Holistische Modellierung und Interpretation von Szenen und Situationen basierend auf symbolischen, probabilistischen und subsymbolischen Modellen

Von der Fakultät für Informatik, Wirtschafts- und Rechtswissenschaften der Carl von Ossietzky Universität Oldenburg zur Erlangung des Grades und Titels eines

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

angenommene Dissertation

von Herrn Paulin Pékezou Fouopi

geboren am 08.09.1983 in Bafoussam / Kamerun

Gutachter
Prof. Dr. Frank Köster

Weitere Gutachter
Prof. Dr. Philipp Slusallek

Tag der Disputation: 08.11.2019

Inhaltsverzeichnis

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig und unter Nutzung der angegebenen Quellen und Hilfsmittel verfasst habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Braunschweig, den 27.06.2019

Paulin Pékezou Fouopi

Zusammenfassung

Automatisierte Fahrsysteme benötigen zur Planung kollisionsfreier Trajektorien eine robuste und echtzeitfähige Modellierung und Interpretation von Szenen und Situationen. Die Aufgaben der Szenen- und Situationserfassung sind sehr komplex, da eine große Diversität an Faktoren wie Szenenbeleuchtung, Witterung, Sensorrauschen, Verdeckungen, Objektklassen, Relationen zwischen Szenenobjekten und allgemeines Wissen berücksichtigt werden müssen. Darüber hinaus müssen die rechenaufwendigen Erfassungsalgorithmen im Allgemeinen in Echtzeit in eingebetteten Systemen mit eingeschränkten Ressourcen laufen.

In der Literatur basierten die Ansätze zur Erfassung von Szenen und Situationen hauptsächlich auf Wissensbasen (WB), probabilistischen graphischen Modellen (PGM) und tiefen neuronalen Netzen (TNN). Während diese Methoden für einzelne Aufgaben der Szenen- und Situationserfassung sehr gute Ergebnisse liefern, ist eine Kombination dieser Methoden immer noch eine Herausforderung. In der Literatur zeigte sich, dass die holistische Betrachtung von Szenenelementen und Situationsaspekten verwendet wurde, um die Erfassung von Szenen und Situationen anhand von Kontextinformationen zu verbessern. Allerdings hat die holistische Betrachtung den Nachteil, dass die Komplexität des Systems durch das modellierte Kontextwissen erhöht wird. Darüber hinaus bleibt die explizite Modellierung des Vorwissens, die meistens manuell erfolgt, sehr aufwendig.

In dieser Arbeit wird ein schichtbasiertes System vorgeschlagen, das WB-, PGM- und TNN-basierte Module zur holistischen Erfassung von Szenen und Situationen kombiniert. Die WB modellieren Kontextinformationen als Vorwissen über Szenenelemente und Situationsaspekte explizit auf einer symbolischen Ebene mithilfe von Ontologien und Regelbasen. Um die WB mit Unsicherheiten zu erweitern, werden Teile dieser WB in Form von PGM abgebildet. TNN, die subsymbolische Informationen wie z. B. Kamerabilder verarbeiten, ergänzen das System. Methoden zur optimalen Integration der unterschiedlichen Module des Systems, zur Modellierung, zum Lernen, zur Inferenz und zur Handhabung der Komplexität des Systems werden erläutert. Dazu werden Konzepte zum Umgang mit Inkonsistenzen zwischen dem modellierten Vorwissen und den Evidenzen sowie mit Unstimmigkeiten zwischen den Modulen des Systems eingebracht. Das System hat den Vorteil, dass es gleichzeitig symbolische und subsymbolische Informationen verarbeiten kann. Weiterhin können Teile des Systems aufgrund der expliziten und symbolischen Modellierung leicht interpretiert und erweitert werden. Ferner ist der Aufwand zur softwaretechnischen Erweiterung der Komponente des modular aufgebauten Systems minimal.

Die Anwendung des vorgeschlagenen Systems zur kontextkonsistenten Evaluation der semantischen Segmentierung von Kamerabildern mit TNN basierend auf der *intersection-over-union*-Metrik zeigt eine relative Verbesserung der Segmentierung einiger Klassen um bis zu 101 % im Vergleich zu der Evaluation ohne Kontextinformationen. Darüber hinaus zeigen die Ergebnisse der Konsistenzschätzung der Regionen aus der semantischen Segmentierung, dass bis zu 13,5 % der falsch segmentierten Regionen inkonsistent sind, während nur 4,4 % der richtig segmentierten Regionen inkonsistent sind. Die Konsistenzschätzung könnte deshalb zur Verbesserung der semantischen Segmentierung verwendet werden. Eine weitere Anwendung des Systems zeigt anhand des Beispiels „*Kind folgt Ball*“, dass dieses Systems geeignet ist, die zeitliche Entwicklung von seltenen Situationen mit minimalem Aufwand explizit zu modellieren und zu lernen sowie kontextkonsistent und in Echtzeit zu prädizieren.

Abstract

In order to plan safe trajectories, automated driving systems require robust and real-time algorithms for modeling and understanding of scenes and situations. Modeling and understanding of scenes and situations remain very complex tasks, since a large diversity of factors such as scene illumination, weather, sensor noise, occlusion, object classes, relations between scene objects and general knowledge must be considered. In addition, these complex algorithms must run in real-time on embedded systems, where the resources are limited.

Approaches for modeling and understanding of scenes and situations in the literature were mainly based on three methods: knowledge bases (KB), probabilistic graphical models (PGM) and deep neural networks (DNN). While approaches based on these methods provide very good results for individual tasks of scenes and situations modeling and interpretation, a combination of these methods remains a challenge. The literature also showed that, the holistic modeling of scenes and situation using context information improved the detection of scene elements and the interpretation of situation aspects. However, holistic approaches mostly lead to high system complexity due to the modeled prior knowledge. Moreover, explicit modelling of prior knowledge remains laborious, since the modelling is usually done manually.

This work proposes a layer-based system that combines three modules based on KB, PGM and DNN for holistic modeling and understanding of scenes and situations. The KB-based module consists of symbolic models, which explicitly represent context information as prior knowledge about scene elements and situation aspects using ontologies and logical rules. Furthermore, parts of the KB, which contain uncertainties, are described in the second module based on PGM. The last module of the system uses DNN to process sub-symbolic information such as camera images. Methods to optimally integrate, model, learn and infer the different modules of the system as well as concepts for handling the complexity of the system are explained. Moreover, concepts to deal with inconsistencies between the modeled prior knowledge and the evidence as well as with inconsistencies between the different modules of the system are addressed. The proposed system has the advantage that it can process symbolic and sub-symbolic information simultaneously. Furthermore, parts of the system, which are explicitly and symbolic modeled, can be easily interpreted and extended. In addition, the software modules of the system can be extended with minimal efforts, since the system is modular, and layer based.

The application of the proposed system, which uses context information for evaluating the semantic segmentation of camera images with DNN, shows a relative improvement of the intersection-over-union metric for some classes of up to 101% compared to the evaluation of the sematic segmentation without context information. Additionally, the results of the consistency estimation of the semantic segmented regions show that up to 13.5% of the wrong-segmented regions are inconsistent, while only 4.4% of the correctly segmented regions are inconsistent. The consistency estimation could therefore be used to improve semantic segmentation. Another application of the system shows with the example "*Child follows Ball*" that this system is suitable to explicitly model, learn and context-consistently predict the temporal evolution of rare situations in real time while keeping the system tractable.

Danksagung

Diese Arbeit entstand im Rahmen meiner wissenschaftlichen Tätigkeit am Institut für Verkehrssystemtechnik (TS) des DLR in Braunschweig. Hiermit möchte ich den Menschen, die mich bei der Verfassung dieser Arbeit unterstützt haben, danken.

Ich möchte mich zuerst bei meinem Doktorvater Prof. Dr. Frank Köster für die fachliche Betreuung, Unterstützung und Forderung tief bedanken. Ebenso bin ich meinem Zweitprüfer, Prof. Dr.-Ing. Philipp Slusallek für die hilfreichen Gespräche und konstruktiven Hinweise sehr dankbar. Als weitere, möchte ich meinem Gruppen- und Abteilungsleiter Sascha Knake-Langhorst insbesondere für die Erschaffung der Rahmenbedingungen, die die Verfassung dieser Arbeit erleichtert haben, sowie für den regelmäßigen Austausch, die fachliche Betreuung und die Unterstützung besonders danken. Außerdem gilt ein besonderer Dank dem Institutsleiter Prof. Dr.-Ing. Karsten Lemmer für die Möglichkeit, die ich bekommen habe, diese Arbeit im Institut TS zu verfassen. Weiterhin bedanke ich mich bei ihm für die konstruktiven Gespräche.

Darüber hinaus möchte ich meinen Kolleginnen und Kollegen im Institut TS für die großartige Zusammenarbeit und die schönen Momente danken. Besonders mochte ich mich bei Dr.-Ing. Álvaro Catalá Prat, Kay Gimm, Gurucharan Srinivas, Joshua Niemeijer, Stephan Lapoehn, Daniel Heß, David Käthner, Jonas Rieck, Claus Kaschwich und Tamara Scharf bedanken.

Schließlich, danke ich meiner ganzen Familie, insbesondere meiner Frau, meinen Kindern, meinen Eltern und Geschwistern sowie meinen Freunden für die großartige Unterstützung und Motivation. Durch sie konnte ich die Herausforderungen dieser Arbeit meistern.

1 Einleitung

In diesem Kapitel wird zunächst die Motivation des automatisierten Fahrens vorgestellt. Danach werden Probleme, auf die automatisierte Fahrsysteme bei der Umfelderfassung stoßen, erläutert. Basierend auf diesen Problemen werden die Fragestellungen dieser Arbeit formuliert. Dazu werden die Ziele und erwarteten Ergebnisse des Systems, das in dieser Arbeit vorgeschlagen wird, vorgestellt. Die Darstellung des Aufbaus dieser Arbeit wird dieses Kapitel abschließen.

1.1 Motivation

Das automatisierte Fahren hat als Ziel die Verbesserung der Effizient und der Umweltfreundlichkeit des Straßenverkehrs. Darüber hinaus werden durch den Einsatz von automatisierten Fahrsystemen der Fahrkomfort und die Sicherheit im Straßenverkehr erhöht. Weiterhin wird die Fahrzeit effizienter genutzt. [2, 3]. Um Klarheit im Bereich des automatisierten Fahrens zu schaffen, wurden von der *Society of Automotive Engineers* (*SAE*) unterschiedliche Automatisierungsstufen unter der Norm *SAE J3016* als Standard vorgeschlagen [4, 5]:

Stufe 0 – Keine automatisierte Fahrfunktion: Der Fahrer übernimmt alle Fahraufgaben.

Stufe 1 – Assistiertes Fahren: Der Fahrer übernimmt alle Fahraufgaben und wird dabei von Fahrerassistenzsystemen in der Längs- oder Querführung unterstützt.

Stufe 2 – Teilautomatisierte Fahrfunktionen: Fortgeschrittene Fahrerassistenzsysteme sind in der Lage, das Fahrzeug in bestimmten Situationen gleichzeitig in der Längs- oder Querrichtung zu führen. Der Fahrer beobachtet jederzeit das System sowie das Umfeld und führt andere Fahraufgaben aus.

Stufe 3 – Bedingte automatisierte Fahrfunktionen: Das automatisierte Fahrsystem ist in der Lage, in bestimmten Situationen alle Aspekte der Fahraufgabe zu übernehmen. Der Fahrer muss immer bereit sein, auf Forderung des automatisierten Fahrsystems die Kontrolle des Fahrzeugs zu übernehmen. Jedoch muss der Fahrer das Umfeld nicht jederzeit beobachten, wenn das automatisierte Fahrsystem das Fahrzeug fährt.

Stufe 4 – Hochautomatisierte Fahrfunktionen: Das automatisierte Fahrsystem ist in der Lage, in bestimmten Situationen alle Aspekte der Fahraufgabe zu übernehmen. Dazu muss das automatisierte Fahrsystem in der Lage sein, das Fahrzeug in einen sicheren Zustand zu führen, wenn der Fahrer die Kontrolle nicht übernehmen kann. Der Fahrer muss das Umfeld nicht beobachten, wenn das Fahrzeug von dem automatisierten Fahrsystem kontrolliert wird. Der Fahrer sollte die Möglichkeit haben, die Kontrolle des Fahrzeugs zu übernehmen.

Stufe 5 – Vollautomatisierte Fahrfunktionen: Das autonome Fahrsystem ist in der Lage, in allen Situationen alle Aspekte der Fahraufgabe zu übernehmen. Der Fahrer sollte die Möglichkeit haben, die Kontrolle des Fahrzeugs zu übernehmen.

Während die *SAE*-Stufen 0, 1 und 2 auf dem Markt als Serienprodukte etabliert sind, stellt die *SAE*-Stufe 3 hohe Anforderungen an die Interaktion zwischen dem Fahrer und dem Automationssystem. Eine wesentliche Frage, die hier beantwortet werden soll, ist, wie viel Zeit ein Fahrer braucht, um die Kontrolle des Fahrzeugs in einer bestimmten Situation zu übernehmen. Bis jetzt bietet lediglich der Hersteller Audi ein Serienfahrzeug der Klasse A8 mit *SAE*-Stufe 3 an [6]. Die *SAE*-Stufen 4 und 5 setzen voraus, dass das technische System einige bzw. alle Situationen beherrschen muss. Dies ist eine große Herausforderung bezogen auf die Diversität und die Komplexität von Verkehrssituationen. Aufgrund der Herausforderungen der Stufen 3 bis 5 wurden diese Stufen in den Projekten wie *PROMOTHEUS* [7, 8], *DARPA Grand/Urban Challenge* [9, 10], *HAVEit* [11], *AdaptiVe* [12], *Stadtpilot* [13], *AutoMate* [14] und *Google Self-Driving Car* [15] prototypisch implementiert, getestet und validiert. Weiterhin verfügen Automobilhersteller wie z. B. Daimler [16, 17], Volvo [18] und Audi [19] über Prototypen, die in der Lage sind, bis *SAE*-Stufe 4 zu fahren.

In Zusammenarbeit mit den traditionellen Automobilherstellern haben seit einigen Jahren weitere Unternehmen und Start-ups wie *Waymo* [20], *Uber* [21], *Zoox* [22], *Aurora* [23], etc. Prototypen entwickelt, die vor allem neue Mobilitätsdienste mit *SAE*-Stufe 4 in den Bereichen *Robo-Taxi* und *Carsharing* anbieten sollen. Diese Dienste sind im Vergleich zum privaten Personenkraftwagen (PKW) mit *SAE*-Stufe 4 attraktiver, da die Anforderungen an das automatisierte Fahrsystem für solche Dienste geringer als bei privaten PKW ist. Darüber hinaus sind die Geschäftsmodelle dieser neuen Mobilitätsdienste mit *SAE*-Stufe 4 rentabler als bei privaten PKW mit *SAE*-Stufe 4 [3].

Um die Fahraufgaben zu erledigen, muss das automatisierte Fahrzeug an erster Stelle das Fahrzeugumfeld wahrnehmen und interpretieren. Dafür müssen folgende Aufgaben gelöst werden:

1. Die Schätzung der Position und Bewegung des Ego-Fahrzeugs
2. Die Szenenerfassung
    a. Die Erkennung von Freiraum und Hindernissen
    b. Die Objekterkennung
    c. Die semantische Segmentierung der Szene
3. Die Situationserfassung
    a. Die Schätzung der Relevanz von Verkehrsteilnehmern
    b. Die Erkennung und Prädiktion der zeitlichen Entwicklung von Situationen
    c. Die Schätzung und Prädiktion der Kritikalität von Situationen

Die o. g. Aufgaben sind aus folgenden Gründen schwer zu lösen: Die Vielfältigkeit und Komplexität von Situationen und Szenen vor allem im urbanen Raum erschwert die Suche nach Merkmalen, um Szenen- und Situationselemente robust und zuverlässig zu beschreiben. Objekte, auch derselben Klasse, können von einer Szene zu einer anderen aufgrund von Sensoreinflussfaktoren wie die Temperatur, die Beleuchtung, das Material, die Witterung, Verdeckungen und Rauschen anders aussehen. Ferner hängt die zeitliche Entwicklung einer Situation nicht nur von Faktoren ab, die in der Szene sichtbar sind, sondern auch von allgemeinem Wissen. Die Algorithmen zu Erfassung von Szenen und Situationen sind aufgrund der o. g. Komplexität rechenaufwendig. Gleichzeitig soll die Erfassung von Szenen und Situationen in Echtzeit unter Beachtung von eingeschränkten Ressourcen in eingebetteten Systemen gelöst werden. Aufgrund der genannten Schwierigkeiten werden unterschiedliche Methoden verwendet, um die Aufgaben der Umfelderfassung zu lösen. Die wichtigsten Methoden sind wissensbasierte Modelle, probabilistische graphische Modelle (PGM) und tiefe neuronale Netze (TNN).

Für die Objekterkennung und die semantische Segmentierung werden oft Ansätze des maschinellen Lernens verwendet. Diese Ansätze haben zwei Phasen: die Extraktion von Merkmalen aus den Sensordaten und die Klassifikation. TNN sind Modelle, die aktuell die besten Detektionsraten liefern. Diese Modelle haben den Vorteil, dass Klassifikationsmerkmale anhand von großen Datenmengen automatisch gelernt werden. Solche Merkmale führen zur besseren Klassifikation im Vergleich zu Merkmalen, die von Menschen entworfen werden. Obwohl TNN die besten Ergebnisse liefern, haben diese Modelle folgende Nachteile:

1. *Blackbox*-Modelle: Aufgrund der Komplexität der TNN-Modelle bleibt es schwierig, die Entscheidung dieser Modelle zu erklären. Damit kann nicht garantiert werden, welche Entscheidung diese Modelle in Situationen treffen werden, die in den Trainingsdaten kaum bzw. nicht repräsentiert waren.
2. Mangel an expliziten Kontextinformationen: TNN-Modelle benutzen ein lokales rezeptives Feld zur Generierung von Merkmalen und lernen Kontextinformationen implizit. Damit sind sie nicht in der Lage, den globalen Kontext zu erfassen. Eine direkte Modellierung der Kontextabhängigkeiten in solchen Modellen bleibt offen.
3. Komplexität des Trainings- und Inferenzprozesses: Aufgrund der hohen Anzahl an Parametern benötigen TNN-Modelle viel Zeit und viele Daten für das Lernen. Ebenso ist die Inferenz rechenaufwendig.

Für die Prädiktion der zeitlichen Entwicklung von Situationen werden meistens PGM-Modelle verwendet. Diese Modelle bieten die Möglichkeit an, das Wissen über das Verhalten von Verkehrsteilnehmern in Form von Graphen zu modellieren. Darüber hinaus können die PGM-Modelle Unsicherheiten, die in realen Szenen existieren, anhand von Wahrscheinlichkeitsverteilungen modellieren. Die Inferenz mit PGM-Modellen liefert eine kontextkonsistente Prädiktion der zeitlichen Entwicklung von Situationen. Jedoch haben PGM-Modelle vor allem den Nachteil, dass die Modellierung oft manuell erfolgt und bei komplexen Aufgaben mit viel Aufwand verbunden ist. Darüber hinaus ist die Inferenz von komplexeren PGM-Modellen rechenaufwendig. Wissensbasierte Ansätze, die das Wissen formal und explizit modellieren, können verwendet werden, um die zeitliche Situationsentwicklung zu prädizieren. Der Vorteil dieser Ansätze ist, dass die Entscheidungen des Systems sich mit einfachen Verfahren erklären lassen, da das Wissen symbolisch modelliert wird. Der Hauptnachteil hier wie bei PGM-Modellen bleibt die Komplexität der Modellierung des Wissens, da diese Modellierung oft manuell erfolgt. Darüber hinaus können wissensbasierte Modelle keine Unsicherheiten in dem Wissen modellieren.

Die holistische Betrachtung von Szenen- und Situationselementen, das heißt, die Modellierung der Abhängigkeiten von Szenenelementen zu Situationselementen sowie von Situationselementen zu Szenenelementen kann zur kontextkonsistenten Szenen- und Situationserfassung beitragen. Allerdings ist die Modellierung von Kontextabhängigkeiten oft mit der Erhöhung der Komplexität des Systems verbunden.

1.2 Fragestellung der Arbeit

Obwohl wissensbasierte Modelle, PGM und TNN für unterschiedliche Aufgaben der Umfelderfassung geeignet sind, ist keine dieser Methoden für alle Aufgaben der Umfelderfassung optimal. Darüber hinaus wird die holistische Modellierung von Szenen und Situationen aufgrund der Komplexität des Systems oft nicht adressiert. In dieser Arbeit sollten deswegen folgende Fragen erörtert werden:

1. Wie können wissensbasierte Modelle, PGM und TNN in einem System zur Umfelderfassung optimal kombiniert werden?
2. Wie kann die holistische Betrachtung von Szenen und Situationen in Echtzeit zur Verbesserung der Umfelderfassung beitragen?

Um die o. g. Fragen zu beantworten, wird ein System entworfen, prototypisch implementiert und validiert. Das System muss folgende Anforderungen erfüllen:

- Die Modellierung von Vorwissen mit Unsicherheiten
- Die einfache Erklärbarkeit der Entscheidungen des Systems
- Die einfache Erweiterung des Systems sowohl aus softwaretechnischer als auch aus inhaltlicher Sicht
- Echtzeitfähigkeit

Das entworfene System wird verwendet, um folgende Aufgaben der Umfelderfassung zu lösen:

- Die Analyse der Güte der semantischen Segmentierung unter Betrachtung von Kontextinformationen
- Die Erkennung von Inkonsistenzen der semantischen Segmentierung basierend auf Kontextinformationen
- Die Prädiktion der zeitlichen Entwicklung von seltenen Situationen basierend auf Kontextabhängigkeiten

Sind die o. g. Aufgaben der Umfelderfassung gelöst, werden folgende Ergebnisse erwartet:

- Trainiertes und evaluiertes Modell zur semantischen Segmentierung
- Wissensbasis (WB) mit Kontextinformationen einiger relevanter Elemente von Szenen und Situationen im Verkehr

- Trainierte und evaluierte Modelle zur Erkennung von Inkonsistenzen der semantischen Segmentierung
- Trainierte und evaluierte Modelle zur kontextkonsistenten Prädiktion der zeitlichen Situationsentwicklung
- Konzepte zur systematischen Erkennung und Handhabung von Inkonsistenzen zwischen den Kontextinformationen und den Evidenzen

1.3 Aufbau der Arbeit

In der Abbildung 1-1 ist der Aufbau der Arbeit grafisch dargestellt. In Kapitel 2 werden die theoretischen Grundlagen für die Methoden, die in dieser Arbeit verwendet werden, vorgestellt. Das Kapitel 3 wird den Stand der Technik für die Lösung der Aufgaben der Szenen- und Situationserfassung darstellen. Das Konzept, das in dieser Arbeit vorgeschlagen wird, wird in Kapitel 4 veranschaulicht. In Kapitel 5 wird die semantische Segmentierung von Kamerabildern vorgestellt. Das Kapitel 6 beschäftigt sich mit der Erkennung von Inkonsistenzen der semantischen Segmentierung. Das Kapitel 7 adressiert die Prädiktion der zeitlichen Entwicklung von seltenen Situationen. Diese Arbeit wird in Kapitel 8 zusammengefasst.

<table>
<tr><td colspan="3">Kapitel 2: Theoretische Grundlagen</td></tr>
<tr><td colspan="3">Kapitel 3: Stand der Technik</td></tr>
<tr><td colspan="3">Kapitel 4: Konzept zur holistischen Szenen- und Situationserfassung</td></tr>
<tr><td>Kapitel 5: Semantische Segmentierung von Kamerabildern</td><td>Kapitel 6: Erkennung von Inkonsistenzen der semantischen Segmentierung</td><td>Kapitel 7: Kontextabhängige Prädiktion der zeitlichen Situationsentwicklung</td></tr>
<tr><td colspan="3">Kapitel 8: Zusammenfassung und Ausblick</td></tr>
</table>

Abbildung 1-1: Grafische Darstellung des Aufbaus der Arbeit

1.4 Liste der eigenen Veröffentlichungen

In der ersten Spalte der Tabelle 1-1 sind die Veröffentlichungen, die im Rahmen dieser Arbeit entstanden sind, aufgelistet. Die Kapitel, die die Inhalte der Veröffentlichungen darstellen sind in der zweiten Spalte der Tabelle 1-1 zu finden.

Tabelle 1-1: Liste der eigenen Veröffentlichungen

Veröffentlichung	Kapitel
Pekezou Fouopi, P.; Srinivas, G.; Knake-Langhorst, S.; Köster, F.; Niemeijer, J.: Holistische Szenenmodellierung und -Interpretation basierend auf subsymbolischen, symbolischen und probabilistischen Methoden: VDI-Fachkonferenz Umfelderfassung im Fahrzeug 2018.	4 und 6
Lapoehn, S.; Pekezou Fouopi, P.; Löper, C.; Knake-Langhorst, S.; Hesse, T.: Semantische Netze als Wissensbasis automatisierter Fahrzeuge: VDI/VW-Gemeinschaftstagung: Fahrassistenzsysteme und automatisiertes Fahren 2016.	7

2 Theoretische Grundlagen

In Kapitel 1 wurden wissensbasierte Modelle, PGM und TNN als Methoden vorgestellt, die oft zur Lösung der Aufgaben der Szenen- und Situationserfassung verwendet werden. In den folgenden Kapiteln werden die theoretischen Grundlagen dieser Methoden erläutert, da sie für diese Arbeit relevant sind. So stellt der erste Abschnitt die Begriffe und Konzepte der Wissensrepräsentation mit dem Fokus auf Ontologien und die Prädikatenlogik dar. Im zweiten Abschnitt werden PGM erläutert. Markov-Logik-Netze (MLN), die PGM und Formeln der Prädikatenlogik erster Stufe (engl. *FOL*) kombinieren, werden im dritten Abschnitt veranschaulicht. Künstliche neuronale Netze werden im vierten Abschnitt vorgestellt. Im fünften Abschnitt wird dieses Kapitel zusammengefasst.

2.1 Wissensrepräsentation

Die Wissensrepräsentation beschäftigt sich mit der symbolischen und expliziten Modellierung des Domänenwissens anhand von Symbolen und Regeln. In der Wissensrepräsentation werden zwei Gebiete unterschieden: Das erste Gebiet entwickelt Repräsentationsformalismen, Beschreibungsmittel der Symbole und Inferenzverfahren. Das zweite Gebiet, auch Wissensmodellierung genannt, beschreibt das Wissen einer Domäne anhand der Formalismen und Mittel, die im ersten Gebiet entwickelt wurden [24]. Das Ergebnis der Wissensmodellierung wird WB genannt. In der Wissensrepräsentation werden oft formale Modelle generiert, da diese Modelle maschinell verarbeitet werden können.

2.1.1 Prädikatenlogik erster Stufe (*FOL*)

FOL als Unterklasse der klassischen Logik bietet Formalismen, Mittel und Kalküle zur Wissensrepräsentation an. Diese werden in den nächsten Abschnitten erläutert.

2.1.1.1 Syntax

FOL-Formeln werden anhand folgender Symbole gebaut: Konstanten, Variablen, Funktionen und Prädikate. Konstanten sind konkrete Objekte einer Domäne. Die Menge der Objekte einer Domäne wird durch Variablen repräsentiert. Funktionen bilden Objekt-Tupel auf anderen Objekten ab. Die Prädikate modellieren Objektrelationen und Attribute [25]. In *FOL* wird die Menge der Terme wie folgt definiert [26]:

- Alle Variablen und Konstanten sind Terme.
- $f(t_1, \dots, t_n)$ ist ein Term, wobei $t_1, \dots, t_n$ Terme sind. f ist ein Funktionssymbol.
- Terme, die keine Variablen enthalten, werden Grundterme genannt.

Formeln werden anhand der Terme gemäß der folgenden Definition gebaut [26]:

- Sind $t_1, \dots, t_n$ Terme und p ein Prädikatensymbol, so ist $p(t_1, \dots, t_n)$ eine Formel. Diese Formel wird auch atomare Formel genannt.
- $p(t_1, \dots, t_n)$ und $\neg p(t_1, \dots, t_n)$ werden jeweils positive und negative Literale genannt.
- Sind F_1 und F_2 zwei Formeln, so sind auch $uopF_1$ und $F_1 bopF_2$ Formeln. $uop \in \{\neg, ()\}$ ist ein unärer logischer Operator, während $bop \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$ ein binärer logischer Operator ist.
- $\exists x\ F_1$, $\forall x\ F_1$ sind Formeln, wobei x eine Variable ist und F_1 eine Formel ist. $\exists$ und $\forall$ sind jeweils der Existenz- und Allquantor.
- Geschlossene Formeln, auch Aussagen genannt, sind Formeln, bei denen jede Variable im Wirkungsbereich eines Quantors liegt. Freie Variable sind Variablen, die nicht im Wirkungsbereich eines Quantors liegen.

Eine atomare Formel wird Grundatom oder Grundprädikat genannt, wenn sie nur Grundterme beinhaltet. Eine Grundformel ist eine Formel, die nur Grundprädikate enthält [25].

Eine Formel $F = K_1 \wedge \ldots \wedge K_n$ liegt in der konjunktiven Normalform (KNF) vor, wenn sie aus einer Konjunktion von Klauseln $K_i, i = 1, \ldots, n$ besteht und jede $K_i = L_{i1} \vee \ldots \vee L_{im}$ eine Disjunktion von Literalen ist [26].

Eine Hornklausel ist eine Klausel mit höchstens einem positiven Literal. Das positive Literal einer Hornklausel wird Kopf genannt. Eine Hornklausel mit einem einzigen positiven Literal heißt Fakt [26].

Eine *FOL*-WB $WB = \{F_i : i = 1, \ldots, n\}$ ist eine Menge von *FOL*-Formeln.

2.1.1.2 Semantik

Eine Belegung $\mathcal{B}$ oder Interepretation ist eine Abbildung der Menge von Konstanten und Variablen auf einer Menge von Objekten aus der Welt. Darüber hinaus bildet $\mathcal{B}$ die Menge der Funktions- und Prädikatsymbole jeweils auf Funktionen und Relationen der Welt ab [26].

Der Wahrheitswert einer Formel F unter der Belegung $\mathcal{B}$ wird berechnet, indem zunächst alle atomaren Formeln von F anhand von $\mathcal{B}$ ausgewertet werden. Jede atomare Formel bekommt den Wahrheitswert wahr oder falsch. Danach ergibt sich der Wahrheitswert von F durch die Auswertung der logischen Operatoren von F mit den Wahrheitswerten der ausgewerteten atomaren Formeln von F. Die Formel $\forall x\ F$ ist wahr, wenn F für alle Belegungen von x wahr ist. Existiert eine Belegung von x, wo F wahr ist, dann ist auch $\exists x\ F$ wahr [26].

Eine Formel ist erfüllbar, wenn sie für mindestens eine Belegung wahr ist. Ist eine Formel bei allen Belegungen wahr, wird sie Tautologie genannt. Eine unerfüllbare Formel ist für alle Belegungen falsch [26].

2.1.1.3 Abfrage und Inferenz

In der Logik ist die Abfrage, ob eine Formel F aus einer *FOL*-WB WB folgt (notiert $WB \vDash F$), üblich, das heißt, ob F wahr ist für alle Belegungen, für die WB wahr ist. $WB \vDash F$ gilt genau dann, wenn $WB \Rightarrow F$ eine Tautologie ist (notiert $\emptyset \vDash WB \Rightarrow F$) und WB keine freien Variablen enthält. Dies ist das Deduktionstheorem. In diesem Fall kann anhand des Widerspruchsbeweises gezeigt werden, dass $WB \wedge \neg F$ nicht erfüllbar ist [26]. Die Inferenz von F, gegeben WB und $WB \Rightarrow F$, wird deduktive Inferenz genannt.

Zur Inferenz der Abfrage $WB \vDash F$ werden Beweis- und Resolutionskalküle verwendet. Beweiskalküle sind für Anwendungen durch Menschen gedacht und werden in dieser Arbeit nur beispielhaft genannt. Ein Beispiel für einen Beweiskalkül ist der *Modus Ponens* kombiniert mit der Quantoreneliminination. Resolutionskalküle verwenden im Vergleich zu anderen Kalkülen nur zwei Regeln und nutzen Beweiskalküle, um den Suchraum zu reduzieren. Damit wird die Komplexität und die Rechenzeit reduziert [26]. Für Formeln in der KNF ist der Beweis der Unerfüllbarkeit anhand von Resolutionskalkülen korrekt und vollständig. Die WB muss für die Anwendung von Resolutionskalkülen widerspruchsfrei sein. Eine WB ist widerspruchsfrei, wenn in dieser WB keine Formeln der Form $F \wedge \neg F$ vorkommen [26].

2.1.2 Ontologien und Beschreibungslogik

Ontologien verwenden Beschreibungslogiken als Sprache zur Wissensrepräsentation, wobei viele Beschreibungslogiken Fragmente von *FOL* sind. In diesem Abschnitt werden die Grundlagen der Ontologien und Beschreibungslogiken vorgestellt. Der Bezug zu *FOL* wird präsentiert.

2.1.2.1 Definition

Eine Ontologie ist eine formale symbolische Beschreibung des Domänenwissens anhand von Begriffen und Relationen der Domäne [27]. Ontologien werden anhand von Beschreibungssprachen der Logik, auch Beschreibungslogiken genannt, modelliert. *Web Ontology Language (OWL-) 2* ist eine weitverbreite Beschreibungslogik, die zur Klasse $\mathcal{SROIQ}$ gehört [28].

Beschreibungslogiken sind oft Fragmente der *FOL*, wobei die Begriffe und die Relationen der Domäne äquivalent zu unären und binären Prädikaten sind. Ontologien können deshalb maschinell verarbei-

tet werden, was ein Vorteil ist. Relationen werden in der Beschreibungslogik auch Rollen genannt. Beschreibungslogiken bestehen aus einer Menge von Aussagen (Axiome) zur Modellierung von Fakten. Axiome werden in assertionale (*ABox*) Axiome, terminologische (*TBox*) Axiome und relationale *(RBox*) Axiome aufgeteilt [29].

TBox-Axiome beschreiben Beziehungen zwischen den Begriffen und Relationen der Domäne mithilfe der Subsumption- $\sqsubseteq$ und Äquivalenzoperatoren $\equiv$. *TBox*-Axiome können somit verwendet werden, um die Taxonomie der Begriffe aufzubauen. Der Fakt $Person \equiv Mensch$ bedeutet z. B., dass der Begriff $Person$ zum Begriff $Mensch$ äquivalent ist. Komplexe Begriffe werden mithilfe der booleschen Operatoren Konjunktion $\sqcap$, Disjunktion $\sqcup$ und Negation $\neg$ aufgebaut. Der *Top*-Begriff $\top$ ist die Menge aller Individuen und der *Bottom*-Begriff $\bot$ die leere Menge. Die Quantoren $\forall$ und $\exists$ beschreiben jeweils universelle und existenzielle Restriktionen auf Relationen. Die Vergleichsoperatoren $\leq$ und $\geq$ dienen zur Restriktion der Anzahl der Individuen. Instanzen eines Begriffs können in einer Enumeration definiert werden [29].

Die *ABox*-Axiome assoziieren Individuen zu den Begriffen und Relationen der Domäne. Individuen sind äquivalent zu Konstanten der *FOL*. Der Fakt $Mutter(Julia)$ bedeutet, dass das Individuum $Julia$ eine Instanz des Begriffs $Mutter$ ist. Viele Beschreibungslogiken setzen nicht den eindeutigen Namen von Individuen voraus. Deshalb werden die Operatoren $\approx$ und $\not\approx$ verwendet, um Individuennamen zu vergleichen [29].

Die *RBox*-Axiome definieren Eigenschaften der Relationen mithilfe von Subsumption- $\sqsubseteq$ und Äquivalenzoperatoren $\equiv$. Komplexere Relationen können anhand der Komposition $\circ$, Disjunktion $\vee$ und Inverse $-$ generiert werden. Zum Beispiel bedeutet der Fakt $ElternVon \equiv KindVon^{-}$, dass die Relation $ElternVon$ die Inverse der Relation $KindVon$ ist. Die universellen und die leeren Relationen werden manchmal in Beschreibungslogiken angeboten. Relationseigenschaften wie die Symmetrie, Reflexivität und die Transitivität können als *RBox*-Axiome ausgedrückt werden [29].

2.1.2.2 Semantik

Die Semantik von Ontologien wird durch die Semantik der Beschreibungslogik gegeben. Da viele Beschreibungslogiken Fragmente der *FOL* sind, wird ihre Semantik durch die Semantik der *FOL* vorgegeben. Da Ontologien oft nur Teile des Domänenwissens modellieren, verwenden viele Beschreibungslogiken die *Open-world-assumption*-Annahme (*OWA*) während der semantischen Interpretation. Die *OWA*-Annahme bedeutet, dass Axiome, die wegen mangelnder Informationen keinen Wahrheitswert bekommen, als unbekannt betrachtet werden [29].

Eine Begriffsübersetzungsfunktion $T^x(A) = A(x)$ und eine Relationsübersetzungsfunktion $T^{xy}(r) = r(x, y)$ werden verwendet, um die Begriffe, Relationen und Axiome einer Ontologie in die *FOL* zu überführen, wobei A ein Konzept, r eine Relation ist und x und y Variablen sind. Diese Übersetzungsfunktionen sind in der Abbildung 2-1 dargestellt. Die logische Interpretation des Axioms $A \sqsubseteq B$ bedeutet, dass das Konzept B hinreichende Eigenschaften von A definiert ($T^x(A) \rightarrow T^x(B)$). Das Axiom $A \equiv B$ bedeutet, dass das Konzept B notwendige und hinreichende Eigenschaften von A definiert und umgekehrt ($T^x(A) \Leftrightarrow T^x(B)$) [1].

Konzeptausdruck A	Übersetzung $T^x(A)$
$\top$	$x = x$
$\bot$	$x \neq x$
$\{o_1, ..., o_n\}$	$x = o_1 \vee ... \vee x = o_n$
C	$C(x)$
$C \sqcap D$	$T^x(C) \wedge T^x(D)$
$C \sqcup D$	$T^x(C) \vee T^x(D)$
$\neg C$	$\neg T^x(C)$
$\exists r$	$\exists y T^{x,y}(r)$
$\exists r.C$	$\exists y(T^{x,y}(r) \wedge T^y(C))$
$\forall r.C$	$\forall y(T^{x,y}(r) \rightarrow T^y(C))$
$\leq nr$	$\exists^{\leq n} y(T^{x,y}(r))$
$\geq nr$	$\exists^{\geq n} y(T^{x,y}(r))$
$= nr$	$\exists^{=n} y(T^{x,y}(r))$
$\leq nr.C$	$\exists^{\leq n} y(T^{x,y}(r) \wedge T^y(C))$
$\geq nr.C$	$\exists^{\geq n} y(T^{x,y}(r) \wedge T^y(C))$
$= nr.C$	$\exists^{=n} y(T^{x,y}(r) \wedge T^y(C))$

Abbildung 2-1: Übersetzung von Begriffen, Relationen und Axiomen der Deskriptionslogik zu FOL-Formeln (Tabelle aus [1])

Ist eine Ontologie anhand der o. g. Konzeptübersetzungsfunktion zu *FOL*-Formeln übersetzt worden, dann kann die Belegung $\mathcal{B}$ ähnlich wie bei der *FOL* verwendet werden, um die in *FOL* überführte Ontologie zu interpretieren (siehe Abschnitt 2.1.1.2).

Eine Ontologie heißt konsistent, wenn es mindestens eine Belegung gibt, für die alle Axiome der Ontologie erfüllbar sind [29]. In der Praxis deutet eine inkonsistente Ontologie auf Fehler in der Modellierung dieser Ontologie hin. Solche Fehler sollen beseitig werden.

2.1.2.3 Inferenz

Die Inferenz einer Ontologie wird *Reasoning* genannt. Eine Hauptaufgabe des *Reasoner* ist zu bestimmen, ob die Ontologie O konsistent ist. Dafür werden zunächst die Axiome von O in die Formelmenge F^O der *FOL*, wie im Abschnitt 2.1.2.2 beschrieben, übersetzt. Danach können *FOL*-Kalküle (siehe Abschnitt 2.1.1.3) verwendet werden, um zu prüfen, ob F^O erfüllbar ist. Ist F^O erfüllbar, dann ist die Ontologie O konsistent [30].

Eine weitere Aufgabe der Inferenz ist der Subsumptionstest. Dabei werden Subsumptionsrelationen zwischen den Begriffen der Ontologie O abgeleitet. Ist F^O die Menge von *FOL*-Formeln der Ontologie O und A und B zwei Begriffe von O, dann ist $A \sqsubseteq B$ ein Begriff von O, wenn $T^x(A) \Rightarrow T^x(B)$ aus F^O folgt ($F^O \Rightarrow \left(T^x(A) \Rightarrow T^x(B)\right)$ ist eine Tautologie) [1]. Algorithmen zur Berechnung dieser Erfüllbarkeit wurden im Abschnitt 2.1.1.3 vorgestellt. Der Subsumptionstest kann auf alle Begriffe angewendet werden, um eine Begriffshierarchie (Taxonomie) aufzubauen. Dieses Verfahren wird auch Klassifikation genannt [1].

Der Instanztest prüft, ob ein Individuum i eine Instanz des Begriffs A ist. Dafür wird geprüft, ob $F^O \Rightarrow A(i)$ eine Tautologie ist. Das *Retrival-Reasoning* inferiert alle Individuen, die Instanzen der Begriffe von O sind. Das Realisierung-*Reasoning* ist das Gegenteil des *Retrival-Reasoning*. Hier werden alle Begriffe bestimmt, für die das gegebene Individuum i eine Instanz ist [1].

Neben den Instanztest wird auch geprüft, ob die Relation $r(i,j)$ zwischen zwei Individuen i und j erfüllbar ist. Diese Relation ist erfüllbar, wenn $F^O \Rightarrow r(i,j)$ eine Tautologie ist.

Alle *Reasoning*-Aufgaben können mit Algorithmen des Abschnitts 2.1.1.3 gelöst werden, da alle diese Aufgaben sich als *FOL*-Formeln ausdrücken lassen.

Die Komplexität der Inferenz einer Ontologie hängt nicht nur von der Größe der Ontologie sondern auch von der Beschreibungslogik ab, die zur Modellierung dieser Ontologie verwendet wurde. So ist die Komplexität der Inferenz für Beschreibungslogiken der $\mathcal{SROIQ}$-Klasse *NExpTime-hard* ($O\left(2^{p(n)}\right)$), wobei p und n jeweils ein Polynom und die Anzahl der Elemente der Ontologie sind. Jedoch sind die Reasoner in der Praxis so optimiert, dass die Inferenzzeit deutlich unter $O\left(2^{p(n)}\right)$ liegt [31, 32]

2.2 Probabilistische graphische Modelle

2.2.1 Definition

PGM sind eine Kombination von Graphen und Wahrscheinlichkeiten zur Modellierung von komplexen Aufgaben [33].

Ein Graph $G = (\mathcal{V}, \mathcal{E})$ besteht aus einer Menge von Knoten $\mathcal{V}$ und Kanten $\mathcal{E}$. Eine Kante $(i,j) \in \mathcal{E}$ verbindet die Knoten $v_i, v_j \in \mathcal{V}$. In gerichteten Graphen verbindet die Kante $(i,j) \in \mathcal{E}$ den Elternknoten v_i mit dem Kindknoten v_j. Darüber hinaus existiert keine Kante (j,i). Existieren die Kanten $(i,j) \in \mathcal{E}$ und $(j,i) \in \mathcal{E}$ in dem Graph G, dann ist dieser Graph ungerichtet. Knoten werden durch Kreise grafisch dargestellt. Ungerichtete Kanten verbinden die Knoten mit Segmenten. Gerichtete Kanten verbinden die Eltern- und die Kindknoten mit Pfeilen, wobei die Pfeile in Richtung der Kindknoten zeigen.

Ein Graph G ist vollständig, wenn jeder Knoten des Graphs mit jedem anderen Knoten des Graphs mit einer Kante verbunden ist.

$$\forall v_i, v_j \in \mathcal{V}, \exists (i,j) \in \mathcal{E} \qquad (1)$$

Eine Untermenge der Knoten $U \subseteq \mathcal{V}$ ist eine Clique von G, wenn der induzierte Teilgraph $H = (U, E), E \subseteq \mathcal{E}$ vollständig ist.

In der vom Graph G induzierten Wahrscheinlichkeitsverteilung ist jeder Knoten $v_i \in \mathcal{V}$ durch die Zufallsvariable $X_i \in \mathbb{X}_i$ repräsentiert. Jede Kante $(i,j) \in \mathcal{E}$ beschreibt die probabilistische Abhängigkeit zwischen den Zufallsvariablen $X_i \in \mathbb{X}_i$ und $X_j \in \mathbb{X}_j$. $X = (X_1, \dots, X_n) \in \mathbb{X}, \mathbb{X} = \mathbb{X}_1 \times \dots \times \mathbb{X}_n$ ist die multivariate Zufallsvariable über alle Zufallsvariablen des PGM-Modells. Ein Beispielmodell ist in der Abbildung 2-2 dargestellt. Knoten können beobachtbar (siehe graugefüllte Knoten in der Abbildung 2-2) oder verdeckt (siehe weißgefüllte Knoten in der Abbildung 2-2) sein.

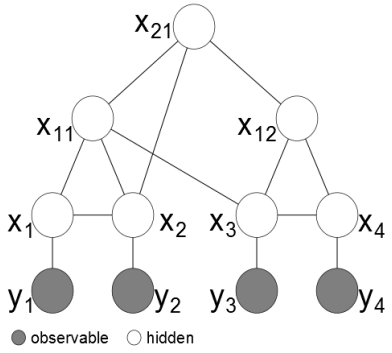


Abbildung 2-2: Beispielgraph eines PGM-Modells mit einer Baumstruktur

Ein Vorteil der PGM ist, dass sie auf den Bereichen der Graphen- und Wahrscheinlichkeitstheorie aufbauen, wobei diese Bereiche über gut etablierte Algorithmen und Methoden verfügen. Darüber hinaus ermöglichen die PGM das Aufteilen der komplexen multivariaten Verbundwahrscheinlichkeitsverteilung $P(X)$ über alle Zufallsvariablen $X_i \in \mathbb{X}_i$ des Graphs in einfacheren Verbundwahrscheinlichkeitsverteilungen über die Zufallsvariablen, die bedingt unabhängig voneinander sind. Damit wird eine komplexe Aufgabe einfacher und schneller gelöst. Die Zufallsvariable X_1 ist bedingt unabhängig von X_2 gegeben X_3 (abgekürzt $(X_1 \perp X_2 | X_3)$) für die Wahrscheinlichkeitsverteilung P wenn die Gleichung

$$P(X_1, X_2 | X_3) = P(X_1 | X_3)\, P(X_2 | X_3) \qquad (2)$$

erfüllt ist.

2.2.2 Markov-Netze

Markov-Netze (engl. *Markov Random Fields*) sind eine Klasse von PGM, die auf ungerichteten graphischen Modellen basieren. Sei $\mathcal{H}$ der Gaph eines Markov-Netzes und $C_1, \dots, C_k$ die Cliquen von $\mathcal{H}$, die multivariate Verbundwahrscheinlichkeitsverteilung $P_{\mathcal{H}}$ faktorisiert über $\mathcal{H}$

$$P_{\mathcal{H}}(X) = \frac{1}{Z} P'_{\mathcal{H}}(X) \tag{3}$$

mit

$$P'_{\mathcal{H}}(X) = \prod_{i=1}^{k} \pi_i[C_i] \tag{4}$$

und

$$Z = \sum_{X \in \mathbb{X}} P'_{\mathcal{H}}(X) \tag{5}$$

$\{\pi_i[C_i]: Val(C_i) \longrightarrow \mathbb{R}^+ | i = 1, \dots, k\}$ sind Faktorfunktionen, die auch Clique-Potentialfunktionen genannt werden. $Val(C_i)$ beschreibt die Menge aller möglichen Werte der Zufallsvariablen von C_i. Z ist die Partitionsfunktion zur Normierung der Verbundwahrscheinlichkeitsverteilung $P_{\mathcal{H}}$. $P_{\mathcal{H}}$ wird auch Gibbs-Verteilung von $\mathcal{H}$ genannt [34]. In der Praxis ist die logarithmische Transformation der Faktorfunktion in die Energiefunktion $\epsilon_i[C_i] = -ln(\pi_i[C_i])$ üblich. Damit wird sichergesellt, dass die Verbundwahrscheinlichkeitsverteilung

$$P_{\mathcal{H}}(X) = \frac{1}{Z} exp\left(-\sum_{i=1}^{k} \epsilon_i[C_i]\right) \tag{6}$$

immer positiv wird [34].

2.2.2.1 Lokale Unabhängigkeit

Der Markov-*Blanket* $N_{\mathcal{H}}(v_i) = \{v_j \in \mathcal{V}: (i,j) \in \mathcal{E}, (j,i) \in \mathcal{E}, j = 1, \dots, n, j \neq i\}$ des Knotens v_i von $\mathcal{H}$ ist die Menge aller Knoten von $\mathcal{H}$, die eine Kante zu v_i haben. Jeder Knoten v_i von $\mathcal{H}$ ist bedingt unabhängig von allen anderen Knoten von $\mathcal{H}$ gegeben $N_{\mathcal{H}}(v_i)$ [34].

2.2.2.2 Globale Unabhängigkeit

Der Weg zwischen zwei Knoten v_i und v_j von $\mathcal{H}$ heißt *aktiv*, gegeben ist die Knotenteilmenge $V \subseteq \mathcal{V}$, wenn es keinen beobachtbaren Knoten von V auf dem Weg zwischen v_i und v_j gibt. Seien $V_1 \subseteq \mathcal{V}$, $V_2 \subseteq \mathcal{V}$ und $V_3 \subseteq \mathcal{V}$ Knotenteilmengen von $\mathcal{H}$. V_3 separiert V_1 und V_2 (notiert $sep_{\mathcal{H}}(V_1; V_2|V_3)$), wenn es keinen *aktiven* Weg zwischen jedem Knoten von V_1 und V_2, gegeben V_3, gibt. Sind V_1 und V_2 von V_3 separiert, dann sind V_1 und V_2 bedingt unabhängig, gegeben V_3 [34].

$$I(\mathcal{H}) = \{(V_1 \perp V_2 | V_3): sep_{\mathcal{H}}(V_1; V_2|V_3)\} \tag{7}$$

2.2.3 Abfrage und Inferenz

Die meistverwendete Abfrage in PGM ist die Schätzung der bedingten Wahrscheinlichkeit $P(A|\,E), A \subseteq X, E \subseteq X, A \cap E = \emptyset$ der Teilmenge der Zufallsvariablen A, gegeben die Teilmenge der Zufallsvariablen E. A wird Abfrageparameter und E Evidenz genannt [34].

Eine weitere Abfrage ist die wahrscheinlichste Zuordnung von Zufallsvariablen. Wie bei der bedingten Wahrscheinlichkeit ist hier auch die Evidenz E gegeben. Gesucht ist allerdings die wahrscheinlichste Zuordnung aller nichtevidenten Zufallsvariablen, die die Verbundwahrscheinlichkeit $P(A, E)$ maximiert. Diese Art von Abfrage wird wahrscheinlichste Erklärung (*MPE: most probable explanation*) genannt [34].

$$MPE(A) = argmax_A\big(P(A,E)\big), A = X - E \tag{8}$$

Eine Generalisierung der *MPE*-Abfrage ist die *Maximum-a-posteriori*-Abfrage (*MAP*). Die *MAP*-Abfrage schätzt die wahrscheinlichste Zuordnung der Zufallsvariablen A, die die bedingte Wahrscheinlichkeit $P(A|\ E)$ maximiert [34].

$$MAP(A) = argmax_A\big(P(A|E)\big) \tag{9}$$

Um die o. g. Abfragen zu beantworten, werden Inferenzalgorithmen verwendet. Eine exakte Inferenz ist in der Praxis mit einer hohen Komplexität verbunden. Um die Komplexität der Inferenz zu reduzieren, werden Optimierungsverfahren und *Sampling*-Algorithmen (z. B. *Gibbs Sampling*) verwendet [34].

2.2.4 Lernen von PGM

Bei PGM-Modellen können die Struktur des Graphs und die Parameter der Verbundwahrscheinlichkeitsverteilung gelernt werden. In diesem Abschnitt wird sich auf das Lernen der Parameter der Verbundwahrscheinlichkeitsverteilung beschränkt.

Das Lernen der Parameter der Verbundwahrscheinlichkeitsverteilung basiert auf der *Likelihood*-Funktion. Die *Likelihood*-Funktion ist die Wahrscheinlichkeitsverteilung des PGM-Modells für gegebene Trainingsdaten. Für das Markov-Netz mit dem Graph $\mathcal{H}$ und die Trainingsdatenmenge $D = \{x^1, \dots, x^m\}$ ist die *Likelihood*-Funktion

$$L(\epsilon, D) = \prod_{j=1}^{m} \frac{1}{Z} exp\left(-\sum_{i=1}^{k} \epsilon_i\left[x_i^j\right]\right) \tag{10}$$

x_i^j ist der Wert der Clique C_i, gegeben das Trainingssample x^j. Der *Maximum-Likelihood*-Schätzer schätzt die Parameter der Energiefunktion ϵ, die die *Likelihood*-Funktion maximieren. Da die *Likelihood*-Funktion konkav in ϵ ist, kann das Maximierungsproblem anhand von Verfahren wie dem Gradientenaufstieg gelöst werden [34].

2.3 Markov-Logik-Netze

2.3.1 Definition

Von Richardson und Domingos [25] eingeführt, modellieren MLN Unsicherheiten in *FOL*-WB mit Wahrscheinlichkeiten.

Ein MLN-Modell ist durch die endliche Tupel-Menge $L = \{(F_i, w_i): i = 1, \dots, n;\ n \in \mathbb{N}\}$ definiert, wobei F_i eine *FOL*-Formel und $w_i \in \mathbb{R}$ die Gewichtung dieser Formel ist. Je größer die Gewichtung einer Formel ist, desto wahrscheinlicher ist die Erfüllung dieser Formel. Formeln mit unendlichen Gewichtungen werden sichere oder allgemeingültige Formeln genannt. Das Markov-Netz $M_{L,C}$ wird durch das MLN-Modell L und die endliche Menge logischer Konstanten $C = \{C_j: j = 1, \dots, m;\ m \in \mathbb{N}\}$ wie folgt definiert [25]:

- Alle Variablen der Formel F_i von L werden durch Konstanten aus C ersetzt.
- $M_{L,C}$ enthält einen Knoten für jedes Grundprädikat von L. Die zu jedem Knoten assoziierte Zufallsvariable X_i ist binär mit dem Wertebereich $\{0,1\}$. $X_i = 0$ bedeutet, dass das Grundprädikat den Wahrheitswert *falsch* hat. $X_i = 1$ bedeutet, dass das Grundprädikat *wahr* ist.

- $M_{L,C}$ enthält ein Feature $f_k(x) \in \{0,1\}$ für jede Grundformel von L. Das Feature $f_k(x)$ nimmt den Wert 0 an, wenn die Grundformel *falsch* ist und 1 sonst. Die Gewichtung w_k des Features f_k ist die Gewichtung der entsprechenden Formel F_i.
- $M_{L,C}$ enthält eine Kante zwischen zwei Knoten, wenn die entsprechenden Grundprädikate in mindestens einer Grundformel vorkommen.

Ein MLN kann als ein Template betrachtet werden, womit Markov-Netze für gegebene logische Konstantenmengen aufgebaut werden. Markov-Netze, die mit unterschiedlichen Konstantenmengen aufgebaut wurden, werden Grund-Markov-Netze genannt. Die Struktur aller Grund-Markov-Netze des MLN-Modells L wird durch L gegeben [25]. Die Verbundwahrscheinlichkeit einer gegebenen Belegung der Grundatome von $M_{L,C}$ ist durch

$$P(X = x) = \frac{1}{Z} \prod_{i=1}^{n} \phi_i\left(x_{\{i\}}\right)^{\beta_i(x)} \qquad (11)$$

gegeben, wobei Z der Normierungsfaktor ist. $\beta_i(x) = \sum_{x' \in x_{\{i\}}} f_i(x')$ ist die Anzahl der Grundformeln von F_i mit dem Wahrheitswert *wahr* für die gegebene Belegung x. $x_{\{i\}}$ enthält die Wahrheitswerte aller Grundatome von F_i. $\phi_i\left(x_{\{i\}}\right) = e^{w_i}$ ist die Potentialfunktion. Ein Vergleich der Gleichung (11) mit der allgemeinen Verbundwahrscheinlichkeit von Markov-Netzen in (3) und (4) zeigt, dass jede Grundformel eines Grund-Markov-Netzes eine Clique ist. Alle Grundformeln der Formel F_i haben die gleiche Potentialfunktion ϕ_i, die die logarithmische Abbildung der Gewichtung w_i von F_i ist. Analog zu der Gleichung (6) ist die Gleichung (12) die logarithmische Darstellung der Verbundwahrscheinlichkeit aus (11) für Grund-Markov-Netze [25].

$$P(X = x) = \frac{1}{Z}\, exp\left(\sum_{i=1}^{n} w_i \beta_i(x)\right) \qquad (12)$$

Die Gewichtung w_i kann als die negative Energiefunktion interpretiert werden. Die Verbundwahrscheinlichkeit der Gleichung (12) erreicht ihr Maximum, wenn $w_i \to +\infty$ eine große positive Zahl und die Formel F_i für die Belegung x erfüllbar ist. Formeln, die in den Trainigsdaten öfter erfüllbar sind, bekommen deshalb positive Gewichtungen [25].

Ein Beispiel eines MLN-Modells $L = \{(F_1, w_1), (F_2, w_2)\}$ aus [25] ist in der Tabelle 2-1 exemplarisch dargestellt. Die Formel F_1 bedeutet, dass *Rauchen Krebs* verursachen kann. Da in der Praxis aber nicht alle Raucher Krebs haben, wird diese Formel mit der Gewichtung w_1 gewichtet. Die Formel F_2 besagt, dass zwei Freunde das gleiche Rauchverhalten haben. Entweder rauchen beide oder keiner raucht. Auch hier gibt es in der Praxis Freunde, wo nur einer Raucher ist. Diese Tatsache wird durch die Gewichtung w_2 modelliert. In der KNF besitzt F_2 zwei Klauseln. Jede Klausel wird als eine eigenständige Formel betrachtet und wird mit der Hälfte von w_2 gewichtet.

Tabelle 2-1: Beispiel eines MLN-Modells aus [25]

Formelname	Formel	Gewichtung
$\boldsymbol{F_1}$	$\forall x\ smokes(x) \Rightarrow cancer(x)$	$w_1 = 1{,}5$
$\boldsymbol{F_2}$	$\forall x\ \forall y\ friend(x,y) \Rightarrow \left(smokes(x) \Leftrightarrow smokes(x)\right)$	$w_2 = 2{,}2$

Die Abbildung 2-3 stellt das Grund-Markov-Netz von L aus der Tabelle 2-1 für die Konstantenmenge $C = \{A, B\}$ dar. Alle variablen des MLN-Modells L werden durch Kostanten aus C ersetzt, um die Grundprädikate zu generieren. Alle Grundprädikate sind gemäß der Definition von Grund-Markov-Netzen Knoten des Grund-Markov-Netzes. Kanten verbinden Knoten der Grundprädikate, die in einer Grundformel enthalten sind. Eine Grundformel von F_1 ist rot umkreist, während eine von F_2 blau umkreist ist. Die Anzahl der Knoten des Grund-Markov-Netzes steigt linear für unäre Prädikate und exponentiell für multivariate Prädikate mit der Anzahl der Konstanten auf. Je mehr ein Grund-

Markov-Netz Knoten und Kanten hat, desto höher ist die Rechenzeiten während der Trainings- und Inferenzphase. Die Komplexität eines Grund-Markov-Netzes ist deshalb durch die Anzahl der Knoten und Kanten dieses Netzes gegeben. Analog kann die Komplexität eines Markov-Logik-Netzes durch die Anzahl der Klausel dieses Netzes definiert werden.

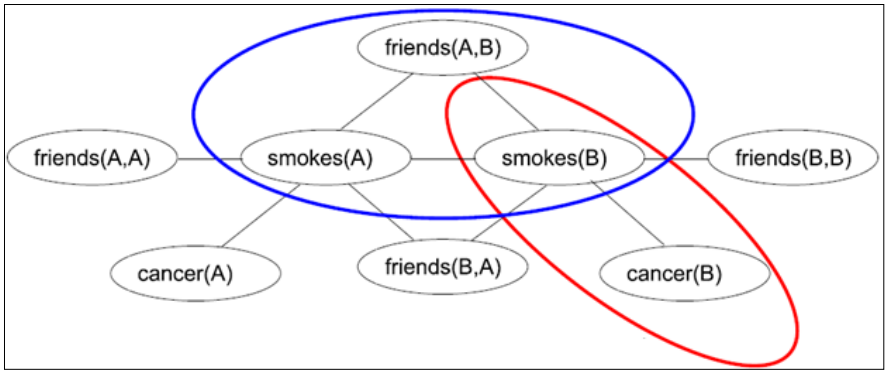


Abbildung 2-3: Beispiel eines Grund-Markov-Netzes für das MLN-Modell aus der Tabelle 2-1 und die Konstantenmenge $C = \{A, B\}$ (Originalbild aus [25])

Damit das Markov-Netz $M_{L,C}$ unabhängig von der Domäne eine Wahrscheinlichkeitsverteilung modelliert, müssen folgende Punkte erfüllt werden [25]:

- Eindeutige Namen: Konstanten mit unterschiedlichen Namen repräsentieren unterschiedliche Objekte.
- *Domain-closure*: Alle Objekte der Domäne werden durch Konstanten aus C und Funktionen aus L repräsentiert.
- Bekannte Funktionen: Alle Funktionen von L bilden alle Konstanten von C auf andere Konstanten von C ab.

2.3.2 Lernen der Parameter

In diesem Abschnitt liegt der Fokus auf dem Lernen der Gewichtung von Formeln. Das Lernen bzw. verfeinern der Struktur von MLN kann von Verfahren des induktiven logischen Programmierens realisiert werden [25, 35]. Diese Verfahren sind für diese Arbeit nicht relevant und werden deshalb nicht erläutert.

2.3.2.1 Generatives Lernen der Gewichtungen

Das generative Lernen der Gewichtungen hat als Ziel, die Gewichtungen der Verbundwahrscheinlichkeit aus der Gleichung (12) zu schätzen. Dafür wird für gegebene Trainingsdaten in Form des Vektors x die partielle Ableitung der *Log-Likelihood*-Funktion

$$\frac{\partial P_w(X = x)}{\partial w_i} = \beta_i(x) - \sum_{x' \in \mathbb{X}} P_w(X = x')\beta_i(x') \qquad (13)$$

der Verbundwahrscheinlichkeit aus (12) verwendet, um die *Log-Likelihood*-Funktion zu maximieren. $P_w(X = x')$ ist die Funktion aus (12) mit dem Gewichtungsvektor w. $\beta_i(x)$ zählt die Anzahl der Erfüllbarkeit von F_i gegeben x. Die Berechnung dieser Gleichung ist unlösbar, da $\beta_i(x)$ *P-vollständig* ist. Darüber hinaus ist die Berechnung von $P_w(X = x')$ mit der Inferenz des Modells verbunden, was rechenaufwendig ist. In der Praxis wird deshalb die *Pseudo-Log-Likelihood*-Funktion verwendet. Der Vorteil der *Pseudo-Log-Likelihood*-Funktion ist, dass das Modell nicht inferiert wird, was das Training beschleunigt. Die *Pseudo-Log-Likelihood*-Funktion kann mithilfe des *Memory-limited-Broyden-Fletcher-Goldfarb-Shanno*-(*L-BFGS*-)Algorithmus maximiert werden [25]. Dieser Algorithmus hat eine Inferenzkomplexität von $O(k * m * n)$, wobei k, m und n jeweils die Anzahl der Iterationen, die Anzahl der *BFGS*-Update pro Iteration und die Dimension des Inputvektors sind [36]. Ein Nachteil der *Pseudo-Log-Likelihood*-Funktion ist, dass die Inferenz dieser Funktion mit nicht benachbarten Knoten fehlerbehaftet sein kann [37].

2.3.2.2 Diskriminatives Lernen der Gewichtungen

In vielen Anwendungen sind die Evidenz- und Abfrage-Prädikate im Vorfeld schon bekannt. In solchen Fällen können anstatt der Verbundwahrscheinlichkeit aus (12) die Parameter der bedingten Wahrscheinlichkeit geschätzt werden.

$$P\big(Y = y \big| X = x, M_{L,C}\big) = \frac{1}{Z_x}\, exp\left(\sum_{j \in G_Y} w_j g_j(x, y)\right) \qquad (14)$$

X und Y sind jeweils die Evidenz- und Abfrage-Prädikate. G_Y ist die Menge der Grundklauseln von $M_{L,C}$, die ein Abfrage-Prädikat enthalten. $g_j(x, y) = 1$, wenn die j-Klausel von G_Y wahr ist, gegeben x und y. Ist die j-Klausel von G_Y *falsch* für gegebene x und y, dann ist $g_j(x, y) = 0$. Z_x ist der Normierungsfaktor [37]. Zum Schätzen der Parameter der bedingten Wahrscheinlichkeit aus der Gleichung (14) wird wie beim generativen Lernen die bedingte *Log-Likelihood*-Funktion maximiert. Allerding wird hier keine probabilistische Inferenz, sondern eine *MAP*-Inferenz verwendet. Der Vorteil ist, dass die *MAP*-Inferenz deutlich schneller als die probabilistische Inferenz ist [37]. Singla et. al [37] zeigten, dass MLN-Modelle, die diskriminativ trainiert wurden, deutlich bessere Ergebnisse als generativ trainierte MLN-Modelle lieferten. Für das diskriminative Lernen wurde in [37] der *Voted-perceptron*-Algorithmus in Kombination mit dem *MaxWalkSat-Solver* verwendet. Der in [38] verwendete *Conjugate-gradient*-Algorithmus lieferte bessere Ergebnisse als der *Voted-perceptron*-Algorithmus.

2.3.2.3 Interpretation der gelernten Gewichtungen

Hat eine Formel des MLN-Modells keine gemeinsame Variable mit allen anderen Formeln des MLN-Modells, dann ist die Gewichtung dieser Formel äquivalent zu dem Logarithmus der relativen Häufigkeit, dass diese Formel erfüllt wird über alle Trainingsdaten. In der Praxis teilen sich Formeln des MLN-Modells oft Variablen. In solchen Fällen können die gelernten Gewichtungen nicht mehr eins zu eins mit den relativen Häufigkeiten der Erfüllung der Formeln verglichen werden. Die Gewichtungen werden so gelernt, dass der Fehler zwischen den geschätzten und den tatsächlichen Wahrscheinlichkeiten aller Formeln, die sich Variablen teilen, minimiert wird [25].

2.3.3 Inferenz

Während der Inferenz muss als erstes das Markov-Netz $M_{L,C}$ aufgebaut werden. Die Komplexität für den Aufbau des Markov-Netzes $M_{L,C}$ ist $O(|p| * |C|^a)$, wobei $|p|$, $|C|$ und a jeweils die Anzahl der Prädikate, die Anzahl der Konstanten und die maximale Stelligkeit der Prädikate von L sind. In der Praxis wird $M_{L,C}$ um das minimale Netz $M \subseteq M_{L,C}$ vereinfacht, was für die Inferenz notwendig ist. Das Netz M enthält aufgrund der globalen Abhängigkeit in Markov-Netzen (siehe Gleichung (7)) die Knotenmenge V_1 von $M_{L,C}$ aus allen Grundprädikaten von F_1 sowie die Knotenmenge V_2 aus allen Grundprädikaten von F_2. Darüber hinaus enthält M die Knotenmenge V_3 von $M_{L,C}$, die nicht von V_1 separierbar gegeben V_2 ist ($V_3 \notin sep_{\mathcal{H}}(V_1; V_3|V_2)$). Alle Knoten von M werden mit Kanten verbunden, wenn diese Knoten auch in $M_{L,C}$ verbunden sind. Der Algorithmus zur Generierung von M ist in der Abbildung 2-4 exemplarisch dargestellt [25].

function ConstructNetwork(F_1, F_2, L, C)
  inputs: F_1, a set of ground atoms with unknown truth values (the "query")
    F_2, a set of ground atoms with known truth values (the "evidence")
    L, a Markov logic network
    C, a set of constants
  output: M, a ground Markov network
  calls: $MB(q)$, the Markov blanket of q in $M_{L,C}$
$G \leftarrow F_1$
while $F_1 \neq \emptyset$
  for all $q \in F_1$
    if $q \notin F_2$
      $F_1 \leftarrow F_1 \cup (MB(q) \setminus G)$
      $G \leftarrow G \cup MB(q)$
    $F_1 \leftarrow F_1 \setminus \{q\}$
return M, the ground Markov network composed of all nodes in G, all arcs between them in $M_{L,C}$, and the features and weights on the corresponding cliques

Abbildung 2-4: Algorithmus zur Vereinfachung des Markov-Netzes $M_{L,C}$ während der Inferenz (Bild aus [25])

Das aufgebaute Markov-Netz $M_{L,C}$ wird oft verwendet, um die Wahrscheinlichkeit, dass die Formel F_1 (Abfrage) erfüllt wird, gegeben F_2 (Evidenz), zu schätzen:

$$P(F_1|F_2, M_{L,C}) = \frac{\sum_{x \in X_{F_1} \cap X_{F_2}} P(X = x|M_{L,C})}{\sum_{x \in X_{F_2}} P(X = x|M_{L,C})} \quad (15)$$

X_{F_i} ist die Belegungsmenge, für die F_i *wahr* ist. $P(X = x|M_{L,C})$ ist die Verbundwahrscheinlichkeit, wie in der Gleichung (12) definiert. Eine exakte Inferenz der Gleichung ist aufgrund der Komplexität der logischen (*NP-vollständig*) und probabilistischen Inferenz (*P-vollständig*) auch für kleine MLN sehr komplex. In dem Fall, wo F_1 und F_2 Gundprädikate sind, ist die Gleichung (15) jedoch einfacher zu lösen [25].

Zur Vereinfachung der Inferenz des Markov-Netzes werden *Markov-Chain-Monte-Carlo-(MCMC-)* Verfahren verwendet. Ein Verfahren dieser Klasse ist das *Gibbs-Sampling*-Verfahren. Dieses Verfahren sampelt in mehreren Schritten den Wert der Zufallsvariable X_l (Wahrheitswert eines Grundprädikats), gegeben sein Markov-*Blanket* $N_M(X_l)$, anhand der Gleichung

$$P(X_l|N_M(X_l)) = \frac{exp(\sum_{f_i \in F_l} w_i f_i(X_l, N_M(X_l)))}{\sum_{x \in \{0,1\}} exp(\sum_{f_i \in F_l} w_i f_i(X_l = x, N_M(X_l)))} \quad (16)$$

F_l ist die Menge der Formeln, die das Grundprädikat von X_l enthalten. Die Funktion $f_i(X_l, N_M(X_l))$ gibt den Wahrheitswert der Formel F_i für die Belegung $X_l, N_M(X_l)$. Der *Gibbs-Sampler* wird p-mal für alle Zufallsvariablen von M durchgeführt und gibt die Menge $S = \{X_M^i, k = 1, \dots, p;\ p \in \mathbb{N}\}$ zurück, wobei X_M^i die Menge der Wahrheitswerte aller Zufallsvariablen von M ist. Die bedingte Wahrscheinlichkeit aus der Gleichung (16) ist dann die Ratio zwischen der Anzahl an Samples X_M^i, für die F_1 *wahr* ist, und der Anzahl der Elemente von S. Zur Initialisierung des *Gibbs-Sampling*-Verfahrens wird der *MaxWalkSat*-Algorithmus verwendet [25].

2.4 Künstliche neuronale Netze

2.4.1 Definition

Künstliche neuronale Netze sind mathematische Modelle, die die Struktur und die Funktion der Neuronen des Nervensystems von Menschen und Tieren in einer stark vereinfachten Form nachbilden. Biologische Neuronen verarbeiten Informationen, die an ihren Eingängen, sogenannten Dendriten, in Form von elektrischen Signalen liegen und generieren neue Informationen in Form von elektrischen Signalen. Die von den Neuronen generierten Informationen werden an andere Neuronen über das

Axon zur weiteren Verarbeitung übergeben. Der prototypische Aufbau von biologischen Neuronen ist in der Abbildung 2-5 dargestellt [39].

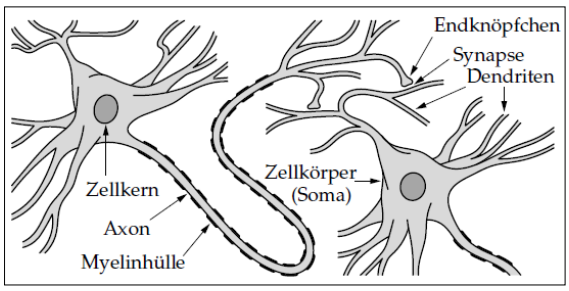


Abbildung 2-5: Prototypischer Aufbau von biologischen Neuronen (Bild aus [39])

Künstliche Neuronen verwenden eine lineare Funktion und eine Aktivierungsfunktion, um die Funktion der biologischen Neuronen abzubilden (siehe Gleichung (17)). Die lineare Funktion ist eine gewichtete Summe aller Eingänge x_i des Neurons mit den Gewichten w_i und dem Bias b. Die Aktivierungsfunktion bildet den Wert der linearen Funktion in einem gegebenen Wertebereich ab. Einige Beispiele von Aktivierungsfunktionen sind die Schwellenwertfunktion, die rektifizierten Lineareinheiten (engl. *ReLU*: *Rectified Linear Unit*) und die Sigmoid-Funktion [39].

$$\hat{y}(x) = f\left(\sum_{i=1}^{n} w_i\, x_i + b\right) \qquad (17)$$

Geometrisch betrachtet, modelliert ein künstliches Neuron eine Trennhyperebene. Ein Neuron kann deshalb nur zwei Klassen voneinander trennen. Ein neuronales Netz mit mehreren Neuronen kann komplexere Hyperebenen modellieren und somit mehr als zwei Klassen voneinander trennen.

2.4.2 Struktur

Die Struktur eines neuronalen Netzes kann beliebig komplex sein. In der Praxis werden jedoch Neuronen in Schichten angeordnet. Sind Neuronen einer Schicht nur mit Neuronen der nachfolgenden Schichten verbunden, spricht man von vorwärtsgerichteten neuronalen Netzen (siehe Abbildung 2-6). Sind Zyklen in neuronalen Netzen zu finden, weil Neuronen mit sich selbst oder mit Neuronen vorangegangener Schichten verbunden sind, werden diese Netze rückgekoppelte oder rekurrente neuronale Netze (RNN) genannt. Die erste Schicht des neuronalen Netzes wird Eingabeschicht genannt. Diese Schicht wird direkt mit den Eingaben verbunden. Die Ausgabeschicht ist die letzte Schicht des Netzes. Alle Schichten dazwischen werden verdeckte Schichten genannt. Ein Netz mit mehr als einer verdeckten Schicht wird tiefes neuronales Netz genannt. In der Praxis können TNN mehrere tausend Schichten haben (siehe bspw. MOE-Modell [40] und ResNet-1001 [41]).

Das rezeptive Feld eines Neurons ist die Menge aller Neuronen der vorangegangenen Schicht, die mit diesem Neuron verbunden sind [42].

Eine Schicht heißt vollständig vernetzt, wenn jedes Neuron dieser Schicht mit allen Neuronen der vorangegangenen Schicht verbunden ist (siehe zweite und dritte Schicht der Abbildung 2-6). Das rezeptive Feld eines Neurons einer vollständig vernetzten Schicht enthält also alle Neuronen der vorangegangenen Schicht. Da jedes Neuron ein Parameter pro Verbindung mit einem anderen Neuron hat, steigt die Anzahl der Parameter der vollständig vernetzten Schicht exponentiell mit der Anzahl der Neuronen der vorangegangenen Schicht an. Damit werden die Komplexität und der Rechenaufwand des Netzes erhöht. Der Vorteil einer vollständig vernetzten Schicht ist die Fähigkeit, globale Informationen der vorangegangenen Schicht zu erfassen.

Die Abbildung 2-6 stellt die Struktur eines einfachen vorwärtsgerichteten neuronalen Netzes dar. Dieses Netz hat zwei verdeckte Schichten und ist somit ein tiefes Netz. Die verdeckten Schichten und die Ausgabeschicht sind vollständig vernetzte Schichten.

