die in der Informationstheorie ein Maß für die Differenz zweier Wahrscheinlichkeitsverteilungen ist.

$$x_{KL} = \underset{x}{\operatorname{argmax}} \frac{1}{|C|} \sum_{c=1}^{|C|} KL(P_{\theta^{(c)}} || P_C)$$

$$KL(P_{\theta^{(c)}} || P_C) = \sum_i P_{\theta^{(c)}}(y_i|x) * \log(\frac{P_{\theta^{(c)}}(y_i|x)}{P_C(y_i|x)}))$$

$$P_C(y_i|x) = \frac{1}{|C|} \sum_{c=1}^{|C|} P_{\theta^{(c)}}(y_i|x)$$

$P_C(y_i|x)$ entspricht dabei der Entscheidung des Komitees, das x zur Klasse y_i gehört und entspricht der Gesamtverteilung. Diese wird dann mit den Einzelverteilungen der Modelle $P_{\theta^{(c)}}$ über die Funktion $KL(P_{\theta^{(c)}} || P_C)$ vergleichen.

2.1.4.3 Expected Error Reduction

Bei dieser Methode wird nicht wie bisher die Änderung des Modells gemessen. Stattdessen wird der zukünftige Fehler des Modells geschätzt, den es auf der Menge U der nicht klassifizierten Beispiele erzeugt, nachdem es mit der Menge $L_{t+1} = L_t \cup \langle x, y_i \rangle$ trainiert wurde. Dann wird das Trainingsbeispiel gewählt, durch das der Fehler am weitesten abgesenkt werden kann.

Eine Möglichkeit hierzu wäre, das Beispiel x zu wählen, durch das die Anzahl der Fehlvorhersagen minimiert wird.

$$x = \underset{x}{\operatorname{argmin}} \sum_i P_\theta(y_i|x)(\sum_{u=1}^{|U|} 1 - P_{\theta^{+\langle x,y_i \rangle}}(\hat{y}|x^{(u)}))$$

Hierbei ist die Klassenzugehörigkeit der $x \in U$ nicht geben, sondern wird aus der Klassenverteilung geschätzt. $P_{\theta^{+\langle x,y_i \rangle}}$ entspricht dabei der Wahrscheinlichkeitsverteilung des Modells θ, das zusätzlich mit dem Beispiel $\langle x, y_i \rangle$ trainiert wurde.

Eine Alternative dazu wäre, das Beispiel x zu wählen, mitdem die erwartete Entropie aller $x^{(u)} \in U$ in Kombination mit der Wahrscheinlichkeitsverteilung $P_{\theta^{+\langle x,y_i \rangle}}$ über den Klassen minimiert wird. Eine geringe erwartete Entropie bedeutet, dass die Verteilung über U ungleichmäßig ist. Im besten Falle sogar so, dass gilt:

$$\forall x \in U \; \forall j \exists i : \; i \neq j \wedge P_\theta(y_i|x) = 1 \wedge P_\theta(y_j|x) = 0$$

Dies würde bedeuten, dass die Menge U durch das Training mit $x^{(u)}$ perfekt erkannt werden könnte.

$$x = \underset{x}{\operatorname{argmin}} \sum_i P_\theta(y_i|x)(- \sum_{u=1}^{|U|} \sum_j P_{\theta^{+\langle x,y_i \rangle}}(y_j|x^{(u)}) * \log(P_{\theta^{+\langle x,y_i \rangle}}(y_j|x^{(u)})))$$

2.1.5 Analyse des Active Learning

Nach all den theoretischen Betrachtungen stellt sich nun die Frage nach der Umsetzbarkeit, der Effektivität und der Effizienz des Ganzen. Burr Settles geht an dieser Stelle auf die empirische und die theoretische Analyse ein. Er beschreibt, dass es aus theoretischer Sicht möglich ist, mit starken Einschränkungen obere Schranken für folgende Punkte nachzuweisen:

- Die Anzahl der benötigten Trainingsbeispiele.
- Die Anzahl der benötigten Iterationen.

Einige Einschränkungen sind im Folgenden aufgelistet:

- Kein Rauschen auf den Trainingsdaten.
- Elemente aus U unterliegen einer festen Klassenverteilung.
- Die Existenz von Modellen, die Objekte perfekt klassifizieren.

Bei Beweisen, die keinen Restriktionen unterliegen, sieht es nicht so gut aus. Diese Beweise beruhen teilweise auf unlösbaren Algorithmen oder auf Methoden, die zu komplex sind, um sie in der Praxis umzusetzen.

Unter empirischer Betrachtung finden sich viele Projekte, bei denen Active Learning zu positiven Resultaten führte. Zudem wurde die Anzahl der Objekte, denen eine Klasse zugewiesen werden musste, deutlich reduziert. Trotzdem wurde der gewünschte Grad an Genauigkeit erreicht.

2.2 Objekterkennung mit Rechteckmerkmalen

Paul Viola und Michael Jones veröffentlichten 2001 mit ihrer Arbeit „Robust Real-time Object Detection“ [VJ04] eine Methode zur Objekterkennung. Ihr Ziel war zum einen eine hohe Detektionsrate zu erreichen und zum anderen den Erkennungsprozess sehr schnell durchzuführen. Vier Schwerpunkte stechen in Ihrer Arbeit dabei hervor.

- Die Verwendung von Rechteckmerkmalen.
- Die Form der Bildrepräsentation als Integral-Image.
- Die Konstruktionsmethode einer Entscheidungskaskade.
- Das Arrangement einer Vielzahl an binären Klassifikatoren in einer Kaskade.

Der Entschluss das Rahmenwerk merkmalsbasiert, statt pixelbasiert zu gestalten, wurde überwiegend durch zwei Gründe motiviert. Zum einen ist es damit möglich Wissen über die Domäne ad hoc zu codieren und zum anderen ist es performat. Die verschiedenen Merkmalstypen sind in Abbildung 2.4 dargestellt. Durch diese Merkmale ist es möglich einfache Strukturen, wie z.B. Kanten oder Helligkeitsunterschiede, zu erkennen.

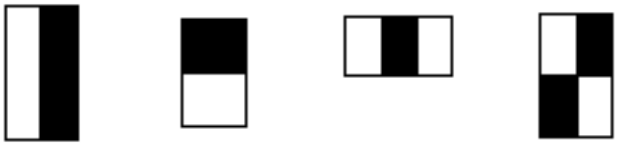


Abbildung 2.4: Basis aus Rechteckmerkmalen von Viola und Jones [VJ04]

Dabei wird der Wert eines Merkmals durch die Differenz der Grauwertsummen der schwarzen und weißen Rechteckflächen bestimmt. Dafür kann eine Methode genutzt werden, welche die entsprechenden Flächen schneller berechnet als dies durch eine Summation der einzelnen Pixelwerte möglich wäre.

An dieser Stelle wird der zweite Schwerpunkt genauer erläutert. Dieser entspricht der Bildrepräsentation durch ein Integral-Image. Dabei handelt es sich um eine Struktur, mit der die Auswertung eines Bildes beschleunigt werden soll.

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

$ii(x,y)$ entspricht einem Flächeninhalt, welcher der Summation der Pixelwerte eines Bereichs gleichkommt. Der markierte Bereich in Abbildung 2.5 entspricht daher dem Flächeninhalt von $ii(Punkt\_1)$. Wobei $i(x',y')$ für den Wert des Pixels (x',y') steht.

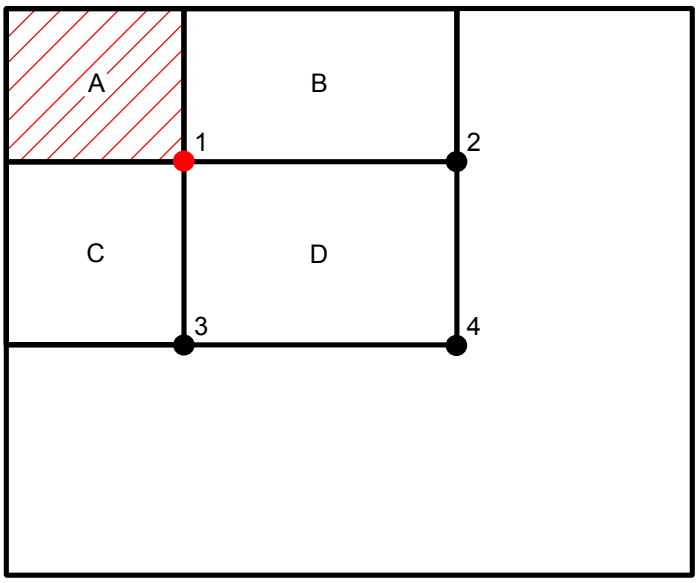


Abbildung 2.5: Flächenberechnung mit Hilfe des Integral-Image. A-D kennzeichnen Rechtecke, die durch den Punkt $(0,0)$ und einen der angegebenen Punkte $1-4$ gebildet werden.

Die Generierung eines Integral-Image kann, wie im Folgenden gezeigt wird, in Linearzeit zu der Anzahl der Pixel vollzogen werden. Berechnet werden wird $ii(x, y)$ mithilfe folgernder Rekursionsgleichung:

$$s(x, y) = s(x, y-1) + i(x, y)$$

$$ii(x, y) = ii(x-1, y) + s(x, y)$$

$$s(x, -1) = 0$$

$$ii(-1, y) = 0$$

Dadurch ist die Berechnung des Flächeninhaltes mit nur vier Zugriffen möglich.

$$A_{1-4} = D - B - C + A$$

$$A_{1-4} = ii(4) - ii(2) - ii(3) + ii(1)$$

Nachdem es nun möglich ist, die verwendeten Merkmale effizient auszuwerten kann ein Klassifikator trainiert werden. Dieser besteht aus einer Kombination verschiedener schwacher Klassifikatoren. Jedem dieser Klassifikatoren ist dabei genau ein Merkmal zugeordnet. Anhand eines Schwellwertes und der Auswertung eines Merkmals kann eine binäre Klassifikation erfolgen.

Für die Erstellung eines kombinierten Klassifikators verwenden Viola und Jones AdaBoost.

2.3 Boosting

Dabei handelt es sich um überwachte Lernverfahren des maschinellen Lernens, welche versuchen eine möglichst optimale Kombination an schwachen Klassifikatoren zu ermitteln. Diese bilden dann zusammen einen stärkeren Klassifikator. Bereits 1990 beschrieb Robert E. Schapire in seiner Arbeit „The Strength of Weak Learnability“ [Sch90] das Training eines starken Klassifikators auf Grundlage von schwachen Klassifikatoren. 1995 verfolgte Yoav Freund diese Idee weiter und veröffentlichte in seiner Arbeit „Boosting a Weak Learning Algorithm By Majority“ [Fre90] eine ähnliche Methode. Schwache Klassifikatoren bilden auch hier einen starken. Dessen Ausgabe entsteht durch einen Mehrheitsentscheid der schwachen Klassifikatoren. Auf diese Weise entstandene Knoten können als Eingabe anderer Knoten dienen. Eine so entstandene Struktur hat große Ähnlichkeit mit einem Multilagen-Perzeptron. Im selben Jahr veröffentlichten sie in Kooperation eine Publikation [FS97] in der sie den ersten in der Praxis einsetzbaren adaptiven Boosting-Algorithmus vorstellten. Sein Name lautet AdaBoost. Kurz darauf beschrieb Leo Breiman [Bre98] [Bre96] [Bre99] eine ganze Klasse von Algorithmen. Diese werden als „arcing algorithms“ bezeichnet. Interessant dabei ist, dass AdaBoost nur eine spezielle Form dieser Algorithmen darstellt.

2.3.1 Begriffsdefinitionen

Beispiel Ein Beispiel wird im Allgemeinen durch eine endliche Menge $M = m_1, ..., m_k$ von Merkmalen beschrieben. Jedem Beispiel wird eine Klasse $y \in Y$ zugeordnet. Formal handelt es sich also um einen Vektor $\langle m_1, ..., m_n, y_i \rangle \in m_1 \times ... \times m_k \times Y$. Häufig wird ein Beispiel auch als Objekt oder Instanz bezeichnet.

Merkmal Ein Merkmal entspricht einem definieren Bereich in einem Beispiel. Es wird abhängig von dem gewählten Modell ausgewertet. Die Auswertung wird häufig auch Antwort des Merkmals genannt und mit $f(m, x)$ beschrieben. Wobei x für das Objekt steht, indem das Merkmal m ausgewertet werden soll. Im Falle von Rechteckmerkmalen entspricht die Antwort einer Flächendifferenz.

Schwacher Klassifikator Dies sind Funktionen, welche eine Hypothese $h_{m_i}(x)$ über das ihnen zugeordnete Merkmal m_i anstellen. Diese muss nicht perfekt sein. Es genügt, wenn sie ein wenig besser ist als Raten. Also wenn etwas mehr als 50% der Beispiele korrekt zugeordnet werden können. Häufig wird ein schwacher Klassifikator mit einem Merkmal gleichgesetzt.

Hypothese Eine Hypothese ist eine Abbildung der Antwort eines Merkmals auf eine Klasse $y \in Y$.

Starker Klassifikator Dabei handelt es sich um eine Menge $K = h_{m_{i1}}(x), \ldots, h_{m_{ik}}(x)$ von schwachen Klassifikatoren. Während dem Training wird entschieden, welche Elemente in die Menge aufgenommen werden. Die Zugehörigkeit eines Beispieles x zu einer Klasse $y \in Y$ wird dann anhand aller Hypothesen entschieden. Ein starker Klassifikator wird häufig auch als Knoten bezeichnet.

2.3.2 Offline Boosting

Diese Art des Boosting entspricht der Idee von Yoav Freund und Robert E. Schapire. Durch Kombination von schwachen Klassifikatoren wird ein starker kreieren. Der Zusatz „offline“ soll verdeutlichen, dass dem verwendeten Lernalgorithmus zu jeder Zeit tausende positive und negative Trainingsbeispiele zur Verfügung stehen. Auf deren Grundlage erfolgt die Auswahl der schwachen Klassifikatoren. Das Training nach dieser Methode kann durchaus mehrere Tage in Anspruch nehmen. Im Folgenden werden nun zwei Offline Boosting Algorithmen näher erläutert.

2.3.2.1 AdaBoost

Gegeben sei dabei eine Menge an Beispielen $(x_1, y_1) \ldots (x_k, y_k)$ mit jeweils einem Initialgewicht. x_j ist dabei das $j'te$ Trainingsbeispiel. $y_j \in 0, 1$ gibt an ob es negativ (0) oder positiv (1) ist. Die h_i kennzeichnen einen schwachen binären Klassifikator. Im Folgenden ist der Algorithmus beschrieben:

Schritt 1: Normierung der Gewichte der Beispiele.

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

Schritt 2: Für jedes Merkmal i wird ein Klassifikator h_i trainiert.

Schritt 3: Finden des Merkmals h_i dessen Klassifikationsfehler e_i, über allen Beispielen j, minimal ist.

$$e_i = \sum_j w_j \left| h_i(x_j) - y_j \right|$$

Schritt 4: Neugewichten der Beispiele.

$$w_{t+1,j} = w_{t,j} * \beta_i^{1-e_i}$$

$$\beta_i = \frac{e_i}{1 - e_i}$$

Schritt 5: Gewicht des Merkmals h_i berechnen.

$$\alpha_i = log \frac{1}{\beta_i}$$

Dies wird T mal wiederholt, und man erhält somit einen stärkeren Klassifikator $h(x)$.

$$h(x) = (\sum_{i=1}^{T} \alpha_i h_i(x)) \geq (\frac{1}{2} \sum_{i=1}^{T} \alpha_i)$$

Daraus geht hervor: Je mehr Merkmale hinzugefügt werden umso besser ist das Ergebnis. Gleichzeitig steigt jedoch auch der Aufwand der Auswertung. Dieses Dilemma lässt sich lösen, indem die Anzahl der Merkmale pro Klassifikator einschränkt und damit eine Fehlklassifikation erlaubt wird. Dafür werden jedoch mehrere Klassifikatoren erzeugt, welche in einer Kaskade angeordnet werden. Dies ist exemplarisch in Abbildung 2.6 dargestellt. Die Aufgabe jedes Knotens ist eine Teilmenge der negativen Bereiche, die getestet werden, auszusondern. Am Ende der Kaskade sollten dann möglichst nur positive Regionen übrig bleiben.

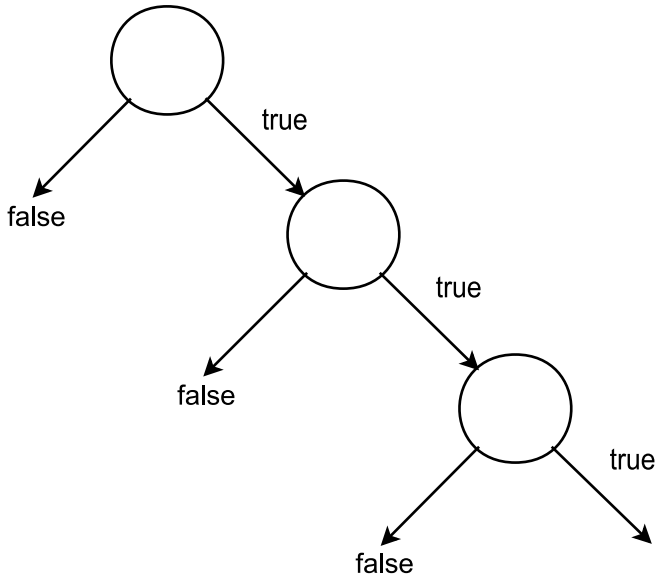


Abbildung 2.6: Kaskade an Klassifikatoren

2.3.2.2 Direct-Feature-Selection

Die Direct-Feature-Selection, welche in der Arbeit „Learing a Rare Event Detection Cascade by Direct Feature Selection“ [WRM03] von Jiynxin Wu *et al.* vorgestellt wird, ist eine Methode einen Klassifikator zu generieren. Dies ist dem Training eines Kaskadenknotens gleichzusetzen. Der Vorteil dieser Methode liegt im Vergleich zu AdaBoost darin, dass die Zeit zur Generierung eines Knotens deutlich reduziert wurde. Dadurch wird die Erzeugung der gesamten Kaskade beschleunigt. Ihre Untersuchungen haben ergeben, dass die Direct-Feature-Selection im Vergleich zu AdaBoost c.a. 100 mal schneller ist. Dies liegt unter anderem daran, dass AdaBoost die Gewichte der Trainingsmenge in jedem Iterationsschritt t neu berechnet. Deshalb müssen die schwachen Klassifikatoren im Schritt $t + 1$ neu trainiert werden.

Hier ist dies nicht der Fall. Die Merkmale werden einmalig auf Grundlage einer positiven Menge P und einer negativen Menge N vor der Knotengenerierung trainiert. Dies bedeutet, dass für jedes Merkmal ein optimaler Schwellwert auf Grundlage von P und N gesucht wird.

Sowohl bei Viola und Jones als auch bei Jiynxin Wu *et al.* wird die Menge N nach der Erstellung des Knotens verändert. Dabei werden Elemente entfernt, die durch den Knoten korrekt eingestuft wurden. N wird danach mit Elementen aufgefüllt, die von der bisher erzeugten Kaskade als korrekt klassifiziert werden, jedoch negativ sind.

Die Selektion der Merkmale verläuft bei der Direct-Feature-Selection gemäß Abbildung 2.7(b). d entspricht der minimalen Detektionsrate des Knotens und f der maximalen Falschdetekti-

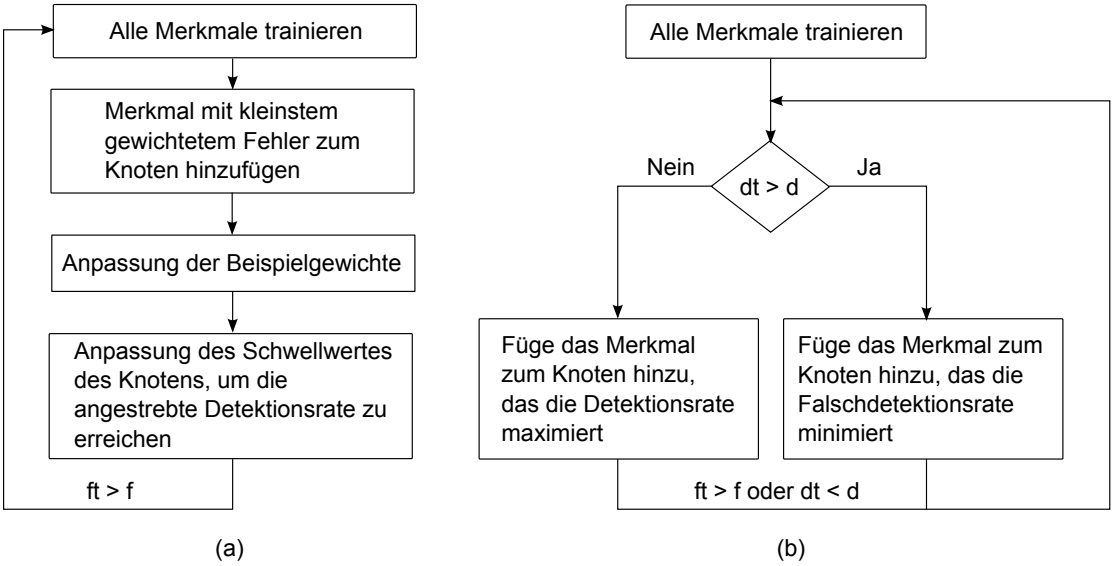


Abbildung 2.7: Ablauf der Knotengenerierung für (a) AdaBoost und (b) Direct-Feature-Selection. d entspricht der minimalen angestrebten Erkennungsrate des Knotens und f der maximalen Falschdetektionsrate. d_t und f_t entsprechen der Detektions- und Falschdetektionsrate im Iterationsschritt t

onsrate. Solange die Detektionsrate d_t nicht größer als d und die Fehlerrate f_t nicht kleiner als f ist, wird dem Knoten ein Merkmal hinzugefügt. Falls $d_t < d$ ist, wird das Merkmal hinzugefügt, welches d_{t+1} maximiert. Wenn $d_t \geq d$ gilt, wird das Merkmal hinzugefügt, das f_{t+1} minimiert.

Nachdem das Ensemble erstellt wurde, werden die Gewichte der einzelnen Merkmale bestimmt. Dies geschieht nach dem gleichen Prinzip wie bei AdaBoost. Zuletzt wird der Schwellwert des Knotens ermittelt. Er wird so eingestellt, dass der Knoten die Erkennungsrate d erreicht.

2.3.3 Online Boosting

Um mit Hilfe von Online Boosting einen starken Klassifikator zu trainieren, stehen nicht zu jeder Zeit tausende Beispiele zur Verfügung. Hier werden dem Lernalgorithmus die Beispiele sequenziell übergeben. Das Modell wird einzeln mit jedem Beispiel trainiert. Anschließend findet die Selektion der schwachen Klassifikatoren statt. Auf diese Weise ist es möglich, den Klassifikator später einfach weiter zu trainieren. Die bis zu diesem Zeitpunkt trainierten Aspekte gehen dabei nicht verloren. Sie werden lediglich der neuen Situation angepasst. Beim Offline Boosting müssten neue Beispiele erst in die Trainingsmenge eingefügt werden. Danach würde die Erzeugung eins neuen Klassifikators anhand der erweiterten Trainingsmenge erfolgen.

Im nächsten Abschnitt wird die Onlinevariation des am häufigsten verwendeten Offline Boosting Algorithmus beschrieben.

2.3.3.1 Online AdaBoost

Nikunj C. Oze beschrieb in seiner Arbeit „Online Bagging and Boosting“ [Oza05] eine Onlinevariation des original AdaBoost. Bei seinem Algorithmus wird die Funktionsweise des Offline AdaBoost imitiert. Die Schwierigkeit hierbei besteht darin, dass dem Algorithmus zu jedem Trainingszeitpunkt nur ein Beispiel bekannt ist. Ein Ensemble an schwachen Klassifikatoren alleine aufgrund dieses Wissens zu erstellen, entspricht jedoch nicht dem Konzept des AdaBoost. Prinzipiell wird für die Erstellung des starken Klassifikators eine Vielzahl an Beispielen benötigt. Dieses Problem wird bei der Variante von Oze durch das Erstellen einer Statistik gelöst. Diese codiert, wie gut ein Merkmal die bisherigen Beispiele erkannt hat, mit denen das Training erfolgte. Hierzu ist es nötig jeden schwachen Klassifikator um entsprechende Variablen zu erweitern. Der Ablauf des Algorithmus 2.1 wird im folgenden näher beschrieben.