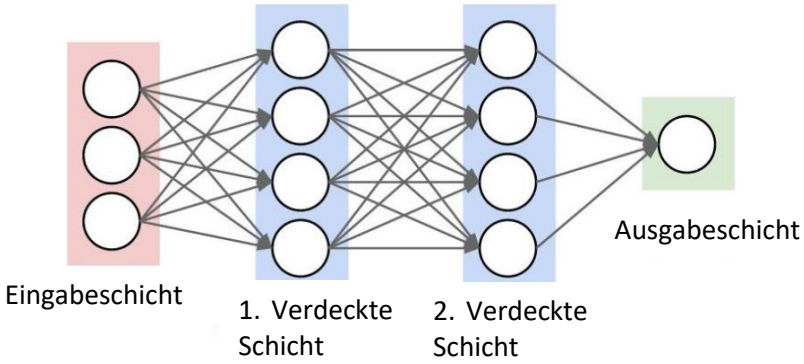


Abbildung 2-6: Struktur eines vollständig vernetzten neuronalen Netzes. Das Netz hat drei Schichten, wobei zwei Schichten verdeckt sind (Originalbild aus [43])

Eine Schicht heißt Faltungsschicht mit dem Filterkern $H \in \mathbb{R}^{n \times m \times l}$, wenn jedes Neuron dieser Schicht mit allen Neuronen der vorangegangenen Schicht, die sich in dem Fenster zentriert um den Filterkern befinden, verbunden ist. Ein Neuron der Faltungsschicht wendet die Faltungsoperation mit einem trainierbaren Filterkern auf die Neuronen der vorangegangenen Schicht an. Das rezeptive Feld eines Neurons einer Faltungsschicht entspricht der Dimension des Filterkerns dieser Schicht. In der Praxis werden oft *ReLU*-Einheiten als Aktivierungsfunktion von Faltungsschichten verwendet. Eine Faltungsschicht hat den Vorteil, dass sie im Vergleich zu einer vollständig vernetzten Schicht wenige Parameter hat. Der Nachteil ist, dass ein Neuron einer Faltungsschicht aufgrund des eingeschränkten rezeptiven Felds lediglich lokale Informationen der vorangegangenen Schicht erfassen kann. Faltungsschichten spielen bei der Verarbeitung von strukturierten Daten wie etwa Kamerabilder eine große Rolle, da die Mehrheit der Informationen in strukturierten Daten wie Bildern lokal ist.

Ein Netz mit Faltungsschichten wird gefaltetes neuronales Netz (*CNN*: *Convolutional Neural Networ*k) genannt. Jede Schicht eines *CNN*-Modells berechnet eine Aktivierungskarte mit der Aktivierungskarte der vorangegangenen Schicht als Input. Die Aktivierungskarte ist ein 3-D-Volumen (w, h, d), wobei w, h und d jeweils die Breite, die Höhe und Tiefe der Aktivierungskarte sind.

In einer *Pooling*-Schicht ist jedes Neuron wie bei einer Faltungsschicht mit allen Neuronen in einem gegebenen Fenster der vorangegangenen Schicht verbunden. Allerding wendet ein Neuron der *Pooling*-Schicht keine Faltungsoperation, sondern andere Operationen wie das Maximum, den Mittelwert, den Median etc. auf die Neuronen der vorangegangenen Schicht an. *Pooling*-Schichten werden verwendet, um die Breite und Höhe der Aktivierungskarte der vorangegangenen Schicht zu reduzieren. Die Tiefe dieser Aktivierungskarte bleibt unverändert.

Die Struktur von tiefen *CNN*-Modellen besteht oft aus einer Reihe von Faltungsschichten, *ReLU*-Einheiten und *Pooling*-Schichten. Solche Netze werden auch Basisnetze genannt und sind für das Lernen und Generieren von Merkmalen zuständig. Die ersten Schichten enthalten aufgrund der kleinen rezeptiven Felder lokale und detailreiche Informationen wie Kanten und Ecken. Tiefere Schichten haben relativ zu den Inputschichten größere rezeptive Felder, da das Inputbild herunterskaliert wird. Diese Schichten können deshalb globale Informationen wie Objekteteile oder ganze Objekte im Inputbild erfassen. Dabei verlieren sie an detailreichen Informationen. Basisnetze werden, bezogen auf den Anwendungsfall, an der Ausgangsschicht erweitert. Dafür kommen vollständig vernetzte Netze oder gefaltete Netze infrage.

Die Struktur von TNN kann manuell generiert werden. Ansätze zum Lernen der Struktur von TNN werden intensiv erforscht. Jedoch liegen solche Ansätze außerhalb des Fokus dieser Arbeit und werden hier nicht vorgestellt.

2.4.3 Lernen der Parameter

Zum Lernen der Parameter w_i und b_j eines neuronalen Netzes wird oft der Fehler-Rückübertragungsalgorithmus (*back propagation algorithm*) verwendet. Dieser Algorithmus mini-

miert die Fehlerfunktion $L(\hat{y}, y)$ in der Gleichung (18) mithilfe des Gradientenabstiegsverfahrens, wobei d eine Distanzfunktion zwischen dem Zielwert y und dem geschätzten Wert $\hat{y}$ ist.

$$L(\hat{y}, y) = d\left(f\left(\sum_{i=1}^{n} w_i\, x_i + b\right), y\right) \tag{18}$$

Dieser Algorithmus setzt voraus, dass die die Fehlerfunktion $L(\hat{y}, y)$ stetig und differenzierbar ist. Es werden deshalb stetige und differenzierbare Aktivierungsfunktionen wie die *ReLU*-Einheit und die Sigmoid-Funktion in neuronalen Netzen verwendet. Da der Zielwert y für das Lernen der Parameter vorhanden sein muss, spricht man von *überwachtem Lernen*. Die Generierung des Zielwerts y ist für einige Anwendungen sehr aufwendig und stellt einen Nachteil des überwachten Lernens dar.

2.4.4 Analyse der Komplexität

Sei n_p die Anzahl der Parameter eines neuronalen Netzes. Die Komplexität der Inferenz dieses neuronalen Netzes hängt linear von n_p ab ($\mathcal{O}(n_p)$). Bei CNN, wo die Faltungsfilter ort-invariant sind, hängt die Komplexität der Inferenz des Netzes linear von n_p und von der Dimension $n \times m$ der Inputmatrix ab ($\mathcal{O}(n * m * n_p)$).

Die Komplexität beim Lernen der Parameter eines neuronalen Netzes mit dem Fehler-Rückübertragungsalgorithmus hängt linear von n_p und von der Anzahl der Lerniterationen n_{it} ab ($\mathcal{O}(n_{it} * n_p)$). Der Einfluss der Dimension $n \times m$ der Inputmatrix bei CNN mit ort-invarianten Faltungsfiltern auf die Komplexität beim Lernen der Parameter ist linear ($\mathcal{O}(n * m * n_{it} * n_p)$).

Aufgrund der Abhängigkeit der Komplexität von neuronalen Netzen von deren Parameteranzahl, wird die Komplexität dieser Netze in der Praxis oft durch die Parameteranzahl angegeben.

2.4.5 Anwendung von neuronalen Netzen zur Regression und Klassifikation

Neuronale Netze werden oft für Regressions- und Klassifikationsaufgaben verwendet.

Für den Regressionsfall wird das neuronale Netz verwendet, um reelle Werte zu schätzen. Diese Werte könnten bspw. die Position, die Breite und die Höhe eines Objekts im Kamerabild sein. In der Trainingsphase wird die Fehlerfunktion aus (18) minimiert, wobei y und die Inputs x_i bekannt sind. In der Inferenzphase wird der Wert von $\hat{y}$ aus der Gleichung (17) geschätzt, gegeben die Inputs x_i und die gelernten Parameter des neuronalen Netzes.

Für die Klassifikation wird oft der *Softmax*-Klassifikator verwendet. Dieser Klassifikator verwendet die *Softmax*-Funktion, um die klassenbedingte Wahrscheinlichkeit zu modellieren, wobei $x \in \mathbb{R}^{m \times n}$ der Input ist.

$$Softmax(K = k|x) = \frac{e^{\hat{y}_k(x)}}{\sum_{l=1}^{m} e^{\hat{y}_l(x)}} \tag{19}$$

$\hat{y}_k(x)$ ist der Wert der Funktion aus (17) für die Klasse k aus der Menge K aller Zielklassen. In der Trainingsphase wird die Kreuzentropie zwischen der Wahrscheinlichkeit aus (19) und der *Ground-Truth*-Wahrscheinlichkeit minimiert. In der Praxis wird die *Ground-Truth*-Wahrscheinlichkeit als Vektor der Dimension K eingegeben. Dieser Vektor hat den Wert 1 in der Position k_{GT} der *Ground-Truth*-Klasse und 0 sonst. Damit ergibt sich die folgende Kreuzentropie-Fehlerfunktion:

$$L(\hat{y}, y) = -\log\big(Softmax(K = k_{GT}|x)\big) \tag{20}$$

Die Kreuzentropie-Fehlerfunktion wird in der Trainingsphase minimiert, wenn die geschätzte *Ground-Truth*-Wahrscheinlichkeit $Softmax(K = k_{GT}|x)$ gegen eins geht. Dies führt dazu, dass das trainierte Netz dazu tendiert, *Softmax*-Werte, die gegen eins gehen, auch für falsche Klassifikationen zu inferieren. *Softmax*-Werte können deshalb nicht als Unsicherheiten des Netzes in der Inferenzphase verwendet werden.

2.5 Zusammenfassung

In diesem Kapitel wurden die Grundlagen der Methoden, die als Fundament für die Ansätze dieser Arbeit dienen, kompakt vorgestellt.

Im ersten Abschnitt wurden die Methoden der Wissensrepräsentation dargestellt. Dabei wurde vor allem auf die Syntax und die Semantik von *FOL* und von Ontologien eingegangen. Darüber hinaus wurden Inferenzalgorithmen zur Gewinnung von neuem Wissen erläutert. Die hier vorgestellten Formalismen stellen eine solide Grundlage zur symbolischen Wissensmanipulation und sind maschinell interpretierbar.

Der zweite Abschnitt stellte PGM-Modelle dar. Diese Modelle basieren auf der Wahrscheinlichkeits- und Graphentheorie. Es wurde deshalb die Struktur dieser Modelle als Graphen vorgestellt. Ferner wurde eine Klasse von PGM-Modellen namens Markov-Netze vorgestellt. Danach wurden Verfahren zum Lernen der Parameter der Wahrscheinlichkeitsverteilungen von PGM-Modellen sowie zur probabilistischen Inferenz erläutert. PGM-Modelle sind passende *Frameworks* zur Modellierung von Wissen mit Unsicherheiten.

MLN-Modelle, die eine Kombination von Markov-Netzen und *FOL*-Formeln sind, wurden im Detail im vierten Abschnitt vorgestellt. Methoden zum Lernen der Parameter von MLN-Modellen sowie zur Inferenz von MLN-Modellen wurden erläutert. MLN-Modelle ermöglichen, das symbolisch modellierte Wissen mit Unsicherheiten zu ergänzen.

Der vierte Abschnitt beschäftigte sich mit künstlichen neuronalen Netzen. Künstliche neuronale Netze sind mathematische Modelle der vereinfachten Funktion der Neuronen des Nervensystems von Menschen und Tieren. Hier wurden vor allem TNN-Modelle erläutert. Diese Modelle schalten mehrere Neuronenschichten hintereinander, um komplexere Funktionen zu modellieren. Algorithmen zum Lernen der Parameter von TNN-Modellen wurden vorgestellt. Außerdem wurden konkrete Anwendung von TNN-Modellen zur Klassifikation und Regression illustriert. TNN-Modelle sind in der Lage, subsymbolische Informationen zu verarbeiten.

In den nächsten Kapiteln werden die in diesem Kapitel vorgestellten Methoden verwendet, um den in dieser Arbeit vorgeschlagenen Lösungsansatz zu entwerfen, zu implementieren und zu testen.

3 Stand der Technik

Um eine kollisionsfreie Trajektorie zu planen, müssen automatisierte Fahrsysteme die Szene und die Situation anhand von Sensoren, die am Fahrzeug montiert sind, erfassen. In diesem Kapitel wird der Stand der Technik für die Szenen- und Situationserfassung im Kontext des automatisierten Fahrens vorgestellt. Viele Ansätze aus der Literatur basieren auf den Methoden, die in Kapitel 2 präsentiert wurden. Im ersten Abschnitt werden zunächst die Begriffe Szene und Situation definiert. Im zweiten Abschnitt werden die Sensoren, die Inputdaten für die Erfassung von Szenen und Situationen liefern, beschrieben. Methoden und Architekturen zur Fusion dieser Sensordaten werden im dritten Abschnitt erläutert. Zwei wichtige Aufgaben der Szenenerfassung, nämlich die bildbasierte Objekterkennung und die semantische Segmentierung, werden in den Abschnitten vier und fünf im Detail vorgestellt. Der sechste Abschnitt wird sich mit den Ansätzen der Situationsmodellierung und –Interpretation beschäftigen. Während die Ansätze der Abschnitte vier, fünf und sechs die Szene und die Situation getrennt betrachten, werden im Abschnitt sieben Ansätze zur holistischen Betrachtung von Szenen und Situationen vorgestellt. Ansätze, die Steuerungsparameter des Fahrzeugs direkt aus den Daten lernen, ohne die Szene und die Situation explizit zu erfassen, werden im achten Abschnitt diskutiert. Der letzte Abschnitt fasst dieses Kapitel zusammen.

3.1 Definition der Begriffe Szene und Situation

In der Literatur gibt es viele Definitionen der Begriffe Szene und Situation. Die Autoren aus [44] haben mehrere Definitionen aus der Literatur gesammelt, um daraus konsistente Definitionen zu generieren. Von Ulbrich et al. [44] wird eine Szene wie folgt definiert:

„*Eine Szene beschreibt eine Momentaufnahme des Umfelds, welche die Szenerie, dynamische Elemente, die Selbstrepräsentation aller Akteure und Beobachter wie auch die Verknüpfung dieser Entitäten umfasst. Einzig eine Szenenrepräsentation in einer simulierten Welt kann allumfassend sein (objektive Szene, Ground Truth). In der realen Welt ist sie immer unvollständig, fehlerbehaftet, unsicherheitsbehaftet und aus der Perspektive eines oder mehrerer Beobachter (subjektive Szene).*“

Akteure sind selbsthandelnde Elemente. Beobachter sind Elemente, die eine Szene wahrnehmen können. Diese können auch Akteure sein [44]. Da eine Szene immer eine subjektive Sicht eines oder mehrerer Beobachter ist, ist es sinnvoll, die Unsicherheiten der erfassten Szene zu modellieren und zu schätzen. Funktionen, die zur Entscheidungsfindung wahrgenommene Szenen als Input haben, sollen die Unsicherheiten der Szenen in ihrer Entscheidungsfindung betrachten.

Die Bestandteile einer Szene aus [44] sind in der Abbildung 3-1 dargestellt.

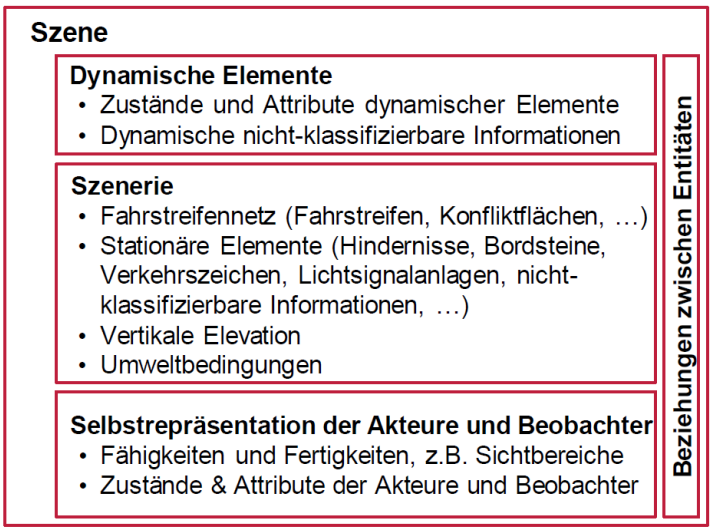


Abbildung 3-1: Beispiel einer Szenenrepräsentation nach [44] (Bild aus [44])

Die Situation wird in [44] wie folgt definiert:

„*Eine Situation beschreibt die Gesamtheit der Umstände, die für die Auswahl geeigneter Verhaltensmuster zu einem bestimmten Zeitpunkt zu berücksichtigen sind. Sie umfasst alle relevanten Bedingungen, Möglichkeiten und Determinanten von Handlungen. Eine Situation wird aus der Szene durch einen Prozess der Informationsauswahl und -augmentieren abgeleitet, basierend auf transienten (z. B. missionspezifischen) wie auch permanenten Zielen und Werten. Folglich ist eine Situation immer subjektiv, indem sie die Sicht eines Elements repräsentiert.*“

Folgende Aspekte dieser Definition sind für diese Arbeit relevant:

- Die Situation wird von der Szene abgeleitet.
- Die Situation enthält alle Elemente der Szene, die für die Ziele und Werte des Ego-Fahrzeugs relevant sind. Die Ziele und Werte werden durch transiente oder permanente Faktoren wie die aktuelle Mission des Ego-Fahrzeugs, die Fahranweisungen eines Operators, die Verhaltenspräferenzen des Fahrzeugnutzers, Verkehrsregeln etc. beeinflusst [44].
- Elemente der Szene, die eine potenzielle Interaktion mit dem Ego-Fahrzeug haben können (z. B. Kreuzung der Wege mit dem Ego-Fahrzeug), sind für das Ego-Fahrzeug relevant.
- Die Situationsanalyse und –Interpretation bewertet bestimmte Aspekte der Situation (auch Situationsaspekte genannte):
    - die Schätzung der Relevanz von Szenenelementen,
    - die Schätzung der Konsistenz der Szenenelemente,
    - die kontextkonsistente erneute Klassifikation der Szenenelemente,
    - die Schätzung des kontextabhängigen Verhaltens von Verkehrsteilnehmern und
    - die Schätzung der Kritikalität der Situation.
- Die Situation enthält aufgrund ihrer Subjektivität Unsicherheiten. Diese Unsicherheiten sollen in der Situation modelliert und geschätzt werden.
- Die Situation dient als Input für die Planung der Trajektorie des Ego-Fahrzeugs.

Die Abbildung 3-2 stellt die Situationsrepräsentation aus [44] dar. Die Situationserfassung beschäftigt sich mit der Erfassung aller Elemente und Aspekte der Situation basierend auf den Inputs der Sensoren und der Szene. Aus den o. g. Definitionen ergeben sich drei Abstraktionsschichten: die Sensorschicht, die Szeneschicht und die Situationsschicht. Diese Schichten werden im nächsten Abschnitt verwendet, um den Stand der Technik zu beschreiben.

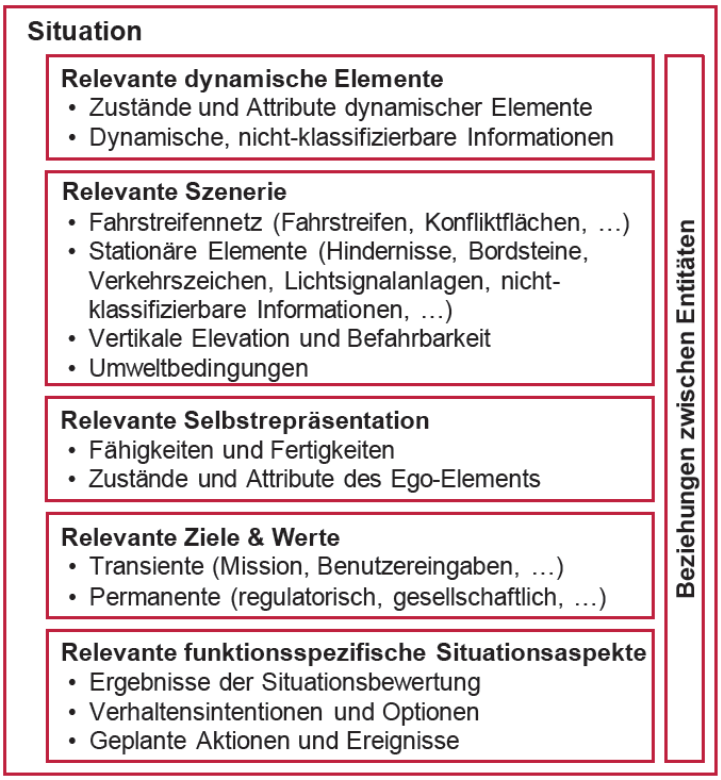


Abbildung 3-2: Beispiel einer Situationsrepräsentation nach [44] (Bild aus [44])

3.2 Sensoren zur Situationserfassung

Zur Erfassung der Situation werden unterschiedlichen Sensoren verwendet, um die Schwäche einzelner Sensoren zu kompensieren. Im Kontext des automatisierten Fahrens werden folgende Sensoren oft eingesetzt:

- Kameras: Kameras sind passive optische Sensoren, die vor allem viele semantische Informationen erfassen können. Daten aus den Kameras sind Farben- oder Grauwertbilder. Diese Daten werden oft als Inputs von TNN zur Objekterkennung und semantischen Segmentierung verwendet. Kameras haben auch den Vorteil, dass sie bei der Beschaffung kostengünstig sind. Der Hauptnachteil ist die Empfindlichkeit gegenüber der Beleuchtung und Witterung sowie die fehlenden Tiefeninformationen [45, 46]. Stereokameras ermöglichen, neben Bildern auch Tiefeninformationen zu gewinnen. Da die Tiefenschätzung auf Bildern basiert, haben Stereokameras auch die Vor- und Nachteile der Monokameras. Darüber hinaus hängt die Qualität der Tiefeninformationen von der Bildauflösung sowie von der Entfernung der Punkte zu der Kamera ab. Während Punkte, die näher an der Kamera liegen, gute Tiefenwerte bekommen, sinkt die Genauigkeit der Tiefe exponentiell mit der Entfernung der Punkte zu der Kamera ab.
- *Radio Detection and Ranging* (Radar): Radarsensoren sind aktive Sensoren, die elektromagnetische Wellen aussenden und die Echos der gesendeten Wellen auswerten. Aus der Laufzeit der gesendeten Welle und dem Frequenzunterschied zum Echo wird der Abstand und die Geschwindigkeit des Objekts, das das Echo verursacht hat, geschätzt [45]. Radarsensoren haben den Vorteil, dass sie in extremen Witterungen zuverlässig arbeiten können. Darüber hinaus sind sie preisgünstig. Sie werden deshalb oft für die Objekterkennung und Verfolgung im *Automotive*-Umfeld verwendet. Ein Nachteil dieser Sensoren ist die geringe Auflösung [46].
- *Light Detection and Ranging* (*Lidar*): *Lidar*-Sensoren sind optische Sensoren, die zur Messung von Entfernungen Lichtstrahlen anhand von Dioden-Arrays senden und die Reflexionen der gesendeten Lichtstrahlen auswerten. Die Entfernung wird durch die Laufzeit der gesendeten Lichtstrahlen und die Lichtgeschwindigkeit berechnet. *Lidar*-Sensoren haben den Vorteil, dass sie Entfernungen mit sehr hoher Genauigkeit messen können [45]. Der Hauptnachteil

ist, dass die gesendeten Lichtstrahlen bei dunklen und durchsichtigen Oberflächen sehr wenig reflektiert werden.

- Laserscanner-Sensoren verwenden das Prinzip der *Lidar*-Sensoren kombiniert mit einem rotierenden Spiegel. Solche Sensoren messen Entfernungen mit einer hohen Genauigkeit und Auflösung. Der Hauptnachteil dieser Sensoren ist der hohe Anschaffungspreis sowie der bewegliche Spiegel. Ansätze zur Lösung dieser Nachteile werden intensiv erforscht [45, 46].
- Ultraschallsensoren verwenden akustische Wellen, um die Entfernung im Nahbereich zu schätzen. Die Entfernung wird Anhand der Laufzeit des gesendeten akustischen Signals und der Schallgeschwindigkeit berechnet. Diese Sensoren sind aufgrund der präzisen Messung im Nahbereich sowie des geringeren Anschaffungspreises und der Robustheit gegenüber Witterungen oft zur Überwachung des Nahbereichs in Serienfahrzeugen eingesetzt [45, 46].

In der Praxis werden Sensoren unterschiedlicher Typen so ausgerichtet, dass sie möglichst viel Überlappung haben. Darüber hinaus decken die Sensoren das Gesamtumfeld des Ego-Fahrzeugs ab. Damit werden die Redundanz und die Robustheit des Systems sichergestellt. Die Auswahl der Sensortypen und die Anzahl der Sensoren hängen von o. g. Eigenschaften, Vor- und Nachteilen der Sensoren ab.

Eine weitere Informationsquelle für selbstfahrende Fahrzeuge stellen hochgenaue digitale Karten dar. Diese Karten enthalten wichtige Information über Infrastrukturelemente (Straßen, Fahrspuren, Verkehrszeichen etc.) und sind sehr hilfreich in komplexen Situationen, in denen die Sensoren des Ego-Fahrzeugs nicht alle relevanten Informationen erfassen können. Der Hauptnachteil dieser Karten ist der Aufwand bei der Generierung und Wartung.

Weitere Sensoren wie die *Car-2-X*-Kommunikation und mobile Geräte (z. B. Smartphones) werden immer relevanter. Diese liegen jedoch außerhalb des Rahmens dieser Arbeit und werden nicht erläutert.

3.3 Architekturen der Sensordatenfusion

Da mehrere Sensoren für die Umfelderfassung verwendet werden, wurden mehrere Fusionsarchitekturen vorgeschlagen. Ein weitverbreitetes Modell ist das *Joint-Directors-of–Laboratories*-(*JDL*-)Fusionsmodell. Dieses Modell und seine Anpassungen werden in den nächsten Abschnitten beschrieben.

3.3.1 *JDL*-Fusionsmodell

Das *JDL*-Modell ist eine funktionale Architektur zur Sensordatenfusion mit unterschiedlichen Abstraktionsebenen. Dieses Modell wurde ursprünglich für militärische Anwendungen entwickelt. Die Abbildung 3-3 stellt dieses Modell dar

Die erste Hauptanpassung des ursprüngliches *JDL*-Datenfusionsmodells wurde von Steinberg et al. [47] vorgeschlagen. Dieses Modell hat folgende funktionale Abstraktionsebenen [47, 48]:

- Level 0 − *Sub-Object Data Assessment*: Diese Ebene ist für die Sensordatengewinnung und -vorverarbeitung zuständig. Sensordaten können Kamerabilder, Laserscanner-Punktwolken oder Radar-Reflexionen etc. sein.
- In Level 1 − *Object Assessment* − werden Entitäten erkannt und prädiziert. Diese können sowohl andere Verkehrsteilnehmer als auch Spurmarkierungen, Verkehrszeichen, Verkehrsampeln oder das Ego-Fahrzeug etc. sein.
- Das Level 2 − *Situation Assessment* − beschäftigt sich mit der Schätzung und Prädiktion von Relationen zwischen Entitäten aus dem Level 1. Die Relationen können bspw. die Belegung einer Fahrspur von einem Fahrzeug oder die Absicht eines Fahrzeugs sein.
- Im Level 3 − *Impact Assessment* − wird der Einfluss von Verhalten und Absichten der geschätzten Entitäten auf die Situation geschätzt. Hier wird die Kritikalität der Situation bewertet.

- Das Level 4 – *Process Refinement* – ist ein Teil der Ressourcenverwaltung. Hier werden Daten verwaltet und bereitgestellt. Diese Ebene ist nicht Teil des Kerns des Fusionsmodells und wird deshalb selten adressiert.

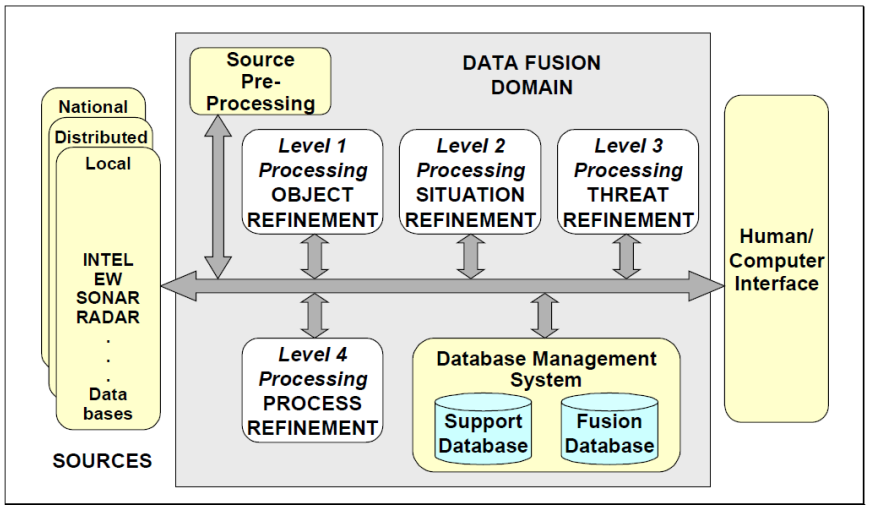


Abbildung 3-3: Ursprüngliches JDL-Datenfusionsmodell (Bild aus [47])

Der Vorteil des *JDL*-Modells ist die Modularität der Architektur. Darüber hinaus stellt dieses Modell eine holistische Betrachtung der Abstraktionsebenen dar, da die Kommunikation zwischen allen Ebenen bidirektional ist. Die Abhängigkeiten zwischen den Abstraktionsebenen können verwendet werden, um das Prozessieren einzelner Ebenen zu verbessern. So können z. B. Informationen der Situationsebene (Ebene 2) verwendet werden, um die Objekte zu erkennen (Ebene 1). Allerdings ist die holistische Betrachtung mit der Erhöhung der Komplexität des Modells verbunden [48].

3.3.2 *ProFusion2*-(*PF2*-)Modell

Zur Anwendung im *Automotive*-Bereich, wurde das *JDL*-Modell aus [47] in [48] angepasst und *PF2*-Modell genannt (siehe Abbildung 3-4). Das *PF2*-Modell unterscheidet sich von dem angepassten *JDL*-Modell durch folgende Punkte [48]:

1. Die Zusammenfassung von Abstraktionsebenen in Schichten: Die Wahrnehmungsschicht (*Perception Layer*) fasst die Ebenen 0 und 1 zusammen. Die Ebenen 2 und 3 werden in die Entscheidungsanwendungsschicht (*Decision Application Layer*) integriert. Die Ebene 5, die für die Mensch-Maschine-Schnittstelle zuständig ist, wird Aktions-/*HMI*-Schicht (*Action/HMI Layer*) genannt.
2. Die Hierarchie zwischen den Schichten: Die Kommunikation zwischen den Schichten ist unidirektional von links nach rechts. Die Wahrnehmungsschicht wird als erstes berechnet, gefolgt von der Entscheidungsanwendungsschicht. Die Aktions-/*HMI*-Schicht kommt als letztes.
3. Das Level 4 wird nicht adressiert, da dieses nicht zum Kern der Sensorfusion gehört.

Der Vorteil des *PF2*-Modells ist die Verringerung der Komplexität der Kommunikation zwischen den Schichten im Vergleich zum *JDL*-Modell, da die Kommunikation zwischen den Schichten unidirektional ist. Dies kann gleichzeitig ein Nachteil sein, da Informationen aus den höheren Schichten in den niedrigen Schichten nicht verfügbar sind. Um diesen Nachteil zu kompensieren, werden bidirektionale Kommunikationen zwischen den Ebenen einer Schicht erlaubt. Das *PF2*-Modell kann deshalb als ein Hybridmodell betrachtet werden [48].

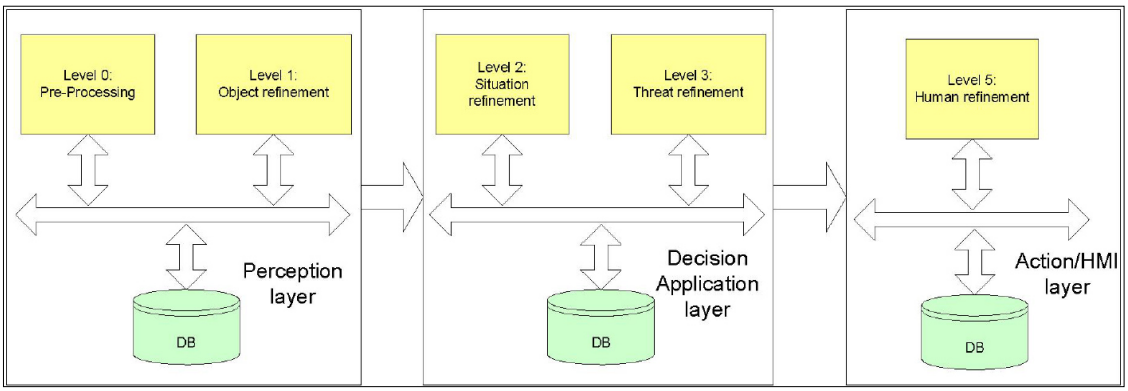


Abbildung 3-4: PF2-Sensordatenfusionsmodell (Bild aus [48])

Weitere Fusionsmodelle, die in der Literatur vorgeschlagen wurden, werden hier nicht diskutiert, da sie meistens auf dem *JDL*- oder *PF2*-Modell basieren.

3.4 Bildbasierte Objekterkennung

Die Objekterkennung ist wichtig bei der Szenenerfassung, da Objekte Hindernisse sind, die in einigen Fällen selbst handeln können. Objekte als Entitäten zu erfassen und ihre Dynamik zu prädizieren, ermöglicht die Entwicklung der Situation zu betrachten und Risiken frühzeitig zu erkennen. Die Objekterkennung besteht aus zwei Schritten: die Detektion und die Klassifikation. Diese Schritte werden in den nächsten Abschnitten erläutert. Eine Übersicht über aktuelle Ansätze wird gegeben.

3.4.1 Objektdetektion

Bei der Detektion werden Objekthypothesen generiert. Objekthypothesen werden meistens als 2-D- oder 3-D-*Bounding-Boxen* modelliert. Verfahren der Objekthypothesen wurden von Hosang et al. [49] in Details diskutiert. Ein einfaches Verfahren, um Objekthypothesen zu generieren, war das sogenannte Schiebefenster-Verfahren. Dieses Verfahren durchsuchte alle Bereiche des Bildes und generierte für jeden Bereich Objekthypothesen. Die Anzahl der Bereiche sowie der Hypothesen, die pro Bereich generiert wurden, konnten parametrisiert werden. Dieses Verfahren war einfach zu implementieren. Allerdings konnte das Verfahren eine große Menge an Hypothesen generieren, was den Rechenaufwand der Klassifikation erhöhte. Generierte dieses Verfahren wenige Objekthypothesen, konnte dadurch die Güte der Klassifikation sinken. Es ist also wichtig, Verfahren zu verwenden, die so wenige Hypothesen wie möglich generieren, ohne die Güte der darauf basierenden Klassifikation zu verschlechtern. Neue Verfahren zur Generierung von Hypothesen, von Hosang et al. [49] *detection proposals* genannt, setzten voraus, dass Objekte Merkmale hatten, die sie vom Hintergrund unterschieden. Diese Merkmale konnten dann verwendet werden, um die Wahrscheinlichkeit zu schätzen, dass eine Hypothese ein Objekt enthielt. Damit wurden nur Hypothesen mit hohen Wahrscheinlichkeiten klassifiziert. Gute Merkmale führten zu einer Reduzierung der Objekthypothesen, ohne die Klassifikation zu gefährden. *Selective-Search* [50], *Geodesic* [51], *Objectness* [52] und *Edge Boxes* [53] sind einige *Detection-proposals*-Verfahren, die laut Hosang et al. [49] beste Evaluationsergebnisse lieferten.

3.4.2 Objektklassifikation

Während der Klassifikationsphase werden Merkmale für die detektierten Objekthypothesen generiert und die Klassenwahrscheinlichkeiten der Objekthypothesen werden anhand der generierten Merkmale geschätzt.

Merkmale sollen Objekte eindeutig beschreiben. Diese können manuell generiert oder anhand von maschinellen Lernverfahren wie das *tiefe Lernen* gelernt werden. Die Klassifikatoren bilden die Merkmale von Objekthypothesen auf Klassenwahrscheinlichkeiten dieser Hypothesen ab.

3.4.2.1 Flaches Lernen

Beim flachen Lernen werden die Merkmale meistens manuell generiert. Die Klassifikatoren sind äquivalent zu neuronalen Netzen weniger als zwei verdeckte Schichten und werden deshalb als flache Klassifikatoren bezeichnet. Bei Dalal und Triggs [54] wurden Personen anhand des Merkmals namens *Histogram of Oriented Gradients* (*HOG*) und eines Klassifikators namens *Support Vector Machine* (*SVM*) klassifiziert. Wang et al. [55] kombinierten *HOG*-Merkmale mit *Local-Binary-Pattern*-Merkmalen, um Personen mit einem *SVM*-Klassifikator zu klassifizieren. *Deformable Parts Model* wurden als Merkmal von Felzenszwalb et al. [56] verwendet, um Objekte mit einem *Latent-SVM*-Klassifikator zu klassifizieren. Zur Klassifikation von Fahrzeugen in Stereokamerabildern kombinierten Kowsari et al. [57] *Haar-Wavelet*-Merkmale mit dem *Multi-view-AdaBoost-Ensemble*-Modell. Während auf flachem Lernen basierte Klassifikatoren einfach zu implementieren und zu verstehen sind, haben sie den Nachteil, dass die Klassifikation unter den Fehlern und Mängeln der manuell generierten Merkmale leidet.

3.4.2.2 Tiefes Lernen

Beim tiefen Lernen werden die Merkmale aus großen Datenmengen mit tiefen Modellen wie TNN gelernt. *AlexNet* [58], *VGGNet* [59], *GoogleNet* [60], *ResNet* [61] und *SENet* [62] sind einige TNN, die die besten Klassifikationsergebnisse auf dem *ImageNet*-Datensatz erzielten. Das *ResNet*-Modell [61] erreichte 2015 eine Top-5-Fehlerrate von 3,57 % auf dem *ImageNet Large Scale Visual Recognition Challenge* (*ILSVRC*) und war somit besser als die menschliche Fehlerrate, die bei 5,1 % lag.

Die Netze *AlexNet* [58] und *VGGNet* [59] sind exemplarisch in der Abbildung 3-5 dargestellt. Das Basisnetz des *AlexNet*-Modells ist relativ einfach im Vergleich zu aktuellen Netzen und besteht lediglich aus fünf Faltungs- (abgekürzt *Conv*) und drei *Max-Pooling*-Schichten (abgekürzt *Pool*). Das anwendungsspezifische Netz für die Klassifikation besteht aus zwei vollständig vernetzten Schichten (abgekürzt *FC*) und dem *Softmax*-Klassifikator. Das *VGGNet*-Modell namens *VGG-16* erhöht die Tiefe des Netzes im Vergleich zum *AlexNet* mit jeweils 13 Faltungs- und vier *Max-Pooling*-Schichten im Basisnetz sowie drei vollständig vernetzen Schichten im anwendungsspezifischen Netzt. Die Erhöhung der Tiefe des *VGG-16*-Modells im Vergleich zum *AlexNet* erklärt die Verbesserung der Klassifikationsgüte mit dem *VGG-16*-Modell.

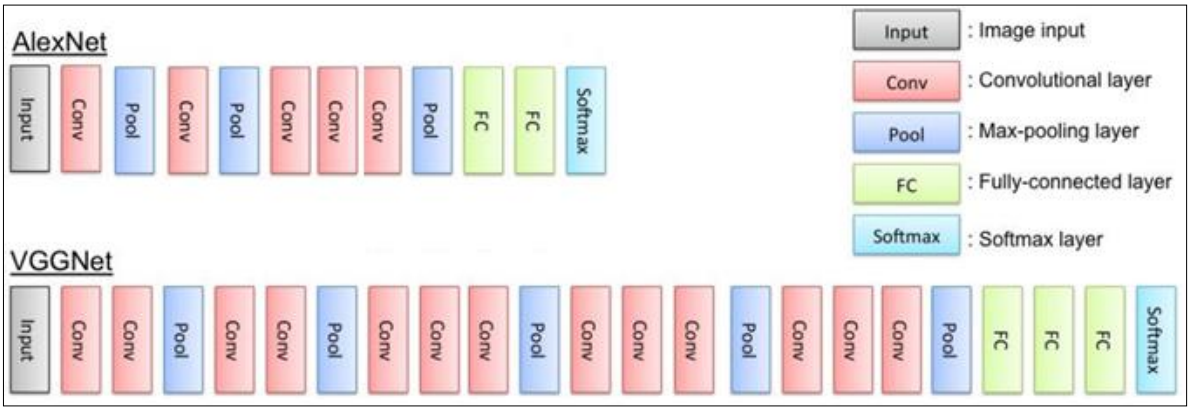


Abbildung 3-5: Architektur der tiefen neuronalen Netze AlexNet [58] (oben) und VGG-16 [59] (unten) (Bild aus [63] angepasst)

TNN wie *OverFeat* [64], *R-CNN* [65], *Faster R-CNN* [66], *YOLO* [67], *SSD* [68] kombinieren die Detektion und die Klassifikation in einem einzigen Netz und erreichen die besten Erkennungsraten. Ein Vergleich der Objekterkennungsnetze *Faster R-CNN* [66] und *SSD* [68] bei Huang et al. [69] zeigt, dass *Faster R-CNN* auf dem *Microsoft-COCO*-Datensatz [70] eine bessere Erkennungsrate hatte, während *SSD* schneller ist.

3.4.3 Zusammenfassung der bildbasierten Objekterkennung

Die Objekterkennung bestand aus der Hypothesengenerierung und der Klassifikation. Die Hypothesengenerierung konnte alle Regionen des Bildes durchsuchen oder mithilfe von *Objects-proposal*-Verfahren gezielt bestimmte Regionen durchsuchen. Mit dem *Objects-proposal*-Verfahren wurde die Anzahl der zu klassifizierenden Hypothesen reduziert, ohne die Klassifikationsgüte zu verschlechtern. Bei der Klassifikation wurde zwischen flachen und tiefen Modellen unterschieden. Tiefe Modelle wie TNN haben seit 2012 die beste Klassifikationsgüte. Bei einigen Datensätzen waren TNN sogar besser als Menschen.

Sowohl flache als auch tiefe Modelle hatten Schwierigkeiten, wenn die Szenenkomplexität durch Faktoren wie Verdeckungen, Reflektionen, Sensorrauschen, niedrige Sensorauflösungen erhöht wird, da die zugrundeliegenden aussehenbasierten Merkmale durch die o. g. Faktoren verrauscht wurden. Darüber hinaus haben TNN aufgrund der hohen Komplexität der Netze den Nachteil, dass Merkmale, die für Klassifikationsergebnisse relevant waren, schwer nachzuvollziehen sind. Somit konnten die Konfidenzen solcher Klassifikatoren bei unerwarteten Situationen nicht vorhergesagt werden.

3.5 Semantische Segmentierung von Kamerabildern

Die semantische Segmentierung beschäftigt sich mit der Klassifikation von einzelnen Pixeln in Bildern. Im Vergleich zur Objekterkennung hat sie den Vorteil, dass die Szene vollständig semantisch klassifiziert wird. Wichte Regionen für selbstfahrende Fahrzeuge wie die Straße, Bordsteine, Spurmarkierungen und weitere Hindernisse, die sich nicht als *Bounding-Boxen* erfassen lassen, können von der semantischen Segmentierung erkannt und klassifiziert werden. Die semantische Segmentierung spielt deshalb eine wichtige Rolle im Kontext des automatisierten Fahrens und wird ausführlich in dieser Arbeit vorgestellt.

Ähnlich wie bei der Klassifikation von Objekten besteht die semantische Segmentierung aus zwei Phasen: der Extraktion von Merkmalen und der Klassifikation.

3.5.1 Manuelle Generierung von Merkmalen und Klassifikation

Einfache Merkmale, die für die semantische Segmentierung relevant sind, sind die Farbe, der Grauwert, die Textur und die Kanten [71]. Komplexere Merkmale wurden entwickelt, um die Segmentierung zu verbessern. Shotton et al. [72] verwendeten ein Merkmal namens *texture-layout filters*. Dieses Merkmal kombinierte Texturinformationen mit den räumlichen Anordnungen dieser Texturelemente [72]. Um die Komplexität der Segmentierung zu reduzieren, wurden Pixel anhand von Merkmalen in Regionen (sogenannten *Texton*) gruppiert. Alle Pixel einer Region gehörten zu derselben semantischen Klasse. Die Klassifikation erfolgte dann auf Regionen, die deutlich weniger waren, als die Pixelanzahl. Die Schritte zur Generierung von *Texton* sind in der Abbildung 3-6 dargestellt. Aus dem Inputbild wurden Texturmerkmale anhand von unterschiedlichen Filtern (siehe *filter bank* in der Abbildung 3-6) generiert. Danach wurden die Texturmerkmale verwendet, um *Texton* zu generieren. Pixel, die zu einem *Texton* gehörten, hatten die gleiche Farbe (siehe *texton map* in der Abbildung 3-6) [72].

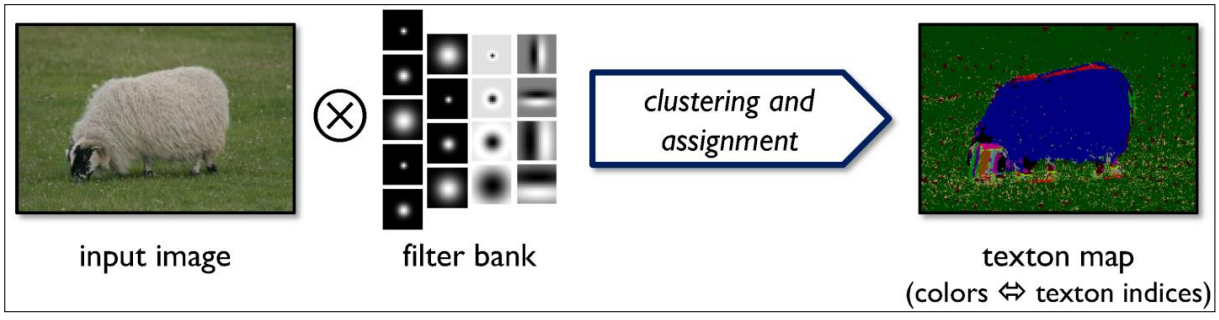


Abbildung 3-6: Exemplarische Darstellung der Schritte zur Generierung von Texton (Bild aus [72])

Für die Klassifikation anhand der o. g. manuell generierten Merkmale wurden in der Literatur oft *Conditional Random Fields* (*CRF*) verwendet. *CRF* sind PGM-Modelle, die anstelle der Verbundwahrscheinlichkeitsverteilung von Markov-Netzen aus der Gleichung (7) die bedingte Wahrscheinlichkeitsverteilung der Ausgaben, gegeben die Eingaben, modellieren. Der Vorteil ist, dass bedingte Wahrscheinlichkeitsverteilungen einfacher als Verbundwahrscheinlichkeitsverteilungen zu lernen sind. Für Anwendungen, wo die Ein- und Ausgaben klar definiert sind, sind *CRF* im Vergleich zu Markov-Netzen besser. Für die semantische Segmentierung sind die Knoten des *CRF*-Modells Pixel bzw. Super-Pixel. Eine unäre Potentialfunktion modelliert die Abhängigkeit zwischen der semantischen Klasse der Pixel bzw. Super-Pixel und den Inputmerkmalen. Die n-stellige Potentialfunktion modelliert die Abhängigkeit zwischen den semantischen Klassen von benachbarten Pixeln bzw. Super-Pixeln. Die Parameter der Potentialfunktionen können mit den Algorithmen aus dem Abschnitt 2.2.4 gelernt werden. Die Segmentierung eines gegebenen Bildes erfolgt mit dem *MAP*-Verfahren aus Abschnitt 2.2.3.

Shotton et al. [72] verwendeten die *texture-layout*-Filter, die Farbe und die absolute Position der Pixel als Merkmale von unären Potentialfunktionen eines *CRF*-Modells. Die binäre Potentialfunktion des *CRF*-Modells nahm als Inputmerkmal den Farbenunterschied zwischen zwei benachbarten *Texton*. Weitere Kontextinformationen, wie das gemeinsame Auftreten oder die relative Position von zwei Pixeln bzw. Super-Pixeln, gegeben ihre semantischen Klassen, wurden unter anderem von Ladicky et al. und Rabinovich et al. [73, 74] als Inputmerkmal von binären Potentialfunktionen von *CRF*-Modellen verwendet. Die Integration von Kontextinformationen führte zur Verbesserung der Segmentierung.

Der Vorteil von *CRF*-basierten Segmentierungen war, dass Kontextinformationen mit *CRF*-Modellen modelliert werden konnten. Damit wurden Inkonsistenzen in der Segmentierung reduziert. Der Nachteil lag darin, dass die Inferenz sehr komplex war. Darüber hinaus waren die manuell generierten Merkmale, wie bei der Objekterkennung oft, nicht ausreichend, um eine gute Segmentierungsgenauigkeit zu erreichen.

3.5.2 Lernen von Merkmalen mit tiefem Lernen

Mit dem Erfolg von TNN bei der Objekterkennung wurde von Long et al. [75] die sogenannten vollständig gefalteten Netze (*FCN*: *Fully Convolutional Networks*) zur semantischen Segmentierung verwendet. Diese vollständig gefalteten Netze werden in dem nächsten Abschnitt ausführlich beschrieben.

3.5.2.1 *FCN* zur semantischen Segmentierung

Das *FCN*-Modell funktioniert wie folgt [75]:

1. Faltungsbildung (siehe *convolutionalization* in der Abbildung 3-7): Das Netz, das für die Objektklassifikation verwendet wurde (z. B. *AlexNet* [58]), wurde angepasst, indem die vollständig vernetzten Schichten zu gefalteten Schichten umgewandelt wurden. Damit wurde die räumliche Abhängigkeit zwischen der Aktivierungskarte der Ausgabeschicht (siehe *Heatmap* in der Abbildung 3-7) und dem Eingangsbild sichergestellt. Die Aktivierungskarte war ein 3-D-Volumen $S \in \mathbb{R}^{m \times n \times k}$, wobei die Tiefe k der Anzahl der Eingangsklassen entsprach. Die Breite und die Höhe der Aktivierungskarte entsprachen der herunterskalierten Breite und Höhe des Eingangsbildes. Das hier verwendete Netz wurde *Encoder* genannt. Die Abbildung 3-7 stellt die Faltungsbildung dar.

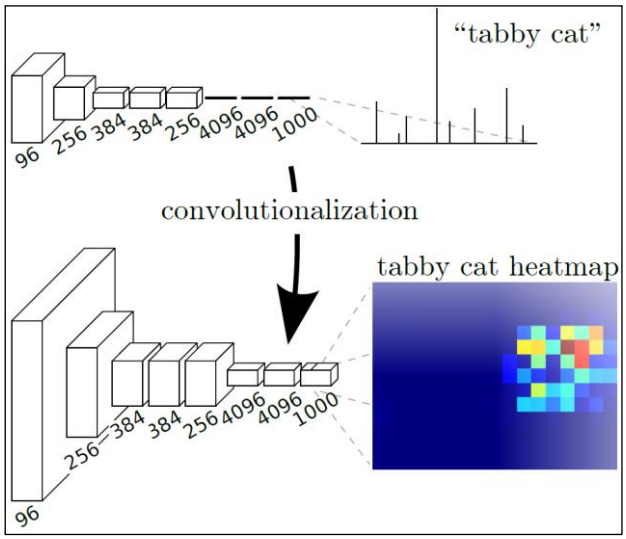


Abbildung 3-7: Generierung von Faltungsschichten aus vollständig vernetzten Schichten (Bild aus [75])

2. Entfaltung: Sie diente zum Hochskalieren der Aktivierungskarte auf die Größe des Eingangsbildes. Die Entfaltung invertierte also die Faltungen, die im *Encoder* stattfanden. Das für die Entfaltung verwendete Netz wurde *Decoder* genannt. Die Parameter der Entfaltung konnten gelernt werden. Eine andere Möglichkeit war, eine bilineare Interpolation in der Entfaltung zu verwenden. Eine *Softmax*-Funktion (siehe Gleichung (19)) berechnete die pixelweise Klassenwahrscheinlichkeiten mit der Aktivierungskarte des *Decoders* als Input.

Der *Encoder* und der *Decoder* wurden gemeinsam trainiert, da der *Decoder* lediglich die Faltung invertierte und somit mit dem Lernverfahren aus dem Abschnitt 2.4.3 kompatibel war.

Obwohl das o. g. *FCN*-Netz mit dem *VGG-16*-Netz als *Encoder* gute Ergebnisse lieferte, war die Segmentierung aufgrund der geringen Auflösung der Aktivierungskarte des *Encoders* grob. Um dieses Problem zu lösen, wurde die Aktivierungskarte am Ausgang des *Encoders (conv7*-Schicht in der Abbildung 3-8*)* mit der detailreichen Aktivierungskarte der vorangegangenen Schichten *pool4* und *pool3* elementweise addiert. Dieses Verfahren konnte als eine Integration von lokalen Kontextinformationen aus den Schichten *pool4* und *pool3* mit der globalen Kontextinformation aus der tieferen Schicht *conv7* interpretiert werden. Die entstandenen Modelle wurden jeweils *FCN-16s* und *FCN-8s* genannt (siehe Abbildung 3-8). Im Jahr 2014 erreichte das beste Modell *FCN-8s* 20 % relative Verbesserung der Metrik *mean Intersection Over Union* (*mIOU*) im Vergleich zum dem Stand der Technik auf den *PASCAL-VOC2011-* [76] und *PASCAL-VOC2012*-Datensätzen [77]. Gleichzeitig konnte das *FCN-8s* die Inferenzzeit um mehr als hundertfach verbessern [75]. Das *FCN-8s*-Modell wird deshalb als ein Meilenstein im Bereich der semantischen Segmentierung betrachtet. Viele aktuelle Modelle basieren darauf [78].

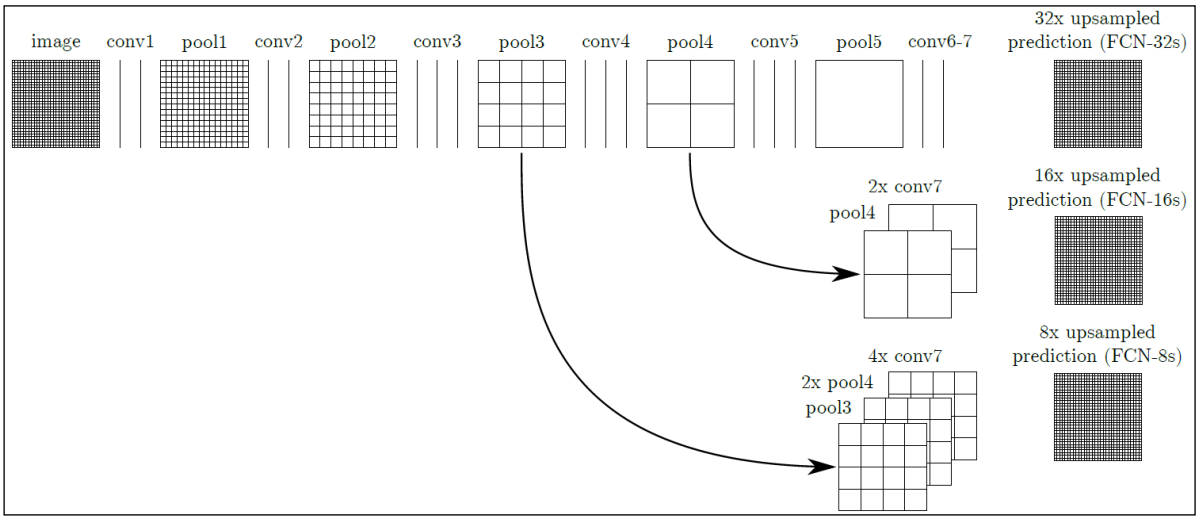


Abbildung 3-8: Illustration der Kombination der Aktivierungskarte aus unterschiedlichen Schichten zur detailreichen semantischen Segmentierung von Bildern (Bild aus [75])

3.5.2.2 *SegNet*-Modell

Basierend auf dem Erfolg des *FCN*-Modells wurde von Badrinarayanan et al. [79] im Jahr 2015 das *SegNet*-Modell vorgeschlagen. Ähnlich wie beim *FCN-8s-Modell* besaß dieses Modell das *VGG-16*-Modell als *Encoder*. Der Unterschied zum *FCN-8s* lag in dem *Decoder*. Der *SegNet-Decoder* verwendete die Indexe, die bei den *Pooling*-Schichten im *Encoder* verwendet wurden, um die Aktivierungskarte der Ausgangsschicht des *Encoders* schrittweise hoch zu skalieren, bis die Auflösung des Inputbildes erreicht wurde. Nach jeder Skalierungsoperation wurden die gleichen Faltungsschichten, die im *Encoder* verwendet wurden, im *Decoder* angewendet. Die letzte Schicht des *Decoders* war der *Softmax*-Klassifikator zur Generierung der Pixel-Klassenwahrscheinlichkeiten. Die Abbildung 3-9 stellt das *SegNet*-Modell dar. Im Vergleich zum *FCN-8s*-Modell hatte das *SegNet*-Modell den Vorteil, dass es wenig Speicher während der Inferenz verbrauchte. Basierend auf der *mIOU*-Metrik haben beide Modelle ähnliche Genauigkeiten auf dem *Cambridge-driving-Labelled-Video-Database*-(*CamVid*-)Datensatz [80]. Das *FCN-8s*-Modell war bei der Inferenz schneller als das *SegNet*-Modell [79]. Auf dem *Daimler-Cityscapes-Dataset*-(*DCS*-) Datensatz [81] war das *FCN-8s*-Modell 10 % besser als das *SegNet*-Modell, bezogen auf die *mIOU*-Metrik (siehe Tabelle 3-1).

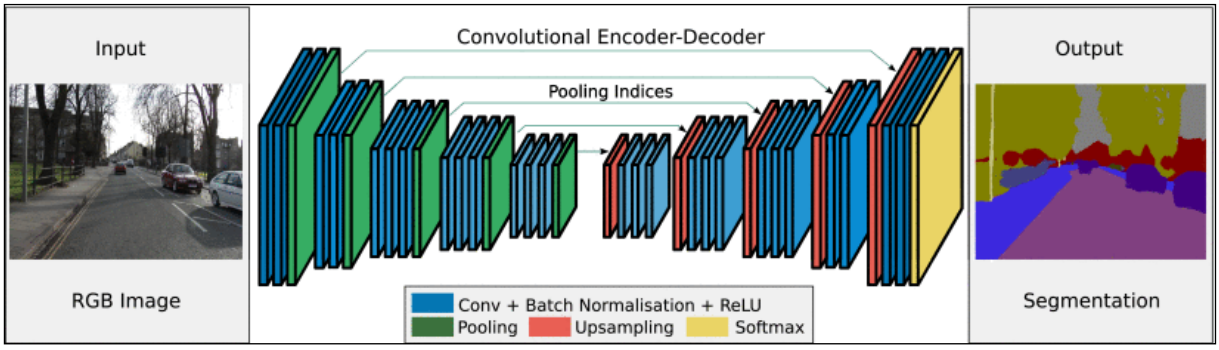


Abbildung 3-9: Architektur des SegNet-Modells mit dem VGG-16-Modell als Encoder (Bild aus [79])

3.5.2.3 *Dilation10*-Modell

Eine Möglichkeit, die Segmentierung zu verbessern, war das rezeptive Feld und die Auflösung der Faltungsschichten des *Encoders* zu erhöhen, um gleichzeitig mehr globale und lokale Kontextinformationen zu erfassen. Eine einfache Erhöhung der rezeptiven Felder der Faltungskerne führte zu einer exponentiellen Steigerung der Parameteranzahl der Faltungskerne. Um dieses Problem zu adressieren, wurden von Yu und Koltun [82] die sogenannten gedehnten Faltungen vorgeschlagen. Die Idee dabei war, mehrere Faltungskerne zu verwenden, wobei alle diese Kerne die gleiche Anzahl von Parametern hatten. Der Unterschied zwischen den Faltungskernen bestand in der räumlichen Auflö-

sung, die durch den Dehnungsparameter l gesteuert wurde. Die Abbildung 3-10 stellt den Einfluss des Dehnungsparameters $l \in \{1, 2, 4\}$ auf dem rezeptiven Feld der Faltungskerne mit der Dimension 3×3 dar. Die roten Punkte trugen zu der Faltung des Punktes im Zentrum des Bildes bei. Der grüne Bereich war das rezeptive Feld des Faltungskerns. Für $l = 1$ blieb das rezeptive Feld unverändert (siehe grüner Bereich in (a) Abbildung 3-10). Für $l = 2$ erreichte man ein 7×7 rezeptives Feld (siehe grüner Bereich in (b) Abbildung 3-10) durch die sukzessive Anwendung der Faltung mit $l = 1$ und $l = 2$ mit jeweils 9 Parametern. Die gedehnte Faltung hatte in diesem Fall 18 Parameter. Eine normale Faltung mit einem 7×7 rezeptiven Feld hätte 49 Parameter gehabt. Die gedehnte Faltung erreichte also das gleiche rezeptive Feld mit einer linearen Steigerung der Parameter, während eine normale Faltung mit einer exponentiellen Steigerung der Parameter verbunden war. Im Bild (c) der Abbildung 3-10 war $l = 4$ und das rezeptive Feld 15×15. Analog zum Fall $l = 2$ hatte die gedehnte Faltung insgesamt 27 Parameter. Die gedehnten Faltungen wurden in dem Modell *Dilation10* angewendet. Dieses Modell hatte etwas mehr als 5 % absolute *mIOU* als das *FCN-8s*-Modell auf dem *PASCAL-VOC2012*-Testdatensatz [77], [82]. In der Tabelle 3-1 beträgt die relative Verbesserung des *Dilation10*-Modells im Vergleich zum *FCN-8s*-Modell auf dem *DCS*-Testdatensatz 2,5 %. Allerdings ist das *Dilation10*-Modell mit ca. vier Sekunden Inferenzzeit achtmal langsamer als das *FCN-8s*-Modell.

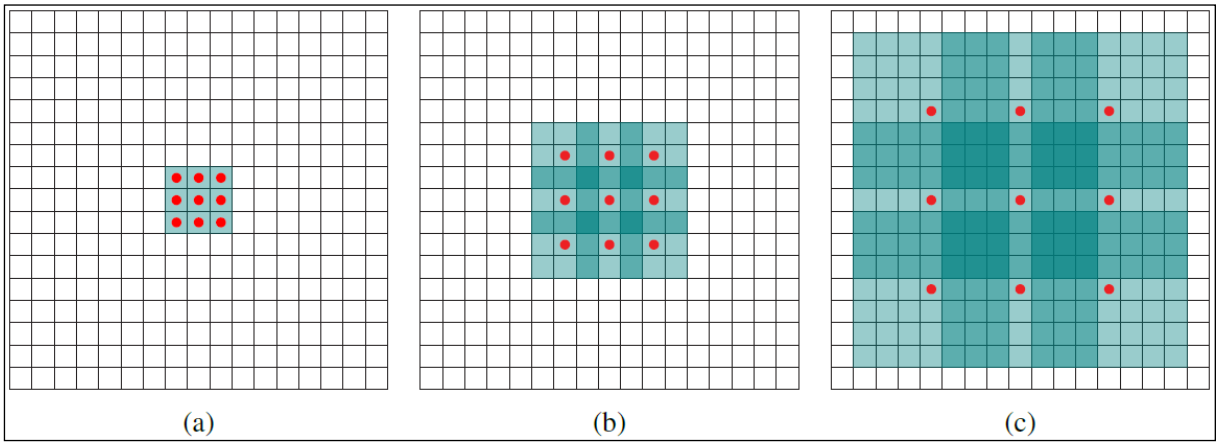


Abbildung 3-10: Schematische Darstellung der rezeptiven Felder der gedehnten Faltungskerne mit dem Dehnungsparameter $l = 1$ (Bild (a)), $l = 2$ (Bild (b)) und $l = 4$ (Bild (c)). Die roten Punkte tragen zu der Faltung des Punktes im Zentrum des Bildes bei. Der grüne Bereich ist das rezeptive Feld des Faltungskerns. (Bild aus [82])

3.5.2.4 *PSPNet*-Modell

Eine andere Möglichkeit, globale und lokale Kontextinformationen zu verwenden, um die Segmentierung zu verbessern, war die Erweiterung der letzten *Pooling*-Schicht des *Encoders* mit mehreren *Pooling*-Schichten, wobei jede Schicht einen anderen Skalierungsfaktor hatte. Diese Idee wurde von Zhao et al. [83] mit dem *PSPNet*-Modell vorgeschlagen. Die Abbildung 3-11 stellt die Architektur des *PSPNet*-Modells dar. Das *Pyramid-Pooling*-Modul bestand aus vier parallelen *Pooling*-Schichten mit unterschiedlichen Skalierungsfaktoren. Jede *Pooling*-Schicht verwendete zusätzlich eine Faltungsschicht. Der *Decoder* skalierte die herunterskalierten Aktivierungskarten hoch und konkatenierte diese. Eine letzte Faltungsschicht sorgte für die Generierung der Pixelklassenscore. Die Evaluation des *PSPNet*-Modells mit *ResNet-101*-Modell als *Encoder*, eine Version des *ResNet*-Modells [61] mit 101 Schichten auf dem *DCS*-Testdatensatz, zeigte eine 24 % relative Verbesserung der *mIOU* im Vergleich zum *FCN-8s*-Modell. Allerding verwendete das *FCN-8s*-Modell das im Vergleich zum *ResNet101* einfachere *VGG-16*-Modell als *Encoder*.

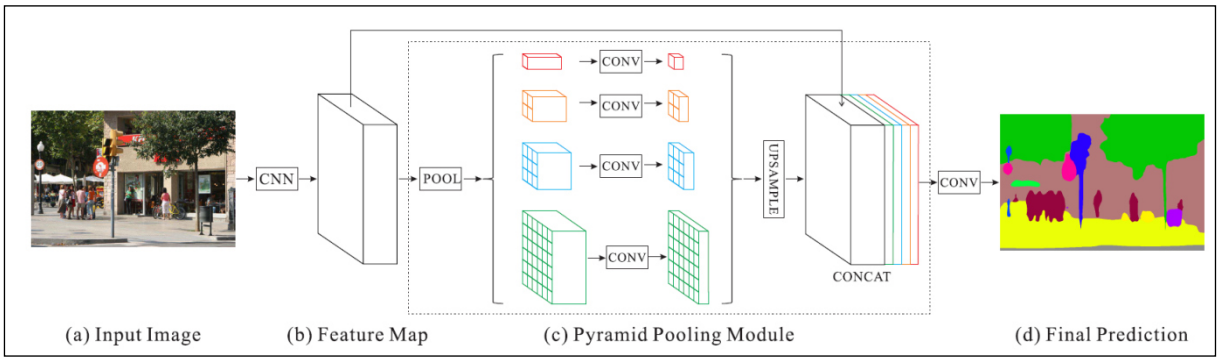


Abbildung 3-11: Architektur des PSPNet-Modell aus [83] (Bild aus [83])

3.5.2.5 *DeepLab*-Modell

Von Chen et al. [84] wurde die das *Pyramid-Pooling-Modul* aus [83] mit den gedehnten Faltungsschichten aus [82] kombiniert. Dazu hatte der *Decoder* zwei Hochskalierungsschichten anstatt einer wie in [83] (siehe Bild (c) in der Abbildung 3-12). Diese Kombination wurde in das *DeepLabv3+*-Modell integriert. Das *DeepLabv3+*-Modell mit dem *ResNet101*-Modell als *Encoder* verbesserte den *mIOU*-Wert des *PSPNet*-Modells um ca. 1,1 % auf dem *DCS*-Testdatensatz.

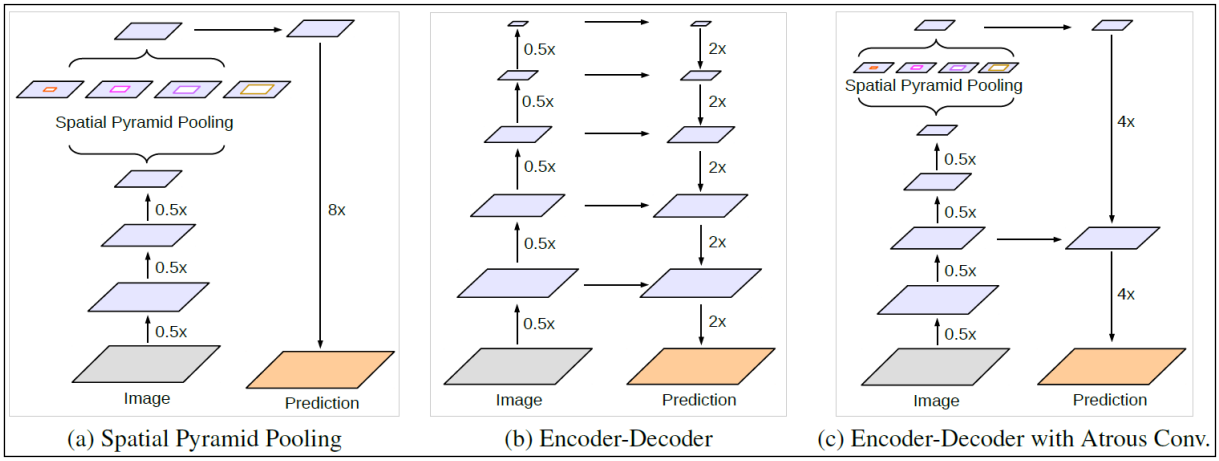


Abbildung 3-12: Schematische Darstellung des DeepLabv3+-Modells (c) als Kombination des PSPNet-Modells (a) und der Encoder-Decoder-Architektur (b) (Bild aus [84])

3.5.2.6 *DUC*-Modell

Das *Dense-Upsampling–Convolution*-(*DUC*-)Modell wurde von Wang et al. [85] vorgeschlagen. Die Idee war, ein *DUC*-Modul als *Decoder* zu verwenden. Sei W und H die Breite und die Höhe des Eingangsbildes des *Encoders*, W/d und H/d die Breite und die Höhe der herunterskalierten Aktivierungskarte der letzten Schicht des *Encoders* und L die Anzahl der Klassen. Faltete man die herunterskalierte Aktivierungskarte der letzten Schicht des *Encoders* mit d^2L Faltungsschichten, bekam man eine Aktivierungskarte mit der Dimension $\frac{W}{d} \times \frac{H}{d} \times d^2L = W \times H \times L$. Dies bedeutete, dass der Output des *DUC*-Moduls die Informationen aller Pixel des Inputbildes enthielt. Diese Informationen müssen nur noch angeordnet werden, um auf die Dimension des Inputbildes zu kommen. Der *DUC-Decoder* hatte den Vorteil, dass alle seine d^2L-Faltungsschichten gelernt werden konnten. Darüber hinaus wurde eine hybride gedehnte Faltung vorgeschlagen, um das Gitternetz-Problem der gedehnten Faltung zu adressieren. Die entstandenen hybriden gedehnten Faltungsschichten hatten im Vergleich zu den gedehnten Faltungsschichten eine variable Dehnungsrate. Die Architektur des *DUC*-Modells mit dem *RestNet-101* als *Encoder* (*RestNet-DUC* genannt) ist in der Abbildung 3-13 dargestellt. Auf dem *DCS*-Testdatensatz erreichte das beste *RestNet-DUC*-Modell 80,1 % *mIOU* [85] und war somit ca. 2,5 % schlechter als das *DeepLabv3+*-Modell.

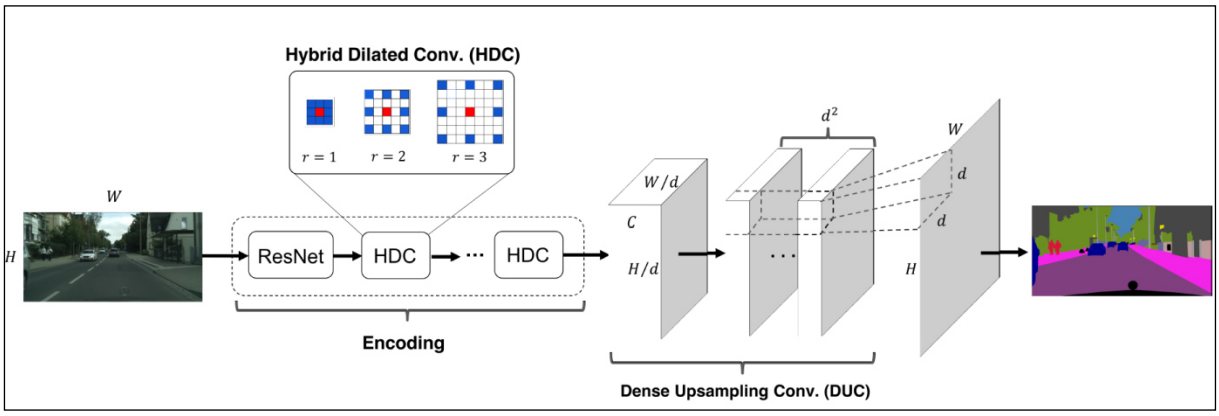


Abbildung 3-13: Architektur des ResNet-DUC-Modells mit RestNet-101 als Encoder (Bild aus [85])

Tabelle 3-1: Ausschnitt der Evaluation einiger relevanter Modelle, die auf der DCS-Benchmark-Seite publiziert wurden, basierend auf dem DCS-Testdatensatz [86]

Name	Klassen-mIOU (%)	Kategorien-mIOU (%)	Laufzeit (s)
DeepLabv3+ [84]	82,1	92	keine Eingabe
PSPNet [83]	81,2	91,2	keine Eingabe
RestNet-DUC [85]	80,1	keine Eingabe	keine Eingabe
Dilation10 [82]	67,1	86,5	4
FCN-8s [75]	65,3	85,7	0,5
SegNet basic [79]	57	79,1	0,06

3.5.2.7 Modellierung von Unsicherheiten in TNN

Die bis jetzt vorgestellten TNN haben den Nachteil, dass sie keine Unsicherheiten modellieren. Um dieses Problem zu lösen, wurde in [87] eine *Monte-Carlo-Sampling*-Methode basierend auf dem *Dropout*-Verfahren vorgestellt. Das *Dropout*-Verfahren wurde oft beim Trainieren von TNN verwendet, um die Generalisierung der Netze zu verbessern. Dabei werden im *Forward pass* einige Knoten zufällig ausgeschaltet. Das Netz ist damit gezwungen, redundante Merkmale zu lernen. Geht man davon aus, dass Unsicherheiten im Netz durch Mangel an redundanten Merkmalen verursacht werden, und verwendet man zur Inferenzzeit das *Dropout*-Verfahren mehrmals für das gleiche Eingabebild, kann der Output des Netzes als Sample einer Normalverteilung betrachtet werden. Der Mittelwert und die Varianz dieser Verteilung entsprechen jeweils der Prädiktion und der Unsicherheit des Netzes bezogen auf das Inputbild. Dieses *Sampling*-Verfahren wurde von Kendall et al. [88] verwendet, um Unsicherheiten in das *SegNet*-Modell [79] zu integrieren. Die Architektur des *Bayesian-SegNet*-Modells ist in der Abbildung 3-14 dargestellt. Das gleiche Bild wurde mehrmals vom Netz mit dem *Dropout*-Verfahren inferiert. Die Outputs der *Softmax*-Funktion wurden als Samples einer Gauß-Verteilung betrachtet. Die Segmentierung war der pixelweise Mittelwert der Samples und die pixelweise Varianz dieser Samples war die pixelweise Unsicherheit der Segmentierung. Die Evaluation des *Bayesian-SegNet*-Modells auf dem *CamVid*-Datensatz zeigte eine relative Verbesserung des *mIOU*-Wertes von ca. 26 % im Vergleich zum *Segnet*-Modell. Darüber hinaus war die vom *Bayesian-SegNet* inferierte Unsicherheit kleiner bei Klassen mit hohen *Intersection-Over-Union*-(*IOU*-)Werten und größer bei Klassen mit kleinen *IOU*-Werten. Dies deutete darauf hin, dass das *Dropout*-Verfahren in der Lage war, die Unsicherheit des Modells zu modellieren [88]. Ein Nachteil des *Bayesian-SegNet*-Modells war der höhere Speicherbedarf und die größere Inferenzzeit im Vergleich zum *SegNet*-Modell.

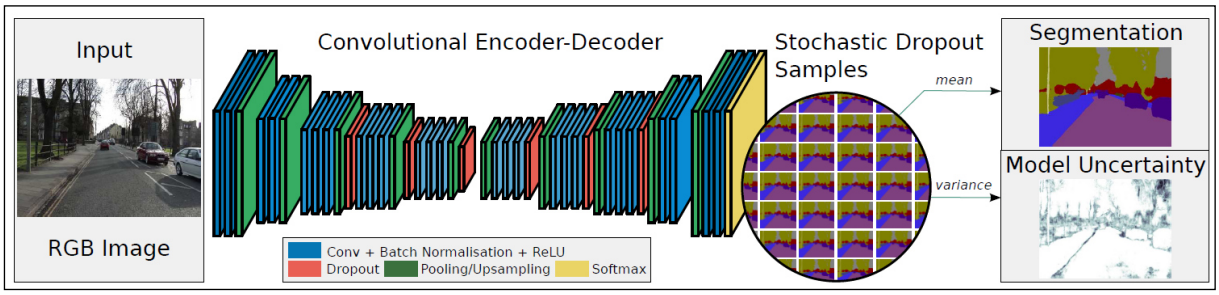


Abbildung 3-14: Architektur des Bayesian-SegNet-Modells. Das Dropout-Verfahren wird verwendet, um die Unsicherheit der semantischen Segmentierung zu erfassen. (Bild aus [88])

3.5.3 Weitere Modelle mit dem Fokus auf Kontextinformationen

In der Literatur wurden Ansätze zur Kombination von *CRF* und TNN vorgeschlagen, um die semantische Segmentierung zu verbessern, da *CRF*-Modelle geeignet waren, um lokale und globale Kontextinformationen zu modellieren. Diese Informationen waren entscheidend für eine kontextkonsistente Segmentierung. Von den Autoren in [89–96] wurden bspw. *CRF*-Modelle als *Post-Processing* verwendet oder mit neuronalen Netzen approximiert.

Von Rota Bulò et al. [97] wurde eine Variante der *Batch*-Normierung namens *In-Place Activated Batch Normalization (InPlace-ABN)* vorgeschlagen. Die Idee war, den Speicherbedarf der *Batch*-Normierung während des Trainings zu reduzieren und somit größere *Batches* zu verwenden. Diese Variante der *Batch*-Normierung wurde ins *ResNeXt-101*-Modell integriert. Das *ResNeXt-101*-Modell war eine Anpassung des *ResNet101*-Modells, in dem bestimmte Faltungsschichten der *Residual*-Einheiten gruppiert wurden. Das *ResNeXt-101*-Modell wurde dann als *Encoder* des *DeepLabv3+*-Modells von Chen et al. [98] verwendet. Das Modell wurde mit dem *InPlace-ABN*-Verfahren trainiert. Die Evaluation auf dem *DCS*-Testdatensatz zeigte eine relative Verbesserung der *mIOU* um ca. 1 % im Vergleich zum originalen *DeepLabv3+*-Modell.

Knaujia et al. [99] verwendeten ein MLN-Modell, um die semantische Segmentierung der Bodenebene in Videodaten zu verbessern. Dafür wurde zuerst die relative Position zwischen der Bodenebene und Fußgänger sowie die Wahrscheinlichkeit einer richtigen Segmentierung in einem MLN-Modell modelliert. Danach wurde die segmentierte Bodenebene bezogen auf die detektierten Fußgänger angepasst.

Die Integration von Kontext in TNN-Modelle durch das gemeinsame Lernen von mehreren Aufgaben wurde auch in der Literatur vorgeschlagen. Eine Übersicht dieser Ansätze hat Ruder in [100] dargestellt. Von Teichmann et al. [101] wurden die Freiraumerkennung, die Erkennung der Szenegeometrie und die Fahrzeugerkennung kombiniert. Kendall et al. [102] kombinierten die Tiefenschätzung, die semantische Segmentierung und die Instanz-Segmentierung. Von Liao et al. [103] wurde eine Kombination der Szenenklassifikation und der semantischen Segmentierung vorgeschlagen.

Von Zheng et al. und Liang et al. [93, 104] wurden RNN-Modelle verwendet, um mehr Kontext in die semantische Segmentierung zu bringen.

Gatta et al. [105] verwendeten die Ausgaben von vollständig vernetzten Schichten als eine Art globalen Kontext. Dazu wurde auch die *Heatmap* der Klassen als Prior verwendet. Dies verbesserte die Segmentierung und reduzierte die Inkonsistenzen.

Eine weitere Möglichkeit, den Kontext in die semantische Segmentierung zu integrieren, war die Fehlerfunktion des TNN-Modells bezogen auf die Relevanz von semantischen Klassen zu gewichten. So wurde von Chen et al. [106] die Relevanz der Pixel im Kontext des automatisierten Fahrens in der Fehlerfunktion integriert. Damit wurde das Modell gezwungen, besser relevante semantische Klassen zu segmentieren.

3.5.4 Zusammenfassung der Ansätze der semantischen Segmentierung

Zwei Klassen von Ansätzen der semantischen Segmentierung wurden in diesem Kapitel vorgestellt. Die erste Klasse war die Kombination von manuell generierten Merkmalen und *CRF*-Modellen. Während Kontextabhängigkeiten mit *CRF* gut modelliert wurden, waren diese Kontextabhängigkeiten meistens nicht ausreichend, um eine gute Segmentierungsgenauigkeit zu erreichen. Darüber hinaus führten die Fehler der manuell generierten Merkmale zur Verringerung der Segmentierungsgüte.

Die zweite Klasse verwendete TNN, um Merkmale automatisch anhand von Trainingsdaten zu lernen. Das Modell war oft ein *Encoder*-*Decoder*-Modell. Der *Encoder* generierte eine herunterskalierte Aktivierungskarte. Der *Decoder* skalierte die Aktivierungskarte hoch und berechnete die pixelweisen Klassenwahrscheinlichkeiten mit der *Softmax*-Funktion. Solche Modelle zeigten die besten Segmentierungsergebnisse. Der Hauptnachteil war jedoch die Komplexität der Modelle, die dazu führte, dass ihre Funktionsweise schwer nachzuvollziehen war. Darüber hinaus war die Integration von Kontextinformation durch das lokale rezeptive Feld der Neuronen eingeschränkt. Ein weiterer Nachteil dieser Modelle war die sogenannten *Adversarial*-Bilder, die die Segmentierung von TNN fälschten, während Menschen solche Bilder richtig segmentierten. *Adversarial*-Beispiele wurden u. a. von Metzen et al. [107] vorgestellt.

Hybride Ansätze zur Kombination der beiden Ansatzklassen wurden auch vorgeschlagen. Oft wurden *CRF* als *Post-Processing* von TNN verwendet. Manchmal wurden *CRF* in TNN nachgemacht. Solche hybriden Ansätze führten in der Regel zu einer besseren Klassifikationsgüte. Allerdings war die Kombination der Ansätze immer noch sehr mühsam, da sich die Algorithmen zum Lernen und Inferieren von *CRF* und TNN stark unterschieden.

Die Tabelle 3-2 fasst die Ansätze zusammen.

Tabelle 3-2: Zusammenfassung der Ansätze zur semantischen Segmentierung von Kamerabildern

Name	**PGM**	**TNN**	**Hybrid**
Beschreibung	Kombination von manuell generierten Merkmalen und *CRF*. *CRF* klassifizierten die Pixel mit probabilistischer Inferenz	Verwendung von TNN, um Merkmale anhand von Trainingsdaten zu generieren. Der Encoder generierte herunterskalierte Aktivierungskarten. Der Decoder skalierte die Aktivierungskarte hoch und berechnete die pixelweise Klassifikation mit der Softmax-Funktion	Kombination von *CRF* und TNN. *CRF* wurden entweder als Post-Processing verwendet oder in TNN nachgebildet
Pro	- zur Modellierung von lokalen und globalen Kontextabhängigkeiten geeignet - kontextkonsistente Segmentierung - probabilistische Inferenz nachvollziehbar - Unsicherheiten modelliert	- beste Segmentierungsergebnisse - automatisches Lernen von Merkmalen - Integration von lokalem Kontext	- Pro von *CRF* und TNN
Contra	- manuell generierte Merkmale manchmal fehlerbehaftet oder mangelhaft - Modellierung von *CRF*	- Blackbox - benötigt viel Trainingsdaten - Kontext durch das rezeptive Feld der Neuro-	- Blackbox - benötigt viel Trainingsdaten - Integration von PGM und TNN aufgrund der

Name	PGM	TNN	Hybrid
	mühsam bei komplexen Aufgaben wie Segmentierung - Segmentierungsgüte nicht gut genug	nen eingeschränkt - Unsicherheit schwer zu modellieren	Unterschiede zwischen CRF- und TNN-Algorithmen mühsam
Beispiele	- *TextonBoost* [72] - *CRF-GraphCut* [73] - *Object in Context* [74]	- *FCN* [75] - (*Bayesian) SegNet* [79, 88] - *DeepLab* [84] - *DUC* [85]	- *DeepLab + CRF* [89] - *CRF + RNN* [93] - *MLN* [99]

3.6 Situationsmodellierung und Interpretation

Die Situationserfassung besteht aus zwei Aufgaben: der Modellierung und der Interpretation der Situation. Diese beiden Aufgaben werden in den nächsten Abschnitten erläutert. Ein Überblick über den Stand der Technik wird gegeben.

Die Situationsmodellierung beschäftigt sich mit der Generierung des Situationsmodells, wobei dieses Modell im Abschnitt 3.1 definiert wurde. Ein Situationsmodell enthält alle für die Fahraufgabe relevanten Elemente der Szene wie die dynamischen Objekte und die Szenerie. Dazu werden Relationen zwischen diesen Szenenelementen inferiert. Das Situationsmodell wird darüber hinaus durch die Situationsinterpretation mit weiteren Attributen erweitert.

Die Situationsinterpretation nimmt als Input das Situationsmodell und interpretiert bestimmte Aspekte der Situation. Situationsaspekte sind Hypothesen, die geprüft werden sollen, damit das Ego-Fahrzeug die richtige Entscheidung treffen kann [108]. Einige Beispielaspekte sind die Konsistenz der Szenenelemente, die Relevanz der Szenenelemente und die Prädiktion der Situationsevolution über die Zeit.

Gemäß dem *JDL*-Fusionsmodell [47] gibt es keine strenge Hierarchie zwischen der Szenen- und der Situationserfassung. Die Situationserfassung kann Szenenelemente beeinflussen und umgekehrt. Dies ist eine holistische Betrachtung von Szenen und Situation. Allerdings ist diese bidirektionale Kommunikation zwischen der Szene und der Situation mit einer großen Komplexität verbunden. Das *PF2*-Fusionsmodell [48] schlug eine strenge *Bottom-up*-Hierarchie zwischen der Szenen- und der Situationserfassung vor, um die Komplexität der Aufgabe zu reduzieren. In dieser Hierarchie wird die Szene zunächst erfasst und als Input für die Situationsmodellierung und Interpretation bereitgestellt. Dieser *Bottom-up*-Ansatz wird oft im *Automotive*-Bereich verwendet.

3.6.1 Semantische Anreicherung des Situationsmodells

Die semantische Anreicherung eines Modells ist die Erweiterung dieses Modells mit Konzepten, Attributen und Relationen, die in einem semantischen Modell modelliert wurden [109]. Da Informationen aus der Szene als Symbole vorhanden sind, sind semantische Modelle geeignete Methoden, um Szenenelemente des Situationsmodells mit semantischen Informationen anzureichern. Zur semantischen Anreicherung wurden folgende Ansätze in der Literatur vorgeschlagen:

Ulbrich et al. [99, 110] modellierten das Fahrzeugumfeld im urbanen Raum anhand einer Ontologie. Diese Ontologie stellte Szenenobjekte sowie metrische, topologische und semantische Relationen zwischen diesen Objekten in einem Graph dar. Der entstehende Graph wurde für die Situationsinterpretation und die Aktionsprädiktion verwendet.

Von Hülsen et al. und Hummel [111, 112] wurden semantische Beziehungen zwischen Szenenelementen wie z. B. die Spurzugehörigkeit, die relative Position und die Verkehrsregeln mit einer Ontologie und logischen Regeln modelliert, um die Situation an einer Kreuzung zu analysieren. Ein ähnlicher Ansatz wurde von Nienhüser und Zöllner [113] vorgestellt. Die Autoren kombinierten eine Onto-

logie mit logischen Regeln und *MLN*-Modellen, um den Fahrspuren die Verkehrszeichen zuzuordnen, Baustellen zu erkennen und die Relevanz von Ampeln zu schätzen.

Johnson et al. [114] verwendeten einen Szenegraph zur semantischen Suche von Bildern. Der Szenengraph war ein Modell, das Szenenobjekte, ihre Attribute und ihre Relationen enthielt. Aus einer textuellen Anfrage des Benutzers wurde ein Szenengraph generiert. Die Ähnlichkeit zwischen diesem Graph und den Graphen der Bilder der Datenbank wurden mit einem *CRF* geschätzt und ähnliche Bilder wurden zurückgegeben.

Semantische Modelle modellieren das Wissen explizit anhand von Symbolen. Der Vorteil ist, dass die WB für andere Anwendungen leicht wiederverwendbar ist. Darüber hinaus kann die WB von menschlichen Experten mit minimalem Aufwand angepasst werden. Die Komplexität der WB bei komplexen Aufgaben in solchen Methoden ist der Hauptnachteil. Die Modellierung von komplexem Wissen bleibt sehr mühsam, da das Wissen oft manuell generiert wird. Eine komplexe WB führt oft zu einer rechenaufwendigen Inferenz (siehe Abschnitt 2.1.2.3). Darüber hinaus werden Unsicherheiten in den semantischen Modellen nicht betrachtet. Andere Methoden wie PGM, die Unsicherheiten modellieren, sind besser geeignet.

3.6.2 Schätzung der Relevanz von Szenenelementen

Die Relevanz von Szenenelementen bezieht sich auf die die Ziele und Werte des Ego-Fahrzeugs. Die Ziele und Werte des Ego-Fahrzeugs werden durch transiente oder permanente Faktoren wie die aktuelle Route des Ego-Fahrzeugs, die Fahranweisungen eines Operators, die Verhaltenspräferenzen des Fahrzugsnutzers, Verkehrsregeln etc. beeinflusst [44]. Elemente der Szene, die den Weg des Ego-Fahrzeugs potenziell kreuzen können, sind für das Ego-Fahrzeug relevant. Möchte bspw. das Ego-Fahrzeug an einer Kreuzung aufgrund des Routings in einer Straße rechts abbiegen, sind Fußgänger und Fahrradfahrer, die diese Straße überqueren wollen, relevant. Im Gegenteil sind Fahrzeuge, die in der Gegenrichtung durch die Kreuzung fahren, nicht relevant, da sie den Weg des Ego-Fahrzeugs sehr wahrscheinlich nicht kreuzen werden. Die Schätzung der Relevanz ist also äquivalent zur Schätzung der potenziellen Kollisionswahrscheinlichkeit. Die Schätzung der Relevanz liegt außerhalb des Fokus dieser Arbeit und wird daher nicht im Detail erläutert.

3.6.3 Schätzung der Konsistenz von Szenenelementen

Elemente einer Szene sind konsistent, wenn diese Elemente Attribute und Relationen haben, die dem Vorwissen entsprechen. Galleguillos und Belongie [115] stellten Kontextinformationen vor, die bei der Objekterkennung von der menschlichen visuellen Wahrnehmung verwendet werden. Folgende Informationen wurden vorgeschlagen:

1. Der semantische Kontext:
    a. Die Wahrscheinlichkeit des gleichzeitigen Auftretens von Objekten: Bestimmte Objekte treten öfter gemeinsam auf, z. B. Ampeln stehen in der Regel an Kreuzungen.
    b. Die Existenz eines Objekts in einer Szene: Bestimmte Objekte tauchen in bestimmten Szenen auf, z. B. Fußgänger sind mehr an einer Kreuzung in urbanen Räumen zu erkennen als am Straßenrand auf der Autobahn.
2. Der räumliche Kontext:
    a. Die Supportebene: Objekte, die nicht fliegen, stehen üblicherweise auf einer Supportebene, z. B. Fußgänger laufen auf Fußgängerwegen und Fahrzeuge fahren auf der Straße.
    b. Die relative Position von Objekten: Bestimmte Objekte treten oft in der gleichen räumlichen Konfiguration auf, z. B. der Himmel befindet sich oberhalb der Straße.
    c. Die Position eines Objekts in einer Szene: Objektpositionen hängen von der Szene ab.
3. Der Skalierungskontext:
    a. Die relative Größe von Objekten, z. B. LKW sind größer als PKW.
    b. Das Seitenverhältnis: Objekte haben bestimmte Seitenverhältnisse.

Globale Kontextinformationen in [115] wurden als solche beschrieben, die die Abhängigkeit zwischen der Szene und den Szenenelementen modellieren, währen lokale Kontextinformationen die Abhängigkeiten zwischen Szenenelementen erfassen.

In der Literatur wurden oft die o. g. Kontextinformationen direkt in die Modelle zur Objekterkennung und semantischen Segmentierung integriert, um eine kontextkonsistente Klassifikation bzw. Segmentierung zu ermöglichen [116]. Lediglich wenige Autoren adressierten die Schätzung der Konsistenz von Objekten und Regionen.

Bei Choi et al. [116] wurden Kontextinformationen, die ähnlich zu den o. g. Informationen waren, verwendet, um die Konsistenz von Szenen und Objekten zu schätzen. Dafür wurden die Wahrscheinlichkeit des gleichzeitigen Auftretens von Objekten und die Supportebene mit einem PGM-Modell modelliert. Das PGM-Modell besaß eine Baumstruktur und wurde zur Laufzeit verwendet, um die Konsistenz des Outputs eines Klassifikators zu schätzen. Dieses PGM-Modell war in der Lage, ca. 72 % der inkonsistenten Objekte zu erkennen. Die Supportebene spielte bei der Erkennung von Inkonsistenzen die wesentliche Rolle. Die Fehler der Konsistenzschätzung lagen zum großen Teil an dem Mangel an Kontextinformationen, um die Szene verstehen zu können, sowie an den Fehlern der Objektklassifikation.

Ruiz-Sarmiento et al. [117] modellierten Objekteigenschaften wie die Größe, die Orientierung, die Farbe etc. sowie die semantischen und räumlichen Abhängigkeiten zwischen Objekten in einer Ontologie. Danach wurde ein *CRF*-Modell zur semantischen Segmentierung anhand der Ontologie generiert und trainiert. Die semantische Segmentierung wurde mit der Ontologie auf Inkonsistenzen überprüft. Inkonsistenzen wurden von Menschen bestätigt und die Ontologie und/oder das *CRF*-Modell wurden angepasst. Die ersten Ergebnisse des Systems waren zufriedenstellend. Der Hauptnachteil des Systems war die fehlende Unsicherheit bei der Schätzung der Konsistenz mit der Ontologie.

Die Tabelle 3-3 fasst noch einmal die Ansätze zur Erkennung der Konsistenz von Szenenelementen zusammen.

Tabelle 3-3: Zusammenfassung der Ansätze zur Erkennung der Konsistenz von Szenenelementen

Name	**PGM**	**Wissensbasiert**	**Hybrid**
Beschreibung	Modellierung von Kontextabhängigkeiten und Unsicherheiten mit PGM	Modellierung von Kontextabhängigkeiten mit einer WB	Kombination von PGM und WB zur Modellierung von Kontextabhängigkeiten und Unsicherheiten
Pro	- Modellierung von Abhängigkeiten und Unsicherheiten	- symbolisches Modell - Entscheidungsprozess nachvollziehbar	- Pro von PGM und WB - einheitliches Framework
Contra	- Modellierung erfolgt oft manuell - Inferenz kann sehr komplex sein	- Modellierung erfolgt oft manuell - Unsicherheiten werden nicht betrachtet - Inferenz kann sehr komplex sein - Beschränkung auf Symbole	- Modellierung erfolgt oft manuell - Inferenz kann sehr komplex sein - Beschränkung auf Teilaspekte von PGM und WB
Beispiele	- *Out-of-context Objects* [116]	- *Probability and Common-sense* [117]	keine Angabe

3.6.4 Prädiktion der Situationsentwicklung über die Zeit

Ein wichtiger Teil der Situationsinterpretation ist die Prädiktion der Situationsentwicklung über die Zeit. Dabei spielt das aktuelle und zukünftige Verhalten von Verkehrsteilnehmern eine wesentliche Rolle. Das Verhalten von Verkehrsteilnehmern hängt von internen und externen Faktoren ab, die teilweise nicht beobachtbar sind. Interne Faktoren können die semantische Klasse des Objekts, das Ziel, die Motivation und die Absicht des Verkehrsteilnehmers sein. Die Umgebung, die Verkehrsregeln und das Verhalten anderer Objekte sind mögliche externe Faktoren.

Zur Lösung der Fahraufgabe schlugen Evans et al. [118] ein dreistufiges Modell vor: die operative, die taktische und die strategische Ebene. Dieses Modell wird häufig als Grundlage für die Modellierung des Verhaltens von Verkehrsteilnehmern verwendet.

Auf der operativen Ebene wurde das Verhalten der Verkehrsteilnehmer durch kontinuierliche Zustandsgrößen wie Trajektorien dargestellt. Ansätze auf dieser Ebene basierten oft auf dynamischen Modellen wie Kalman-Filter (siehe z. B. [119–121]). Die Kombination von dynamischen Modellen mit PGM wie bspw. *Gaussian Process Dynamical Models* [122] und dynamische Baye'sche Netze [123] wurden auch vorgeschlagen. Ansätze auf der operativen Ebene haben den Vorteil, dass die Modelle einfach sind und die prädizierten Trajektorien ohne weitere Anpassung für die Planung der Trajektorie des Ego-Fahrzeugs verwendet werden können. Die Schwierigkeit, die Abhängigkeit der Trajektorie von Verkehrsteilnehmern von externen Faktoren zu modellieren, bleibt ein Nachteil dieser Ansätze. Daher sind diese Ansätze für die kurzfristige Vorhersage geeignet.

Die taktische Ebene befasst sich mit der mittelfristigen Verhaltensprädiktion. Auf dieser Ebene wurde das Verhalten von Verkehrsteilnehmern durch Manöver modelliert. Die von Nagel [124] vorgeschlagenen Manöver sind in der Literatur weit verbreitet. Ansätze auf dieser Ebene integrieren Kontextinformationen und Faktoren, die das Verhalten von Verkehrsteilnehmern beeinflussen können in PGM, um die Manöver der Verkehrsteilnehmer kontextkonsistent vorherzusagen. Rehder und Kloeden [125] modellierten das Ziel eines Fußgängers als eine latente Variable der Verteilung über mögliche Zustände des Fußgängers. Für die Fahrzeugmanövervorhersage wurden Baye'sche Modelle [126, 127], Gauß'sche Mischungsmodelle [128] und stochastische Multi-Agentensimulation [129] vorgeschlagen. Von Souza und Santos [130] wurde ein manöverbasierter Ansatz verwendet, um die Szene zu analysieren. Das Vorwissen und die Verkehrsregeln wurden mit einem MLN-Modell modelliert. Das MLN-Modell wurde verwendet, um die Position und das Verhalten des Ego-Fahrzeugs zu schätzen. Probabilistische Ansätze können mit Unsicherheiten umgehen und eignen sich für komplexe Situationen, in denen Wahrscheinlichkeitsverteilung aus den Daten gelernt werden können. Im Gegensatz zu probabilistischen Ansätzen modellieren wissensbasierte Ansätze explizit das Vorwissen über das Verhalten von Verkehrsteilnehmern. Nur wenige Autoren verwenden wissensbasierte Ansätze wie Situationsdiagrammbäume [131] und Ontologien mit logischen Regeln [132, 133] für die Prädiktion der Situationsentwicklung. Der Hauptvorteil wissensbasierter Ansätze besteht darin, dass das Verhalten des Systems aufgrund des explizit modellierten Wissens leicht erklärt werden kann. Bekannte Probleme dieser Ansätze sind die Schwierigkeiten, das Vorwissen zu generieren, da dies zum großen Teil manuell erfolgt, und die hohe Inferenz-Komplexität bei komplexen Aufgaben.

Die strategische Ebene ist für die langfristige Verhaltensvorhersage zuständig. Diese Ebene ist nicht Gegenstand dieser Arbeit und wird hier nicht behandelt.

Tabelle 3-4: Zusammenfassung der Ansätze zur Prädiktion der zeitlichen Entwicklung von Situationen

Name	Dynamische Modelle	PGM	Wissensbasiert	Hybrid
Beschreibung	Modellierung des dynamischen Modells eines Objekts auf der operativen	Modellierung von Faktoren, die das Verhalten beeinflussen, und Unsicherheiten mit	Modellierung von Faktoren, die das Verhalten beeinflussen, mit WB auf der taktischen	Kombination von dynamischen Modellen, PGM und WB

Name	**Dynamische Modelle**	**PGM**	**Wissensbasiert**	**Hybrid**
	Ebene	PGM	Ebene	
Pro	- einfache Modelle	- Abhängigkeiten des Verhaltens von internen und externen Faktoren sowie Unsicherheiten modellierbar	- formales Modell ➔ Entscheidungsprozess nachvollziehbar	- Pro von dynamischen Modellen, PGM und WB - einheitliches Framework
Contra	- nur Dynamik, Einfluss durch externe Faktoren nicht betrachtet	- Modellierung erfolgt oft manuell - Inferenz kann sehr komplex sein	- Modellierung erfolgt oft manuell - Unsicherheiten werden nicht betrachtet - Inferenz kann sehr komplex sein - Einschränkung auf Symbole	- Modellierung erfolgt oft manuell - Inferenz kann sehr komplex sein
Beispiele	- *Kalman-Filter* [120, 121]	- Baye'sche Modelle [126] - Gauß'sche Mischungsmodelle [128]	- Situationsdiagrammbäume [131] - Ontologien mit logischen Regeln [132, 133]	- dynamische Baye'sche Netze [123] - *MLN* [130]

3.7 Holistische Betrachtung von Szenen- und Situationselementen

Die holistische Betrachtung von Szenen- und Situationselementen, das heißt die kontextkonsistente Modellierung der Abhängigkeiten zwischen Elementen der Szene und der Situation, ist hilfreich, um komplexere Szenen und Situationen zu verstehen. Experimente der menschlichen visuellen Wahrnehmung zeigten, dass Kontext-informationen eine wichtige Rolle dabei spielten. Die in [115] vorgestellten Kontext-informationen (siehe Abschnitt 3.6.3) wurden hauptsächlich verwendet, um die Objekterkennung und die semantische Segmentierung zu verbessern [116]. Da die Objekterkennung und die semantische Segmentierung Aufgaben der Szenenmodellierung sind und Kontextinformationen oft Relationen, die zum Situationsmodell gehören, kann die Verwendung von Kontextinformationen zur Verbesserung der Objekterkennung und semantischen Segmentierung als eine holistische Betrachtung von Szenen- und Situationselementen verstanden werden. Kontextinformationen können implizit oder explizit modelliert werden.

Bei der impliziten Modellierung von Kontextinformationen werden Kontextinformationen indirekt beim Trainieren des Systems mitgelernt. Diese Art der Modellierung wird oft bei TNN verwendet. Die ersten Faltungsschichten von *CNN* lernen den lokalen Kontext aufgrund des eingeschränkten rezeptiven Felds der Neuronen. Tiefere Faltungsschichten haben wegen des Herunterskalierens der Inputdaten breitere rezeptive Felder und erfassen mehr globale Kontextinformationen. Vollständig vernetzte Schichten haben ein uneingeschränktes rezeptives Feld und können somit den globalen Kontext erfassen [105]. Eine weitere Möglichkeit den Kontext zu lernen, wären TNN, die mehrere Aufgaben gleichzeitig lernen. Wie TNN am Beispiel der semantischen Segmentierung den Kontext verwenden, wurde im Abschnitt 3.5.2 erläutert. Da aber TNN selbst die Kontextinformationen lernen und die

Modelle meistens sehr komplex sind, bleibt es schwierig nachzuvollziehen, welche Kontextinformationen in welcher Form in dem Entscheidungsprozess der TNN Einfluss haben.

Bei der expliziten Modellierung von Kontextinformationen werden Kontextinformationen als zusätzliche Merkmale für die Erfassung von Szenen und Situationen verwendet. Dieser Art der Modellierung von Kontextinformationen wurde für die Objekterkennung [134, 135], die Szenenkategorisierung [136, 137] und die Schätzung der Szenengeometrie [138, 139] verwendet. Dort wurden die Abhängigkeiten, die durch die in [115] vorgeschlagenen Kontextinformation entstanden waren, mithilfe von *CRF* modelliert. Ähnliche Ansätze zur Objektklassifikation wurden von Hensel und Stiller et al. [140, 141] vorgeschlagen. Kontextinformationen aus [115] wurden erweitert, in einer Ontologie modelliert und mit logischen Regeln ergänzt. Um die Unsicherheit zu betrachten, wurden MLN anhand dieser Regeln gelernt. Die gelernten MLN wurden zur Laufzeit verwendet, um gegebene Bildregionen zu klassifizieren. Durch die Integration von expliziten Kontextinformationen wurde die Erfassung von Szenen und Situationen verbessert. Die Erfassung wurde aber auch komplexer. Aufgrund der expliziten Modellierung der Kontextinformationen waren die Einflüsse dieser Kontextinformationen auf die Detektoren im Vergleich zu der impliziten Modellierung von Kontextinformationen besser zu analysieren und zu interpretieren.

Die in [115] vorgeschlagenen Kontextinformationen waren auf die Wahrnehmung einzelner Bilder fokussiert. Der zeitliche Kontext spielt bei der Verarbeitung von Bildsequenzen eine wichtige Rolle und wurde von dynamischen Modellen wie Kalman-Filter bis hin zu komplexeren RNN verwendet.

Kontextinformationen wurden selten verwendet, um die Konsistenz von Szenenelementen zu schätzen. Dies spielt aber eine wichtige Rolle bei Modellen, die keine explizite Modellierung des Kontexts wie TNN ermöglichen. Dort wurde oft der Kontext als *Post-Processing* verwendet, um mit *CRF* den Output von TNN zu verbessern. Die von *CRF* nicht korrigierten Fehler des TNN wurden nicht betrachtet, obwohl diese gute Erkenntnisse für die Verbesserung des TNN liefern könnten.

Kontextinformationen könnten auch nützlich sein, um die Existenz von „verdeckten“ Objekten zu antizipieren. Dies ist ein sehr wichtiger Beitrag zum vorausschauenden Fahren im Kontext des automatisierten Fahrens. Solche Anwendungen wurden ebenfalls selten in der Literatur angesprochen.

3.8 *End-to-end*-Lernen der Steuerungselemente des Ego-Fahrzeugs

Als neuer Ansatz wurde von Bojarski et al. [142] ein *End-to-end*-TNN-Modell zur Prädiktion des Lenkwinkels des Ego-Fahrzeugs vorgeschlagen. Dieses Modell lernte aus den Sensordaten, Steuerwerte für das Ego-Fahrzeug wie den Lenkwinkel zu prädizieren. Dazu entwickelten Bojarski et al. [143] ein Verfahren, um die Regionen des Inputbildes zu markieren, die einen großen Einfluss auf den Output des *End-to-end*-Modells [142] hatten. Die Ergebnisse zeigten, dass das *End-to-end*-Modell Regionen des Bildes verwendete, die den Freiraum des Ego-Fahrzeugs einschränkten, wie Fahrzeuge, Spurmarkierungen und Bordsteine, um den Lenkwinkel des Ego-Fahrzeugs zu prädizieren. In einigen Beispielen waren aber auch Regionen, die aufgrund ihrer Entfernung zum Ego-Fahrzeug keinen Einfluss auf die Entscheidung des Modells haben sollen, als wichtige Regionen markiert worden. Der Hauptvorteil von *End-to-end*-TNN-Modellen war, dass Trainingsdaten einfach zu generieren waren. Hier reichten Inputbilder des Fahrzeugumfelds und die entsprechenden Lenkwinkel aus. Dies war eine enorme Vereinfachung im Vergleich zu Modellen, die zuerst die Szene und Situation erfassten und darauf basierend die Trajektorie des Ego-Fahrzeugs planten. Ob *End-to-end*-TNN-Modelle auch komplexe Situationen im urbanen Raum behandeln können, bleibt eine offene Frage. In [144] zeigten Pei et al., dass das Modell aus [142] die falsche Entscheidung traf bei Änderung der Beleuchtung des Inputbildes oder beim Verdecken einiger Bereiche des Bildes. Ein menschlicher Fahrer würde trotz solcher Änderungen des Bildes die richtige Entscheidung treffen. Solche Fehler von *End-to-end*-TNN-Modellen zeigten, dass solche Modelle noch Verbesserungspotenzial hatten. Hubschneider et al. [145] verwendeten den Output des *End-to-end*-Modells zum Initialisieren der Trajektorienplanung. Der Trajektorienplaner verfeinerte die initialen Trajektorien unter Beachtung der erkannten Objekte. Damit wurde eine kollisionsfreie Trajektorie auch bei einer falschen Initialisierung durch das *End-to-end*-Modell sichergestellt. Shalev-Shwartz et al. [146] schlugen die Verwendung von tiefenverstärk-

tem Lernen (*DRL*: *Deep Reinforcement Learning)* als Modell vor für das Lernen von Fahrstrategien anhand von erkannten Objekten und nicht von Sensordaten wie in [142]. Dabei lernte das *DRL*-Modell implizit, die Entwicklung der Situation zu prädizieren. Die Evaluation der gelernten Fahrstrategien in der Simulation zeigte gute Ergebnisse. Die Autoren betonten aber auch, dass dieser Ansatz nur für eine Komfort-Trajektorienplanung geeignet war, wo es keine klaren Verkehrsregeln gab und es mehr um eine Verhandlung zwischen Verkehrsteilnehmern ging (z. B. im mehrspurigen Kreisverkehr mit fehlenden Spurmarkierungen). Für die endgültige Trajektorienplanung wurde die zeitliche Entwicklung der Situation explizit geschätzt.

3.9 Zusammenfassung

In diesem Kapitel wurden Ansätze aus der Literatur zur Erfassung von Szenen und Situationen vorgestellt.

Im ersten Abschnitt wurde eine Definition der Begriffe Szene und Situation vorgeschlagen. Während die Szene nur eine Momentaufnahme des Umfelds ist, werden auf der Situationsebene die für die Fahraufgabe relevanten Szenenelemente betrachtet. Darüber hinaus werden bestimmte Aspekte der Situation wie die zeitliche Entwicklung oder die Kritikalität der Situation interpretiert.

Im zweiten Abschnitt wurden Sensoren vorgestellt, die im *Automotive*-Bereich verwendet werden. Die Vor- und Nachteile einzelner Sensoren wurden diskutiert. In der Praxis werden mehrere Sensoren zur robusten Erfassung des Umfelds eingesetzt.

Der dritte Abschnitt stellte das *JDL*-Modell und seine Erweiterung zum *PF2*-Modell vor. Diese beiden Modelle schlugen schichtbasierte Architekturen zur Sensordatenfusion vor. Dies bot den Vorteil, dass Teile dieser Architektur mit minimalem Aufwand geändert werden konnten.

Die Objekterkennung und die semantische Segmentierung wurden als wichtigste Aufgaben der Szenenerfassung in den Abschnitten vier und fünf vorgestellt. Für diese beiden Aufgaben lieferten TNN-Modelle die besten Ergebnisse. Der Hauptnachteil von TNN-Modellen bleibt die Schwierigkeit, diese Modelle zu verstehen und zu interpretieren.

Der sechste Abschnitt beschäftigte sich mit den Aufgaben der Situationsmodellierung und –Interpretation. Zur Lösung dieser Aufgaben wurden vor allem PGM und wissensbasierte Modelle verwendet. Diese Modelle hatten den Vorteil, dass sie relevante Abhängigkeiten zwischen den Szenenelementen modellieren konnten. Der Hauptnachteil war der Aufwand bei dem Entwurf dieser Modelle, da der Entwurf oft manuell erfolgte.

Die holistische Betrachtung von Szenen und Situationen, die im Abschnitt sieben diskutiert wurde, ermöglichte eine kontextkonsistente Erfassung von Szenen und Situationen. Hier wurden oft PGM-Modelle verwendet. Holistische Ansätze hatten den Nachteil, dass sie wegen der Modellierung von Kontextabhängigkeiten sehr komplex werden konnten.

End-to-end-Ansätze, die die Steuerungselemente des Fahrzeugs direkt aus den Sensordaten schätzen, wurden im achten Abschnitt kurz erläutert. Diese Ansätze verwendeten TNN und hatten den Vorteil, dass die Trainingsdaten einfach zu gewinnen waren. Allerdings zeigten diese Ansätze ihre Grenze in einfachen *Adversarial*-Beispielen.

Im nächsten Kapitel wird ein Konzept vorgestellt, das sich auf den Stand der Technik bezieht und Lösungen zu einigen der festgestellten Probleme aus der Literatur liefert.

4 Konzept des Systems zur holistischen Szenen- und Situationserfassung

In Kapitel 3 wurden Ansätze vorgestellt, die zur Lösung der Aufgaben der Szenen- und Situationserfassung verwendet wurden. Dabei wurde Folgendes festgestellt:

1. Sowohl für die Objekterkennung als auch für die semantische Segmentierung lieferten TNN die besten Klassifikationsergebnisse. Allerdings waren TNN für die explizite Modellierung von Kontextinformationen, die für eine kontextkonsistente Klassifikation sorgten, nicht geeignet. TNN lernten Kontextinformationen implizit aus den Trainingsdaten. Inwiefern die von TNN gelernten Kontextinformationen zur Klassifikation beitrugen, blieb aufgrund der Komplexität der TNN eine Herausforderung. Die Integration von explizit modellierten Kontextinformationen in den TNN wurde oft mit *CRF* als *Post-Processing* realisiert. *CRF*-Modelle schafften in der Tat, einige Inkonsistenzen der TNN zu korrigieren. Allerdings wurden Informationen über Inkonsistenzen von TNN nicht weiter verwendet. Diese Informationen sind aber sehr wichtig, um den Entscheidungsprozess von TNN-Modellen zu verstehen.
2. Für die Prädiktion der zeitlichen Situationsentwicklung waren PGM die besten Ansätze. Was daran lag, dass die Entwicklung der Situation von vielen Faktoren und Kontextinformationen abhängig war, die zum Teil nicht beobachtbar waren. Darüber hinaus mussten Unsicherheiten der Daten betrachtet werden. PGM waren dafür geeignet, Unsicherheiten und Abhängigkeiten zu modellieren. Allerdings wurden die PGM wegen der vielen Abhängigkeiten sehr komplex, was zur aufwendigen Modellierung und Inferenz führte. Darüber hinaus konnten seltene Ereignisse während der Trainingsphase zum Teil nicht erfasst werden, da diese kaum repräsentativ in den Trainingsdaten waren. Eine künstliche Erweiterung von Trainingsdaten, um die Anzahl der seltenen Ereignisse zu erhöhen, würde dazu führen, dass das System diese Ereignisse nicht mehr als selten betrachtet und während der Inferenz diese fälschlicherweise öfter prädiziert.

In diesem Kapitel wird ein Konzept zur Lösung der Aufgaben der Szenen- und Situationserfassung vorgeschlagen. Das System kombiniert die Vorteile von TNN, PGM und Wissensbasen, um Inkonsistenzen der TNN-basierten semantischen Segmentierung zu erkennen und die zeitliche Entwicklung von Situationen mit seltenen Ereignissen zu prädizieren. Eine frühere Version des vorgeschlagenen Konzepts wurde in [147] publiziert. Der erste Abschnitt dieses Kapitels stellt die in dieser Arbeit vorgeschlagene Lösung vor. Danach wird eine konzeptuelle Sicht des vorgeschlagenen Lösungsansatzes im Abschnitt zwei veranschaulicht. Alle Teile des Konzepts werden ausführlich und detailliert dargestellt. Der dritte Abschnitt fasst dieses Kapitel zusammen.

4.1 Lösungsansatz

Die holistische Modellierung von Szenen und Situationen, das heißt die ganzheitliche Betrachtung von Szenen und Situationen, hilft bei der Lösung der Aufgaben der Szenen- und Situationserfassung. Dieser Idee folgt diese Arbeit.

Das *PF2*-Fusionsmodell, das die Aufgaben der Szenen- und Situationserfassung in Schichten aufteilt, wird in dieser Arbeit übernommen und erweitert. Dieses Modell hatte den Vorteil, dass das System modular aufgebaut wurde. So konnten Teile des Systems angepasst werden, ohne einen großen Einfluss auf das Gesamtsystem zu haben. *End-to-end*-Modelle, die aus Sensordaten Steuerungswerte des Ego-Fahrzeugs schätzen, waren nicht für komplexe Situationen geeignet und deshalb für diese Arbeit nicht passend. Die in dieser Arbeit vorgeschlagene Erweiterung des *PF2*-Fusionsmodells enthält folgende Punkte (siehe Abbildung 4-1):

1. Die Sensor- und Szenenebenen entsprechen den Ebenen 0 und 1 des *PF2*-Modells. Diese Ebenen werden in der Perzeptionsschicht zusammengefasst.
2. Die Situationsebene entspricht den Ebenen 1 und 2 des *PF2*-Modells. Diese Ebenen werden auch in der Entscheidungsschicht zusammengefasst.
3. Eine bidirektionale Kommunikation zwischen der Szenen- und der Situationsebene wird ermöglicht. Damit wird die holistische Modellierung von Szenen und Situationen sichergestellt.

Die Vorteile der holistischen Modellierung von Szenen und Situationen wurde im Abschnitt 3.7 vorgestellt.

4. Unterschiedliche Ansätze, die aus der Literatur die besten Ergebnisse liefern, werden eingesetzt, um Aufgaben der Szenen- und Situationsebene zu lösen. Dies hat den Vorteil, dass einzelnen Aufgaben optimal gelöst werden. Gleichzeitig sollte die Komplexität des Gesamtsystems handhabbar bleiben.
5. Das System wird mit einer WB und PGM ergänzt. Die WB enthält Kontextinformationen über Szenenelemente und Situationsaspekte. Die Modellierung der WB wird explizit von den Modulen, die die WB zur Lösung der Aufgaben der Szenen- und Situationsschichten verwenden, getrennt. Dies hat den Vorteil, dass die WB für unterschiedliche Anwendungen verwendet werden kann. Weiterhin werden Änderungen der WB mit minimalem Aufwand in die Module integriert, die die WB für die Inferenz verwenden. Die WB enthält das Wissen, das mit datenbasierten Ansätzen nur mit viel Aufwand gelernt werden kann. Gleichzeitig sollte dieses Wissen auf der symbolischen Ebene einfach zu modellieren sein. Das Wissen, das Vorteile der formalen Modellierung nutzt, wird ebenfalls in der WB modelliert. Die Komplexität der WB kann somit beherrscht werden. Die WB wird um PGM erweitert, um Unsicherheiten in das Wissen zu integrieren.

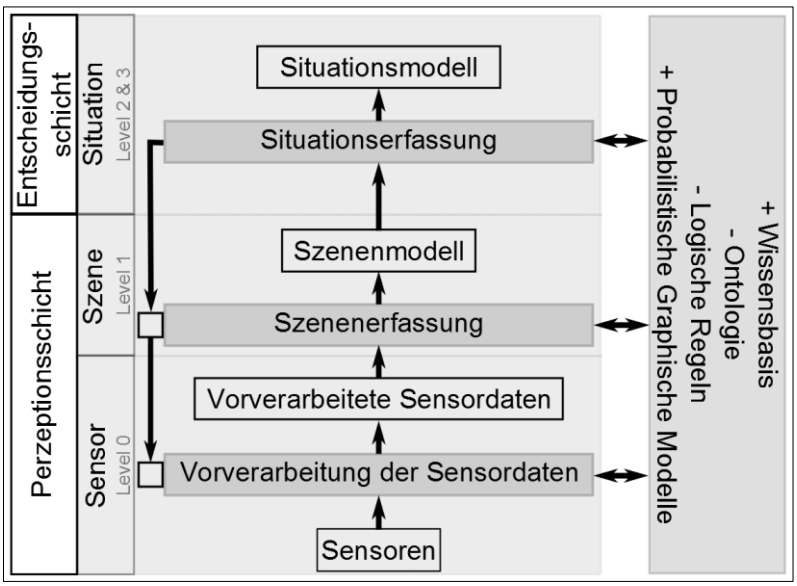


Abbildung 4-1: Erweiterung des PF2-Modells mit einer WB und PGM zur holistischen Szenen- und Situationserfassung

4.2 Konzeptuelle Sicht des Lösungsansatzes

In der Literatur wurden viele Ansätze vorgeschlagen, um die Aufgaben der Szenen- und Situationserfassung zu lösen. Diese Ansätze basierten hauptsächlich auf folgenden grundlegenden Methoden:

1. TNN: TNN arbeiten hauptsächlich auf einer subsymbolischen Ebene und lernen robuste Merkmale aus großen Trainingsdatenmengen. Die darauf basierenden Klassifikatoren sind die besten für die Aufgaben der Szenenerfassung wie die Objekterkennung und die semantische Segmentierung. TNN sind weniger für die Interpretation von Situationsaspekten geeignet.
2. PGM: PGM kombinieren Graphen mit Wahrscheinlichkeiten zur Modellierung von Abhängigkeiten und Unsicherheiten in komplexen Aufgaben. Sie sind für die Interpretation von Situationsaspekten gut geeignet, da Situationsaspekte von vielen Faktoren abhängig sind. PGM können sowohl subsymbolische als auch symbolische Daten verarbeiten.
3. WB: Eine WB modelliert das Wissen einer Domäne explizit anhand von Symbolen und Regeln. Hier werden oft formale Modelle zur Modellierung auf einer symbolischen Ebene verwendet.

Wissensbasierte Ansätze sind für die Interpretation von Situationsaspekten, wo keine Unsicherheiten vorliegen, passend.

Da keine der o. g. Methoden aus der Literatur in der Lage war, alle Aufgaben der Szenen- und Situationserfassung am besten zu lösen, wird in dieser Arbeit ein Konzept vorgeschlagen, das diese Methoden integriert, um von deren Vorteilen zu profitieren. Dabei sollen die Nachteile der Methoden, die in der Literatur beschrieben wurden, reduziert werden. Die Abbildung 4-2 präsentiert das Konzept dieser Arbeit mit drei unterschiedlichen Modulen. Für alle Module wird zwischen einer Offline- und einer Onlinephase unterschieden. Die Hintergrundfarbe der Vierecke codiert die Relevanz dieser Vierecke für diese Arbeit. Blaugefärbte Vierecke werden nur kurz adressiert. Der Fokus dieser Arbeit liegt auf gelb- und insbesondere auf rotgefärbten Vierecken. Diese werden in den nächsten Abschnitten im Detail erläutert.

4.2.1 Wissensbasiertes Modul

Wissensbasierte Module modellieren das Wissen explizit und symbolisch in einer WB. Dieses Wissen kann zur Laufzeit verwendet werden, um neues Wissen zu inferieren. Aus der Literatur waren wissensbasierte Ansätze für die Modellierung und Interpretation der Situation geeignet. Das wissensbasierte Modul dieser Arbeit ist deshalb in der Abbildung 4-2 mehr der Entscheidungs- als der Perzeptionsschicht zugeordnet. Die beiden Phasen des wissensbasierten Moduls werden in den nächsten Abschnitten erläutert.

4.2.1.1 Modellierung von Vorwissen in einer WB

In der Modellierungsphase wird die WB entworfen. Das Wissen wird formal modelliert, um eine maschinelle Verarbeitung zu ermöglichen. Dies geschieht offline. Die Modellierung von Wissensbasen wurde oft in der Literatur angesprochen. Von Hensel [140] und Bohlken und Menzel [148] wurden Ontologien mit logischen Regeln als Lösung zur expliziten Modellierung des Vorwissens vorgeschlagen. Ontologien wurden im Abschnitt 2.1.2 vorgestellt und sind in Kombination mit logischen Regeln ideal zur Wissensmodellierung. Diese Idee wird in dieser Arbeit übernommen. Die Ontologie enthält als Konzepte Elemente der Szene und der Situation, wie die Verkehrsteilnehmer, die Fahrspuren, Verkehrszeichen und Manöver, sowie die Relationen zwischen diesen Elementen. Die logischen Regeln beschreiben die logischen Abhängigkeiten zwischen den Konzepten und Relationen der Ontologie. Damit wird komplexeres Wissen abgebildet. Eine Beispielregel könnte die Tatsache sein, dass alle Fahrzeuge, die auf einer Spur mit einer roten Ampel fahren, als zulässiges Manöver stoppen müssen. Die Integration von Unsicherheiten mithilfe von Wahrscheinlichkeiten in der Ontologie wurde von Yang und Calmet [149] und Carvalho [150] vorgeschlagen. Allerdings wurde die Ontologie sehr komplex, was in dieser Arbeit vermieden werden soll. Das Wissen mit Unsicherheiten wird deshalb in einem anderen Modul modelliert. Dieses Modul wird in den kommenden Abschnitten erläutert. Um die Komplexität der WB handhabbar zu halten, werden folgende Lösungen vorgeschlagen:

1. Die WB wird bezogen auf die Anwendung so abstrakt wie möglich gehalten. Damit wird die WB klein gehalten.
2. Die WB enthält das Wissen, das von den Daten nicht gelernt werden kann oder nur mit großem Aufwand lernbar ist. Dies betrifft vor allem das allgemeine Wissen.
3. Die WB enthält das Wissen, das durch explizite und formale Modellierung einen Vorteil hat. Hier sind vor allem Straßenverkehrsregeln gut aufgehoben. Im Kontext des automatisierten Fahrens soll im optimalen Fall bewiesen werden, dass Verkehrsregeln vom Ego-Fahrzeug nicht verletzt werden. Eine WB der Straßenverkehrsregeln kann diese Anforderungen erfüllen. Eine WB mit Straßenverkehrsregeln wurde bspw. von Shalev-Shwartz et al. [146] vorgeschlagen.
4. Die WB wird hierarchisch aufgebaut. Sie besteht aus einer Metawissensbasis für das domänenübergreifende Wissen und mehreren domänenspezifischen Wissensbasen. Eine ähnliche Hierarchie der WB wurde in [148] vorgeschlagen.

Die WB wird hauptsächlich mithilfe von Expertenwissen hergestellt. Für diese Arbeit stellt diese manuelle Arbeit jedoch keine große Herausforderung dar, da die Komplexität der WB minimal gehalten wird. Verfahren zur automatischen Gewinnung von Wissen aus den Daten werden nicht erörtert, da sie nicht im Fokus dieser Arbeit liegen.

4.2.1.2 Inferenz des wissensbasierten Moduls

Da die o. g. WB ein formales Modell ist, kann sie maschinell verarbeitet werden. Die in dieser Arbeit vorgeschlagene WB ist in Form von *FOL*-Regeln vorhanden und kann somit mit Kalkülen der Prädikatenlogik inferiert werden. Die Inferenz nimmt in der Onlinephase als Input die modellierte WB (siehe Schritt 1 in der Abbildung 4-2) und Evidenzen (siehe Schritt 11, 14 und 15 in der Abbildung 4-2). Evidenzen enthalten erfasste Elemente der Szene und Situation, wie die Verkehrsteilnehmer, die Szenerie und die Relationen zwischen diesen Elementen. Diese kommen sowohl von den PGM- als auch von den TNN-Modulen. Der *Reasoner* prüft zunächst die Konsistenz der WB. Danach inferiert der *Reasoner* die Erfüllbarkeit aller Formeln der WB. Die Outputs des *Reasoners* sind bspw. die Konsistenz der WB und das semantisch angereicherte Situationsmodell (siehe Schritt 16 in der Abbildung 4-2). Inkonsistenzen der WB bezogen auf die Evidenzen werden Experten bereitgestellt, um ggf. die WB anzupassen (siehe Schritt 17 in der Abbildung 4-2). Die Anpassung der WB wird im Abschnitt 4.2.4 erläutert. Methoden der logischen Inferenz wurden im Abschnitt 2.1.1 beschrieben. Um die Komplexität der Inferenz zu reduzieren, wird die WB klein gehalten, da die Inferenzkomplexität exponentiell mit der Größe der Ontologie steigt (siehe Abschnitt 2.1.2.3). Ferner wird die WB hierarchisch aufgebaut, sodass Teile der WB parallel inferiert werden können.

4.2.2 Probabilistisches Modul

Aus dem Stand der Technik wurden PGM sowohl für die Situations- als auch für die Szenenerfassung erfolgreich angewendet. Das probabilistische Modul dieser Arbeit ist in der Abbildung 4-2 deshalb sowohl der Entscheidungs- als auch der Perzeptionsschicht zugeordnet. Darüber hinaus waren PGM in der Literatur sehr gut geeignet, um das Wissen mit Unsicherheiten zu modellieren. Deshalb werden PGM-Modelle in dieser Arbeit hauptsächlich verwendet, um das Vorwissen mit Unsicherheiten zu modellieren. Die nächsten Abschnitte erläutern das probabilistische Modul dieser Arbeit.

4.2.2.1 Modellierung von Vorwissen und Unsicherheiten mit PGM-Modellen

In [25] wurden MLN vorgestellt. Diese Modelle erweitern *FOL*-Regeln mit Unsicherheiten. Für eine gegebene Menge logischer Konstanten sind diese Modelle äquivalent zu der Klasse von PGM namens Markov-Netze. MLN sind vielversprechend und wurden unter anderem von diversen Autoren [113, 140, 141] erfolgreich für die Objekterkennung und Situationsinterpretation angewendet. In dieser Arbeit werden MLN eingesetzt, da MLN ein passendes *Framework* sind, um das Wissen aus der WB, die im Abschnitt 4.2.1 vorgestellt wurde, mit Unsicherheiten zu erweitern. Die Modellierung von MLN findet offline statt. Dafür werden Konzepte und Relationen der WB als Prädikate der MLN verwendet (siehe Schritte 2 in der Abbildung 4-2). Die Formeln der MLN werden dann durch logische Verknüpfungen der Prädikate aus der Ontologie generiert. Darüber hinaus werden Axiome und *FOL*-Regeln der WB in MLN integriert. Da die WB nach der Modellierung auf Konsistenz geprüft wird und die MLN auf der konsistenten WB basieren, sind die MLN auch konsistent. Damit wird sichergestellt, dass das modellierte Vorwissen durchgehend in der WB und in MLN konsistent ist. Eine Formel würde bspw. das Wissen über die räumliche Anordnung von Regionen in natürlichen Bildern wie folgt beschreiben: Seien zwei Regionen der Klassen *Straße* und *Himmel* im Bild, dann wird sehr wahrscheinlich die Region *Himmel oberhalb* der Region *Straße* sein. Weitere Formeln zur Beschreibung des allgemeinen Wissens über die zeitliche Situationsentwicklung werden auch in den MLN modelliert. Anders als in [149] und [150] wird in dieser Arbeit die WB explizit von Wissen mit Unsicherheiten getrennt, da die Komplexität der WB sonst steigen würde. Ähnlich wie bei der Modellierung der WB werden bei MLN zur Verringerung der Komplexität folgende Maßnahmen getroffen:

1. MLN werden bezogen auf die Anwendung so abstrakt wie möglich gehalten, um ihre Komplexität minimal zu halten.

2. MLN enthalten das Vorwissen, das nicht von den Daten gelernt werden kann oder nur mit großem Aufwand lernbar ist. Dies betrifft vor allem das allgemeine Vorwissen über seltene Ereignisse.
3. MLN enthalten nur das Wissen, das Unsicherheiten beinhaltet.
4. MLN werden hierarchisch aufgebaut. Meta-MLN modellieren das domänenübergreifende Wissen. Kleine MLN sind für das domänenspezifische Wissen zuständig.

MLN werden wie Wissensbasen hauptsächlich mithilfe von Expertenwissen hergestellt. Da die WB und die MLN in dieser Arbeit das minimale Wissen modellieren sollten, ist die manuelle Generierung von MLN mit minimalem Aufwand verbunden. Verfahren zur automatischen Gewinnung von MLN werden nicht betrachtet. Eine detaillierte Beschreibung von MLN ist im Abschnitt 2.2.2 zu finden.

4.2.2.2 Lernen von Parametern von PGM-Modellen

Die Struktur und die Gewichtungen von MLN können gelernt werden. In dieser Arbeit werden lediglich die Gewichtungen von modellierten MLN gelernt. Die Struktur der MLN wird während der Modellierung festgelegt. Das Lernen von Gewichtungen nimmt als Input die modellierten MLN-*FOL*-Regeln (siehe Schritt 3 in der Abbildung 4-2) und Trainingsdaten. Trainingsdaten sind eine Menge von logischen Konstanten und Belegungen, die die Wahrheitswerte der Prädikate der modellierten MLN-*FOL*-Regeln enthalten (siehe Schritt 4 in der Abbildung 4-2). Verfahren zum Lernen der Gewichtung von MLN-*FOL*-Regeln wurden im Abschnitt 2.3.2 beschrieben und werden in dieser Arbeit übernommen. Die gelernten Gewichtungen der Formeln von MLN können wie folgt interpretiert werden: Formeln, die für die Belegungen der Trainingsdaten öfter erfüllt wurden, bekommen positive Gewichtungen. Je öfter eine Formel in den Trainingsdaten erfüllt wurde, desto größer wird ihre Gewichtung sein. Formeln, die öfter in den Trainingsdaten nicht erfüllt wurden, werden negativ gewichtet.

4.2.2.3 Inferenzalgorithmen für PGM-Modelle

Die Inferenz von MLN benötigt als Input trainierte MLN und Evidenzen. Trainierte MLN werden offline generiert und bereitgestellt (siehe Schritt 5 in der Abbildung 4-2). Die Evidenzen sind bspw. erfasste Elemente der Szene wie Regionen der semantischen Segmentierung, erkannte Verkehrsteilnehmer sowie räumliche und zeitlichen Relationen zwischen diesen Elementen. Die Evidenzen werden im Schritt 12 in der Abbildung 4-2 bereitgestellt. Das Ziel der Inferenz ist die Schätzung der Wahrscheinlichkeit, dass Formeln der MLN erfüllt werden, gegeben die trainierten MLN und die Evidenzen. Dazu wird neben der Wahrscheinlichkeit der Erfüllbarkeit von Formeln auch den Wahrheitswert der Formeln geschätzt. Approximationsalgorithmen zur Inferenz von MLN wurden in den Abschnitten 2.2.3 und 2.3.3 vorgestellt und werden in dieser Arbeit verwendet. Die Inferenz findet online statt.

Ein Output der Inferenz ist bspw. die Konsistenz von Regionen, die von der semantischen Segmentierung generiert wurde, gegeben ihre semantischen Klassen, ihre räumlichen Relationen und die trainierten MLN, die das Vorwissen über die räumlichen Relationen von Regionen in Bildern modellieren. Ein weiterer Output ist die Prädiktion der zeitlichen Entwicklung der Situation, gegeben erkannte Objekte, ihre Relationen und die trainierten MLN. Diese trainierten MLN enthalten das Vorwissen über die potenziellen zeitlichen Entwicklungen der Situation. Der Schritt 13 in der Abbildung 4-2 stellt die Outputs dieser Inferenz dar.

Da MLN auf logischen Regeln basieren, ist es möglich, mit einem geringen Aufwand die Outputs der Inferenz zu erklären. Dies ist ein Vorteil im Vergleich zu anderen Modellen wie TNN, die Outputs haben, die sich nur teilweise und mit großem Aufwand erklären lassen.

Die Komplexität der Inferenz von MLN steigt exponentiell mit der Größe der MLN-Modelle und der Anzahl der Variablen, die inferiert werden sollen auf (siehe Abschnitt 2.3.3). Um die Inferenzkomplexität zu reduzieren, sollen MLN so wenige Klausel wie möglich haben. Eine Hierarchie von MLN, die parallel inferiert werden können, trägt dazu bei, die Komplexität der Inferenz zu senken.

Inkonsistente Regionen der semantischen Segmentierung, die von MLN inferiert wurden, sowie Unstimmigkeiten zwischen der MLN-Inferenz und die Evidenzen werden Experten zur Verfügung gestellt, um ggf. das System anzupassen. Dieses Verfahren wird im Abschnitt 4.2.4 beschrieben und entspricht dem Schritt 19 der Abbildung 4-2.

4.2.3 TNN-basiertes Modul

Ansätze, die auf TNN basieren, lernen die Funktionen, die die Inputdaten auf die Outputs abbilden, anhand von Trainingsdaten. Die Inputdaten sind meistens subsymbolische Daten wie bspw. Kamerabilder. TNN sind besonders gut bei der Objekterkennung und der semantischen Segmentierung. Deshalb ist das TNN-basierte Modul dieser Arbeit in der Abbildung 4-2 mehr der Perzeptions- als der Entscheidungsschicht zugeordnet. In den nächsten Abschnitten werden die Schritte zur Modellierung, zum Training und zur Inferenz von TNN erläutert.

4.2.3.1 Modellierung

Viele TNN-Architekturen wie *VGGNet* [59] und *ResNet* [61] wurden in der Literatur vorgeschlagen. Es ist üblich, eine dieser Architekturen auszuwählen und bei Bedarf anzupassen. Für die Objekterkennung und die semantische Segmentierung wurden einige TNN-Beispiele in den Abschnitten 3.4 und 3.5 vorgestellt. Diese TNN-Modelle werden in dieser Arbeit übernommen. Die Komplexität der TNN-Modelle hängt vor allem von der Anzahl der Parameter der Modelle ab (siehe Abschnitt 2.4.4). Diese Parameter hängen wiederum von der Tiefe der Modelle, den Typen der Schichten und von den rezeptiven Feldern der Neuronen ab. Diese Faktoren werden bei der Auswahl der Modelle berücksichtigt.

Welche Inputs und Merkmale die Modelle benötigen, hängt von den Modellen und von den Anwendungen ab. Inputdaten für *CNN*-Modelle sollen strukturierte Daten sein. Farben- und Grauwertbilder, Tiefenkarten und Bildsequenzen sind Beispiele von Inputdaten, die oft für *CNN*-Modelle verwendet werden. Merkmale werden von TNN in der Trainingsphase gelernt. Methoden zur Integration von manuell generierten Kontextinformationen als Merkmale in den TNN werden hier nicht adressiert. Der Grund ist, dass TNN Merkmale aus den Daten lernen sollen. Die von TNN gelernten Merkmale sind der Hauptgrund für die Erfolge von TNN. Weitere manuell generierte Merkmale als Kontextinformationen in den TNN zu integrieren, würde also der Idee der TNN widersprechen. Darüber hinaus bleiben Verfahren, um den Einfluss von manuell generierten Merkmalen auf die Entscheidung der TNN zu untersuchen, sehr aufwendig und komplex. Das ist der Grund, warum die Mehrheit der Ansätze, die in der Literatur vorgeschlagen wurden, um die Kontextinformationen in TNN zu integrieren, keine manuell generierten Merkmale integriert (siehe Abschnitt 3.5.3). Im Gegenteil dazu ist die Modellierung von Kontextinformation mit Wissensbasen und PGM besser geeignet, da der Einfluss dieser Informationen auf die Modelle nachvollziehbar ist.

4.2.3.2 Lernen

Das Lernen der Parameter von TNN erfolgt mit dem Fehler-Rückübertragungsalgorithmus, wobei die Struktur der TNN in der Modellierungsphase festgelegt wurde (siehe Schritt sechs in der Abbildung 4-2). Dieser Algorithmus setzt voraus, dass Trainingsdaten mit den entsprechenden *Ground-Truth*-Daten vorhanden sind. Darüber hinaus benötigen TNN sehr viele Trainingsdaten. Die Generierung von *Ground-Truth-Daten* erfolgt hauptsächlich manuell und kann deshalb sehr aufwendig werden. Allerdings existieren bereits Trainingsdaten für die Objekterkennung und die semantische Segmentierung. Beispiele sind *PASCAL VOC2011* [76], *PASCAL VOC2012* [77], *Mapillary Vistas Dataset* [151], *DCS*-Datensatz [81] und *KITTI Dataset* [152]. Der *DCS*-Datensatz [81] wurde mit dem Fokus auf das automatisierte Fahren entwickelt und stellt alle Informationen bereit, die für diese Arbeit relevant sind. Dieser Datensatz wird in dieser Arbeit verwendet, um die TNN zu trainieren (siehe Schritt sieben in der Abbildung 4-2).

Verfahren zur besseren Generalisierung von TNN, wie das *Dropout*, die frühzeitige Unterbrechung des Trainings, die Gewichtungsstrafe und die *Batch*-Normalisierung, werden bei Bedarf angewendet.

Das offline trainierte Modell wird für die Inferenz online verwendet. Weitere Informationen zum Lernen der Parameter von TNN sind im Abschnitt 2.4.3 zu finden.

4.2.3.3 Inferenz

Die Inferenz von TNN bildet die Inputdaten auf die Outputs in der Onlinephase ab, wobei die Abbildungsfunktion in der Trainingsphase gelernt wurde. Die Outputs von TNN können bspw. klassifizierte *Bounding*-Boxen oder Pixel sein (Schritt zehn in der Abbildung 4-2). Die Outputs der TNN liefern Evidenzen für die PGM (Schritt zwölf in der Abbildung 4-2) und die WB (Schritt elf in der Abbildung 4-2). Darüber hinaus werden TNN-Outputs, die bezogen auf die Outputs der PGM- und WB-Module inkonsistent sind, bereitgestellt, um das Gesamtsystem ggf. anzupassen (siehe Abschnitt 4.2.4).

4.2.4 Umgang mit Inkonsistenzen und Unstimmigkeiten zwischen den Modulen

Ein weiterer Aspekt dieser Arbeit ist die Entwicklung von Konzepten zur systematischen Handhabung von Inkonsistenzen und Unstimmigkeiten. Inkonsistenzen und Unstimmigkeiten treten auf, wenn die Evidenzen dem Vorwissen widersprechen. Treten solche Fälle auf, gibt es drei mögliche Erklärungen:

1. Die Evidenzen sind fehlerbehaftet und das Vorwissen ist korrekt. In diesem Fall sollen die Modelle zur Generierung von Evidenzen angepasst werden.
2. Die Evidenzen sind korrekt und das Vorwissen ist fehlerbehaftet. In diesem Fall sollen die Modelle zur Modellierung von Vorwissen angepasst werden.
3. Die Evidenzen und das Vorwissen sind fehlerbehaftet. In diesem Fall sollen die Modelle zur Generierung von Evidenzen und Vorwissen angepasst werden.

Eine weitere Quelle für Inkonsistenzen liegt vor, wenn mehrere Module des Systems, die denselben Output generieren, sich widersprechen. Hier soll festgestellt werden, welche Module recht haben, um die anderen Module anzupassen.

Zur Handhabung von Inkonsistenzen und Unstimmigkeiten der Module werden in der Onlinephase Unstimmigkeiten zwischen dem wissensbasierten Modul, dem probabilistischen Modul und dem TNN-basierten Modul sowie Inkonsistenzen zwischen den Evidenzen und dem Vorwissen gespeichert. Diese Inkonsistenzen und Unstimmigkeiten werden dann in der Offlinephase von Experten bewertet, die WB, die PGM und die TNN werden entsprechend angepasst, neue trainiert, getestet und validiert, bevor sie für Onlineanwendungen verfügbar gemacht werden (siehe Schritte 17 bis 24 in der Abbildung 4-2). Diese Vorgehensweise ist im Bereich von sicherheitskritischen Systemen wie dem automatisierten Fahren wichtig, da so sichergestellt wird, dass die verwendeten Modelle korrekt und konsistent sind.

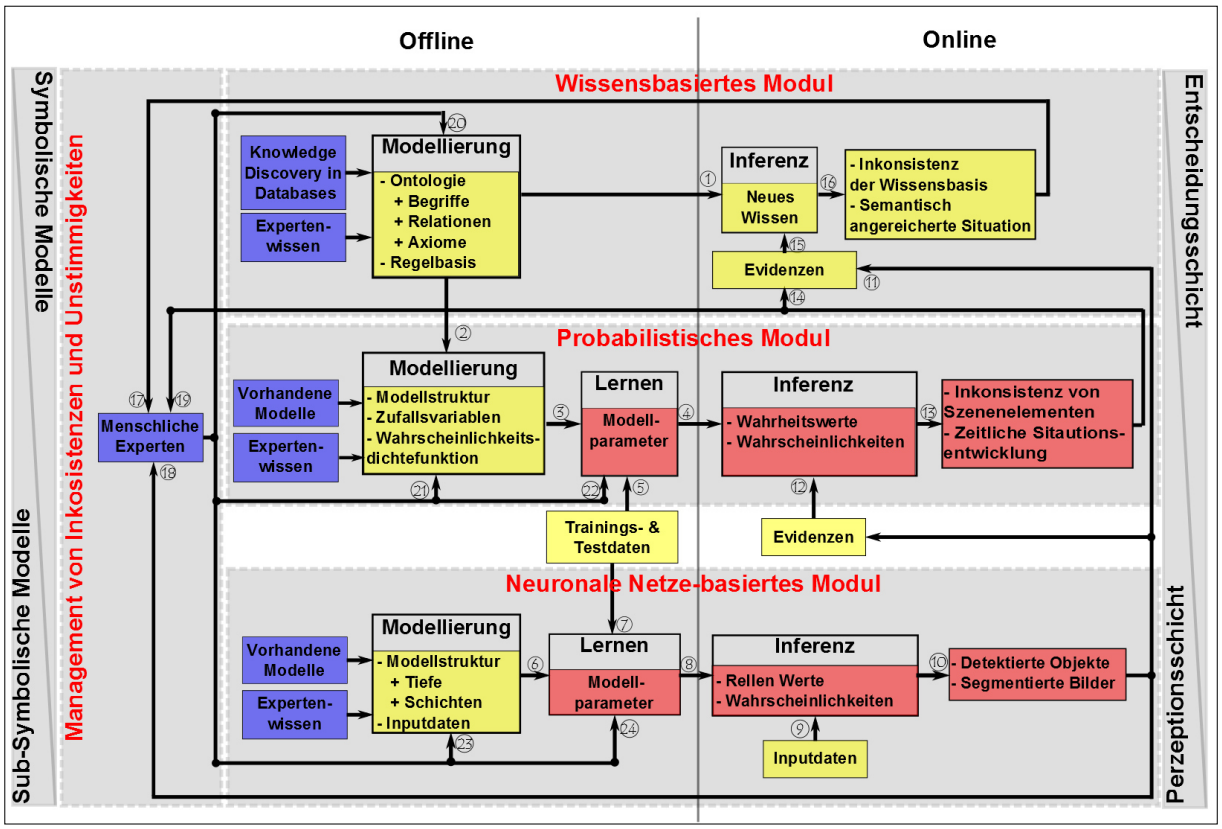


Abbildung 4-2: Konzeptuelle Übersicht des Systems zur holistischen Modellierung und Interpretation von Szenen und Situationen

4.3 Zusammenfassung

In diesem Kapitel wurde ein Lösungsansatz zu den Aufgaben der Szenen- und Situationserfassung vorgeschlagen. Dieser Ansatz basiert auf dem weitverbreiteten *PF2*-Datenfusionsmodell. Das *PF2*-Datenfusionsmodell wurde um ein *Top-down*-Verfahren erweitert, um eine holistische Modellierung von Szenen und Situationen zu ermöglichen. Dies hat den Vorteil, dass die Szenen- und Situationserfassung verbessert wird. Weiterhin wurde das *PF2*-Modell mit einer WB und PGM erweitert. Die WB enthielt Kontextinformationen, die für die holistische Betrachtung von Szenen und Situationen relevant waren. Die WB wurde mit PGM erweitert, um das Vorwissen mit Unsicherheiten zu modellieren. Das Vorwissen wurde explizit von den Modulen der Szenen- und Situationserfassung getrennt, damit Änderungen des Vorwissens mit minimalem Aufwand in die Module der Szenen- und Situationserfassung integriert werden können.

Aus dem Lösungsansatz wurde ein Konzept vorgeschlagen. Dieses Konzept basierte auf der Kombination folgender Methoden: WB, PGM und TNN. Diese Methoden wurden erfolgreich in der Literatur verwendet, um einzelne Aufgaben der Szenen- und Situationserfassung zu lösen. Allerdings wurde keine dieser Methode verwendet, um alle Aufgaben der Szenen- und Situationserfassung zu lösen. Das Konzept nutzte also die Vorteile dieser Methoden für die Aufgaben, die von diesen Methoden am besten gelöst wurden, und schlug Lösungen vor, um die Nachteile der Methoden zu mindern. Das System bestand deshalb aus einem wissensbasierten Modul, einem probabilistischen Modul und einem TNN-basierten Modul. Diese Module wurden Aufgaben zugordnet, die sie gemäß der Literatur am besten lösten. Alle Module wurden in einer Offlinephase modelliert und ggf. gelernt. In der Onlinephase wurden die offline gelernten Modelle zur Inferenz verwendet. Lösungen für die Probleme der Modellierung, des Lernens und der Inferenz wurden vorgeschlagen. Weiterhin wurde ein Konzept zum Umgang mit Inkonsistenzen zwischen dem Vorwissen und den Evidenzen sowie mit Unstimmigkeiten zwischen den Outputs der Module vorgeschlagen. Diese Inkonsistenzen und Unstimmigkeiten wurden durch Experten offline bewertet. Module wurden bei Bedarf angepasst und validiert, bevor

diese wieder für die Onlinephase verwendet wurden. Das vorgeschlagene System hatte folgende Vorteile:

1. Die gleichzeitige Verarbeitung von symbolischen und subsymbolischen Daten.
2. Die Integration von Vorwissen mit Unsicherheiten.
3. Teile des Systems mit expliziter und symbolischer Modellierung konnten einfach erklärt und inhaltlich erweitert werden.
4. Das System ließ sich aufgrund des modularen Aufbaus softwaretechnisch einfacher erweitern.
5. Durch die Erkennung und die Handhabung von Inkonsistenzen und Unstimmigkeiten des Systems wurde die Korrektheit des Systems garantiert.

In den nächsten Kapiteln wird das hier vorgeschlagene Konzept verwendet, um zwei konkrete Aufgaben der Szenen- und Situationserfassung zu lösen. Das Konzept wird prototypisch implementiert, getestet und validiert.

5 Semantische Segmentierung von Kamerabildern mit TNN

In Kapitel 4 wurde ein Konzept für die Lösung der Aufgaben der Szenen- und Situationserfassung vorgeschlagen. Dieses Konzept schlug drei Module vor, die auf Wissensbasen, PGM und TNN basierten. In diesem Kapitel wird die Umsetzung dieses Konzepts für die semantische Segmentierung vorgestellt. Der Fokus liegt auf der Verwendung von Kontextinformation, um die semantische Segmentierung zu evaluieren. Damit wird ein Beitrag zur holistischen Erfassung von Szenen und Situationen geleistet, da die hier verwendeten Kontextinformationen Elemente der Situation sind während die semantische Segmentierung neben der Objekterkennung eine der wichtigsten Aufgabe der Szenenerfassung ist. Im ersten Abschnitt dieses Kapitels wird die Aufgabe der semantischen Segmentierung im Detail vorgestellt. Die Anwendung von TNN zur semantischen Segmentierung wird im zweiten Abschnitt erläutert. Im dritten Abschnitt wird das TNN-Modell namens *FCN-8s*-Modell, das als Meilenstein für die semantische Segmentierung betrachtet wird, veranschaulicht. Der vierte Abschnitt beschäftigt sich mit dem Training des *FCN-8s*-Modells. Die Inferenz mit dem *FCN-8s*-Modell wird im Abschnitt fünf adressiert. Die quantitative Evaluation des trainierten *FCN-8s*-Modells wird im sechsten Abschnitt präsentiert. Bei der Evaluation wird der Einfluss von Kontextinformationen auf die Evaluationsergebnisse demonstriert. Die in diesem Kapitel gewonnenen Ergebnisse werden im siebten Abschnitt diskutiert. Die Zusammenfassung des Kapitels findet im achten Abschnitt statt.

5.1 Vorstellung der Aufgabe

Gegeben sei ein Bild $I \in \mathbb{R}^{w \times h \times c}$ mit der Breite w, der Höhe h und der Anzahl der Kanäle c sowie $n \in \mathbb{N}$ die Anzahl der semantischen Klassen. Die semantische Segmentierung ist eine Funktion $f_\Theta(I) = I_s$, die das Inputbild I auf das semantisch segmentierte Bild $I_s \in \mathbb{N}^{w \times h}$ abbildet. Jeder Pixel des Bildes I_s enthält die semantische Klasse des Pixels ($I_s(i,j) \in \{0, \dots, n-1\}$). Die Abbildung 5-1 stellt die semantische Segmentierung anhand einer Szene im urbanen Raum aus dem *DCS*-Datensatz [81] exemplarisch dar. Das Inputbild I ist ein RGB-Farbenbild. Das segmentierte Bild I_s codiert die semantische Klasse der Pixel mithilfe von Farben in RGB-Format.