Zunächst werden die Statistikvariablen $\lambda_{correct}$ und λ_{wrong} aller schwachen Klassifikatoren $h_m \in H$ mit null initialisiert. $\lambda_{correct}$ codiert die Beispiele die korrekt erkannt wurden. Während λ_{wrong} die Beispiele codiert, die falsch erkannt wurden. Danach wird für jedes eintreffende Beispiel $\langle x, y\rangle$ die Funktion $OnlineBoosting(H, L_o, \langle x, y\rangle)$ aufgerufen. L_o entspricht dabei einem online Algorithmus, der zum Training eines schwachen Klassifikators eingesetzt werden kann. y steht für die Klasse der x angehört.

Das Gewicht λ des Trainingsbeispiels wird zunächst auf eins gesetzt. Danach wird jeder schwache Klassifikator h_m k-mal mit L_0 trainiert. Das k wird anhand einer Poisson-Verteilung bestimmt, die von λ abhängig ist. Nun folgt der Test ob h_m es schafft, x korrekt der Klasse y zuzuordnen. Ist dies der Fall, so wird die Statistikvariable $\lambda_m^{correkt}$ von h_m aktualisiert. Falls nicht, wird λ_m^{wrong} von h_m adaptiert. Danach erfolgt die Berechnung des neuen geschätzten Fehlers von h_m. Wie auch bei AdaBoost wird das Gewicht eines Beispiels verringert, falls es richtig zugeordnet wurde, und erhöht falls nicht. Das Gewicht drückt dabei aus, wie gut die bisher trainierten schwachen Klassifikatoren das Beispiel erkennen. Nachdem alle $h_m \in H$ trainiert wurden, kann ein starker Klassifikator h_{fin} für jede Klasse $y \in Y$ erstellt werden.

2.4 t-Distributed Stochastic Neighbor Embedding (t-SNE)

Bei t-SNE handelt es sich um ein Verfahren zur Dimensionsreduzierung hoch-dimensionaler Daten. Sie werden im Anschluss im nieder-dimensionalen Raum visualisieren. Meist im R^2 oder R^3. Laurens van der Maaten und Geoffrey Hinton beschreiben dabei ihr vorgehen in ihrer Arbeit „Visualizing Data using t-SNE“ [MH08].

Es gibt eine Vielzahl an Verfahren zu diesem Zweck. Zu den klassischen linearen Methoden gehören beispielsweise die Principal Components Analysis (PCA) oder das Multidimensional Scaling (MDS). Diese versuchen Datenpunkte, die sich im hoch-dimensionalem Raum

Algorithmus 2.1 Online AdaBoost

procedure OnlineBoosting($H, L_o, \langle x, y \rangle$)
  $\lambda \leftarrow 1$
  for all base model $h_m \in H$ **do**
    Setze k entsprechend Poisson(λ)
    for $i \leftarrow 1, k$ **do**
      $h_m \leftarrow L_o(h_m, \langle x, y \rangle)$
    end for
    if $y = h_m(x)$ **then**
      $\lambda_m^{correct} \leftarrow \lambda_m^{correct} + \lambda$
      $error_m = \frac{\lambda_m^{wrong}}{\lambda_m^{correct} + \lambda_m^{wrong}}$
      $\lambda \leftarrow \lambda * \frac{1}{2(1-error_m)}$
    else
      $\lambda_m^{wrong} \leftarrow \lambda m^{wrong} + \lambda$
      $error_m = \frac{\lambda_m^{wrong}}{\lambda_m^{correct} + \lambda_m^{wrong}}$
      $\lambda = \lambda * \frac{1}{2error_m}$
    end if
  end for
  $h_{fin}(x) = \text{argmin}_{y \in Y} \sum_{m=1}^{|M|} \log(\frac{1-error_m}{error_m}) * I(h_m(x) = y))$
  return h_{fin}
end procedure

unähnlich sind, im nieder-dimensionalen Raum voneinander fernzuhalten. Dadurch können jedoch vorhandene Strukturen verloren gehen.

Zu den nicht linearen Methoden zählen folgende Verfahren:

- Sammon mapping
- Curvilinear Components Analysis (CCA)
- Stochastic Neighbor Embedding (SNE)
- Isomap
- Maximum Variance Unfolding (MVU)
- Locally Linear Embedding (LLE)
- Laplacian Eigenmaps

Die Schwächen dieser Verfahren bestehen zum einen darin, dass einige nicht in der Lage sind, Strukturen korrekt abzubilden. Zum anderen darin, dass sie bei Anwendung auf realen Datensätzen wenig erfolgreich waren. Auch ist häufig das sogenannte „Crowding Problem“ zu beobachten.

Ziel von t-SNE ist die Auswirkungen dieser Probleme zu verringern. Dabei sollen sowohl lokale als auch globale Strukturen des hoch-dimensionalen Raums erhalten bleiben. Zusätzlich soll der Reduktionsprozess beschleunigt werden.

2.4.1 Crowding Problem

Gegeben sei dabei beispielsweise eine Reduktion der $x \in R^m$ auf $y \in R^2$ mit der Abbildung $f: x_i \rightarrow y_i$. Betrachtet wird im Folgenden eine Sphäre mit x_i als Zentrum, Radius r und den Nachbarn x_j von x_i. Diese befinden sich innerhalb der Sphäre. Weiterhin wird angenommen, dass alle x_j im Volumen gleich verteilt sind. Werden nun die Entfernungen von y_i zu den anderen Punkten y_j modelliert, so erhält man folgendes Problem:

Um alle Distanzen korrekt zu modellieren, würde eine enorme Fläche benötigt. Das bedeutet, dass die Distanzen zwischen y_i und den weit entfernten y_j so groß wären, dass nicht alle Punkte auf einmal darstellbar sind. Daher wird der Maßstab verändert. Bei diesem Verkleinerungsprozess drängen sich Punkte, die nahe an y_i liegen, langsam bei y_i zu einem Haufen zusammen.

2.4.2 SNE

Im Folgenden wird die grundlegende Funktionsweise des SNE-Algorithmus näher erläutert.

Die Idee hierbei ist, allen x_i im hoch-dimensionalen Raum einem y_i im nieder-dimensionalen Raum zuzuordnen. Dabei soll die Wahrscheinlichkeitsverteilung P_i eines Punkes x_i der Wahrscheinlichkeitsverteilung Q_i des Punktes y_i möglichst ähnlich sein. P_i wird dabei anhand der Distanzen von x_i zu seinen Nachbar x_j erstellt. Bei Q_i werden die Distanzen zwischen y_i und dessen Nachbarn y_j genutzt. Die Verteilungen werden durch eine Gaußfunktion, mit x_i bzw. y_i als Zentrum, modelliert. Somit ergibt sich für die Wahrscheinlichkeitsverteilung P_i folgende Formel:

$$p_{j|i} = \frac{\exp(\frac{-||x_i - x_j||^2}{2\sigma_i^2})}{\sum_{k \neq i} \exp(\frac{-||x_i - x_k||^2}{2\sigma_i^2})}$$

Die Wahrscheinlichkeiten $p_{i|i}$ werden dabei auf null gesetzt, da nur Distanzen zwischen verschiedenen Punkten verglichen werden. Die σ_i werden an die Dichte der Region angepasst in der sich das x_i befindet. Sind dort viele Elemente angesiedelt, so wird es klein gewählt. Sonst entsprechende größer. Dabei wird versucht das σ_i so zu wählen, dass die errechnete Perplexität $\mathrm{Perp}(P_i)$ einer Verteilung P_i der voreingestellten Perplexität entspricht. Dieser Parameter kann als Richtwert für die Anzahl der Nachbarn von x_i angesehen werden.

$$\mathrm{Perp}(P_i) = 2^{\mathrm{H}(P_i)}$$

$$\mathrm{H}(P_i) = -\sum p_{j|i} \log p_{j|i}$$

$H(P_i)$ entspricht dabei der Shannon Entropie der Verteilung P_i. Analog ergibt sich für die Wahrscheinlichkeitsverteilung Q_i folgende Formel:

$$q_{j|i} = \frac{\exp(-||y_i - y_j||^2)}{\sum_{k \neq i} \exp(-||y_i - y_k||^2)}$$

Wobei die Varianz der Gaußverteilung hier auf $\frac{1}{\sqrt{2}}$ gesetzt wird. Des Weiteren gilt hier auch $q_{i|i} = 0$, da nur Distanzen zwischen verschiedenen Punkten verglichen werden.

Als Maß für die Gleichheit zweier Verteilungen wird hier die Kullback-Leibler Divergenz genutzt. Sie fungiert als Kostenfunktion.

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log(\frac{p_{j|i}}{q_{j|i}})$$

Ziel ist diese Kostenfunktion mittels eines Gradientenabstiegs zu minimieren.

$$\frac{\delta C}{\delta y_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

Zu beachten gilt, dass diese Kostenfunktion nicht symmetrisch ist. Daher entstehen hohe Kosten, falls zwei naheliegende Punkte (x_i, x_j), auf weit voneinander entfernte Punkte (y_i, y_j) abgebildet werden. Wohingegen nur geringe Kosten entstehen, falls (x_i, x_j) weit auseinanderliegen und (y_i, y_j) nahe beieinander.

2.4.3 Optimierung

Obwohl SNE annehmbare Visualisierungen erzeugt, wird die Nutzbarkeit durch zwei Faktoren eingeschränkt. Zum einen durch die aufwendige Kostenfunktion, welche die Verteilungen aller Punkte miteinander vergleicht. Zum anderen tritt hier, wie bei anderen Verfahren auch, das Crowding Problem auf.

Um das Verfahren zu Beschleunigen wird die Kullback-Leibler Divergenz zwischen Gesamtverteilungen P und Q minimiert, statt wie bisher eine Summe an Kullback-Leibler Divergenzen.

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log(\frac{p_{ij}}{q_{ij}})$$

Um dem Crowding Problem entgegen zu wirken, wird für die Berechnung der Wahrscheinlichkeitsverteilung im nieder-dimensionalen Raum eine Studentsche t-Verteilung verwendet. Diese bietet den Vorteil, dass die Werte an den Flanken dieser Funktion höher sind als die bei einer Gaußverteilung. Dadurch werden größere Distanzen im hoch-dimensionalen Raum

besser im nieder-dimensionalen Raum wiedergegeben. Zudem ist sie einfacher zu berechnen. Für die Wahrscheinlichkeitsverteilung Q ergibt sich damit folgende Formel:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

Ein weiteres Problem stellen die Punkte da, deren Position nicht genau durch andere Punkte bestimmt wird. Diese haben die Eigenschaft, dass die Distanzen zu allen Nachbarn groß sind. Daher sind die entsprechenden Wahrscheinlichkeiten $p_{j|i}$ sehr klein. Dadurch ist ihr Einfluss auf die Kostenfunktion sehr gering. Um diesen freien Punkten entgegenzuwirken, wird jedem ein Mindestbeitrag zugewiesen, indem sichergestellt wird, dass $\sum_i p_{ij} > \frac{1}{2n}$ für alle Punkte x_i gilt. Somit ergibt sich für die Wahrscheinlichkeitsverteilung P folgende Formel:

$$p_{ij} = \frac{(p_{j|i} + p_{i|j})}{2n}$$

Nach diesen Optimierungen ergibt sich nun für die Minimierung der Kostenfunktion mittels Gradientenabstieg folgender Ausdruck:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}$$

3 Themenbezogen Arbeiten

In diesem Kapitel werden einige Rahmenwerke und Methoden beschrieben, bei denen Konzepte des Active Learning zum Einsatz kommen.

3.1 Confidence-Based Active Learing

In der Arbeit „Confidence-Based Active Learing“ [LS06] von Mingkun Li und Ishwar K. Sethi, wird eine Methode für das Training von Klassifikatoren vorgestellt. Hierbei verwenden sie Strategien des Active Learning. Für die Selektion der Beispiele, mit denen das nächste Training erfolgen soll, wird das Uncertainty Sampling in Verbindung mit einem Pool-Based Sampling verwendet. Jedoch nicht auf konventionelle Art. Sie verwenden dabei als Maß für die Unsicherheit eines Beispiels den geschätzten Fehler der Klassifikation. Dieser Fehler entspricht einer Bewertung der Ausgabe des Klassifikators. Eine hohe Bewertung steht dabei für eine hohe Detektionssicherheit und einen geringen Fehler. Der Fehler kann auch als Wahrscheinlichkeit interpretiert werden, dass ein Beispiel einer Objektklasse angehört.

Eine wichtige Eigenschaft der Ausgabe ist, dass in ihr alle Merkmale codiert sind, die ein Klassifikator zur Entscheidungsfindung verwendet hat. Das heißt, hier findet eine Transformation des mehrdimensionalen Merkmalsraums auf einen eindimensionalen Raum statt. Da ein Beispiel anhand einer Vielzahl an Merkmalen beschrieben werden kann, wird somit auch der Beispielraum auf den eindimensionalen Raum transformiert. Dies hat zur Folge, dass Regionen der Unsicherheit, welche der Klassifikator nicht eindeutig zuordnen kann, auf Intervalle des eindimensionalen Raums abgebildet werden. Dieser Vorgang ist in Abbildung 3.1 dargestellt.

Gemäß dem Active Learning kann anhand der Unsicherheit entschieden werden, welche Beispiele für das Training von hoher Bedeutung sind. Da hier die Unsicherheit dem Fehler entspricht, werden alle Beispiele selektiert, deren Fehler sich im Intervall $[t_1, t_2]$ befindet.

$$Q(x) = \begin{cases} 1, & \text{wenn } t_1 \leq \text{Fehler}(x) \leq t_2 \\ 0, & \text{sonst} \end{cases}$$

Daher wird ersichtlich, dass dieses Verfahren nur auf Klassifikatoren anwendbar ist, welche die notwendige Ausgabe aufweisen. Dies stellt in der Regel jedoch kein Problem dar, da viele Klassifikatoren bereits über eine solche Ausgabe verfügen bzw. dahingegen modifiziert wurden oder werden können.

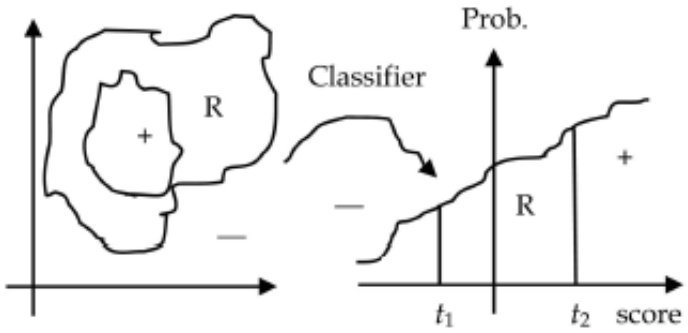


Abbildung 3.1: Transformation der Region der Unsicherheit auf ein Intervall des eindimensionalen Raums durch den Klassifikator. Die Wahrscheinlichkeit entspricht dabei dem Fehler. Dieser entspricht der Ausgabe des Klassifikators. [LS06]

3.2 Rahmenwerk zur Fahrzeugerkennung

In ihrer Arbeit „A General Active-Learning Framework for On-Road Vehicle Recognition and Tracking“ [ST10] von Sayanan Sivaraman *et al.* wird ein Rahmenwerk zur Erkennung von Fahrzeugen präsentiert. Ein Hauptbestandteil des Rahmenwerks ist das Training des Klassifikators. Hierbei werden Methoden des Active Learning verwendet. Ihre Hoffnung bestand darin mit dieser Art des Trainings besser mit den Variationsmöglichkeiten von Fahrzeugen umgehen zu können. Das heißt, im Vergleich zu konventionellen Trainingsmethoden, eine höhere Robustheit sowie Detektionssicherheit des Klassifikators zu erzielen. In Verbindung mit einem anderen System sollte es dann möglich sein potenzielle Gefahrensituationen im Straßenverkehr zu erkennen und den Fahrer über diese zu informieren.

Der Ablauf des Trainings sieht vor, dass zunächst ein initialer Klassifikator passiv bzw. offline trainiert wird. Hierzu wurden 7500 positive und 20500 negative Beispiele verwendet. Danach wird er auf ein Testvideo angewendet. Mithilfe des Stream-Based Selective Sampling werden Beispiele aus unsicheren Regionen selektiert. Ein Nutzer ordnet ihnen Klassen zu. Das erneute offline Training erfolgt sobald genügend Beispiele gesammelt wurden. Hier 10000 positive und 12172 negative. Der gesamte Trainingsablauf wird in Abbildung 3.2 dargestellt. In Abbildung 3.3 ist ein Auszug des Selektionsprozesses der Beispiele zu sehen.

Als Klassifikator wird hier eine Kaskade an binären Klassifikatoren verwendet, die jeweils mit AdaBoost trainiert werden. Bei den verwendeten Merkmalen handelt es sich um Rechteckmerkmale.

Ihre Erwartungshaltung, mithilfe des Active Learning den Trainingsaufwand zu reduzieren, wurde durch Tests bestätigt. Dabei erzielten sie Resultate, die mit den Ergebnissen konventioneller Methoden vergleichbar sind.

Abbildung 3.2: Dargestellt ist der Ablauf des Klassifikatortrainings. Zunächst wird ein initialer Klassifikator trainiert. Danach wird auf Grundlage seiner Detektionen der Trainingszyklus des Active Learning durchlaufen. [ST10]

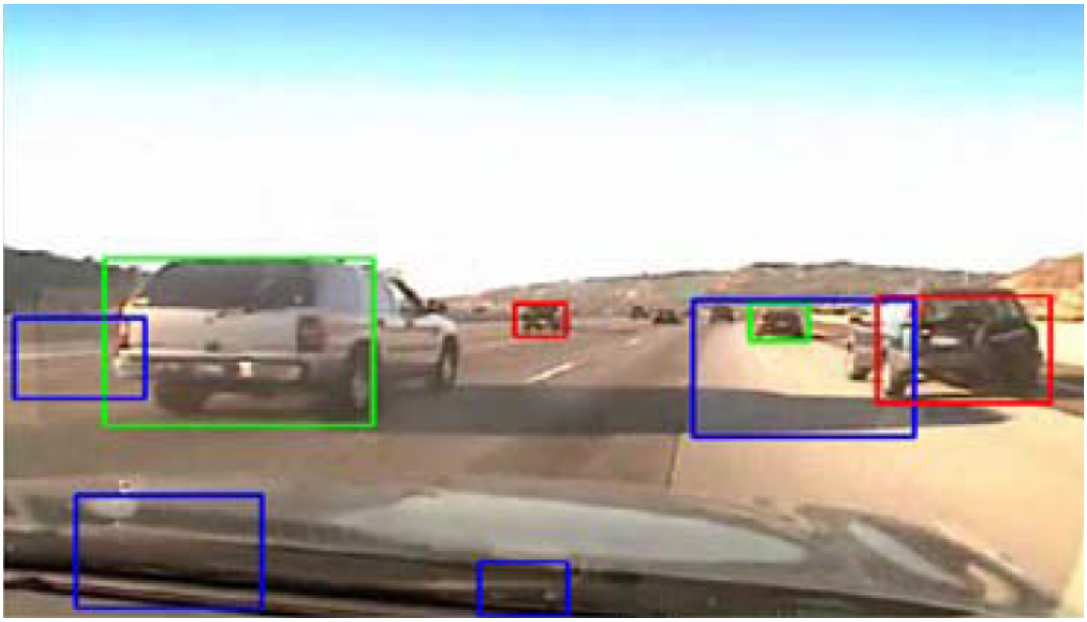


Abbildung 3.3: Dargestellt ist ein Auszug des Selektionsprozesses der Beispiele. Rote Rechtecke deuten auf negativ erkannte Regionen. Die Grünen auf Positive. Blau Rechtecke weisen auf Region mit hoher Unsicherheit hin. Diese eignen sich daher besonders gut für das Training. Ihnen wird demzufolge eine Klasse zugewiesen. [ST10]

3.3 Rahmenwerk zur Visualisierung des Klassifikatorverhaltens

Christin Seifert *et al.* beschrieben in ihrer Arbeit „User-based active learning“ [SG10] eine abgewandelte Form des Active Learning. Dabei wird nicht automatisch anhand des Informationsgehaltes entschieden, welche Beispiele dem Nutzer für die Klassenzuteilung dargelegt werden. Die Auswahl dieser Beispiele wird durch den Nutzer selbst getroffen. Um diesen Selektionsprozess zu unterstützen, wurden bei der Implementierung des Rahmenwerkes folgende Punkte beachtet:

- Die Visualisierung sollte es erlauben, das Verhalten des Klassifikators zu beurteilen.
- Falsch zugeordnete Beispiele sollten schnell und einfach erkannt werden.
- Der Nutzer sollte in der Lage sein, schnell ganze Bereiche markieren und Klassifizieren zu können.
- Die Visualisierung sollte unabhängig vom Klassifikator sein. Vorausgesetzt es ist möglich, durch seine Ausgabe eine a-posteriori Wahrscheinlichkeitsverteilung zu erstellen. Des Weiteren darf jedem Beispiel nur eine Klasse zugeordnet werden.

Abbildung 3.4 zeigt eine Darstellung der Detektionen eines Klassifikators. Dabei ist jeder Klasse eine Farbe zugeordnet. Die Detektionen werden anhand der a-posteriori Wahrscheinlichkeitsverteilung positioniert. Ihre Farbe entspricht der Klasse, der sie zugeordnet wurden. In dieser Ansicht können nun falsch klassifizierte Beispiele erkannt und markiert werden. Im Anschluss an das Markieren folgt die Klassenzuordnung. Danach wird der Klassifikator trainiert.

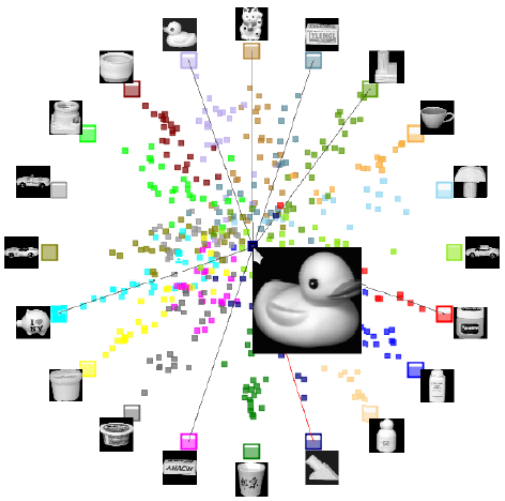


Abbildung 3.4: Dargestellt ist die Positionierung der Detektionen anhand ihrer a-posteriori Wahrscheinlichkeitsverteilung über den Klassen. Ihre Farbe entspricht der Klasse, der sie zugeordnet wurden. Mögliche Objektklassen sind kreisförmig mit Repräsentanten dargestellt. [SG10]

4 Systembeschreibung

Durch das Active Learning werden bereits vielversprechende Erfolge erzielt um sowohl die Trainingszeit, als auch die Anzahl der benötigten Trainingsbeispiele zu reduzieren. Voraussetzung dafür ist, dass bereits ein trainierter Klassifikator vorliegt. Dieser wird von den Mechanismen des Active Learning benötigt um Trainingsbeispiel zu bestimmen mit denen ein zusätzliches Training erfolgt. Dies soll seine Klassifikationsleistung verbessern. Man kann diesen initialen Klassifikator als Richtungsvorgabe für das Active Learning verstehen. Je besser die Vorgabe ist, desto präziser erfolgt die Selektion von Beispielen, mit denen gezielt Schwachpunkte des Klassifikators trainiert werden.

Die Aufgabe des Nutzers ist dabei sehr eingeschränkt. Er ist dazu da die Beispiele, die ihm durch das System präsentiert werden, einzuordnen. Aus welchen spezifischen Gründen ein Beispiel ausgewählt wurde, wird ihm nicht vermittelt. Induktiv ist dabei nur ersichtlich, dass durch das Training mit diesen Beispielen der größte Lernerfolg erzielt werden kann. Anhand einer Erweiterung des Active Learning um interaktive Aktionsmöglichkeiten soll der Nutzer Einblick in die Funktionsweise des Klassifikators sowie der verwendeten Merkmale erhalten. Auf diese Weise sollte es dem Nutzer möglich sein gezielt Wissen in das Training einfließen zu lassen, um die Güte es Klassifikators mit möglichst wenig Beispielen zu verbessern. Durch die Integration des Nutzers in den Trainingsablauf sollte ein verhältnismäßig guter initialer Klassifikator nicht länger notwendig sein. Die Vorgabe der Trainingsrichtung würde, mithilfe des interaktiven Trainings, durch den Nutzer bestimmt werden.

Damit es dem Benutzer möglich ist das Verhalten des Klassifikators verstehen und beeinflussen zu können, muss dessen Funktionsweise aufgezeigt werden. Daher sind mehrere Ansichten notwendig, die dem Benutzer dahin gehend unterstützen. Die logische Erweiterung bei der Nutzung mehrerer Sichten ist die Verknüpfung dieser. Wird in einer Ansicht etwas verändert, so muss sich dies auf die restlichen Ansichten auswirken falls diese direkt oder indirekt davon betroffen sind. Diesen Sachverhalt beschreibt Jonathan C. Roberts in seiner Arbeit „State of the Art: Coordinated & Multiple Views in Exploratory Visualization“ [Rob07]. Die verschiedenen Ansichten sollten es dem Nutzer ermöglichen Antworten auf folgende Fragen zu finden:

- Welche Gestalt hat die Kaskade und welche Merkmale enthält sie?
- Wie werden Bildregionen durch die Kaskade ausgewertet?
- Welchen Einfluss haben die Merkmale bei der Auswertung?
- Wie ähnlich sind sich Regionen, die ausgewertet wurden?
- Welchen Einfluss haben Veränderungen, die der Nutzer vornimmt?