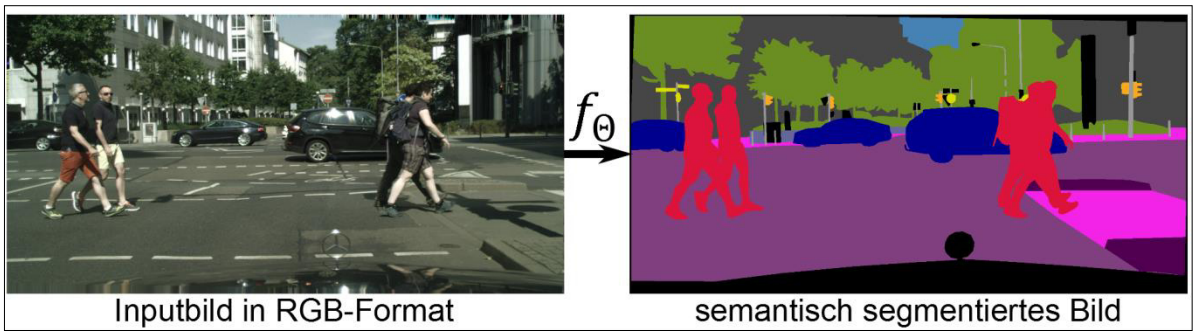


Abbildung 5-1: Darstellung der semantischen Segmentierung von Bildern anhand eines Beispiels an einer Kreuzung im urbanen Raum. Das Inputbild (links) wird mit der Funktion f_Θ auf das semantisch segmentierte Bild (rechts) abgebildet. (Bilder aus [81])

5.2 Semantische Segmentierung mit TNN

In Abschnitt 3.5 wurden unterschiedliche Verfahren zur semantischen Segmentierung vorgestellt. Das von Long et al. [75] vorgestellte *FCN-8s*-Modell war ein Meilenstein für die semantische Segmentierung mit TNN. Neue Modelle, die in der Literatur vorgestellt wurden, basierten auf der Idee des *FCN-8s*-Modells. Das *FCN-8s*-Modell wir deshalb in dieser Arbeit verwendet. Die Modellierung, das Training, das Testen und die Evaluation des *FCN-8s*-Modells wird in den nächsten Abschnitten erläutert (siehe Schritte sechs bis zehn des TNN-basierten Moduls der Abbildung 4-2).

5.3 Beschreibung des *FCN-8s*-Modells

Die Abbildung 5-2 stellt die Architektur des *FCN-8s*-Modells dar. Wie in Abschnitt 3.5.2.1 beschrieben, verwendet das *FCN-8s*-Modell als *Encoder* das *VGG-16*-Netz [59], wobei die drei letzten vollständig vernetzten Schichten durch Faltungsschichten ersetzt wurden (siehe *Encoder* in der Abbil-

dung 5-2). Der *Encoder* besteht insgesamt aus 16 Faltungsschichten, die in der Abbildung 5-2 rot markiert sind. *<m>x Conv<n>x<n>x<c> + ReLU* steht für m-Faltungsschichten mit $n \times n$-Filterkernen, wobei jede Schicht *c*-Faltungsfilter hat und eine *ReLU*-Einheit als Aktivierungsfunktion besitzt. Analog steht *<m> x Pool<n>x<n>* für m-*Pooling*-Schichten mit $n \times n$-Filterkernen. Der *Encoder* besitzt vier davon. Die Aktivierungskarte der letzten Faltungsschicht des *Encoders* wird auch *Score*-Karte genannt. Diese *Score*-Karte wird vom *Decoder* in drei Entfaltungsschichten mit jeweils 4×4, 4×4 und 16×16 Filterkernen auf die Größe des Inputbildes hochskaliert. Der Output der letzten *Pooling*-Schicht des *Encoders* wird mit einer Faltungsschicht mit 20 Filterkernen der Größe 1×1 gefaltet und elementweise mit dem Output der ersten Entfaltungsschicht des *Decoders* addiert. Analog dazu wird der Output der vorletzten *Pooling*-Schicht des *Encoders* mit einer Faltungsschicht mit 20 1×1-Filterkernen gefaltet und elementweise mit dem Output der zweiten Entfaltungsschicht des *Decoders* addiert. Damit wird die Segmentierung mit detailreichen Informationen ergänzt. Die letzte Schicht des *Decoders* berechnet die Wahrscheinlichkeit der Klassenzugehörigkeit jedes Pixels mit der *Softmax*-Funktion. Die letzte Faltungsschicht des *Encoders* sowie alle Schichten des *Decoders* besitzen jeweils 20 Filter. Die Zahl 20 entspricht der Anzahl der semantischen Klassen, die segmentiert werden sollen. Durch die Auswahl der Schichten des *Encoders* und des *Decoders* können der *Encoder* und der *Decoder* gleichzeitig trainiert werden, was ein Vorteil ist.

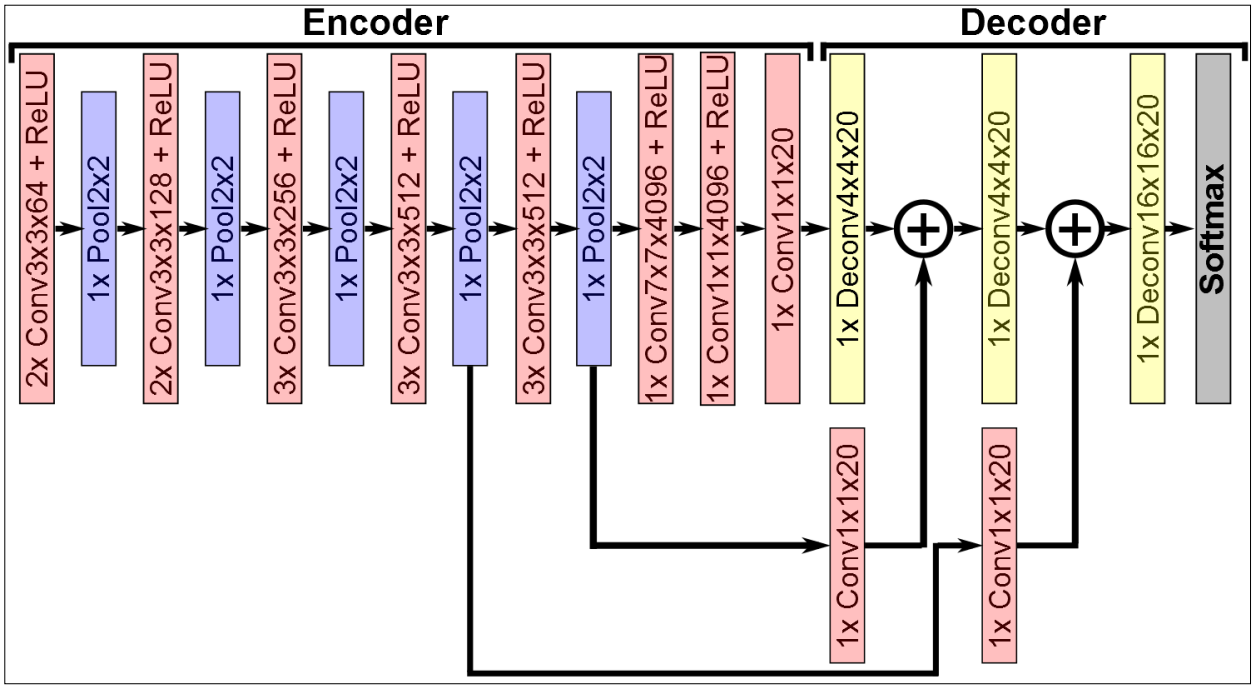


Abbildung 5-2: Architektur des FCN-8s-Modells [75] mit dem VGG-16-Modell [59] als Encoder. Convnxnxc steht für eine Faltungsschicht mit c-Faltungsfiltern, wobei jeder Filterkern die Dimension n x n hat. Poolnxn steht für eine Pooling-Schicht mit nxn-Filterkern. Deconvnxnxc steht für eine Entfaltungsschicht mit c-Entfaltungsfiltern und jeweils nxn-Filterkern.

5.4 Training des *FCN-8s*-Modells

Zum Lernen der Parameter des *FCN-8s*-Modells sollen zu den Inputbildern die passenden annotierten Bilder vorhanden sein. Die Generierung der annotierten Bilder ist mit einem großen Aufwand verbunden. In dieser Arbeit wurde deshalb zum Trainieren der online verfügbare *DCS*-Datensatz [81] verwendet.

5.4.1 Beschreibung des *DCS*-Datensatzes

Der *DCS*-Datensatz [81] besteht aus 5000 RGB-Bildern und den dazu passenden fein-annotierten Bildern. Zusätzlich werden 20 000 RGB-Bilder mit grob-annotierten Annotationen bereitgestellt. Dieser Datensatz wurde in 50 Städten in Deutschland und Zürich aufgenommen. Die Daten enthalten 35 semantische Klassen. Die Klassen werden in acht Kategorien gruppiert. Die Klassen und Kategorien

sind in der Tabelle 5-1 dargestellt. Klassen mit Trainings-ID 255 werden im Training und in der Evaluation ignoriert. Die RGB-Werte zur Farbendarstellung der Klassen sind in den drei letzten Spalten zu finden.

Tabelle 5-1: Tabellarische Darstellung der Klassen und Kategorien des DCS-Datensatzes sowie weitere Metainformationen

Klassenname	**Klassen-ID**	**Training-ID**	**Kategoriename**	**Kategorie-ID**	**R**	**G**	**B**
unlabeled	0	255	void	0	0	0	0
ego_vehicle	1	255	void	0	0	0	0
rectification_border	2	255	void	0	0	0	0
out_of_roi	3	255	void	0	0	0	0
static	4	255	void	0	0	0	0
dynamic	5	255	void	0	111	74	0
ground	6	255	void	0	81	0	81
road	7	0	flat	1	128	64	128
sidewalk	8	1	flat	1	244	35	232
parking	9	255	flat	1	250	170	160
rail_track	10	255	flat	1	230	150	140
building	11	2	construction	2	70	70	70
wall	12	3	construction	2	102	102	156
fence	13	4	construction	2	190	153	153
guard_rail	14	255	construction	2	180	165	180
bridge	15	255	construction	2	150	100	100
tunnel	16	255	construction	2	150	120	90
pole	17	5	object	3	153	153	153
polegroup	18	255	object	3	153	153	153
traffic_light	19	6	object	3	250	170	30
traffic_sign	20	7	object	3	220	220	0
vegetation	21	8	nature	4	107	142	35
terrain	22	9	nature	4	152	251	152
sky	23	10	sky	5	70	130	180
person	24	11	human	6	220	20	60
rider	25	12	human	6	255	0	0
car	26	13	vehicle	7	0	0	142
truck	27	14	vehicle	7	0	0	70
bus	28	15	vehicle	7	0	60	100
caravan	29	255	vehicle	7	0	0	90

Klassenname	Klas-sen-ID	Trai-ning-ID	Kategoriename	Kategorie-ID	R	G	B
trailer	30	255	vehicle	7	0	0	110
train	31	16	vehicle	7	0	80	100
motorcycle	32	17	vehicle	7	0	0	230
bicycle	33	18	vehicle	7	119	11	32
license_plate	-1	-1	vehicle	7	0	0	142

Die Abbildung 5-3 stellt die Verteilung der Klassen für die 5000 fein-annotierten Bilder dar. Aus dieser Abbildung lässt sich feststellen, dass die Klassen *road, sidewalk, vegetation, building* und *car* am häufigsten in den Trainingsdaten vorkommen. Die vulnerablen Verkehrsteilnehmer (*person* und *rider)*, die im Kontext des automatisierten Fahrens besonders geschützt werden sollen, sind ausreichend in dem Datensatz repräsentiert. Damit ist der *DCS*-Datensatz geeignet, um Modelle für die Wahrnehmung im urbanen Raum zu trainieren. Klassen, die beim Training und bei der Evaluation ignoriert werden, sind mit *2* markiert. Diese Klassen werden ignoriert, da diese nicht eindeutig zu einer Klassen beim Generieren der Trainingsdaten zugewiesen werden konnten [81].

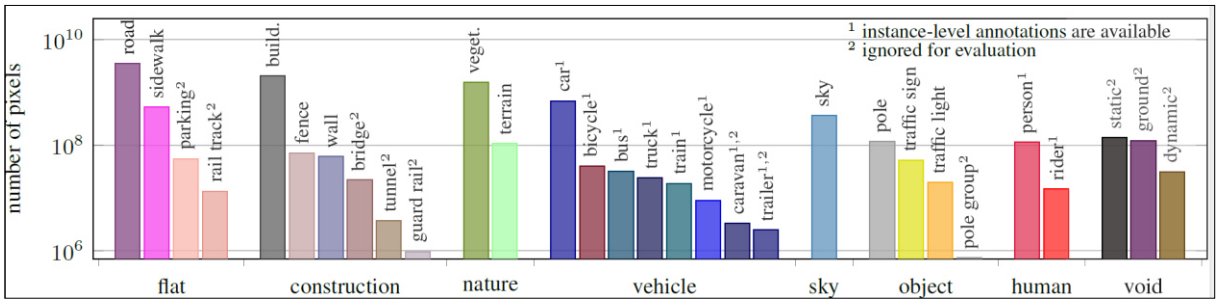


Abbildung 5-3: Klassenverteilung für fein-annotierte Bilder des DCS-Datensatzes und die zu den Klassen passenden Kategorien (Bild aus [81])

Der *DCS*-Datensatz ist in drei Datensätze aufgeteilt: den Trainings-, den Test- und den Validierungsdatensatz. Der Trainingsdatensatz wird für das Training der Segmentierungsmodelle verwendet. 2975 der 5000 fein-annotierten Bilder sind in diesem Datensatz zu finden. Der Validierungsdatensatz wird für die Kreuz-Validierung der trainierten Modelle verwendet. Davon sind 500 Bilder verfügbar. Zu den Trainings- und Validierungsdatensätzen werden auch die Annotationen bereitgestellt. 1525 der 5000 fein-annotierten Bilder sind in dem Testdatensatz zu finden. Zu den Testdaten sind die *Ground-Truth*-Daten der semantischen Segmentierung nicht verfügbar. In dieser Arbeit wurden lediglich die 5000 fein-annotierten Bilder verwendet. Darüber hinaus wurden die Klassen, die beim Training und bei der Evaluation ignoriert werden sollten, in der Klasse *unknow* zusammengefasst. Damit ergaben sich insgesamt 20 semantische Klassen.

5.4.2 Training des *FCN-8s*-Modells

Das Training der semantischen Segmentierung hat als Ziel die Minimierung der normierten Kreuzentropie-Fehlerfunktion, wobei N die Anzahl der Trainingsbilder ist.

$$L = -\frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{w*h}\sum_{i=1}^{w}\sum_{j=1}^{h}\log\left(softmax\left(K = I_{GT}^{n}(i,j)|\hat{y}\left(I^{n}(i,j)\right)\right)\right)\right) \qquad (21)$$

$I^n \in \mathbb{R}^{w\times h\times c}$ ist das n-te Inputbild und $I_{GT}^n \in \mathbb{N}^{w\times h}$ die zu diesem Bild passende Annotation. $\hat{y}(I^n) \in \mathbb{R}^{w\times h\times k}$ ist die *Scoremap* der letzten Entfaltungsschicht des *Decoders,* gegeben das Inputbild I^n. $softmax$ ist die *Softmax*-Funktion aus der Gleichung (19). Diese Fehlerfunktion wird minimiert,

wenn die *Softmax*-Funktion für jeden Pixel die Wahrscheinlichkeit der richtigen semantischen Klasse auf eins schätzt.

Zur Minimierung dieser Fehlerfunktion wurde der Rückübertragungsalgorithmus verwendet. Das *Caffe-Framework* [153] wurde für die Implementierung verwendet. Das *Caffe*-Tool bietet eine Python- und eine C++-Schnittstelle an. In dieser Arbeit wurde die Python-Schnittstelle verwendet. Der Rückübertragungsalgorithmus wurde vom *Caffe*-Tool bereitgestellt. Dazu boten die Autoren des *FCN-8s*-Modells Python-Skripte zum Trainieren der Modelle. Diese Skripte wurden unter Shelhamer [154] heruntergeladen und für diese Arbeit leicht angepasst. Ferner wurde ein kleines Python-Skript zur Anpassung der Trainingsdaten implementiert.

Für das Training wurden folgende Einstellungen vorgenommen: Die *Batch*-Größe wurde aufgrund des limitierten *GPU*-Speichers auf eins gesetzt. Darüber hinaus wurden die Bilder und die dazu passende Annotationen auf die Größe 1505×752 herunterskaliert. Die Lernrate wurde auf $1e^{-10}$ gesetzt. Das mit dem *ILSVRC*-2014-Datensatz [155] vortrainierte *VGG-16*-Modell wurde für die Initialisierung der Parameter des *Encoders* verwendet. Alle Schichten des *Encoders* wurden trainiert. Die Entfaltungsschichten wurden durch eine lineare Interpolation initialisiert. Das Training der Entfaltungsschichten war hier nicht nötig, da es keine wesentliche Änderung der Ergebnisse gebracht hätte [75]. Zwei *Dropout*-Schichten wurden vor und nach der vorletzten Faltungsschicht des *Encoders* hinzugefügt.

Das trainierte *FCN-8s*-Modell hatte ca. 134,5 Millionen Parameter. Das Training dauerte ca. 48 Stunden für insgesamt 300 000 Iterationen.

Die Abbildung 5-4 stellt die normierte Fehlerfunktion des *FCN-8s*-Modells während des Trainings dar, wobei die blaue Kurve die Fehlerfunktion des Trainingsdatensatzes zeigt und die rote Kurve die normierte Fehlerfunktion des Validierungsdatensatzes ist. Die beiden Fehlerfunktionen fielen bis ca. 20 000 Iterationen stark ab. Danach fiel die Fehlerfunktion des Trainingsdatensatzes weiter ab. Die Fehlerfunktion des Validierungsdatensatzes blieb zwischen 20 000 und 70 000 Iterationen relativ konstant. Danach stieg sie wieder auf. Dies bedeutet, dass das Training nach ca. 70 000 Iterationen in *Überanpassung* geriet. Das trainierte Modell von 70 000 Iterationen war also ein guter Kandidat für die Inferenz.

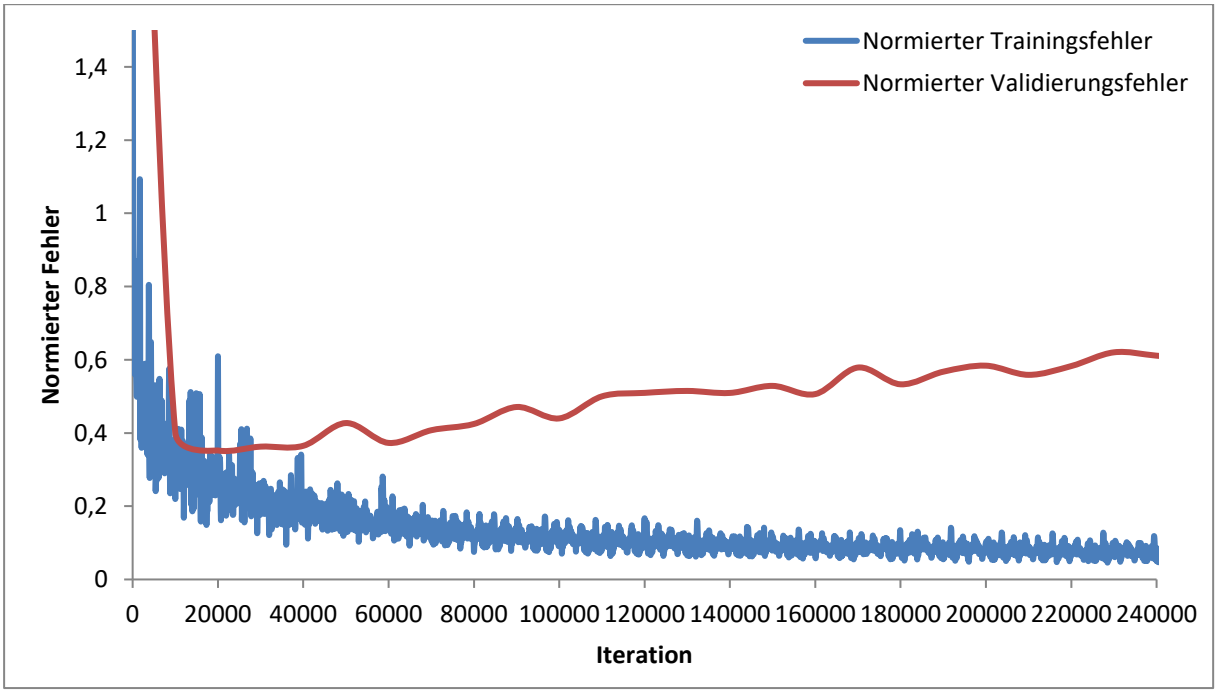


Abbildung 5-4: Grafische Darstellung der normierten Fehlerfunktion des FCN-8s-Modells während des Trainings mit dem DCS-Trainingsdatensatz (blaue Kurve) und dem DCS-Validierungsdatensatz (rote Kurve)

Die *NVIDIA GTX 1080ti graphics processing unit* (*GPU*) wurde in dieser Arbeit verwendet. Diese *GPU* verfügt über 11 GB Speicher und 3584 CUDA-Kerne [156]. Weitere wichtige Daten zu dieser *GPU* sind in der Tabelle 5-2 zu finden.

Tabelle 5-2: Tabellarische Darstellung einiger Eigenschaften der NVIDIA GTX 1080ti GPU [156]

Beschreibung	Wert
NVIDIA CUDA® Cores	3584
Basistaktung (MHz)	1480
Boost-Takt (MHz)	1582
Speichertyp – Geschwindigkeit	11 Gbit/s
Standard-Speicherkonfiguration	11 GB GDDR5X
Breite der Speicherschnittstelle	352-Bit
Speicherbandbreite (GB/s)	484
Busunterstützung	PCIe 3.0
Höhe	4.376"
Länge	10.5"
Tiefe	2 Steckplätze
Max. Temperatur des Grafikprozessors (in °C)	91
Leistungsaufnahme der Grafikkarte (W)	250 W
Empfohlene Systemleistung (W)	600 W
Zusätzliche Stromanschlüsse	6-pin, 8-pin

5.5 Inferenz mit dem *FCN-8s*-Modell

Die Inferenz wurde mithilfe des *Caffe*-Tools auf der *NVIDIA GTX 1080ti GPU* ausgeführt. Dafür wurde das trainierte Modell nach 70 000 Iterationen verwendet. Die Inputbilder stammen aus dem *DCS*-Validierungsdatensatz. Jedes Bild war 1505 Pixel breit und 752 hoch.

Die Inferenz mit dem *FCN-8s*-Modell dauerte im Schnitt ca. 390 ms pro Bild und ließ sich vor allem durch die Auflösung des Inputbildes erklären. Bei geringeren Auflösungen war die Inferenz deutlich schneller, da die Komplexität der Inferenz linear von der Dimension des Inputbildes abhängig war (siehe Abschnitt 2.4.4). Allerdings verschlechterte sich die Segmentierungsgüte bei kleinen Objekten, wenn die Bildauflösung reduziert wurde.

Der benötigte *GPU*-Speicher pro Bild lag im Schnitt bei vier GB, wobei die *GPU* 11 GB zur Verfügung stellte. Das *FCN-8s*-Modell war also, was den Speicher angeht, bezogen auf die verfügbare *GPU* wenig komplex.

5.6 Quantitative Evaluation des *FCN-8s*-Modells

Zur Evaluation des trainierten *FCN-8s*-Modells wurde die weitverbreitete *IOU*-Metrik verwendet.

$$IOU = \frac{TP}{TP + FP + FN} \tag{22}$$

TP, *FP* und *FN* stehen jeweils für richtig positiv (*true positive*), falsch positiv (*false positive*) und falsch negativ (*false negative*). Der beste Wert der *IOU*-Metrik ist 100 %, während 0 % der schlechteste Wert ist. Die optimale Segmentierung sollte sich von dem 100-%-*IOU*–Wert annähern. *mIOU* steht für den mittleren *IOU*-Wert.

5.6.1 Evaluation unter Betrachtung aller Pixel

In diesem Abschnitt werden alle Pixel der Inputbilder bei der Evaluation betrachtet. Dabei werden sowohl die Klassen als auch die Kategorien einzeln und im Mittel evaluiert.

5.6.1.1 Evaluation der Klassen

Die Tabelle 5-3 stellt die Ergebnisse der Evaluation des trainierten *FCN-8s*-Modells auf dem *DCS*-Validierungsdatensatz vor. Hier wurden die Inputbilder auf die Größe der Trainingsbilder auf 1505×752 herunterskaliert. Der *mIOU*-Wert über alle semantischen Klassen betrug 60,7 %. Dieser Wert war 7,4 % relativ schlechter als der *mIOU*-Wert des *FCN-8s*-Modells, das mit dem *DCS*-Testdatensatz evaluiert und auf der *DCS-Benchmark*-Seite publiziert wurde [157]. Das *FCN-8s*-Modell der *DCS-Benchmark*-Seite hatte einen *mIOU*-Wert von 65,2 %. Dieser Unterscheid lag daran, dass der *DCS*-Testdatensatz Unterschiede zu dem Validierungsdatensatz zeigte. Dazu wurde das *FCN-8s*-Modell der *DCS-Benchmark*-Seite mit den Bildern in der vollen Auflösung auf dem Testdatensatz evaluiert. Aufgrund der Einführung der Klasse *unknow* in dieser Arbeit unterscheidet sich die Anzahl der in dieser Arbeit evaluierten Klassen von den Klassen der *DCS-Benchmark*-Seite. Dies kann auch die Unterschiede der *mIOU*-Werte erklären.

Eine detaillierte Analyse der Evaluationsergebnisse aus der Tabelle 5-3 zeigte, dass das *FCN-8s*-Modell folgende Eigenschaften hatte:

1. Klassen, die oft in den Trainingsdaten vorkamen und eine große Ausdehnung in den Bildern hatten, wie bspw. *road, sky, building, vegetation* und *car*, wiesen *IOU*-Werte von mehr als 85 % auf. Der Hauptgrund war, dass diese Klassen während der Trainingsphase einen größeren Einfluss auf die Fehlerfunktion hatten. Die richtige Segmentierung dieser Klassen führte zu einer schnelleren Minimierung der Fehlerfunktion. Somit lernte das *FCN-8s*-Modell, diese Klassen so gut wie möglich zu segmentieren. Die Klasse *road* mit einem *IOU*-Wert von 92,2 % wurde am besten segmentiert.
2. Klassen, die zwar öfter in den Trainingsdaten vorkamen, aber eine kleine Ausdehnung in den Bildern hatten, wie etwa *pole* und *person*, wurden mit den *IOU*-Werten zwischen 40,3 % und 66,7 % etwas schlechter segmentiert als die Klassen mit größerer Ausdehnung in den Bildern. Die Informationen über diese Klassen gingen im *Encoder* teilweise verloren, da der *Encoder* vor der Segmentierung die Inputbilder um Faktor $1/16$ herunterskalierte.
3. Klassen, die eine größere Ausdehnung in den Bildern hatten, aber seltener vorkamen, wie etwa *bus, truck* und *train*, wurden ebenfalls schlechter segmentiert. Hier lagen die *IOU*-Werte zwischen 39,9 % und 60,8 %. Der Grund dafür war, dass diese Regionen aufgrund ihrer Größe, ihrer homogenen Textur und der eingeschränkten rezeptiven Felder nicht eindeutig in dem Merkmalraum abgebildet wurden. Solche Fehler wurden im Abschnitt 3.5.2 diskutiert.
4. Die Klasse *unknow* enthielt sowohl klein als auch groß ausgedehnte Objekte sowie seltene und nicht seltene Klassen. Der *IOU*-Wert dieser Klasse war deshalb mit 66,6 % etwas größer als der *mIOU*-Wert.

Tabelle 5-3: Evaluationsergebnisse des FCN-8s-Modells mit den Klassen des DCS-Validierungsdatensatzes anhand der IOU-Metrik

Klassenname	IOU	Klassenname	IOU	Klassenname	IOU
unknow	0,666	traffic_light	0,489	car	0,893
road	0,922	traffic_sign	0,587	truck	0,399
sidewalk	0,675	vegetation	0,884	bus	0,608
building	0,849	terrain	0,506	train	0,438
wall	0,362	sky	0,889	motorcycle	0,394
fence	0,426	person	0,667	bicycle	0,637
pole	0,403	rider	0,451	**mIOU**	0,607

Die o. g. Eigenschaften des in dieser Arbeit trainierten *FCN-8s*-Modells stimmten zum größten Teil mit den Evaluationsergebnissen des *FCN-8s*-Modells, das mit dem *DCS*-Testdatensatz evaluiert und auf der *DCS-Benchmark*-Seite publiziert wurde [86], überein. Die Abbildung 5-5 stellt das Histogramm der *IOU*-Werte über alle Klassen für beide Modelle dar, wobei das blaue Histogramm die *IOU*-Werte des in dieser Arbeit trainierten *FCN-8s*-Modells repräsentiert, während das rote Histogramm dem Verlauf des Modells aus der *DCS-Benchmark*-Seite entspricht. Beide Histogramme hatten einen ähnlichen Verlauf. Die kleinen Unterschiede zwischen den beiden Histogrammen ließen sich durch die Argumente der Unterschiede der *mIOU*-Werte aus dem oberen Abschnitt erklären.

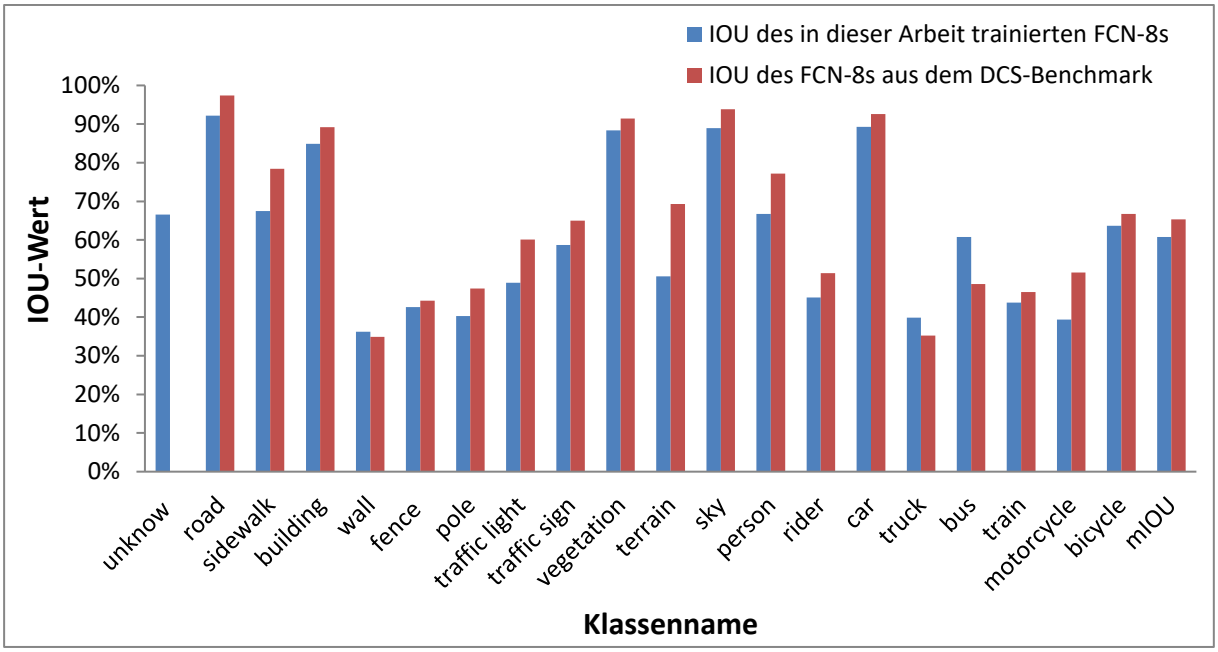


Abbildung 5-5: Histogramm der IOU-Werte über alle Klassen des in dieser Arbeit trainierten FCN-8s-Modells (blaues Histogramm) und des FCN-8s-Modells aus der DCS-Benchmark-Seite [86] (rotes Histogramm)

Um ein besseres Verständnis der Ergebnisse des *FCN-8s*-Modells zu schaffen, stellt die Abbildung 5-6 die Konfusionsmatrix dar. Die *Ground-Truth*-Klassen sind auf den Zeilen dargestellt. Die Spalten stellen die Segmentierungsklassen des *FCN-8s*-Modells dar. Die beiden ersten Zeilen und Spalten sind jeweils die Namen der semantischen Klassen und die entsprechenden Klassen-IDs. Die Werte der Konfusionsmatrix sind in Prozent angegeben. Die Konfusionsmatrix bestätigte die o. g. Eigenschaften des *FCN-8s*-Modells:

1. In den Klassen der Tabelle 5-3 mit *IOU*-Werten größer 85 % wurden mindestens 90 % der *Ground-Truth*-Pixel richtig segmentiert.
2. Klassen mit einer kleinen Ausdehnung in den Bildern wurden oft falsch als ihre Hintergrundklassen segmentiert. So wurden bspw. die Pixel der Klassen *pole, traffic ligth* und *traffic sign* öfter falsch als die Hintergrundklassen *building* oder *vegetation* segmentiert.
3. Bei Regionen, die ähnlich aussahen, wurden die Klassen präferiert, die öfter in den Trainingsdaten vorkamen. Die Klassen *wall, fence, train* und *truck* wurden oft falsch als *building* segmentiert. Ähnlich wurde die Klasse *rider* oft als *person* segmentiert. Die Klassen *truck, bus* und *motorcycle* wurden auch als *car* falsch segmentiert. Diese Beobachtung stimmte auch für die Klassen *vegetation* und *terrain*.
4. Da die Klasse *unknow* Pixel enthielt, die während der Generierung von Trainingsdaten nicht eindeutig einer semantischen Klasse zugeordnet werden konnten, könnten die Pixel dieser Klasse auch zu den anderen Klassen gehören. Die Mehrheit der Pixel, die als *unknow* annotiert wurden, könnte auch zu den Klassen *road*, *sidewalk* und *building* gehören. Dies erklärt,

warum die Klasse *unknow* öfter falsch als die Klasse *road*, *sidewalk* oder *building* segmentiert wurde.

		road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	unknow
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
road	0	96.5	1.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.3	0.0	0.0	0.0	0.0	0.0	1.5
sidewalk	1	4.5	88.8	0.4	0.1	0.3	0.7	0.0	0.0	0.3	0.6	0.0	0.5	0.0	0.4	0.0	0.0	0.0	0.0	0.3	3.2
building	2	0.0	0.3	91.5	0.2	0.2	1.1	0.2	0.3	2.3	0.0	0.6	0.4	0.0	0.3	0.2	0.0	0.0	0.0	0.1	2.0
wall	3	5.9	6.1	12.8	43.0	6.6	1.6	0.0	0.3	12.2	0.7	0.0	1.1	0.1	0.9	1.2	0.0	0.1	0.0	0.7	6.6
fence	4	0.4	2.5	16.1	2.9	55.2	2.5	0.0	1.3	5.7	0.5	0.0	1.2	0.1	1.3	0.8	0.1	0.2	0.1	1.5	7.7
pole	5	0.2	4.1	14.6	0.2	0.9	57.0	1.6	1.1	10.3	0.7	1.0	1.4	0.1	1.3	0.2	0.1	0.0	0.1	0.9	4.2
traffic light	6	0.0	0.0	8.3	0.0	0.0	2.8	78.4	0.6	6.9	0.0	0.5	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	2.2
traffic sign	7	0.3	0.3	7.3	0.0	0.8	1.9	0.5	79.1	3.5	0.0	0.2	0.6	0.1	1.1	0.1	0.0	0.0	0.0	0.1	3.9
vegetation	8	0.0	0.2	2.1	0.0	0.1	0.5	0.1	0.2	94.8	0.5	0.5	0.1	0.0	0.2	0.0	0.0	0.0	0.0	0.1	0.4
terrain	9	1.8	9.7	0.4	1.0	0.7	0.7	0.0	0.0	14.4	66.1	0.0	0.1	0.1	0.5	0.0	0.0	0.0	0.0	0.3	4.0
sky	10	0.0	0.0	1.0	0.0	0.0	0.3	0.1	0.0	1.0	0.0	97.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6
person	11	0.9	2.0	3.3	0.1	0.1	0.8	0.0	0.1	1.0	0.0	0.0	85.9	2.1	1.3	0.0	0.1	0.0	0.1	1.2	0.8
rider	12	0.5	0.8	2.1	0.1	0.2	0.4	0.1	0.1	1.6	0.1	0.0	16.3	64.3	2.3	0.0	0.1	0.0	2.6	7.9	0.8
car	13	0.6	0.2	0.6	0.0	0.1	0.2	0.0	0.1	0.4	0.0	0.0	0.3	0.1	96.4	0.3	0.2	0.0	0.1	0.2	0.5
truck	14	2.9	0.2	5.0	0.7	0.6	0.4	0.2	0.6	1.7	0.0	2.0	0.5	0.2	20.4	56.6	3.6	0.1	0.1	0.0	4.1
bus	15	0.5	0.1	2.0	0.1	0.3	1.1	0.2	0.6	2.4	0.0	0.0	0.4	0.4	10.8	5.9	72.1	1.8	0.0	0.1	1.2
train	16	0.4	0.1	11.8	0.0	2.6	2.2	0.2	0.7	3.3	0.1	1.0	0.3	0.0	1.2	2.5	19.4	52.4	0.0	0.1	1.6
motorcycle	17	0.7	2.8	1.8	0.0	1.5	1.3	0.0	0.1	0.8	0.0	0.0	4.1	4.0	11.1	0.0	0.1	0.1	62.8	6.9	2.0
bicycle	18	0.4	2.2	2.0	0.0	0.3	1.1	0.0	0.0	1.0	0.3	0.0	2.2	4.0	1.5	0.0	0.0	0.0	1.7	81.9	1.3
unknow	19	9.6	5.3	4.4	0.2	0.5	0.9	0.2	0.6	1.3	0.6	0.4	0.6	0.0	0.7	0.2	0.1	0.0	0.1	0.4	73.9

Abbildung 5-6: Konfusionsmatrix des FCN-8s-Modells auf den Klassen des DCS-Validierungsdatensatzes. Die Ground-Truth- und die Segmentierungsklassen sind jeweils in den Zeilen und Spalten angeordnet. Die Zahlen sind in Prozent angegeben.

5.6.1.2 Evaluation der Kategorien

Neben den Klassen stellt der *DCS*-Datensatz Kategorien zur Verfügung. Kategorien fassten Klassen zusammen, die eine ähnliche Semantik hatten. Wie die Klassen in Kategorien zusammengefasst wurden, stellte die

Tabelle 5-1 dar. So wurden bespielweise die Klassen *person* und *rider* unter der Kategorie *human* zusammengefasst, während die Kategorie *vehicle* alle Klassen, die Verkehrsmittel waren, enthielt. Die *IOU*-Werte des *FCN-8s*-Modells mit den Kategorien des *DCS*-Validierungsdaten-satzes sind in der Tabelle 5-4 dargestellt. Der *mIOU*-Wert über alle Kategorien lag bei 78 %. Dies entsprach einer relativen

Verbesserung von 28,5 % im Vergleich zu dem *mIOU*-Wert über alle Klassen aus der Tabelle 5-3. Diese deutliche Verbesserung ließ sich durch die Tatsache erklären, dass Klassen derselben Kategorien aufgrund der semantischen Ähnlichkeiten oft nicht leicht zu trennen waren. Diese Tatsache wurde anhand der Konfusionsmatrix der Abbildung 5-6 in dem vorherigen Abschnitt erläutert. Die Eigenschaften, die das *FCN-8s*-Modell für die Evaluation der Klassen aufwies, galten auch für die Evaluation der Kategorien. So hatten die Kategorien *flat*, *construction*, *nature*, *sky* und *vehicle* aufgrund ihrer Häufigkeiten in den Trainingsdaten mindestens 84 % *IOU*-Werte. Die Kategorien *human* und *object* enthielten klein ausgedehnte Regionen und wurden somit mit *IOU*-Werten zwischen 48,9 % und 68,7 % schlechter segmentiert. Die Kategorie *void* enthielt als einzige Klasse *unknow* und zeigte somit dasselbe Verhalten wie die Klasse *unknow*.

Tabelle 5-4: Evaluationsergebnisse des FCN-8s-Modells mit den Kategorien des DCS-Validierungsdatensatzes anhand der IOU-Metrik

Kategorienname	**IOU**
void	0,666
flat	0,918
construction	0,843
object	0,489
nature	0,883
sky	0,889
human	0,687
vehicle	0,868
mIOU	0,78

Die Unterschiede, die beim Vergleich des in dieser Arbeit trainierten *FCN-8s*-Modells mit dem *FCN-8s*-Modell aus der *DCS-Benchmark*-Seite für die Klassen festgestellt wurden, ließen sich für die Kategorien auch feststellen. Der *mIOU*-Wert der Kategorien des in dieser Arbeit trainierten *FCN-8s*-Modells entsprach einer relativen Verschlechterung von 7,6 % im Vergleich zum *mIOU*-Wert der Kategorien des *FCN-8s*-Modells aus der *DCS-Benchmark*-Seite [86]. Die Histogramme der *IOU*-Werte über alle Kategorien für beide Modelle stellt die Abbildung 5-7 dar. Die beiden Histogramme zeigen einen ähnlichen Verlauf, wobei das blaue Histogramm die Kategorie-*IOU* des in dieser Arbeit trainierten *FCN-8s*-Modells repräsentiert, während das rote Histogramm dem Verlauf des Modells aus der *DCS-Benchmark*-Seite entspricht. Die Unterschiede zwischen den beiden Histogrammen ließen sich durch die Argumente der Unterschiede der Klassen-*IOU* dieser Modelle aus dem Abschnitt 5.6.1.1 erklären.

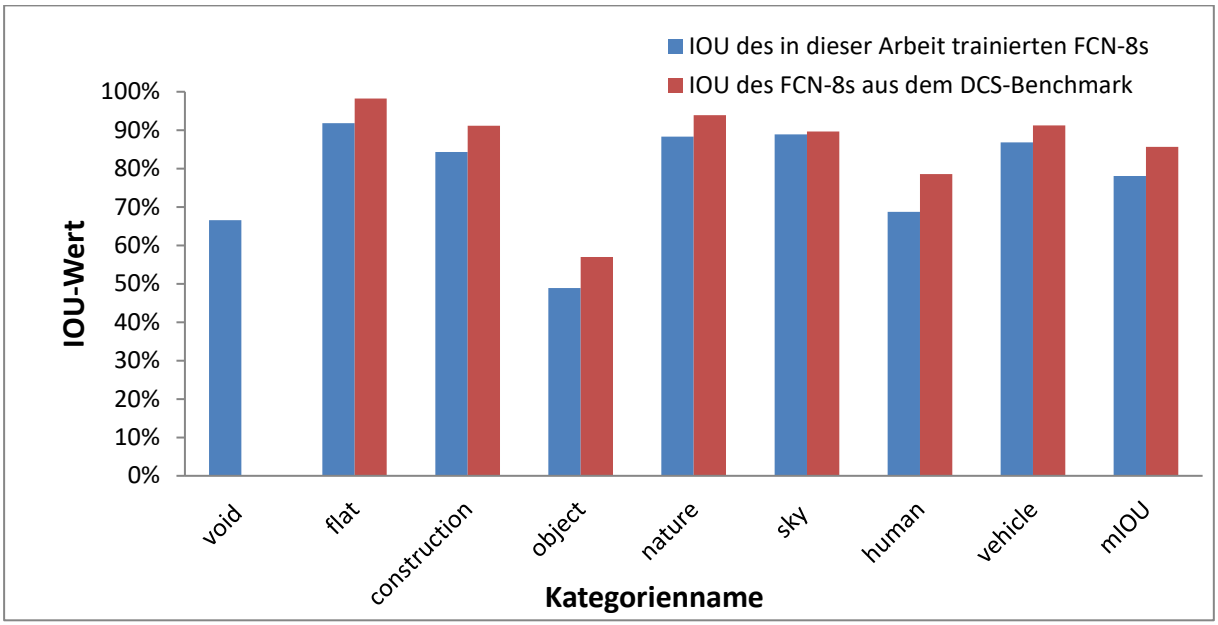


Abbildung 5-7: Histogramm der IOU-Werte über alle Kategorien des in dieser Arbeit trainierten FCN-8s-Modells (blaues Histogramm) und des Modells aus der DCS-Benchmark-Seite [86] (rotes Histogramm)

Ähnlich wie bei der Evaluation des *FCN-8s*-Modells für die semantischen Klassen stellt die Abbildung 5-8 die Konfusionsmatrix des *FCN-8s*-Modells für die Kategorien dar. Ähnliche Eigenschaften des *FCN-8s*-Modells wie bei den Klassen ließen sich hier feststellen:

1. Die Kategorien mit *IOU*-Werten größer 84 % hatten mindestens 90 % der *Ground-Truth*-Pixel richtig segmentiert.
2. Kategorien mit einer kleinen Ausdehnung wie *object* und *human* wurden oft falsch als ihre Hintergrundklassen *construction* und *nature* segmentiert.
3. Die Kategorie *void* zeigte ein ähnliches Verhalten wie die Klasse *unknow*, da die Klasse *unknow* die einzige Klasse in dieser Kategorie war. Die Kategorie *void* enthielt viele falsche Segmentierungen der Kategorien *flat* und *construction*, da die Kategorien *flat* und *construction* die Klassen enthielten, die sehr ähnlich mit der Klasse *unknow* der Kategorie *void* waren.

		void	flat	construction	object	nature	sky	human	vehicle
		0	1	2	3	4	5	6	7
void	0	73.9	14.9	5.1	1.6	1.9	0.4	0.6	1.5
flat	1	1.7	97.4	0.1	0.1	0.2	0.0	0.1	0.4
construction	2	2.4	0.8	90.4	1.7	2.7	0.5	0.5	0.9
object	3	3.9	2.9	12.9	67.7	8.5	0.8	1.2	2.1
nature	4	0.6	0.8	2.3	0.8	94.7	0.5	0.2	0.3
sky	5	0.6	0.0	1.0	0.4	1.0	97.0	0.0	0.0
human	6	0.8	2.7	3.4	0.9	1.1	0.0	86.9	4.1
vehicle	7	0.8	1.0	1.3	0.5	0.7	0.1	0.9	94.7

Abbildung 5-8: Konfusionsmatrix des FCN-8s-Modells auf den Kategorien des DCS-Validierungsdatensatzes. Die Ground-Truth- und die Segmentierungskategorien sind jeweils in den Zeilen und Spalten angeordnet. Die Zahlen der Konfusionsmatrix sind in Prozent angegeben.

5.6.2 Evaluation unter Betrachtung der für die Fahraufgabe relevanten Pixel

Die Evaluationsergebnisse aus dem Abschnitt 5.6.1 betrachtete alle Pixel in den Bildern. Allerdings spielt im Kontext des automatisierten Fahrens die Relevanz von Pixeln bzw. Objekten eine Rolle bei der Evaluation von Klassifikatoren. In diesem Abschnitt wird das trainierte *FCN-8s*-Modell mit Fokus auf die Relevanz von Pixeln evaluiert. Diese Evaluation trägt zur holistischen Betrachtung von Szenen und Situation, da die Relevanz als Teil der Situationsinterpretation für die Evaluation eines Szenenelements verwendet wird.

5.6.2.1 Berechnung der Erreichbarkeitsmenge eines dynamischen Systems

Eine Möglichkeit, um die Relevanz eines Pixels zu schätzen, ist zu schauen, ob dieses Pixel von dem Ego-Fahrzeug erreichbar wäre, gegeben die dynamischen Eigenschaften des Ego-Fahrzeugs. Althoff et al. [158] schränkten die Erreichbarkeitsmenge eines dynamischen Systems durch den *Kammschen* Kreis mit Zentrum

$$c(t) = \begin{pmatrix} c_x(t) \\ c_y(t) \end{pmatrix} = \begin{pmatrix} v * t * cos(\psi) \\ v * t * sin(\psi) \end{pmatrix} = \quad (23)$$

und Radius

$$r(t) = 0{,}5 * a * t^2 = \quad (24)$$

wobei v, a und ψ jeweils die Geschwindigkeit, die maximale Beschleunigung und die Rotation um die z-Achse des Systems sind, ein. t ist die Zeit. Die x-, y- und z-Achsen zeigen jeweils in die longitudinale Fahrtrichtung, die laterale Fahrtrichtung nach links und nach oben. In der Abbildung 5-9 ist die

graue Region ein Beispiel für die Erreichbarkeitsmenge eines Systems zwischen t_k und t_{k+1} mit den Parametern $v = 20\,m/s$, $a = 10\,m/s^2$ und $\psi = 0°$.

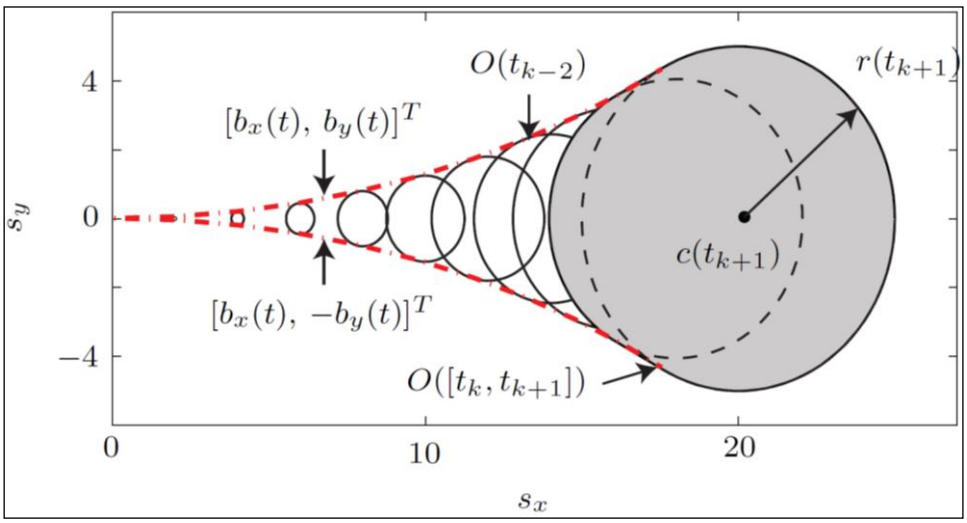


Abbildung 5-9: Beispiel der Erreichbarkeitsmenge eines Systems zwischen t_k und t_{k+1} mit $v = 20\,m/s$, $a = 10\,m/s^2$ und $\psi = 0°$ (Bild aus [158])

Mit der Erreichbarkeitsmenge aus den Gleichungen (23) und (24) wird wie folgt die Relevanz eines Pixels definiert: Ein Pixel ist für das Ego-Fahrzeug relevant, wenn dieses sich in der Erreichbarkeitsmenge des Ego-Fahrzeugs im Zeitintervall $[t_k, t_{k+n}], n \in \mathbb{N}$ befindet. Dabei soll auch die Dynamik dieses Pixels betrachtet werden.

Der *DCS*-Datenatzt stellt neben den Bildern und den dazu passenden Annotationen Disparitätskarten zur Verfügung. Diese Disparitätskarten wurden mit einem Stereokamerasystem gewonnen. Anhand dieser Disparitätskarten können die Pixel in der 3-D-Welt, wie in der folgenden Gleichung definiert, projektiert werden.

$$p(d) = \begin{pmatrix} z \\ x \\ y \end{pmatrix} = \begin{pmatrix} f * b/d \\ z * (u - U_0)/f \\ z * (v - V_0)/f \end{pmatrix} \tag{25}$$

$d \in \mathbb{N}$ ist die Disparität des Pixels *p* mit den Bildkoordinaten $(u, v) \in \mathbb{N}^2$. $f \in \mathbb{R}$ ist die Brennweite der Kamera mit dem optischen Zentrum $(U_0, V_0) \in \mathbb{N}^2$ und b die Basisbreite des Stereokamerasystems. Der Ursprung des Koordinatensystems ist die Mitte der Stereobasis. Die z- und x-Achsen zeigen jeweils in der longitudinalen Fahrrichtung nach vorne und in der lateralen Fahrtrichtung nach rechts. Die y-Achse zeigt nach unten.

Die Abbildung 5-10 zeigt beispielhaft die in dieser Arbeit verwendete Erreichbarkeitsmenge des Ego-Fahrzeugs als hellgraue Fläche für das Intervall $t = [0s, 3s]$, die maximale Beschleunigung $a = 4\,m/s^2$ und die Orientierung $\psi = 90°$. Auf dem linken Bild hat das Ego-Fahrzeug die Geschwindigkeit $v = 30\,km/h$ und kann innerhalb von 3 s Regionen maximal 45 m in der Tiefe und $\pm$ 20 m in der Breite erreichen. Das Ego-Fahrzeug auf dem rechten Bild kann Regionen bis zu 60 m in der Tiefe und $\pm$ 20 m in der Breite mit einer Geschwindigkeit von $v = 50\,km/h$ erreichen.

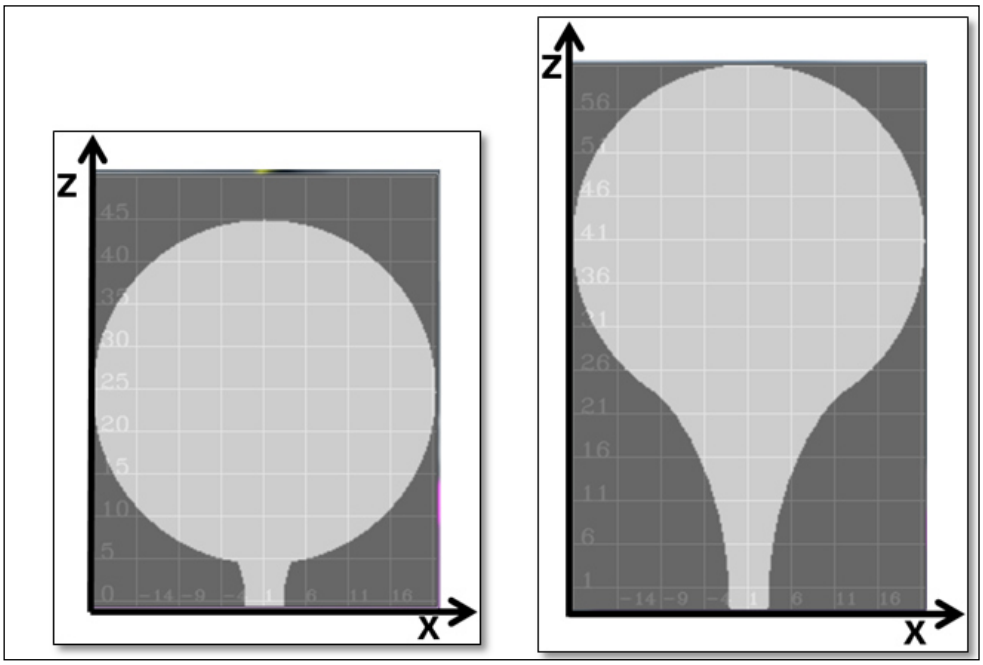


Abbildung 5-10: Erreichbarkeitsmenge des Ego-Fahrzeugs für die Parameter $t = [0s, 3s]$, $v = 30\,km/h$ (links), $v = 50\,km/h$ (rechts), $a = 4\,m/s^2$ und $\psi = 90°$. Die hellgraue Region ist erreichbar, während die dunkelgraue unerreichbar ist.

Basierend auf den Disparitätskarten des *DCS*-Datensatzes wurden Pixel bestimmt, die sich in der Erreichbarkeitsmenge des Ego-Fahrzeugs mit den Parametern aus der Abbildung 5-10 befinden. Diese Pixel sind in den Bildern der zweiten Spalte der Abbildung 5-11 mit den Farben rot bis grün markiert, wobei rot nah bedeutet und grün fern. Alle Pixel mit der Farbe schwarz sind unerreichbar. Das Bild in der zweiten Spalte, erste Zeile entspricht der Erreichbarkeitsmenge mit den Parametern $t = [0\ \mathrm{s}, 3\ \mathrm{s}]$, $\mathrm{v} = 30\ \mathrm{km/h}$ und $\mathrm{a} = 4\ \mathrm{m/s^2}$ (siehe linkes Bild in der Abbildung 5-10). In der zweiten Spalte, zweite Zeile entspricht das Bild der Erreichbarkeitsmenge mit den Parametern $t = [0\ \mathrm{s}, 3\ \mathrm{s}]$, $\mathrm{v} = 50\ \mathrm{km/h}$ und $\mathrm{a} = 4\ \mathrm{m/s^2}$ (siehe rechtes Bild in der Abbildung 5-10). Die Kamerabilder sind in der ersten Spalte der Abbildung 5-11 dargestellt.

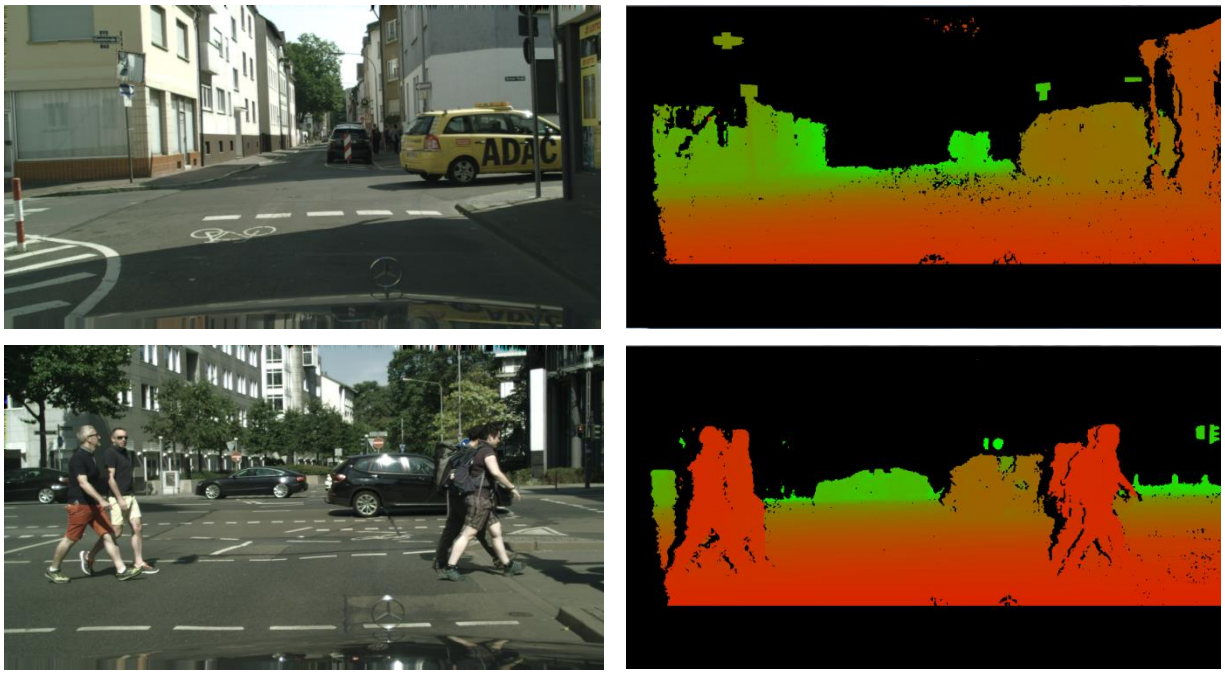


Abbildung 5-11: Zwei Bilder aus dem DCS-Validierungsdatensatz (erste Spalte) [81] und die Pixel, die vom Ego-Fahrzeug erreichbar sind für die Parameter $t = [0\ s, 3\ s]$, $a = 4\ m/s^2$ und $\psi = 90°$, $v = 30\ km/h$ (zweite Spalte, erste Zeile), $v = 50\ km/h$ (zweite Spalte, zweite Zeile). Die Farbe der Pixel der zweiten Spalte codiert die Entfernung. Rot bedeutet nah, grün bedeutet fern und schwarz unerreichbar.

Für die Berechnung der Erreichbarkeit von Pixeln wurde die Dynamik der Pixel in dieser Arbeit nicht betrachtet, da der *DCS*-Datensatz keine Informationen dazu bereitstellt. Darüber hinaus sind die *DCS*-Daten nicht als Videosequenz verfügbar. So können auch Ansätze mit dem optischen Fluss hier nicht angewendet werden. Um trotzdem sicherzustellen, dass so viele relevante Pixel wie möglich betrachtet wurden, wurden die Parameter der Erreichbarkeitsmenge großzügig ausgewählt. Damit wurde die Erreichbarkeitsmenge vergrößert.

5.6.2.2 Evaluation der erreichbaren Pixel anhand der semantischen Klassen

Zur Evaluation der semantischen Segmentierung für relevante Pixel wurde ähnlich wir bei der Evaluation für alle Pixel die *mIOU*-Metrik über alle semantischen Klassen anhand des *DCS*-Validierungsdatensatzes berechnet. Dabei wurden lediglich die Pixel betrachtet, die erreichbar waren. Die Abbildung 5-12 stellt den Verlauf der *mIOU*-Werte über alle Klassen auf dem *DCS*-Validierungsdatensatz dar, wobei die Erreichbarkeitsmenge mit den folgenden Parametern berechnet wurde: $t = [0\ \mathrm{s}, 3\ \mathrm{s}]$, $\mathrm{v} = \{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, \infty\}\ \mathrm{km/h}$, $\mathrm{a} = 4\ \mathrm{m/s^2}$ und $\psi = 90°$. Für den Fall $\mathrm{v} = \infty$ wurden alle Pixel bei der Evaluation betrachtet (siehe Abschnitt 5.6.1). Diese Abbildung zeigte eine kleine Verbesserung der *mIOU*-Werte der Klassen für relevante Pixel von bis zu 3,5 % im Vergleich zu dem Fall, in dem alle Pixel betrachtet wurden (Geschwindigkeit $\mathrm{v} = \infty$).

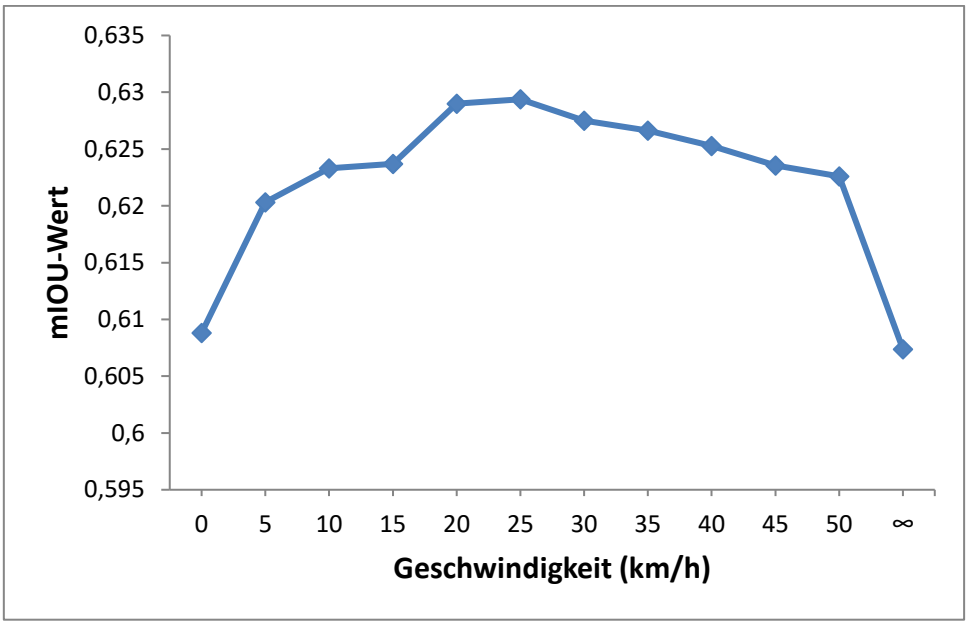


Abbildung 5-12: Verlauf der mIOU-Werte über alle Klassen auf dem DCS-Validierungsdatensatz für die erreichbaren Pixel. Die Erreichbarkeitsmenge des Ego-Fahrzeugs wurde mit $t = [0\ s, 3\ s]$, $v = \{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, \infty\}\ km/h$, $a = 4\ m/s^2$ und $\psi = 90°$ berechnet.

Um den Effekt der relevanten Pixel auf die einzelnen Klassen zu betrachten, wurde der Verlauf der *IOU*-Werte für jede semantische Klasse in der Abbildung 5-13 dargestellt. Dabei stellten sich folgende Aspekte heraus:

1. Die Klassen, die zwar öfter in den Trainingsdaten vorkamen, aber eine kleine Ausdehnung in den Bildern hatten wie *pole, person, traffic light* und *traffic sign* zeigten eine deutliche Verbesserung des *IOU*-Werts, wenn diese sich näher an dem Ego-Fahrzeug befanden. So erreichte bspw. die Klasse *traffic light* eine relative Verbesserung des *IOU*-Werts von 70 % mit der Geschwindigkeit des Ego-Fahrzeugs $\mathrm{v} = 15\ \mathrm{km/h}$ im Vergleich zu dem Fall, wo alle Pixel betrachtet wurden. Die Klasse *person*, die aufgrund der Gefährdung ihrer Vulnerabilität im Verkehr besonders behandelt werden sollte, erreichte eine relative Verbesserung des *IOU*-Werts

von 20,3 % mit der Geschwindigkeit des Ego-Fahrzeugs $\mathrm{v} = 10\,\mathrm{km/h}$. Diese Verbesserungen verstärkten die Begründung aus dem Abschnitt 5.6.1. Zwar gingen die Informationen über diese Klassen im *Encoder* teilweise verloren, weil der *Encoder* die Inputbilder um Faktor $1/16$ herunterskalierte, allerdings waren diese Regionen größer, je näher sie an dem Ego-Fahrzeug waren. So wurden die Informationen über diese Regionen im *Encoder* zum größten Teil beibehalten.

2. Die Klassen, die öfter in den Trainingsdaten vorkamen und eine große Ausdehnung im Bild hatten, wurden mal besser mal schlechter segmentiert. Während die Klassen *road* und *car* leichte Verbesserungen zeigten, wurden die Hintergrundklassen *sky, building* und *vegetation* schlechter segmentiert. So zeigte bspw. die Klasse *vegetation* eine relative Verschlechterung des *IOU*-Werts von 22,2 % mit der Geschwindigkeit des Ego-Fahrzeugs $\mathrm{v} = 10\,\mathrm{km/h}$ im Vergleich zu dem Fall, wo alle Pixel betrachtet wurden. Dies widersprach etwa der Begründung aus dem Abschnitt 5.6.1. Allerdings zeigt die Abbildung 5-14, dass der Anteil an erreichbaren *Ground-Truth*-Pixeln in dem *DCS*-Validierungsdatensatz von Klasse zu Klasse unterschiedlich war. Dabei wurde die Erreichbarkeitsmenge des Ego-Fahrzeugs mit $\mathrm{t} = [0\,\mathrm{s}, 3\,\mathrm{s}]$, $\mathrm{v} = 50\,\mathrm{km/h}$, $\mathrm{a} = 4\,\mathrm{m/s^2}$ und $\psi = 90°$ berechnet. Während in den Klassen wie bspw. *road* und *car* jeweils 59,3 % und 75 % der Pixel erreichbar waren, waren die Pixel der Klassen *building, vegetation* und *sky* nur zu 13,4 %, 10,2 % und 1 % erreichbar. Wenige falsche Segmentierungen hatten deshalb einen großen Einfluss auf die Klassen, die wenige erreichbare Pixel hatten. Eine ausführliche Analyse der Konfusionsmatrix in den nächsten Abschnitten unterstützt diese Erläuterung.
3. Klassen, die eine größere Ausdehnung in den Bildern hatten, aber seltener in den Trainingsdaten vorkamen wie etwa *bus, truck* und *train,* wurden ebenfalls unter Betrachtung der erreichbaren Pixel schlechter segmentiert. Der Grund dafür war, dass die relevanten Pixel dieser Klassen oft näher an dem Ego-Fahrzeug lagen. So wurden die Pixel dieser Klassen aufgrund ihrer größeren Ausdehnung und der eingeschränkten rezeptiven Felder der Neuronen des Modells falsch segmentiert.
4. Die Klasse *unknow* wurde unter Betrachtung der Relevanz schlechter segmentiert. So ließ sich eine deutliche Verschlechterung des *IOU*-Werts von mehr als 200 % mit der Geschwindigkeit des Ego-Fahrzeugs $\mathrm{v} = 10\,\mathrm{km/h}$ im Vergleich zu dem Fall, in dem alle Pixel betrachtet wurden, feststellen. Dies widersprach auch hier der Begründung aus dem Abschnitt 5.6.1. Allerdings stellte auch die Abbildung 5-14 dar, dass die Pixel der Klasse *unknow* nur zu 17,4 % erreichbar waren. Ein kleiner Anteil an falsch segmentierten Pixeln hatte einen großen Einfluss auf die *IOU*-Werte. Eine ausführliche Analyse der Konfusionsmatrix in den nächsten Abschnitten liefert weitere Erklärungen.

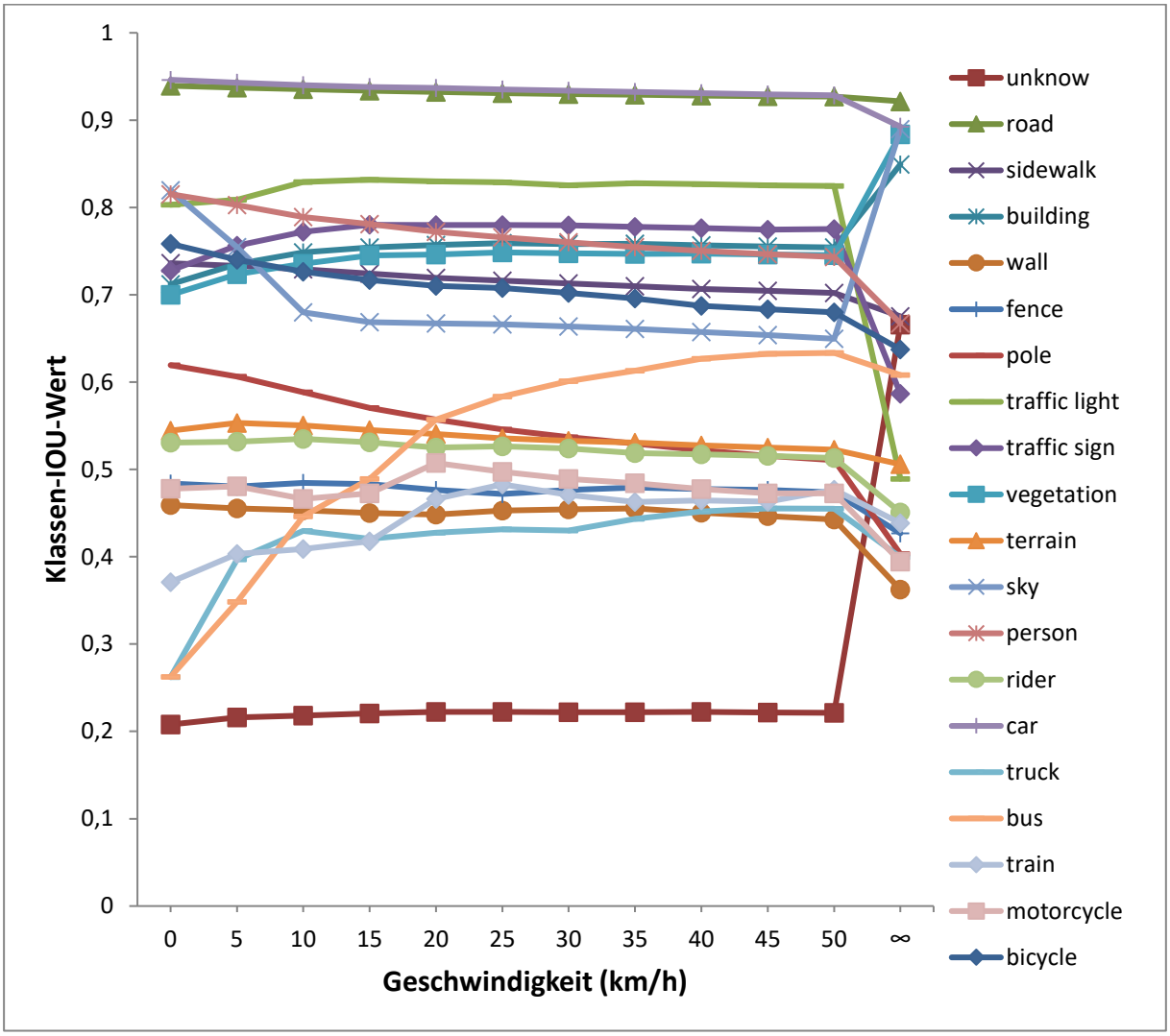


Abbildung 5-13: Verlauf der IOU-Werte einzelner semantischer Klassen auf dem DCS-Validierungsdatensatz für die erreichbaren Pixel. Die Erreichbarkeitsmenge des Ego-Fahrzeugs wurde mit $t = [0\,s, 3\,s]$, $v = \{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, \infty\}\,km/h$, $a = 4\,m/s^2$ *und* $\psi = 90°$ *berechnet.*

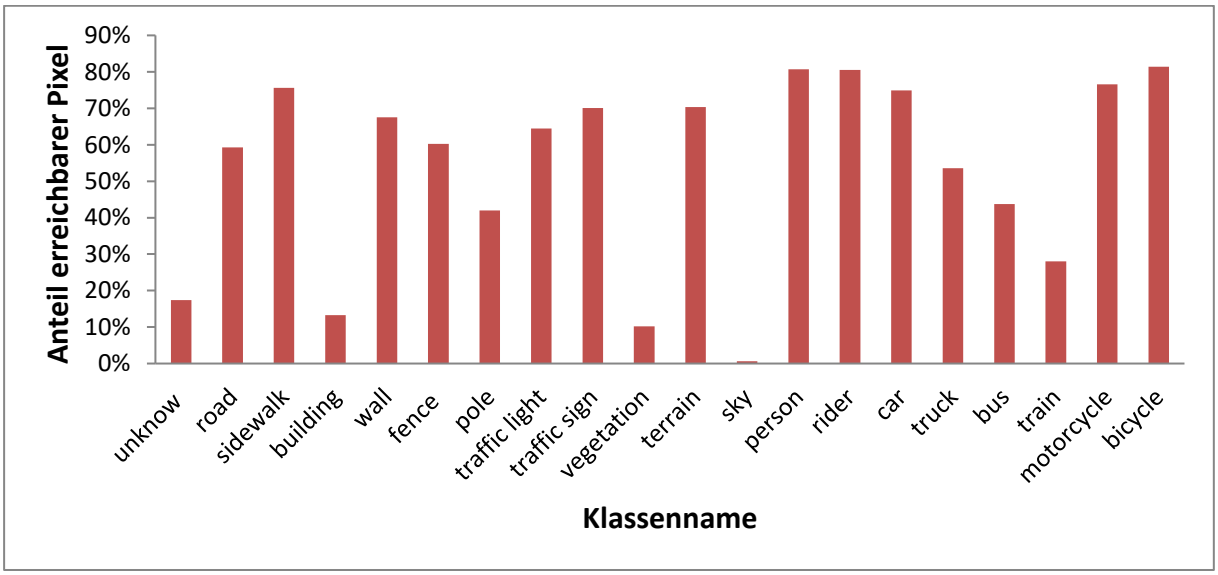


Abbildung 5-14: Anteil von erreichbaren Ground-Truth-Pixeln im Vergleich zu allen Ground-Truth-Pixeln. Die Erreichbarkeitsmenge des Ego-Fahrzeugs wurde mit $t = [0\,s, 3\,s]$, $v = 50\,km/h$, $a = 4\,m/s^2$ *und* $\psi = 90°$ *berechnet.*

Ähnlich wie bei der Evaluation des *FCN-8s*-Modells für alle Pixel, wurde die Konfusionsmatrix des *FCN-8s*-Modells unter Betrachtung der Erreichbarkeit von Pixeln generiert. Die Abbildung 5-15 stellt diese Konfusionsmatrix dar, wobei die Erreichbarkeitsmenge des Ego-Fahrzeugs mit $\mathrm{t} = [0\,\mathrm{s}, 3\,\mathrm{s}]$, $\mathrm{v} = 50\,\mathrm{km/h}$, $\mathrm{a} = 4\,\mathrm{m/s^2}$ und $\psi = 90°$ berechnet wurde. Folgende Eigenschaften ließen sich anhand dieser Abbildung feststellen:

1. Die Klassen, die zwar öfter in den Trainingsdaten vorkamen, aber eine kleine Ausdehnung in den Bildern hatten wie *pole, person, traffic light* und *traffic sign,* zeigten eine deutliche Verbesserung der *IOU*-Werte, da sich diese näher an dem Ego-Fahrzeug befanden. Beispielsweise wurden unter Beachtung der Erreichbarkeit 89,5 % der *Ground-Truth*-Pixel der Klasse *person* richtig segmentiert. Dies entsprach einer relativen Verbesserung von 9 % im Vergleich zu dem Fall, in dem alle Pixel betrachtet wurden.
2. Die Hintergrundklassen *sky, building* und *vegetation*, die unter Betrachtung der Relevanz von Pixeln schlechter segmentiert wurden im Vergleich zu dem Fall, wo alle Pixel betrachtet wurden, zeigten auch hier eine Verschlechterung der richtig segmentierten *Ground-Truth*-Pixel. Die Klasse *vegetation*, die laut Abbildung 5-14 nur 10,2 % ihrer *Ground-Truth*-Pixel erreichbar hatte, beinhaltete 88,4 % der *Ground-Truth*-Pixel, die richtig segmentiert wurden. Im Vergleich zu dem Fall, wo alle Pixel betrachtet wurden, wurde eine relative Verschlechterung von 6,8 % festgestellt. Diese Verschlechterung wurde hauptsächlich durch die Klasse *terrain* verursacht. Während in dem Fall, in dem alle Pixel betrachtet wurden, nur 0,54 % der Pixel der Klasse *vegetation* als *terrain* falsch segmentiert wurden, wurden in dem Fall mit erreichbaren Pixeln 3,7 % der Pixel der *Ground-Truth*-Klasse *vegetation* als *terrain* falsch segmentiert. Die Anzahl der Pixel der *Ground-Truth*-Klasse *vegetation*, die als *terrain* falsch segmentiert wurde, stieg also um 585 %. Darüber hinaus wurden 15 % der *Ground-Truth*-Pixel der Klasse *terrain* falsch als *vegetation* segmentiert. Zusammengefasst hatte die Klasse *vegetation* im Vergleich zu der geringeren Anzahl an erreichbaren Pixeln viele falsche Segmentierungen. Diese führten zu einer Verschlechterung des *IOU*-Werts der Klasse *vegetation*. Ähnlich ließen sich die Verschlechterungen der *IOU*-Werte der Klassen *building* und *sky* erklären.
3. Bei der Klasse *unknow*, waren nur 17,4 % der Pixel erreichbar, während der *IOU*-Wert sich um ca. 200 % verschlechterte. Dass die Pixel der Klasse *unknow* falsch als *road* und *sidewalk* segmentiert wurden, wurde schon erläutert. Allerdings fanden die falschen Klassifikationen in der Erreichbarkeitsmenge statt, wo die Klasse *unknow* insgesamt wenig *Ground-Truth*-Pixel hatte. Damit wurde der Einfluss der falschen Segmentierungen verschärft.

		road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	unknow
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
road	0	96.4	1.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.3	0.0	0.0	0.0	0.0	0.0	1.2
sidewalk	1	3.4	91.0	0.3	0.1	0.2	0.7	0.0	0.0	0.2	0.4	0.0	0.5	0.0	0.3	0.0	0.0	0.0	0.0	0.3	2.4
building	2	0.1	1.7	86.0	0.8	0.6	1.9	0.0	0.2	1.6	0.0	0.1	1.3	0.1	0.6	0.2	0.0	0.0	0.1	0.6	3.9
wall	3	4.3	6.1	9.3	50.7	7.7	1.3	0.0	0.2	9.9	0.6	0.0	1.1	0.1	0.4	1.2	0.0	0.1	0.0	0.8	6.0
fence	4	0.3	2.3	13.0	2.4	60.9	2.1	0.0	1.6	3.7	0.4	0.0	1.0	0.1	0.8	1.2	0.1	0.1	0.2	2.0	7.7
pole	5	0.3	7.5	5.8	0.4	1.1	68.4	0.3	0.4	3.6	1.3	0.0	2.0	0.1	1.6	0.2	0.0	0.0	0.2	1.6	5.2
traffic light	6	0.0	0.0	5.0	0.0	0.0	2.2	87.7	0.5	2.6	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.4
traffic sign	7	0.2	0.4	4.6	0.1	1.0	1.9	0.3	85.1	1.9	0.0	0.1	0.4	0.0	0.9	0.0	0.0	0.0	0.0	0.2	2.9
vegetation	8	0.2	1.5	2.0	0.2	0.5	0.9	0.0	0.1	88.4	3.7	0.0	0.5	0.1	0.5	0.0	0.0	0.0	0.0	0.4	1.0
terrain	9	1.4	6.9	0.2	1.0	0.6	0.8	0.0	0.0	15.0	67.8	0.0	0.1	0.1	0.3	0.0	0.0	0.0	0.0	0.4	3.4
sky	10	0.0	0.0	0.9	0.0	0.0	0.4	0.6	0.0	0.3	0.0	97.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3
person	11	1.0	1.9	1.9	0.1	0.1	0.5	0.0	0.0	0.4	0.0	0.0	89.4	2.0	1.0	0.0	0.0	0.0	0.1	0.9	0.5
rider	12	0.5	0.8	1.3	0.1	0.2	0.3	0.0	0.0	0.9	0.0	0.0	12.7	70.8	1.6	0.0	0.0	0.0	1.7	8.4	0.6
car	13	0.5	0.1	0.2	0.0	0.0	0.1	0.0	0.1	0.2	0.0	0.0	0.2	0.0	97.7	0.2	0.1	0.0	0.1	0.2	0.2
truck	14	1.3	0.4	1.1	0.1	0.2	0.2	0.1	0.1	0.4	0.0	2.4	0.7	0.1	21.0	61.4	5.8	0.2	0.1	0.0	4.5
bus	15	0.6	0.2	0.5	0.1	0.5	1.0	0.0	0.5	0.4	0.0	0.0	0.1	0.3	10.9	6.8	76.2	1.2	0.0	0.2	0.3
train	16	0.7	0.3	8.7	0.0	1.1	0.4	0.0	0.6	0.0	0.0	0.0	0.1	0.0	0.4	3.8	24.1	58.9	0.1	0.2	0.5
motorcycle	17	0.5	3.1	1.0	0.0	1.8	1.4	0.0	0.2	0.4	0.0	0.0	3.5	4.0	5.2	0.0	0.1	0.1	71.7	5.3	1.7
bicycle	18	0.3	2.0	1.4	0.0	0.2	0.9	0.0	0.0	0.5	0.3	0.0	1.4	4.4	1.0	0.0	0.0	0.0	1.5	85.1	0.8
unknow	19	26.8	22.4	5.5	0.8	1.7	2.1	0.1	0.7	2.3	2.4	0.1	2.2	0.2	2.2	0.5	0.3	0.1	0.3	1.5	28.1

Abbildung 5-15: Konfusionsmatrix des FCN-8s-Modells auf den Klassen des DCS-Evaluationsdatensatzes mit dem Fokus auf erreichbaren Pixeln. Die Erreichbarkeitsmenge des Ego-Fahrzeugs wurde mit $t = [0\,s, 3\,s]$, $v = 50\,km/h$, $a = 4\,m/s^2$ *und* $\psi = 90°$ *berechnet. Die Ground-Truth- und die Segmentierungsklassen sind jeweils in den Zeilen und Spalten angeordnet. Die Zahlen sind in Prozent angegeben.*

Zusammengefasst wurden durch die Betrachtung von erreichbaren Pixeln die *IOU*-Werte einiger Klassen verbessert. Gleichzeitig hatten sich die *IOU*-Werte einiger Klassen verschlechtert. Folgende Unterschiede wurden festgestellt:

1. Klassen mit kleiner Ausdehnung in den Bildern, die aufgrund der Nähe am Fahrzeug besser sichtbar waren, wurden deutlich besser segmentiert (siehe bspw. die Klasse *person*).
2. Ebenso wurden Klassen mit einer größeren Ausdehnung, die anteilig in der Erreichbarkeitsmenge ausreichend vorhanden waren, ebenfalls leicht verbessert (siehe bspw. *road* und *sidewalk*).
3. Klassen, die zwar unter Betrachtung aller Pixel oft in den Bildern vorkamen, aber wenig in der Erreichbarkeitsmenge vorhanden waren, wurden schlechter segmentiert. Diese Klassen hatten in der Regel eine große Ausdehnung im Hintergrund (siehe *building, vegetation*). In der Erreichbarkeitsmenge wurden diese Klassen aufgrund der Ähnlichkeit mit anderen Klassen und den eingeschränkten rezeptiven Feldern der Neuronen falsch segmentiert.

4. Klassen, die unter Betrachtung aller Pixel selten in den Bildern vorkamen, aber dafür groß ausgedehnt waren, wurden aufgrund der Nähe zur Kamera in der Erreichbarkeitsmenge und den eingeschränkten rezeptiven Feldern der Neuronen ebenfalls falsch segmentiert.

5.6.2.3 Evaluation der erreichbaren Pixel anhand der Kategorien

Analog zur Evaluation der Klassen der semantischen Segmentierung für relevante Pixel wurden bei der Evaluation für die erreichbaren Pixel die *IOU*-Werte über alle Kategorien anhand des *DCS*-Validierungsdatensatzes berechnet. Dabei wurden lediglich die Pixel betrachtet, die erreicht wurden. Die Abbildung 5-16 stellt den Verlauf der *mIOU*-Werte über alle Kategorien auf dem *DCS*-Validierungsdatensatz dar, wobei die Erreichbarkeitsmenge mit den folgenden Parametern berechnet wurde: $t = [0\,\mathrm{s}, 3\,\mathrm{s}]$, $\mathrm{v} = \{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, \infty\}\,\mathrm{km/h}$, $\mathrm{a} = 4\,\mathrm{m/s^2}$ und $\psi = 90°$. Für den Fall $\mathrm{v} = \infty$ wurden alle Pixel bei der Evaluation betrachtet (siehe Abschnitt 5.6.1). Diese Abbildung zeigte, dass die *mIOU*-Werte der Kategorien für relevante Pixel um bis zu 10,6 % schlechter waren im Vergleich zu dem Fall, wo alle Pixel betrachtet wurden (Geschwindigkeit $\mathrm{v} = \infty$).

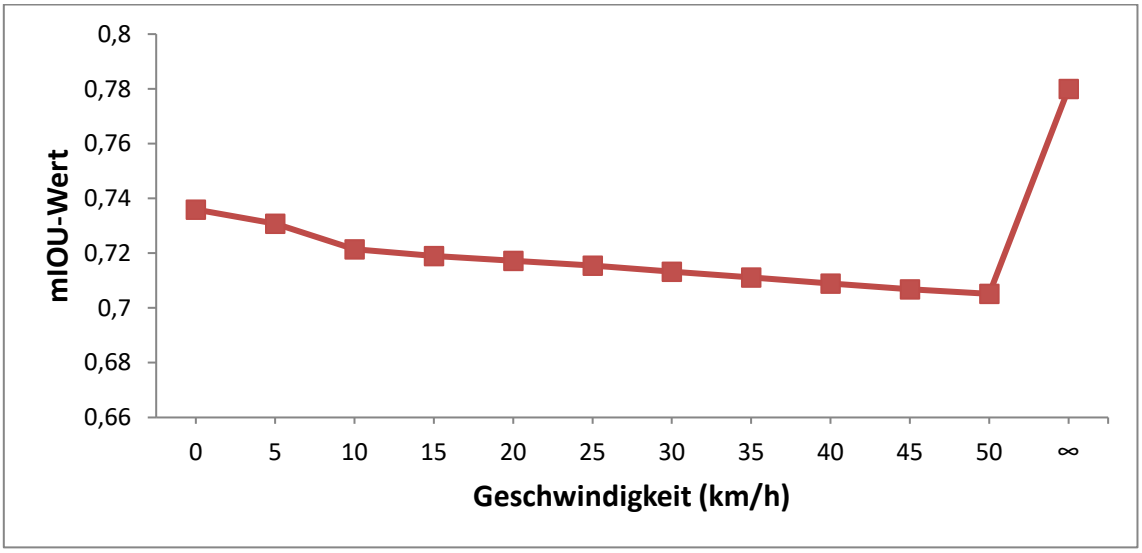


Abbildung 5-16: Verlauf der mIOU-Werte über alle Kategorien auf dem DCS-Validierungsdatensatz für die erreichbaren Pixel. Die Erreichbarkeitsmenge des Ego-Fahrzeugs wurde mit $t = [0\,s, 3\,s]$, $v = \{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, \infty\}\,km/h$, $a = 4\,m/s^2$ *und* $\psi = 90°$ *berechnet.*

Eine detaillierte Darstellung der *IOU*-Werte für die einzelnen Kategorien zeigte in der Abbildung 5-17, dass die Verschlechterung der *mIOU*-Werte der Kategorien für den Fall, dass nur relevante Pixel betrachtet wurden, durch die Verschlechterung der *IOU*-Werte der zu den Kategorien entsprechenden Klassen verursacht wurde. So wurden die Kategorien *void* und *sky* deutlich schlechter, da die zu diesen Kategorien passenden Klassen *unknow* und *sky* deutlich schlechtere *IOU*-Werte für relevante Pixel hatten im Vergleich zu dem Fall, in dem alle Pixel betrachtet wurden. Analog führte die Verschlechterung der *IOU*-Werte der Klassen *building* und *vegetation* jeweils zu einer Senkung der *IOU*-Werte der Kategorie *construction* und *nature*. Die Kategorien *human, object* und *flat* blieben entweder konstant oder verbesserten leicht ihre *IOU*-Werte. Die Kategorie *human* z. B., die die Klassen *person* und *rider* enthielt, erreichte mit dem *IOU*-Wert von 80,25 % bei der Geschwindigkeit $\mathrm{v} = 10\,\mathrm{km/h}$ eine relative Verbesserung von 16,8 %. Die Kategorie *flat* hatte den besten *IOU*-Wert von 93 %.

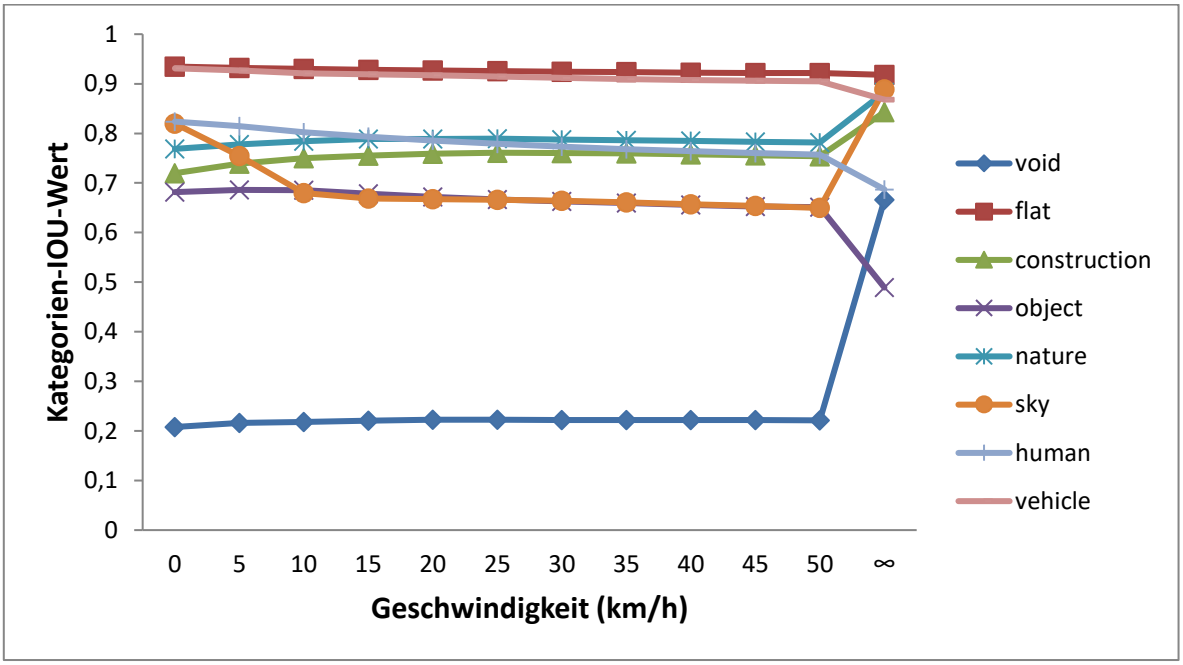


Abbildung 5-17: Verlauf der IOU-Werte einzelner Kategorien auf dem DCS-Validierungsdatensatz für die erreichbaren Pixel. Die Erreichbarkeitsmenge des Ego-Fahrzeugs wurde mit $t = [0\ s, 3\ s]$, $v = \{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, \infty\}\ km/h$, $a = 4\ m/s^2$ *und* $\psi = 90°$ *berechnet.*

Die Abbildung 5-18 stellt die Konfusionsmatrix des *FCN-8s*-Modells unter Betrachtung der Erreichbarkeit von Pixeln für die Kategorien des *DCS*-Datensatzes dar. Diese Abbildung verdeutlichte die Verschlechterung der Kategorien *void, construction, sky* und *nature*. In der Kategorie *void* bspw. befanden sich ca. 49,2 % der falschen Segmentierungen der Kategorie *flat*, wobei die Kategorie *flat* die Klassen *road* und *sidewalk* enthielt. Dies entsprach einer Verschlechterung von 230 % der falschen Segmentierungen im Vergleich zu dem Fall, bei dem alle Pixel betrachtet wurden. Die beiden Klassen *road* und *sidewalk* wurden in dem oberen Abschnitt schon als Grund für die falsche Klassifikation der Klasse *void* beschrieben. Die Abbildung 5-18 verstärkte diese Begründung noch einmal.

		void	flat	construction	object	nature	sky	human	vehicle
		0	1	2	3	4	5	6	7
void	0	28.1	49.2	7.9	2.8	4.7	0.1	2.4	4.9
flat	1	1.4	97.6	0.1	0.1	0.2	0.0	0.2	0.4
construction	2	4.6	3.0	83.5	2.3	3.1	0.1	1.3	2.1
object	3	3.9	4.2	6.4	78.3	3.6	0.1	1.2	2.3
nature	4	1.6	3.8	2.5	0.9	89.8	0.0	0.5	0.9
sky	5	0.3	0.0	0.9	1.0	0.3	97.5	0.0	0.0
human	6	0.5	2.7	2.0	0.6	0.5	0.0	90.3	3.4
vehicle	7	0.4	0.8	0.5	0.3	0.3	0.1	0.9	96.6

Abbildung 5-18: Konfusionsmatrix des FCN-8s-Modells auf den Kategorien des DCS-Evaluationsdatensatzes mit dem Fokus auf erreichbaren Pixeln. Die Erreichbarkeitsmenge des Ego-Fahrzeugs wurde mit $t = [0\,s, 3\,s]$, $v = 50\,km/h$, $a = 4\,m/s^2$ und $\psi = 90°$ berechnet. Die Ground-Truth- und die Segmentierungskategorien sind jeweils in den Zeilen und Spalten angeordnet. Die Zahlen sind in Prozent angegeben.

5.6.2.4 Evaluation der für das Ego-Fahrzeug relevanten Verkehrszeichen und –Ampeln

Im Kontext des automatisierten Fahrens müssen Verkehrszeichen und –Ampeln (im *DCS*-Datensatz *traffic light* und *traffic sign* genannt) besonders betrachtet werden, da diese die Grundlage für die Verkehrsregeln darstellen. Die Missachtung dieser Infrastrukturelemente könnte zu einer kritischen Situation führen. In dieser Arbeit wurde die Frage gestellt, wie gut das *FCN-8s*-Modell relevante Verkehrszeichen und –Ampeln erkannte. Dafür wurde die Evaluation auf relevante Verkehrszeichen und –Ampeln eingeschränkt. Eine Verkehrsampel ist relevant, wenn diese einer Straße zugeordnet ist, auf der sich das Ego-Fahrzeug befindet oder in der nahen Zukunft fahren wird. Darüber hinaus muss die Ampel so ausgerichtet sein, dass ihr Zustand vom Ego-Fahrzeug aus erkennbar ist. Diese Definition gilt auch für Verkehrszeichen. Ferner werden Verkehrszeichen ausgeschlossen, die lediglich zur Information dienen (z. B. Straßennamen, Parkschilder etc.).

Die Abbildung 5-19 stellt den Verlauf der *IOU*-Werte für die relevanten und erreichbaren Pixel der Klassen *traffic light* und *traffic sign* auf dem *DCS*-Validierungsdatensatz dar, wobei die Erreichbarkeitsmenge mit den folgenden Parametern berechnet wurde: $t = [0\,\mathrm{s}, 3\,\mathrm{s}]$, $\mathrm{v} = \{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, \infty\}\,\mathrm{km/h}$, $\mathrm{a} = 4\,\mathrm{m/s^2}$ und $\psi = 90°$. Für den Fall $\mathrm{v} = \infty$ wurden alle Pixel bei der Evaluation betrachtet. Diese Abbildung zeigte, dass die *mIOU*-Werte der Klassen für relevante und erreichbare Pixel deutlich verbessert wurden im Vergleich zu dem Fall, der alle Pixel oder nur erreichbare Pixel betrachtete. So zeigte die Klasse *traffic light* mit einem *IOU*-Wert zwischen 96,3 % und 98,5 % in dem Fall, bei dem nur relevante und erreichbare Pixel betrachtet wurden, eine relative Verbesserung von bis zu 101 % im Vergleich zu dem Fall, wo weder die Relevanz noch die Erreichbarkeit betrachtet wurden. Die Klasse *traffic sign* zeigte eine etwas geringere relative Verbesserung von 59,7 % mit *IOU*-Werten zwischen 91,8 % und 93,6 %. Wie in den oberen Abschnitten erläutert, waren erreichbare Pixel der Klassen *traffic light* und *traffic sign* näher an dem Fahrzeug und somit besser zu segmentieren waren. Darüber hinaus waren relevante *traffic light* und *traffic sign* nicht nur näher an dem Fahrzeug, sondern auch so ausgerichtet, dass ihr vorderer Teil für die Kamera gut sichtbar war. Somit waren texturreiche Informationen für die Segmentierung verfügbar.

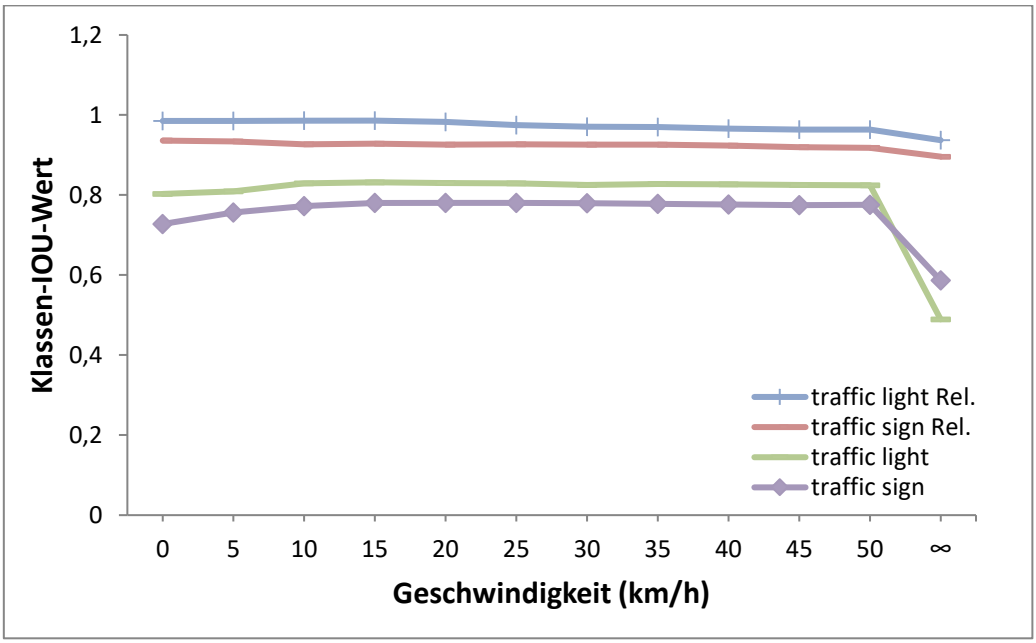


Abbildung 5-19: Verlauf der IOU-Werte der Klassen traffic light und traffic sign auf dem DCS-Validierungsdatensatz für die erreichbaren und relevanten Pixel. Die Erreichbarkeitsmenge des Ego-Fahrzeugs wurde mit $t = [0\,s, 3\,s]$, $v = \{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, \infty\}\,km/h$, $a = 4\,m/s^2$ *und* $\psi = 90°$ *berechnet.*

5.6.3 Evaluation der semantischen Segmentierung als Grundlage für die Schätzung von Freiraum und Hindernissen

Im Kontext des automatisierten Fahrens ist neben der semantischen Segmentierung die Erkennung von Freiraum und Hindernissen von großer Bedeutung. Dabei spielen Hindernisse, die unmittelbar den Freiraum des Ego-Fahrzeugs eingrenzen, eine wichtige Rolle. Im Zusammenhang mit dem *DCS*-Datensatz, gehören alle Pixel, die als Straße segmentiert werden, zum Freiraum des Ego-Fahrzeugs. Alle anderen Klassen außer *sky* sind als Hindernisse zu betrachten. Es entstehen also drei Kategorien: die Freiraum-, *sky*- und die Hinderniskategorien. Basierend auf diesen Kategorien und der Tiefenkarte wurde in dieser Arbeit der Freiraum des Ego-Fahrzeugs als *Occupancy Grid Map* (*OGM*) geschätzt. Die *OGM*-Karte unterteilt das Ego-Fahrzeugumfeld in ein zweidimensionales Gitter mit rechteckigen Zellen. Jede Zelle der *OGM*-Karte enthält die Wahrscheinlichkeit, dass die Zelle frei oder von einem Hindernis belegt ist. Ein hoher Wert der Belegungswahrscheinlichkeit bedeutet, dass die Zelle belegt ist, während ein niedriger Wert darauf hinweist, dass die Zelle frei ist [159]. Die *OGM*-Karte ist weit verbreitet und wird oft in der Robotik angewendet [160].

5.6.3.1 Berechnung der *OGM*-Karte anhand der semantischen Segmentierung

Zur Berechnung der *OGM*-Karte mithilfe der semantischen Segmentierung wurde in dieser Arbeit der Ansatz von Badino et al. [161] angepasst. Die Abbildung 5-20 stellt die Schritte zur Schätzung der *OGM*-Karte dar. Zunächst wurde das Inputbild mit dem *FCN-8s*-Modell segmentiert (siehe Schritte eins und zwei der Abbildung 5-20). Danach wurde das segmentierte Bild mithilfe der Disparitätskarte in die *U*-Disparitätskarte akkumuliert (siehe Schritte drei und sieben der Abbildung 5-20). Zellen der *U*-Disparitätskarte, die Hindernisse mit einer Höhe über einem gegebenen Schwellenwert hatten, wurden als Hindernis markiert (siehe schwarze Regionen auf der *U*-Disparitäts-*OGM*-Karte der Abbildung 5-20). Der Schwellenwert für die Höhe von Hindernissen wurde auf 0 cm für die Klasse *sidewalk* und 25 cm für alle anderen Klassen der Kategorie Hindernis gesetzt. Es wurden nur Pixel betrachtet, die für die gegebenen dynamischen Parameter des Ego-Fahrzeugs erreichbar waren. In der Abbildung 5-20 wurden für die Berechnung der Erreichbarkeitsmenge die Parameter $\mathrm{t} = [0\,\mathrm{s}, 3\,\mathrm{s}]$, $\mathrm{v} = 50\,\mathrm{km/h}$, $\mathrm{a} = 4\,\mathrm{m/s^2}$ und $\psi = 90°$ verwendet (siehe Schritte vier bis sechs der Abbildung 5-20). Die

Regionen, die sich vor Hindernissen in der *U*-Disparitätskarte befanden, wurden als frei markiert (siehe weiße Regionen auf der *U*-Disparitäts-*OGM*-Karte der Abbildung 5-20). Alle Regionen nach den Hindernissen wurden als unbekannt markiert (siehe graue Regionen auf der *U*-Disparitäts-*OGM*-Karte der Abbildung 5-20). Der Output war dann die *U*-Disparitäts-*OGM*-Karte akkumuliert (siehe Schritt acht der Abbildung 5-20). Diese *U*-Disparitäts-*OGM*-Karte wurde dann in den X-Z-kartesischen Raum projiziert, um die kartesische *OGM*-Karte zu generieren (siehe Schritte neun und zehn der Abbildung 5-20).

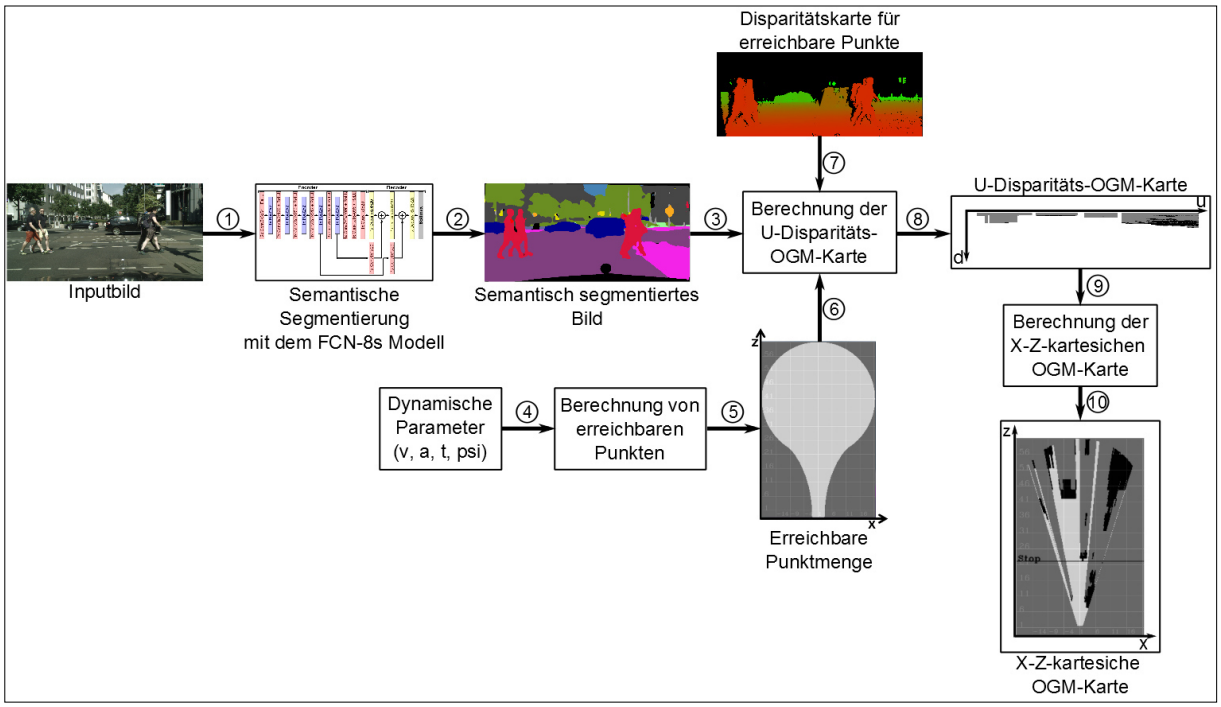


Abbildung 5-20: Schematische Darstellung der Schritte zur Schätzung der OGM-Karte mithilfe der semantischen Segmentierung. Das Inputbild und die Disparitätskarte stammen aus dem DCS-Datensatz [81]. Die Erreichbarkeitsmenge des Ego-Fahrzeugs wurde mit $t = [0\,s, 3\,s]$, $v = 50\,km/h$, $a = 4\,m/s^2$ und $\psi = 90°$ berechnet. Der Schwellenwert für die Höhe von Hindernissen wurde auf 0 cm für die Klasse sidewalk und 25 cm für alle anderen Klassen gesetzt.

Die Abbildung 5-24 und die Abbildung 5-25 stellen einige Ergebnisse der Berechnung der *OGM*-Karte mithilfe der semantischen Segmentierung auf dem *DCS*-Validierungsdatensatz dar. Die Ergebnisse für einzelne Bilder sind spaltenweise angeordnet. Die Zeilen enthalten jeweils das Inputbild (erste Zeile), die *Ground Truth* der semantischen Segmentierung (zweite Zeile), die Segmentierung mit dem *FCN-8s*-Modell (dritte Zeile), die Tiefenkarte für erreichbare Pixel (vierte Zeile), die kartesische *OGM*-Karte mit der *Ground Truth* der semantischen Segmentierung als Input (fünfte Zeile) und die kartesische *OGM*-Karte mit der semantischen Segmentierung des *FCN-8s*-Modells als Input (sechste Zeile). Die Inputbilder, der *Ground Truth* der semantischen Segmentierung und die Disparitätskarten stammen aus dem *DCS*-Datensatz [81]. Die Erreichbarkeitsmenge des Ego-Fahrzeugs wurde mit $\mathrm{t} = [0\,\mathrm{s}, 3\,\mathrm{s}]$, $\mathrm{v} = 50\,\mathrm{km/h}$, $\mathrm{a} = 4\,\mathrm{m/s^2}$ und $\psi = 90°$ berechnet. Der Schwellenwert für die Höhe von Hindernissen wurde auf 0 cm für die Klasse *sidewalk* und 25 cm für alle anderen Klassen der Hinderniskategorien gesetzt. In der Abbildung 5-24 sind die *OGM*-Karten mit der semantischen Segmentierung des *FCN-8s*-Modells als Input (sechste Zeile) sehr ähnlich den *OGM*-Karten mit der *Ground Truth* der semantischen Segmentierung als Input (fünfte Zeile). Die Erklärung dazu ist, dass das *FCN-8s*-Modell sehr wenige Fehler dort hatte, wo die erreichbaren Pixel der Kategorien Freiraum, *sky* und Hindernis verwechselt wurden. In der dritten Spalte wurden bspw. viele Pixel der Klasse *fence* falsch als *building* segmentiert. Da aber sowohl die Klasse *building* als auch die Klasse *fence* zu den Hindernissen gehörte, hatte diese falsche Segmentierungen keinen Einfluss auf die *OGM*-Karte. Anders als in der Abbildung 5-24 wurden in der Abbildung 5-25 Beispiele dargestellt, bei denen sich die *OGM*-Karten mit der semantischen Segmentierung des *FCN-8s*-Modells als Input von den *OGM*-Karten mit dem

Ground Truth der semantischen Segmentierung als Input unterschieden. Diese Unterschiede traten auf, wenn das *FCN-8s*-Modell Pixel der Freiraum-Kategorie mit den Pixeln der Hindernis-Kategorie verwechselte. So segmentierte das *FCN-8s*-Modell, in dem Bild die erste und dritte Spalte, Teile der Klasse *road* als *sidewalk*, was zur Verschlechterung der *OGM*-Karten führten. In der zweiten Spalte wurde die Klasse *unknow* falsch als *sidewalk* segmentiert.

5.6.3.2 Evaluation der mit der semantischen Segmentierung geschätzten *OGM*-Karte anhand der *Mean-Squared-Error*-Metrik

Zur Evaluation der mit der semantischen Segmentierung geschätzten *OGM*-Karte wurde der mittlere quadratische Fehler (*MSE*: *Mean Squared Error*) verwendet.

$$MSE(OGM_{GT}, OGM_{FCN}) = \frac{1}{m * n}\sum_{i=1}^{m}\sum_{j}^{n}\left(OGM_{GT}(i,j) - OGM_{FCN}(i,j)\right)^2 \qquad (26)$$

$OGM_{GT} \in [0,1]^{m \times n}$ und $OGM_{FCN} \in [0,1]^{m \times n}$ sind jeweils die *OGM*-Karten mit der *Ground Truth* der semantischen Segmentierung und der semantischen Segmentierung des *FCN-8s*-Modells als Input.

Die Abbildung 5-21 stellt die mittleren $MSE(OGM_{GT}, OGM_{FCN})$-Werte über alle Daten des *DCS*-Validierungsdatensatzes dar, wobei erreichbare Pixel mit den Parametern $t = [0\,\mathrm{s}, 3\,\mathrm{s}]$, $\mathrm{v} = \{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}\,\mathrm{km/h}$, $\mathrm{a} = 4\,\mathrm{m/s^2}$ und $\psi = 90°$ berechnet wurden. Diese mittleren *MSE*-Werte lagen zwischen 2,7 % und 5,7 %. Je größer die Geschwindigkeit des Ego-Fahrzeugs war, desto größer war der *MSE*-Wert. Dies ließ sich dadurch erklären, dass mit steigender Geschwindigkeit die erreichbare Menge größer wurde und das *FCN-8s*-Modells mehr Fehler bei fernen als bei nahen Pixeln machte.

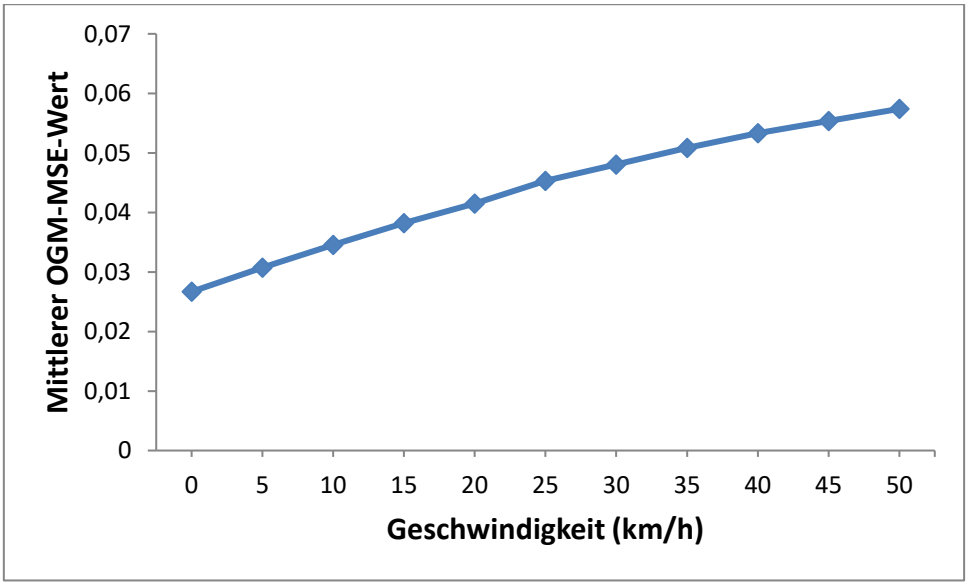


Abbildung 5-21: Verlauf der mittleren $MSE(OGM_{GT}, OGM_{FCN})$-Werte über alle Daten des DCS-Validierungsdatensatzes. Die erreichbaren Pixel wurden mit den Parametern $t = [0\,s, 3\,s]$, $v = \{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}\,km/h$, $a = 4\,m/s^2$ und $\psi = 90°$ berechnet.

Die in der Abbildung 5-24 dargestellten *OGM* hatten *MSE*-Werte von 4,4 % (erste Spalte), 1,4 % (zweite Spalte) und 3,9 % (dritte Spalte), die unter dem mittleren *MSE*-Wert 5,7 % für die entsprechende Erreichbarkeitsmenge lagen. Wie schon im Abschnitt 5.6.3.2 erläutert, waren diese niedrigen *MSE*-Werte durch die gute Segmentierung der Kategorien Freiraum und Hindernis durch das *FCN-8s*-Modell zu erklären.

Verglichen mit den *MSE*-Werten der Abbildung 5-24 lagen die *MSE*-Werte der Abbildung 5-25 mit jeweils 27 % (erste Spalte), 23 % (zweite Spalte) und 12,5 % (dritte Spalte) deutlich über dem mittleren *MSE*-Wert von 5,7 %. Solche Fälle traten allerdings selten auf. Hier wurden viele Pixel der Kategorie Freiraum als Hindernis falsch segmentiert und umgekehrt.

5.6.3.3 Evaluation der mit der semantischen Segmentierung geschätzten *OGM*-Karte anhand der *Time Headway* (THW)

Während die Evaluation der *OGM*-Karte mit der *MSE*-Metrik alle Regionen der *OGM*-Karte gleich bewertete, spielten im Kontext des automatisierten Fahrens Hindernisse, die unmittelbar den Freiraum des Ego-Fahrzeugs eingrenzten oder die Trajektorie des Ego-Fahrzeugs kreuzen konnten, eine besondere Rolle. Die falsche Detektion dieser Hindernisse könnte zu einer Kollision mit dem Ego-Fahrzeug führen. Um diese besonderen Hindernisse zu betrachten, wurde die *Time-Headway*-(*THW*-)Metrik zur Evaluation verwendet. Zur Schätzung des *THW*-Wertes wurde die Zeit, die das Ego-Fahrzeug in der *OGM*-Karte zum Erreichen des ersten Hindernisses brauchte, in der Fahrtrichtung berechnet. Hier wurde davon ausgegangen, dass das Ego-Fahrzeug geradeaus fahren würde, da der Datensatz die Trajektorie des Ego-Fahrzeugs nicht bereitstellte. Lediglich GPS-Informationen des Ego-Fahrzeugs waren für einzelne *Frames* verfügbar. Darüber hinaus wurden Hindernisse betrachtet, die sich in einem Korridor von zwei Metern in der Fahrtrichtung des Ego-Fahrzeugs befanden. Die Entfernung zum ersten Hindernis in der Fahrtrichtung des Ego-Fahrzeugs ist in den *OGM*-Karten der Abbildung 5-24 und Abbildung 5-25 durch eine schwarze horizontale Stopplinie gekennzeichnet. Für jedes Inputbild wurde jeweils der relative $\Delta THW_{rel}(OGM_{GT}, OGM_{FCN})$-Wert (siehe Gleichung (28)) und der absolute $\Delta THW_{abs}(OGM_{GT}, OGM_{FCN})$-Wert (siehe Gleichung (27)) zwischen dem $THW(OGM_{FCN})$-Wert und $THW(OGM_{GT})$-Wert berechnet. *Der* $THW(OGM_{FCN})$-Wert ist der *THW*-Wert*,* bei dem die *OGM*-Karte mit der semantischen Segmentierung des *FCN-8s*-Modells als Input berechnet wurde, während der $THW(OGM_{GT})$-Wert die *OGM*-Karte mit der *Ground Truth* der semantischen Segmentierung als Input verwendete.

$$\Delta THW_{abs}(OGM_{GT}, OGM_{FCN}) = |THW(OGM_{GT}) - THW(OGM_{FCN})| \tag{27}$$

$$\Delta THW_{rel}(OGM_{GT}, OGM_{FCN}) = \frac{\Delta THW_{abs}(OGM_{GT}, OGM_{FCN})}{THW(OGM_{GT})} \tag{28}$$

Im optimalen Fall sollen sich der $\Delta THW_{rel}(OGM_{GT}, OGM_{FCN})$-Wert und $\Delta THW_{abs}(OGM_{GT}, OGM_{FCN})$-Wert 0 annähern.

In der Abbildung 5-22 sind die mittleren $\Delta THW_{abs}(OGM_{GT}, OGM_{FCN})$-Werte über alle Daten des *DCS*-Validierungsdatensatzes dargestellt, wobei die erreichbaren Pixel mit den Parametern $t = [0\,\mathrm{s}, 3\,\mathrm{s}]$, $\mathrm{v} = \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}\,\mathrm{km/h}$, $\mathrm{a} = 4\,\mathrm{m/s^2}$ und $\psi = 90°$ berechnet wurde. Dabei lagen die mittleren $\Delta THW_{abs}(OGM_{GT}, OGM_{FCN})$-Werte zwischen 16 ms und 48 ms. Dies bedeutet, dass es im Mittel maximal 48 ms Zeitunterschied zwischen $THW(OGM_{GT})$-Werten und $THW(OGM_{FCN})$-Werten gab. Die $\Delta THW_{abs}(OGM_{GT}, OGM_{FCN})$-Werte analog zu den *MSE*-Werten aus Abschnitt 5.6.3.2 stiegen mit der steigenden Geschwindigkeit des Ego-Fahrzeugs an, da die semantische Segmentierung mit dem *FCN-8s*-Modell wegen der steigenden Tiefe der Erreichbarkeitsmenge mehr Fehler machte.

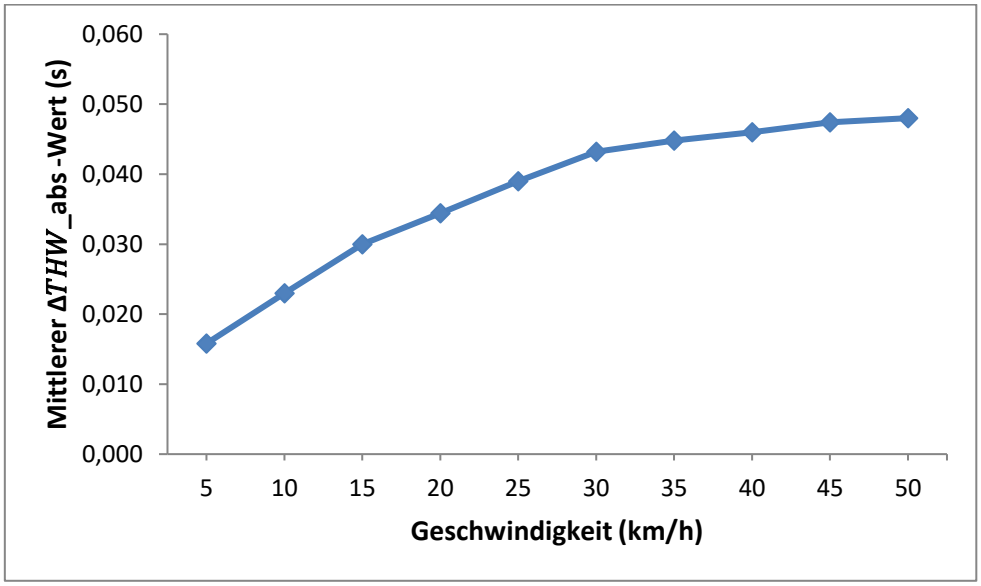


Abbildung 5-22: Verlauf der mittleren $\Delta THW_{abs}(OGM_{GT}, OGM_{FCN})$-Werte über alle Daten des DCS-Validierungsdatensatzes. Die erreichbaren Pixel wurden mit den Parametern $t = [0\,s, 3\,s]$, $v = \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}\,km/h$, $a = 4\,m/s^2$ und $\psi = 90°$ berechnet.

Die Abbildung 5-23 stellt die mittleren $\Delta THW_{rel}(OGM_{GT}, OGM_{FCN})$-Werte über alle Daten des *DCS*-Validierungsdatensatzes dar, wobei erreichbare Pixel mit den Parametern $t = [0\,\mathrm{s}, 3\,\mathrm{s}]$, $\mathrm{v} = \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}\,\mathrm{km/h}$, $\mathrm{a} = 4\,\mathrm{m/s^2}$ und $\psi = 90°$ berechnet wurden. Dabei lagen die mittleren $\Delta THW_{rels}(OGM_{GT}, OGM_{FCN})$-Werte zwischen 1 % und 2,8 % und stiegen mit der steigenden Geschwindigkeit des Ego-Fahrzeugs an.

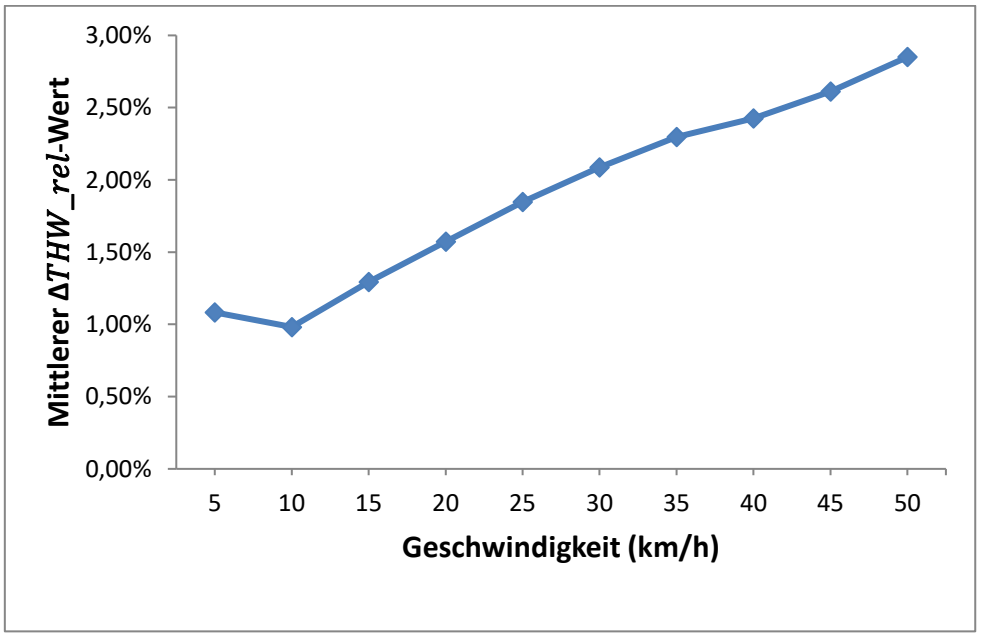
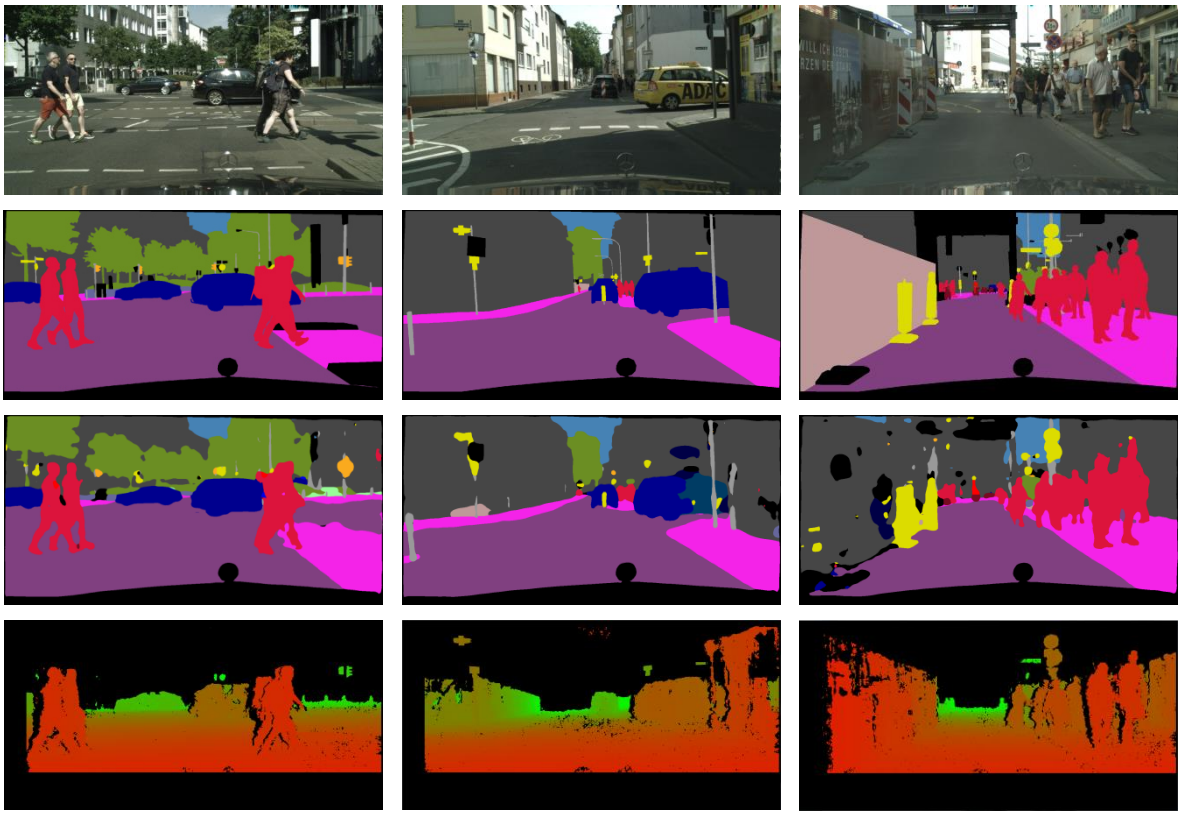


Abbildung 5-23: Verlauf der mittleren $\Delta THW_{rel}(OGM_{GT}, OGM_{FCN})$-Werte über alle Daten des DCS-Validierungsdatensatzes. Die erreichbaren Pixel wurden mit den Parametern $t = [0\,s, 3\,s]$, $v = \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}\,km/h$, $a = 4\,m/s^2$ und $\psi = 90°$ berechnet.

Die $\Delta THW_{abs}(OGM_{GT}, OGM_{FCN})$-Werte der in der Abbildung 5-24 dargestellten *OGM*-Karten lagen alle mit jeweils 0 ms in den Spalten eins bis drei deutlich unter dem mittleren ΔTHW_{abs}-Wert 48 ms für die entsprechende Erreichbarkeitsmenge. Analog lagen auch die ΔTHW_{rel}-Werte mit 0 % deutlich unter dem mittleren Wert von 2,8 %. Die *OGM*-Karten dieser Abbildung zeigten Beispiele, in denen die ΔTHW_{abs}-Werte und ΔTHW_{rel}-Werte sich dem Optimum annäherten. Diese Bilder hatten, wie im Abschnitt 5.6.3.2 beschrieben, auch sehr gute *MSE*-Werte.

Die ΔTHW_{abs}- und ΔTHW_{rel}-Werte der Abbildung 5-25 für die erste Spalte lagen mit jeweils 9 ms und 1 % deutlich unter den mittleren Werten 48 ms und 2,8 %, obwohl der *MSE*-Wert bei 27 % deutlich über dem mittleren *MSE*-Wert von 5,7 % lag. Der Grund dafür war, dass der Fehler der *OGM*-Karte mehr im Hintergrund auftrat, während Hindernisse, die unmittelbar den Freiraum des Ego-Fahrzeugs einschränkten, richtig segmentiert wurden. Dieses Beispiel zeigte, *dass THW-Werte* geeigneter als *MSE*-Werte waren, um die *OGM*-Karte zu evaluieren, da die *THW*-basierte Evaluation vor allem relevante Hindernisse in der Bewertung betrachtete, während die *MSE*-basierte Evaluation auch Hintergrundhindernisse betrachtete, die nicht unmittelbar in Interaktion mit dem Ego-Fahrzeug standen und somit eine geringere Relevanz hatten. Die zweite und dritte Spalte der Abbildung 5-25 zeigen Beispiele, in denen schlechte ΔTHW_{abs}- und ΔTHW_{rel}- Werte erreicht wurden. So lag bspw. der ΔTHW_{abs}-Werte der *OGM*-Karte in der dritten Spalte mit 1,1 s deutlich über dem Mittelwert von 48 ms. Dabei lag auch der *MSE*-Wert dieser *OGM* mit 12,5 % deutlich über dem mittleren *MSE*-Wert von 5,7 %. In diesem Beispiel wurde die Straße auf der rechten Seite des Bildes falsch als Fußgängerweg segmentiert, was den Freiraum des Ego-Fahrzeugs einschränkte. Allerdings ging die Annahme bei der Berechnung des *THW*-Werts davon aus, dass das Ego-Fahrzeug geradeaus fahren würde. Würde aber das Ego-Fahrzeug in diesem Beispiel seiner Spur folgen, hätte der falsch segmentierte Fußgängerweg nicht auf dem Weg des Ego-Fahrzeugs gelegen. Somit wäre der ΔTHW_{abs}-Wert deutlich kleiner.



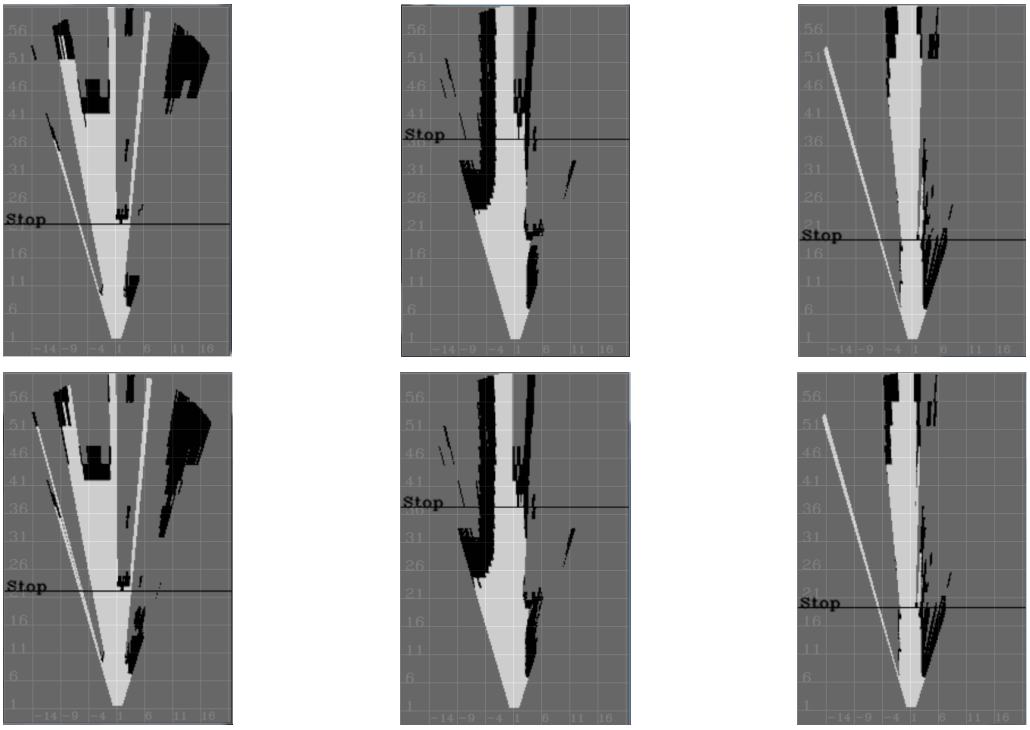
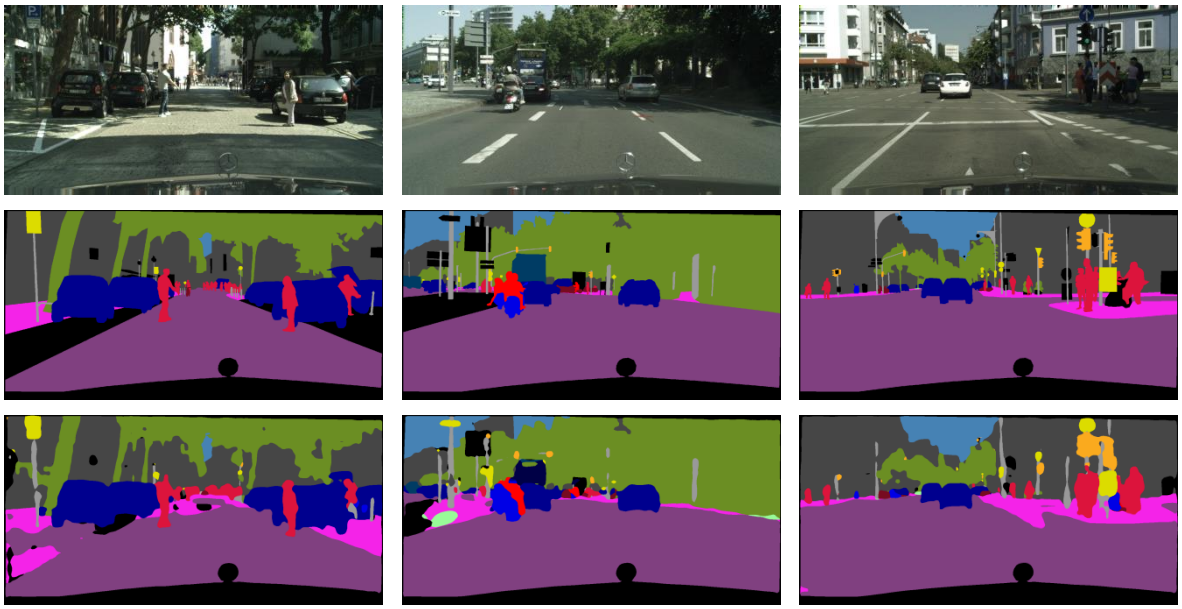


Abbildung 5-24: Beispielergebnisse der gut gelungenen Schätzung der OGM-Karte mithilfe der semantischen Segmentierung für ein paar Bilder aus dem DCS-Validierungsdatensatz. Die Zeilen enthalten jeweils das Inputbild (erste Zeile), die Ground Truth der semantischen Segmentierung (zweite Zeile), die Segmentierung mit dem FCN-8s-Modell (dritte Zeile), die Tiefenkarte für die erreichbaren Pixel (vierte Zeile), die kartesische OGM-Karte mit der Ground Truth der semantischen Segmentierung als Input (fünfte Zeile) und die kartesische OGM-Karte mit der semantischen Segmentierung des FCN-8s-Modells als Input (sechste Zeile). Die Inputbilder, die Ground Truth der semantischen Segmentierung und die Disparitätskarten stammen aus dem DCS-Datensatz [81]. Die Erreichbarkeitsmenge des Ego-Fahrzeugs wurde mit $t = [0\,s, 3\,s]$, $v = 50\,km/h$, $a = 4\,m/s^2$ und $\psi = 90°$ berechnet. Der Schwellenwert für die Höhe von Hindernissen in der OGM-Karte wurde auf 0 cm für die Klasse sidewalk und 25 cm für alle anderen Klassen gesetzt.



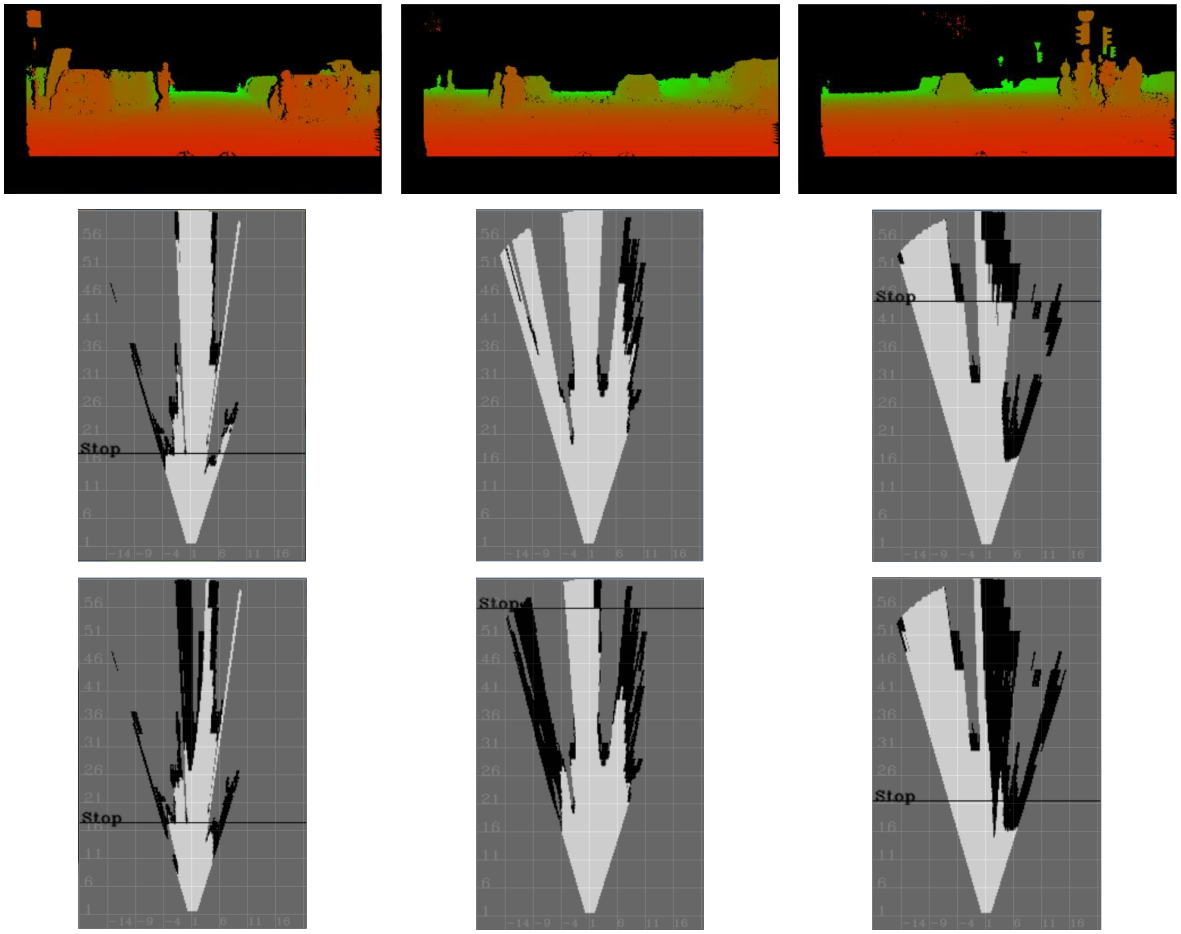


Abbildung 5-25: Beispielergebnisse der schlecht gelungenen Schätzung der OGM-Karte mithilfe der semantischen Segmentierung für ein paar Bilder aus dem DCS-Validierungsdatensatz. Die Zeilen enthalten jeweils das Inputbild (erste Zeile), die Ground Truth der semantischen Segmentierung (zweite Zeile), die Segmentierung mit dem FCN-8s-Modell (dritte Zeile), die Tiefenkarte für die erreichbaren Pixel (vierte Zeile), die kartesische OGM-Karte mit der Ground Truth der semantischen Segmentierung als Input (fünfte Zeile) und die kartesische OGM-Karte mit der semantischen Segmentierung des FCN-8s-Modells als Input (sechste Zeile). Die Inputbilder, die Ground Truth der semantischen Segmentierung und die Disparitätskarten stammen aus dem DCS-Datensatz [81]. Die Erreichbarkeitsmenge des Ego-Fahrzeugs wurde mit $t = [0\ s, 3s\ ]$, $v = 50\ km/h$, $a = 4\ m/s^2$ und $\psi = 90°$ berechnet. Der Schwellenwert für die Höhe von Hindernissen in der OGM-Karte wurde auf 0 cm für die Klasse sidewalk und 25 cm für alle anderen Klassen gesetzt.

5.7 Diskussion

Das *FCN-8s*-Modell, das in dieser Arbeit trainiert und evaluiert wurde, ist ein Meilenstein im Bereich der semantischen Segmentierung. Viele erfolgreiche Modelle basieren auf dieser Idee. In diesem Abschnitt werden die Ergebnisse des *FCN-8s*-Modells bewertet. Darüber hinaus werden Ideen für zukünftige Arbeiten zur Verbesserung der semantischen Segmentierung vorgeschlagen.

5.7.1 Komplexität des *FCN-8s*-Modells

Das in dieser Arbeit verwendete *FCN-8s*-Modell war mit ca. 134,5 Millionen Parametern ein komplexes Modell, obwohl die Tiefe des *Encoders* im Vergleich zum aktuellen TNN deutlich geringer war. Aufgrund der Komplexität des Modells war es nicht möglich, mehr als ein Bild pro Trainingsiteration zu verwenden. Um die Anzahl der Bilder pro Iterationen zu erhöhen, hätte man die Auflösung der Inputbilder reduzieren müssen. Eine Reduzierung der Auflösung der Bilder hätte aber zur Verschlechterung der Segmentierung von klein ausgedehnten Regionen geführt. Das Training auf der *NVIDIA GTX 1080ti GPU* dauerte ca. 47 Stunden für 300 000 Iterationen, wobei das trainierte Modell nach ca. 70 000 Iterationen gemäß der Kreuzvalidierung optimal war. Die Dauer des Trainings nach 70 000 Iterationen betrug ca. elf Stunden und war üblich für das Training von TNN. Der Speicherbedarf des

trainierten *Caffe*-Modells war mit ca. 0,52 GB für TNN normal. Allerdings war so ein Speicherbedarf in eingebetteten Systemen als hoch zu bewerten. Zukünftige Arbeiten sollten den Fokus auf die Reduzierung der Komplexität von TNN legen.

Die Menge an Trainingsbildern war im Vergleich zu anderen TNN gering. Allerdings dauerte die Annotation der Bilder ca. 1,5 Stunden pro Bild [81]. So dauerte die Annotation für die gesamten 5000 Bilder ca. 312,5 Tage. Das Training des *FCN-8s*-Modells konvergierte trotz der geringeren Anzahl an Trainingsbildern, weil das *FCN-8s*-Modell am Anfang des Trainings mit einem *VGG-16*-Modell initialisiert wurde, wobei das *VGG-16*-Modell mit dem *ILSVRC*-2014-Datensatz vortrainiert wurde. Der *ILSVRC*-2014-Datensatz enthält bis zu 1,28 Millionen Bilder [155]. Zukünftige Arbeiten zur semiautomatischen Annotation von Daten sowie zum Entwurf von Modellen, die wenige Trainingsdaten benötigen, könnten helfen, die Komplexität der Annotation von Daten zu reduzieren.

Die Inferenz auf der *NVIDIA GTX 1080ti GPU* dauerte im Schnitt ca. 390 ms pro Bild und war deutlich höher als die obere Grenze der Inferenzzeit von 100 ms im Kontext des automatisierten Fahrens. Eine Verringerung der Auflösung des Inputbildes reduzierte die Inferenzzeit. Allerdings wurde die Segmentierung von klein ausgedehnten Regionen verschlechtert. Der benötigte *GPU*-Speicher für die Inferenz pro Bild lag im Schnitt bei vier GB, wobei die *GPU* ca. 11 GB zur Verfügung stellte. Das *FCN-8s*-Modell war also, was den Inferenzspeicher angeht, bezogen auf den verfügbaren *GPU*-Speicher wenig komplex. Die für die Inferenz verwendete *GPU* war aufgrund des Stromverbrauchs für eine Integration ins Fahrzeug nicht geeignet. Die Komplexität der Inferenz mit dem *FCN-8s*-Modell sollte deswegen in der Zukunft mit Hardware getestet werden, die im Fahrzeug integriert werden kann. Lee [162] stellte eine Version des *FCN-8s*-Modells vor, das mit dem Tool namens *TensorRT* [163] optimiert wurde. Diese optimierte Version des *FCN-8s*-Modells erreichte eine Inferenzzeit von 50 ms pro Bild auf dem *NVIDIA-DrivePX-AutoChauffeur*-Rechner [164]. Die Bilder stammen aus dem *DCS*-Validierungsdatensatz und hatten eine Auflösung von 1024 x 512 Pixeln. Die Inferenzzeit lag deutlich unterhalb der oberen Grenze von 100 ms. Darüber hinaus wurde der *NVIDIA-DrivePX-AutoChauffeur*-Rechner für die Inferenz von TNN im Fahrzeug entwickelt. Damit wäre diese optimierte Version des *FCN-8s*-Modells für die Anwendung im Kontext des automatisierten Fahrens geeignet. Allerdings zeigte der *mIOU*-Wert der Klassen des optimierten *FCN-8s*-Modells auf dem *DCS*-Validierungsdatensatz mit 48,1 % im Vergleich zu dem in dieser Arbeit trainierten *FCN-8s*-Modell eine relative Verschlechterung von ca. 26,2 %. Zukünftige Arbeiten zur Evaluation des optimierten *FCN-8s*-Modells unter Beachtung der Relevanz von Pixeln, wie in dieser Arbeit vorgeschlagen, wären hilfreich, um die Tauglichkeit des optimierten *FCN-8s*-Modells im Kontext des automatisierten Fahrens zu beurteilen.

5.7.2 Evaluation mit der *IOU*-Metrik unter Betrachtung aller Pixel

Die Evaluation des *FCN-8s*-Modells auf dem *DCS*-Validierungsdatensatz anhand der *IOU*-Metrik zeigte, dass dieses Modell mit dem *mIOU*-Wert der Klassen von 60,7 % den optimalen *IOU*-Wert von 100 % nicht erfüllte. Eine detaillierte Betrachtung der *IOU*-Werte für einzelne Klassen zeigte jedoch eine hohe Interklassen-Fluktuation der *IOU*-Werte:

1. Je öfter Klassen in den Trainingsdaten vorkamen und je breiter diese Klassen in den Bildern waren, desto besser waren diese Klassen segmentiert. Die Klassen *road, sky, building, vegetation* und *car* z. B. wiesen *IOU* von bis zu 92 % auf, die schon näher an dem optimalen *IOU*-Wert lagen.
2. Je seltener eine Klasse in den Trainingsdaten vorkam, desto schlechter wurde diese segmentiert. Auch Klassen, die größere Ausdehnung in den Bildern hatten, aber seltener vorkamen wie etwa *bus, truck* und *train,* wurden schlechter segmentiert.
3. Je kleiner die Ausdehnung einer Klasse in den Trainingsdaten war, desto schlechter war die Klasse segmentiert (siehe bspw. *pole* und *person*).
4. Je ähnlicher Klassen waren, desto größer war die Verwechslung zwischen diesen Klassen (siehe bspw. *person* und *rider*, *building* und *wall*, *road* und *sidewalk*).

Die Zusammenfassung von Klassen in Kategorien führte mit einem *mIOU*-Wert von 78 % zu einer relativen Verbesserung des Kategorien-*mIOU*-Werts um 28,5 % im Vergleich zu dem Klassen-*mIOU*-Wert. Diese Verbesserung ließ sich durch die Verringerung der Verwechselung zwischen ähnlichen Klassen erklären, da ähnliche Klassen oft derselben Kategorie angehörten. Hier zeigte die Betrachtung von einzelnen Kategorien auch unterschiedliche Kategorien-*IOU*-Werte. Die Unterschiede ließen sich mit den o. g. Eigenschaften des *FCN-8s*-Modells für die Klassen-*IOU*-Werte erklären. So hatte bspw. die Kategorie *flat* durch die Zusammenfassung der Klassen *road* und *sidewalk* den besten *IOU*-Wert von 91,80 %. Dieser Werte näherten sich dem optimalen *IOU*-Wert an.

5.7.3 Evaluation mit der *IOU*-Metrik unter Betrachtung von erreichbaren Pixeln

Da im Kontext des automatisierten Fahrens Pixel aufgrund ihrer Entfernung zum Ego-Fahrzeug unterschiedlich relevant sind, wurde anhand der dynamischen Parameter $t = [0\,\mathrm{s}, 3\,\mathrm{s}]$, $\psi = 90°$, $\mathrm{v} = \{0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, \infty\}\,\mathrm{km/h}$ und $\mathrm{a} = 4\,\mathrm{m/s^2}$ die Erreichbarkeitsmenge des Ego-Fahrzeugs bestimmt. Nur Pixel, die erreichbar waren, wurden dann in der Evaluation betrachtet. Dies führte zu einer relativen minimalen Verbesserung der Klassen-*mIOU* um 3,5 % im Vergleich zu dem Fall, in dem alle Pixel betrachtet wurden. Dabei wurden folgen Eigenschaften festgestellt:

1. Je näher klein ausgedehnte Regionen dem Ego-Fahrzeug waren, desto besser waren diese Regionen segmentiert. So erreichte bspw. die Klasse *person*, die vulnerable Verkehrsteilnehmer enthält, eine relative Verbesserung des *IOU*-Wertes von 20,3 % mit der Geschwindigkeit des Ego-Fahrzeugs $\mathrm{v} = 10\,\mathrm{km/h}$ im Vergleich zu dem Fall, bei dem alle Pixel betrachtet wurden.
2. Je näher breit ausgedehnte Regionen an dem Ego-Fahrzeug lagen, desto schlechter wurden diese Regionen segmentiert: Regionen, die öfter in den Trainingsdaten vorkamen und/oder breit gedehnt und meistens im Hintergrund waren, hatten im Vergleich zu der geringeren Anzahl an erreichbaren Pixeln viele falsche Segmentierungen. Diese führten zu einer Verschlechterung der *IOU*-Werte dieser Klassen (siehe bspw. *building, vegetation, sky* und *unknow*). Die Klassen *road* und *sidewalk* wurden im Gegenteil leicht verbessert, da diese Regionen im Vordergrund lagen und somit viele erreichbare Pixel hatten.
3. Je näher texturähnliche Regionen am Ego-Fahrzeug waren, desto besser wurden diese segmentiert, da im Nahbereich in den Bildern texturreiche Merkmale für die Segmentierung verfügbar waren (siehe bspw. *road* und *sidewalk, rider* und *person*).