Das in dieser Arbeit entstandene interaktive Trainingssystem setzt die angesprochenen Punkte um. Zu Beginn kann der Nutzer einige Beispiele aus einem Video extrahieren. Hierbei ist nicht zwangsweise eine große Anzahl erforderlich, da der Klassifikator in mehreren Schritten trainiert und somit verbessert wird. Eine große initiale Trainingsmenge ist jedoch nicht ausgeschlossen und würde zu einem Klassifikator führen, der bereits zu Beginn keine schlechten Ergebnisse liefert. Nach dem Training mit den extrahierten Beispielen folgt die Evaluation eines Videoabschnittes. Die Güte der Detektionen wird grafisch dargestellt. Ebenso der durch das Training entstandene Klassifikator. Danach folgt das zyklische interaktive Training. Der Nutzer kann einige Detektionen auswählen und im Detail analysieren. Durch die Erkenntnisse der Analyse bildet er die nächste Trainingsmenge. Mit dieser wird der Klassifikator online trainiert. Des Weiteren kann der Nutzer mithilfe der Erkenntnisse gezielt Verbesserungen am Klassifikator vornehmen.

Das Vorgehen bei der Navigation in den Detektionsdaten entspricht hier dem Mantra auf das Ben Shneiderman in seiner Arbeit „The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations“ [Shn96] stieß:

> „overview first - zoom and filter - then details on demand“

In seiner Arbeit beschreibt er wie verschiedene Datentypen und Datenstrukturen grafisch dargestellt werden können. Ziel dabei ist, dem Nutzer die Daten verständlich zu machen. Daher soll es möglich sein, einzelne Objekte oder Strukturen in den visualisierten Daten zu erkennen bzw. nach ihnen zu suchen.

Auch Keim *et al.* setzen sich in ihrer Arbeit „Visual Analytics: Scope and Challenges“ [KMS+08] mit diesem Thema auseinander. Jedoch legen sie dabei besonderen Wert auf eine automatisierte Analyse der Daten, bevor sie visualisiert und bearbeitet werden. Nach ihren Erfahrungen und Recherchen ergibt sich für den Bereich der visuellen Analyse von Daten ein Ablauf, der durch folgendes Mantra beschrieben wird:

> „analyse first - show the important - zoom, filter and analyse further - details on demand“

Demzufolge werden die Daten zuerst untersucht. Dabei werden Daten mit bestimmten Eigenschaften extrahiert. Die Eigenschaften werden durch entsprechende Parameter festgelegt. Danach folgt ein Bearbeitungszyklus bestehend aus der grafischen Darstellung der Daten und einer genaueren Analyse dieser durch den Benutzer. Die Analyse umfasst die Möglichkeit zur Navigation in den Daten. Diese können bearbeitet werden. Zudem ist es möglich die Parameter zu verändert, welche die automatisierte Analyse verwendet. Dieser Ablauf und die grundlegenden Komponenten, der visuellen Analyse von Daten, sind in Abbildung 4.1 dargestellt.

Das Mantra bzw. dieser Bearbeitungszyklus wurde in dem hier entstandenen Rahmenwerk umgesetzt. Die Daten setzen sich zum einen aus den Detektionen und zum anderen aus dem Klassifikator bzw. seiner Merkmale zusammen. Die Darstellung der Detektionen vermittelt dem Nutzer, wie der Klassifikator Regionen bewertet hat. Die automatisierte Analyse wird durch die Auswahl der Detektionen umgesetzt. Der Nutzer stellt die Parameter dabei so

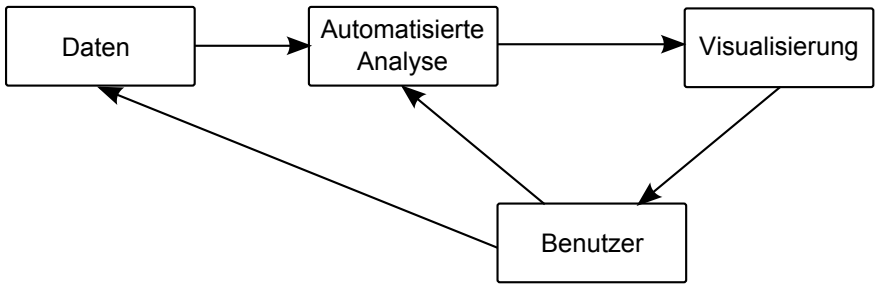


Abbildung 4.1: Dargestellt ist der Ablauf der visuellen Analyse von Daten. Zunächst werden die Daten anhand voreingestellter Kriterien analysiert. Die Daten, die den Kriterien entsprechen, werden visualisiert. Danach wird dem Benutzer eine weitere Verarbeitung ermöglicht. Die Ergebnisse dieser Verarbeitung können sowohl Auswirkung auf die automatisierte Analyse als auch die Daten selbst haben.

ein, dass Detektionen ausgewählt werden bei denen die Klassifikation nicht eindeutig ist. Für weiterführende Analysen stehen zwei weitere Ansichten zur Verfügung. Beide setzten voraus, dass zuvor eine Menge an Detektionen ausgewählt wurde. Diese können dann zum einen nach ihrer Ähnlichkeit in Clustern angeordnet werden, um einen Eindruck über die Detektionen zu erhalten. Dadurch ist es dem Nutzer möglich Elemente schnell und einfach zur nächsten Trainingsmenge hinzufügen. Zum anderen können Beispiele durch die Kaskade ausgewertet werden, um so etwas über sein Verhalten in Erfahrung zu bringen. Hierdurch kann der Nutzer den Klassifikator gezielt modifizieren. Der geschilderte Ablauf ist in Abbildung 4.2 dargestellt.

Die Komponenten des Systems, sowie deren Funktionen und Verknüpfungen, werden im Folgenden näher beschrieben.

4.1 Klassifikator

Die Aufgabe dieser Komponente besteht darin einen Kaskadenklassifikator zu trainieren, um mit diesem eine Auswertung von Bildregionen durchzuführen. Somit sollen Objekte erkannt werden, welche den Elementen der positiven Trainingsmenge entsprechen. Die Erkennung wird anhand von Rechteckmerkmalen durchgeführt. Die Verwendung eines Kaskadenklassifikators und von Rechteckmerkmalen beruht auf den positiven Resultaten der Arbeit „Robust Real-time Object Detection“ [VJ04] von Paul Viola und Michael Jones. Jedoch wurde hier ein Online Boosting statt einem Offline Boosting verwendet. Das Online Boosting ist hier, aufgrund des iterativen Trainingsablaufs mit unterschiedlich großen Trainingsmengen, besser geeignet. Dadurch ist es möglich, das Training zu unterbrechen und zu einem anderen Zeitpunkt vorzuführen.

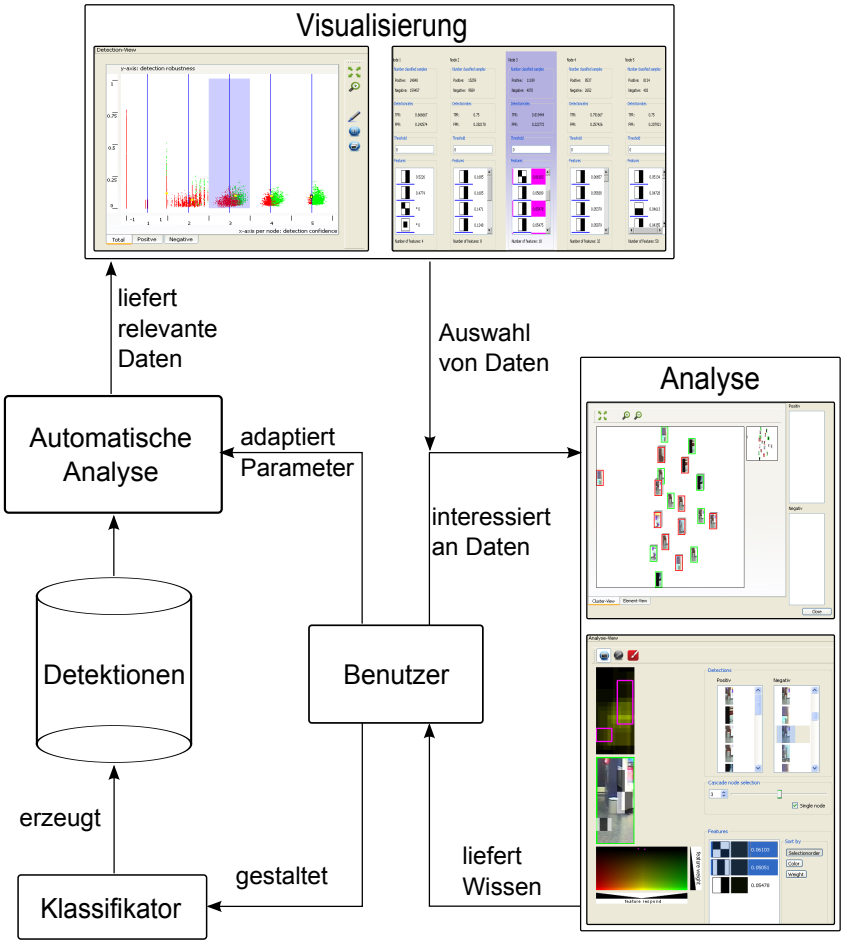


Abbildung 4.2: Dargestellt ist der Ablauf des visuellen Analyseprozesses des entstandenen System. Durch den Klassifikator werden Detektionen erzeugt. Die automatisierte Analyse selektiert relevante Daten. Diese werden visualisiert. Der Benutzer kann eine beliebige Untermenge näher untersuchen und daraus Wissen erhalten, um den Klassifikator zu gestalten oder die visualisierte Datenmenge weiter eingrenzen.

Gemäß der beschriebenen Aspekte wurden zwei online Trainingsalgorithmen für Kaskadenklassifikatoren auf ihre Tauglichkeit hin untersucht. Sie wurden ausgewählt, da mit ihnen vielversprechende Erfolge erzielt wurden.

Der Algorithmus 4.1 zur Erstellung einer Kaskade wird genutzt, um ein Objekt in einer Sequenz an Bildern zu verfolgen. Dabei wird der Klassifikator gleichzeitig trainiert. Dazu wurde eine zufällig erstellte Kaskade H, die $|H| = M$ Merkmale enthält, mit jedem Bild trainiert. Dabei wird zunächst die Confidence Map (CM) aktualisiert. Diese sagt aus, wie wahrscheinlich es ist, dass sich das Objekt in einer bestimmten Region befindet. Danach erfolgt das Training mit dem positiven Beispiel $x^+_{t,k}$. Dieses wird mithilfe der Confidence Map ermittelt, indem die Region mit der höchsten Wahrscheinlichkeit gesucht wird. λ steht für das

Algorithmus 4.1 Algorithmus zum Onlinetraining eines Kaskadenklassifikators [VSF08]

Require: Zufällig initialisierten starken Klassifikator H
Require: Menge an Merkmal F mit $|F| = |H| = M$
Require: Online Trainingsmethode für die Merkmale L_0
Require: Leere Confidence Map CM
  for $t = 1, 2, ..., T$ **do**
    for jedes Suchfenster $x_{t,k}$ in Frame$_t$ **do**
      $CM_t \leftarrow H_t(x_t)$
    end for
    $x^+_{t,k} \leftarrow \text{argmax}_k(CM_{t,k})$
    $\lambda \leftarrow 1$
    OnlineBoosting($H_t, L_0, (x^+_{t,k}, 1), \lambda$) [Oza05]
    for jedes negative Beispiel $x^-_{t,k}$ **do**
      $\lambda \leftarrow 1$
      OnlineBoosting($H_t, L_0, (x^-_{t,k}, 0), \lambda$) [Oza05]
    end for
    for $l = 1, ..., L$ **do**
      for $h_m \in H, m = 1, ..., M$ **do**
        if $e_m <= \theta_l$ **then**
          $H_l \leftarrow h_m$
        end if
      end for
    end for
  end for

initiale Gewicht des Beispiels. Entsprechend werden negative Beispiele $x^-_{t,k}$ extrahiert, indem Regionen mit einer geringen Wahrscheinlichkeit gesucht werden. Die Menge F, mit $|F| = M$ vielen Merkmalen, wird mit den Beispielen trainiert. Dafür wird ein online Lernalgorithmus L_0 verwendet. Sobald alle Merkmale trainiert wurden, erfolgt die Erstellung der neuen Kaskade. Diese besteht aus L Knoten. Jedem Knoten ist ein Schwellwert θ_l zugeordnet. Die Knoten der Kaskade werden sequenziell erzeugt, indem ein Merkmal $m \in F$ einem Knoten H_l hinzugefügt wird, falls sein Fehler $e_m \leq \theta_l$ ist. Jedes Merkmal wird genau einem Knoten zugewiesen. Wichtig dabei ist, dass die θ_l streng monoton ansteigen. Dadurch werden die guten Merkmale am Anfang der Kaskade positioniert. Von ihnen existieren oftmals nur wenige. Sie erkennen die meisten negativen Regionen. Am Ende der Kaskade werden viel Merkmale angeordnet. Sie weisen eine hohe Fehlerrate auf. Daher wird eine große Anzahl von ihnen benötigt, um eine Region zu klassifizieren.

Der Algorithmus in seiner jetzigen Form musste leicht verändert werden. Dahin gehend wurde der Mechanismus der Confidence Map entfernt, da die Positionen der Beispiele vorgegeben sind. Auch sollte es möglich sein, mehr als ein einzelnes positives Beispiel für das Training zu verwenden. Algorithmus 4.2 enthält die vorgenommenen Modifikationen.

Algorithmus 4.2 Modifizierter Algorithmus zum Onlinetraining eines Kaskadenklassifikators [VSF08]

Require: Zufällig initialisierten starken Klassifikator H
Require: Menge an Merkmal F mit $|F| = |H| = M$
Require: Online Trainingsmethode für die Merkmale L_0
  for $t = 1, 2, ..., T$ **do**
    for jedes positive Beispiel $x^+_{t,j}$ **do**
      $\lambda \leftarrow 1$
      OnlineBoosting($H_t, L_0, (x^+_{t,j}, 1), \lambda$) [Oza05]
    end for
    for jedes negative Beispiel $x^-_{t,k}$ **do**
      $\lambda \leftarrow 1$
      OnlineBoosting($H_t, L_0, (x^-_{t,k}, 0), \lambda$) [Oza05]
    end for
  end for
  for $l = 1, ..., L$ **do**
    for $h_m \in H, m = 1, ..., M$ **do**
      if $e_m <= \theta_l$ **then**
        $H_l \leftarrow h_m$
      end if
    end for
  end for

Bei näherer Betrachtung werden jedoch einige Schwachpunkte deutlich. Zum einen wird eine statische Menge an Merkmalen trainiert. Falls sich diese Merkmale zu Beginn in ungeeigneten Regionen befinden, beispielsweise nur im Hintergrund und keines auf dem Objekt selbst, so wird der Klassifikator auf den Hintergrund trainiert. Dadurch würde er nicht die gewünschten Objekte erkennen. Zum anderen die geringe Anzahl der Merkmale. Das Training erfolgte in der Arbeit [VSF08] von Ingrid Visentini *et al.* mit 100, 200 und 400 Merkmalen. Wohingegen Paul Viola und Michael Jones für einen Klassifikator mit 32 Knoten 4297 Merkmale nutzten. Des Weiteren kann die Anzahl der Merkmale pro Knoten nicht beeinflusst werden. Sie ergibt sich durch die Fehlerraten der Merkmale.

Diese Probleme löst die Methode von Helmut Grabner und Horst Bischof, welche in ihrer Arbeit „On-line Boosting and Vision“ [GB06] beschrieben ist. Dabei handelt es sich um einen online Lernalgorithmus für das Training eines monolithischen Klassifikator. Dies entspricht dem Training eines Knotens einer Kaskade. Bei dieser Methode werden während des Trainings regelmäßig neue Merkmale betrachtet. Zudem ist es möglich, die Anzahl der Merkmale pro Knoten zu regulieren. Der Ablauf des Trainings ist in Algorithmus 4.3 dargestellt.

Ein Knoten wird dabei mit einem Beispiel der Trainingsmenge trainiert. Das Gewicht λ dieses Beispiels wird zu Beginn auf den Wert eins gesetzt. λ kann als Kennwert dafür angesehen werden, wie gut das aktuelle Beispiel bisher gelernt wurde. Um einen Knoten zu trainieren, erfolgt das sequenzielle Training jedes Selektors des Knotens mit dem Beispiel

Algorithmus 4.3 Algorithmus zum Onlinetraining eines Klassifikators [GB06]

Require: Beispiel $(x, y), y \in -1, 1$
Require: Starken Klassifikator h^{strong}
Require: Statistikvariablen $\lambda^{corr}_{n,m}, \lambda^{wrong}_{n,m}$
  Initiales Gewicht des Beispiels $\lambda \leftarrow 1$
  for Selektoren $n = 1, ..., N$ **do**
    for Merkmale $m = 1, ..., M$ des Selektors n **do**
      $h^{weak}_{n,m} \leftarrow$ update$(h^{weak}_{n,m}, (x, y), \lambda)$
      if $h^{weak}_{n,m}(x) = y$ **then**
        $\lambda^{corr}_{n,m} \leftarrow \lambda^{corr}_{n,m} + \lambda$
      else
        $\lambda^{wrong}_{n,m} \leftarrow \lambda^{wrong}_{n,m} + \lambda$
      end if
      $e_{n,m} \leftarrow \frac{\lambda^{wrong}_{n,m}}{\lambda^{corr}_{n,m} + \lambda^{wrong}_{n,m}}$
    end for
    $m^+ \leftarrow$argmax$_m(e_{n,m})$
    $e_n \leftarrow e_{n,m}$
    $h^{sel}_n \leftarrow h^{weak}_{n,m^+}$
    if $e_n = 0$ oder $e_n > \frac{1}{2}$ **then**
      Abbruch
    end if
    $\alpha_n \leftarrow \frac{1}{2} \ln(\frac{1-e_n}{e_n})$
    if $h^{sel}_n(x) = y$ **then**
      $\lambda \leftarrow \lambda * \frac{1}{2(1-e_n)}$
    else
      $\lambda = \lambda * \frac{1}{2e_n}$
    end if
    $m^- \leftarrow$ argmax$_m(e_{n,m})$
    Ersetze m^- durch neues Merkmal m^*
    $\lambda^{corr}_{n,m^*} \leftarrow 1$
    $\lambda^{wrong}_{n,m^*} \leftarrow 1$
  end for

und seinem Gewicht λ. Ein Selektor besteht aus einer Menge an Merkmalen. Sein Training erfolgt, indem jedes ihm zugehörige Merkmal mit dem Beispiel und λ trainiert wird. Hierfür sollte ein online Lernalgorithmus verwendet werden. Dieser adaptiert den Schwellwert eines Merkmals unter Berücksichtigung von λ. Eine Möglichkeit dafür wird in [GB06] angegeben. Dabei werden mithilfe zweier Gaußverteilungen $\mathcal{N}^-(\mu^-, \sigma^-)$ und $\mathcal{N}^+(\mu^+, \sigma^+)$ die Wahrscheinlichkeiten $P^+(1|f_j(x))$ und $P^-(-1|f_j(x))$ modelliert. Diese drücken aus, wie wahrscheinlich es ist, dass das Merkmal f_j eine Region x als positiv bzw. negativ einstuft. Das Training erfolgt durch die Anwendung eines Kalman-Filters auf die entsprechende

Gaußverteilung. Ist das Beispiel positiv, so wird $\mathcal{N}^+$ adaptiert sonst $\mathcal{N}^-$. Für jedes Merkmal wird dadurch ein Schwellwert für die positiven (μ^+) und negativen (μ^-) Beispiele erlernt.

$$K_t = \frac{P_{t-1}}{P_{t-1} + R}$$

$$\mu_t = K_t * f_j(x) + (1 - K_t) * \mu_{t-1}$$

$$\sigma_t^2 = K_t * (f_j(x) - \mu_t)^2 + (1 - K_t) * \sigma_{t-1}^2$$

$$P_t = (1 - K_t) * P_{t-1}$$

R entspricht dabei einem Rauschen und wird mit 0.01 initialisiert. P_0 erhält zu Beginn den Wert 1000, μ_0 den Wert null und σ_0^2 ebenso. λ kann nun auf zwei Arten einfließen. Entweder kann ein Merkmal k mal trainiert werden wobei $k \sim \text{Poisson}(\lambda)$ gewählt wird. Oder es fließt als Lernrate ein. Dabei erfolgt zunächst die Berechnung der Differenz μ_{diff}. Sie entspricht der Entfernung zum geschätzten neuen Schwellwert μ_t. Die Schätzung erfolgt durch den Kalman-Filter. Danach erfolgt die Berechnung des neuen Schwellwerts $\hat{\mu_t}$. Dabei wird der letzte Schwellwert $\hat{\mu_{t-1}}$ in die Richtung μ_t verschoben.

$$\mu_{diff} = \mu_t - \hat{\mu_{t-1}}$$

$$\hat{\mu_t} = \hat{\mu_{t-1}} + \mu_{diff} * \lambda$$

Aus μ^+ und μ^- wird schließlich der Gesamtschwellwert θ_j errechnet.

$$\theta_j = \frac{|\mu^+ + \mu^-|}{2}$$

Außerdem wird die Statistik des Merkmals aktualisiert. Diese drückt aus, wie gut das Merkmal die Beispiele einordnet, mit denen es bis zu diesem Zeitpunkt trainiert wurde. Nachdem das Training eines Selektors abgeschlossen ist, wird das Merkmal h_{n,m^+}^{weak} mit dem geringsten Fehler $e_{n,m}$ ermittelt und als Knotenmerkmal markiert. Pro Selektor existiert zu jedem Zeitpunkt maximal eines. Mit diesem besten Merkmal wird nun das Gewicht λ adaptiert. Kann das Beispiel durch das Merkmal korrekt erkannt werden, so wird λ verringert sonst erhöht. Des Weiteren wird das Merkmal mit dem größten Fehler durch ein neues ersetzt. Dadurch ist es möglich im Laufe der Zeit auf Merkmale zu stoßen, welche die meisten Elemente korrekt klassifizieren. Mit den markierten Merkmalen wertet ein Knoten nun Bildregionen aus. Durch die Anzahl der Selektoren kann die Anzahl der Merkmale eines Knotens reguliert werden. Abbildung 4.3 zeigt die Veränderung der besten Merkmale der Selektoren eines Knotens durch das Training mit einem Beispiel.