Aufgrund der schlechten Segmentierung einiger Klassen, vor allem *unknow*, *sky* und *vegetation*, war der Kategorien-*mIOU*-Wert für erreichbare Pixel um bis zu 10,6 % schlechter im Vergleich zu dem Fall, wo alle Pixel betrachtet wurden. Dies erklärte sich durch die Verwechslung zwischen Klassen unterschiedlicher Kategorien. Der Verlauf der *IOU*-Werte einzelner Kategorien korrelierte mit dem Verlauf der *IOU*-Werte zu der den Kategorien entsprechenden Klassen. Die Kategorien *flat, object, human* und *vehicle* wurden leicht verbessert, da die entsprechenden Klassen verbesserte *IOU*-Werte hatten.

Während sowohl der Klassen-*mIOU*-Wert als auch der Kategorien-*mIOU*-Wert weit von dem optimalen Wert entfernt waren, näherten sich vor allem die Kategorien *flat* und *vehicle* mit *IOU*-Werten von etwas mehr als 93 % dem optimalen *IOU*-Wert.

5.7.4 Evaluation mit der *IOU*-Metrik unter Betrachtung von erreichbaren und relevanten Pixeln der Klassen *traffic light* und *traffic sign*

In dieser Arbeit wurden Pixel der Klassen *traffic light* und *traffic sign*, die für die Fahraufgabe relevant waren, anhand der *IOU*-Metrik evaluiert. Ein *Traffic-light*- oder *Traffic-sign*-Objekt war relevant, wenn dieses Objekt unmittelbar oder in der nächsten Zukunft die zulässigen Manöver des Ego-Fahrzeugs einschränken konnte. Dazu wurde noch unterschieden, ob das Objekt zu der Erreichbarkeitsmenge des Ego-Fahrzeugs gehörte. So zeigten Pixel der Klassen *traffic light* mit *IOU*-Werten zwischen 96,3 % und 98,5 % sowie *traffic sign* mit *IOU*-Werten zwischen 91,8 % und 93,6 % in dem Fall, bei dem nur relevante und erreichbare Pixel betrachtet wurden, eine relative Verbesserung von bis

zu jeweils 101 % und 59,7 % im Vergleich zu dem Fall, wo weder die Relevanz noch die Erreichbarkeit der Pixel betrachtet wurden. Diese *IOU*-Werte lagen deutlich näher an dem optimalen Wert von 100 %.

5.7.5 Evaluation der semantischen Segmentierung als Grundlage für die Erkennung von Freiraum und Hindernissen

Diese Evaluation fokussierte sich auf die *OGM*-Karte, die anhand der semantischen Segmentierung geschätzt wurde. Die *OGM*-Karte erfasste den Freiraum und die Hindernisse vor dem Ego-Fahrzeug und wurde oft als Grundlage für die Trajektorienplanung verwendet. Für den Aufbau der *OGM*-Karte wurden die Parameter des Ego-Fahrzeugs $t = [0\,\mathrm{s}, 3\,\mathrm{s}], \psi = 90°$, $\mathrm{v} = \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}\,\mathrm{km/h}$ und $\mathrm{a} = 4\,\mathrm{m/s^2}$ so ausgewählt, dass diese bestmöglich dynamische Fahrsituationen im urbanen Raum abdeckten.

Für die Evaluation wurde der *MSE*-Wert der *OGM*-Karten, die mit der semantischen Segmentierung des *FCN-8s*-Modells und der *Ground Truth* der semantischen Segmentierung geschätzt wurden, verwendet. Die *MSE*-Werte der *OGM*-Karten lagen im Schnitt zwischen 2,7 % und 5,7 % und waren somit sehr nah am optimalen 0-%-*MSE*-Wert.

Eine weitere Metrik, die zur Evaluation der *OGM*-Karten verwendet wurde, ist die *THW*-Metrik, wobei der *THW*-Wert die Zeit war, die das Ego-Fahrzeug benötigte, um das erste Hindernis in seiner Fahrtrichtung zu erreichen. Die mittleren relativen *THW*-Differenzen zwischen den *THW*-Werten der *OGM*-Karten, die mit der *Ground Truth* der semantischen Segmentierung und mit der semantischen Segmentierung des *FCN-8s*-Modells generiert wurden, über alle Daten des *DCS*-Validierungsdatensatzes lagen zwischen 1 % und 2,8 %. Analog lagen die mittleren absoluten *THW*-Differenzen zwischen 16 ms und 48 ms. Sowohl die mittleren relativen *THW*-Differenzen als auch die mittleren absoluten *THW*-Differenzen lagen somit sehr nah an den optimalen Werten von jeweils 0 % und 0 ms.

Die *MSE*- und die *THW*-Metriken zeigten also, dass die *OGM*-Karten, die mit der semantischen Segmentierung des *FCN-8s*-Modells generiert wurden, gut genug waren für die Planung der Trajektorien von automatisierten Fahrzeugen.

Bei der Berechnung der *THW*-Werte wurde die Dynamik der Hindernisse nicht betrachtet, da es keine Möglichkeit gab, diese Informationen zu extrahieren. Darüber hinaus wurde angenommen, dass das Ego-Fahrzeug geradeaus fahren würde, da verfügbare Informationen zu dem Ego-Fahrzeug nur für einzelne *Frames* vorhanden waren. Diese fehlenden Informationen könnten einen Einfluss auf die *THW*-Werte haben. Es wäre sinnvoll, die *THW*-Werte erneut mit Datensätzen zu berechnen, in denen die Trajektorie des Ego-Fahrzeugs sowie die Trajektorien aller dynamischen Hindernisse vorhanden sind. Darüber hinaus sollten die Daten als Sequenzen verfügbar sein.

5.7.6 Ideen zur Verbesserung der semantischen Segmentierung

Zum Lösen der Probleme des *FCN-8s*-Modells wurde im Abschnitt 3.5 folgende Lösungen aus der Literatur zur Verbessrung der Architektur des Modells vorgeschlagen:

1. Die gedehnte Faltung [84]: Hier wurden die rezeptiven Felder der Neuronen exponentiell erhöht, während die Anzahl der Parameter der Faltungskerne nur linear anstieg. So konnten die Neuronen groß ausgedehnte Regionen besser erfassen.
2. *Pyramid-Pooling*-Modul [83]: Parallele *Pooling*-Schichten mit unterschiedlichen Skalierungsfaktoren am Ausgang des *Encoders* waren hilfreich, um klein ausgedehnte Objekte besser zu erfassen.
3. Verwendung eines besseren *Encoders*: Die Ersetzung des *VGG-16*-Modells durch das *ResNet-101*-Modell führte zu einer Verbesserung der semantischen Segmentierung.
4. Die Verwendung eines besseren *Decoders* wie bspw. *DUC* [85] oder von mehreren Entfaltungsschichten [84] verbesserten ebenfalls die Segmentierung.

Die Abbildung 5-26 stellt den Verlauf der Klassen-*IOU*-Werte *des FCN-8s*-Modells (rote Kurve) und des *DeepLabv3+*-Modells (blaue Kurve) dar. Beide Modelle wurden auf der *DCS-Benchmark*-Seite [86] publiziert und mit dem *DCS*-Testdatensatz evaluiert. Das *DeepLabv3+*-Modell [84] integrierte die o. g. Lösungen und führte mit dem Klassen-*mIOU*-Wert 82,14 % und dem Kategorien-*mIOU*-Wert 92 % zu den besten *mIOU*-Werten auf dem *DCS*-Testdatensatz. Im Vergleich zu dem *FCN-8s*-Modell, das auf der *DCS-Benchmark*-Seite publiziert wurde, brachte das *DeepLabv3+*-Modell eine relative Verbesserung des Klassen-*mIOU*-Wertes von 25,7 % und des Kategorien-*mIOU*-Wertes von 7,38 %. Der Klassen-*mIOU*-Wert des *DeepLabv3+*-Modells blieb auch analog zu dem Klassen-*mIOU*-Wert des *FCN-8s*-Modells entfernt von dem optimalen *IOU*-Wert, während der Kategorien-*mIOU*-Wert sich dem optimalen *IOU*-Wert annäherte. Die Abbildung 5-26 zeigt die deutliche Verbesserung der Klassen-*IOU*-Werte durch das *DeepLabv3+*-Modell bei Klassen, die das *FCN-8s*-Modell Problem hatte (siehe bspw. *wall, fence, rider, truck* und *bus*). Die Klasse *terrain* blieb die einzige Klasse, die bei den beiden Modellen einen *IOU*-Wert unter 75 % hatte und durch das *DeepLabv3+*-Modell kaum verbessert wurde. In der Konfusionsmatrix in der Abbildung 5-6 war die Klasse *terrain* oft falsch als *vegetation* oder *sidewalk* segmentiert. In den Trainingsdaten ähnelte sich die Klasse *terrain* mit den Klassen *sidewalk* und *vegetation* sehr. Wegen dieser Ähnlichkeit schaffte es sehr wahrscheinlich das *DeepLabv3+*-Modell nicht, den *IOU*-Wert der Klasse *terrain* im Vergleich zu dem *IOU*-Wert des *FCN-8s*-Modells zu verbessern.

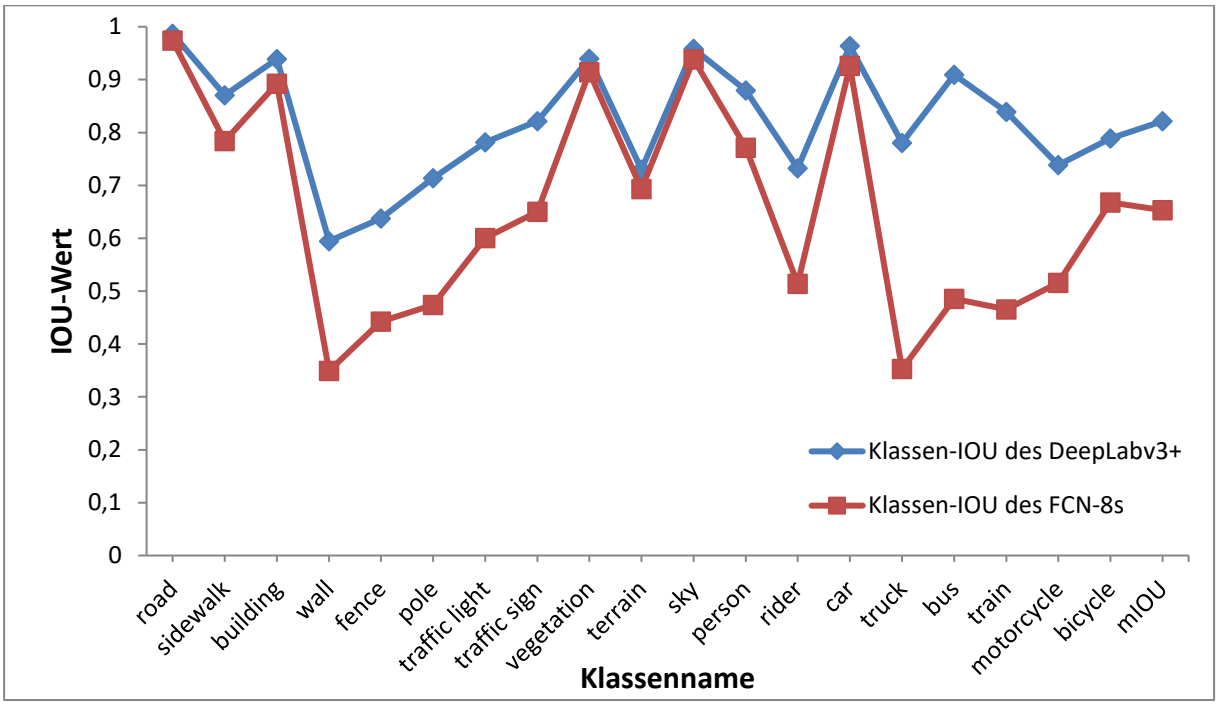


Abbildung 5-26: Verlauf der Klassen-IOU-Werte des FCN-8s-Modells (rote Kurve) und des DeepLabv3+-Modells (blaue Kurve). Beide Modelle wurden auf der DCS-Benchmark-Seite [86] publiziert und mit dem DCS-Testdatensatz evaluiert.

In der Abbildung 5-27 zeigte der Verlauf der Kategorien-*IOU*-Werte des *FCN-8s*-Modells (rote Kurve) und des *DeepLabv3+*-Modells (blaue Kurve), dass die Verbesserung der Kategorien durch das *DeepLabv3+*-Modell im Vergleich zu der Verbesserung der Klassen deutlich geringer war. So lag die relative Verbesserung des Kategorien-*mIOU*-Wertes des *DeepLabv3+*-Modells bei 7,38 % im Vergleich zum Kategorien-*mIOU*-Wert des *FCN-8s*-Modells. Beide Modelle wurden auf der *DCS-Benchmark*-Seite [86] publiziert und mit dem *DCS*-Testdatensatz evaluiert. Die größte relative Verbesserung des Kategorien-*IOU*-Werts durch das *DeepLabv3+*-Modell von 35,26 % lag bei der Kategorie *object*. Trotz dieser Verbesserung war der *IOU*-Wert der Kategorie *object* mit 77,12 % deutlich weit von dem optimalen Wert entfernt. Die Erklärung dafür war, dass die Kategorie *object* Klassen mit klein ausgedehnten Regionen enthielt, die trotz der Anwendung der o. g. Verbesserungsmethoden aufgrund der Herunterskalierung im *Encoder* falsch segmentiert wurden.

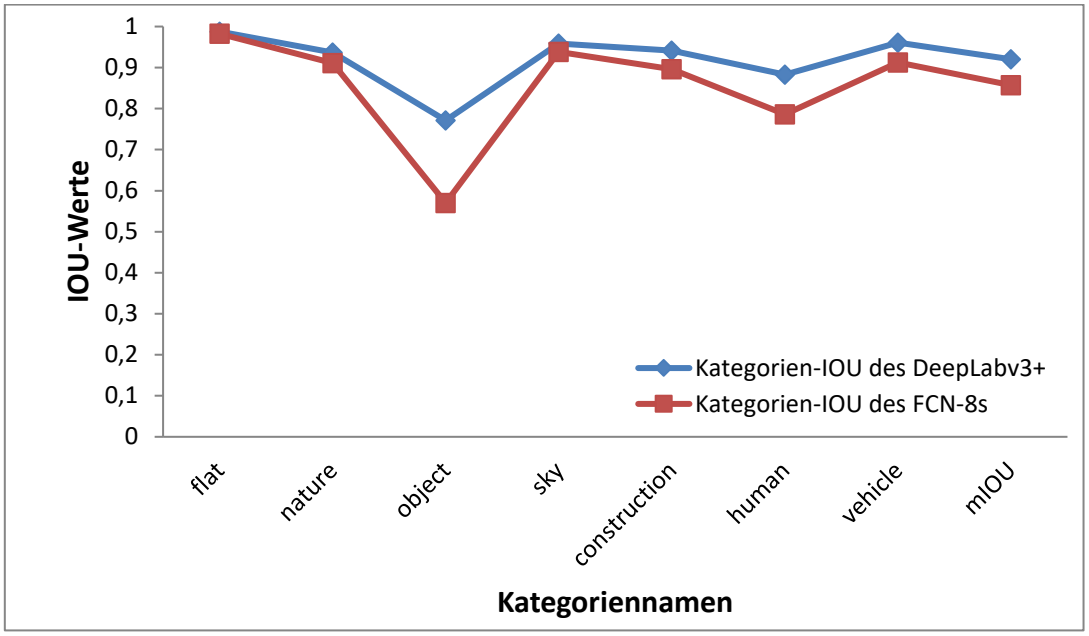


Abbildung 5-27: Verlauf der Kategorien-IOU-Werte des FCN-8s-Modells (rote Kurve) und des DeepLabv3+-Modells (blaue Kurve). Beide Modelle wurden auf der DCS-Benchmark-Seite [86] publiziert und mit dem DCS-Testdatensatz evaluiert.

Zusammengefasst zeigte das *DeepLabv3+*-Modell durch die Anwendung der o. g. Methoden eine deutliche Verbesserung der Klassen- und Kategorien-*IOU*-Werte. Allerdings lag mehr als die Hälfte der Klassen- und Kategorien-*IOU*-Werte unter 95 % und war somit nicht nah genug an dem optimalen Wert von 100 %. Der Mangel an Kontextinformationen kombiniert mit der eingeschränkten Auflösung der Kamera erklärte die Fehler der Segmentierung durch das *DeepLabv3+*-Modell. Obwohl die Inferenzzeit und der Speicherbedarf des *DeepLabv3+*-Modells auf der *DCS-Benchmark*-Seite nicht vorhanden waren, wurde anhand des *DeepLabv3+-TensorFlow*-Modells namens *xception65_cityscapes_trainfine* (siehe [165]) festgestellt, dass dieses *DeepLabv3+*-Modell im Schnitt ca. 1 s und 4,4 GB für die Inferenz eines Bildes aus dem DCS-Validierungsdatensatz mit der Auflösung 1505×752 auf der *GTX 1080ti GPU* brauchte. Damit war das *DeepLabv3+*-Modell bis zu 2,5-mal langsamer als das *FCN-8s*-Modell. Ebenfalls erhöhte sich der Speicherverbrauch des *DeepLabv3+*-Modells um 10 % im Vergleich zum *FCN-8s*-Modell. Die Verbesserung der *IOU*-Werte des *DeepLabv3+*-Modells im Vergleich zum *FCN-8s*-Modell führte also zur Erhöhung der Komplexität des Modells zur semantischen Segmentierung.

Einfachere Architekturen, die optimale Ergebnisse in der Erreichbarkeitsmenge des Ego-Fahrzeugs liefern und gleichzeitig weniger Rechenzeit und Speicher benötigen, sollen erforscht werden. Solche Architekturen können bspw. automatisch gelernt und optimiert werden (siehe bspw. [166–171]). Um die Erfassung außerhalb der Erreichbarkeitsmenge des Ego-Fahrzeugs zu ermöglichen, können Information von anderen Verkehrsteilnehmern über die *Car-2-X*-Kommunikation vom Ego-Fahrzeug verwendet werden. Damit können Regionen erfasst werden, die von den Sensoren des Ego-Fahrzeugs bspw. aufgrund der Entfernung und von Verdeckungen schwer zu erfassen sind. Darüber hinaus sollen die semantischen Klassen bezogen auf die Relevanz für den Anwendungsfall so ausgewählt werden, dass diese sich anhand von visuellen Merkmalen unterscheiden lassen.

Die Integration von Kontextabhängigkeiten kann helfen, die semantische Segmentierung zu verbessern. Neverova et al. [172] verwendeten autoregressive TNN zur zeitlichen Prädiktion der semantischen Segmentierung und zeigten gute Ergebnisse auf dem *DCS*-Datensatz. Allerdings wurde in [172] lediglich die Prädiktion der semantischen Segmentierung adressiert. Die Kombination der Prädiktion der Segmentierung mit der aktuellen Segmentierung würde sehr wahrscheinlich helfen, die Segmentierung zu verbessern. Neben der Prädiktion der semantischen Segmentierung, sollten auch Ansätze zu Prädiktion der Szene als Bilder untersucht werden (siehe bspw. [173–175]). *CRF* wurden oft ge-

nutzt, um räumliche Abhängigkeit zu modellieren. Damit würde die Segmentierung verbessert (siehe bspw. [89, 93, 95, 96]). Allerdings bleibt die Integration von *CRF* und TNN in ein einheitliches Modell ein offenes Problem und sollte adressiert werden. Das gemeinsame Lernen von mehreren Aufgaben führte zur Verbesserung der einzelnen Aufgaben (siehe bspw. [91, 101]). Allerdings bleiben die Integration und die Minimierung der Fehlerfunktion all dieser Aufgaben ein offenes Problem. Dieses Problem sollte erforscht werden.

Die Minimierung der Fehlerfunktion in der Gleichung (21) hat den Nachteil, dass Klassen, die öfter in den Trainingsdaten vorkommen, präferiert werden, während seltene Klassen oft falsch segmentiert werden. In dem *DCS*-Trainingsdatensatz kommen die Klassen wie z. B. *truck*, *motorcycle* und *bus* selten vor. Diese seltenen Klassen haben sowohl für das *FCN-8s*-Modell als auch für das *Deeplabv3+*-Modell oft Klassen-*IOU*-Werte unterhalb des mittleren Klassen-*mIOU*-Werts. Um den Einfluss der Klassenanteile in den Trainingsdaten der Fehlerfunktion auszugleichen, wurden von verschiedenen Autoren [176–180] mögliche Lösungen vorgeschlagen. Solche Ansätze sollten in der Zukunft weiter untersucht werden.

Die Max-*Pooling*-Schichten, die verwendet werden, um die Dimension der Daten in *CNN* zu reduzieren, haben den Nachteil, dass feine Strukturen im *Encoder* verloren gehen (siehe bspw. *pole*). Diese Strukturen sind aber in einigen Fällen für die Fahraufgabe relevant. Verfahren zu Datenkompression, die gleichzeitig feine relevante Strukturen in Bildern beinhalten, sollten adressiert werden (siehe bspw. [181]).

Zur Integration von Unsicherheiten in der semantischen Segmentierung schlugen Kendall et al. [88] das *Bayesian-SegNet*-Modell vor, da die *Softmax*-Funktion am Ende des *Decoders* keine Unsicherheiten lieferte. Dabei wurde zur Inferenzzeit das *Dropout*-Verfahren mehrmals für dasselbe Inputbild angewendet. Der Output des Netzes wurde dann als Samples einer Normalverteilung betrachtet. Die Ergebnisse zeigten, dass das Netz Unsicherheiten an den Rändern von Objekten oder bei Objekten, die sich ähnlich waren, richtig erfassen konnte. Allerdings war das Verfahren speicher- und rechenaufwendig, da das gleiche Bild mehrfach inferiert wurde. Einfachere Verfahren sollten in der Zukunft erforscht werden.

Folgende Probleme, die allgemein bei TNN auftauchen, sollten auch in der Zukunft untersucht werden:

1. Annotation von Trainingsdaten: Die Annotation von Trainingsdaten erfolgt oft manuell und ist meistens sehr aufwendig. Ansätze zur semi-automatischen Generierung von annotierten Trainingsdaten [182], zum semi-überwachten Lernen und zum Entwurf von Modellen, die wenige Trainingsdaten benötigen [183], sollten erforscht werden.
2. Das katastrophale Vergessen ist die Tatsache, dass ein TNN-Modell die Informationen vergisst, die zur Lösung einer Aufgabe gelernt wurde, wenn eine neue Aufgabe gelernt wird. Verfahren zur Minderung des katastrophalen Vergessens sollten gesucht werden (siehe bspw. [184]).
3. Umgang mit *Adversarial*-Beispielen: *Adversarial*-Beispiele sind Inputdaten, von denen die Statistik verändert wurde. Diese Beispiele führen bei der Inferenz von TNN zur falschen Ergebnissen, weil TNN oft die Statistik und nicht die Semantik der Daten lernen [185]. Meistens liefern Menschen für solche Beispiele die richtigen Inferenzergebnisse. Ilyas et al [186] zeigten die Existenz von nicht robusten Merkmalen in Trainingsdaten, die zu einer guten Generalisierung beim Training von TNN führten und gleichzeitig zur Generierung von *Adversarial*-Beispielen dienten. *Adversarial*-Beispiele stellen ein Risiko für TNN-Modelle dar, die in sicherheitskritischen Fahrsystemen eingesetzt werden (siehe bspw. [107, 187, 188]). Verfahren, um mit solchen *Adversarial*-Beispielen umzugehen, sollten untersucht werden [189–191].
4. Testen und Absichern von TNN: TNN sind aufgrund ihrer Komplexität schwer zu testen und abzusichern. Dies stellt ein Hindernis für die Integration von TNN in sicherheitskritische Systeme dar. Verfahren zum Testen und absichern von TNN-Modellen sollten deshalb erforscht

werden siehe bspw. [144, 192, 193]). Weiterhin sollten Verfahren zur Verbesserung der Erklärbarkeit von TNN untersucht werden (siehe bspw. [143, 194–197]).

5.8 Zusammenfassung

Dieses Kapitels befasste sich mit der semantischen Segmentierung von Kamerabildern, das heißt die Klassifikation von Pixeln in Kamerabildern. Der in Kapitel 4 vorgeschlagene Ansatz wurde für die semantische Segmentierung angewendet. Dieser Ansatz beinhaltete ein Modul, das auf TNN basierte. Dieses Modul wurde für die semantische Segmentierung verwendet, da TNN-Modelle in der Literatur die besten Ergebnisse für die semantische Segmentierung lieferten. Das TNN-Modell namens *FCN-8s* wurde als Modell für die Segmentierung in dieser Arbeit gewählt, da dieses Modell ein Meilenstein für die semantische Segmentierung mit TNN darstellte. Der Fokus lag hier in der Bewertung der semantischen Segmentierung unter Beachtung von Kontextinformationen.

5.8.1 Struktur des *FCN-8s*-Modells

Das *FCN-8s*-Modell war die Kombination aus einem *Encoder* und einem *Decoder*. Der *Encoder* generierte robuste Merkmale anhand des angepassten *VGG-16*-Modells. Die Dimension der *Score*-Karte aus der letzten Schicht des *Encoders* war im Vergleich zum Inputbild um $1/16$ herunterskaliert. Der *Decoder* verwendete Entfaltungsschichten, um die herunterskalierte *Score*-Karte des *Encoders* auf die Dimension des Inputbildes hoch zu skalieren. Nachdem die *Score*-Karte hochskaliert wurde, wurde ein *Softmax*-Klassifikator verwendet, um die Wahrscheinlichkeiten der semantischen Klassen einzelner Pixel zu berechnen. Um detailreiche Informationen in die Segmentierung zu integrieren, wurden neben der *Score*-Karte aus der letzten Schicht des *Encoders* auch weitere *Score*-Karten aus den letzten und vorletzten *Pooling*-Schichten des *Encoders* verwendet. Durch die Auswahl der Schichten des *Encoders* und des *Decoders* hatte das *FCN-8s*-Modell den Vorteil, dass der *Encoder* und der *Decoder* gemeinsam trainiert wurden. Die Tiefe des *FCN-8s*-Modells war im Vergleich zu aktuellen TNN-Modellen gering.

5.8.2 Training und Komplexität des *FCN-8s*-Modells

Zum Trainieren des *FCN-8s*-Modells wurde der *DCS*-Datensatz verwendet. Dieser Datensatz bestand aus ca. 5000 Bildern von urbanen Szenen aus Deutschland und Zürich. Neben den Bildern wurden die Annotationen der semantischen Segmentierung bereitgestellt. Der Aufwand der Generierung dieser Daten war mit etwas mehr als 300 Tagen hoch. Das Training auf der *NVIDIA GTX 1080ti GPU* wurde nach 300 000 Iterationen abgebrochen und dauerte ca. 47 Stunden. Das Modell nach ca. 70 000 Iterationen wurde für die Inferenz verwendet, da dieses Modell bezogen auf die Kreuzvalidierung optimal war. Dieses optimale Modell wurde nach ca. elf Trainingsstunden generiert. Die Trainingszeit war für TNN-Modelle üblich. Das trainierte TNN-Modell hatte ca. 134,5 Millionen Parameter und benötigte ca. 0,52 GB Speicherplatz auf der Festplatte. Das *FCN-8s*-Modell wurde in der Offlinephase mit dem *Caffe*-Tool trainiert.

5.8.3 Inferenz und Evaluation des trainierten *FCN-8s*-Modells

Das trainierte *FCN-8s*-Modell wurde auf dem *DCS*-Validierungsdatensatz inferiert. Die mittlere Inferenzzeit und der Speicherbedarf bei Bildern der Auflösung 1505×752 auf der *NVIDIA GTX 1080ti GPU* betrugen jeweils ca. 390 ms und ca. vier GB. Damit war das *FCN-8s*-Modell nicht echtzeitfähig. Das *TensorRT*-Tool von *Nvidia* wurde als mögliche Lösung vorgeschlagen, um die Inferenzzeit zu reduzieren.

Die Evaluation des trainierten *FCN-8s*-Modells auf dem *DCS*-Validierungsdatensatz unter Betrachtung aller Pixel ergab einen Klassen-*mIOU*-Wert von 60,7 % und einen Kategorien-*mIOU*-Wert von 78 %. Beide *mIOU*-Werte waren entfernt von dem optimalen *IOU*-Wert von 100 %. Dazu variierten die einzelnen Klassen- und Kategorien-*IOU*-Werte zwischen 36 % und 92 %. So näherten sich einige Klassen und Kategorien dem optimalen *IOU*-Wert bereits besser an. Die fehlerbehaftete Segmentierung war vor allem bei Klassen mit kleiner Ausdehnung, bei Klassen mit ähnlichem Aussehen sowie Klassen, die wenig in den Trainingsdaten vorkamen, festzustellen.

Die Einschränkung der Evaluation durch Kontextinformationen auf erreichbaren Pixeln führte zu einer kleinen relativen Verbesserung des Klassen-*mIOU*-Werts um bis zu 3,5 % im Vergleich zu dem Fall, in dem alle Pixel bei der Evaluation betrachtet wurden. Jedoch verschlechterte sich der Kategorien-*mIOU*-Wert um 10,6 %. Während Klassen mit kleinen Ausdehnungen wie erwartet eine relative Verbesserung des *IOU*-Wertes von bis zu 20,3 % zeigten, verschlechterten sich die *IOU*-Werte von Klassen, die mehr im Hintergrund waren und somit wenig erreichbare Pixel hatten, um bis zu 200 %. Damit war das *FCN-8s*-Modell nicht immer besser, wenn die Regionen der Kamera näher waren.

Die Klassen *traffic light* und *traffic sign*, die in der Verkehrssteuerung eine wesentliche Rolle spielen, wurden besonders betrachtet, wenn Pixel dieser Klassen einen Einfluss auf das Manöver des Ego-Fahrzeugs hatten. Hier wurden *IOU*-Werte von bis zu 98,5 % erreicht. Damit wurde bestätigt, dass das *FCN-8s*-Modell im Mittel zwar nur ausreichende Klassen-*IOU*-Werte lieferte, aber für bestimmte Klassen mit einer wesentlichen Semantik im Verkehr sehr nah an dem optimalen *IOU*-Wert war. Die holistische Betrachtung von Szenen und Situationen durch die Verwendung von Kontextinformationen und der Relevanz von bestimmten Klassen bei dem Entwurf, der Implementierung und der Evaluation von TNN-Modellen zur Anwendung im autonomen Fahren sollten in der Zukunft adressiert werden.

Eine Einschränkung der *IOU*-Metrik war, dass lediglich einzelne Pixel bewertet wurden. Allerdings sind Fehler bei einzelnen Pixeln von bestimmten Klassen im Kontext des autonomen Fahrens nicht kritisch. Es wurde deshalb die semantische Segmentierung des *FCN-8s*-Modells als Grundlage für die Berechnung von *OGM*-Karten verwendet. Die Evaluation der *OGM*-Karten anhand der *MSE*- und der *THW*-Metriken zeigte, dass diese *OGM*-Karten als Basis für die Planung der Trajektorien von autonomen Fahrzeugen gut genug waren.

Im letzten Teil der Evaluation wurde das *FCN-8s*-Modell mit einem der aktuell besten Modelle zur semantischen Segmentierung namens *Deeplabv3+*-Modell verglichen. Trotzt der deutlichen Verbesserung des *Deeplabv3+*-Modells im Vergleich zu dem *FCN-8s*-Modell war der Klassen-*mIOU*-Wert des *Deeplabv3+*-Modells mit 82,14 % noch entfernt von dem optimalen *IOU*-Wert. Dafür war das *Deeplabv3+*-Modell mit einer Inferenzzeit von 1 s deutlich langsamer als das *FCN-8s*-Modell. Zukünftige Arbeiten sollten den Entwurf von einfachen Modellen, die Kontextinformation und Unsicherheiten integrieren, behandeln. Gleichzeitig sollten die Kontextinformationen und die Anwendungsfälle bei der Evaluation dieser Modelle eine wesentliche Rolle spielen.

Das in diesem Kapitel vorgestellt *FCN-8s*-Modell zur semantischen Segmentierung basierte auf TNN. TNN-Modelle haben den Nachteil, dass die Inferenz wegen ihrer Komplexität schwer nachvollziehbar ist. So ist es schwierig, die Segmentierungsergebnisse eines gegebenen Inputbildes zu erklären und den Einfluss von Kontextinformationen auf diese Ergebnisse zu prädizieren. Im nächsten Kapitel wird untersucht, inwiefern Kontextinformationen vom *FCN-8s*-Modell bei der Segmentierung einbezogen werden und ob die Verletzung der Kontextabhängigkeiten mit falschen Segmentierungen zusammenhängt.

6 Erkennung von Inkonsistenzen der semantischen Segmentierung

In Kapitel 5 wurde die semantische Segmentierung mithilfe von TNN vorgestellt. Dabei wurde das *FCN-8s*-Modell mit dem *DCS*-Datensatz trainiert und evaluiert. Das beste Modell auf dem *DCS*-Testdatensatz namens *DeepLabv3+*-Modell nutzte die Idee des *FCN-8s*-Modells und erweiterte diese Idee mit weiteren Kontextinformationen. Zur Integration von Kontextinformationen in der semantischen Segmentierung mit TNN wurden folgende Methoden vorgeschlagen:

1. Die Erweiterung der rezeptiven Felder der Neuronen
2. Die Verwendung von unterschiedlichen Skalierungsfaktoren im *Encoder*
3. Die Verwendung von *CRF* als *Post-Processing* von TNN oder den Nachbau von *CRF* in TNN
4. Die Verwendung von tieferen *Encodern* und *Decodern*

Viele dieser Methoden wurden in dem *DeepLabv3+*-Modell integriert und führten zur Verbesserung der semantischen Segmentierung. Trotz der Verbesserung der semantischen Segmentierung blieb aufgrund der Komplexität von TNN schwer nachzuvollziehen, wie die Kontextinformationen von TNN für die Segmentierung betrachtet wurden. Deshalb war es nicht möglich, eine Aussage über die Qualität der Segmentierung mit TNN zu treffen, wenn der Input diese TNN-Szenen, die in den Trainingsdaten selten oder nie vorkamen, beinhaltete. Lediglich die Integration von *CRF* als *Post-Processing* lieferte die Möglichkeit, die in *CRF* integrierten Kontextinformationen explizit zu modellieren und zu interpretieren. Trotz der Integration von *CRF*-Modellen sowie von weiteren Kontextinformationen in die semantische Segmentierung war der Fokus mehr auf die Verbesserung der semantischen Segmentierung gesetzt. Die Erkennung von Inkonsistenzen der semantischen Segmentierung spielt aber eine wesentliche Rolle bei komplexeren Modellen wie TNN, da die Merkmale, die von diesen Modellen zur Segmentierung verwendet werden, komplex und schwer nachzuvollziehen sind. Lediglich wenige Autoren adressierten die Konsistenzschätzung in der Literatur.

In diesem Kapitel wird die Integration von Kontextinformationen, die explizit und symbolisch modelliert wurden, in die semantische Segmentierung zur Erkennung von Inkonsistenzen betrachtet. Das Ziel dieses Kapitels ist zu prüfen, inwiefern Kontextinformationen bei der Inferenz von TNN-Modellen betrachtet werden. Dies bedeutet, dass dieses Kapitel untersuchen soll, ob es eine Korrelation zwischen falsch segmentierten und inkonsistenten Regionen gibt. Dafür werden die wissensbasierten, PGM-basierten und TNN-basierten Module des in Kapitel 4 vorgestellten Konzepts verwendet. Der erste Abschnitt dieses Kapitels stellt die Aufgabe der Konsistenzschätzung vor. Im zweiten Abschnitt wird der Lösungsansatz basierend auf dem Konzept des Kapitels 4 illustriert. Die Abschnitte drei und vier werden die Module des Lösungsansatzes im Detail erläutern. In diesen Abschnitten werden die Modellierung, die Implementierung und die Evaluation der Module des Lösungsansatzes veranschaulicht. Sowohl quantitative als auch qualitative Evaluationen werden angesprochen. Im fünften Abschnitt werden die Ergebnisse des vorgeschlagenen Lösungsansatzes diskutiert. Der Abschnitt sechs fasst dieses Kapitel zusammen.

6.1 Vorstellung der Aufgabe

Gegeben sei ein Bild $I \in \mathbb{R}^{w \times h \times c}$, die Anzahl der semantischen Klassen $n \in \mathbb{N}$, das semantisch segmentierte Bild $I_s \in \mathbb{N}^{w \times h}$ und das Vorwissen KB über die semantischen Klassen. Das segmentierte Bild I_s kann als eine endliche Menge von Regionen $R^{I_s} = \{R_1^{I_s}, \dots, R_m^{I_s}\}, m \in \mathbb{N}$ modelliert werden, wobei jede Region $R_k^{I_s}$ eine Menge von zusammenhängenden Pixeln ist, die dieselbe semantische Klasse *c* haben.

$$R_k^{I_s} = \{\{(x_1, y_1), \dots, (x_l, y_l)\}, c\} \tag{29}$$

Gesucht ist die Wahrscheinlichkeit der Konsistenz aller Regionen von I_s, $K(R^{I_s}) = \{K(R_1^{I_s}), \dots, K(R_m^{I_s})\}$, gegeben die Regionen des semantisch segmentierten Bildes und das Vorwissen.

$$P_\theta\left(\left(K(R^{I_s})\right) \middle| R^{I_s}, KB\right) \tag{30}$$

$K\left(R_k^{I_S}\right) \in \{0,1\}$ ist eine binäre Zufallsvariable, die den Wert 1 annimmt, wenn die Regionen $R_k^{I_S}$ konsistent sind und 0, wenn diese Region inkonsistent ist. Je mehr eine Region das Vorwissen erfüllt, desto größer ist die Konsistenzwahrscheinlichkeit dieser Region. Regionen, die das Vorwissen weniger erfüllen, haben entsprechend eine kleine Konsistenzwahrscheinlichkeit und sind somit inkonsistent. Die Abbildung 6-1 stellt die Schritte zur Schätzung der Konsistenzwahrscheinlichkeit von semantisch segmentierten Regionen anhand eines Beispielbildes des *DCS*-Validierungsdaten-satzes dar. Das Inputbild I (Bild aus [81]) wird zunächst mit dem *FCN-8s*-Modell semantisch segmentiert. Danach wird die Konsistenz der semantisch segmentierten Regionen aus R^{I_S} inferiert, gegeben sei das Vorwissen. Regionen mit einer Konsistenzwahrscheinlichkeit kleiner als dem empirisch bestimmten Schwellenwert (hier 0,45) sind inkonsistent (siehe weiß markierte Regionen im rechten Bild der Abbildung 6-1). In den nächsten Abschnitten wird das Vorwissen modelliert und die Parameter θ der Wahrscheinlichkeitsverteilung aus der Gleichung (30) gelernt. Die semantische Segmentierung wurde ausführlich in Kapitel 5 dargestellt und wird hier vorausgesetzt.

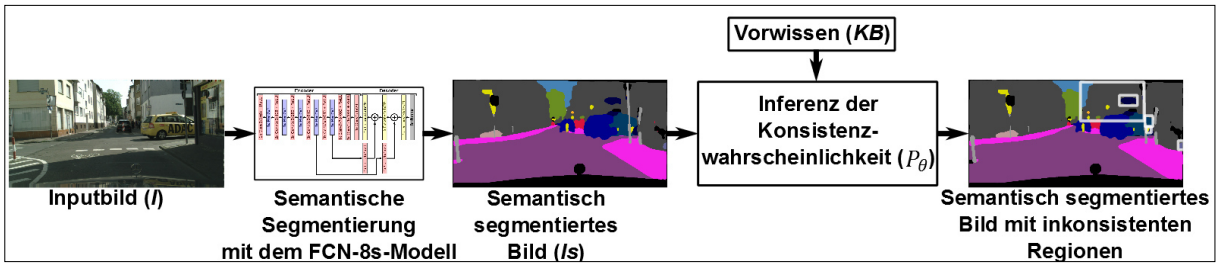


Abbildung 6-1: Grafische Darstellung der Aufgabe der Konsistenzschätzung von Regionen aus der semantischen Segmentierung anhand eines Beispielbildes des DCS-Validierungsdatensatzes (Inputbild aus [81])

6.2 Lösungsansatz

Zur Schätzung der Konsistenz von Regionen aus der semantischen Segmentierung wurde das in dieser Arbeit vorgeschlagene Konzept (siehe Abschnitt 4.2) angewendet. Die Abbildung 6-2 stellt die Anwendung dieses Konzepts dar. Zunächst wird das Vorwissen über Szenenelemente und Situationsaspekte in einer WB bestehend aus einer Ontologie und logischen Regeln modelliert (Schritt eins). Die WB wird dann mit Evidenzen inferiert, um neues Wissen zu gewinnen (Schritte zwei bis vier). Im nächsten Schritt wird die WB als Input für die Modellierung von MLN verwendet (Schritte fünf und sechs). Die MLN modellieren die Konsistenz von Regionen mit gegebenem Vorwissen. Die Parameter der modellierten MLN werden anhand von Annotationen der semantischen Segmentierung aus dem *DCS*-Datensatz gelernt (Schritte sieben bis neun). Gegeben die gelernten MLN und die mit dem *FCN-8s*-Modell semantisch segmentierten Regionen, wird die Konsistenz der segmentierten Regionen inferiert (Schritte zehn bis 14). Regionen, mit einer Konsistenzwahrscheinlichkeit kleiner als dem empirischen Schwellenwert (z. B. 0,45), sind im Bild in Schritt 14 weiß markiert. Eine ausführliche Beschreibung der Module aus der Abbildung 6-2 erfolgt in den nächsten Abschnitten. Insbesondere wird die Schätzung von Konsistenzwahrscheinlichkeiten im Abschnitt 6.4 erläutert.

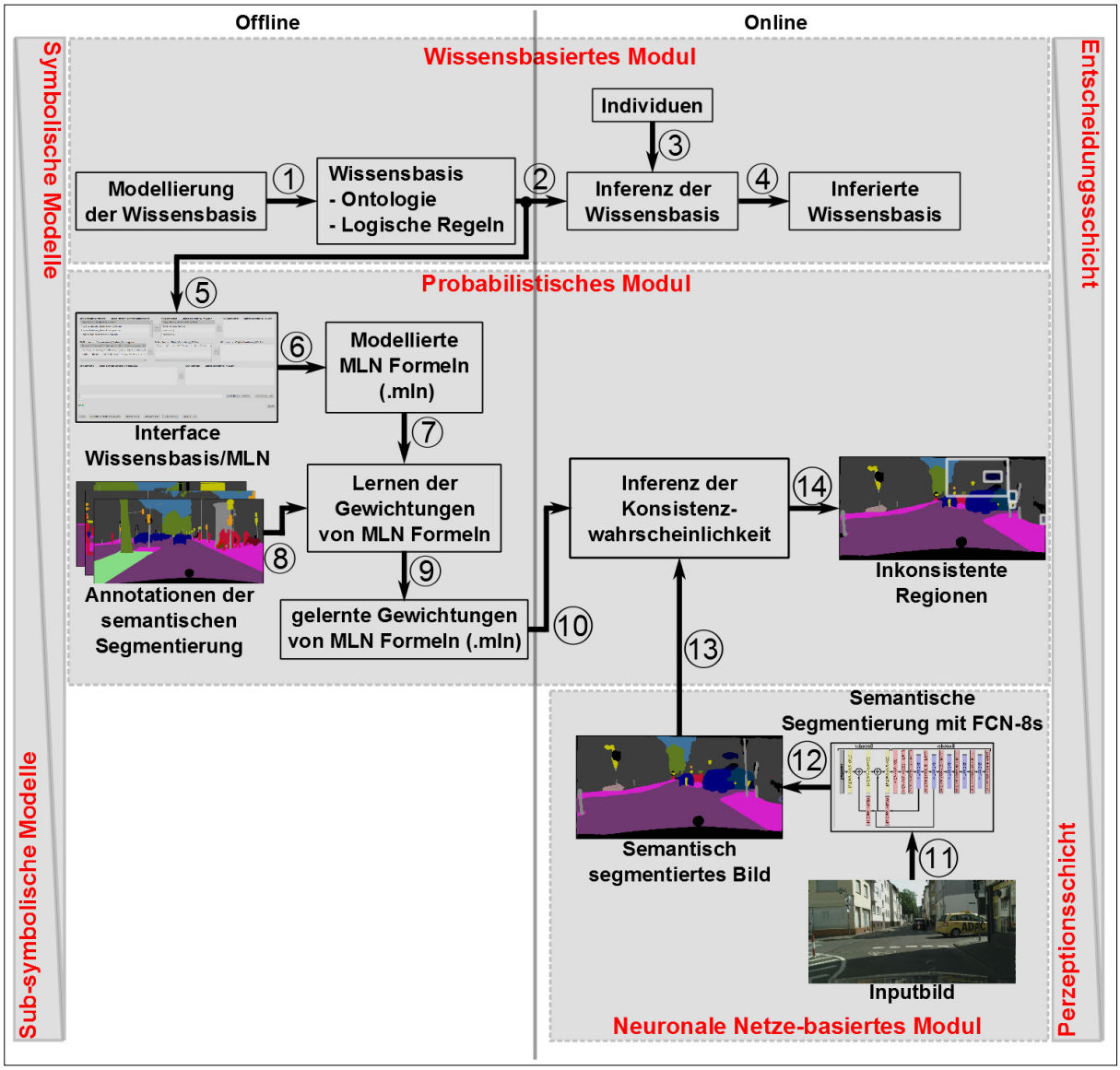


Abbildung 6-2: Konzeptuelle Sicht des Systems zur Konsistenzschätzung von Regionen der semantischen Segmentierung. Die Bilder und die Ground Truth der semantischen Segmentierung stammen aus dem DCS-Datensatz ([81]).

Ein Vorteil des vorgeschlagenen Ansatzes im Vergleich zu den Ansätze von Choi et al. [116] und Ruiz-Sarmiento et al. [117] liegt darin, dass das Vorwissen sowohl symbolisch als auch mit Unsicherheiten modelliert wird. So können Entscheidungen der Konsistenzschätzung besser interpretiert werden. Gleichzeitig können Unsicherheiten des Vorwissens erfasst werden. Darüber hinaus lassen sich Teile des Systems aufgrund der symbolischen Modellierung der WB und der expliziten Trennung des Vorwissens von TNN mit wenig Aufwand erweitern. Eine frühere Version des vorgeschlagenen Lösungsansatzes wurde in [147] publiziert.

6.3 Wissensbasiertes Modul

In diesem Modul wird zunächst das Vorwissen über Szenenelemente und Situationsaspekte modelliert. Danach wird neues Wissen inferiert (siehe Schritte eins bis vier der Abbildung 6-2). Diese Schritte werden im Detail in den nächsten Abschnitten beschrieben.

6.3.1 Modellierung der WB

Zur Modellierung der $\mathcal{WB} = \{\mathcal{O}, \mathcal{F}\}$ wurde, wie im Abschnitt 4.2.1 beschrieben, eine Ontologie $\mathcal{O}$ und eine Menge logischer Regeln $\mathcal{F}$ verwendet (siehe Schritt eins der Abbildung 6-2). Diese Modellierung fand in der Offlinephase statt.

6.3.1.1 Beschreibung der Ontologie

Die Ontologie $\mathcal{O} = \{\mathcal{B}, \mathcal{R}, \mathcal{T}Box, ABox, RBox\}$ enthielt als Begriffe Szenenelemente und weitere abstrakte Begriffe. Szenenelemente, die für diese Arbeit relevant waren, wurden durch die Klassen und Kategorien des DCS-Datensatzes definiert (siehe

Tabelle 5-1). Darüber hinaus wurde die Szene sowie die Konsistenz von Szenenelementen mit den abstrakten Begriffen $scene$ und $ObjConsistence$ modelliert. Somit war die Menge der für diese Arbeit relevanten Begriffe definiert durch

$$\begin{aligned} \mathcal{B} &= \mathcal{C} \cup \mathcal{K} \cup \{scene, ObjConsistence\} \\ \mathcal{C} &= \{road, sidewalk, building, wall, fence, pole, traffic\_light, \\ &\quad traffic\_sign, vegetation, terrain, sky, person, rider, car, truck, \\ &\quad bus, train, motorcycle, bicycle, unknow\} \\ \mathcal{K} &= \{void, flat, construction, object, nature, human, \\ &\quad vehicle, scene, ObjConsistence\} \end{aligned} \tag{31}$$

Die Kontextinformationen, die von Galleguillos und Belongie [115] vorgeschlagen und in Abschnitt 3.6.3 ausführlich beschrieben wurden, stellten die Relationsmenge $\mathcal{R}$ dar, die für diese Arbeit relevant war.

$$\begin{aligned} \mathcal{R} = \{&CoOccurence, OccurenceInScene, SupportedBy, Support, Above, Below, \\ &Inside, Around, PositionInScene, AspectRatio, AreaGreater, AreaSmaller\} \end{aligned} \tag{32}$$

$CoOccurence$ bedeutete, dass zwei Regionen in einem Bild gemeinsam auftraten. $OccurenceInScene$ stand für die Existenz eines Objekts in einer Szene. Die räumlichen Relationen $Support$, $Above$, $Below$, $Inside$ und $Around$ beschrieben die relative Position von zwei Regionen in einer Szene, während $PositionInScene$ die Position einer Region in einer Szene beschrieb. Der Skalierungskontext wurde durch die Relationen $AspectRatio$, $AreaGreater$ und $AreaSmaller$ definiert. $AspectRatio$ stand für das Seitenverhältnis einer Region. $AreaGreater$ modellierte die Relation zwischen zwei Regionen, wobei die Fläche der ersten Region größer als die der zweiten Region war. Bei $AreaSmaller$ war die Fläche der ersten Region kleiner als die der zweiten.

Neben den Begriffen und Relationen wurden *TBox*- und *RBox*-Axiome verwendet, um die Ontologie zu beschreiben. Die Tabelle 6-1 stellt diese Axiome der Ontologie dar. *TBox*-Axiome wurden vor allem verwendet, um die Taxonomie zwischen den Begriffen zu definieren. Neben der Taxonomie zwischen Kategorien und Klassen, die schon im *DCS*-Datensatz definiert wurde, wurden die Begriffe $conceptual\_object$ und $physical\_object$ als Spezialisierungen des *Top*-Begriffs $\top$ definiert. Der Begriff $physical\_object$ umfasste alle Szenenelemente, die reale Objekte und domänenunabhängig waren. Unter $conceptual\_object$ wurden Begriffe verstanden, die abstrakte Objekte beschrieben. Diese Begriffe waren von den Anwendungsdomänen abhängig. Bohlken et al. [148] verwendeten eine ähnliche Taxonomie. Der Vorteil dieser Modellierung war die Möglichkeit, das domänenunabhängige Wissen an einer zentralen Stelle zu modellieren und für die Modellierung von domänenabhängigem Wissen bereitzustellen. So konnten sich mehrere domänenabhängige Anwendungen die gleiche Ontologie teilen und diese mit domänenspezifischem Wissen ergänzen. Als weitere Elemente der Taxanomie wurden die Kategorien des *DCS*-Datensatzes als Spezialisierungen von $physical\_object$ modelliert, während die Begriffe $scene$ und $ObjConsistence$, die keine realen Szenenobjekte waren, als Spezialisierung von $conceptual\_object$ modelliert. *RBox*-Axiome modellierten die Eigenschaften von Relationen wie bspw. die Inverse, die Disjunktion und die Reflexivität. So wurde bspw. die Relation $Above$ als die Inverse von $Below$ definiert.

Tabelle 6-1: Beschreibung der TBox- und RBox-Axiome der Ontologie zur Modellierung des Kontextwissens von Szenenelementen aus dem DCS-Datensatz und der Relationen zwischen diesen Szenenelementen

Name	Axiome	Typ	Bedeutung

Name	Axiome	Typ	Bedeutung
A_0	$conceptual\_object \sqcup physical\_object \sqsubseteq \top$	*TBox*	Definition der Menge aller Begriffe
A_1	$ObjConsistence \sqcup scene \sqsubseteq conceptual\_object$	*TBox*	Begriffe von konzeptuellen Objekten
A_2	$flat \sqcup construction \sqcup object \sqcup nature \sqcup human \sqcup vehicle \sqcup void \sqsubseteq physical\_object$	*TBox*	Alle Kategorien sind spezielle physikalische Objekte
A_3	$disjoint(void, flat, construction, object, nature, human, vehicle)$	*TBox*	Die Menge der Kategorien ist disjunkt
A_4	$road \sqcup sidewalk \sqsubseteq flat$	*TBox*	Klassen sind spezielle Kategorien
A_5	$building \sqcup wall \sqcup fence \sqsubseteq construction$	*TBox*	Klassen sind spezielle Kategorien
A_6	$pole \sqcup traffic\_light \sqcup traffic\_sign \sqsubseteq object$	*TBox*	Klassen sind spezielle Kategorien
A_7	$vegetation \sqcup terrain \sqsubseteq nature$	*TBox*	Klassen sind spezielle Kategorien
A_8	$person \sqcup rider \sqsubseteq human$	*TBox*	Klassen sind spezielle Kategorien
A_9	$car \sqcup truck \sqcup bus \sqcup train \sqcup motorcycle \sqcup bicycle \sqsubseteq vehicle$	*TBox*	Klassen sind spezielle Kategorien
A_{10}	$unknow \sqsubseteq void$	*TBox*	Klassen sind spezielle Kategorien
A_{11}	$disjoint(road, sidewalk, building, wall, fence, pole, traffic\_light, traffic\_sign, vegetation, terrain, sky, person, rider, car, truck, bus, train, motorcycle, bicycle, unknow)$	*TBox*	Die Menge der Klassen ist disjunkt
A_{12}	$Above \equiv Below^-$	*RBox*	Inverse Relationen
A_{13}	$Inside \equiv Around^-$	*RBox*	Inverse Relationen
A_{14}	$Support \equiv SupportedBy^-$	*RBox*	Inverse Relationen
A_{15}	$AreaGreater \equiv AreaSmaller^-$	*RBox*	Inverse Relationen
A_{16}	$disjoint(Support, Above, Below, Inside, Around)$	*RBox*	Disjunkte Relationen
A_{17}	$CoOccurence \equiv CoOccurence^-$	*RBox*	Symmetrische Relation

Die Abbildung 6-3 stellt die mit *Protégé* [198] modellierte Taxonomie $\mathcal{T}$ der *TBox*-Axiome aus der Tabelle 6-1 dar. Dabei wurde *OWL-2* als Beschreibungssprache verwendet. Die Knoten dieses Graphs repräsentieren die Begriffe der Ontologie und die gerichteten Kanten modellieren die Spezialisierung (*Is-a*-Beziehung) zwischen den Begriffen. Der Wurzelknoten wird in *OWL-2 owl:Thing* genannt und entspricht dem *Top*-Begriff $\top$.

Abbildung 6-3: Taxonomie $\mathcal{T}$ der Ontologie von Szenenelementen aus dem DCS-Datensatz sowie Begriffe zur Modellierung von Kontextwissen. Die Grafik wurde mit dem OWLViz-Tool von Protégé generiert [198].

6.3.1.2 Erweiterung der Ontologie mit logischen Regeln

Logische Regeln dienten dazu, die Ontologie mit komplexerem Wissen zu erweitern. Hier wurden *FOL*-Formeln verwendet, da die Beschreibungslogik der Ontologie ein Unterteil dieser Prädikatenlogik war. Die Gleichung (33) zeigt die Menge der logischen Formeln zur Beschreibung der Konsistenz von zwei Regionen o_1 und o_2 der semantischen Klassen c_1 und c_2, gegeben die Relation $r \in R$ aus der Relationsmenge R der Gleichung (32). c_1 und c_2 waren Elemente der semantischen Klassenmenge C aus der Gleichung (31).

$$\mathcal{F} = \{\forall o_1, \forall o_2, \forall c_1, c_2 \in \mathcal{C}, \forall r \in \mathcal{R}: (c_1(o_1) \wedge c_2(o_2) \wedge r(o_1, o_2)) \Longrightarrow (ObjConsistence(o_1) \wedge ObjConsistence(o_2))\} \quad (33)$$

Die Formelmenge $\mathcal{F}$ hatte also als Prädikate die Begriffe und Relationen der Ontologie $\mathcal{O}$. Betrachtete man alle mögliche Werte von c_1, c_2 und r, ergaben sich insgesamt $(|\mathcal{R}| * |\mathcal{C}|)^2$ logische Formeln aus der Gleichung (33). Eine Instanz der Formel aus der Gleichung (33) für $c_1 = car$, $c_2 = road$ und $r = Above$ sah wie folgt aus:

$$\forall o_1, \forall o_2: (car(o_1) \wedge road(o_2) \wedge Above(o_1, o_2)) \Longrightarrow (ObjConsistence(o_1) \wedge ObjConsistence(o_2)) \quad (34)$$

Die Modellierung der logischen Regeln aus der Gleichung (33) erfolgte in *Protégé* mit der Sprache *Semantic Web Rule Language* (*SWRL*). *SWRL* ist eine Sprache zur Modellierung von logischen Regeln als Horn-Klausel in Zusammenhang mit *OWL*-Ontologien [199].

6.3.2 Inferenz anhand der modellierten WB

In der Inferenzphase wurde der *Pellet-Reasoner* [32], der in *Protégé* verfügbar war, verwendet. Dieser *Reasoner* prüfte zunächst die Konsistenz der WB. Danach wurden die in Abschnitt 2.1.2.3 vorgestellten Subsumptions-, Instanz-, und Relationstests ausgeführt. Darüber hinaus wurde eine logische Inferenz der logischen Regeln, wie im Abschnitt 2.1.1.3 beschrieben, ausgeführt. Die Tabelle 6-2 stellt zwei Beispiele der Inferenz für zwei Individuen i_1 und i_2, die Instanzen der Begriffe car und $road$ waren, dar. Im ersten Fall (erste Zeile der Tabelle 6-2) wurde zu den beiden Individuen i_1 und i_2 die Relation $Above(i_1, i_2)$ zwischen den Instanzen $i_1$und i_2 als Evidenz festgelegt. Der Konsistenztest ergab, dass die WB konsistent war. Dazu lieferte der Instanztest in Kombination mit der Inferenz der logischen Regeln, dass i_1 und i_2 Individuen des Begriffs $ObjConsistence$ waren, da die Prämissen der Implikation in der Formel (34) erfüllt waren. Darüber hinaus wurde die Relation $Below(i_2, i_1)$ inferiert, da diese als Inverse der Relation $Above$ in der Ontologie definiert wurde (siehe Tabelle 6-1, Axiom A_{12}). In der zweiten Zeile der Tabelle 6-2 wurde zu dem Input des ersten Falls die Relation $Below(i_1, i_2)$ als Evidenz hinzugefügt. Die Inferenz ergab eine inkonsistente WB, da die Relation $Above$ und $Below$ als disjunkt in der Ontologie definiert waren. Somit dürften die Relationen $Above(i_1, i_2)$ und $Below(i_1, i_2)$ nicht gleichzeitig als Evidenz existieren. Alle anderen Tests waren aufgrund der inkonsistenten Ontologie nicht ausführbar. *Protégé* lieferte eine textuelle Erklärung der Inkonsistenz, die benutzt wurde, um die Inkonsistenz aufzuheben. Im ersten Fall dauerte die Inferenz der WB ca. 140 ms. Im zweiten Fall betrug die Inferenzzeit ca. 20 ms. Die Inferenzzeit im zweiten Fall war deutlich kleiner als die Inferenzzeit im ersten Fall, da die Ontologie im zweiten Fall inkonsistent war und einige Tests, die im ersten Fall ausgeführt wurden, im zweiten Fall ausgefallen waren. Allgemein hängte die Inferenzzeit exponentiell von der Größe der WB ab (siehe Abschnitt 2.1.2.3).

Tabelle 6-2: Beispielergebnisse der Inferenz anhand der modellierten WB

Fall	Input der WB	Inferenz-Output
Fall 1	**Individuen**: $car(i_1)$ und $road(i_2)$	**Konsistenztest**: konsistent
	Relationen: $Above(i_1, i_2)$	**Instanzstest**: $ObjConsistence\ (i_1)$ und $ObjConsistence\ (i_2)$
		Relationstest: $Below(i_2, i_1)$
		Laufzeit: 140 ms
Fall 2	**Individuen**: $car(i_1)$ und $road(i_2)$	**Konsistenztest**: inkonsistent
	Relationen: $Above(i_1, i_2)$ und $Below(i_1, i_2)$	**Instanzstest**: nicht ausführbar
		Relationstest: nicht ausführbar
		Laufzeit: 20 ms

6.4 Erweiterung der WB mit probabilistischen graphischen Modellen

Die Formelmenge in der Gleichung (33) bedeutete, dass zwei Regionen immer konsistent waren, gegeben die Prämissen der Implikation. Die Prämissen der Implikation waren die semantischen Klassen der Regionen und die Relation zwischen diesen Regionen. In der Realität war dies nicht immer wahr, da Regionen abhängig von ihren semantischen Klassen bestimmte Relationen präferierten und nur in solchen Fällen konsistent waren. Zum Beispiel haben zwei Regionen i_1 und i_2 der Klassen $road(i_1)$ und $sky(i_2)$ meistens in realen Szenen die Relation $Above(i_2, i_1)$. Traten die Regionen i_1 und i_2 in einer Szene mit der Relation $Above(i_2, i_1)$ auf, dann waren diese Regionen sehr wahrscheinlich konsistent. Traten diese Regionen aber mit der Relation $Below(i_2, i_1)$ auf, dann waren diese Regionen mit einer hohen Wahrscheinlichkeit inkonsistent. Dies bedeutete, dass die Formel der Gleichung (33) mit Unsicherheiten erweitert werden sollte. Wie im Konzept im Abschnitt 4.2.2.1 beschrieben, stellen MLN ein passendes *Framework* dar, um *FOL*-Regeln mit Unsicherheiten zu modellieren. Die Schritte zum Modellieren, Lernen und Inferieren von MLN-Modellen sind in der Abbildung 6-2 in den Schritten fünf bis 14 dargestellt. Diese Schritte werden ausführlich in den nächsten Abschnitten vorgestellt.

6.4.1 Modellierung von Formeln des MLN-Modells

Das MLN-Modell, das in dieser Arbeit verwendet wurde, besaß als logische Formeln die Begriffe, Relationen, Axiome und logische Formeln der im Abschnitt 6.3.1 modellierten WB. Eine grafische Schnittstelle, um aus der WB MLN-Formeln (*.mln*-Dateien) zu generieren, wurde im Rahmen einer im DLR Institut für Verkehrssystemtechnik von dem Autor dieser Arbeit betreuten Bachelor-Arbeit [200] entwickelt und in dieser Arbeit übernommen. Die Abbildung 6-4 stellt diese Schnittstelle grafisch dar. Begriffe und Relationen der Ontologie (siehe *OWL concepts & relations* in der Abbildung 6-4) sowie logische Regeln der WB (siehe *SWRL-rules* in der Abbildung 6-4) wurden aus der von *Protégé* generierten *.owl*-Datei der WB extrahiert und als Prädikate für die die Generierung von MLN-Prädikaten (siehe *MLN predicates* in der Abbildung 6-4) und MLN-Formeln (siehe *MLN rules* in der Abbildung 6-4) bereitgestellt. Wertebereiche, die in der Ontologie modelliert wurden, wurden im Fenster *OWL domains* geladen und in das MLN-Modell über den Bereich *MLN domains* integriert. Weitere Prädikate und logische Regeln für das MLN-Modell konnten im Textfeld im unteren Bereich der grafischen Schnittstelle eingegeben werden. Mit den Knöpfen *Insert the FOL Predicate* und *Insert the FOL Rule* konnten die eingegebenen Prädikate und Regeln in die MLN-Formeln integriert werden. Dabei wurde sichergestellt, dass die Prädikate der eingegebenen logischen Regeln in der Ontologie vorhanden waren, und dass die Syntax von MLN-Formeln eingehalten wurde. Die grafische Schnittstelle bot auch die Möglichkeit an, MLN aus Textdateien mit *FOL*-Formeln zu generieren (siehe *FOL predicates* und *FOL rules* in der Abbildung 6-4). Sämtliche Knöpfe zum Aufladen der Inputdateien (*.owl*, *.mln* und *.txt*) und zum Speichern der editierten Dateien (*.mln* und *.txt*) waren im unteren Bereich der grafischen Schnittstelle angeordnet.

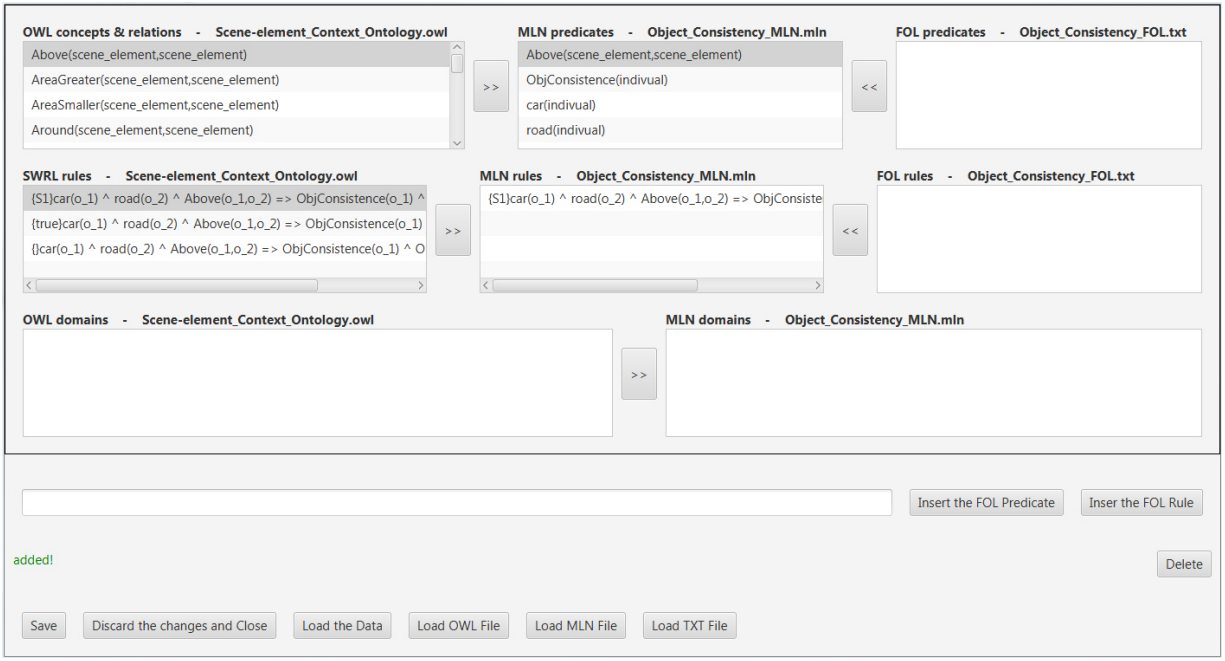


Abbildung 6-4: Grafische Schnittstelle zur Generierung von MLN-Formeln aus einer WB bestehend aus einer OWL-Ontologie und SWRL-Regeln

Die Abbildung 6-5 stellt einen Ausschnitt der modellierten MLN-Formeln $\mathcal{F}_{mln}$ mit der in der Abbildung 6-4 dargestellten grafischen Schnittstelle dar. Diese logischen Formeln dienten als Input zum Lernen der Gewichtungen von MLN-Formeln zur Schätzung der Konsistenz von semantisch segmentierten Regionen basierend auf den räumlichen Relationen $Support$, $SupportedBy$, $Above$, $Below$, $Inside$ und $Around$. Die Prädikate dieser MLN-Formeln waren eine Teilmenge der Begriffe und Relationen der Ontologie aus den Gleichungen (31) und (32). Diese Prädikate sind in der Abbildung 6-5 in den Zeilen drei bis 13 zu finden. Darüber hinaus wurde eine Teilmenge der Ontologie-Axiome aus der Tabelle 6-1 als Axiome der MLN-Formeln in den Zeilen 16 bis 25 der Abbildung 6-5 modelliert. MLN-Axiome sind logische Formeln, die mit einem Punktzeichen enden. Die Gewichtungen der Axiome werden nicht gelernt, sondern auf einen hohen Wert gesetzt. Evidenzen, die die Axiome verletzen, werden somit unwahrscheinlicher. Eine Teilmenge der Formelmenge $\mathcal{F}$ aus Gleichung (33), die die Konsistenz von Regionen für die räumlichen Relationen $Support$, $SupportedBy$, $Above$, $Below$, $Inside$ und $Around$ modellierte, ergänzte die MLN-Formeln (siehe Zeilen 29 bis 39 der Abbildung 6-5). Die Einschränkung auf die o. g. räumlichen Relationen diente zur Reduktion der Komplexität des MLN-Modells, weil die Anzahl der Formeln von $\mathcal{F}$ aus der Gleichung (33) exponentiell mit der Anzahl der Relationen und semantischen Klassen anstieg (siehe Abschnitt 6.3.1.2). Wie die Gewichtungen der MLN-Formeln gelernt wurden, wird im nächsten Abschnitt beschrieben.

```
//Predicate
Above(scene_element,scene_element)
Below(scene_element,scene_element)
Inside(scene_element,scene_element)
Around(scene_element,scene_element)
Support(scene_element,scene_element)
SupportedBy(scene_element,scene_element)
ObjConsistence(scene_element)
car(scene_element)
road(scene_element)
...
unknow(scene_element)

//Axioms: Reversible
Above(o1,o2) <=> Below(o2,o1).
Inside(o1,o2) <=> Around(o2,o1).
Support(o1,o2) <=> SupportedBy(o2,o1).
//Axioms: disjoint
(Above(o1,o2) ^ !Below(o1,o2) ^ !Inside(o1,o2) ^ !Around(o1,o2) ^ !Support(o1,o2) ^ !SupportedBy(o1,o2)) v
(!Above(o1,o2) ^ Below(o1,o2) ^ !Inside(o1,o2) ^ !Around(o1,o2) ^ !Support(o1,o2) ^ !SupportedBy(o1,o2)) v
(!Above(o1,o2) ^ !Below(o1,o2) ^ Inside(o1,o2) ^ !Around(o1,o2) ^ !Support(o1,o2) ^ !SupportedBy(o1,o2)) v
(!Above(o1,o2) ^ !Below(o1,o2) ^ !Inside(o1,o2) ^ Around(o1,o2) ^ !Support(o1,o2) ^ !SupportedBy(o1,o2)) v
(!Above(o1,o2) ^ !Below(o1,o2) ^ !Inside(o1,o2) ^ !Around(o1,o2) ^ Support(o1,o2) ^ !SupportedBy(o1,o2)) v
(!Above(o1,o2) ^ !Below(o1,o2) ^ !Inside(o1,o2) ^ !Around(o1,o2) ^ !Support(o1,o2) ^ SupportedBy(o1,o2)).

//Rules
//{S1}
car(o_1) ^ road(o_2) ^ Above(o_1,o_2) => ObjConsistence(o_1) ^ ObjConsistence(o_2)
//{S2}
car(o_1) ^ road(o_2) ^ Below(o_1,o_2) => ObjConsistence(o_1) ^ ObjConsistence(o_2)
//{S3}
car(o_1) ^ road(o_2) ^ Inside(o_1,o_2) => ObjConsistence(o_1) ^ ObjConsistence(o_2)
//{S4}
car(o_1) ^ road(o_2) ^ Around(o_1,o_2) => ObjConsistence(o_1) ^ ObjConsistence(o_2)
//{S5}
car(o_1) ^ road(o_2) ^ Support(o_1,o_2) => ObjConsistence(o_1) ^ ObjConsistence(o_2)
//{S6}
car(o_1) ^ road(o_2) ^ SupportedBy(o_1,o_2) => ObjConsistence(o_1) ^ ObjConsistence(o_2)
...
```

Abbildung 6-5: Ausschnitt der modellierten MLN-Formeln $\mathcal{F}_{mln}$ mit der in der Abbildung 6-4 dargestellten grafischen Schnittstelle. Diese Formeln dienten zum Lernen der Gewichtung des MLN-Modells zur Schätzung der Konsistenz von Regionen der semantischen Segmentierung basierend auf den räumlichen Relationen $Support$, $SupportedBy$, $Above$, $Below$, $Inside$ und $Around$.

6.4.2 Lernen der Gewichtungen von MLN-Formeln

Zum Lernen von Gewichtungen der MLN-Formeln aus dem Abschnitt 6.4.1 müssen zunächst Trainingsdaten generiert werden. In dieser Arbeit wurde der *DCS*-Trainingsdatensatz verwendet. Dieser Datensatz wurde ausführlich im Abschnitt 5.4.1 beschrieben. Das *Alchemy*-Tool [201] wurde für das Training verwendet. Dieses Tool nahm als Input die logischen Regeln des MLN-Modells und die Trainingsdaten als Textdateien. Als Output wurden die Gewichtungen der logischen Formeln mit den Algorithmen aus dem Abschnitt 2.3.2 gelernt. Die Trainingsdaten enthielten die Wahrheitswerte der atomaren Formeln von $\mathcal{F}_{mln}$ für unterschiedliche Belegungen. Die logischen Konstanten der Belegungen waren die semantisch segmentierten Regionen, die in den annotierten Bildern vorhanden waren. Atomare Formeln von $\mathcal{F}_{mln}$ waren die semantischen Klassen der Regionen, die räumlichen Relationen zwischen Regionspaaren und die Konsistenz von Regionen. Die semantischen Klassen der Regionen waren in den annotierten Daten vorhanden. Die räumlichen Relationen zwischen Regionspaaren und die Konsistenz von Regionen wurden geschätzt. Die nächsten Abschnitte beschreiben den Lernprozess im Detail.

6.4.2.1 Schätzung der räumlichen Relation zwischen zwei semantisch segmentierten Regionen

Gegeben seien zwei Regionen $o_i \in R^{I_S}$ und $o_j \in R^{I_S}$ im semantisch segmentierten Bild I^S. Gesucht ist die räumliche Relation zwischen den beiden Regionen aus der Menge der räumlichen Relationen

$$\mathcal{R}_s = \{Support, SupportedBy, Above, Below, Inside, Around\} \tag{35}$$

6.4.2.1.1. Schätzung der räumlichen Relation $\{Above, Below, Inside, Around\}$

Zur Schätzung der räumlichen Relationen aus der Teilmenge

$$\mathcal{R}_{s1} = \{Above, Below, Inside, Around\} \quad (36)$$

zwischen den Regionen o_i und o_j wurde von Galleguillos et al. [202, 203] der Merkmalvektor $F(o_i, o_j)$ vorgeschlagen.

$$F(o_i, o_j) = \left(\mu(o_i, o_j), DoA(o_i, o_j), DoA(o_j, o_i)\right)^T \quad (37)$$

Und

$$\mu(o_i, o_j) = \frac{\mu_y(o_i) - \mu_y(o_j)}{Height(I^s)} \quad (38)$$

$\mu(o_i, o_j) \in [-1, +1]$ ist die mit der Höhe des Bildes normierte Differenz zwischen den Schwerpunkten der Flächen der Regionen o_i und o_j entlang der Y-Achse.

$$DoA(o_i, o_j) = \frac{\beta(o_i)/\beta(o_j)}{A(o_i)} \quad (39)$$

$DoA(o_i, o_j) \in [0,1]$ (*DoA*: *Difference over Area*) ist die Differenzmenge zwischen den *Bounding*-Boxen der Regionen o_i und o_j normiert mit der Fläche der Region o_i. Die Funktion *DoA* ist antisymmetrisch ($DoA(o_i, o_j) \neq DoA(o_j, o_i)$), da die Flächen der Regionen o_i und o_j unterschiedlich sind ($A(o_i) \neq A(o_j)$) und der Differenzoperator / antisymmetrisch ist ($\left(\beta(o_i)/\beta(o_j)\right) \neq \left(\beta(o_j)/\beta(o_i)\right)$).

Basierend auf dem o. g. Merkmalvektor $F(o_i, o_j)$ wurden dann in [202, 203] vier Clusterzentren für die räumlichen Relationen aus der Menge $\mathcal{R}_{s1}$ gelernt. Die gelernten Clusterzentren entsprachen der Intuition, dass die Relationen $Above$ und $Below$ sich vor allem durch die Verteilung der Schwerpunktzentren der Regionen entlang der Y-Achse unterschieden. So hatten Punkte, die zum Cluster der Relation $Above$ gehörten, meistens negative $\mu(o_i, o_j)$-Werte. Dies bedeutete, dass der Y-Wert $\mu_y(o_i)$ des Zentrums der Region o_i kleiner als der Y-Wert $\mu_y(o_j)$ des Zentrums der Region o_j war. Punkte des Clusters der Relation $Below$ hatten im Gegenteil positive $\mu(o_i, o_j)$-Werte. Bei den Relationen $Inside$ und $Around$ unterschieden sich die Punkte der Cluster durch die *DoA*-Werte. Punkte, die zum Cluster der Relation $Inside$ gehörten, hatten $DoA(o_i, o_j)$-Werte, die gegen 0 gingen, während die $DoA(o_j, o_i)$-Werte sich 1 annäherten. Dies bedeutet, dass die Region o_i, die in der Region o_j war, meistens eine deutlich kleinere Fläche als die Region o_j hatte. Für den Cluster der Relation $Around$ zeigten die dazu gehörten Punkte im Merkmalraum das umgekehrte Verhalten im Vergleich zum Cluster der Relation $Inside$.

In dieser Arbeit wurde der o. g. Merkmalvektor $F(o_i, o_j)$ aus [202, 203] für die Schätzung der Relationen aus $\mathcal{R}_{s1}$ übernommen. Dieser Merkmalvektor wurde verwendet, um vier Clusterzentren für die Relationen aus der Menge $\mathcal{R}_{s1}$ mit dem *DCS*-Trainingsdatensatz zu lernen. Hier wurde lediglich $1/5$ des *DCS*-Trainingsdaten-satzes zum Lernen verwendet. Die gelernten Clusterzentren sind in der Tabelle 6-3 zu finden. Die Clusterzentren entsprachen den Cluster-Eigenschaften aus [202, 203]. Die Clusterzentren der Klassen $Above$ und $Below$ unterschieden sich hauptsächlich durch die normierte Differenz $\mu(o_i, o_j)$ zwischen den Schwerpunktzentren der Regionen o_i und o_j. Im Gegenteil waren für die Relationen $Inside$ und $Around$ die Merkmale $DoA(o_i, o_j)$ und $DoA(o_j, o_i)$ am wichtigsten.

Tabelle 6-3: Gelernte Clusterzentren der räumlichen Relationen $Above$, $Below$, $Inside$ und $Around$ auf dem DCS-Trainingsdatensatz

Relation	Clustername	$\mu(o_i, o_j)$	$DoA(o_i, o_j)$	$DoA(o_j, o_i)$
Above	$Cluster_{abv}$	-0.144469	0.998424	0.998865
Below	$Cluster_{blw}$	0.109163	0.998583	0.998249
Inside	$Cluster_{ins}$	-0.01398	0.94131	0.0566892

Relation	Clustername	$\boldsymbol{\mu(o_i, o_j)}$	$\boldsymbol{DoA(o_i, o_j)}$	$\boldsymbol{DoA(o_j, o_i)}$
Around	$Cluster_{ard}$	0.0143428	0.0571607	0.949507

Zum Schätzen der räumlichen Relation aus der Menge $\mathcal{R}_{s1}$ zwischen zwei gegebenen Regionen o_i und o_j wurde zunächst der Merkmalvektor $F(o_i, o_j)$ berechnet. Danach wurde die euklidische Distanz zwischen $F(o_i, o_j)$ und allen Clusterzentren der Tabelle 6-3 berechnet. Die Relation des Clusterzentrums mit der kürzesten Distanz zu $F(o_i, o_j)$ wurde dann als Relation zwischen den Regionen o_i und o_j gewählt. Die Abbildung 6-6, Abbildung 6-7, Abbildung 6-8 und Abbildung 6-9 stellen die Häufigkeitsmatrizen der geschätzten Relationen auf dem *DCS*-Trainingsdatensatz für die Relationen aus der Menge $\mathcal{R}_{s1}$ dar. Die Häufigkeitsmatrizen wurden anhand der Wahrscheinlichkeitsverteilung

$$\pi_r(ind(c_1), ind(c_2)) = P((r \in \mathcal{R}_{s1}) | c_1 \in \mathcal{C}, c_2 \in \mathcal{C}) \quad (40)$$

generiert, wobei r eine räumliche Relation aus $\mathcal{R}_{s1}$ war. c_1 und c_2 waren die semantischen Klassen aus der Menge $\mathcal{C}$. $ind(c_1)$ und $ind(c_2)$ waren die Klassen-IDs von c_1 und c_2. Die Wahrscheinlichkeitsverteilung aus der Gleichung (40) erfüllte die folgende Bedingung:

$$\forall c_1, c_2 \in \mathcal{C}, c_1 \neq c_2, \sum_{r \in \mathcal{R}_{s1}} \pi_r(ind(c_1), ind(c_2)) = 1 \quad (41)$$

Die Interpretation der Häufigkeitsmatrizen zeigte, dass die räumliche Anordnung von Regionen in natürlichen Bildern von diesen Häufigkeitsmatrizen richtig erfasst wurde. So hatten Klassen wie *sky*, *traffic light* und *traffic sign* meistens die Relation $Above$ zu anderen Klassen. Diese Schätzung war konsistent, da diese Klassen in natürlichen Bildern oft im oberen Bereich der Bilder zu finden waren. Beispielsweise bedeutete $\pi_{abv}(ind(sky), ind(roab)) = 98{,}8\ \%$ in der Abbildung 6-6, dass $98{,}8\ \%$ der Regionen der Klasse *sky* oberhalb von Regionen der Klasse *road* in den Trainingsdaten lagen. Dies entsprach der natürlichen Anordnung der Klassen *sky* und *road*. Darüber hinaus hatten die Häufigkeitsmatrizen der Relationen $Above$ und $Below$ mehr rot-markierte Zellen (siehe π_{abv} in der Abbildung 6-6 und π_{blw} in der Abbildung 6-7) als die Häufigkeitsmatrizen der Relationen $Inside$ und $Around$ (siehe π_{ins} in der Abbildung 6-8 und π_{ard} in der Abbildung 6-9), da die Relationen $Above$ und $Below$ häufiger als $Inside$ und $Around$ auftraten. Diese Beobachtung ließ sich dadurch erklären, dass die Relationen $Inside$ und $Around$ häufig nur zwischen benachbarten Regionen vorkamen, während die Relationen $Above$ und $Below$ auch nicht benachbarte Regionspaare betrachteten. Somit hatten die Relationen $Above$ und $Below$ mehr Regionspaare als $Inside$ und $Around$. Die Häufigkeitswerte $\pi_{ard}(ind(road), ind(car)) = 30{,}1\ \%$ und $\pi_{ard}(ind(sidewalk), ind(car)) = 7{,}4\ \%$ zeigten, dass Regionen der Klasse *road* häufiger um Regionen der Klasse *car* herum waren als Regionen der Klasse *sidewalk*. Dies entsprach der Erwartung, dass „*Fahrzeuge üblicherweise auf der Straße fahren*“. Die Intuition, dass „*Fußgänger auf dem Fußgängerweg gehen*“ wurde durch die Häufigkeitswerte $\pi_{ard}(ind(road), ind(person)) = 24{,}8\ \%$, $\pi_{ard}(ind(building), ind(person)) = 25{,}5\ \%$ und $\pi_{ard}(ind(sidewalk), ind(person)) = 16{,}4\ \%$ widerlegt. Die Häufigkeitswerte zeigten eher, dass Straßen und Gebäude häufiger um Fußgänger herum waren als Fußgängerwege um Fußgänger. Dies lag daran, dass zum einen die Trainingsdaten öfter urbane Szenen zeigten, in denen Fußgänger die Straße überquerten oder Gebäude im Hintergrund waren. Darüber hinaus war die Relation $Inside$ nicht immer äquivalent zu der *Ist-auf*-Relation. Die *Ist-auf*-Relation war mehr mit der Relation $Support$ äquivalent. Die Relation $Support$ wird später in dieser Arbeit erläutert, um diese Behauptung zu bekräftigen. Eine weitere Beobachtung der Häufigkeitsmatrizen zeigte, dass π_{abv} die transponierte Matrix von π_{blw} ($\pi_{abv} = (\pi_{blw})^T$) und π_{ins} die transponierte Matrix von π_{ard} ($\pi_{ins} = (\pi_{ard})^T$) war. Damit erfüllten diese Matrizen die Axiome A_{10} und A_{11} der Tabelle 6-1, wo die Relationen $Above$ und $Inside$ jeweils als Inversen der Relationen $Below$ und $Around$ in der WB modelliert wurden.

		road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	unknow	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	Marg
road	0	49.3	26.0	22.4	28.1	27.0	24.6	8.8	16.6	18.9	27.4	0.6	24.4	26.1	23.9	19.9	18.6	16.5	29.5	28.3	24.5	23.1
sidewalk	1	24.8	66.3	24.1	30.5	29.5	27.3	5.1	16.8	21.1	30.4	0.6	26.5	30.7	32.8	17.8	19.6	16.7	34.9	30.7	28.0	26.2
building	2	39.3	36.6	52.1	43.4	39.4	27.6	30.9	30.4	25.5	49.4	20.9	29.9	34.9	31.9	36.7	37.1	35.5	40.4	36.0	28.0	31.1
wall	3	37.0	38.6	28.8	85.2	34.4	35.3	16.4	25.0	25.2	46.0	2.5	37.3	42.6	41.1	29.2	31.6	26.3	51.3	44.8	35.6	34.2
fence	4	39.2	43.2	28.1	39.1	82.8	36.0	16.9	26.6	25.0	48.3	4.1	40.0	44.5	43.4	29.5	33.9	26.0	52.9	45.4	36.3	35.3
pole	5	38.9	38.6	27.4	43.3	42.9	75.3	33.6	34.5	28.0	44.0	22.4	39.7	46.2	36.6	45.4	41.1	39.3	49.2	44.4	30.1	35.0
traffic light	6	89.3	94.4	31.6	81.0	80.0	34.3	97.6	46.1	32.4	89.9	22.8	74.8	88.7	72.5	69.4	67.8	60.0	94.6	95.2	42.8	51.6
traffic sign	7	89.6	78.3	30.8	89.1	85.2	36.0	45.7	97.0	31.5	74.2	22.2	66.0	75.3	60.4	58.9	58.4	54.3	83.8	80.8	41.6	48.3
vegetation	8	42.9	43.5	26.0	40.6	39.7	29.0	32.2	31.5	53.8	41.4	17.6	37.6	42.3	36.0	39.1	37.2	40.2	50.8	46.6	29.6	33.1
terrain	9	26.1	32.3	28.3	37.1	33.0	31.4	8.4	19.7	22.4	79.0	1.5	32.4	34.5	35.3	19.1	17.0	17.4	42.0	39.8	31.3	29.2
sky	10	98.8	99.2	34.4	96.0	93.2	45.7	55.1	65.2	36.1	97.6	60.9	95.1	97.9	96.8	93.2	91.6	85.2	99.4	99.5	47.1	59.3
person	11	37.3	38.6	24.8	41.1	38.8	35.8	20.1	26.5	25.1	51.3	2.4	84.3	45.0	37.2	34.8	37.3	28.3	48.3	43.3	34.0	33.8
rider	12	39.9	47.9	26.1	44.8	41.9	39.4	7.8	20.0	24.4	55.3	1.3	42.3	73.7	41.6	25.2	32.3	25.7	34.5	29.7	39.1	34.2
car	13	29.2	38.2	23.7	38.4	36.3	30.4	13.8	19.9	21.8	40.7	1.0	33.5	38.1	81.9	27.2	27.1	24.4	40.7	39.2	30.4	30.0
truck	14	60.1	72.2	32.2	64.3	63.4	41.8	24.3	32.7	30.8	77.1	2.2	51.5	66.9	45.1	79.5	44.2	33.3	81.0	71.9	41.1	44.0
bus	15	60.1	68.2	32.5	61.0	61.4	39.5	24.3	32.8	28.5	74.9	2.5	46.8	51.6	47.5	46.2	79.1	50.0	70.0	66.5	41.4	43.1
train	16	59.1	73.9	31.4	64.9	55.0	38.2	23.4	30.6	29.0	75.6	2.6	46.3	61.4	58.7	66.7	46.9	76.8	72.7	73.7	38.0	42.5
motorcycle	17	31.7	40.8	24.3	40.6	37.2	36.2	5.4	15.0	23.4	47.2	0.6	37.0	25.6	40.4	13.9	27.5	22.7	84.7	40.8	37.5	31.9
bicycle	18	29.6	33.9	23.4	38.1	33.3	33.4	4.8	17.9	22.9	42.5	0.5	34.8	22.3	38.7	20.1	22.2	19.5	44.0	80.4	34.4	29.8
unknow	19	35.6	35.3	27.7	41.6	41.5	30.0	41.4	40.1	28.7	39.9	23.3	36.8	44.0	35.3	43.1	42.0	39.1	47.2	41.4	75.1	34.7

Abbildung 6-6: Häufigkeitsmatrizen π_{abv} der geschätzten Relationen Above auf dem DCS-Trainingsdatensatz. Die Zahlen sind in Prozent angegeben.

		road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	unknow	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	Marg
road	0	49.3	24.8	39.3	37.0	39.2	38.9	89.3	69.6	42.9	26.1	98.8	37.3	39.9	29.2	60.1	60.1	59.1	31.7	29.6	35.6	38.6
sidewalk	1	26.0	66.3	36.6	38.6	43.2	38.6	94.4	78.3	43.5	32.3	99.2	38.6	47.9	38.2	72.2	68.2	73.9	40.8	33.9	35.3	41.5
building	2	22.4	24.1	52.1	28.8	28.1	27.4	31.6	30.8	26.0	28.3	34.4	24.8	26.1	23.7	32.2	32.5	31.4	24.3	23.4	27.7	27.0
wall	3	28.1	30.5	43.4	85.2	39.1	43.3	81.0	69.1	40.6	37.1	96.0	41.1	44.8	38.4	64.3	61.0	64.9	40.6	38.1	41.6	43.7
fence	4	27.0	29.5	39.4	34.4	82.8	42.9	80.0	65.2	39.7	33.0	93.2	38.8	41.9	36.3	63.4	61.4	55.0	37.2	33.3	41.5	42.1
pole	5	24.6	27.3	27.6	35.3	36.0	75.3	34.3	36.0	29.0	31.4	45.7	35.8	39.4	30.4	41.8	39.5	38.2	36.2	33.4	30.0	32.5
traffic light	6	8.8	5.1	30.9	16.4	16.9	33.6	97.6	45.7	32.2	8.4	55.1	20.1	7.8	13.8	24.3	24.3	23.4	5.4	4.8	41.4	30.5
traffic sign	7	16.6	16.8	30.4	25.0	26.6	34.5	46.1	97.0	31.5	19.7	65.2	26.5	20.0	19.9	32.7	32.8	30.6	15.0	17.9	40.1	32.6
vegetation	8	18.9	21.1	25.5	25.2	25.0	28.0	32.4	31.5	53.8	22.4	36.1	25.1	24.4	21.8	30.8	28.5	29.0	23.4	22.9	28.7	26.5
terrain	9	27.4	30.4	49.4	46.0	48.3	44.0	89.9	74.2	41.4	79.0	97.6	51.3	55.3	40.7	77.1	74.9	75.6	47.2	42.5	39.9	45.8
sky	10	0.6	0.6	20.9	2.5	4.1	22.4	22.8	22.2	17.6	1.5	60.9	2.4	1.3	1.0	2.2	2.5	2.6	0.6	0.5	23.3	16.0
person	11	24.4	26.5	29.9	37.3	40.0	39.7	74.8	66.0	37.6	32.4	95.1	64.3	42.3	33.5	51.5	46.8	46.3	37.0	34.8	36.8	39.1
rider	12	26.1	30.7	34.9	42.6	44.5	46.2	88.7	75.3	42.3	34.5	97.9	45.0	73.7	38.1	66.9	51.6	61.4	25.6	22.3	44.0	42.3
car	13	23.9	32.8	31.9	41.1	43.4	36.6	72.5	60.4	36.0	35.3	96.8	37.2	41.6	81.9	45.1	47.5	58.7	40.4	38.7	35.3	39.6
truck	14	19.9	17.8	36.7	29.2	29.5	45.4	69.4	58.9	39.1	19.1	93.2	34.8	25.2	27.2	79.5	46.2	66.7	13.9	20.1	43.1	38.9
bus	15	18.6	19.6	37.1	31.6	33.9	41.1	67.8	58.4	37.2	17.0	91.6	37.3	32.3	27.1	44.2	79.1	46.9	27.5	22.2	42.0	38.5
train	16	16.5	16.7	35.5	26.3	26.0	39.3	60.0	54.3	40.2	17.4	85.2	28.3	25.7	24.4	33.3	50.0	76.8	22.7	19.5	39.1	36.6
motorcycle	17	29.5	34.9	40.4	51.3	52.9	49.2	94.6	83.8	50.8	42.0	99.4	48.3	34.5	40.7	81.0	70.0	72.7	84.7	44.0	47.2	48.1
bicycle	18	28.3	30.7	36.0	44.8	45.4	44.4	95.2	80.8	46.6	39.8	99.5	43.3	29.7	39.2	71.9	66.5	73.7	40.8	80.4	41.4	43.9
unknow	19	24.5	28.0	28.0	35.6	36.3	30.1	42.8	41.6	29.6	31.3	47.1	34.0	39.1	30.4	41.1	41.4	38.0	37.5	34.4	75.1	33.2

Abbildung 6-7: Häufigkeitsmatrizen π_{blw} der geschätzten Relationen Below auf dem DCS-Trainingsdatensatz. Die Zahlen sind in Prozent angegeben.

		road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	unknow	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	Marg
road	0	50.7	16.4	14.7	4.7	5.5	10.3	0.3	2.9	19.3	13.6	0.1	13.5	10.6	16.8	7.8	10.8	15.2	6.2	9.3	11.2	13.2
sidewalk	1	32.8	33.7	22.4	10.3	10.9	13.0	0.1	2.5	23.7	16.7	0.1	18.5	14.1	21.7	9.0	10.3	5.0	10.4	14.8	14.3	17.2
building	2	23.6	17.0	47.9	10.7	12.2	19.1	10.7	11.5	25.0	9.4	19.3	19.8	16.2	21.6	11.7	12.9	13.6	14.1	16.8	18.0	18.5
wall	3	30.3	20.6	17.2	14.8	14.5	11.9	0.9	1.5	27.3	8.2	0.6	14.8	9.2	15.5	3.2	6.6	7.0	5.8	11.2	10.5	14.5
fence	4	28.3	16.3	20.3	12.1	17.2	9.2	0.5	2.5	25.8	7.8	1.2	12.9	8.8	14.5	3.5	2.1	15.0	6.0	11.5	8.6	13.5
pole	5	26.2	21.1	25.9	9.5	11.9	24.7	13.6	17.2	27.1	13.6	21.3	14.5	10.5	24.2	8.6	12.5	14.9	8.8	13.3	19.2	19.3
traffic light	6	1.6	0.4	26.7	1.7	2.5	18.5	2.4	4.0	28.1	1.0	18.9	4.9	3.5	13.5	5.3	6.6	13.7	0.0	0.0	9.1	12.8
traffic sign	7	10.8	2.4	27.4	4.4	5.7	12.3	4.1	3.0	28.4	3.8	10.2	6.2	4.4	17.6	6.1	7.7	11.0	0.7	0.7	9.8	12.8
vegetation	8	18.9	11.7	23.6	6.9	9.5	15.9	7.2	8.7	46.2	9.2	18.2	11.6	8.8	17.1	8.3	9.6	11.6	5.1	8.4	14.1	14.9
terrain	9	32.9	20.6	12.9	8.7	10.9	11.0	0.7	2.3	27.0	21.0	0.5	12.4	7.8	16.3	3.4	6.5	4.7	6.3	10.7	12.9	15.1
sky	10	0.4	0.1	25.4	0.9	1.5	10.6	3.3	2.4	28.1	0.4	39.1	1.5	0.6	1.9	1.9	4.6	8.4	0.0	0.0	4.8	11.2
person	11	24.8	16.4	25.5	6.8	8.3	9.9	0.2	1.3	25.7	3.9	1.0	15.7	7.4	17.9	6.4	8.4	13.8	8.2	9.9	12.2	14.2
rider	12	23.4	7.3	22.7	3.5	4.8	4.0	0.0	0.4	24.5	2.3	0.2	5.3	26.3	9.4	3.7	9.7	4.3	18.9	16.8	5.8	11.5
car	13	30.1	7.4	22.8	5.0	5.8	8.8	0.2	2.0	25.2	7.8	0.3	11.4	10.9	18.1	10.9	11.1	6.1	7.1	9.1	9.8	12.9
truck	14	12.2	0.9	19.4	3.2	3.5	4.1	0.9	2.3	21.8	0.4	2.7	7.3	4.3	16.8	20.5	1.9	0.0	2.5	3.2	5.4	9.2
bus	15	10.5	1.9	17.4	0.7	2.6	6.9	1.3	1.1	24.7	1.6	1.4	7.5	6.5	14.3	7.7	20.9	3.1	0.0	4.6	5.2	8.9
train	16	9.3	4.4	19.5	1.8	4.0	7.6	2.9	4.1	19.2	2.3	3.9	11.7	8.6	10.8	0.0	0.0	23.2	0.0	2.5	7.1	9.3
motorcycle	17	32.6	13.8	21.3	2.2	3.9	5.7	0.0	0.5	20.7	4.6	0.0	6.5	20.9	11.8	2.5	2.5	4.5	15.3	7.4	6.6	12.2
bicycle	18	32.7	20.6	23.8	5.9	9.8	8.9	0.0	0.6	22.1	7.0	0.1	12.0	31.2	12.9	4.8	6.7	4.2	7.8	19.6	10.4	15.5
unknow	19	28.7	22.4	26.3	12.4	13.6	20.8	6.6	8.5	27.6	16.0	24.9	17.0	11.1	24.5	10.4	11.4	15.9	8.6	13.9	24.9	19.7

Abbildung 6-8: Häufigkeitsmatrizen π_{ins} der geschätzten Relationen $Inside$ auf dem DCS-Trainingsdatensatz. Die Zahlen sind in Prozent angegeben.

		road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	unknow	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	Marg
road	0	50.7	32.8	23.6	30.3	28.3	26.2	1.6	10.8	18.9	32.9	0.4	24.8	23.4	30.1	12.2	10.5	9.3	32.6	32.7	28.7	25.1
sidewalk	1	16.4	33.7	17.0	20.6	16.3	21.1	0.4	2.4	11.7	20.6	0.1	16.4	7.3	7.4	0.9	1.9	4.4	13.8	20.6	22.4	15.1
building	2	14.7	22.4	47.9	17.2	20.3	25.9	26.7	27.4	23.6	12.9	25.4	25.5	22.7	22.8	19.4	17.4	19.5	21.3	23.8	26.3	23.4
wall	3	4.7	10.3	10.7	14.8	12.1	9.5	1.7	4.4	6.9	8.7	0.9	6.8	3.5	5.0	3.2	0.7	1.8	2.2	5.9	12.4	7.6
fence	4	5.5	10.9	12.2	14.5	17.2	11.9	2.5	5.7	9.5	10.9	1.5	8.3	4.8	5.8	3.5	2.6	4.0	3.9	9.8	13.6	9.0
pole	5	10.3	13.0	19.1	11.9	9.2	24.7	18.5	12.3	15.9	11.0	10.6	9.9	4.0	8.8	4.1	6.9	7.6	5.7	8.9	20.8	13.2
traffic light	6	0.3	0.1	10.7	0.9	0.5	13.6	2.4	4.1	7.2	0.7	3.3	0.2	0.0	0.2	0.9	1.3	2.9	0.0	0.0	6.6	5.2
traffic sign	7	2.9	2.5	11.5	1.5	2.5	17.2	4.0	3.0	8.7	2.3	2.4	1.3	0.4	2.0	2.3	1.1	4.1	0.5	0.6	8.5	6.4
vegetation	8	19.3	23.7	25.0	27.3	25.8	27.1	28.1	28.4	46.2	27.0	28.1	25.7	24.5	25.2	21.8	24.7	19.2	20.7	22.1	27.6	25.4
terrain	9	13.6	16.7	9.4	8.2	7.8	13.6	1.0	3.8	9.2	21.0	0.4	3.9	2.3	7.8	0.4	1.6	2.3	4.6	7.0	16.0	9.9
sky	10	0.1	0.1	19.3	0.6	1.2	21.3	18.9	10.2	18.2	0.5	39.1	1.0	0.2	0.3	2.7	1.4	3.9	0.0	0.1	24.9	13.4
person	11	13.5	18.5	19.8	14.8	12.9	14.5	4.9	6.2	11.6	12.4	1.5	15.7	5.3	11.4	7.3	7.5	11.7	6.5	12.0	17.0	12.9
rider	12	10.6	14.1	16.2	9.2	8.8	10.5	3.5	4.4	8.8	7.8	0.6	7.4	26.3	10.9	4.3	6.5	8.6	20.9	31.2	11.1	12.0
car	13	16.8	21.7	21.6	15.5	14.5	24.2	13.5	17.6	17.1	16.3	1.9	17.9	9.4	18.1	16.8	14.3	10.8	11.8	12.9	24.5	17.5
truck	14	7.8	9.0	11.7	3.2	3.5	8.6	5.3	6.1	8.3	3.4	1.9	6.4	3.7	10.9	20.5	7.7	0.0	2.5	4.8	10.4	7.9
bus	15	10.8	10.3	12.9	6.6	2.1	12.5	6.6	7.7	9.6	6.5	4.6	8.4	9.7	11.1	1.9	20.9	0.0	2.5	6.7	11.4	9.5
train	16	15.2	5.0	13.6	7.0	15.0	14.9	13.7	11.0	11.6	4.7	8.4	13.8	4.3	6.1	0.0	3.1	23.2	4.5	4.2	15.9	11.5
motorcycle	17	6.2	10.4	14.1	5.8	6.0	8.8	0.0	0.7	5.1	6.3	0.0	8.2	18.9	7.1	2.5	0.0	0.0	15.3	7.8	8.6	7.8
bicycle	18	9.3	14.8	16.8	11.2	11.5	13.3	0.0	0.7	8.4	10.7	0.0	9.9	16.8	9.1	3.2	4.6	2.5	7.4	19.6	13.9	10.8
unknow	19	11.2	14.3	18.0	10.5	8.6	19.2	9.1	9.8	14.1	12.9	4.8	12.2	5.8	9.8	5.4	5.2	7.1	6.6	10.4	24.9	12.3

Abbildung 6-9: Häufigkeitsmatrize π_{ard} der geschätzten Relationen $Around$ auf dem DCS-Trainingsdatensatz. Die Zahlen sind in Prozent angegeben.

6.4.2.1.2. Schätzung der räumlichen Relationen $Above$, $Below$, $SupportedBy$ und $Support$

Für die Schätzung der Relationen $Support$ und $SupportedBy$ wurde die Teilmenge $\mathcal{R}_{s1}$ durch $\mathcal{R}_{s2}$ ersetzt, wobei $SupportedBy$ die Inverse der Relation $Support$ war ($Support \equiv SupportedBy^{-}$).

$$\mathcal{R}_{s2} = \{Above, Below, SupportedBy, Support\} \tag{42}$$

Dabei wurden die Relationen $Inside$ und $Around$ durch $SupportedBy$ und $Support$ ersetzt. Der Grund dafür war, dass die Relationen $SupportedBy$ und $Support$ ähnlich wie bei $Inside$ und $Around$ häufig benachbarte Regionen erfassten. Die Integration dieser vier Relationen in einer Menge würde redundante Informationen beinhalten.

Ähnlich wie bei der Schätzung der Relationen aus $\mathcal{R}_{s1}$ wurde auch hier der Merkmalvektor $F_s(o_i, o_j)$ verwendet:

$$F_s(o_i, o_j) = \left(\beta^b(o_i, o_j), BoW(o_i, o_j), BoW(o_j, o_i)\right)^T \tag{43}$$

$$\beta^b(o_i, o_j) = \frac{\beta_y^b(o_i) - \beta_y^b(o_j)}{Height(I^s)} \quad (44)$$

$\beta^b(o_i, o_j) \in [-1, +1]$ ist die mit der Höhe des Bildes normierte Differenz zwischen den Y-Koordinaten der unteren Kanten der *Bounding*-Boxen der Regionen o_i und o_j.

$$BoW(o_i, o_j) = \frac{|Border(o_i \,/\, o_j)|}{W(o_i)} \quad (45)$$

$BoW(o_i, o_j) \in [0,1]$ (*BoW* : *Border over Width*) ist die Länge der Grenze des unteren Bereichs der Region o_i mit allen Regionen im Bild außer o_j, normiert mit der Breite der Region o_i. Die Funktion BoW ist antisymmetrisch ($BoW(o_i, o_j) \neq BoW(o_j, o_i)$), da die Breiten der Regionen o_i und o_j unterschiedlich sein könnten ($W(o_i) \neq W(o_j)$) und die *Border*-Funktion antisymmetrisch ist ($Border(o_i \,/\, o_j) \neq Border(o_j \,/\, o_i)$).

Analog zu den Clusterzentren der Relationen aus $\mathcal{R}_{s1}$ wurden auch hier vier Clusterzentren gelernt. Die Tabelle 6-4 stellt die gelernten Clusterzentren dar. Diese Cluster zeigten ähnliche Merkmale wie die Cluster der Tabelle 6-3. Die Clusterzentren der Relationen $Above$ und $Below$ unterschieden sich hauptsächlich durch die normierte Differenz $\beta^b(o_i, o_j)$ zwischen den Y-Koordinaten der unteren Kanten der Regionen o_i und o_j. Im Gegenteil waren für die Relationen $SupportedBy$ und $Support$ die Merkmale $BoW(o_i, o_j)$ und $BoW(o_j, o_i)$ am wichtigsten.

Tabelle 6-4: Gelernte Clusterzentren der räumlichen Relationen aus der Menge $\mathcal{R}_{s2} = \{Above, Below, SupportedBy, Support\}$ auf dem DCS-Trainingsdatensatz

Relation	Clustername	$\boldsymbol{\mu(o_i, o_j)}$	$\boldsymbol{BoW(o_i, o_j)}$	$\boldsymbol{BoW(o_j, o_i)}$
Above	$Cluster_{abv}$	-0.25032	0.998877	0.998592
Below	$Cluster_{blw}$	0.249298	0.999744	0.990639
SupportedBy	$Cluster_{spb}$	-0.0693009	0.0988628	1
Support	$Cluster_{spp}$	-0.000141789	0.999225	0.0978628

Die Häufigkeitsmatrizen der geschätzten Relationen $Above$ (siehe π_{abv}^s in Abbildung 6-10), $Below$ (siehe π_{blw}^s in Abbildung 6-11), $SupportedBy$ (siehe π_{spb}^s in Abbildung 6-12) und $Support$ (siehe π_{spp}^s in Abbildung 6-13) wurden analog zu den Häufigkeitsmatrizen aus dem Abschnitt 6.4.2.1.1 mit dem *DCS*-Trainingsdatensatz generiert. Die Matrizen π_{abv}^s und π_{blw}^s zeigten ähnliche Verhalten wie π_{abv} (siehe Abbildung 6-6) und π_{blw} (siehe Abbildung 6-7). Anders als die Matrize π_{ard} der Abbildung 6-9 erfasste die Matrize π_{spp}^s die Relation *ist-auf*. So bestätigte der Häufigkeitswert $\pi_{spp}^s(ind(sidewalk), ind(person)) = 24{,}2\ \%$ die Erwartung, dass „*Fußgänger oft auf dem Fußgängerweg gehen*“. Die Erwartung, dass „*Fahrzeuge oft auf der Straße fahren*“, wurde durch den Häufigkeitswert $\pi_{spp}^s(ind(road), ind(car)) = 32{,}3\ \%$ bekräftigt.

		road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	unknow	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	Marg
road	0	100.0	25.5	34.5	35.4	35.8	36.1	4.6	16.1	33.3	29.5	0.5	31.5	36.7	27.4	30.6	31.2	30.0	34.1	34.1	28.5	29.8
sidewalk	1	27.7	100.0	30.5	30.0	32.3	30.7	2.8	15.3	32.1	29.7	0.4	29.5	40.4	38.4	35.4	35.9	31.0	35.4	31.6	28.8	30.3
building	2	53.5	41.0	100.0	48.4	43.7	34.4	31.7	30.9	27.6	51.3	28.9	37.9	47.6	38.0	47.5	46.7	42.9	49.8	42.3	29.2	37.4
wall	3	55.0	38.1	34.7	100.0	35.5	41.0	11.0	20.9	33.8	44.4	2.9	41.8	49.4	43.7	41.8	41.3	43.1	52.4	46.9	36.6	38.1
fence	4	55.4	43.0	33.7	39.6	100.0	40.0	11.9	22.4	33.3	47.2	3.7	41.7	50.8	44.3	42.5	46.8	35.4	52.9	47.0	36.2	38.7
pole	5	48.5	36.4	34.5	46.4	44.9	100.0	33.6	33.7	35.1	40.2	32.1	43.1	47.8	37.4	49.2	46.9	44.1	49.8	44.3	34.4	39.1
traffic light	6	95.1	96.8	33.7	86.6	85.1	35.4	100.0	46.4	37.7	92.9	31.1	84.0	95.3	84.0	83.4	80.8	70.4	96.4	97.1	44.1	55.4
traffic sign	7	77.7	80.4	33.2	72.9	71.8	37.0	46.3	100.0	37.8	77.2	31.5	72.2	81.1	69.6	70.2	67.3	66.2	87.1	84.6	43.0	52.3
vegetation	8	55.0	46.9	27.5	44.8	43.2	35.0	32.1	32.5	100.0	44.5	21.7	46.2	55.1	41.1	50.7	45.7	44.8	57.6	53.5	31.5	38.8
terrain	9	36.7	32.3	33.6	37.4	35.4	34.3	5.2	17.0	33.4	100.0	1.1	39.7	44.0	40.7	32.6	35.5	25.9	44.5	41.7	31.4	33.6
sky	10	99.4	99.4	41.4	96.5	95.1	56.4	57.9	65.0	42.9	98.2	100.0	96.9	99.0	98.2	96.4	94.5	88.7	99.6	99.5	47.7	66.5
person	11	43.9	36.2	33.1	43.7	41.8	40.0	12.3	22.9	38.0	50.9	2.3	100.0	49.0	40.1	45.0	46.7	42.1	50.5	45.5	36.8	37.7
rider	12	51.2	50.9	34.2	45.0	40.8	40.2	2.8	15.6	37.3	52.1	0.7	44.0	100.0	45.0	37.9	42.4	41.5	33.5	34.4	40.4	38.2
car	13	31.5	41.1	30.3	39.7	38.5	32.7	7.6	18.0	31.0	41.4	0.7	35.6	42.9	100.0	35.3	36.6	34.9	43.8	42.2	31.5	33.3
truck	14	44.4	60.9	37.7	53.6	48.5	42.1	11.8	24.0	37.7	63.2	1.4	47.5	57.5	41.6	100.0	49.0	30.0	65.9	59.7	40.2	41.3
bus	15	40.7	58.2	35.3	55.4	48.4	40.5	10.8	24.4	35.2	60.4	2.2	44.0	51.4	44.9	42.9	100.0	41.7	59.2	55.6	40.3	40.3
train	16	47.0	64.5	34.3	55.2	50.5	40.6	15.8	24.9	31.6	68.2	4.7	47.8	53.8	51.6	70.0	58.3	100.0	51.9	61.8	36.4	41.3
motorcycle	17	43.4	41.5	33.1	42.9	39.9	41.3	3.6	12.6	36.6	48.2	0.4	39.2	28.0	44.4	31.8	40.8	44.4	100.0	42.3	38.3	36.4
bicycle	18	43.5	35.5	32.8	40.4	37.8	39.1	2.9	14.4	36.0	43.5	0.5	39.6	27.5	42.6	35.9	38.3	35.8	46.5	100.0	36.6	35.3
unknow	19	40.8	35.9	29.4	45.6	44.2	34.8	42.6	41.2	32.3	40.2	31.3	41.2	50.6	37.8	49.5	48.3	41.5	51.1	44.8	100.0	38.5

Abbildung 6-10: Häufigkeitsmatrize π^{S}_{abv} der geschätzten Relationen Above auf dem DCS-Trainingsdatensatz. Die Zahlen sind in Prozent angegeben.

		road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	unknow	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	Marg
road	0	100.0	27.7	53.5	55.0	55.4	48.5	95.1	77.7	55.0	36.7	99.4	43.9	51.2	31.5	44.4	40.7	47.0	43.4	43.5	40.8	47.9
sidewalk	1	25.5	100.0	41.0	38.1	43.0	36.4	96.8	80.4	46.9	32.3	99.4	36.2	50.9	41.1	60.9	58.2	64.5	41.5	35.5	35.9	43.5
building	2	34.5	30.5	100.0	34.7	33.7	34.5	33.7	33.2	27.5	33.6	41.4	33.1	34.2	30.3	37.7	35.3	34.3	33.1	32.8	29.4	33.7
wall	3	35.4	30.0	48.4	100.0	39.6	46.4	86.6	72.9	44.8	37.4	96.5	43.7	45.0	39.7	53.6	55.4	55.2	42.9	40.4	45.6	47.0
fence	4	35.8	32.3	43.7	35.5	100.0	44.9	85.1	71.8	43.2	35.4	95.1	41.8	40.8	38.5	48.5	48.4	50.5	39.9	37.8	44.2	45.9
pole	5	36.1	30.7	34.4	41.0	40.0	100.0	35.4	37.0	35.0	34.3	56.4	40.0	40.2	32.7	42.1	40.5	40.6	41.3	39.1	34.8	37.7
traffic light	6	4.6	2.8	31.7	11.0	11.9	33.6	100.0	46.3	32.1	5.2	57.9	12.3	2.8	7.6	11.8	10.8	15.8	3.6	2.9	42.6	29.6
traffic sign	7	16.1	15.3	30.9	20.9	22.4	33.7	46.4	100.0	32.5	17.0	65.0	22.9	15.6	18.0	24.0	24.4	24.9	12.6	14.4	41.2	32.2
vegetation	8	33.3	32.1	27.6	33.8	33.3	35.1	37.7	37.8	100.0	33.4	42.9	38.0	37.3	31.0	37.7	35.2	31.6	36.6	36.0	32.3	35.3
terrain	9	29.5	29.7	51.3	44.4	47.2	40.2	92.9	77.2	44.5	100.0	98.2	50.9	52.1	41.4	63.2	60.4	68.2	48.2	43.5	40.2	46.9
sky	10	0.5	0.4	28.9	2.9	3.7	32.1	31.1	31.5	21.7	1.1	100.0	2.3	0.7	0.7	1.4	2.2	4.7	0.4	0.5	31.3	21.0
person	11	31.5	29.5	37.9	41.8	41.7	43.1	84.0	72.2	46.2	39.7	96.9	100.0	44.0	35.6	47.5	44.0	47.8	39.2	39.6	41.2	44.3
rider	12	36.7	40.4	47.6	49.4	50.8	47.8	95.3	81.1	55.1	44.0	99.0	49.0	100.0	42.9	57.5	51.4	53.8	28.0	27.5	50.6	49.9
car	13	27.4	38.4	38.0	43.7	44.3	37.4	84.0	69.6	41.1	40.7	98.2	40.1	45.0	100.0	41.6	44.9	51.6	44.4	42.6	37.8	43.9
truck	14	30.6	35.4	47.5	41.8	42.5	49.2	83.4	70.2	50.7	32.6	96.4	45.0	37.9	35.3	100.0	42.9	70.0	31.8	35.9	49.5	48.1
bus	15	31.2	35.9	46.7	41.3	46.8	46.9	80.8	67.3	45.7	35.5	94.5	46.7	42.4	36.6	49.0	100.0	58.3	40.8	38.3	48.3	47.7
train	16	30.0	31.0	42.9	43.1	35.4	44.1	70.4	66.2	44.8	25.9	88.7	42.1	41.5	34.9	30.0	41.7	100.0	44.4	35.8	41.5	44.8
motorcycle	17	34.1	35.4	49.8	52.4	52.9	49.8	96.4	87.1	57.6	44.5	99.6	50.5	33.5	43.8	65.9	59.2	51.9	100.0	46.5	51.1	51.5
bicycle	18	34.1	31.6	42.3	46.9	47.0	44.3	97.1	84.6	53.5	41.7	99.5	45.5	34.4	42.2	59.7	55.6	61.8	42.3	100.0	44.8	47.7
unknow	19	28.5	28.8	29.2	36.6	36.2	34.4	44.1	43.0	31.5	31.4	47.7	36.8	40.4	31.5	40.2	40.3	36.4	38.3	36.6	100.0	35.7

Abbildung 6-11: Häufigkeitsmatrize π^{s}_{blw} der geschätzten Relationen Below auf dem DCS-Trainingsdatensatz. Die Zahlen sind in Prozent angegeben.

		road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	unknow	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	Marg
road	0	0.0	16.5	0.7	2.8	1.9	1.8	0.1	1.5	4.4	14.2	0.0	8.0	4.5	8.8	1.4	1.1	1.5	3.8	6.1	8.7	6.2
sidewalk	1	30.3	0.0	0.9	2.0	2.0	3.2	0.0	1.0	5.6	15.7	0.0	10.1	3.9	7.5	0.4	0.7	0.0	4.9	8.6	9.4	8.0
building	2	11.3	27.6	0.0	15.5	18.8	12.0	9.7	9.8	24.2	13.9	2.3	18.4	15.0	29.6	14.1	16.1	19.8	12.9	17.1	16.5	15.6
wall	3	6.7	29.8	1.4	0.0	6.4	2.8	0.6	0.9	6.6	10.4	0.0	9.0	4.2	12.5	0.7	3.3	0.0	4.3	10.4	6.8	7.7
fence	4	6.9	22.7	3.8	18.5	0.0	1.7	0.5	1.4	8.5	13.6	0.0	8.0	5.0	11.8	2.1	0.0	0.0	5.3	10.1	7.1	7.6
pole	5	13.5	29.7	19.0	9.8	13.4	0.0	18.1	21.9	21.3	23.5	7.1	14.7	11.5	28.2	7.7	11.3	14.1	7.8	13.9	20.2	17.9
traffic light	6	0.2	0.4	24.9	1.8	2.5	12.9	0.0	3.8	23.9	1.4	9.0	3.5	2.0	8.2	4.2	7.7	13.2	0.0	0.0	7.5	9.9
traffic sign	7	4.7	3.2	26.1	5.3	4.5	7.4	3.5	0.0	22.8	4.9	2.1	3.7	2.9	11.2	4.1	6.5	8.5	0.2	0.7	8.7	9.6
vegetation	8	7.2	15.4	20.7	14.8	15.0	8.6	6.2	7.0	0.0	17.3	5.8	10.3	6.6	21.0	9.7	15.8	19.1	3.4	7.1	13.1	11.5
terrain	9	19.5	22.2	1.2	7.7	3.8	2.0	0.5	0.9	4.8	0.0	0.1	4.0	2.4	5.3	0.3	1.9	0.0	2.7	4.4	8.5	7.0
sky	10	0.2	0.2	27.3	0.5	1.2	4.5	2.1	1.4	29.6	0.6	0.0	0.8	0.3	1.1	2.2	3.3	6.7	0.0	0.0	3.6	8.3
person	11	16.6	24.2	10.6	5.6	8.5	2.3	0.1	1.1	5.5	5.4	0.0	0.0	4.9	17.1	2.0	2.2	0.5	8.7	11.1	10.3	9.3
rider	12	7.5	4.7	3.2	1.4	3.4	0.6	0.0	0.4	1.1	1.5	0.0	2.1	0.0	5.1	0.7	1.1	0.0	30.1	29.9	2.7	5.6
car	13	32.3	13.1	2.2	4.1	5.4	1.7	0.2	1.1	6.9	12.6	0.0	7.2	7.0	0.0	1.3	0.8	0.0	6.7	9.6	9.2	8.2
truck	14	23.6	3.2	0.7	3.9	6.9	1.1	0.7	1.7	1.9	3.8	0.0	5.5	3.9	21.7	0.0	4.1	0.0	2.4	2.8	2.9	6.0
bus	15	27.0	5.2	1.9	0.0	4.8	1.2	0.8	1.8	3.3	2.3	0.0	7.2	5.1	17.8	4.1	0.0	0.0	0.0	4.4	3.0	6.0
train	16	21.5	4.5	3.1	1.7	14.1	1.3	0.7	0.5	4.5	5.9	0.0	9.6	4.6	13.5	0.0	0.0	0.0	0.0	1.6	10.0	6.1
motorcycle	17	18.7	18.2	4.2	0.5	1.9	1.2	0.0	0.2	2.3	4.6	0.0	1.7	8.5	5.1	0.0	0.0	3.7	0.0	5.9	4.0	5.7
bicycle	18	16.3	24.3	7.8	2.3	5.1	2.6	0.0	0.3	3.5	10.3	0.0	3.8	8.2	5.6	1.6	1.6	0.8	5.4	0.0	6.7	7.4
unknow	19	22.0	25.9	24.9	11.0	12.6	10.6	5.8	7.2	23.1	19.8	17.4	11.8	6.3	21.5	7.4	8.3	12.1	6.6	11.9	0.0	16.0

Abbildung 6-12: Häufigkeitsmatrize π^{s}_{spb} der geschätzten Relationen $SupportedBy$ auf dem DCS-Trainingsdatensatz. Die Zahlen sind in Prozent angegeben.

		road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	unknow	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	Marg
road	0	0.0	30.3	11.3	6.7	6.9	13.5	0.2	4.7	7.2	19.5	0.2	16.6	7.5	32.3	23.6	27.0	21.5	18.7	16.3	22.0	16.0
sidewalk	1	16.5	0.0	27.6	29.8	22.7	29.7	0.4	3.2	15.4	22.2	0.2	24.2	4.7	13.1	3.2	5.2	4.5	18.2	24.3	25.9	18.3
building	2	0.7	0.9	0.0	1.4	3.8	19.0	24.9	26.1	20.7	1.2	27.3	10.6	3.2	2.2	0.7	1.9	3.1	4.2	7.8	24.9	13.4
wall	3	2.8	2.0	15.5	0.0	18.5	9.8	1.8	5.3	14.8	7.7	0.5	5.6	1.4	4.1	3.9	0.0	1.7	0.5	2.3	11.0	7.2
fence	4	1.9	2.0	18.8	6.4	0.0	13.4	2.5	4.5	15.0	3.8	1.2	8.5	3.4	5.4	6.9	4.8	14.1	1.9	5.1	12.6	7.7
pole	5	1.8	3.2	12.0	2.8	1.7	0.0	12.9	7.4	8.6	2.0	4.5	2.3	0.6	1.7	1.1	1.2	1.3	1.2	2.6	10.6	5.4
traffic light	6	0.1	0.0	9.7	0.6	0.5	18.1	0.0	3.5	6.2	0.5	2.1	0.1	0.0	0.2	0.7	0.8	0.7	0.0	0.0	5.8	5.2
traffic sign	7	1.5	1.0	9.8	0.9	1.4	21.9	3.8	0.0	7.0	0.9	1.4	1.1	0.4	1.1	1.7	1.8	0.5	0.2	0.3	7.2	5.9
vegetation	8	4.4	5.6	24.2	6.6	8.5	21.3	23.9	22.8	0.0	4.8	29.6	5.5	1.1	6.9	1.9	3.3	4.5	2.3	3.5	23.1	14.4
terrain	9	14.2	15.7	13.9	10.4	13.6	23.5	1.4	4.9	17.3	0.0	0.6	5.4	1.5	12.6	3.8	2.3	5.9	4.6	10.3	19.8	12.5
sky	10	0.0	0.0	2.3	0.0	0.0	7.1	9.0	2.1	5.8	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	17.4	4.2
person	11	8.0	10.1	18.4	9.0	8.0	14.7	3.5	3.7	10.3	4.0	0.8	0.0	2.1	7.2	5.5	7.2	9.6	1.7	3.8	11.8	8.6
rider	12	4.5	3.9	15.0	4.2	5.0	11.5	2.0	2.9	6.6	2.4	0.3	4.9	0.0	7.0	3.9	5.1	4.6	8.5	8.2	6.3	6.3
car	13	8.8	7.5	29.6	12.5	11.8	28.2	8.2	11.2	21.0	5.3	1.1	17.1	5.1	0.0	21.7	17.8	13.5	5.1	5.6	21.5	14.6
truck	14	1.4	0.4	14.1	0.7	2.1	7.7	4.2	4.1	9.7	0.3	2.2	2.0	0.7	1.3	0.0	4.1	0.0	0.0	1.6	7.4	4.6
bus	15	1.1	0.7	16.1	3.3	0.0	11.3	7.7	6.5	15.8	1.9	3.3	2.2	1.1	0.8	4.1	0.0	0.0	0.0	1.6	8.3	6.0
train	16	1.5	0.0	19.8	0.0	0.0	14.1	13.2	8.5	19.1	0.0	6.7	0.5	0.0	0.0	0.0	0.0	0.0	3.7	0.8	12.1	7.7
motorcycle	17	3.8	4.9	12.9	4.3	5.3	7.8	0.0	0.2	3.4	2.7	0.0	6.7	30.1	6.7	2.4	0.0	0.0	0.0	5.4	6.6	6.4
bicycle	18	6.1	8.6	17.1	10.4	10.1	13.9	0.0	0.7	7.1	4.4	0.0	11.1	29.9	9.6	2.8	4.4	1.6	5.9	0.0	11.9	9.6
unknow	19	8.7	9.4	16.5	6.8	7.1	20.2	7.5	6.7	13.1	8.5	3.6	10.3	2.7	9.2	2.9	3.0	10.0	4.0	6.7	0.0	9.8

Abbildung 6-13: Häufigkeitsmatrize π^{s}_{spp} der geschätzten Relationen $Support$ auf dem DCS-Trainingsdatensatz. Die Zahlen sind in Prozent angegeben.

6.4.2.2 Generierung von Trainingsdaten

Zur Generierung von Trainingsdaten musste für gegebene Regionen o_i und o_j, ihre semantischen Klassen $c_i(o_i), c_i \in \mathcal{C}$ und $c_j(o_j), c_j \in \mathcal{C}$ sowie die räumliche Relation $r(o_i, o_j), r \in \mathcal{R}_s$ die Konsistenz dieser Regionen, das heißt die Wahrheitswerte der Prädikate $ObjConsistence(o_i)$ und $ObjConsistence(o_j)$, geschätzt werden. Die Wahrscheinlichkeit, dass zwei Regionen konsistent waren, war hoch, wenn die räumliche Relation dieser Regionen eine war, die häufig in den Trainingsdaten vorkam, wobei π_r eine der Häufigkeitsmatrizen war, die im Abschnitt 6.4.2.1 beschrieben wurde.

$$P\left(ObjConsistence(o_i), ObjConsistence(o_j) \middle| c_i(o_i), c_j(o_j), r(o_i, o_j)\right) \propto min\left(1, \frac{1}{\kappa} * \pi_r\left(ind(c_i), ind(c_j)\right)\right) \quad (46)$$

$\kappa \in]0,1]$ war ein Skalierungsfaktor, der empirisch geschätzt wurde. $min(\quad)$ war die Minimumfunktion. $ObjConsistence(o_i)$ und $ObjConsistence(o_j)$ hatten den Wahrheitswert 1, wenn die Wahrscheinlichkeit aus der Gleichung (46) größer als der Schwellenwert τ war. In dieser Arbeit wurde $\tau = 0{,}5$ verwendet.

Für alle Regionspaare (o_i, o_j) des *DCS*-Trainingsdatensatzes wurden dann die Wahrheitswerte der Prädikate $c_i(o_i), c_i \in \mathcal{C}$, $c_j(o_j), c_j \in \mathcal{C}$, $r(o_i, o_j), r \in \mathcal{R}_s$, $ObjConsistence(o_i)$ und $ObjConsistence(o_j)$ generiert und als *.db*-Datei für das Training des MLN-Modells gespeichert. Ein Nachteil bei diesem Verfahren war, dass Relationen, die selten in den Trainingsdaten vorkamen, auch wenige Trainingsdaten für das MLN-Modell besaßen. Ausgerechnet diese Relationen deuteten auf Inkonsistenzen hin. Dies führte dazu, dass die Prädikate $ObjConsistence(o_i)$ und $ObjConsistence(o_j)$ meistens mit dem Wahrheitswert *wahr* generiert wurden. Die Prädikate $ObjConsistence(o_i)$ und $ObjConsistence(o_j)$ mit dem Wahrheitswert *falsch* waren wenig in den Trainingsdaten vorhanden. Relationen $r(o_i, o_j), r \in \mathcal{R}_s$, die selten vorkamen, führten dazu, dass die entsprechenden logischen Regeln des MLN-Modells selten erfüllt waren. Da diese logischen Regeln aber auch nur selten verletzt wurden, wurde die Gewichtung dieser Regeln bei 0 geschätzt. Die Schätzung der Wahrscheinlichkeit der Konsistenz für seltene Relationen lag dann bei ca. 0.5, was die Erwartungen nicht erfüllte. Seltene Relationen sollten eine Konsistenzwahrscheinlich haben, die gegen 0 geht. Um dieses Problem zu lösen, wurden für alle semantischen Klassenpaare $c_i \in \mathcal{C}$, $c_j \in \mathcal{C}$ und alle räumlichen Relationen $r \in \mathcal{R}_s$,k-Prädikate $c_i(o_{i\_k})$, $c_j(o_{j\_k})$, $r(o_{i\_k}, o_{j\_k}), r \in \mathcal{R}_s$, $ObjConsistence(o_{i\_k})$ und $ObjConsistence(o_{j\_k})$ generiert. Hier hatten die Prädikate $ObjConsistence(o_{i\_k})$ und $ObjConsistence(o_{j\_k})$ den Wahrheitswert *wahr*. Die Anzahl der Prädikate k wurden durch die folgende Gleichung bestimmt:

$$\#Konsistente\_Prädikate = min\left(1, \frac{1}{\kappa} * \pi_r\left(ind(c_i), ind(c_j)\right)\right) * K \qquad (47)$$

Wobei $\mathrm{K} \in \mathbb{N}$ eine Konstante war, die die maximale Anzahl der Prädikate festlegte. Je größer K war, desto länger dauerte die Trainingsphase. In dieser Arbeit wurde K empirisch auf 15 gesetzt. Darüber hinaus wurden m-Prädikate $c_i(o_{i\_m})$, $c_j(o_{j\_m})$, $r(o_{i\_m}, o_{j\_m}), r \in \mathcal{R}_s$, $!\,ObjConsistence(o_{i\_m})$ und $!\,ObjConsistence(o_{j\_m})$ generiert, wobei die Prädikate $!\,ObjConsistence(o_{i\_m})$ und $!\,ObjConsistence(o_{j\_m})$ den Wahrheitswert *falsch* hatten. Die Anzahl der Prädikate wurde hier durch die Gleichung (48) festgelegt.

$$\#Inkonsistente\_Prädikate = \mathrm{K} - \#Konsistente\_Prädikate \qquad (48)$$

Die Häufigkeitsmatrix π_r in der Gleichung (47) ging gegen 1, wenn die Relation r zwischen Regionspaaren häufig in den Trainingsdaten vorkam. So ging die Anzahl der Prädikate aus der Gleichung (47), in der $ObjConsistence$ den Warheitswert *wahr* hatte, gegen das Maximum K. Gleichzeitig ging die Anzahl der Prädikate aus der Gleichung (48), in der $ObjConsistence$ den Wahrheitswert *falsch* hatte, gegen 0. Dies führte dazu, dass die entsprechenden logischen Regeln des MLN-Models oft erfüllt wurden und somit positiv gewichtet wurden. Die Inferenz dieser Regeln würde mit einer hohen Wahrscheinlichkeit den Wahrheitswert *wahr* für das Prädikat $ObjConsistence$ schätzen. Im Gengensatz dazu ging bei Relationen, die selten zwischen Regionspaaren in den Trainingsdaten vorkamen, der Wert von π_r gegen 0. So ging die Anzahl der Prädikate aus der Gleichung (47), in der $ObjConsistence$ den Warheitswert *wahr* hatte, gegen 0, während die Anzahl der Prädikate aus der Gleichung (48), in der $ObjConsistence$ den Wahrheitswert *falsch* hatte, sich dem Maximum K annäherte. Dies führte dazu, dass die entsprechenden logischen Regeln des MLN-Modells oft nicht erfüllt wurden und somit negativ gewichtet wurden. Damit war die Konsistenzwahrscheinlichkeit von Regionspaaren mit solchen seltenen Relationen sehr gering.

6.4.2.3 Vereinfachung der logischen Formeln des MLN-Modells

Zur Vereinfachung der Inferenzkomplexität wurden die logischen Formeln des MLN-Modells aus der Abbildung 6-5 wie folgt angepasst:

1. Die Inversen der räumlichen Relationen wurden entfernt. Die Menge $\mathcal{R}_s$ aus der Gleichung (35) durch $\hat{\mathcal{R}}_s = \{Support, Above, Inside\}$ ersetzt. Damit wurde die Anzahl der Prädikate reduziert (siehe Zeilen fünf bis zehn der Abbildung 6-14). Die Axiome der inversen Relationen

waren nicht mehr notwendig und wurden entfernt. Die Axiome der disjunkten Relationen blieben unverändert (siehe Zeilen 13 bis 15 der Abbildung 6-14).

2. Die semantischen Klassen aus der Menge $\mathcal{C}$ (siehe Gleichung (31)) wurden in der Konstantenmenge $className$ definiert (siehe Zeilen drei und vier der Abbildung 6-14). Die ersten Buchstaben der Klassennamen wurden gemäß der Syntax von MLN-Konstanten groß geschrieben. Die mit den semantischen Klassen verbundenen Prädikate wurden durch das Prädikat $ObjClass$ ersetzt und die Menge $className$ wurde als Wertbereich des Prädikats $ObjClass$ eingegeben (siehe Zeile zehn der Abbildung 6-14).
3. Das Prädikat $ObjClass$ wurde für die kompakte Modellierung der logischen Regeln in den Zeilen 19 bis 23 der Abbildung 6-14 verwendet. Die erste Variable dieses Prädikats wurde mit „+“-Zeichen versehen. Damit wurde festgelegt, dass während des Trainings des MLN-Modells separate Formeln für jedes Element aus dem Wertbereich von $ObjClass$ generiert werden.

```
2  //Constant
3  className = {Unknow,Road,Sidewalk,Building,Wall,Fence,Pole,Traffic_light,Traffic_sign,
4               Vegetation,Terrain,Sky,Person,Rider,Car,Truck,Bus,Train,Motorcycle,Bicycle}
5  //Predicate
6  Above(scene_element,scene_element)
7  Inside(scene_element,scene_element)
8  Support(scene_element,scene_element)
9  ObjConsistence(scene_element)
10 ObjClass(className!, scene_element)
11
12 //Axioms: disjoint
13 (Above(o1,o2) ^ !Inside(o1,o2) ^ !Support(o1,o2)) v
14 (!Above(o1,o2) ^ Inside(o1,o2) ^ !Support(o1,o2)) v
15 (!Above(o1,o2) ^ !Below(o1,o2) ^ Support(o1,o2)).
16
17 //Rules
18 //{S1}
19 ObjClass(+c1, o_1) ^ ObjClass(+c2, o_2) ^ Above(o_1,o_2) => ObjConsistence(o_1) ^ ObjConsistence(o_2)
20 //{S2}
21 ObjClass(+c1, o_1) ^ ObjClass(+c2, o_2) ^ Inside(o_1,o_2) => ObjConsistence(o_1) ^ ObjConsistence(o_2)
22 //{S3}
23 ObjClass(+c1, o_1) ^ ObjClass(+c2, o_2) ^ Support(o_1,o_2) => ObjConsistence(o_1) ^ ObjConsistence(o_2)
```

Abbildung 6-14: Vereinfachung der logischen Formeln des MLN-Modells aus der Abbildung 6-5

Die o. g. Änderungen der logischen Formeln des MLN-Modells führten zur Anpassung der Prädikate $c_i(o_i), c_i \in \mathcal{C}$, $c_j(o_j), c_j \in \mathcal{C}$ der Trainingsdaten aus dem Abschnitt 6.4.2.2. Diese Prädikate wurden durch $ObjClass(c_i, o_i), c_i \in \mathcal{C}$ und $ObjClass(c_j, o_j), c_j \in \mathcal{C}$ ersetzt. Die Tabelle 6-5 stellt einen Ausschnitt der generierten Trainingssamples für die Klassen *sky* und *road* und die räumlichen Relationen $Above$ und $Support$ dar. Diese Trainingssamples wurden mithilfe der Gleichungen (47) und (48) generiert, wobei die Parameter $K = 15$ und $\kappa = 1$ verwendet wurden. Die Häufigkeit der Relation $Above(o_i, o_j)$ zwischen Instanzen o_i der Klasse *sky* und o_j der Klasse *road* $\pi_{abv}(ind(sky), ind(road))$ betrug 99,4 % (siehe Abbildung 6-10). Dies führte zu 15 Grundprädikaten, bei denen der Wahrheitswert des Grundprädikats $ObjConsistence$ *wahr* war (siehe Gleichungen (47)) und 0 Grundprädikaten, für die der Wahrheitswert von $ObjConsistence$ *falsch* war (siehe Gleichung (48)). Diese Prädikate sind in der zweiten Zeile der Tabelle 6-5 zu finden. Für die Klassen *sky* und *road* waren die Häufigkeitswerte für die Relationen $Above(o_j, o_i)$, $Support(o_i, o_j)$ und $Support(o_j, o_i)$ kleiner 1 %. Somit gab es keine Trainingssamples, in denen der Wahrheitswert des Prädikats $ObjConsistence$ *wahr* war für diese Relationen. Dafür gab es für diese Relationen jeweils 15 Prädikate mit dem Wahrheitswert *falsch* für das Prädikat $ObjConsistence$. Diese Prädikate befinden sich in der dritten Zeile der Tabelle 6-5.

Tabelle 6-5: Beispielhafter Ausschnitt der generierten Trainingssamples für die Klassen sky und road und die räumlichen Relationen Above und Support

Grundprädikate	Grundprädikate	Grundprädikate	Grundprädikate	Grundprädikate

der Klasse $road$	der Klasse sky	der Relationen $Above$ und $Support$	der Konsistenz für die Klasse $road$	der Konsistenz für die Klasse sky
ObjClass(Road,O7_0)	ObjClass(Sky,O23_0)	Above(O23_0,O7_0)	ObjConsistence(O7_0)	ObjConsistence(O23_0)
ObjClass(Road,O7_1)	ObjClass(Sky,O23_1)	Above(O23_1,O7_1)	ObjConsistence(O7_1)	ObjConsistence(O23_1)
ObjClass(Road,O7_2)	ObjClass(Sky,O23_2)	Above(O23_2,O7_2)	ObjConsistence(O7_2)	ObjConsistence(O23_2)
ObjClass(Road,O7_3)	ObjClass(Sky,O23_3)	Above(O23_3,O7_3)	ObjConsistence(O7_3)	ObjConsistence(O23_3)
ObjClass(Road,O7_4)	ObjClass(Sky,O23_4)	Above(O23_4,O7_4)	ObjConsistence(O7_4)	ObjConsistence(O23_4)
ObjClass(Road,O7_5)	ObjClass(Sky,O23_5)	Above(O23_5,O7_5)	ObjConsistence(O7_5)	ObjConsistence(O23_5)
ObjClass(Road,O7_6)	ObjClass(Sky,O23_6)	Above(O23_6,O7_6)	ObjConsistence(O7_6)	ObjConsistence(O23_6)
ObjClass(Road,O7_7)	ObjClass(Sky,O23_7)	Above(O23_7,O7_7)	ObjConsistence(O7_7)	ObjConsistence(O23_7)
ObjClass(Road,O7_8)	ObjClass(Sky,O23_8)	Above(O23_8,O7_8)	ObjConsistence(O7_8)	ObjConsistence(O23_8)
ObjClass(Road,O7_9)	ObjClass(Sky,O23_9)	Above(O23_9,O7_9)	ObjConsistence(O7_9)	ObjConsistence(O23_9)
ObjClass(Road,O7_10)	ObjClass(Sky,O23_10)	Above(O23_10,O7_10)	ObjConsistence(O7_10)	ObjConsistence(O23_10)
ObjClass(Road,O7_11)	ObjClass(Sky,O23_11)	Above(O23_11,O7_11)	ObjConsistence(O7_11)	ObjConsistence(O23_11)
ObjClass(Road,O7_12)	ObjClass(Sky,O23_12)	Above(O23_12,O7_12)	ObjConsistence(O7_12)	ObjConsistence(O23_12)
ObjClass(Road,O7_13)	ObjClass(Sky,O23_13)	Above(O23_13,O7_13)	ObjConsistence(O7_13)	ObjConsistence(O23_13)
ObjClass(Road,O7_14)	ObjClass(Sky,O23_14)	Above(O23_14,O7_14)	ObjConsistence(O7_14)	ObjConsistence(O23_14)
ObjClass(Road,NO7_0)	ObjClass(Sky,NO23_0)	Above(NO7_0,NO23_0)	!ObjConsistence(NO7_0)	!ObjConsistence(NO23_0)
ObjClass(Road,NO7_1)	ObjClass(Sky,NO23_1)	Above(NO7_1,NO23_1)	!ObjConsistence(NO7_1)	!ObjConsistence(NO23_1)
ObjClass(Road,NO7_2)	ObjClass(Sky,NO23_2)	Above(NO7_2,NO23_2)	!ObjConsistence(NO7_2)	!ObjConsistence(NO23_2)
ObjClass(Road,NO7_3)	ObjClass(Sky,NO23_3)	Above(NO7_3,NO23_3)	!ObjConsistence(NO7_3)	!ObjConsistence(NO23_3)
ObjClass(Road,NO7_4)	ObjClass(Sky,NO23_4)	Above(NO7_4,NO23_4)	!ObjConsistence(NO7_4)	!ObjConsistence(NO23_4)
ObjClass(Road,NO7_5)	ObjClass(Sky,NO23_5)	Above(NO7_5,NO23_5)	!ObjConsistence(NO7_5)	!ObjConsistence(NO23_5)
ObjClass(Road,NO7_6)	ObjClass(Sky,NO23_6)	Above(NO7_6,NO23_6)	!ObjConsistence(NO7_6)	!ObjConsistence(NO23_6)
ObjClass(Road,NO7_7)	ObjClass(Sky,NO23_7)	Above(NO7_7,NO23_7)	!ObjConsistence(NO7_7)	!ObjConsistence(NO23_7)
ObjClass(Road,NO7_8)	ObjClass(Sky,NO23_8)	Above(NO7_8,NO23_8)	!ObjConsistence(NO7_8)	!ObjConsistence(NO23_8)
ObjClass(Road,NO7_9)	ObjClass(Sky,NO23_9)	Above(NO7_9,NO23_9)	!ObjConsistence(NO7_9)	!ObjConsistence(NO23_9)
ObjClass(Road,NO7_10)	ObjClass(Sky,NO23_10)	Above(NO7_10,NO23_10)	!ObjConsistence(NO7_10)	!ObjConsistence(NO23_10
ObjClass(Road,NO7_11)	ObjClass(Sky,NO23_11)	Above(NO7_11,NO23_11)	!ObjConsistence(NO7_11)	!ObjConsistence(NO23_11
ObjClass(Road,NO7_12)	ObjClass(Sky,NO23_12)	Above(NO7_12,NO23_12)	!ObjConsistence(NO7_12)	!ObjConsistence(NO23_12
ObjClass(Road,NO7_13)	ObjClass(Sky,NO23_13)	Above(NO7_13,NO23_13)	!ObjConsistence(NO7_13)	!ObjConsistence(NO23_13
ObjClass(Road,NO7_14)	ObjClass(Sky,NO23_14)	Above(NO7_14,NO23_14)	!ObjConsistence(NO7_14)	!ObjConsistence(NO23_14
ObjClass(Road,NO7_15)	ObjClass(Sky,NO23_15)	Support(NO23_15,NO7_15)	!ObjConsistence(NO7_15)	!ObjConsistence(NO23_15
ObjClass(Road,NO7_16)	ObjClass(Sky,NO23_16)	Support(NO23_16,NO7_16)	!ObjConsistence(NO7_16)	!ObjConsistence(NO23_16
ObjClass(Road,NO7_17)	ObjClass(Sky,NO23_17)	Support(NO23_17,NO7_17)	!ObjConsistence(NO7_17)	!ObjConsistence(NO23_17
ObjClass(Road,NO7_18)	ObjClass(Sky,NO23_18)	Support(NO23_18,NO7_18)	!ObjConsistence(NO7_18)	!ObjConsistence(NO23_18
ObjClass(Road,NO7_19)	ObjClass(Sky,NO23_19)	Support(NO23_19,NO7_19)	!ObjConsistence(NO7_19)	!ObjConsistence(NO23_19
ObjClass(Road,NO7_20)	ObjClass(Sky,NO23_20)	Support(NO23_20,NO7_20)	!ObjConsistence(NO7_20)	!ObjConsistence(NO23_20
ObjClass(Road,NO7_21)	ObjClass(Sky,NO23_21)	Support(NO23_21,NO7_21)	!ObjConsistence(NO7_21)	!ObjConsistence(NO23_21
ObjClass(Road,NO7_22)	ObjClass(Sky,NO23_22)	Support(NO23_22,NO7_22)	!ObjConsistence(NO7_22)	!ObjConsistence(NO23_22
ObjClass(Road,NO7_23)	ObjClass(Sky,NO23_23)	Support(NO23_23,NO7_23)	!ObjConsistence(NO7_23)	!ObjConsistence(NO23_23
ObjClass(Road,NO7_24)	ObjClass(Sky,NO23_24)	Support(NO23_24,NO7_24)	!ObjConsistence(NO7_24)	!ObjConsistence(NO23_24
ObjClass(Road,NO7_25)	ObjClass(Sky,NO23_25)	Support(NO23_25,NO7_25)	!ObjConsistence(NO7_25)	!ObjConsistence(NO23_25
ObjClass(Road,NO7_26)	ObjClass(Sky,NO23_26)	Support(NO23_26,NO7_26)	!ObjConsistence(NO7_26)	!ObjConsistence(NO23_26
ObjClass(Road,NO7_27)	ObjClass(Sky,NO23_27)	Support(NO23_27,NO7_27)	!ObjConsistence(NO7_27)	!ObjConsistence(NO23_27
ObjClass(Road,NO7_28)	ObjClass(Sky,NO23_28)	Support(NO23_28,NO7_28)	!ObjConsistence(NO7_28)	!ObjConsistence(NO23_28
ObjClass(Road,NO7_29)	ObjClass(Sky,NO23_29)	Support(NO23_29,NO7_29)	!ObjConsistence(NO7_29)	!ObjConsistence(NO23_29
ObjClass(Road,NO7_30)	ObjClass(Sky,NO23_30)	Support(NO7_30,NO23_30)	!ObjConsistence(NO7_30)	!ObjConsistence(NO23_30
ObjClass(Road,NO7_31)	ObjClass(Sky,NO23_31)	Support(NO7_31,NO23_31)	!ObjConsistence(NO7_31)	!ObjConsistence(NO23_31
ObjClass(Road,NO7_32)	ObjClass(Sky,NO23_32)	Support(NO7_32,NO23_32)	!ObjConsistence(NO7_32)	!ObjConsistence(NO23_32
ObjClass(Road,NO7_33)	ObjClass(Sky,NO23_33)	Support(NO7_33,NO23_33)	!ObjConsistence(NO7_33)	!ObjConsistence(NO23_33
ObjClass(Road,NO7_34)	ObjClass(Sky,NO23_34)	Support(NO7_34,NO23_34)	!ObjConsistence(NO7_34)	!ObjConsistence(NO23_34
ObjClass(Road,NO7_35)	ObjClass(Sky,NO23_35)	Support(NO7_35,NO23_35)	!ObjConsistence(NO7_35)	!ObjConsistence(NO23_35
ObjClass(Road,NO7_36)	ObjClass(Sky,NO23_36)	Support(NO7_36,NO23_36)	!ObjConsistence(NO7_36)	!ObjConsistence(NO23_36
ObjClass(Road,NO7_37)	ObjClass(Sky,NO23_37)	Support(NO7_37,NO23_37)	!ObjConsistence(NO7_37)	!ObjConsistence(NO23_37
ObjClass(Road,NO7_38)	ObjClass(Sky,NO23_38)	Support(NO7_38,NO23_38)	!ObjConsistence(NO7_38)	!ObjConsistence(NO23_38
ObjClass(Road,NO7_39)	ObjClass(Sky,NO23_39)	Support(NO7_39,NO23_39)	!ObjConsistence(NO7_39)	!ObjConsistence(NO23_39
ObjClass(Road,NO7_40)	ObjClass(Sky,NO23_40)	Support(NO7_40,NO23_40)	!ObjConsistence(NO7_40)	!ObjConsistence(NO23_40
ObjClass(Road,NO7_41)	ObjClass(Sky,NO23_41)	Support(NO7_41,NO23_41)	!ObjConsistence(NO7_41)	!ObjConsistence(NO23_41
ObjClass(Road,NO7_42)	ObjClass(Sky,NO23_42)	Support(NO7_42,NO23_42)	!ObjConsistence(NO7_42)	!ObjConsistence(NO23_42
ObjClass(Road,NO7_43)	ObjClass(Sky,NO23_43)	Support(NO7_43,NO23_43)	!ObjConsistence(NO7_43)	!ObjConsistence(NO23_43
ObjClass(Road,NO7_44)	ObjClass(Sky,NO23_44)	Support(NO7_44,NO23_44)	!ObjConsistence(NO7_44)	!ObjConsistence(NO23_44

6.4.2.4 Lernen der Gewichtungen von MLN-Formeln

Zum Lernen der Gewichtungen von MLN-Formeln und der Inferenz von MLN-Modellen wurde das *Alchemy*-Tool [201] verwendet. Dieses Tool lief in einer virtuellen Maschine (VM) mit Ubuntu 10.04 LTS als Betriebssystem. Das Hostsystem hatte als CPU eine Intel-i7-Architektur mit 2,8 GHz und acht Kernen. Die RAM-Größe des Hostsystems betrug acht GB. Die VM durfte auf maximal vier Kernen und 3,3 GB Speicher zugreifen. Das *Alchemy*-Tool nahm als Input für das Training die vereinfachten logischen Formeln des MLN-Modells (siehe Abbildung 6-14) und die generierten Trainingsdaten (siehe Tabelle 6-5). Als Output lieferte dieses Tool die gelernten Gewichtungen der MLN-Formeln. In dieser Arbeit wurden sowohl das diskriminative als auch das generative Lernen verwendet. Darüber hinaus wurde das Prädikat $ObjConsistence$ als Nicht-Evidenz gesetzt, da dieses Prädikat in der Infe-

renzphase als Abfrage-Prädikat diente. Damit wurde sichergestellt, dass die Gewichtungen aller Formeln, die das Prädikat $ObjConsistence$ beinhalteten, gelernt wurden und dass das Prädikat $ObjConsistence$ als Abfrage-Prädikat für die bedingte Wahrscheinlichkeit beim diskriminativen Lernen verwendet wurde. Ferner wurden beim Training atomare Formeln nicht betrachtet. Alle weiteren Einstellungen des *Alchemy*-Tools blieben unverändert. Das *Alchemy*-Tool lernte die Gewichtungen generativ anhand des *L-BFGS*-Algorithmus, was im Abschnitt 2.3.2.1 erwähnt wurde. Für das diskriminative Lernen wurde der *Conjugate-gradient*-Algorithmus in Kombination mit dem *MaxWalk-Sat-Solver* verwendet (siehe Abschnitt 2.3.2.2).