Aber auch hier wurden einige Probleme deutlich. Dies betrifft zum einen das Gewicht λ. Es wird zu Beginn des Trainings für jedes Beispiel auf den Wert eins gesetzt. Das bedeutet, dass jedes Beispiel am Anfang denselben Einfluss auf das Training haben soll. Betrachtet man das Training eines Merkmals mit einer positiven Menge bestehend aus einem Element und einer negativen Menge mit 99 Elementen, dann kann der Fall auftreten, dass alle negativen Elemente korrekt als falsch erkannt werden und das eine positive ebenso. Dadurch würden sich für die Statistikvariablen die Werte 99 für $\lambda_{correct}$ und eins für λ_{wrong} ergeben. Der

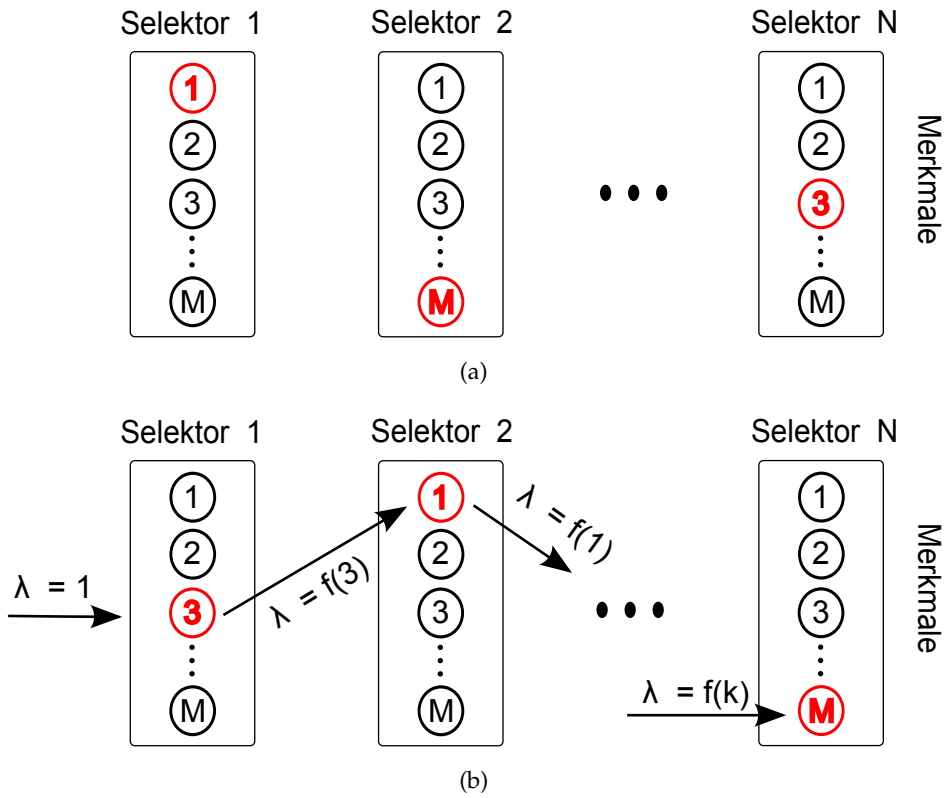


Abbildung 4.3: Abbildung (a) zeigt den Zustand der Selektoren eines Knotens vor dem Training mit einem Beispiel. Die rot gekennzeichneten Merkmale werden momentan zur Auswertung von Bildregionen genutzt. Abbildung (b) zeigt den Trainingsablauf mit einem Trainingsbeispiel. Die Selektoren werden sequenziell trainiert. In jedem Selektor wird jedes Merkmal trainiert. Das Merkmal mit den geringsten Fehler wird als Knotenmerkmal markiert. Das Gewicht des Beispiels λ wird abhängig vom markierten Merkmal adaptiert.

resultierende Fehler beträgt daher $\frac{1}{99+1} = 0.01$. Aufgrund des geringen Fehlers wird dieses Merkmal mit hoher Wahrscheinlichkeit als Kontenmerkmal markiert. Sollte die maximale Anzahl der Merkmale des Knotens auf eins beschränkt sein, so würde dieses Merkmal das positive Beispiel als falsch aussondern. Dies wäre jedoch suboptimal. Um dies zu vermeiden, wird λ an die Trainingsmengen angepasst. Jedes Element der größeren Menge M^- erhält weiterhin das Gewicht eins. Die Gewichte der kleineren Menge M^+ werden auf den Wert $\frac{|M^-|}{|M^+|}$ gesetzt. In dem oben beschriebenen Beispiel würde sich so für $\lambda_{correct}$ und λ_{wrong} der Wert 99 ergeben. Damit ergibt sich ein Fehler von $\frac{99}{99+99} = 0.5$. Dadurch kommt das Merkmal laut Algorithmus nicht für die Markierung infrage.

Dies führt direkt zum nächsten Problem. Dem Markieren der Merkmale. Bisher wird das Merkmal mit dem geringsten Fehler markiert solange dieser ungleich null und kleiner als 0.5 ist. Dabei könnte es jedoch passieren, dass ein Merkmal, das gerade erst als Ersatz für ein

schlechtes Merkmal neu hinzugekommen ist, markiert wird. Falls die geschieht, nachdem es bereits mit vielen Beispielen trainiert wurde, kann dieser Wahl vertraut werden. Wurde es jedoch nur mit einigen wenigen Beispielen trainiert und sollte der Fehlerwert sehr gering sein, dann sollte dem nicht viel Vertrauen geschenkt werden. Um die Auswahl solcher Merkmale zu vermeiden, werden nur Merkmale gewählt, die mit einer Mindestanzahl an Beispielen trainiert wurden und deren Fehlerrate im Toleranzbereich liegt. Die minimale Anzahl von Beispielen mit denen ein Merkmal trainiert werden muss, beträgt hier 100. Dieser Wert hat sich durch Tests ergeben. Der Toleranzbereich wird anhand der Trainingsvorgabe für die maximale Fehlerrate eines Knotens bestimmt. Diese wird hier auf 0.4 gesetzt. Dadurch ergibt sich ein Toleranzbereich von $[0.3, 0.5]$. Theoretisch könnte hier für die maximale Fehlerrate jeder Wert aus dem Intervall $]0, 0.5[$ genutzt werden. Hierdurch würde der resultierende Knoten statistisch gesehen mindestens 50% der negativen Beispiele aussonder. In [VJ04] wird dieser Wert beispielsweise auf 0.3 gesetzt.

Als Nächstes wurde der Algorithmus dahingehen erweitert, dass es ihm möglich ist, die Anzahl der markierten Merkmale pro Knoten zu verändern. Die Adaption erfolgt auf Grundlage der Statistikvariablen der Knoten. Diese werden vor dem Training mit einem Beispiel aktualisiert, indem getestet wird ob der Knoten das Beispiel korrekt klassifiziert. Anhand von ihnen wird der Fehler eines Knotens berechnet. Fall dieser im Toleranzbereich liegt, bleibt die Anzahl gleich. Falls er darüber liegt, wird sie erhöht sonst verringert.

4.2 Beispielextraktion

Diese Komponente dient zur Extraktion von Beispielmengen aus einem Video. Dabei ist es möglich, jedes Frame des Videos über einen Schieberegler bzw. über die Eingabe der gewünschten Framenummer anzuwählen. Die Auswahl eines Beispiels erfolgt, indem ein Bereich markiert wird. Dafür kann mithilfe der Maus ein Rechteck aufgespannt werden. Die ausgewählte Region wird zur Beispielmenge hinzugefügt. Dies ist rechts in Abbildung 4.4 zu sehen. Gespeichert wird dabei Framenummer, Höhe, Breite sowie der Ankerpunkt des Rechtecks. Die Zuordnung zu einer der beiden Beispielmengen erfolgt durch die Schaltflächen „Positive samples“ und „Negative samples“. Ist, wie in Abbildung 4.4 dargestellt, Letztere ausgewählt, so wird das Beispiel in diese Menge eingefügt. Das gesetzte Rechteck wird in geeigneter Farbe in das dargestellte Frame eingezeichnet. Für einen Überblick über die Anzahl der Elemente in den Mengen sorgt die Anzeige direkt unterhalb der Schaltflächen. Um schnell eine große Anzahl an negativen Beispielen erzeugen zu können, wurde die Schaltfläche „Generate negative samples“ hinzugefügt. Sobald diese betätigt wird, erfolgt die Erzeugung der angegebenen Anzahl. Diese wird durch das Eingabefeld, das sich rechts neben der Schaltfläche befindet, spezifiziert. Um Elemente der Mengen im Frame wiederzufinden, können sie hervorgehoben werden. Dazu wird ein Element der Listen selektiert. Falls sich das Element nicht im aktuellen Frame befindet, so wird das benötigte Frame geladen. Dann werden alle in ihm markierten Bereiche angezeigt und das ausgewählte Element hervorgehoben. Falls ein Element nicht länger Teil einer Menge sein soll, so kann es gelöscht werden. Dazu wird es zuvor in der Liste markiert und im Anschluss darauf durch das Betätigen der Entfernen-Taste gelöscht.

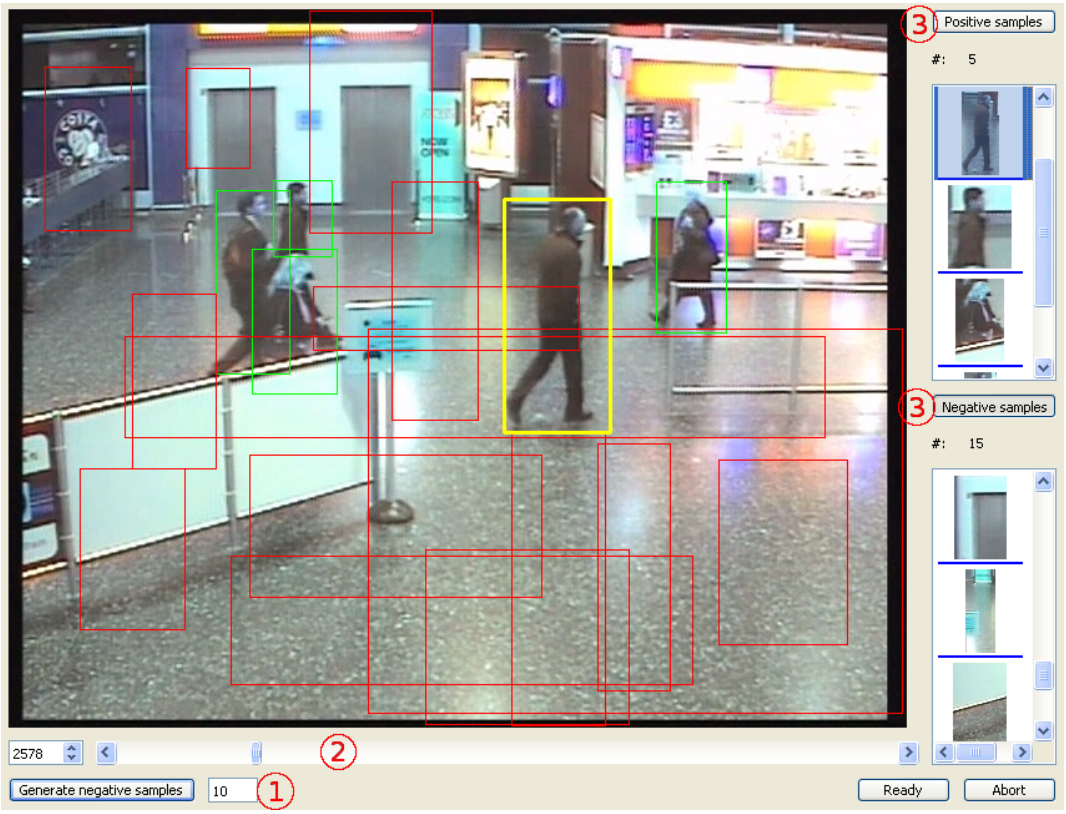


Abbildung 4.4: Dargestellt ist das Extraktionswerkzeug während des Auswahlprozesses. Die Nummern entsprechen Funktionsblöcken. 1. Besteht aus einem Eingabefeld für die Anzahl zu erzeugender negativer Beispiele und der Schaltfläche „Generate negative samples“ 2. Dient zur Auswahl eines Frames durch einen Schieberegler oder über Direktanwahl durch das Eingabefeld links 3. Entspricht jeweils einer Liste an Beispielen, sowie einem Zähler für die Anzahl der Elemente der Liste und einer Schaltfläche durch die bestimmt werden kann, in welche Liste eine ausgewählte Region eingefügt wird. Im dargestellten Frame sind bereits selektierte Bereiche zu sehen. Rot entspricht negativen und grün positiven Elementen. Das gelbe Rechteck ist das Element, das in der positiven List markiert wurde.

4.3 Detektions-Ansicht

In dieser Ansicht wird dargestellt, wie der Klassifikator die ausgewerteten Bildregionen (Detektionen) eines Videoabschnittes bewertet. Im Detail bedeutet dies, dass die Reaktionen der Knoten auf die Detektionen visualisiert werden. Die Reaktion eines Knoten wird durch das Tupel (y_{robust}, x_{conf}) mit $y_{robust} \in [0,1]$ und $x_{conf} \in [-1,1]$ dargestellt. Dabei entspricht x_{conf} der Detektionssicherheit und y_{robust} der Detektionsrobustheit.

$$y_{robust} = \hat{y} * \tilde{y}$$

$$\hat{y} = \frac{\sum_{m \in K_i} -\log(\text{Gewicht}(m))}{-\log(w_{min}) * k}$$

$$\tilde{y} = \frac{\sum_{m \in K_i} \text{Abstand}(m)}{k}$$

$$w_{min} = \underset{\text{Merkmal } m \text{ des Knotens } K_i}{\text{argmin}} (\text{Gewicht}(m))$$

$$k = \text{Merkmalsanzahl}(K_i)$$

$$\text{Abstand}(m) = \frac{f(m) - th_m}{\text{Maximalabstand}(m, th_m)}$$

y_{robust} setzt sich aus der Gewichtsrobustheit $\hat{y}$ und der Auswertungsrobustheit $\tilde{y}$ der Merkmale eines Knotens K_i zusammen. $\hat{y}$ kann als Maß für die Gleichheit der Gewichte angesehen werden. Hat jedes Merkmal dasselbe Gewicht, so ist $\hat{y} = 1$. Diese Komponente ist für y_{robust} wichtig, da die Auswertung eines Knotens nicht maßgeblich von einem Merkmal abhängen sollte. Den schon ein geringes Rauschen in der Bildregion könnte dazu führen, dass positive Region als negativ erkannt wird und umgekehrt. $\tilde{y}$ soll darstellen, wie sicher die Merkmale bei der Auswertung waren. Falls die Auswertungen $f(m_i)$ der Merkmale m_i sehr nahe an ihren Schwellwerten th_{m_i} liegen, so könnte auch hier ein Rauschen für ein Schwanken der Merkmalantworten $F(m_i)$ sorgen.

$$F(x) = \begin{cases} 1, & \text{wenn } f(x) \geq th_m \\ 0, & \text{sonst} \end{cases}$$

$$x_{conf} = \frac{\sum_{m \in K_i} (\text{Gewicht}(m) * F(m))}{\sum_{m \in K_i} (\text{abs}(\text{Gewicht}(m) * F(m)))}$$

x_{conf} kann als Maß für die Sicherheit angesehen werden, mit der der Knoten K_i eine Bildregion ausgewertet hat. Sollte $x_{conf} = 1$ sein so haben alle Merkmale die Region als positiv eingestuft.

Eine Detektion enthält für jeden Knoten, mit dem eine Auswertung stattfand, ein solches Tupel. Entsprechend dieser Tupel werden Punkte gezeichnet. Das i-te Tupel entspricht dabei einem Punkt, der bei Knoten i gezeichnet wird. Falls alle Knoten der Kaskade die Detektion als positiv erkennen, so erhält jeder Punkt die Farbe grün. Erkennt ein Knoten die Detektion als negativ, so werden die Punkte mit Rot eingezeichnet.

Anhand der Visualisierung ist es dem Benutzer möglich, Regionen der Unsicherheit ausfindig zu machen. Diese enthalten Detektionen, die unsicher klassifiziert wurden. Mithilfe der gegebenen Funktionen können Detektionen in solchen Regionen näher untersucht werden.

Die Menüleiste, die rechts in Abbildung 4.6 zu sehen ist, stellt folgende Funktionen zur Verfügung:

Zoom zur Standardansicht: Ermöglicht die Darstellungen aus Abbildung 4.5 wiederherzustellen.

Regionen vergrößern: Erlaubt es einen zuvor markierten Bereich in einer vergrößerten Ansicht zu betrachten.

Region hervorheben: Detektionen in der ausgewählter Region werden in allen Knoten mit schwarz markiert.

Cluster-Ansicht: Detektionen eines Bereichs werden in Clustern nach ihrer Ähnlichkeit angeordnet. Dies ist in Abschnitt 4.5 dargestellt.

Analyse-Ansicht: Analyse der ausgewählten Detektionen wird gestartet. Dies ist in Abschnitt 4.6 dargestellt.

Über die Schaltflächen „Total“, „Positive“ und „Negative“ kann der Benutzer zwischen drei verschiedenen Darstellungen der Detektionen gewechselt werden. Diese sind in Abbildung 4.5 zu sehen. Jede der Funktionen steht in jeder Darstellung zur Verfügung.

Die Detektions-Ansicht ist mit der Cluster- und Analyse-Ansicht verknüpft. Wird in einer der beiden Ansichten eine Detektion selektiert, so werden die entsprechenden Punkte in den drei Darstellungen gelb hervorgehoben. Des Weiteren erhalten Knoten einen blauen Hintergrund, falls diese in der Analyse-Ansicht untersucht werden. Abbildung 4.6 zeigt die Auswirkungen dieser Verknüpfungen.

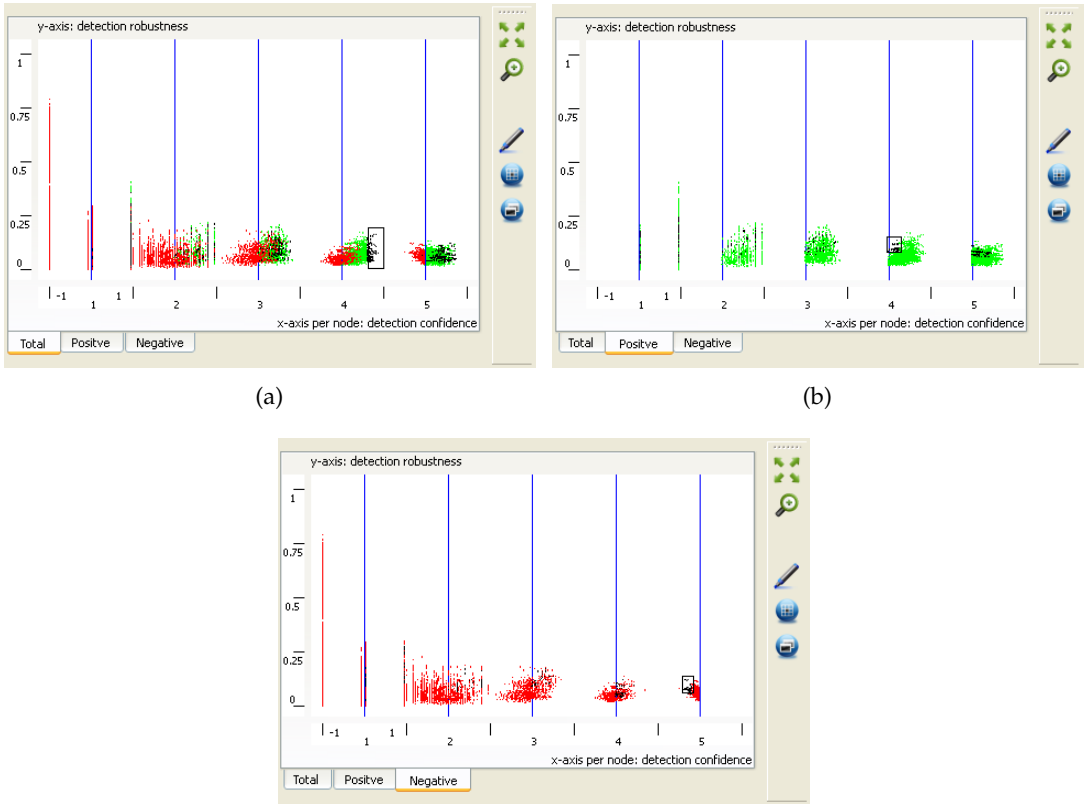


Abbildung 4.5: Zusehen sind drei Darstellungen der Detektionen, die über die Schaltflächen „Total“, „Positive“ und „Negative“ angewählt werden. In (a) werden sowohl positive als auch negative Detektionen eingezeichnet. (b) enthält nur Positive (grün) und (c) nur Negative (rot). Die blauen vertikalen Linien entsprechen den Schwellwerten der Knoten. Des Weiteren ist in jeder Darstellung ein Bereich (schwarzes Rechteck) durch den Benutzer markiert. Detektionen in diesem wurden in allen Knoten hervorgehoben (schwarz). Jeder Knoten verfügt über eine x-Achse mit dem Wertebereich $[-1, 1]$. Diese entspricht der Detektionssicherheit des Knotens. Die y-Achse ist für jeden Knoten identisch und hat den Wertebereich $[0, 1]$. Sie repräsentiert die Detektionsrobustheit.

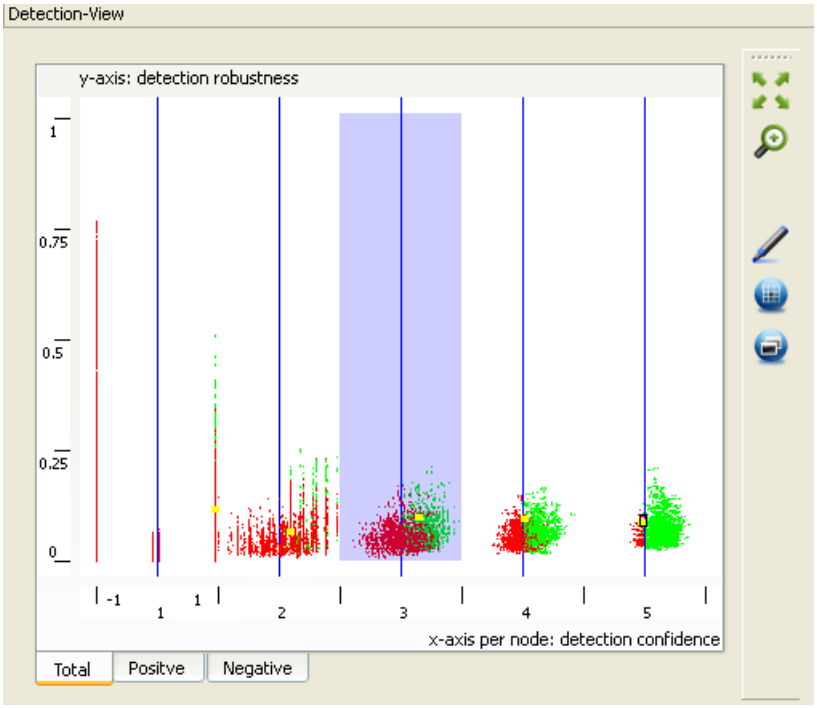


Abbildung 4.6: Gelb hervorgehobene Punkte entsprechen einer Detektionen, die entweder in der Analyse- oder Cluster-Ansicht selektiert wurde. Knoten mit blauem Hintergrund werden momentan in der Analyse-Ansicht genutzt, um eine Detektion auszuwerten.

4.4 Kaskaden-Ansicht

Durch die Kaskaden-Ansicht wird dem Benutzer der Klassifikator dargestellt. Dadurch kann er Veränderungen wahrnehmen, die das Training bewirkt hat. Des Weiteren wird ihm ermöglicht, den Klassifikator zu modifizieren. Abbildung 4.7 zeigt den Klassifikator während der Analyse ausgewählter Detektionen durch die Analyse-Ansicht.

Ein Knoten enthält vier Informationsgruppen:

„Number of classified samples“: Dieses Feld beinhaltet die Anzahl der Bildregionen, die der Knoten bei der letzten Evaluation eines Videoabschnittes ausgewertet hat.

„Detectionrates“: Hier wird die Detektions– und Falschdetektionsrate (TPR und FPR) eines Knoten aufgezeigt. Berechnet werden sie anhand der Statistikvariablen des Knotens. Diese werden während des Trainings aktualisiert.

„Threshold“: Dieses Feld beinhaltet den Schwellwert des Knotens. Der Benutzer kann ihn durch Eingabe einer Zahl im Intervall $[-1.0, 1.0]$ verändern. Die Veränderung wird ab der nächsten Evaluation berücksichtigt.

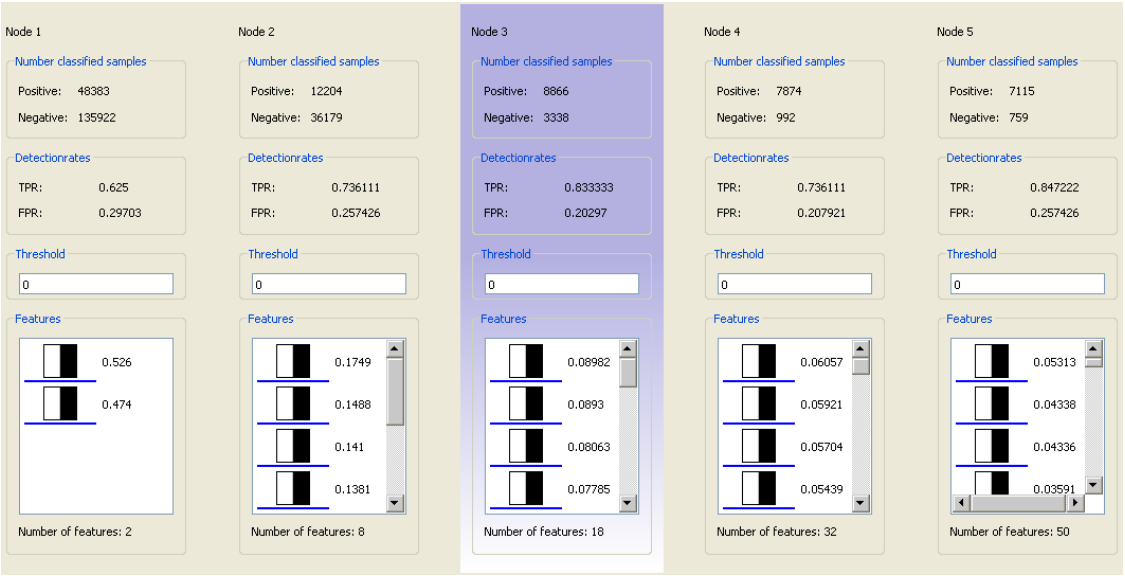


Abbildung 4.7: Dargestellt ist der Klassifikator während der Analyse ausgewählter Detektionen durch die Analyse-Ansicht. Der blau hervorgehobene Knoten ist in den Analyseprozess involviert.

„Features“: Hier werden Informationen über die Merkmale bereitgestellt, die der Knoten zur Klassifikation verwendet. Ein Element der Merkmalliste besteht aus einer Miniaturansicht des Merkmaltyps sowie dem assoziierten Gewicht. Durch einen Rechtsklick öffnet sich ein Kontextmenü. Zum einen werden hier Funktionen zur Modifikation der Liste bereitgestellt. Zum anderen ermöglicht das Menü das Abrufen von Merkmaldetails. Das Kontextmenü ist in Abbildung 4.8 dargestellt.

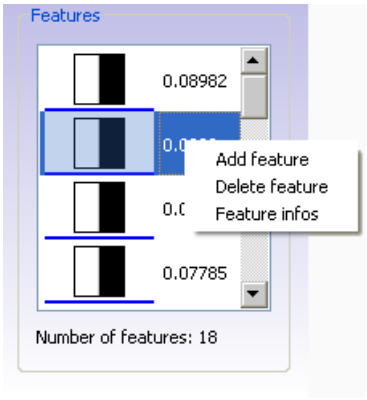
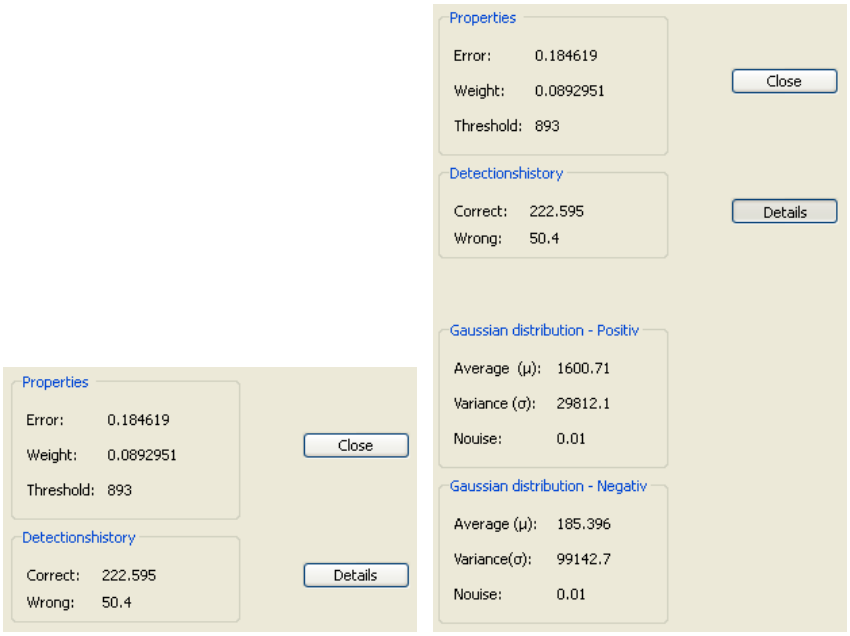


Abbildung 4.8: Kontextmenü zur Bearbeitung der Merkmalliste eines Knotens.

Im Folgenden werden die durch das Kontextmenü bereitgestellten Funktionen erläutert:

„Delete feature“: Damit wird das ausgewählte Merkmal aus dem Knoten gelöscht.

„Feature Info“: Damit können weitere Informationen wie z.B. der Fehler, das Gewicht, der Schwellwert oder die Statistikvariablen, über ein Merkmal abgerufen werden. Dies ist in Abbildung 4.9 dargestellt.



(a) (b)

Abbildung 4.9: (a) entspricht der Darstellung der Merkmalinformationen. (b) zeigt die erweiterte Ansicht. Darin sind Informationen über die Gaußverteilungen zusehen, die beim Training verwendet werden. Sie werden durch das Betätigen der Schaltfläche „Details“ eingeblendet.

„Add feature“: Durch diesen Menüpunkt kann ein neues Merkmal zur Liste hinzugefügt werden. Abbildung 4.10 zeigt den verwendeten Dialog für die Erstellung.

Die Kaskaden-Ansicht ist auf zwei Arten mit der Analyse-Ansicht gekoppelt. Zum einen ist es möglich Merkmale eines Knotens für die Auswertung ihres Verhaltens per „Drag&Drop“ in die Analyse-Ansicht einzufügen. Der Hintergrund der ausgewählten Merkmale wird verändert, damit der Nutzer erkennt, welche Merkmale aus welchem Knoten momentan untersucht werden. Es ist nicht möglich zeitgleich Merkmale aus zwei verschiedenen Knoten zu untersuchen. Abbildung 4.11 stellt diesen Vorgang dar. Zum anderen werden Knoten, deren Verhalten momentan von der Analyse-Ansicht untersucht wird, farblich hervorgehoben. Dies ist in Abbildung 4.7 dargestellt.

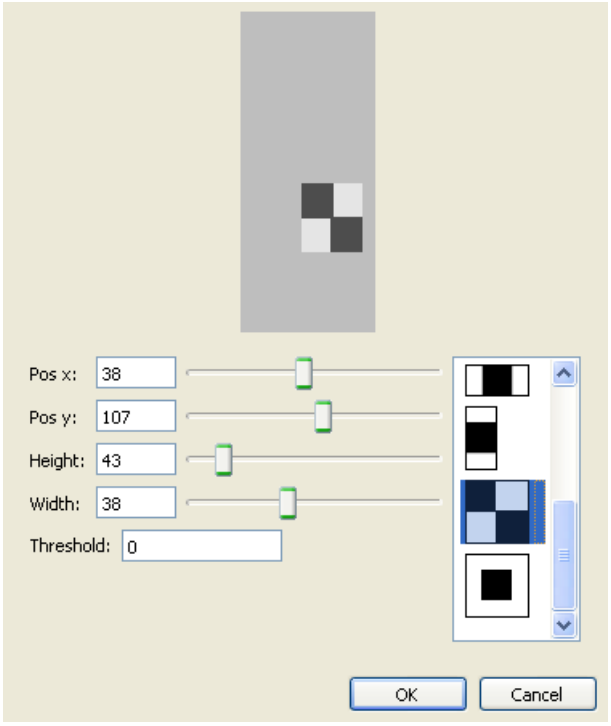
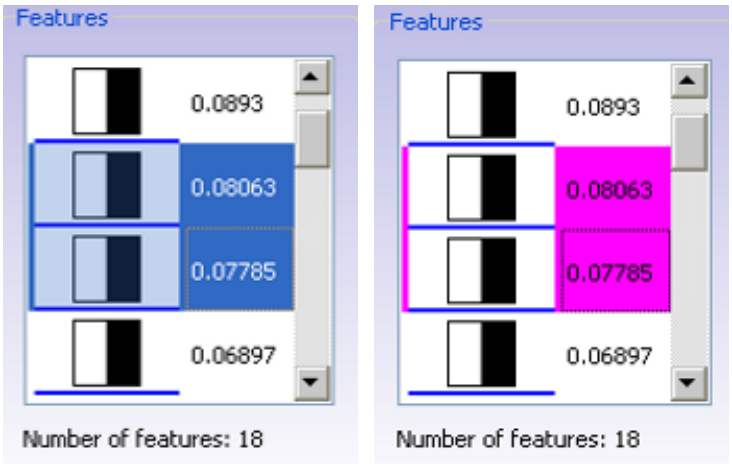


Abbildung 4.10: Dargestellt ist der Dialog, mit dem ein Merkmal erstellt werden kann. Durch die Schieberegler ist sowohl die Position als auch Höhe und Breite veränderbar. Die Auswahlliste ermöglicht, den Typ des Merkmals zu verändern. Zudem kann ein Schwellwert für das Merkmal festgelegt werden. Der graue Bereich entspricht einer vergrößerten Darstellung der durchschnittlichen Größe der positiven Regionen, mit denen bisher trainiert wurde.



(a) vorher (b) nachher

Abbildung 4.11: (a) stellt die Merkmalliste vor dem „Drag&Drop“ da, (b) danach.

4.5 Cluster-Ansicht

In dieser Ansicht werden durch den Benutzer ausgewählten Detektionen gemäß ihrer Ähnlichkeit in Clustern angeordnet. Zum einen wird dadurch aufgezeigt, welche Arten an Region klassifiziert wurden. Zum anderen ist es hier möglich, die Trainingsmengen zu erweitern. Die Ansicht besteht aus drei Funktionskomponenten:

Visualisierung: Sie stellt die Grundlage für die Navigation da und muss dahin gehend Strukturen der Detektionen aufzeigen.

Navigation: Sie ermöglicht die Betrachtung einer Untermenge der Detektionen.

Erweiterung der Trainingsmengen: Der Benutzer ist hier in der Lage die Trainingsmengen schnell und einfach zu erweitern.

Die Struktur, die in den Detektionen aufgezeigt werden soll, ist die Ähnlichkeit dieser. Die Idee dabei ist, ähnliche Detektionen bzw. deren korrespondierende Bildbereiche in gleichen Regionen einer zweidimensionalen Ebene anzuordnen. Demzufolge wird hier ein Algorithmus benötigt, der ein Bild auf eine zweidimensionale Koordinate abbildet. Ein Bild kann als Vektor in einem mehrdimensionalen Raum angesehen werden. Die Elemente des Vektors entsprechen dabei beispielsweise einer Aneinanderreihung der Pixelwerte des Bildes. Demzufolge müsste hier eine Dimensionsreduzierung stattfinden, wobei die Ähnlichkeit der Vektoren im mehrdimensionalen Raum im zweidimensionalen Zielraum möglichst nicht verloren gehen soll. Zu diesem Zweck wurde hier der t-SNE Algorithmus (siehe Abschnitt 2.4) verwendet. Auf den zweidimensionalen Koordinaten wird nun ein Cluster-Algorithmus angewandt, um jedes Element einem Clusterzentrum zuzuweisen. Diese werden angezeigt, falls sie sich in der momentan betrachteten Region befinden.

Der Cluster-Algorithmus arbeitet wie folgt. Für jede Koordinate werden die Nachbarn ermittelt, die sich in der Umgebung mit Radius r befinden. Da die angezeigten Bilder auf eine Größe von 40×40 Pixel skaliert werden, wird hier $r = \sqrt{(2 * 40^2)}$ gesetzt. Auf diese Weise entstehen viele Cluster, deren Schnittmengen überwiegend nicht leer sind. Im nächsten Schritt wird jedes Element genau einem Cluster zugeordnet. Dafür erfolgt zunächst für jedes Cluster die Berechnung der Kerndichte und der durchschnittlichen Entfernung der Elemente zum Zentrum. Daraufhin werden die finalen Cluster gebildet. Dabei wird iterativ das Cluster mit der größten Kerndichte selektiert und als ausgewählt markiert. Statt der Kerndichte kann auch die kleinste durchschnittliche Entfernung zum Zentrum für die Auswahl genutzt werden. Falls ein Element des Clusters auch Teil eines anderen ist, so wird es aus Letzterem entfernt. Der Algorithmus terminiert, sobald alle Elemente markiert sind. Daraufhin werden die finalen Cluster gezeichnet. Dabei wird eine Detektion nicht mehr abstrakt als Punkt, sondern als Miniaturansicht der korrespondierenden Bildregion dargestellt. Um zu vermeiden das sich Elemente in der Ansicht überdecken, wird lediglich das Zentrum jedes Clusters eingezeichnet. Dies ist in Abbildung 4.12 auf der linken Seite zu sehen. Die Zentren der Cluster können sich nicht überlagern, da ihr minimaler Abstand r passend gewählt wurde. Der soeben geschilderte Ablauf ist in Abbildung 4.13 dargestellt.

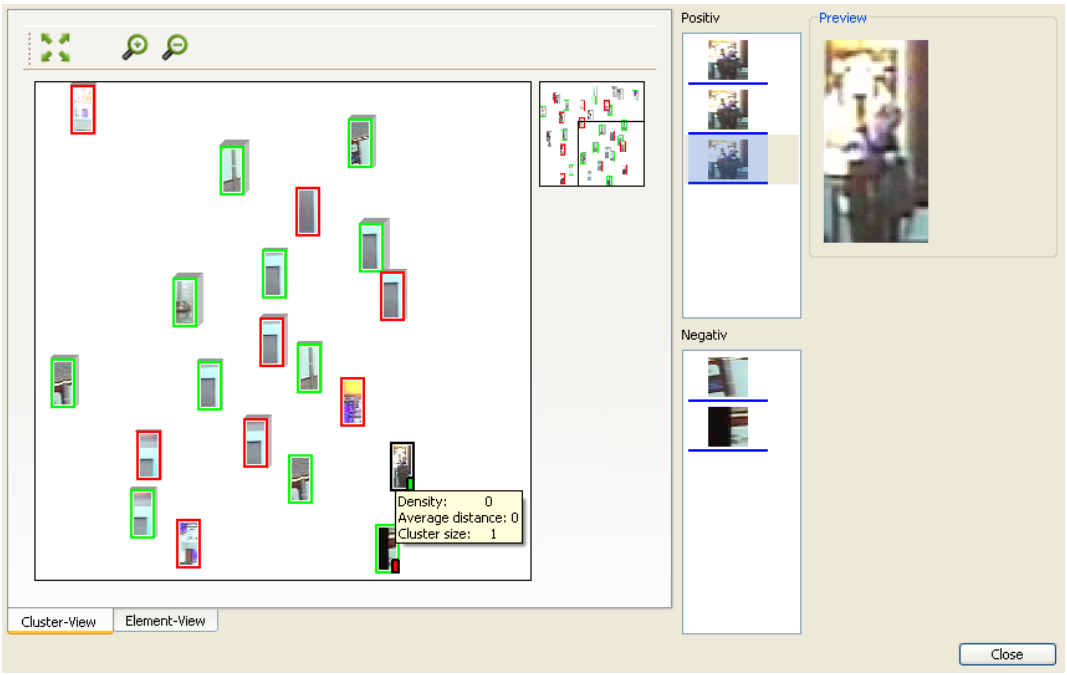
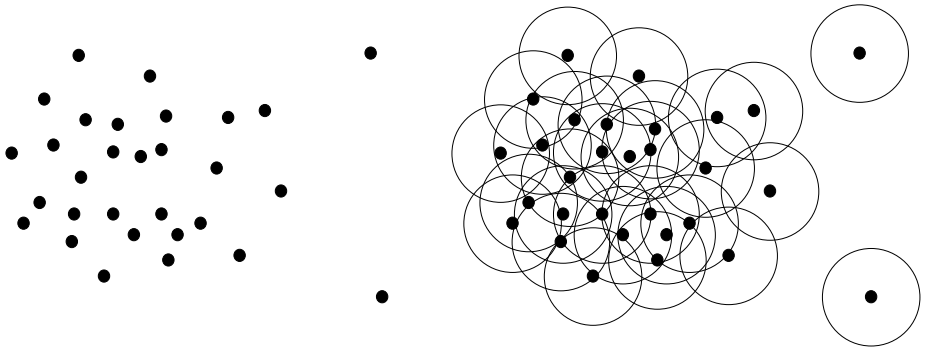


Abbildung 4.12: Diese Ansicht enthält eine Menüleiste (oben rechts) mit Schaltflächen für die Wiederherstellung der initialen Ansicht und für das vergrößern bzw. verkleinern der aktuell betrachteten Region. Durch eine Übersichtskarte (oben mittig) erhält man Auskunft über die Größe und Position der betrachteten Region (links). In dieser sind Repräsentanten der Cluster dargestellt. Positive Cluster sind mit einem grünen Rahmen markiert, negative mit einem roten. Ein schwarzer Rahmen zeigt an, dass dieses Element vergrößert abgebildet ist (rechts im Bild). Ein grünes Rechteck im Repräsentanten (unten rechts) bedeutet, dass dieses Element der positiven Trainingsmenge hinzugefügt wurde. Ein Rotes, dass es in der negativen Trainingsmenge enthalten ist. Die positive Trainingsmenge enthält hier drei Elemente, die Negative zwei. Zudem ist ein Tooltip abgebildet, der Informationen über das Cluster beinhaltet, über dem sich der Mauszeiger befindet.

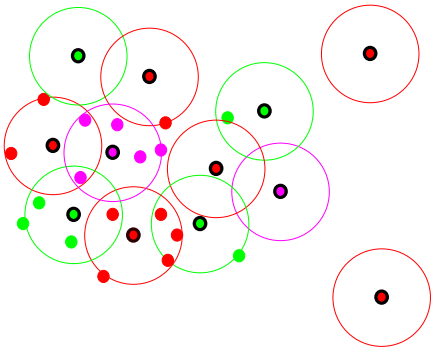
Zu jedem Cluster können per Tooltip Informationen angezeigt werden. Dies ist in Abbildung 4.14 dargestellt.

Die Navigation wird durch das Zusammenwirken von drei Komponenten realisiert. Diese sind im Folgenden aufgelistet:

- Dem Darstellungsbereich für die momentan betrachtete Region.
- Die Menüleiste.
- Die Übersichtskarte.



(a) t-SNE 2D-Koordinaten (b) Bildung möglicher Cluster



(c) Bildung finaler Cluster

Abbildung 4.13: Abgebildet ist der Ablauf des Cluster-Algorithmus. (a) entspricht den zweidimensionalen Koordinaten, die der t-SNE Algorithmus erzeugt. In (b) wird für jeden Punkt ein Cluster angelegt und Punkte innerhalb eines Bereichs mit Radius r werden ihm hinzugefügt. In (c) werden die finalen Cluster erzeugt und jeder Punkt wird genau einem Cluster zugeordnet.

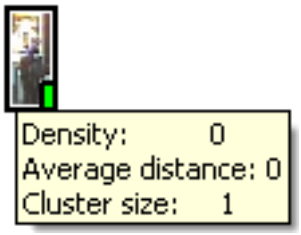


Abbildung 4.14: Dargestellt ist der Tooltip eines Clusters. „Densitiy“ steht dabei für die Kerndichtschätzung des Clusters. „Average distance“ für die durchschnittliche Entfernung der Elemente zum Zentrum und „Cluster size“ für die Anzahl der Elemente des Clusters.

Diese sind in der linken Hälfte der Abbildung 4.12 enthalten. Abhängig von Größe und Position der betrachteten Region wird eine Untermenge an Detektionen selektiert und im Darstellungsbereich visualisiert. Zu Beginn werden alle Detektionscluster angezeigt.

Die Menüleiste befindet sich oben rechts und enthält Schaltflächen für folgende Funktionen:

Zoom zur Standardansicht: Ermöglicht die Wiederherstellung der initialen Darstellung.

Hineinzoomen: Ein Teilbereich der betrachteten Region wird vergrößert dargestellt.

Herauszoomen: Die betrachtete Region wird vergrößert.

Die letzten beiden Funktionen können auch durch das Betätigen des Mausrades genutzt werden. Die Position des Mauszeigers wird dabei zum Mittelpunkt der vergrößerten bzw. verkleinerten Region. Das Hineinzoomen kann auch durch das Aufspannen eines Rechtecks mittels rechter Maustaste vollzogen werden. Bisher wurde lediglich die Größe des angezeigten Bereichs verändert. Aber auch seine Position kann verändert werden. Wird die linke Maustaste betätigt, so kann der Bereich durch Bewegen der Maus verschoben werden.

Die Größe und Position des dargestellten Bereichs wird in der Übersichtskarte eingetragen. Dies dient dem Nutzer als Orientierungshilfe. Die Übersichtskarte ist in Abbildung 4.15 zu sehen.

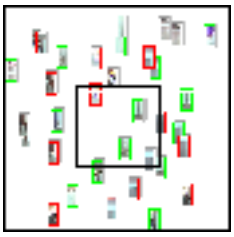


Abbildung 4.15: Dargestellt ist eine Miniaturansicht der initialen Darstellung der Cluster. Die Detektionen des inneren Rechtecks werden im Darstellungsbereich angezeigt.

Für die Bildung der Trainingsmengen ist es möglich sowohl einzelne, als auch mehrere Cluster auf einmal einer Trainingsmenge zuzuordnen. Während die Taste „P“ betätigt ist, können ausgewählte Cluster in die positive Trainingsmenge aufgenommen werden. Durch das Aufspannen eines Rechtecks können mehrere Zentren ausgewählt werden. Mittels Linksklick wird ein Cluster selektiert. Dasselbe ist möglich während, die Taste „N“ gedrückt wird. Jedoch werden die ausgewählten Elemente der negativen Trainingsmenge hinzugefügt. Durch das Betätigen der Taste „V“ kann eine selektierte Miniaturansicht vergrößert betrachtet werden. Dies ist rechts in Abbildung 4.12 dargestellt. Um anzuzeigen das ein Element einer Trainingsmenge angehört, wird in der unteren rechten Ecke der Miniaturansichten eine Markierung platziert. Grün steht für die positive Trainingsmenge, rot für die Negative. In Abbildung 4.16 sind je zwei markierte und nicht markierte Cluster dargestellt. Dabei wurde ein Cluster, dass vom Klassifikator als negativ eingestuft wurde, als positiv gekennzeichnet. Gegenteiliges gilt für das andere markierte Cluster.

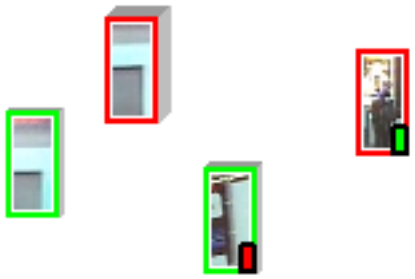


Abbildung 4.16: Abgebildet sind je zwei markierte und nicht markierte Cluster.

Als Erweiterung ist eine zweite Ansicht entstanden. Die Element-Ansicht. In ihr ist es weiterhin möglich, Elemente zu den Trainingsmengen hinzuzufügen. Jedoch kann hier nicht navigiert werden. Dargestellt werden sowohl die Zentren der Cluster, die in der Cluster-Ansicht angezeigt werden, als auch die mit ihnen assoziierten Elemente. Negative Detektionen werden in der negativ Liste dargestellt, während positive Detektionen in die positiv Liste eingefügt werden. Das Hinzufügen von Beispielen zu den Trainingsmengen läuft wie folgt ab. Elemente der Listen werden vom Nutzer markiert und dann, durch das Betätigen der entsprechenden Schaltfläche der Menüleiste, zu einer Trainingsmenge hinzugefügt. Daraufhin ändert sich die Hintergrundfarbe der aufgenommenen Elemente. Grün zeigt an, dass sie sich in der positiven Trainingsmenge befindet, rot steht für die Negative. Da das Hinzufügen von Elementen zu den Trainingsmengen möglich ist, können diese auch wieder entfernt werden. Dafür wird ein Element der Trainingslisten ausgewählt und durch Betätigen der Entfernen-Taste gelöscht. Die beschriebenen Funktionen sind in Abbildung 4.17 dargestellt.

Die Menüleiste enthält folgende Schaltflächen:

Hinzufügen positiv: Die markierten Elemente werden zur positiven Menge hinzugefügt.

Hinzufügen negativ: Die markierten Elemente werden zur negativen Menge hinzugefügt.

Element vergrößern: Ein einzelnes markiertes Element wird vergrößert angezeigt.

Gekoppelt ist die Cluster-Ansicht mit sowohl der Detektions-Ansicht als auch der Analyse-Ansicht. Die vergrößerte Darstellung eines Elementes ist der Auslöser dafür, dass die Detektion in der Detektions-Ansicht in allen Knoten durch einen gelben Punkt markiert wird. In der Analyse-Ansicht wird dieselbe Detektion gesetzt. Das heißt, der korrespondierende Bildbereich wird durch den Klassifikator ausgewertet.

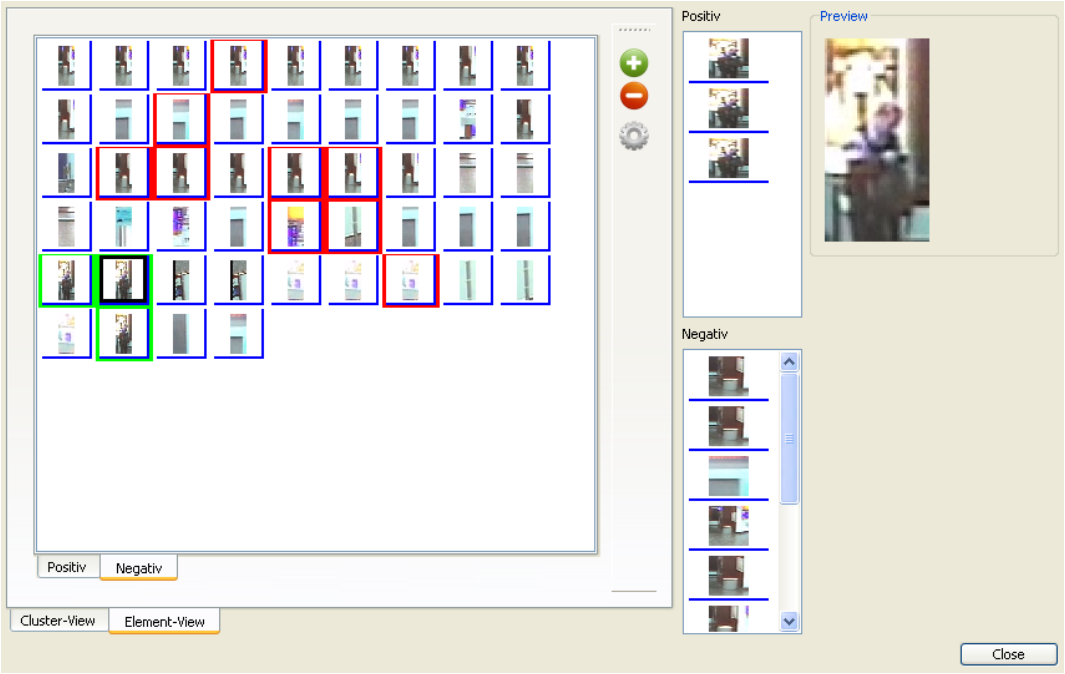


Abbildung 4.17: Abgebildet ist die Element-Ansicht. Die Elemente mit rotem Hintergrund wurden durch das Betätigen der roten Minus-Schaltfläche zur negativen Trainingsliste hinzugefügt. Die Elemente mit grünem Hintergrund wurden durch das Betätigen der grünen Plus-Schaltfläche zur positiven Trainingsliste hinzugefügt. Das Element, das mit einem schwarzen Rand markiert ist, wurde durch das Betätigen der Zahnrad-Schaltfläche vergrößert dargestellt.

4.6 Analyse-Ansicht

In dieser Ansicht kann das Verhalten des Klassifikators analysiert werden. Ziel der Analyse ist es zu verstehen, warum ein Klassifikator eine Region als positiv bzw. negativ einstuft. Mit diesem Verständnis kann er modifiziert werden, um seine Klassifikationsleistung zu verbessern. Da hier eine Entscheidungskaskade als Klassifikator verwendet wird, kann die Analyse auf zwei Ebenen stattfinden. Zum einen auf der Knotenebene und zum anderen auf der Merkmalebene. Auf Knotenebene kann untersucht werden, wie einzelne Teile der Kaskade Bildregionen bewerten. Auf Ebene der Merkmale kann die Bewertung im Detail betrachtet werden. Dabei wird die Reaktion der Merkmale dargestellt. Das heißt, wie sie eine Bildregion einschätzen, welchen Einfluss sie auf die Klassifikation haben und was sie in Kombination bewirken. Des Weiteren ist es möglich Informationen über die Auswertung einer gewählten Menge an Detektionen als Ganzes zu erhalten, sowie einzelne Elemente separat zu betrachten.

Um die Menge als Ganzes zu analysieren, kann die in Abbildung 4.18 dargestellte Multi-Teilansicht verwendet werden. Untersucht wird dort momentan wie zwei Merkmale der Kaskade die gesamte Menge einschätzen. Die Merkmale wurden mittels „Drag&Drop“ aus

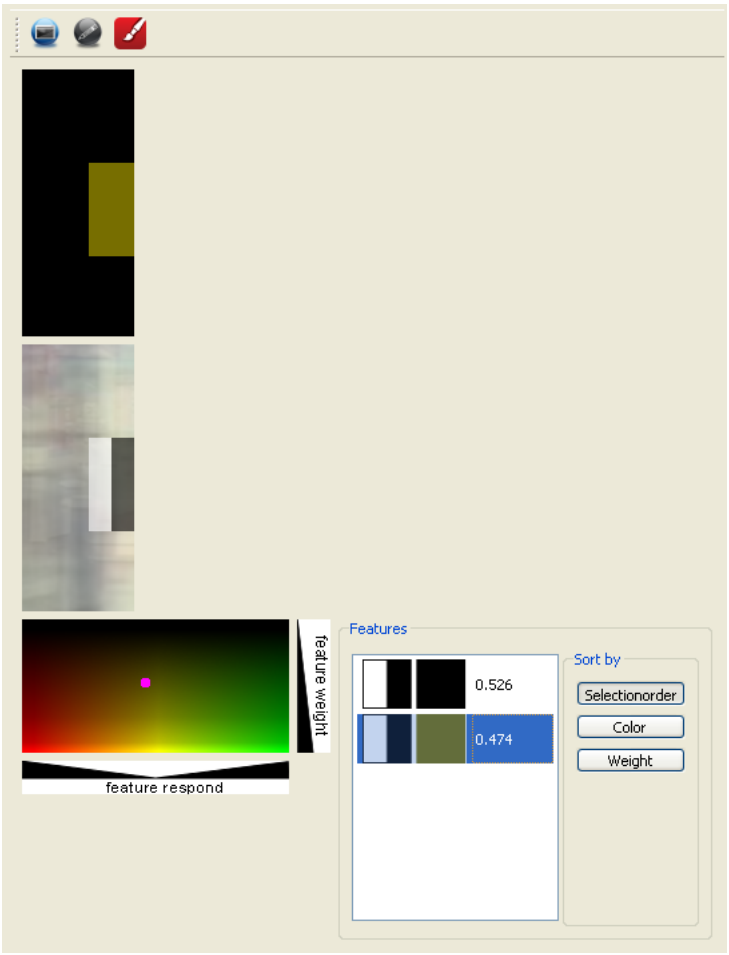


Abbildung 4.18: Abgebildet ist die Multi-Teilansicht. Durch sie kann das Verhalten der Kaskade und ihrer Merkmale untersucht werden. Berücksichtigt werden alle Detektionen. Oben links befindet sich ein Bild, indem die Farbcodierung der selektierten Merkmale, gemäß ihrer Größe und Position, überlagert wird. Darunter befindet sich eine Überlagerung aller Detektionen. Selektierte Merkmale werden dort eingezeichnet. Unten links befindet sich die Farbcode-Tabelle. Hier wird die Farbcodierung der selektierten Merkmale gemäß ihrer Bestandteile dargestellt. Unten rechts befindet sich die Merkmalliste. Im oberen Teil befindet sich die Menüleiste.

der Kaskaden-Ansicht in die Merkmalliste überführt. Ein Element dieser Liste besteht aus drei Teilen:

- Einer Miniaturansicht des Merkmaltyps.
- Der Farbcodierung der Reaktion des Merkmals.
- Dem Gewicht des Merkmals.

Durch die Schaltflächen „selection order“, „color“ und „weight“, ist es möglich die Listenelemente nach der Reihenfolge ihres Hinzufügens, ihrer Farbcodierung oder ihres Gewichtes zu sortieren. Der Benutzer kann so die Merkmale, die in der jeweiligen Kategorie den besten Wert aufweisen, schnell finden und untersuchen. Selektiert der Benutzer Merkmale, so werden sie im Überlagerungsbild aller Detektionen eingezeichnet. Dieses Bild soll dem Nutzer ermöglichen Gemeinsamkeiten in den Bildregionen aufzufinden. Beispielsweise Kanten oder sogar ganze Objekte. Zudem können Merkmale ausfindig gemacht werden, die auffällige Strukturen Überdecken und somit erkennen sollten. Zudem wird ein Überlagerungsbild der farbcodierten Reaktion erstellt. Ausgewertete Regionen werden mit der Farbcodierung der Merkmale überlagert. Erfolgt die Auswertung eines Bereichs anhand vieler Merkmale, so wird dieser durch die Überlagerung hervorgehoben. Damit kann der Nutzer erkennen, dass ein Bereich bei der Auswertung der dargestellten Merkmalkombination von Bedeutung ist. Dies wird in Abbildung 4.21 dargestellt. Die Auswahl von Merkmalen hat des Weiteren zur Folge, dass deren Farbcodierung in der Farbcode-Tabelle hervorgehoben wird. Die Codierung erfolgt anhand der Robustheit der Merkmale und deren Gewicht. Die Robustheit entspricht dem Abstand zwischen Schwellwert und dem Wert, den die Auswertung des Merkmals ergab. Je größer der Betrag dieser Distanz ist, desto robuster ist die Auswertung. Gelb kennzeichnet, dass sie nicht robust ist. Grün steht für eine robuste Auswertung und rot ebenfalls. Durch das Gewicht wird der Einfluss des Merkmals bei der Klassifikation verdeutlicht. Abbildung 4.19 zeigt die Farbcode-Tabelle sowie ein markiertes Merkmal, dessen Farbcodierung hervorgehoben ist.

Um einzelne Detektionen analysieren zu können, erfolgt mithilfe der linken Schaltfläche der Menüleiste der Wechsel zur Einzel-Teilansicht. Diese ist in Abbildung 4.20 dargestellt.

Die Multi-Teilansicht wird dabei um zwei Funktionsgruppen erweitert. Oben rechts befinden sich Listen der Detektionen. Hier kann der Benutzer eine Detektion auswählen und untersuchen, wie der Klassifikator bei der Auswertung dieser vorgeht. Das selektierte Element wird vergrößert angezeigt. Unterhalb der Auswahllisten befinden sich Steuerelemente. Durch diese kann festgelegt werden, mit welchen Teilen der Kaskade die Detektion ausgewertet werden soll. Das Ergebnis der Auswertung wird durch einen roten bzw. grünen Rahmen aufgezeigt. Grün steht dafür, dass die Detektion als positiv erkannt wurde. Rot deutet auf eine negative Erkennung hin. Sowohl mit dem Schieberegler als auch dem Eingabefeld wird der Knoten spezifiziert, bis zum dem die Auswertung der Kaskade erfolgt. Falls das Auswahlfeld „single node“ angewählt ist, so wird nur der spezifizierte Knoten verwendet. In beiden Fällen wird ein Überlagerungsbild der farbcodierten Reaktionen der Merkmale erstellt, die für die Auswertung genutzt wurden. Abbildung 4.21 stellt den Unterschied da.

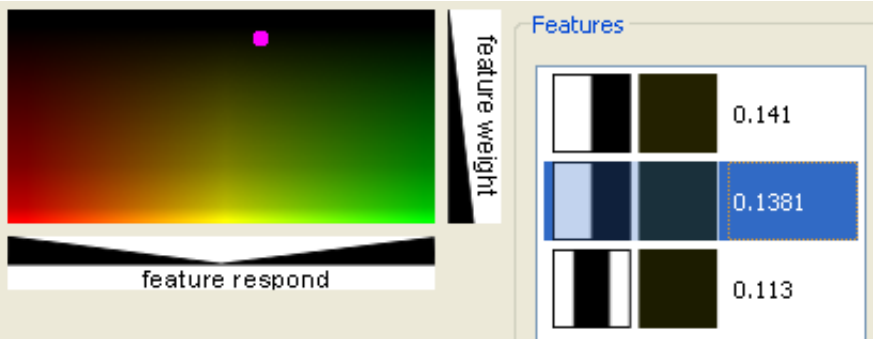


Abbildung 4.19: Abgebildet ist die Farbcode-Tabelle sowie ein markiertes Merkmal, dessen Farbcodierung hervorgehoben ist. Die Codierung setzt sich aus der Robustheit und dem Gewicht des Merkmals zusammen. Gelb steht für eine geringe Robustheit und eine hohe Unsicherheit der Auswertung. Rot für hohe Robustheit und hohe Sicherheit über eine negative Auswertung. Grün für hohe Robustheit und hohe Sicherheit über eine positive Auswertung.

Alle Funktionen, die in der Multi-Teilansicht zur Verfügung stehen, können auch hier genutzt werden. Bei der Darstellung selektierter Merkmale besteht jedoch ein Unterschied. Größe und Position werden im Überlagerungsbild durch Rahmen dargestellt. Dies ist in Abbildung 4.21 zu sehen. Dadurch kann beobachtet werden, ob der Bereich den die selektierten Merkmale auswerten, ebenfalls durch andere Merkmale überdeckt wird. Durch die Färbung kann eingeschätzt werden wie gut und wie häufig ein Bereich ausgewertet wurde. Je kräftiger die Farbe, desto häufiger wurde ein Bereich ausgewertet. Grün steht für eine positive Auswertung und rot für eine negative.

Nachdem mit den vorgestellten Funktionen Wissen über das Verhalten des Klassifikators gesammelt wurde, kann dies genutzt werden, um ihn gezielt zu verbessern. Zum einen ist es möglich neue Merkmale hinzuzufügen. Diese können dann hinsichtlich ihrer Tauglichkeit untersucht werden. Zum anderen können vorhandene Merkmale editiert werden. Dies kann durch den Menüpunkt „Edit feature“ des Kontextmenüs der Merkmalliste geschehen. Sie ist in Abbildung 4.22 abgebildet. Durch den Eintrag „Delete feature“ wird ein Merkmal aus dem Knoten gelöscht, aus dem es entnommen wurde. Wird der Menüpunkt „Feature infos“ ausgewählt so öffnet sich der Dialog, der in Abbildung 4.9 dargestellt ist.

Als Basis für das Editieren eines Merkmals wird der Dialog für die Erzeugung eines neuen Merkmals genutzt. Anstelle des grauen Hintergrunds wird die vergrößerte Ansicht der ausgewählten Detektion dargestellt. Damit kann der Nutzer Merkmale gezielt über Strukturen positionieren, die für die Klassifikation relevant sind. Des Weiteren werden Merkmale, die in der Merkmalliste selektiert sind, angezeigt. Der soeben beschriebene Dialog ist in Abbildung 4.23 dargestellt. In der Merkmalliste wurde ein Merkmal markiert und editiert. Es wurde von seiner ursprünglichen Position aus in die linke untere Ecke verschoben.

Eine weitere Möglichkeit das gewonnene Wissen einzubringen ist die Erstellung des Klassifikators zu beeinflussen. Dahingehend ist die Marker-Ansicht entstanden. Mit ihrer Hilfe

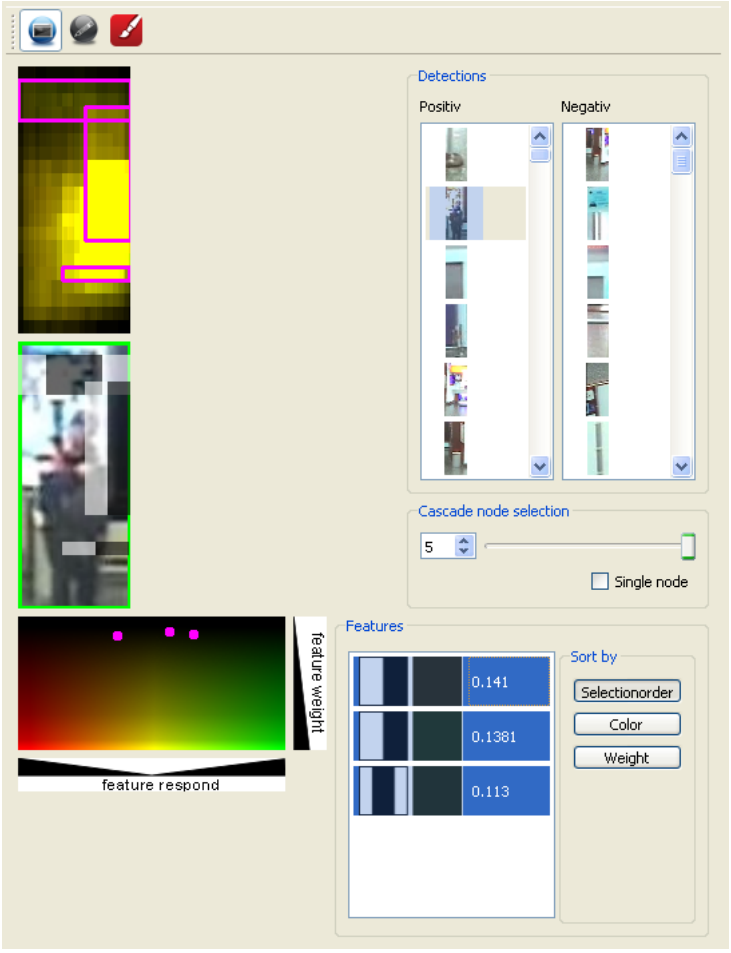


Abbildung 4.20: Abgebildet ist die Einzel-Teilansicht. Diese ist eine Erweiterung der Multi-Teilansicht. Die Erweiterung besteht aus den Auswahllisten (rechts oben) für die Detektionen und Steuerelementen (mittig rechts).

kann der Benutzter Regionen definieren, in denen keine Merkmale enthalten sein dürfen. Diese Ansicht ist in Abbildung 4.24 dargestellt.

Durch die Schaltflächen der Menüleiste können zwei Werkzeuge ausgewählt werden. Zum einen das Werkzeug, mit dem Regionen gesperrt werden können (roter Knopf). Zum anderen das Werkzeug, mit dem gesperrte Regionen freigegeben werden (grüner Knopf). Der Mauszeiger gibt Auskunft über das verwendete Werkzeug. Die Werkzeuge können auf den zwei dargestellten Zeichenbereichen verwendet werden. Im Rechten ist die ausgewählte Detektion zu erkennen. Darin können lokal Regionen gesperrt und freigegeben werden. Lokale Änderungen werden zeitgleich in der globalen Ansicht der gesperrten Regionen aktualisiert. Diese ist in der rechten Zeichenfläche zu sehen. Auch hier ist es möglich Regionen freizugeben und zu sperren. Falls eine globale Änderung die lokale betrifft, so wird diese aktualisiert.

(a) (b)

Abbildung 4.21: Abgebildet sind zwei Möglichkeiten eine Bildregion durch die Kaskade auswerten zu lassen. Links ist jeweils ein Überlagerungsbild der farbcodierten Reaktionen der Merkmale dargestellt. Rechts ist jeweils die Detektion vergrößert abgebildet. In (a) wird die Kaskade bis einschließlich des Knotens 5 ausgewertet, während in (b) nur Knoten 3 betrachtet wurde. Hervorgehoben sind in (a) und (b) jeweils Größe, Position und Typ dreier Merkmale. Diese wurden in der Merkmalliste selektiert.

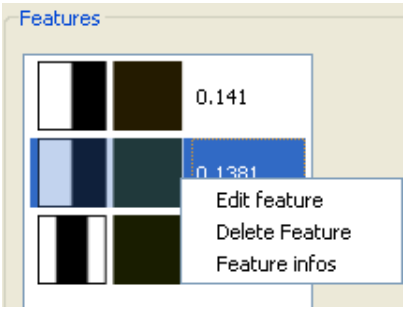


Abbildung 4.22: Abgebildet ist das Kontextmenü der Merkmalliste der Analyse-Ansicht. Mit „Edit feature“ kann das Merkmal verändert werden. „Delete Feature“ löscht das Merkmal. Durch „Feature infos“ werden Informationen des Merkmals angezeigt.

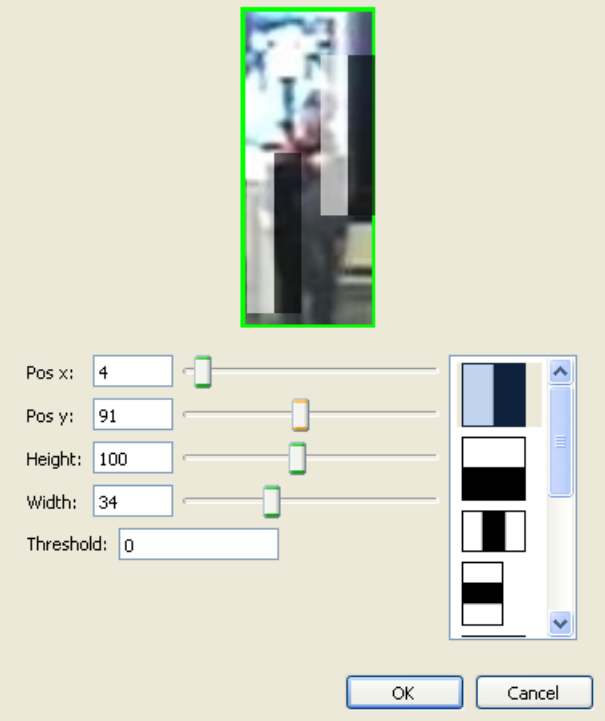


Abbildung 4.23: Dargestellt ist der Dialog, mit dem ein Merkmal editiert wird. Durch die Schieberegler kann die sowohl Position als auch Höhe und Breite verändert werden. Die Auswahlliste ermöglicht die Veränderung des Merkmaltyps. Zudem kann hier der Schwellwert des Merkmals verändert werden.

Diese Ansicht ist mit dem Klassifikator, der Kaskaden- und der Detektions-Ansicht verknüpft. Der Klassifikator wird beim nächsten Trainingsdurchlauf dahin gehend beeinflusst, dass er nur Merkmale aus zugelassenen Regionen verwendet. In der Kaskaden-Ansicht ausgewählte Merkmale können gelöscht oder editiert werden. In sowohl der Kaskaden- als auch der Detektions-Ansicht werden Knoten hervorgehoben, die untersucht werden. In der Detektions-Ansicht wird zudem die Detektion hervorgehoben, die in der Einzel-Teilansicht ausgewählt ist.

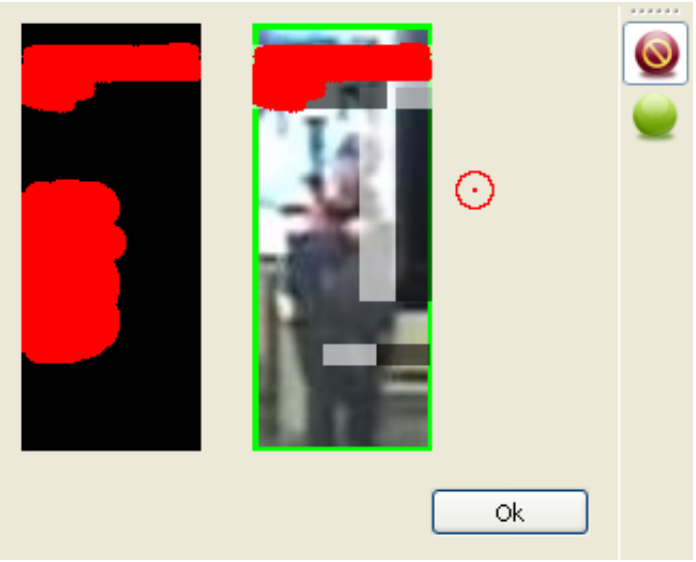


Abbildung 4.24: Abgebildet ist die Marker-Ansicht. Über die Menüleiste ist die Auswahl von Werkzeugen möglich mit denen Regionen gesperrt und freigegeben werden können. Das rechte Bild entspricht der ausgewählten Detektion. Darin können lokal Regionen gesperrt bzw. freigegeben werden. Lokale Änderungen werden in die globale Ansicht übertragen. Diese befindet sich links im Bild.

5 Anwendungsbeispiel

In diesem Kapitel wird ein Klassifikator durch ein interaktives Training erzeugt. Ziel dabei ist, Menschen in Videoausschnitten zu erkennen. Das Video, welches für das Training verwendet wird, entstand in einem Flughafen. Untersucht wird ein Videoausschnitt indem 15 Personen an insgesamt 44 Positionen vorkommen. Die Personen sind in Abbildung 5.1 dargestellt. Die entstandenen Komponenten und deren Funktionen werden in den Trainingszyklen zum Einsatz gebracht. Zudem wird aufgezeigt, wie sich das Training auf den Klassifikator auswirkt.

Abbildung 5.1: Personen im Videoausschnitt.

Zu Beginn wird der Beispielextraktor genutzt, um eine initiale Trainingsmenge zu erzeugen. Sie besteht aus 21 positiven und 60 negativen Beispielen. Mit dieser Beispielmenge erfolgt das erste Training eines zufällig erzeugten Klassifikators. Die resultierende Kaskade ist in Abbildung 5.3 dargestellt.

Danach wird der Videoabschnitt durch die Kaskade ausgewertet. Parameter der Evaluation können durch das in Abbildung 5.4 dargestellte Eingabefenster spezifiziert werden. Durch die Parameter wird bestimmt, welche Detektionen im nächsten Schritt angezeigt werden. Um einen ersten Eindruck über die Güte des Klassifikators zu gewinnen, sind die Achsparameter so eingestellt das jede Detektion angezeigt wird. Für die x-Achse wurde das Intervall $[-1, 1]$ gewählt und für die y-Achse $[0, 1]$.

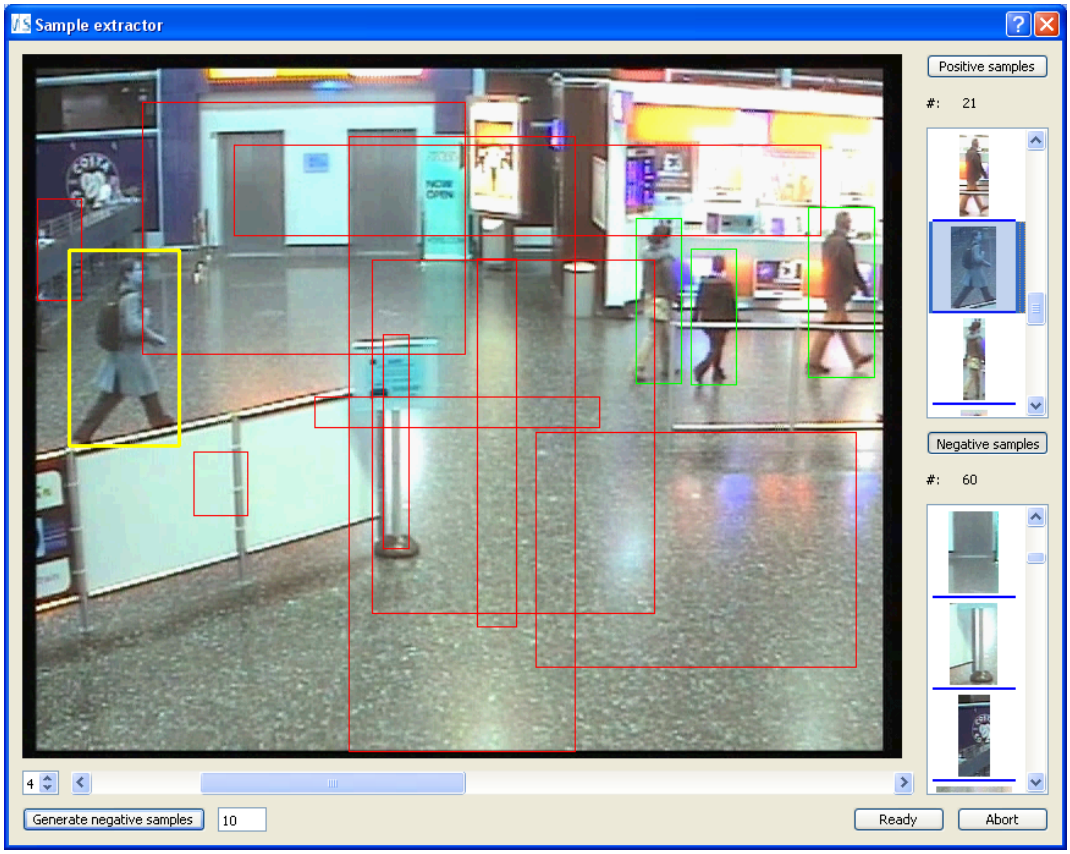


Abbildung 5.2: Abgebildet ist der Beispielextraktor. Aus den ersten sechs Frames wurden 21 positive und 60 negative Beispiele entnommen.

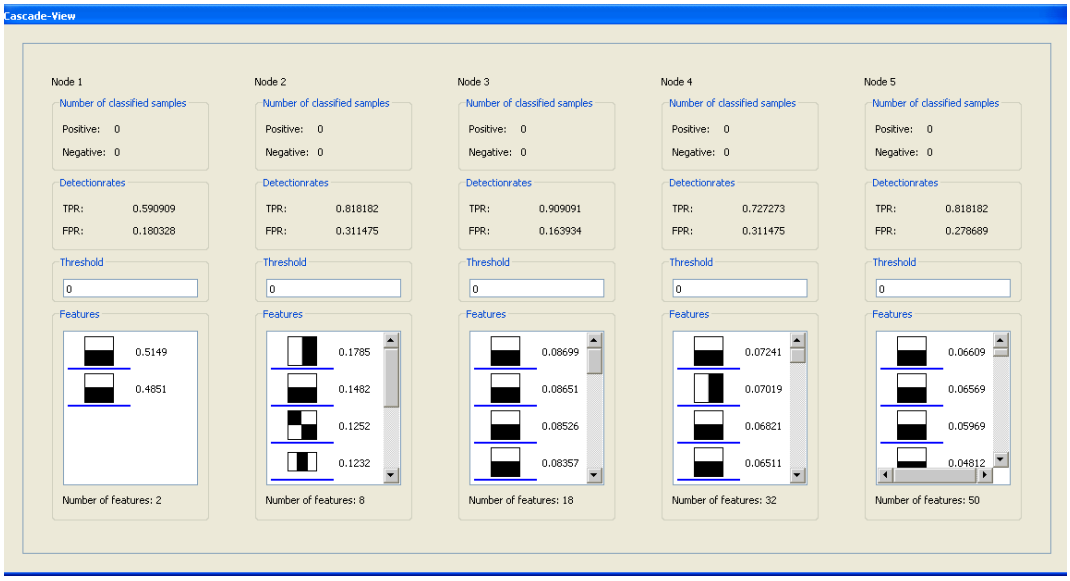


Abbildung 5.3: Klassifikator nach dem ersten Training.

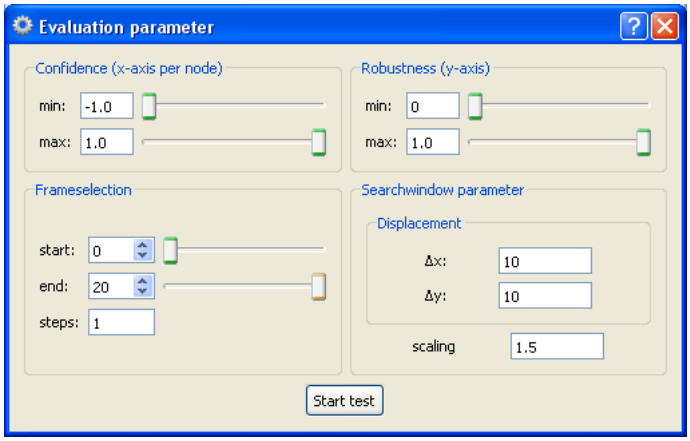
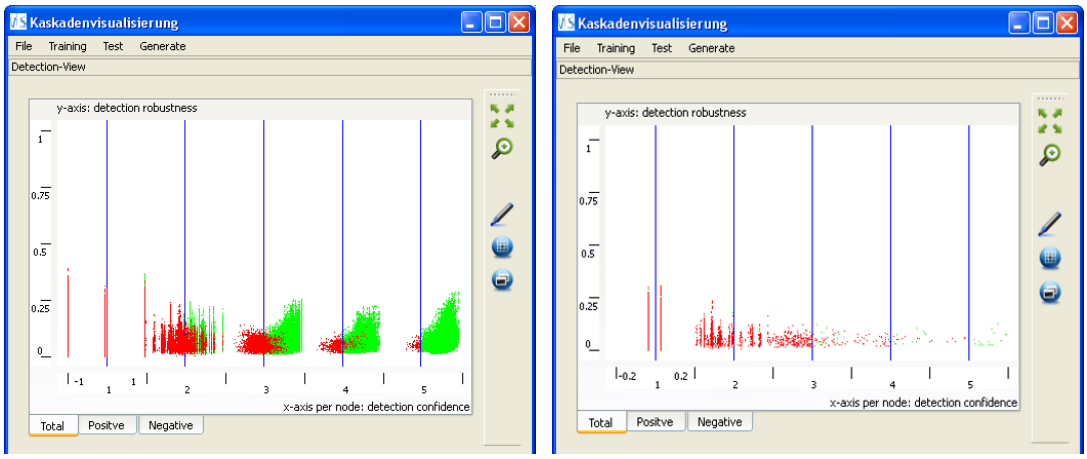


Abbildung 5.4: Abgebildet ist das Eingabefenster der Parameter für die Evaluation. Oben links kann angegeben werden, welche Detektionssicherheit ein Bildbereich aufweisen muss. Oben rechts erfolgt die Angabe der Robustheit. Unten links befinden sich Schaltflächen, mit denen der Videoabschnitt bestimmt werden kann, der ausgewertet werden soll. „steps“ gibt den Abstand zweier Frames an, die ausgewertet werden. Unten rechts können Angaben zum Feinheitsgrad gemacht werden, mit dem ein Frame abgetastet wird. Ein Suchfenster wird dabei um Δx oder Δy verschoben. Mit „scaling“ kann ein Faktor angegeben werden, mit dem das Suchfenster schrittweise vergrößert wird. Somit können Objekte in verschiedenen Größen erkannt werden.

Das Ergebnis dieser Auswertung ist in der Detektions-Ansicht in Abbildung 5.5(a) zu sehen. Erkannt werden 35730 positive Bildregionen. Der überwiegende Teil davon besteht aus Hintergrundbereichen. Die am häufigsten vorkommenden Bereiche sind in Abbildung 5.7 dargestellt. Zudem werden Regionen erkannt in denen Personen oder Teile von ihnen vorkommen. Die am besten erkannten Personen sind in Abbildung 5.6 dargestellt.

Um die enorme Menge an Detektionen einzugrenzen, wird der Videoausschnitt nochmals ausgewertet. Das Intervall der x-Achse wird dabei auf $[-0.2, 0.2]$ gesetzt. Das der y-Achse auf $[0, 1]$. Damit werden nur die Detektionen angezeigt, bei deren Auswertung die Kaskade das festgelegte Maß an Unsicherheit aufweist. Dies ist in Abbildung 5.5(b) dargestellt. Mit dieser eingeschränkten Menge an Detektionen wird jeder Knoten separat untersucht. Dabei werden alle Elemente eines Knotens in die Cluster-Ansicht überführt. Um den Generalisierungseffekt des Trainings zu verdeutlichen, wird nur nach Personen gesucht, die in der initialen Trainingsmenge enthalten sind und als negativ erkannt wurden. Diese werden der neuen Trainingsmenge als positive Beispiele hinzugefügt. Hintergrundbereiche, die als positiv erkannt wurden, werden als negative Beispiele aufgenommen. Abbildung 5.9 stellt dies dar. Angezeigt werden die Elemente des fünften Knotens. Die umkreisten Elemente enthalten die soeben beschriebenen Beispiele.

Die so entstandene Trainingsmenge besteht aus 15 positiven und 20 negativen Beispielen. Mit ihr wird die Kaskade erneut trainiert. Die Kaskade, die sich durch das Training ergeben



(a) (b)

Abbildung 5.5: Abgebildet ist die Detektions-Ansicht nach dem ersten Training. (a) mit Achsparameter x-Achse: $[-1, 1]$; y-Achse: $[0, 1]$. (b) mit x-Achse: $[-0.2, 0.2]$; y-Achse: $[0, 1]$.

Abbildung 5.6: Erkannte Personen nach dem ersten Training.

Abbildung 5.7: Positiv erkannter Hintergrund nach dem ersten Training.

Abbildung 5.8: Einige teilweise erkannte Personen nach dem ersten Training.

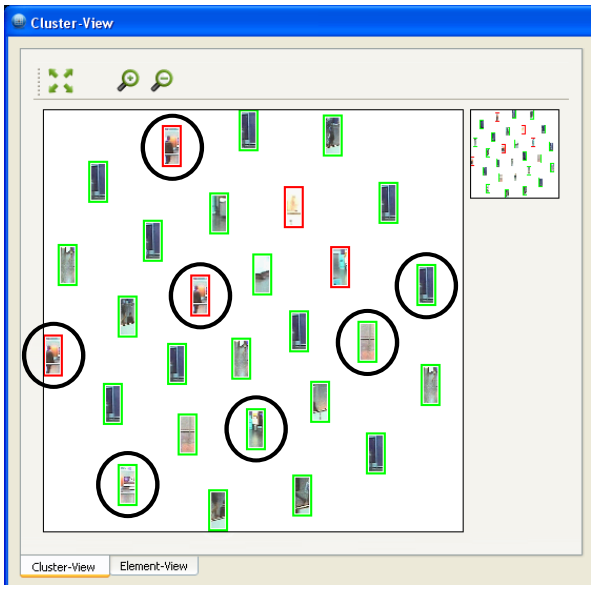


Abbildung 5.9: Abgebildet ist die Cluster-Ansicht während des ersten Trainingszyklus. Betrachtet werden Elemente des fünften Knotens, deren Auswertung mit einer gewissen Unsicherheit verbunden war. Beispiele, die für das Training ausgewählt wurden, sind umkreist.

hat, ist in Abbildung 5.10 dargestellt. Zwei Veränderungen können an ihr festgestellt werden. Zum einen enthalten die Knoten nun andere Merkmale. Zum anderen wurde die Kaskade um einen Knoten erweitert. Dies geschah, da ihre bisherige Fehlerrate zu hoch war.

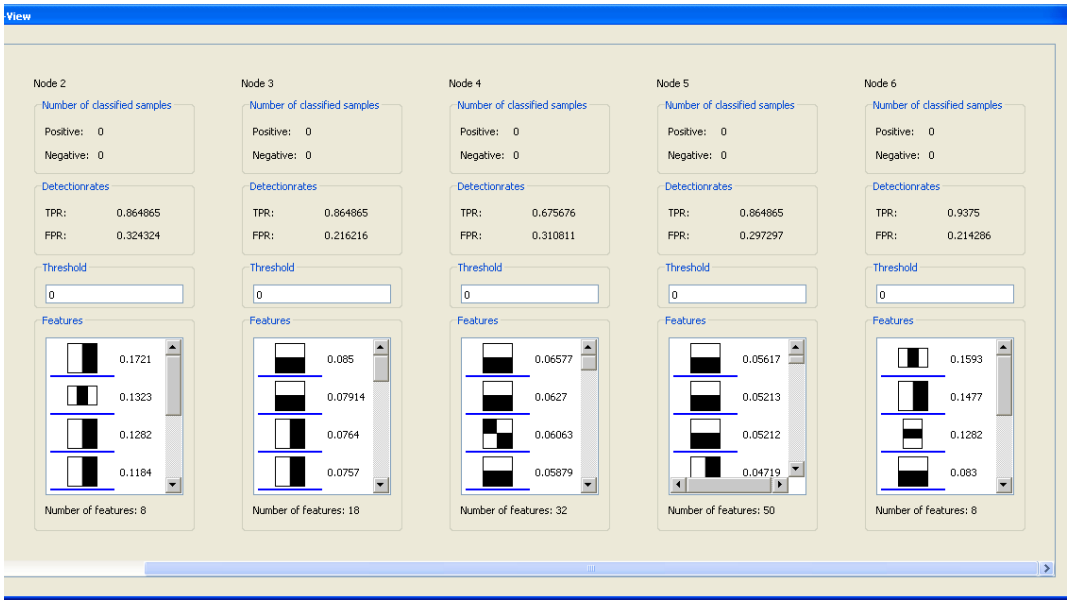


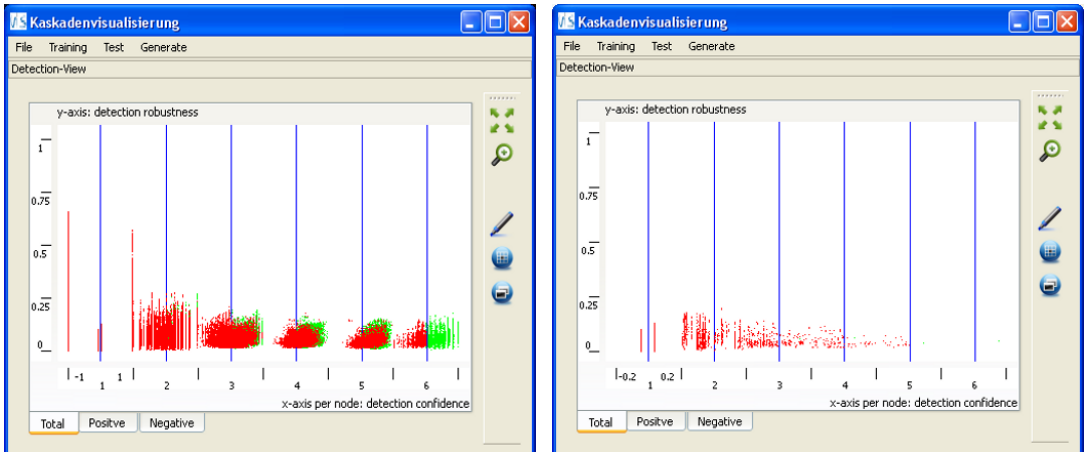
Abbildung 5.10: Klassifikator nach dem zweiten Training.

Um zu Prüfen inwieweit sich der Klassifikator verbessert hat ist zunächst wieder ein Überblick über alle Detektionen entstanden. Dieser ist in Abbildung 5.11(a) zu betrachten. Als positiv erkannt werden 15678 Regionen. Im Vergleich zu Abbildung 5.5(a) ist des Weiteren zu erkennen, dass viele der negativen Detektionen mit einer höheren Robustheit ausgewertet wurden. Dies ist in den ersten Knoten der Kaskade deutlich wahrzunehmen.

Personen, die erkannt wurden, sind in Abbildung 5.12 dargestellt. Auch diesmal wurden alle vorkommenden Personen entweder teilweise oder vollständig erkannt. In Abbildung 5.13 sind Vertreter der Hintergrundregionen dargestellt, welche die Kaskade als positiv erkannt hat.

Um eine genauere Analyse durchzuführen und eine neue Trainingsmenge zu bilden, wurde der Videoabschnitt noch einmal mit anderen Grenzintervallen für die x- und y-Achse ausgewertet. Das Resultat ist in Abbildung 5.11(b) zu sehen. Im Vergleich zu Abbildung 5.5(b) wird deutlich, dass kaum noch positiv erkannte Regionen angezeigt werden. Die Detektionssicherheit für positive Beispiele hat sich daher verbessert.

Die Elemente der Knoten werden diesmal in der Analyse-Ansicht untersucht. Dafür werden die Elemente jedes einzelnen Knoten in die Analyse-Ansicht überführt. In Abbildung 5.15 ist ein Auszug der Analyse des vierten Knoten dargestellt. Darin ist zu erkennen, dass eine negative Bildregion gefunden wurde, die jedoch eine Person enthält. Diese wird hier durch den vierten Knotens ausgewertet. Die Bereiche, die durch die Merkmale des Knotens ausgewertet werden, sind im Überlagerungsbild zu erkennen. Dabei wird deutlich, dass nur



(a) (b)

Abbildung 5.11: Abgebildet ist die Detektions-Ansicht nach dem zweiten Training. (a) mit Achsparameter x-Achse: $[-1, 1]$; y-Achse: $[0, 1]$. (b) mit x-Achse: $[-0.2, 0.2]$; y-Achse: $[0, 1]$.

Abbildung 5.12: Erkannte Personen nach dem zweiten Training.

Abbildung 5.13: Positiv erkannter Hintergrund nach dem zweiten Training.

Abbildung 5.14: Einige teilweise erkannte Personen nach dem zweiten Training.

wenige Merkmale die Person überdecken. Des Weiteren wird untersucht, welche Merkmale maßgeblich an der Auswertung beteiligt sind. Dies ist durch die Kombination der Farbcode-Tabelle und des Überlagerungsbildes möglich. Während der Analyse zeigte sich, dass viele kleine Merkmale in Randregionen platziert wurden. Diese sind zudem nicht ausschlaggebend für die Klassifikation. Daher wurden sie aus dem Knoten gelöscht. Um die Platzierung von Merkmalen in Randgebieten in Zukunft zu vermeiden, wurden diese in der Marker-Ansicht als verboten gekennzeichnet. Dies ist in Abbildung 5.16 zu sehen.

Zuletzt wurde eines der übrigen Randmerkmale editiert. Größe, Position und Typ wurden dabei so verändert, dass es in der Lage sein sollte, Köpfe von Personen zu erkennen. Dies ist in Abbildung 5.17 dargestellt.

Im Anschluss an die Analyse wird die nächste Trainingsmenge gebildet. Sie enthält 41 negative Beispiele und ein positives. Die Negativen enthalten dabei Hintergrundregionen. Das Positive entspricht dem, das in Abbildung 5.15 untersucht wurde.

Das anschließende Training führte zu der Kaskade, die in Abbildung 5.18 dargestellten ist. Zwei Veränderungen sind dabei auffällig. Zum einen haben sich die Merkmale der Knoten teilweise verändert. Zum anderen wurde die Kaskade um einen Knoten erweitert.

Abbildung 5.19(a) stellt den Überblick über die erkannten Detektionen nach dem dritten Training dar. Dabei wurden nur noch 5049 Regionen als positiv eingestuft. Im Vergleich zu Abbildung 5.11(a) ist des Weiteren zu erkennen, dass die Robustheit der Detektionen leicht abgenommen hat. Vergleicht man Abbildung 5.19(b) und 5.11(b), so stellt man fest, dass weiterhin wenige positive Detektionen als unsicher angesehen werden. Die negativen

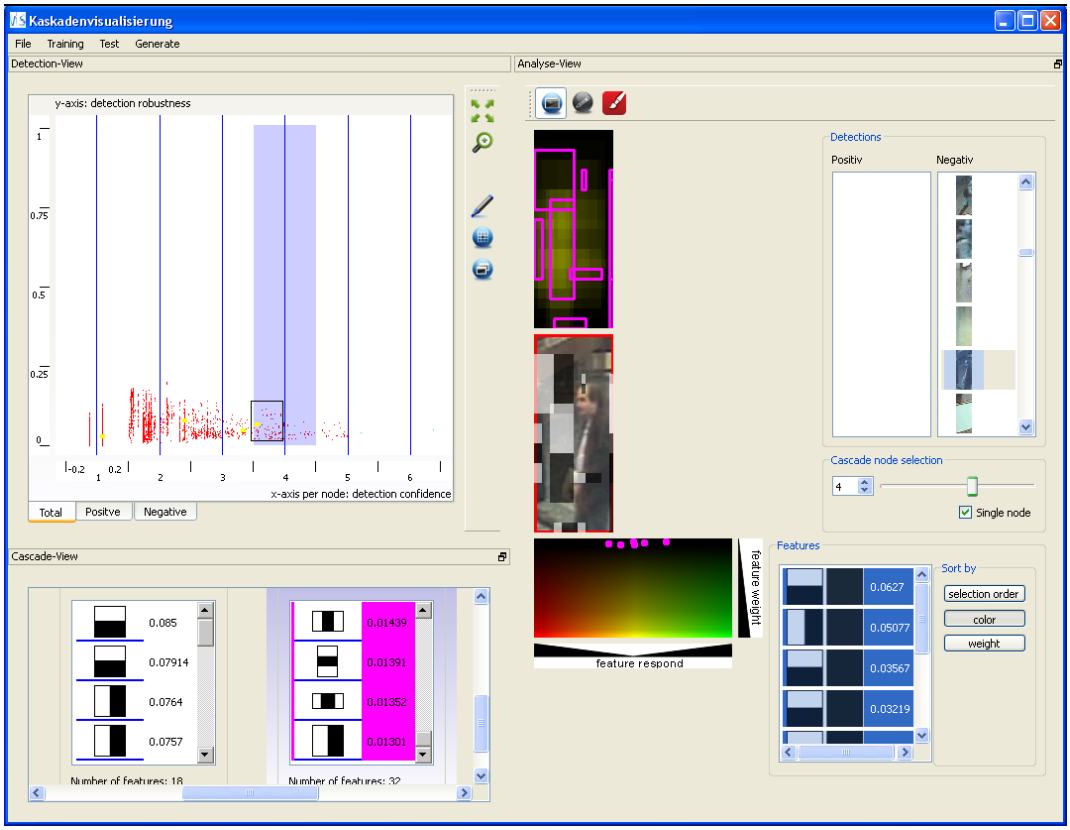


Abbildung 5.15: Abgebildet ist die Analyse-Ansicht während des zweiten Trainingszyklus. Selektiert ist ein Element, das als negativ eingestuft ist, jedoch eine Person enthält. Es wird untersucht, welche Regionen des Elements ausgewertet werden und welche Merkmale dabei beteiligt sind.

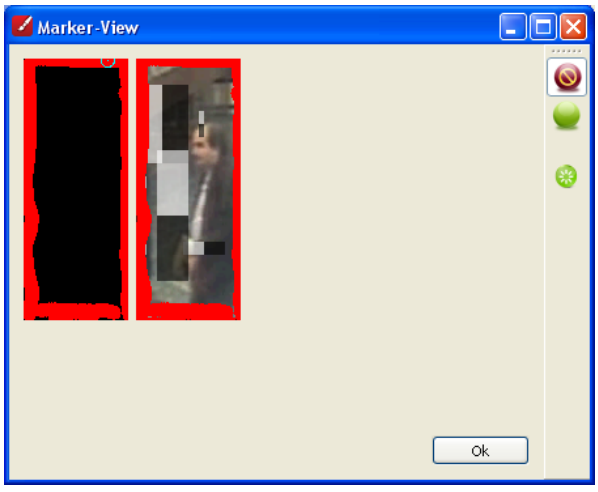


Abbildung 5.16: Markieren von Randgebieten.

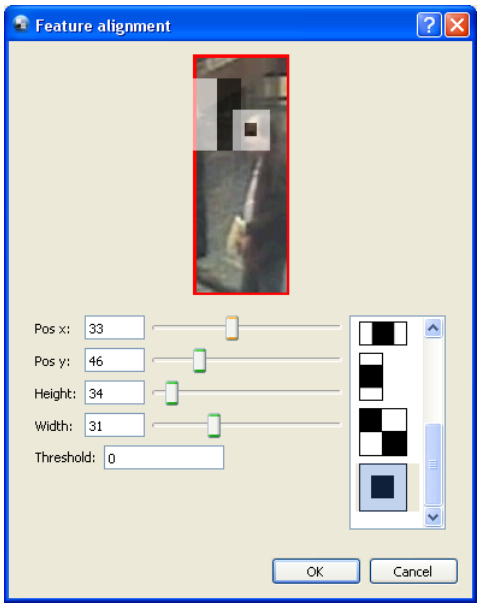


Abbildung 5.17: Editieren eines Merkmals.

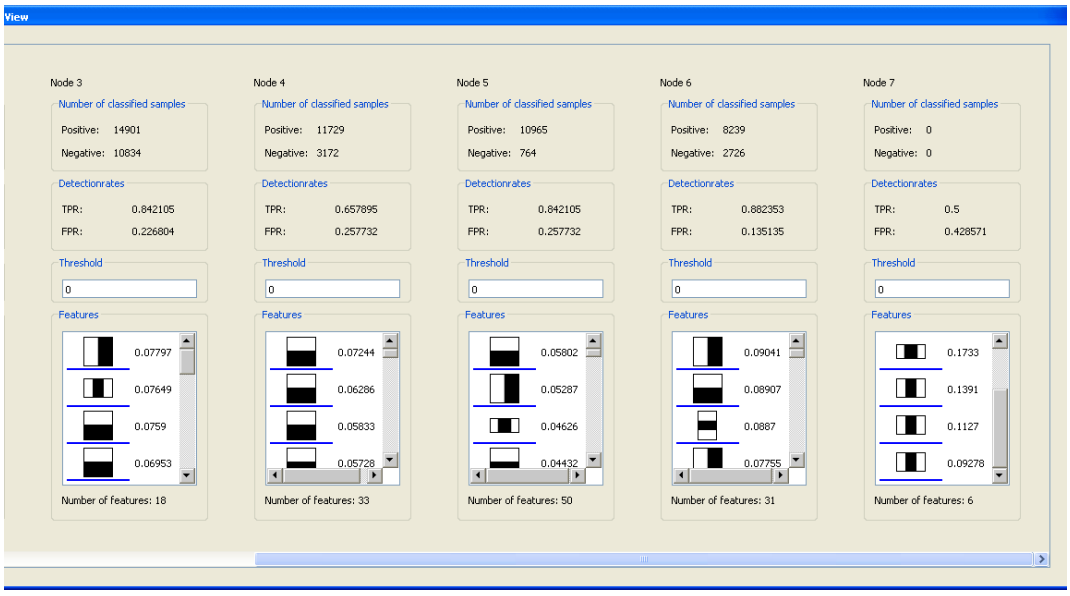
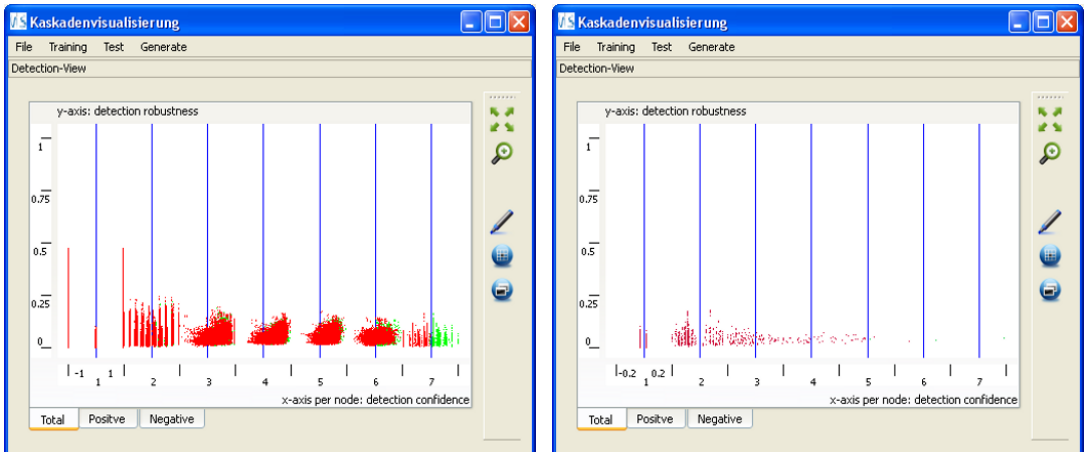


Abbildung 5.18: Klassifikator nach dem dritten Training.

Detektionen im ausgewählten Bereich haben sich zudem auch reduziert. Abbildung 5.20 zeigt, dass noch immer alle Personen erkannt werden. Während Abbildung 5.21 die Art der Hintergrundregionen aufzeigt, die weiterhin als positiv angesehen werden.



(a) (b)

Abbildung 5.19: Abgebildet ist die Detektions-Ansicht nach dem dritten Training. (a) mit Achsparameter x-Achse: $[-1, 1]$; y-Achse: $[0, 1]$. (b) mit x-Achse: $[-0.2, 0.2]$; y-Achse: $[0, 1]$.

Abbildung 5.20: Erkannte Personen nach dem dritten Training.

Um zu prüfen, welchen Nutzen die Aktionen haben, die zuvor in der Analyse-Ansicht durchgeführt wurden, werden alle Detektionen des siebten Knotens in die Analyse-Ansicht überführt. Dies ist in Abbildung 5.23 dargestellt. Aus allen Elementen wurde eines selektiert,

Abbildung 5.21: Positiv erkannter Hintergrund nach dem dritten Training.

Abbildung 5.22: Einige teilweise erkannte Personen nach dem dritten Training.

das eine Person enthält. Die Bildregion wurde anschließend durch die gesamte Kaskade ausgewertet. Im Überlagerungsbild der farbcodierten Merkmalreaktionen ist dabei zu erkennen, dass keine Randgebiete ausgewertet wurden. Hauptsächlich sind Gebiete im Zentrum hervorgehoben. Randregionen als verbotene Bereiche zu kennzeichnen, hat den gewünschten Effekt erzielt. Als Nächstes wird das benutzerdefinierte Merkmal untersucht, das zuvor platziert wurde, um Köpfe von Personen zu erkennen. Diese Maßnahme war nicht sehr effektiv. Durch das Training wurde sein Gewicht auf 0.005 gesetzt. Damit entspricht es dem schlechtesten Merkmal im Knoten. Dies zeigt auch seine Analyse. An der Farbcode-Tabelle ist zu erkennen, dass das Merkmal eine hohe Unsicherheit aufweist. Zudem wird deutlich das seine Position nicht optimal gewählt wurde. Das Merkmal könnte in weiteren Zyklen adaptiert werden, bis es den gewünschten Erfolg erzielt.

Nach drei Trainingszyklen wurde der Klassifikator mit 158 Beispielen trainiert. Darunter befanden sich 37 positiv und 121 negative Beispiele. Die Anzahl der positiven Detektionen wurde nach jedem Zyklus erheblich gesenkt. Nach dem ersten Training waren es 35730. Nach dem zweiten noch 15678. Nach dem letzten lediglich 5049. Dies ist hauptsächlich darauf zurückzuführen, dass Hintergrundbereiche korrekt als falsch erkannt wurden. Zudem konnten nach jedem Zyklus mindestens dreizehn der fünfzehn Personen vollständig erkannt werden. Die Restlichen teilweise.

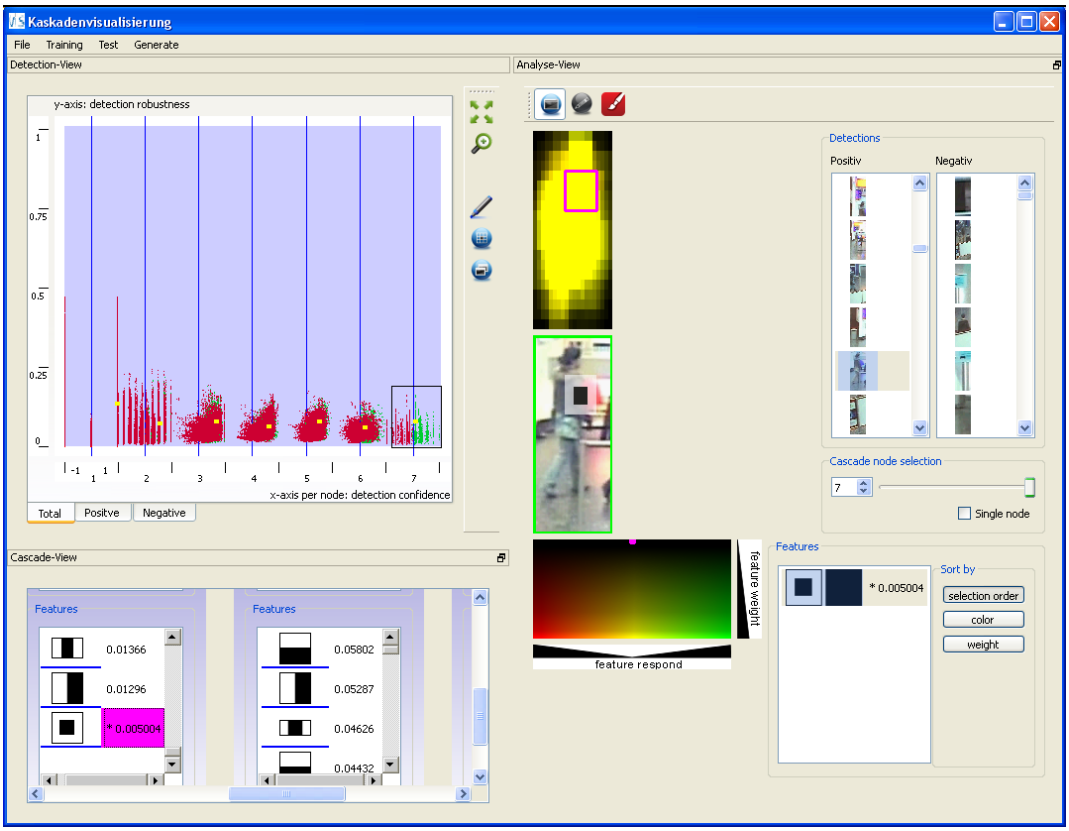


Abbildung 5.23: Analyse-Ansicht zur Überprüfung von Aktionen.

6 Zusammenfassung und Ausblick

Im Laufe dieser Arbeit ist ein System entstanden, welches das Konzept des Interactive Learning umsetzt. Dabei wird dem Benutzer ermöglicht, interaktiv am Training eines Klassifikators beteiligt zu sein. Der hier verwendete Klassifikator entspricht einem Kaskadenklassifikator, der zur Klassifikation von Bildregionen Rechteckmerkmale nutzt. Das interaktive Training entspricht dabei einer Kombination der Strategien des Active Learning sowie der visuellen Analyse von Daten. Methoden dieser werden dazu genutzt, um Wissen über das Klassifikationsverhalten des Kaskadenklassifikators zu erhalten. Oder anders formuliert, zu verstehen, nach welchen Kriterien der Klassifikator eine Bildregion klassifiziert. Dadurch ist es möglich den Klassifikator gezielt mit Beispielen zu trainieren, die Schlüsselaspekte für die Klassifikation enthalten. Dies entspricht dem Active Learning. Die Auswahl der Beispiele erfolgt anhand des erworbenen Wissens. Des Weiteren ist es dem Benutzer möglich, den Klassifikator umzugestalten. Dabei können Merkmale hinzugefügt oder gelöscht werden. Zudem wird ermöglicht Größe, Position oder Typ vorhandener Merkmale zu verändern. Außerdem kann das Boosting des Klassifikators beeinflusst werden. Dies geschieht, indem der Benutzer Regionen des Suchfensters als gesperrt markiert. Dadurch werden nur Merkmale zu einem Knoten hinzugefügt, die zugelassene Regionen überdecken.

Das Training des Klassifikators erfolgt zyklisch. Ein Zyklus besteht aus drei Teilschritten:

- Dem Untersuchen der Daten, um Wissen zu erhalten.
- Die Bildung einer Beispielmenge.
- Das Training des Klassifikators anhand der Beispielmenge.

Im ersten Zyklus wird der Beispielextraktor genutzt, um eine initiale Trainingsmenge an Beispielen aus einem Video zu extrahieren. In jedem weiteren Zyklus kann der Benutzter eine Untermenge an klassifizierten Bildregionen untersuchen. Die Auswahl dieser erfolgt in der Detektions-Ansicht. In der Cluster-Ansicht werden die Bildregionen nach ihrer Ähnlichkeit gruppiert. Dadurch ist es möglich schnell und einfach neue Trainingsmengen zu bilden. Die Analyse-Ansicht ermöglicht eine detaillierte Untersuchung der Bildregionen. Dabei können sie durch Teile des Klassifikators ausgewertet werden. Die Auswahl dieser erfolgt unter Zuhilfenahme der Kaskaden-Ansicht.

Anhand eines Anwendungsbeispiels wurde die Funktionsweise des Systems demonstriert. Dabei erfolgte das Training eines Klassifikators, der Personen in einem Video erkennt. Durchlaufen wurden drei Zyklen. Die Gesamtanzahl der Beispiele, mit denen der Klassifikator trainiert wurde, betrug 158. Darunter befanden sich 37 positiv und 121 negative Beispiele. Des Weiteren konnte festgestellt werden, dass sich die Anzahl der als positiv erkannten Regionen nach jedem Trainingszyklus verringerte. Nach dem ersten Training waren es

35730. Nach dem zweiten noch 15678. Nach dem letzten lediglich 5049. Die Verringerung ist hauptsächlich darauf zurückzuführen, dass Hintergrundbereiche korrekt als falsch erkannt wurden. Deutlich wurde dabei die Fähigkeit des Klassifikators, zu generalisieren. Während des gesamten Trainings wurden lediglich acht von fünfzehn Personen als positiv deklariert. In jedem Schritt konnten jedoch mindestens dreizehn der fünfzehn Personen vollständig erkannt werden. Die Restlichen teilweise.

Ausblick

Das entstandene System bietet einige grundlegende Funktionen einen Klassifikator interaktiv zu trainieren. Erweiterungen dahin gehend sind natürlich durchaus denkbar. Beispielsweise könnte der Benutzer dazu befähigt werden die Gestalt der Kaskade, durch Löschen oder Einfügen von Knoten an beliebiger Stelle, zu verändern. Damit wäre es notwendig ihm die Möglichkeit einzuräumen, Eigenschaften der Knoten festzulegen bzw. diese zu verändern. Dahin gehend ist momentan nur die Adaption der Schwellwerte der Knoten möglich. Weitere Eigenschaften, die beeinflusst werden könnten, sind im Folgenden aufgelistet:

- Die Anzahl der Merkmale eines Knotens.
- Der minimal erlaubte Fehler denn ein Knotenmerkmal aufweisen darf.
- Die Mindestanzahl an Beispielen, mit denen ein Merkmal trainiert werden muss, bevor es zu einem Knoten hinzugefügt werden kann.

Denkbar ist auch eine voll automatisierte Auswertung eines Videos. Das heißt, der Benutzer gibt nicht mehr die Intervalle an in denen die Robustheit und die Detektionssicherheit einer Detektion liegen müssen. Gemäß dem Active Learning würden hier automatisch Detektionen ausgewählt, bei denen der Klassifikator eine hohe Unsicherheit aufweist.

Bisher ist es nur möglich die gesamte Kaskade zu trainiert. Dahin gehend könnte das System erweitert werden, um gezielt einzelne Knoten oder Merkmale mit einer Beispielmenge zu trainieren.

Um die tatsächliche Leistungsfähigkeit zu bewerten, könnte eine weitere Komponente erstellt werden. Durch diese könnte ein Testvideo angegeben werden, indem die Positionen aller nötigen Objekte bekannt sind. Das Testvideo wird dann durch den Klassifikator ausgewertet. Da die Positionen der Objekte bekannt sind, könnten automatisch Leistungskennwerte berechnet werden:

- Genauigkeit
- Wiederholgenauigkeit
- Detektionsrate
- Falschdetektionsrate

Die Berechnung könnte nach jedem Training erfolgen. Die erhaltenen Werte würden gespeichert werden. Bei Bedarf könnte eine Leistungskurve des Klassifikators für die bisherigen Trainingszyklen erstellt werden.

Des Weiteren müssten noch einige Modifikationen am System erfolgen, sodass es fähig ist, mit großen Datenmengen umzugehen. Momentan ist lediglich eine Auswertung von 250 Frames bei einer Auflösung von $720x576$ möglich. Damit ergeben sich c.a. 4.2 Millionen Suchfenster die getestet werden. Die Auswertung dieser kann durch eine Parallelisierung beschleunigt werden. Dies ist jedoch nicht der kritische Punkt. Mehr Aufmerksamkeit sollte der Zeichenroutine und der dafür benötigten Datenstruktur gewidmet werden. Die Erstellung der Datenstruktur nimmt momentan c.a. 60% der Zeit für die Auswertung in Anspruch.

Literaturverzeichnis

[BL92] E. B. Baum, K. Lang. Query learning can work poorly when a human oracle is used. In *International Jount Conference in Neural Networks*. 1992. (Zitiert auf Seite 13)

[Bre96] L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California at Berkeley, 1996. URL http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.30.8572&rep=rep1&type=pdf. (Zitiert auf Seite 19)

[Bre98] L. Breiman. Arcing Classifiers. *The Annals of Statistics*, 26(3):801–824, 1998. URL http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.30.9410&rep=rep1&type=pdf. (Zitiert auf Seite 19)

[Bre99] L. Breiman. Prediction games and arcing algorithms. *Neural Comput.*, 11:1493–1517, 1999. doi:10.1162/089976699300016106. URL http://dl.acm.org/citation.cfm?id=334369.334370. (Zitiert auf Seite 19)

[CAL94] D. Cohn, L. Atlas, R. Ladner. Improving Generalization with Active Learning. *Mach. Learn.*, 15:201–221, 1994. doi:10.1023/A:1022673506211. URL http://dl.acm.org/citation.cfm?id=189256.189489.

[CGJ95] D. A. Cohn, Z. Ghahramani, M. I. Jordan. Active Learning with Statistical Models, 1995. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.36.1392&rep=rep1&type=pdf.

[Fre90] Y. Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the third annual workshop on Computational learning theory*, COLT '90, pp. 202–216. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. URL http://dl.acm.org/citation.cfm?id=92571.92640. (Zitiert auf Seite 19)

[FS97] Y. Freund, R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55:119–139, 1997. doi:10.1006/jcss.1997.1504. URL http://dl.acm.org/citation.cfm?id=261540.261549. (Zitiert auf Seite 19)

[GB06] H. Grabner, H. Bischof. On-line Boosting and Vision. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, pp. 260–267. IEEE Computer Society, Washington, DC, USA, 2006. doi:10.1109/CVPR.2006.215. URL http://dl.acm.org/citation.cfm?id=1153170.1153451. (Zitiert auf den Seiten 7, 38 und 39)

[GGB06] H. Grabner, M. Grabner, H. Bischof. Real-Time Tracking via On-line Boosting. *Proceedings of the British Machine Vision Conference (BMVC'06)*, 1:47–56, 2006. URL http://www.macs.hw.ac.uk/bmvc2006/papers/033.pdf.

[KK06] A. Karmaker, S. Kwek. A boosting approach to remove class label noise. *Int. J. Hybrid Intell. Syst.*, 3:169–177, 2006. URL http://dl.acm.org/citation.cfm?id=1366989.1366993.

[KMS+08] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, H. Ziegler. Visual Analytics: Scope and Challenges. In S. J. Simoff, M. H. Böhlen, A. Mazeika, editors, *Visual Data Mining*, pp. 76–90. Springer-Verlag, Berlin, Heidelberg, 2008. doi:http://dx.doi.org/10.1007/978-3-540-71080-6_6. URL http://dx.doi.org/10.1007/978-3-540-71080-6_6. (Zitiert auf Seite 34)

[LS06] M. Li, I. K. Sethi. Confidence-based active learning. *IEEE transactions on pattern analysis and machine intelligence*, pp. 1251–1261, 2006. URL http://www.computer.org/portal/web/csdl/doi/10.1109/TPAMI.2006.156. (Zitiert auf den Seiten 29 und 30)

[MH08] L. van der Maaten, G. Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf. (Zitiert auf Seite 24)

[NBB] T. T. Nguyen, N. D. Binh, H. Bischof. Efficient Boosting-Based Active Learning for Specific Object Detection Problems. URL http://www.waset.org/journals/waset/v41/v41-62.pdf.

[NBB08] T. T. Nguyen, N. D. Binh, H. Bischof. An active boosting-based learning framework for real-time hand detection. In *FG*, pp. 1–6. 2008. URL http://dx.doi.org/10.1109/AFGR.2008.4813315.

[Oza05] N. C. Oza. Online Bagging and Boosting. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pp. 2340–2345. IEEE, 2005. doi:10.1109/ICSMC.2005.1571498. URL http://dx.doi.org/10.1109/ICSMC.2005.1571498. (Zitiert auf den Seiten 24, 37 und 38)

[Rob07] J. C. Roberts. State of the Art: Coordinated & Multiple Views in Exploratory Visualization. In *Proceedings of the Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pp. 61–71. IEEE Computer Society, Washington, DC, USA, 2007. doi:10.1109/CMV.2007.20. URL http://dl.acm.org/citation.cfm?id=1270380.1270575. (Zitiert auf Seite 33)

[Sch90] R. E. Schapire. The Strength of Weak Learnability. *Mach. Learn.*, 5:197–227, 1990. doi:10.1023/A:1022648800760. URL http://dl.acm.org/citation.cfm?id=83637.83645. (Zitiert auf Seite 19)

[Set09] B. Settles. Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. URL http://active-learning.net/. (Zitiert auf den Seiten 11 und 15)

[SG10] C. Seifert, M. Granitzer. User-Based Active Learning. In *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, ICDMW '10, pp. 418–425. IEEE Computer Society, Washington, DC, USA, 2010. doi:http://dx.doi.org/10.1109/ICDMW.2010.181. URL http://dx.doi.org/10.1109/ICDMW.2010.181. (Zitiert auf Seite 32)

[Shn94] B. Shneiderman. Dynamic Queries for Visual Information Seeking. *IEEE Softw.*, 11:70–77, 1994. doi:http://dx.doi.org/10.1109/52.329404. URL http://dx.doi.org/10.1109/52.329404.

[Shn96] B. Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, pp. 336–. IEEE Computer Society, Washington, DC, USA, 1996. URL http://dl.acm.org/citation.cfm?id=832277.834354. (Zitiert auf Seite 34)

[SS99] R. E. Schapire, Y. Singer. Improved Boosting Algorithms Using Confidence-rated Predictions. *Mach. Learn.*, 37:297–336, 1999. doi:10.1023/A:1007614523901. URL http://dl.acm.org/citation.cfm?id=337859.337870.

[ST10] S. Sivaraman, M. M. Trivedi. A general active-learning framework for on-road vehicle recognition and tracking. *Trans. Intell. Transport. Sys.*, 11:267–276, 2010. doi:http://dx.doi.org/10.1109/TITS.2010.2040177. URL http://dx.doi.org/10.1109/TITS.2010.2040177. (Zitiert auf den Seiten 30 und 31)

[TSHT03] G. Tur, R. E. Schapire, D. Hakkani-Tür. Active Learning For Spoken Language Understanding. In *in Proceedings of the ICASSP, Hong Kong*. 2003. URL http://www.cs.princeton.edu/~schapire/papers/active-speech.pdf.

[VJ04] P. Viola, M. J. Jones. Robust Real-Time Face Detection. *Int. J. Comput. Vision*, 57:137–154, 2004. doi:10.1023/B:VISI.0000013087.49260.fb. URL http://dl.acm.org/citation.cfm?id=966432.966458. (Zitiert auf den Seiten 17, 18, 35 und 42)

[VSF08] I. Visentini, L. Snidaro, G. L. Foresti. On-line boosted cascade for object detection, 2008. doi:10.1109/ICPR.2008.4761053. URL http://dx.doi.org/10.1109/ICPR.2008.4761053. (Zitiert auf den Seiten 7, 37 und 38)

[VSF10] I. Visentini, L. Snidaro, G. L. Foresti. Cascaded online boosting. *J. Real-Time Image Process.*, 5:245–257, 2010. doi:http://dx.doi.org/10.1007/s11554-010-0154-9. URL http://dx.doi.org/10.1007/s11554-010-0154-9.

[WRM03] J. Wu, J. M. Rehg, M. D. Mullin. Learning a Rare Event Detection Cascade by Direct Feature Selection. In *In NIPS*. MIT Press, 2003. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.69.8770&rep=rep1&type=pdf. (Zitiert auf Seite 22)

Alle URLs wurden zuletzt am 14.12.2011 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Rudolf Netzel)