Die Tabelle 6-6 stellt die MLN-Modelle dar, die in dieser Arbeit gelernt wurden. Diese MLN-Modelle unterscheiden sich bis auf die Varianten MLN_{AbvFAI} und MLN_{AbvFAS} durch die räumlichen Relationsmengen, die für die logischen Regeln dieser MLN-Modelle verwendet wurden. Diese Relationsmengen sind in der zweiten Spalte der Tabelle 6-6 dargestellt. Für die Varianten MLN_{AbvFAI} und MLN_{AbvFAS} wurde zwar die räumliche Relation $Above$ verwendet, aber die Merkmale zur Schätzung dieser Relation unterschieden sich. Für das MLN_{AbvFAI}-Modell wurde der Merkmalvektor aus der Gleichung (37) verwendet, während für das MLN_{AbvFAS}-Modell die Gleichung (43) angewendet wurde. Die Anzahl der Klauseln eines MLN-Modells ließ sich durch die folgende Gleichung berechnen:

$$\#Klausel = |\mathcal{C}|^2 * \left|\hat{R}_{si}\right| * 2 + \#Axiomeklause \tag{49}$$

$|\mathcal{C}|$ und $\left|\hat{R}_{si}\right|$ waren jeweils die Anzahl der semantischen Klassen und der Relationen. So hatten in der dritten Spalte der Tabelle 6-6 die Modelle MLN_{AbvIns} und MLN_{AbvSpp} etwas mehr als zweimal so viele Klauseln wie die anderen MLN-Modelle, da die Modelle MLN_{AbvIns} und MLN_{AbvSpp} zweimal so viele Relationen hatten wie die anderen MLN-Modelle. Dazu hatten die Modelle MLN_{AbvIns} und MLN_{AbvSpp} vier Axiome-Klauseln mehr als die anderen MLN-Modelle. Analog hatte jedes MLN-Modell maximal

$$\#Trainingsgrundprädikate = |K| * 2 * \#Klausel * 4 \tag{50}$$

Trainingsgrundprädikate, wobei K die Konstante aus der Gleichung (47) war. Die Anzahl der Grundprädikate für das Training war also proportional zu der Anzahl der Klauseln der MLN-Modelle. Die vierte Spalte der Tabelle 6-6 stellte diese Anzahl der Trainingsgrundprädikate dar. Für alle MLN-Modelle war die Anzahl der Grundprädikate deutlich kleiner als der maximale Wert aus der Gleichung (50), weil viele Grundprädikate gleichzeitig in mehreren Grundformeln verwendet wurden. Der maximale Wert aus der Gleichung (50) setzte voraus, dass jedes Grundprädikat nur einmal in einer Grundformel verwendet wurde. So hatte beispielsweise das MLN-Modell MLN_{AbvIns} mit 14 629 Grundprädikaten ca. 26,3-mal weniger Grundprädikate als die maximale Anzahl an Grundprädikaten. Die Verwendung der Grundprädikate gleichzeitig in mehreren Grundformeln hatte den Vorteil, dass die Trainingszeit deutlich reduziert wurde.

Die Trainingsdauer aller MLN-Modelle in der fünften Spalte der Tabelle 6-6 zeigte, dass je mehr Klauseln und Grundprädikaten die Modelle hatten, desto länger dauerte das Training. So war das Training von einfacheren MLN-Modellen wie MLN_{Ins} und MLN_{Spp} bis zu zehnmal schneller als das Training von komplexeren MLN-Modellen wie MLN_{AbvIns} und MLN_{AbvSpp}. Das diskriminative Training war bis 30-mal langsamer als das generative Training. Die Trainingsdauer für generative Modelle lag zwischen ca. drei und 48 Minuten, während die Trainingsdauer bei diskriminativen Modellen zwischen 39 und 204 Minuten lag. Die hier gemessenen Trainingszeiten waren bis zu viermal größer als die sogenannte *User-Zeit* von Ubuntu, wobei die *User-Zeit* nur die CPU-Zeit, die vom Prozess benötigt wurde, beinhaltete. Eine mögliche Erklärung war, dass viele Trainings parallel gestartet wurden und somit viele Prozesse auf die CPU warten mussten. Damit stieg die gesamte Trainingsdauer.

Tabelle 6-6: Übersicht der trainierten MLN-Modelle zur Schätzung der Konsistenz von Regionen bezogen auf räumliche Relationen

Name	Räumliche Relationen	#Klausel	#Trainings-grundprädikate	Trainings-dauer/ Generativ (mm:ss)	Trainings-dauer/ Diskriminativ (mm:ss)
$\boldsymbol{MLN_{AbvIns}}$	$\hat{R}_{s1} = \{Above, Inside\}$	1604	14629	35:13	204:20
$\boldsymbol{MLN_{AbvSpp}}$	$\hat{R}_{s2} = \{Above, Support\}$	1604	14646	47:54	157:21
$\boldsymbol{MLN_{AbvFAI}}$	$\hat{R}_{s3} = \{Above\}$	800	7550	10:39	39:18
$\boldsymbol{MLN_{AbvFAS}}$	$\hat{R}_{s4} = \{Above\}$	800	7590	03:37	90:37
$\boldsymbol{MLN_{Ins}}$	$\hat{R}_{s5} = \{Inside\}$	800	7433	03:18	72:22
$\boldsymbol{MLN_{Spp}}$	$\hat{R}_{s6} = \{Support\}$	800	7389	03:22	87:44

6.4.3 Interpretation der gelernten MLN-Modelle

Die MLN-Modelle aus der Tabelle 6-6 hatten Formeln, die sich logische Variablen und Prädikate teilten. Deshalb konnten, wie im Abschnitt 2.3.2.3 beschrieben, die gelernten Gewichtungen nicht ein zu eins mit einzelnen Werten der Häufigkeitsmatrizen π_r, die zur Generierung der Trainingsdaten verwendet wurden (siehe Gleichung (46)), verglichen werden. Die Abbildung 6-15 stellt einen Ausschnitt des generativ trainierten MLN_{AbvSpp}-Modells für die Klassen *road*, *sky* und *car* dar. Eine Formel wurde F_i genannt, wenn diese sich auf der i-ten Zeile der Abbildung 6-15 befand. So stellten die Formeln F_2 bis F_5 die Prädikatendeklaration dar. Die Zeile acht enthielt das Axiom, dass die Relationen *Above* und *Support* sich wechselseitig ausschlossen. Die Formeln F_9 bis F_{11} waren die entsprechenden Klauseln dieses Axioms. Alle Axiom-Formeln endeten mit einem Punktzeichen.

Auf den Zeilen 15 bis 26 waren die logischen Formeln zwischen den Klassen *road* und *sky* zu finden. Zwischen den Klassen *road* und *sky* gab es insgesamt vier Formeln F_{15}, F_{18}, F_{21} und F_{24} auf den Zeilen, die mit Kommentarzeichen (//) anfingen. Nach dem Kommentarzeichen folgten die Gewichtungen w_{15}, w_{18}, w_{21} und w_{24} dieser Formeln. Jede dieser vier Formeln hatte jeweils zwei Klauseln. Die Summe der Gewichtungen der Klauseln einer Formel ergab die Gewichtung der entsprechenden Formel. Die Formel F_{18} mit der Gewichtung $w_{18} = 1{,}765$ modellierte die Wahrscheinlichkeit, dass zwei Regionen $o_1$und o_2 konsistent waren, wenn o_1 der Klasse *sky* gehörte, o_2 der Klasse *road* gehörte und o_1 oberhalb von o_2 war. Im Vergleich zu den Fällen, wo o_2 oberhalb von o_1 (F_{15}) war, o_1 o_2 unterstützte (F_{21}) und o_2 o_1 unterstützte (F_{24}), war die Konsistenzwahrscheinlichkeit von o_1 und o_2 für den Fall, wo o_1 oberhalb von o_2 lag, deutlich höher, da die entsprechende Formel F_{18} mit $w_{18} = 1{,}765$ eine größere Gewichtung hatte, als alle andere Formeln F_{15} ($w_{15} - 1{,}918$), F_{21} ($w_{21} = -0{,}88$) und F_{24} ($w_{24} = -0{,}557$). Dies entsprach auch den Erkenntnissen aus den Häufigkeitsmatrizen π_r des Abschnitts 6.4.2.1.2. Der Häufigkeitswert $\pi_{abv}(10{,}0) = 99{,}4\ \%$ aus der Abbildung 6-10, der für die Generierung der Trainingsdaten der Formel F_{18} relevant war, zeigte, dass sich Regionen der Klassen *sky* mit 99,4 % sehr oft oberhalb von Regionen der Klasse *road* in den Trainingsdaten befanden. Somit waren solche Regionen mit einer hohen Wahrscheinlichkeit konsistent.

Die logischen Formeln zwischen den Klassen *road* und *car* befanden sich auf den Zeilen 28 bis 39. Die Wahrscheinlichkeit, dass zwei Regionen $o_1$und o_2 konsistent waren, wenn o_1 der Klasse *road* gehörte, o_2 der Klasse *car* gehörte und o_1 die Supportebene von o_2 war, wurde durch die Formel F_{34} mit $w_{34} = -0{,}032$ modelliert. Im Vergleich zu den Fällen, wo o_2 oberhalb von o_1 (F_{28}) war, o_1 oberhalb von o_2 (F_{31}) war und o_2 o_1 unterstützte (F_{37}), waren die Konsistenzwahrscheinlichkeiten von o_1 und o_2 für den Fall, wo o_1 die Supportebene von o_2 war, höher, da die entsprechende Formel F_{34} mit $w_{34} = -0{,}032$ eine größere Gewichtung hatte, als alle andere Formeln F_{28} ($w_{28} = -0{,}385$), F_{31} ($w_{31} = -0{,}815$) und F_{37} ($w_{37} = -0{,}543$). Allerdings war die Gewichtung $w_{34} = -0{,}032$ kleiner als

die Gewichtung w_{18} der Formel F_{18}. Die Erklärung dafür war, dass der Häufigkeitswert $\pi_{spp}(10{,}0) = 32{,}3\ \%$ aus der Abbildung 6-13, der für die Generierung der Trainingsdaten für die Formel F_{34} relevant war, deutlich unter dem Häufigkeitswert der $\pi_{abv}(0{,}13) = 99{,}4\ \%$ aus der Abbildung 6-10 zur Generierung der Trainingsdaten für die Formel F_{18} lag. Allerdings wurde auch festgestellt, dass wenn der entsprechende Häufigkeitswert zur Generierung der Trainingsdaten einer Formel F_i größer war als der Häufigkeitswert zur Generierung der Trainingsdaten einer anderen Formel F_j, war die Gewichtung w_i der Formel F_i nicht immer größer als die Gewichtung w_j der Formel F_j. So war die Gewichtung $w_{28} = -0{,}385$ der Formel F_{28} größer als die Gewichtung $w_{31} = -0{,}815$ der Formel F_{31}, obwohl der Häufigkeitswert $\pi_{abv}(0{,}13) = 27{,}4\ \%$ zur Generierung der Trainingsdaten der Formel F_{28} größer als der Häufigkeitswert $\pi_{abv}(13{,}0) = 31{,}5\ \%$ zur Generierung der Trainingsdaten der Formel F_{31} war. Diese Festlegung untermauerte die Aussage, dass die Gewichtungen für die Gesamtwahrscheinlichkeit aller Formeln eines MLN-Modells und nicht für einzelne Formeln des MLN-Modells optimiert wurden.

```
/**** predicate declarations ****/
ObjConsistence(object)
Support(object,object)
Above(object,object)
ObjClass(className!,object)

/**** Axiomes ****/
// !(Above(o1,o2) <=> Support(o1,o2)).
Above(a1,a2) v !Above(a1,a2).
Above(a1,a2) v Support(a1,a2).
!Above(a1,a2) v !Support(a1,a2).
Support(a1,a2) v !Support(a1,a2).

/**** rules ****/
// -1.91879  ObjClass(Road,o1) ^ ObjClass(Sky,o2) ^ Above(o1,o2) => (ObjConsistence(o1) ^ ObjConsistence(o2))
-1.36798    !Above(a1,a2) v !ObjClass(Road,a1) v !ObjClass(Sky,a2) v ObjConsistence(a1)
-0.550812   !Above(a1,a2) v !ObjClass(Road,a1) v !ObjClass(Sky,a2) v ObjConsistence(a2)
// 1.76552  ObjClass(Sky,o1) ^ ObjClass(Road,o2) ^ Above(o1,o2) => (ObjConsistence(o1) ^ ObjConsistence(o2))
0.296735    !Above(a1,a2) v !ObjClass(Road,a2) v !ObjClass(Sky,a1) v ObjConsistence(a1)
1.46878     !Above(a1,a2) v !ObjClass(Road,a2) v !ObjClass(Sky,a1) v ObjConsistence(a2)
// -0.880843  ObjClass(Road,o1) ^ ObjClass(Sky,o2) ^ Support(o1,o2) => (ObjConsistence(o1) ^ ObjConsistence(o2))
-0.575718  !Support(a1,a2) v !ObjClass(Road,a1) v !ObjClass(Sky,a2) v ObjConsistence(a1)
-0.305125  !Support(a1,a2) v !ObjClass(Road,a1) v !ObjClass(Sky,a2) v ObjConsistence(a2)
// -0.55765  ObjClass(Sky,o1) ^ ObjClass(Road,o2) ^ Support(o1,o2) => (ObjConsistence(o1) ^ ObjConsistence(o2))
-0.255395  !Support(a1,a2) v !ObjClass(Road,a2) v !ObjClass(Sky,a1) v ObjConsistence(a1)
-0.302255  !Support(a1,a2) v !ObjClass(Road,a2) v !ObjClass(Sky,a1) v ObjConsistence(a2)

// -0.384925  ObjClass(Road,o1) ^ ObjClass(Car,o2) ^ Above(o1,o2) => (ObjConsistence(o1) ^ ObjConsistence(o2))
-0.0506219  !Above(a1,a2) v !ObjClass(Road,a1) v !ObjClass(Car,a2) v ObjConsistence(a1)
-0.334303   !Above(a1,a2) v !ObjClass(Road,a1) v !ObjClass(Car,a2) v ObjConsistence(a2)
// -0.814827  ObjClass(Car,o1) ^ ObjClass(Road,o2) ^ Above(o1,o2) => (ObjConsistence(o1) ^ ObjConsistence(o2))
-0.248094  !Above(a1,a2) v !ObjClass(Road,a2) v !ObjClass(Car,a1) v ObjConsistence(a1)
-0.566732  !Above(a1,a2) v !ObjClass(Road,a2) v !ObjClass(Car,a1) v ObjConsistence(a2)
// -0.0321062  ObjClass(Road,o1) ^ ObjClass(Car,o2) ^ Support(o1,o2) => (ObjConsistence(o1) ^ ObjConsistence(o2))
-0.127743  !Support(a1,a2) v !ObjClass(Road,a1) v !ObjClass(Car,a2) v ObjConsistence(a1)
0.095637   !Support(a1,a2) v !ObjClass(Road,a1) v !ObjClass(Car,a2) v ObjConsistence(a2)
// -0.543184  ObjClass(Car,o1) ^ ObjClass(Road,o2) ^ Support(o1,o2) => (ObjConsistence(o1) ^ ObjConsistence(o2))
-0.264444  !Support(a1,a2) v !ObjClass(Road,a2) v !ObjClass(Car,a1) v ObjConsistence(a1)
-0.278741  !Support(a1,a2) v !ObjClass(Road,a2) v !ObjClass(Car,a1) v ObjConsistence(a2)
```

Abbildung 6-15: Ausschnitt des generativ trainierten MLN_{AbvSpp}-Modells für die Klassen road, sky und car

Die Erkenntnisse, die für das Beispiel aus der Abbildung 6-15 gewonnen wurden, galten auch für andere MLN-Modelle, die in diesem Abschnitt trainiert wurden.

6.4.4 Metriken zur quantitativen Evaluation der gelernten MLN-Modelle

Das Ziel dieser Evaluation war herauszufinden, ob es eine Korrelation zwischen inkonsistenten Regionen und falsch segmentierten Regionen gab. Im optimalen Fall sollten richtig segmentierte Regionen konsistent sein, während falsch segmentierte Regionen inkonsistent sein sollten. Für die Metriken wurden folgende Definitionen festgelegt:

1. *TP*-Regionen sind Regionen, die richtig segmentiert wurden und gleichzeitig als konsistent geschätzt wurden.
2. *FP*-Regionen sind Regionen, die falsch segmentiert wurden, aber als konsistent geschätzt wurden.
3. *FN*-Regionen sind Regionen, die richtig segmentiert wurden, aber als inkonsistent geschätzt wurden.

4. *TN*-Regionen sind Regionen, die falsch segmentiert wurden und gleichzeitig als inkonsistent geschätzt wurden.

Basierend auf den o. g. Definitionen wurde die *Receiver-Operating-Characteristic*-(*ROC*-)Kurve (siehe Gleichungen (51) und (52)) für unterschiedliche MLN-Modelle generiert. Die *ROC*-Kurve hatte auf ihrer x-Achse *False-Positive-Rate-(FPR-)*Werte und *Recall*-Werte auf ihrer y-Achse.

$$Recall = \frac{TP}{TP + FN} \tag{51}$$

$$FPR = \frac{FP}{TN + FP} \tag{52}$$

Für die *ROC*-Kurve wurde die *Area-Under-the-Curve*-(*AUC*-)Metrik berechnet. Die *AUC*-Metrik berechnete die Fläche zwischen einer Kurve $f(x)$ und der x-Achse.

$$AUC\big(f(x)\big) = \int_{-\infty}^{+\infty} f(x)dx \tag{53}$$

Der *AUC-ROC*-Wert für die *ROC*-Kurve lag zwischen 0 und 1. Je näher ein *AUC-ROC*-Wert an 1 war, desto besser war dieser *AUC-ROC*-Wert. Im optimalen Fall sollte der *AUC-ROC*-Wert also bei 1 liegen.

Neben der *AUC-ROC*-Metrik wurden Histogramme der Verteilung der Konsistenzwahrscheinlichkeiten für *positive* und *negative* Pixel erstellt. Positive Pixel gehörten zu Regionen, die vom *FCN-8s*-Modell richtig segmentiert wurden. Negative Pixel gehörten zu falsch segmentierten Regionen. Im optimalen Fall sollte das Histogramm von positiven Pixeln einen Mittelwert von eins haben, während das Histogramm von negativen Pixeln einen Mittelwert bei null haben sollte. Bei Abweichung des Histogrammmittelwerts vom optimalen Fall sollte die Abweichung des Histogramms von positiven Pixeln mehr als die Abweichung des Histogramms von negativen Pixeln bestraft werden. So ergab sich die Metrik namens *gewichtete Differenz der Histogrammmittelwerte* (GDHM):

$$GDHM = \mu_H^P * 0{,}75 + (1 - \mu_H^N) * 0{,}25 \tag{54}$$

Wobei $\mu_H^P \in [0{,}1]$ und $\mu_H^N \in [0{,}1]$ jeweils den Mittelwert der Histogramme der positiven und negativen Pixel waren. Die Metrik $GDHM \in [0{,}1]$ konvergierte gegen eins im optimalen Fall, wenn $\mu_H^P = 1$ und $\mu_H^N = 0$ erfüllt waren.

6.4.5 Quantitative Evaluation der gelernten MLN-Modelle

Zur Inferenz der trainierten MLN-Modelle wurden das *Alchemy*–Tool und das *MC-SAT*-Verfahren [204] verwendet. Die Inferenz nahm als Input die trainierten MLN-Modelle und die Regionen, die vom *FCN-8s*-Modell aus Kapitel 5 segmentiert wurden, und gab die Konsistenzwahrscheinlichkeiten der segmentierten Regionen zurück (siehe Schritte 10 bis 14 der Abbildung 6-2). Die Bilder des *DCS*-Validierungsdatensatzes wurden als Input der Segmentierung verwendet. In den nächsten Abschnitten werden die Inferenzergebnisse ausführlich präsentiert.

6.4.5.1 Evaluation der generativ trainierten MLN-Modelle

Die Abbildung 6-16 stellt die *ROC*-Kurven der generativ trainierten MLN-Modelle aus der Tabelle 6-6 auf dem *DCS*-Validierungsdatensatz dar. Diese Abbildung zeigte, dass das MLN-Modell MLN_{Spp} am besten war, da seine *ROC*-Kurve namens *ROC-Spp* oberhalb aller anderen Kurven lag. Darüber hinaus war der *AUC-ROC*-Wert des MLN-Modells MLN_{Spp} mit 0,656 aus der Tabelle 6-7 der größte *AUC-ROC*-Wert. Danach folgte das MLN_{Ins}-Modell mit einem *AUC-ROC*-Wert von 0,634. Die entsprechende *ROC*-Kurve *ROC-Ins* des MLN_{Ins}-Modells lag direkt unterhalb der *ROC-Spp*-Kurve. Die Modelle MLN_{AbvFAI} und MLN_{AbvIns} waren schlechter als die o. g. Modelle MLN_{Spp} und MLN_{Ins}. Ferner hat-

ten die Modelle MLN_{AbvFAI} und MLN_{AbvIns} *ROC*-Kurven, die sehr nah aneinander lagen. Die *AUC-ROC*-Werte der Modelle MLN_{AbvFAI} und MLN_{AbvIns} lagen bei 0,584 und 0,577. Anschließen waren die Modelle MLN_{AbvFAS} und MLN_{AbvSpp} am schlechtesten und hatten *AUC-ROC*-Werte bei 0,508 und 0,571. Aus diesen Ergebnissen ließ sich Folgendes feststellen:

1. Die Modelle MLN_{Spp} und MLN_{Ins} waren in der Lage, die Inkonsistenzen am besten zu erfassen. Dass die Differenz der *AUC-ROC*-Werte dieser beiden Modelle von 2,2 % sehr gering war, deutete darauf hin, dass die beiden MLN-Modelle ähnliche Inkonsistenzen erfassten. Diese Vermutung stimmte, da die Merkmale, die zur Schätzung der räumlichen Relationen für die Modelle MLN_{Spp} und MLN_{Ins} sehr ähnlich waren. Diese Merkmale wurden im Abschnitt 6.4.2.1 vorgestellt.
2. Der Unterschied zwischen den Modellen MLN_{AbvFAI} und MLN_{AbvIns} sowie zwischen MLN_{AbvFAS} und MLN_{AbvSpp} war gering. Dies bedeutete, dass die räumliche Relation $Above$ eine wesentliche Rolle bei diesen MLN-Modellen spielte, da diese Relation in den vier MLN-Modellen vorhanden war. Die Relation $Above$ könnte die schlechten Ergebnisse dieser MLN-Modelle erklären. In der Tat zeigten die Häufigkeitsmatrizen π_{abv} (siehe Abbildung 6-6) und π_{abv}^{s} (siehe Abbildung 6-10), die in der Gleichung (47) verwendet wurden, um die Trainingssamples der MLN-Modelle für die Relation $Above$ zu generieren, Maxima von bis zu 99 % für die Klassen *traffic ligth, traffic sign* und *sky*. Die Maxima der restlichen Klassen waren kleiner und lagen meistens zwischen 20 % und 50 %. Der große Unterschied zwischen den Klassenmaxima führte dazu, dass die Klassen mit Maxima von bis zu 99 % öfter als konsistent geschätzt wurden, während Klassen mit Maxima zwischen 20 % und 50 % als inkonsistent geschätzt wurden. Im Vergleich zu den Maxima der Häufigkeitsmatrizen π_{abv} und π_{abv}^{s} waren die Maxima der Häufigkeitsmatrizen π_{ins} (siehe Abbildung 6-8) und π_{spp}^{s} (siehe Abbildung 6-13) deutlich kleiner. Deswegen hatte die Relation $Above$ einen größeren Einfluss als die Relationen $Inisde$ und $Support$ auf die MLN-Modelle MLN_{AbvIns} (Kombination der Relationen $Above$ und $Inisde$) und MLN_{AbvSpp} (Kombination der Relationen $Above$ und $Support$). Dafür waren aber die Unterscheide zwischen den Klassenmaxima in π_{ins} und π_{spp}^{s} deutlich geringer als in π_{abv} und π_{abv}^{s}. So konnten die MLN-Modelle MLN_{Spp} (basierend auf $Support$) und MLN_{Ins} (basierend auf $Inisde$) bessere *AUC-ROC*-Werte als alle andere Modelle, in denen die Relation $Above$ involviert war, erreichen (siehe Tabelle 6-7).

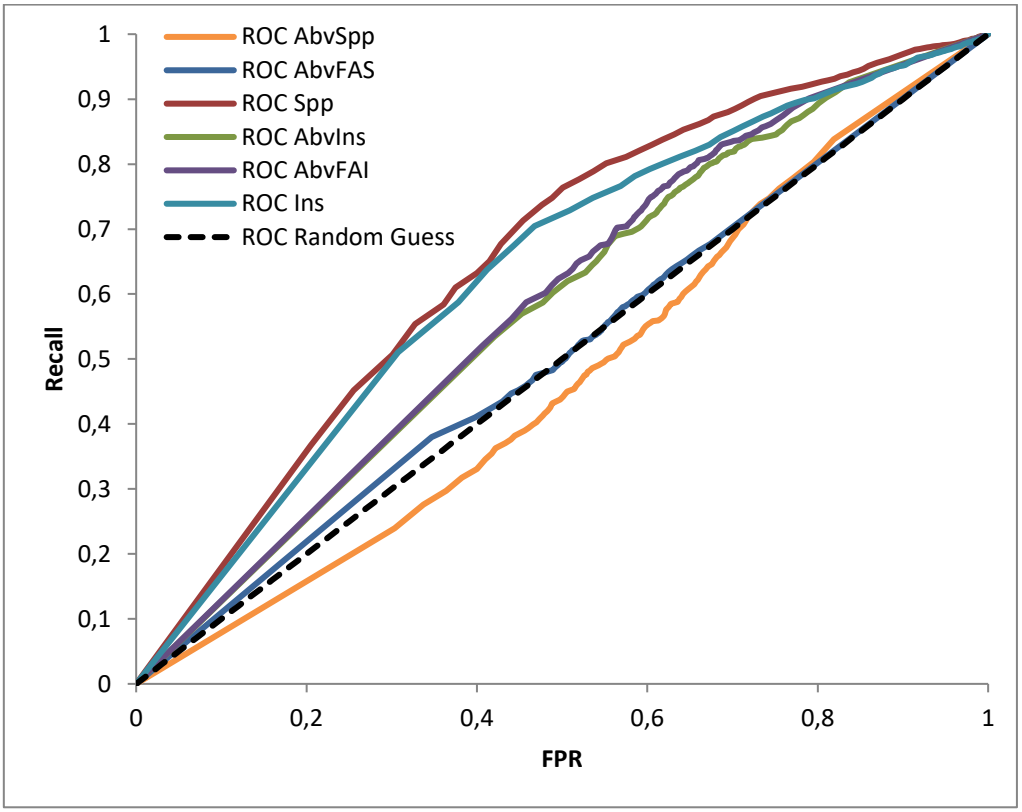


Abbildung 6-16: Evaluation der generativ trainierten MLN-Modelle aus der Tabelle 6-6 auf dem DCS-Validierungsdatensatzes anhand der ROC-Kurve

Die Abbildung 6-17 und Abbildung 6-18 stellen die Histogramme der Konsistenzwahrscheinlichkeiten für die vom *FCN-8s*-Modell jeweils richtig (positiv) und falsch (negativ) segmentierten Pixel auf dem *DCS*-Validierungsdatensatzes dar. Die Konsistenzwahrscheinlichkeiten wurden mit den generativ trainierten MLN-Modellen aus der Tabelle 6-6 geschätzt. Diese Abbildungen bestätigten die Beobachtungen der *AUC*-Metriken, dass die MLN-Modelle MLN_{Spp} und MLN_{Ins} am besten in der Lagen waren, Inkonsistenzen zu erfassen, da die Mittelwerte der positiven Histogramme dieser MLN-Modelle mit $\mu^{P}_{H\_Spp} = 0{,}891$ und $\mu^{P}_{H\_Ins} = 0{,}916$ am nächsten zu eins lagen. Diese beiden MLN-Modelle hatten mit jeweils $GDHM_{Spp} = 0{,}721$ und $GDHM_{Ins} = 0{,}724$ die höchsten *GDHM*-Werte (siehe dritte Spalte der Tabelle 6-7). Die Tatsache, dass die Mittelwerte der negativen Histogramme dieser MLN-Modelle kleiner als die Mittelwerte der positiven Histogramme waren, bedeutete, dass die negativen Pixel inkonsistenter waren als die positiven. Für einen Konsistenzschwellenwert von 0,5 waren bspw. für das MLN-Modell MLN_{Spp} 13,5 % der negativen Pixel inkonsistent, während nur 4,4 % der positiven Pixel inkonsistent waren.

Die Histogramme der Abbildung 6-17 und Abbildung 6-18 bestätigten auch die Behauptung, dass die Relation $Above$ die schlechten Ergebnisse der Modelle MLN_{AbvSpp} und MLN_{AbvIns} verursachte. So hatten die Histogramme der Modelle MLN_{AbvSpp} und MLN_{AbvFAS} sowie MLN_{AbvIns} und MLN_{AbvFAI} sehr ähnliche Verläufe. Auch die *GDHM*-Werte dieser Modelle aus der Tabelle 6-7 waren sehr ähnlich.

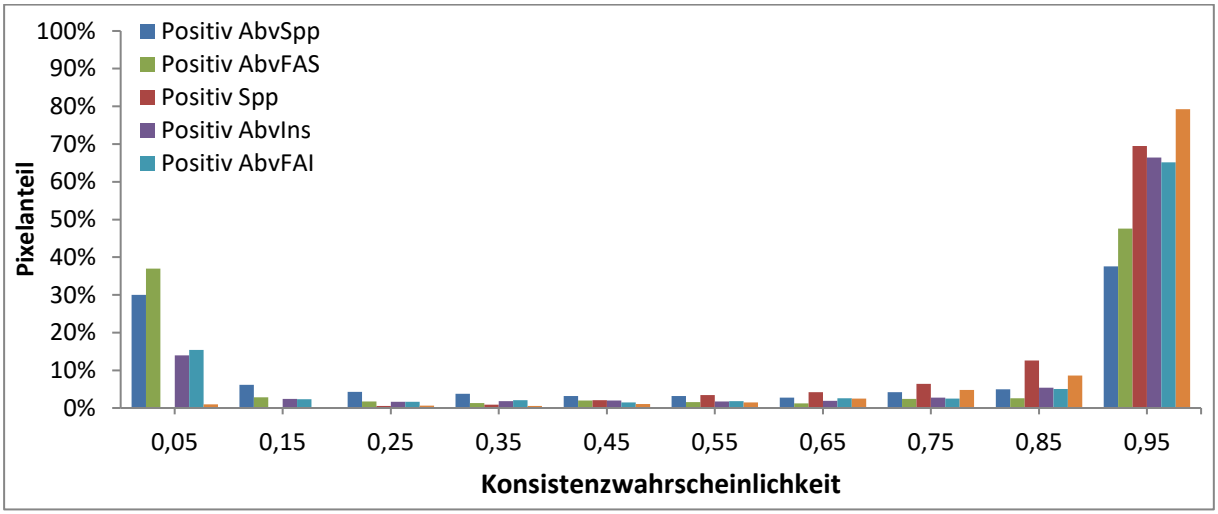


Abbildung 6-17: Histogramm der Konsistenzwahrscheinlichkeiten für die vom FCN-8s-Modell richtig segmentierten Pixel (positiv) auf dem DCS-Validierungsdatensatz. Die Konsistenzwahrscheinlichkeiten wurden mit den generativ trainierten MLN-Modellen aus der Tabelle 6-6 geschätzt.

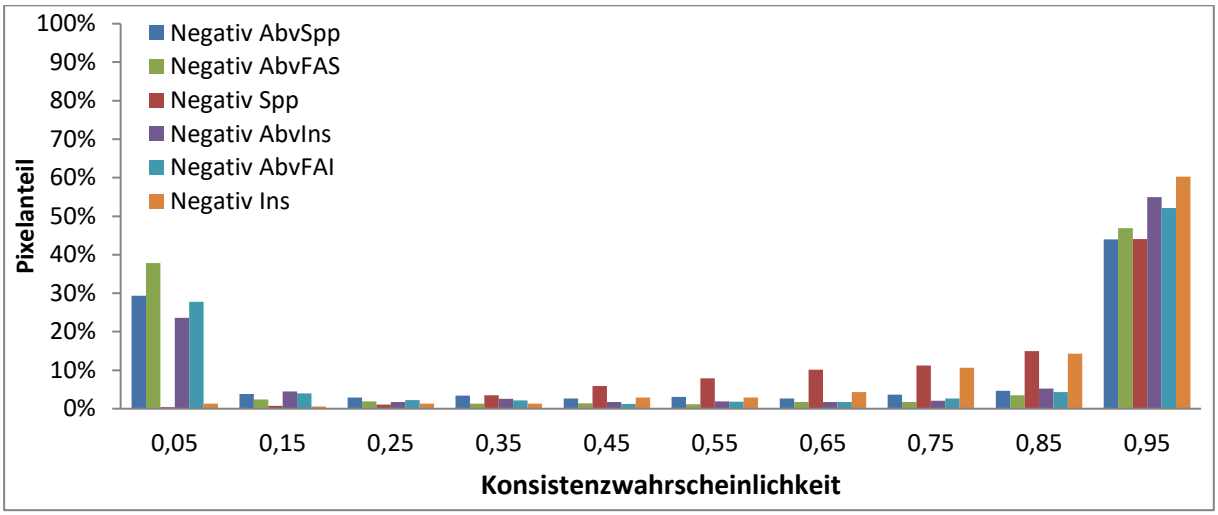


Abbildung 6-18: Histogramm der Konsistenzwahrscheinlichkeiten für die vom FCN-8s-Modell falsch segmentierten Pixel (negativ) auf dem DCS-Validierungsdatensatz. Die Konsistenzwahrscheinlichkeiten wurden mit den generativ trainierten MLN-Modellen aus der Tabelle 6-6 geschätzt.

Die vierte und fünfte Spalte der Tabelle 6-7 zeigen die mittlere *User*- und Gesamtinferenzzeit pro Bild der MLN-Modelle während der Konsistenzschätzung auf dem *DCS*-Validierungsdatensatz. Die *User*-Inferenzzeit war die CPU-Zeit, die die Inferenz benötigte. Die Gesamtinferenzzeit war die Zeit zwischen dem Start und dem Ende der Inferenz. Dabei wurde neben der CPU-Zeit auch die Zeit, die der Inferenzprozess beim Warten auf Ressourcen (z. B. Speicher, CPU etc.) verbraucht hatte, gezählt. Die Ergebnisse zeigten, dass die Inferenzzeiten zwischen 10 s und 42 s lagen. Der Grund für die hohen Inferenzzeiten war die Komplexität der MLN-Modelle. Die einfacheren Modelle MLN_{Spp} und MLN_{Ins} waren mit ca. 10 s *User*-Inferenzzeit und 23 s Gesamtinferenzzeit am Schnelltesten. Diese beiden Modelle waren im Vergleich zu den komplexeren Modellen MLN_{AbvSpp} und MLN_{AbvIns} bis zu viermal schneller. Je komplexer ein MLN-Modell war, desto länger dauerte also die Inferenz. Ein weiterer Grund für die längere Inferenzzeit lag sicherlich in der VM, die für die Inferenz verwendet wurde (siehe Details zur VM im Abschnitt 6.4.2.4), da die VM einen eingeschränkten Zugriff auf die CPU und den Speicher hatte. Darüber hinaus wurden mehrere Inferenzprozesse gleichzeitig gestartet. Die gemessenen *User*-Inferenzzeiten waren bis zu zweimal kleiner als die Gesamtinferenzzeiten. Während der Inferenz mussten also die Prozesse fast die Hälfte der Zeit auf die Ressourcen warten.

Tabelle 6-7: Evaluationsergebnisse der generativ trainierten MLN-Modelle aus der Tabelle 6-6 anhand der mittleren Inferenzzeit, der AUC-ROC- und GDHM-Metriken. Hier wurde der DCS-Validierungsdatensatz verwendet.

Name	AUC-ROC	GDHM	Mittlere User-Laufzeit (s)	Mittlere Gesamt-laufzeit (s)
MLN_{Spp}	**0,656**	0,721	**10,073**	**23,826**
MLN_{Ins}	0,634	**0,724**	10,113	23,839
MLN_{AbvFAI}	0,584	0,659	18,680	35,503
MLN_{AbvIns}	0,577	0,661	19,522	39,914
MLN_{AbvFAS}	0,508	0,524	18,756	35,636
MLN_{AbvSpp}	0,471	0,502	21,099	42,539

6.4.5.2 Evaluation der diskriminativ trainierten MLN-Modelle

Die Evaluation der diskriminativ trainierten MLN-Modelle ist in der Tabelle 6-8 dargestellt. Bezogen auf die *AUC*-Metrik der *ROC*-Kurven waren die diskriminativ trainierten MLN-Modelle kaum von den generativ trainierten MLN-Modellen zu unterscheiden. So lag die relative Verbesserung der *AUC-ROC*-Werte der *ROC*-Kurven der diskriminativ trainierten MLN-Modelle im Vergleich zu den generativ trainierten MLN-Modellen für alle MLN-Modelle außer MLN_{Spp} zwischen 0,6 % und 5 %. Lediglich verschlechterte sich das diskriminativ trainierte Modell MLN_{Spp} um 1,4 %, wobei diese Verschlechterung gering war.

Analog zur *AUC-ROC*-Metrik zeigte die *GDHM*-Metrik für die diskriminativ trainierten MLN-Modelle aus der dritten Spalte der Tabelle 6-8 bis auf MLN_{AbvSpp} kaum Änderungen im Vergleich zu den generativ trainierten MLN-Modellen. Allerdings zeigte der *GDHM*-Wert für die diskriminativ trainierte MLN-Variante MLN_{AbvSpp} eine deutliche relative Verbesserung von 23,5 % im Vergleich zu dem Fall, in dem das MLN-Modell MLN_{AbvSpp} generativ trainiert wurde (siehe dritte Spalte der Tabelle 6-7). Der Grund dafür war, dass die von dem diskriminativ trainierten Modell MLN_{AbvSpp} geschätzten Konsistenzwahrscheinlichkeiten der positiven Pixel größer waren als die Konsistenzwahrscheinlichkeiten der positiven Pixel, die vom generativ trainierten MLN_{AbvSpp}-Modell geschätzt wurden. So zeigte die Abbildung 6-19, dass das diskriminativ trainierte MLN_{AbvSpp}-Modell positive Pixel im Mittel mit einer Konsistenzwahrscheinlichkeit von 0,736 schätzte, während das generativ trainierte MLN_{AbvSpp}-Modell im Mittel mit einer Konsistenzwahrscheinlichkeit von 0,526 positive Pixel schätzte.

Die vierten und fünften Spalten der Tabelle 6-8 zeigten jeweils die mittleren *User*- und Gesamtinferenzzeiten der diskriminativ trainierten MLN-Modelle. Auch hier erklärten sich die Unterschiede der Inferenzzeiten zwischen den MLN-Modellen hauptsächlich durch die Komplexität der MLN-Modelle. Ein einfacheres MLN-Modell wie MLN_{Spp} war mit 10 s *User*-Inferenzzeit fast zweimal schneller als das komplexere MLN_{AbvSpp}-Modell. Die mittleren Gesamtinferenzzeiten waren bis zu sechsmal größer als die *User*-Inferenzzeiten. Der Grund dafür war hier, wie bei den generativ trainierten MLN-Modellen, die Inferenzprozesse, die auf Ressourcen warten mussten. Die *User*-Zeit der diskriminativ trainierten MLN-Modelle war bis zu 24 % höher als die der generativ trainierten MLN-Modelle, obwohl die Inferenz mit dem gleichen Input und den gleichen Algorithmen ausgeführt wurde. Die Unterschiede der gelernten Gewichtungen der MLN-Modelle könnten die Unterschiede der Inferenzzeiten erklären. Analog waren die Gesamtinferenzzeiten der diskriminativ trainierten MLN-Modelle bis zu 52 % größer als die der generativ trainierten MLN-Modelle.

Tabelle 6-8: Evaluationsergebnisse der diskriminativ trainierten MLN-Modelle aus der Tabelle 6-6 anhand der mittleren Laufzeit, der AUC-ROC- und GDHM-Metriken. Hier wurde der DCS-Validierungsdatensatz verwendet.

Name	AUC-ROC	GDHM	Mittlere User-Laufzeit (s)	Mittlere Gesamt-laufzeit (s)
MLN_{Spp}	0,647	0,697	**10,088**	**35,588**
MLN_{Ins}	**0,651**	**0,717**	11,049	35,739
MLN_{AbvFAI}	0,588	0,64	18,950	51,176
MLN_{AbvIns}	0,595	0,693	24,204	60,685
MLN_{AbvFAS}	0,526	0,542	20,449	51,953
MLN_{AbvSpp}	0,498	0,62	23,561	59,834

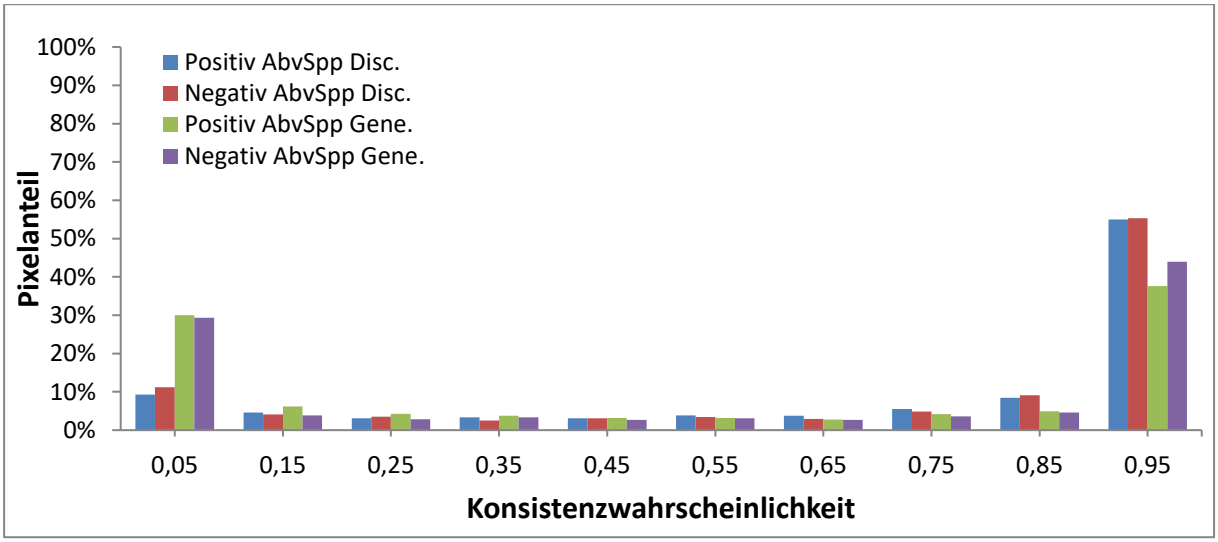


Abbildung 6-19: Histogramm der Konsistenzwahrscheinlichkeiten für die vom FCN-8s-Modell richtig (positiv) und falsch segmentierten (negativen) Pixel auf dem DCS-Validierungsdatensatz. Die Konsistenzwahrscheinlichkeiten wurden mit dem generativ (Gene.) und diskriminativ (Disc.) trainierten MLN-Modell MLN_{AbvSpp} aus der Tabelle 6-6 geschätzt.

6.4.5.3 Evaluation unter Beachtung der Erreichbarkeit von Pixeln

Analog zu der der Evaluation der semantischen Segmentierung wurden die trainierten MLN-Modelle unter Beachtung der Erreichbarkeit von Pixeln evaluiert. Dabei wurde die Erreichbarkeit anhand des Verfahrens aus dem Abschnitt 5.6.2.1 berechnet. Die Tabelle 6-9 stellt die Evaluationsergebnisse der generativ und diskriminativ trainierten MLN-Modelle anhand der mittleren *AUC-ROC*- und *GDHM*-Metriken für die erreichbaren Pixel auf dem *DCS*-Validierungsdatensatz dar. Die Erreichbarkeitsmenge des Ego-Fahrzeugs wurde mit $\mathrm{t} = 3\mathrm{s}$, $\mathrm{v} = 50\,\mathrm{km/h}$, $\psi = 90°$ und $\mathrm{a} = 4\,\mathrm{m/s^2}$ berechnet. Diese Tabelle zeigte, dass bis auf die Modelle MLN_{AbvFAS} und MLN_{AbvSpp} alle andere Modelle relative Änderungen der *AUC*- und *GDHM*-Werte unter 5,5 % aufwiesen zwischen dem Fall, in dem alle Pixel evaluiert wurden und dem Fall, wo die Evaluation sich auf erreichbare Pixel beschränkte. Dies deutete daraufhin, dass diese MLN-Modelle unabhängig von der Pixelerreichbarkeit die Konsistenzwahrscheinlichkeiten schätzen. Das Modell MLN_{AbvFAS}, das diskriminativ trainiert wurde, zeigte eine Verbesserung der *AUC*-Metrik von 16,3 % und der *GDHM*-Metrik von 23,1 %, wenn nur erreichbare Pixel evaluiert wurden. Das generativ trainierte MLN_{AbvFAS}-Modell zeigte ähnliche Ergebnisse. Analog erreichte das generativ trainierte Modell MLN_{AbvSpp} Verbesserung der *AUC* von 13,6 % und der *GDHM* von 12,5 %, wenn nur erreichbare Pixel evaluiert wurden. Die Modelle MLN_{AbvFAS} und MLN_{AbvSpp} zeigten damit eine Abhängigkeit zu den erreichbaren Pixeln.

Tabelle 6-9: Evaluationsergebnisse der generativ und diskriminativ trainierten MLN-Modelle aus der Tabelle 6-6 anhand der mittleren AUC-ROC- und GDHM-Metriken für die erreichbaren Pixel. Die Erreichbarkeitsmenge des Ego-Fahrzeugs wurde mit $t = 3s$, $v = 50\,km/h$, $\psi = 90°$ und $a = 4\,m/s^2$ berechnet. Hier wurde der DCS-Validierungsdatensatz verwendet.

Name	AUC-ROC Generativ	GDHM Generativ	AUC-ROC Diskriminativ	GDHM Diskriminativ
$\boldsymbol{MLN_{Spp}}$	**0,664**	**0,727**	**0,645**	0,70
$\boldsymbol{MLN_{Ins}}$	0,617	0,723	0,642	**0,713**
$\boldsymbol{MLN_{AbvFAI}}$	0,607	0,656	0,618	0,654
$\boldsymbol{MLN_{AbvIns}}$	0,577	0,625	0,603	0,676
$\boldsymbol{MLN_{AbvFAS}}$	0,589	0,645	0,612	0,648
$\boldsymbol{MLN_{AbvSpp}}$	0,535	0,565	0,568	0,674

6.4.6 Qualitative Evaluation der trainierten MLN-Modelle

Die Abbildung 6-20 stellt exemplarisch ein paar Ergebnisse der Konsistenzschätzung anhand von drei Szenen aus dem *DCS*-Validierungsdatensatz dar. Die erste und die zweite Zeile enthalten jeweils das Inputbild und die Annotation der semantischen Segmentierung. Die Ergebnisse der generativ trainierten MLN-Modelle MLN_{Spp}, MLN_{AbvFAS} und MLN_{AbvSpp} sind jeweils in der dritten, vierten und fünften Zeile mit der semantischen Segmentierung aus dem *FCN-8s*-Modell (siehe Abschnitt 5.4) zu finden. Dort sind Regionen mit einer Konsistenzwahrscheinlichkeit kleiner 0,45 mit Vierecken versehen. Je dunkler die Farbe des Vierecks ist, desto kleiner ist die Konsistenzwahrscheinlichkeit dieser Region. Diese MLN-Modelle wurden in der Tabelle 6-7 evaluiert und sind repräsentativ für alle andere MLN-Modelle zur Konsistenzschätzung, da diese jeweils die besten (MLN_{Spp}), die mittleren (MLN_{AbvFAS}) und die schlechtesten (MLN_{AbvSpp}) Evaluationsergebnisse lieferten (siehe Abschnitt 6.4.5).

Die erste Spalte der Abbildung 6-20 stellte eine Szene dar, in der die Straße aufgrund einer Baustelle auf der linken Seite gesperrt war. Obwohl das *FCN-8s*-Modell diese Szene auch im Baustellenbereich gut segmentieren konnte, entstanden wegen der Baustelle räumliche Relationen zwischen Regionen, die in den Trainingsdaten selten vorkamen. So wurden fast alle richtig segmentierten Regionen im Baustellenbereich als inkonsistent geschätzt, da ihre Konsistenzwahrscheinlichkeiten kleiner als der Schellenwert von 0,45 waren. So wurde z. B. die Relation $Support$ zwischen der Region der Klasse *road* und den Regionen der Klasse *fence* ($Support(road, fence)$) sowie *traffic sign* ($Support(road, traffic\ sign)$) als inkonsistent geschätzt. Die Abbildung 6-13 zeigte, dass diese Relationen mit den Häufigkeitswerten $\pi^s_{spp}(0{,}4) = 6{,}9\ \%$ und $\pi^s_{spp}(0{,}7) = 4{,}7\ \%$ sehr selten in den Trainingsdaten vorkamen. So wurden die Konsistenzwahrscheinlichkeiten der Regionen der Klassen *fence* und *traffic sign* im linken Bildbereich vom MLN_{Spp}-Modell unter 0,45 geschätzt (siehe dritte Zeile, erste Spalte der Abbildung 6-20). Die rechte Seite dieser Szene entsprach einer räumlichen Konfiguration, die häufig in den Trainingsdaten vorkam (z. B. $Support(road, car)$ und $Support(car, building)$). So wurden auf dieser Seite richtig segmentierte Regionen als konsistent geschätzt.

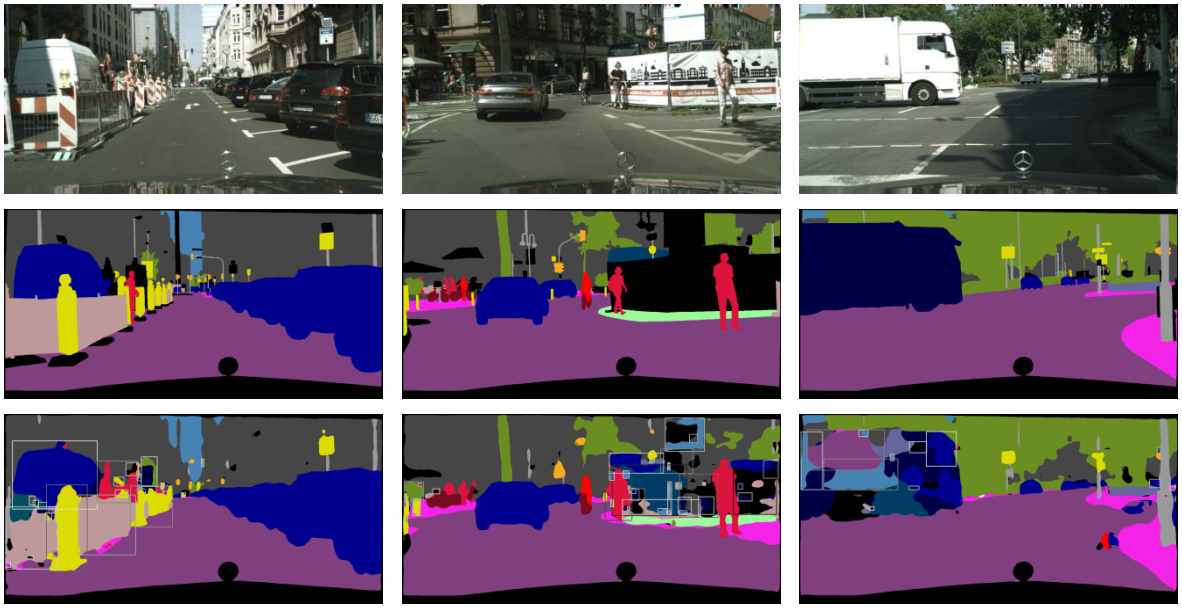
Die Szene in der zweiten Spalte der Abbildung 6-20 enthielt auf der rechten Seite eine Baustelle, die mit einem Zaun abgesperrt wurde. Auf dem Zaun wurde eine Landschaft gemalt. Diese Baustelle stellte eine nicht eindeutig zu segmentierende Region dar und wurde deswegen in den annotierten Daten der Klasse *unknow* zugewiesen. Die semantische Segmentierung lieferte in diesem Baustellenbereich jedoch falsche Ergebnisse. Die räumlichen Relationen zwischen den falsch segmentierten Regionen waren Relationen, die selten in den Trainingsdaten vorkamen. So wurde z. B. die Relation $Support$ zwischen den Regionen der Klasse b*us* und *sky* ($Support(bus, sky)$) sowie den Regionen

der Klassen *person* und *sky* ($Support(person, sky)$) geschätzt. Die Abbildung 6-13 zeigte, dass diese Relationen mit den Häufigkeitswerten $\pi^{s}_{spp}(15{,}10) = 4{,}3\ \%$ und $\pi^{s}_{spp}(11{,}10) = 0{,}8\ \%$ sehr selten in den Trainingsdaten vorkamen. So wurden auch fast alle Regionen im Baustellenbereich vom MLN_{Spp}-Modell als inkonsistent geschätzt (siehe dritte Zeile, zweite Spalte der Abbildung 6-20). Das MLN-Modell MLN_{Spp} konnte also für diese Szene eine Abhängigkeit zwischen falsch segmentierten und inkonsistenten Regionen inferieren.

Eine typische Schwäche des *FCN-8s*-Modells war, Regionen mit breiten homogenen Flächen sowie Klassen, die selten in den Trainingsdaten vorkamen, richtig zu segmentieren. Der LKW (Klasse *truck*) im linken Bereich der Szene in der dritten Spalte der Abbildung 6-20 stellte so eine breite homogene Fläche dar. Dazu kamen Pixel der Klasse *truck* in den Trainingsdaten selten vor. So wurde dieser LKW vom *FCN-8s*-Modell zum Teil als *sky* und *road* segmentiert. Die räumliche Relation $Support$ zwischen den falsch segmentierten Regionen der Klassen *sky* und *road* kam mit einer Häufigkeit von $\pi^{s}_{spp}(10{,}0) = 0\ \%$ nie in den Trainingsdaten vor. Diese Regionen wurden deswegen vom MLN_{Spp}-Modell als inkonsistent geschätzt (siehe dritte Zeile, dritte Spalte der Abbildung 6-20). Hier konnte auch das Modell MLN_{Spp} wie erwünscht die Abhängigkeit zwischen falsch segmentierten und inkonsistenten Regionen herleiten.

Die Ergebnisse des Modells MLN_{AbvFAS} in der vierten Zeile der Abbildung 6-20 zeigten, dass dieses Modell auch in der Lage war, analog zu dem MLN_{Spp}-Modell falsche Segmentierungen als inkonsistent zu schätzen. Allerdings hatte das MLN_{AbvFAS}-Modell die Tendenz, mehr richtig segmentierte Regionen als inkonsistent zu klassifizieren als das Modell MLN_{Spp}. So war das Modell MLN_{AbvFAS} bezogen auf die Evaluation im Abschnitt 6.4.5 schlechter als das MLN_{Spp}-Modell.

Die Ergebnisse des Modells MLN_{AbvSpp}, das die räumliche Relationen der Modelle MLN_{AbvFAS} und MLN_{Spp} kombiniert, waren in der fünften Zeile der Abbildung 6-20 zu finden. Diese Ergebnisse waren sehr ähnlich den Ergebnissen des Modells MLN_{AbvFAS}. Owbwohl viele falsche Segmentierungen als inkonsistent vom MLN_{AbvSpp}-Modell geschätzt wurden, wurden auch richtig segmentierte Regionen als inkonsistent klassifiziert. Dies erklärte die schlechten Evaluationsergebnisse des Modells MLN_{AbvSpp} im Abschnitt 6.4.5.



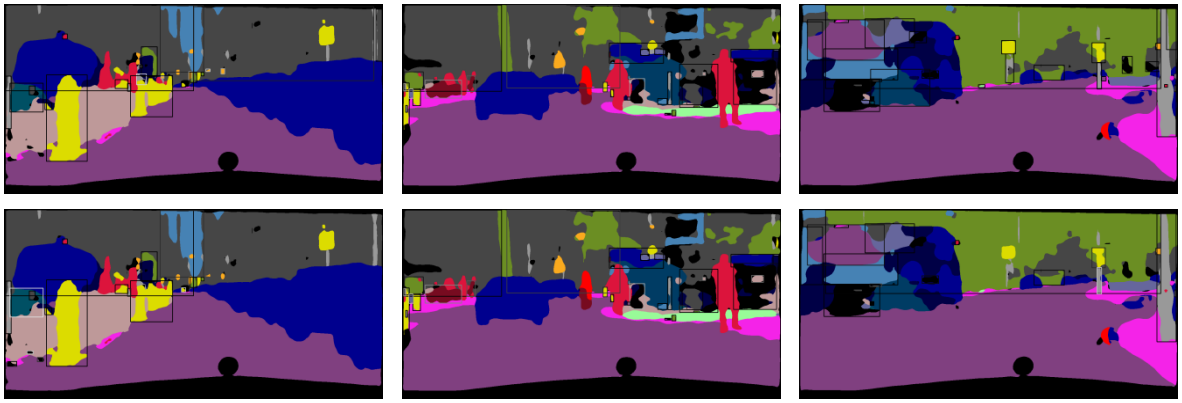


Abbildung 6-20: Beispielergebnisse der Konsistenzschätzung mithilfe der semantischen Segmentierung für ein paar Bilder aus dem DCS-Validierungsdatensatz. Die Zeilen enthalten jeweils: das Inputbild (erste Zeile), die Annotation der semantischen Segmentierung (zweite Zeile), die Segmentierung mit dem FCN-8s-Modell überlagert mit Vierecken um Regionen mit einer Konsistenzwahrscheinlichkeit kleiner 0,45 für die generativ trainierten MLN-Modelle MLN_{Spp} (dritte Zeile), MLN_{AbvFAS} (vierte Zeile) und MLN_{AbvSpp} (fünfte Zeile). Die Inputbilder und die Annotation der semantischen Segmentierung stammen aus dem DCS-Datensatz [81].

6.5 Diskussion

In diesem Abschnitt werden die Ergebnisse des vorgeschlagenen Systems zur Schätzung der Konsistenz von semantisch segmentierten Regionen bewertet. Darüber hinaus werden mögliche Lösungen zur Verbesserung des vorgeschlagenen Systems benannt.

6.5.1 Wissensbasiertes Modul

6.5.1.1 Modellierung der WB

Zur Modellierung der WB wurden eine Ontologie und logische Regeln verwendet. Der Vorteil der WB lag darin, dass das Wissen anhand von Symbolen hierarchisch modelliert wurde. Somit war die WB leicht zu verstehen und zu validieren. Obwohl die WB manuell modelliert wurde, war der Modellierungsaufwand minimal, da das zu modellierende Wissen auf das Wesentliche reduzierte wurde.

6.5.1.2 Inferenz der WB

Die Inferenz der WB anhand von zwei Beispielen zeigte, dass die Ontologie konsistent und korrekt modelliert wurde. Allerdings waren die *SWRL*-Regeln nicht geeignet, um die Konsistenz von Regionen zu schätzen, da diese Regeln keine Unsicherheiten enthielten. Diese Unsicherheiten waren entscheidend, um die Konsistenz von Regionen zu modellieren. Die Inferenzzeit lag bei ca. 140 ms. Damit war die Inferenz nicht echtzeitfähig, da die Inferenzzeit über 100 ms lag, was oft als Echtzeitanforderung für Onlineanwendungen im Kontext des automatisierten Fahrens vorgegeben wird. Allerdings stellte die Überschreitung der maximalen Inferenzzeit in dieser Arbeit kein wesentliches Problem dar, da die Inferenz der WB hier nur offline erfolgte. Der Grund, warum die Inferenz der WB hier nur offline erfolgte, war, dass die Unsicherheiten, die verwendet wurden, um die Konsistenz von Regionen der semantischen Segmentierung zu schätzen, mit PGM außerhalb der WB modelliert wurden. Lediglich diese PGM wurden online verwendet, um die Konsistenz von semantisch segmentierten Regionen zu schätzen. Obwohl die WB in dieser Arbeit nur offline verwendet wurde, war es jedoch für diese Arbeit keine Einschränkung, da die PGM, die für die Konsistenzschätzung von Regionen online verwendet wurden, auf der WB basierten. Somit wurde sichergestellt, dass die PGM zur Konsistenzschätzung von Regionen bezogen auf die WB korrekt und konsistent waren.

Für Anwendungen, die die Online-Inferenz der WB benötigen, könnten zukünftige Arbeiten zur Verringerung der Inferenzzeit relevant werden.

6.5.2 MLN zur Schätzung der Konsistenzwahrscheinlichkeiten von semantisch segmentierten Regionen

6.5.2.1 Modellierung

Zur Schätzung der Konsistenzwahrscheinlichkeiten von semantisch segmentierten Regionen wurde die WB mit MLN erweitert, um Unsicherheiten zu betrachten. Die logischen Regeln der MLN basierten auf der modellierten WB. Damit wurde die Korrektheit der MLN-Modelle bezogen auf die WB sichergestellt. Die Modellierung der MLN-*FOL*-Regeln war mit minimalem Aufwand verbunden, da die zugrundliegende WB eine geringere Komplexität hatte. Darüber hinaus stellte das *Alchemy*-Tool eine kompakte Syntax zur Vereinfachung der Modellierung von MLN-*FOL*-Regeln bereit.

6.5.2.2 Training

Der Aufwand bei der Generierung von Trainingsdaten war minimal, da die Annotationen der semantischen Segmentierung des *DCS*-Trainingsdatensatzes verwendet wurden. Allerdings dauerte laut der Autoren des *DCS*-Datensatzes die Generierung von Annotationen der semantischen Segmentierungen für die gesamten 5000 Bilder ca. 312,5 Tage [81]. Der hohe Aufwand bei der Generierung der Annotationen der semantischen Segmentierungen lag daran, dass einzelne Pixel annotiert wurden. Für MLN-Modelle würde jedoch reichen, wenn menschliche Experten grob die Häufigkeitsmatrizen π_r der Gleichung (47) schätzen würden. Mit diesen grob geschätzten Häufigkeitsmatrizen ließen sich die Trainingsdaten für MLN-Modelle mit geringerem Aufwand generieren.

Die Trainingsdauer der MLN-Modelle lag zwischen drei und 204 Minuten und hing von der Anzahl der Trainingsdaten, der Trainingsmethode und der Komplexität der MLN-*FOL*-Regeln ab. Darüber hinaus spielten die Trainingsarten diskriminativ und generativ eine Rolle bei der Trainingsdauer. Das diskriminative Lernen dauerte länger als das generative Lernen. Die Dauer des Trainingsprozesses war mit maximal 204 Minuten in einigen Fällen nicht zu vernachlässigen. Verfahren zur Verringerung dieser Trainingszeit sollten in der Zukunft erörtert werden.

Die trainierten MLN-Modelle durften nicht die kompakte Syntax der Modellierung von MLN-*FOL*-Regeln verwenden. Deshalb stieg die Anzahl der Klauseln der trainierten MLN-Modelle exponentiell mit der Anzahl der semantischen Klassen und der räumlichen Relationen. Jedoch hatten die MLN-Modelle den Vorteil, dass die gelernten Gewichtungen der *FOL*-Regeln mit geringem Aufwand interpretierbar waren. Somit konnten die von MLN-Modellen geschätzten Konsistenzwahrscheinlichkeit im Vergleich zu dem PGM-Modell aus [116] besser interpretiert werden. Dazu integrierten MLN-Modelle Unsicherheiten, die in [117] fehlten. Zukünftige Arbeiten zum Editieren von komplexeren MLN-Modellen wären hilfreich (siehe bspw. Aguiar et al. [205]).

6.5.2.3 Inferenz

Die Evaluation der Inferenzergebnisse der MLN-Modelle zeigten anhand der Histogramme der Konsistenzwahrscheinlichkeiten, dass die vom *FCN-8s*-Modell falsch segmentierten Regionen öfter inkonsistent waren, als die vom *FCN-8s*-Modell richtig segmentierten Regionen. Darüber hinaus wurden richtig segmentierte Regionen, die aber in einer Szene waren, die selten in den Trainingsdaten vorkam, als inkonsistent geschätzt. Dazu wurde gezeigt, dass die Inferenzergebnisse der MLN-Modelle einfach zu erklären waren. Damit stellten die MLN-Modelle eine geeignete Lösung dar, um das *FCN-8s*-Modell zu überwachen, da das *FCN-8s*-Modell wie alle TNN-Modelle eine *Black Box* war und die Segmentierung des *FCN-8s*-Modells bei seltenen oder neuen Szenen unvorhersehbar blieb.

Die *UAC-ROC*–Werte der diskriminativ und generativ trainierten Modelle unterschieden sich lediglich bei den Modellen mit den schlechtesten *UAC-ROC*–Werten. Bei den besten Modellen mit den besten *UAC-ROC*-Werten gaben keine Unterschiede zwischen dem diskriminativen und generativen Training. Die *UAC-ROC*–Werte einiger MLN-Modelle waren zwar größer als der *UAC-ROC*–Wert des *Random-Guess*-Klassifikators von 50 %, aber entfernt von dem optimalen *UAC-ROC*–Wert von 100 %. Ein Grund dafür war die Tatsache, dass einige Regionen, die vom *FCN-8s*-Modell richtig segmentiert wurden, als inkonsistent geschätzt wurden. Manche dieser Regionen waren in der Tat inkonsistent,

da diese Regionen in räumlichen Relationen auftauchten, die selten in den Trainingsdaten vorkamen. Darüber hinaus wurden einige falsch segmentierte Regionen als konsistent geschätzt. Es war daher wichtig, inkonsistente Regionen zu speichern. Ein menschlicher Experte sollte dann entscheiden, welche der inkonsistenten Regionen richtig oder falsch segmentiert waren und das *FCN-8s*- sowie die MLN-Modelle entsprechend anpassen. Ein weiterer Grund für die kleinen *UAC-ROC*–Werte war der große Unterschied zwischen den Klassenmaxima der Matrizen in der Abbildung 6-6 bis Abbildung 6-10. Diese Unterschiede auszugleichen, könnten in der Zukunft die *UAC-ROC*–Werte der MLN-Modelle verbessert. Zukünftige Arbeiten können sehr wahrscheinlich durch die Integration weiterer Kontextinformationen (z. B. zeitliche Konsistenz, Tiefe, globaler Kontext etc.) die Schätzungsgüte der MLN-Modelle verbessern. Dabei sollten aber auch Verfahren entwickelt werden, die trotz der Integration weiterer Kontextinformationen die Komplexität des Systems handhabbar halten.

Die Inferenzzeiten der MLN-Modelle lagen zwischen 10 s und 42 s. Somit waren alle MLN-Modelle weit entfernt von der optimalen Inferenzzeit von 100 ms für Onlineanwendungen im Kontext des automatisierten Fahrens. Von verschiedenen Autoren [206–211] wurden zwar Ansätze und Tools vorgeschlagen, um die Inferenzzeiten von MLN-Modellen zu reduzieren, aber die Inferenzzeiten waren immer noch größer 100 ms. Solche Ansätze sollten in der Zukunft weiterentwickelt werden.

6.6 Zusammenfassung

In diesem Kapitel wurde die Schätzung der Konsistenz von Regionen aus der semantischen Segmentierung adressiert. Dafür wurden Regionen als konsistent definiert, wenn diese Regionen das Vorwissen erfüllten. Regionen, die das Vorwissen verletzen, wurden entsprechend als inkonsistent festgelegt Die Konsistenzschätzung fand bis jetzt sehr wenig Anklang in der Literatur. Jedoch mit dem Erfolg von TNN bei der semantischen Segmentierung und den Schwierigkeiten bei der expliziten Integration von Vorwissen in TNN, stellte die Konsistenzschätzung ein gutes Mittel dar, um zu prüfen, ob TNN bei der Inferenz das Vorwissen betrachteten. Ferner sollte anhand der Konsistenzschätzung untersucht werden, ob es eine Korrelation zwischen inkonsistenten Regionen und Regionen, die von TNN-Modellen falsch segmentiert wurden, gibt.

6.6.1 Lösungsansatz

Zur Lösung der Aufgabe der Konsistenzschätzung wurde das in Kapitel 4 vorgestellte Konzept umgesetzt. Der Lösungsansatz bestand aus zwei Modulen: ein wissensbasiertes Modul und ein probabilistisches Modul. Der Vorteil dieses Lösungsansatzes im Vergleich zu den Ansätzen der Literatur lag zum einem darin, dass das Vorwissen explizit und formal modelliert wurde. So konnten die Inferenzergebnisse mit dem Vorwissen besser nachvollzogen und das Vorwissens leichter angepasst werden. Zum anderen konnten durch die Verwendung von probabilistischen Modellen Unsicherheiten in das Vorwissen modelliert werden, was zu einer besseren Inferenz führte. Durch die symbolische Modellierung der WB, die explizite Trennung des Vorwissens von TNN und den modularen Aufbau des Systems ließen sich Teile des Systems mit geringerem Aufwand erweitern.

6.6.2 Wissensbasiertes Modul

Das wissensbasierte Modul bestand aus einer Ontologie und einer Regelbasis. Die Ontologie enthielt das Wissen über die Taxonomie der relevanten Begriffe der Domänen. Diese Taxonomie wurde anhand von $\mathcal{T}Box$-Axiomen modelliert. Die Begriffe der Ontologie waren sowohl physikalische Szenenelemente als auch weitere ab-strakte Elemente. Die physikalischen Szenenelemente stammen aus dem *DCS*-Datensatz. Ferner modellierte die Ontologie die Eigenschaften der Relationen der Domänen anhand von $RBox$-Axiomen, wobei die Relationen Kontextabhängigkeiten der Begriffe repräsentierten. Kontextabhängigkeiten enthielten räumliche und Skalierungsabhängigkeiten zwischen Szenenelementen. Die Regelbasis bestand aus einer Menge von *FOL*-Regeln, die die Konsistenz von Regionen modellierten, gegeben die semantischen Klassen dieser Regionen und die Relationen zwischen diesen Regionen. Trotz der manuellen Modellierung der WB war der Modellierungsaufwand minimal, da das zu modellierende Wissen auf das Wesentliche reduziert wurde. Gleichzeitig wurde

die Handhabung der Ontologie gewährleistet. Die Modellierung der WB fand in der Offlinephase statt.

Die modellierte WB wurde anhand eines *Reasoner*s inferiert. Der *Reasoner* prüfte für gegebene Evidenzen vor allem die Konsistenz der WB und die Erfüllbarkeit von Formeln der Regelbasis. Damit wurde sichergestellt, dass das modellierte Vorwissen konsistent war. Die Regelbasis war aufgrund von fehlenden Unsicherheiten nicht in der Lage, die Konsistenz von semantisch segmentierten Regionen zu inferieren. Deshalb wurde die WB in die Klassen von PGM namens MLN überführt und die WB nur offline für die Modellierung verwendet. Die Inferenzzeit war mit 140 ms größer als die Inferenzzeit von 100 ms, die oft als obere Grenze für Onlineanwendungen im Kontext des automatisierten Fahrens vorgegeben wird. Jedoch wurde die WB in dieser Arbeit nur offline verwendet. So stellte diese Überschreitung der Inferenzzeit kein wesentliches Problem dar.

6.6.3 Probabilistisches Modul

Da die WB aufgrund der fehlenden Unsicherheiten nicht in der Lage war, die Konsistenz von Regionen zu schätzen, wurden MLN als Erweiterung der WB verwendet. Alle Formeln der MLN-Modelle wurden aus der modellierten WB extrahiert und syntaktisch angepasst. So wurde sichergestellt, dass die logischen Formeln der MLN-Modelle konsistent waren, da die modellierte WB konsistent war. Die Modellierung der Regeln der MLN-Modelle war mit einem minimalen Aufwand verbunden, da die zugrundliegende WB eine geringere Komplexität hatte. Dazu stellte das Modellierungstool syntaktische Formalismen zur kompakten Modellierung von logischen Formeln bereit. Die Generierung von Formeln der MLN-Modelle passierte in der Offlinephase.

Zum Lernen der Gewichtungen der Formeln der MLN-Modelle wurden in der Offlinephase die Annotationen der semantischen Segmentierung des *DCS*-Trainingsdaten-satzes verwendet, da diese annotierten Daten für das Training des TNN-Modells zur semantischen Segmentierung verwendete und somit eine repräsentative Stichprobe des Vorwissen enthielt. Zur Generierung der Trainingssamples wurden zunächst die Wahrscheinlichkeitsverteilungen der in der WB modellierten räumlichen Relation zwischen Regionen in den annotierten Daten gelernt. Mit der Annahme, dass Regionen konsistenter waren, wenn die räumliche Relation zwischen diesen Regionen öfter in den annotierten Daten der semantischen Segmentierung vorkam, wurden die Trainingssamples der MLN-Modelle generiert. Basierend auf diesen Trainingssamples wurden die Gewichtungen der MLN-*FOL*-Formeln diskriminativ und generativ gelernt. Der Aufwand der Generierung von Trainingsdaten war aufgrund der frei verfügbaren Annotationen der semantischen Segmentierung des *DCS*-Trainingsdatensatzes überschaubar. In dem Fall, wo solche annotierten Daten nicht verfügbar wären, würden sich die Trainingsdaten von Experten mit wenigen Bemühungen generieren lassen können. Die Trainingszeit lag zwischen drei und 204 Minuten und hing von der Komplexität der Formeln der MLN-Modelle, von der Anzahl der Trainingssamples sowie von der Trainingsmethode ab. In einigen Fällen waren die Trainingszeiten also nicht zu vernachlässigen. Verfahren zur Reduzierung dieser Trainingszeiten sollten in der Zukunft erforscht werden. Die trainierten MLN-Modelle konnten nicht die syntaktischen Formalismen der Modellierung von MLN nutzen und hatten deutlich mehr Klauseln als die modellierten MLN-*FOL*-Regeln. Allerdings war die Erhöhung der Klauselanzahl notwendig, um die trainierten MLN-Modelle zu interpretieren. So zeigten die Vergleiche der gelernten Gewichtung der MLN-*FOL*-Formeln, dass diese Gewichtungen bezogen auf die Trainingsdaten konsistent waren.

Die gelernten MLN-Modelle wurden in der Onlinephase verwendet, um die Konsistenzwahrscheinlichkeiten der vom *FCN-8s*-Modell segmentierten Regionen auf dem *DCS*-Validierungsdatensatz zu schätzen. Die Evaluationsergebnisse zeigten, dass die vom *FCN-8s*-Modell falsch segmentierten Regionen inkonsistenter waren als die richtig segmentierten Regionen. Ferner konnten die MLN-Modelle Inkonsistenzen in den Regionen erkennen, die zwar vom *FCN-8s*-Modell richtig segmentiert wurden, aber in seltenen Szenen wie bspw. Baustellen auftraten. Mit diesen Ergebnissen wurde das wesentliche Ziel dieses Kapitels erreicht. Allerdings wurden auch die Konsistenzwahrscheinlichkeiten einiger Regionen aufgrund von fehlenden Kontextabhängigkeiten falsch geschätzt. Die Integration von weiteren Kontextabhängigkeiten würde sicherlich diese falschen Schätzungen verbessern. Die gespei-

cherten inkonsistenten Regionen könnten bei der Auswahl der zu integrierenden Kontextabhängigkeiten eine wesentliche Rolle spielen. Es sollte jedoch auf die Komplexität der MLN bei der Integration von weiteren Kontextinformationen geachtet werden. Der Hauptnachteil bei der Inferenz der MLN-Modelle war die Inferenzzeit. Diese lag im Mittel zwischen 10 s und 42 s. Alle MLN-Modelle dieses Kapitels lagen damit weit über der optimalen Inferenzzeit von 100 ms. Einige Arbeiten zur Verbesserung der Inferenzzeit wurden vorgeschlagen und sollten in der Zukunft intensiv erforscht werden.

7 Kontextabhängige Prädiktion der zeitlichen Situationsentwicklung

Eine wichtige Aufgabe der Situationsinterpretation ist neben der Schätzung der Konsistenz von semantisch segmentierten Regionen, die in Kapitel 6 vorgestellt wurde, die Prädiktion der Situationsentwicklung über die Zeit. Dafür werden vor allem die Existenz und das Verhalten von Verkehrsteilnehmern über die Zeit prädiziert. Diese Prädiktion ist hilfreich, um potenzielle Konflikte zwischen dem Ego-Fahrzeug und anderen Verkehrsteilnehmern zu erkennen und Strategien zu entwickeln, um diese Konflikte zu vermeiden.

Im Abschnitt 3.6.4 wurden Ansätze aus der Literatur vorgestellt, die zur zeitlichen Prädiktion der Situationsentwicklung vorgeschlagen wurden. Einige dieser Ansätze verwendeten Manöver, um das mittelfristige Verhalten von Verkehrsteilnehmern vorherzusagen. Die Parameter der Funktion zur Vorhersage wurden oft anhand von Trainingsdaten gelernt. Der Nachteil solcher Ansätze lag darin, dass seltene Ergebnisse nicht gelernt werden konnten, da diese in den Trainingsdaten kaum repräsentiert waren. Darüber hinaus benötigte die Vorhersagefunktion in einigen Situationen allgemeines Vorwissen, das allein anhand der Trainingsdaten nicht gelernt werden kann.

In diesem Kapitel wird die Anwendung des in Kapitel 4 vorgestellten Konzepts zur Prädiktion der Existenz sowie des Verhaltens von Verkehrsteilnehmern vorgestellt. Der Fokus wird auf seltene Situationen mit starkem Bezug zum allgemeinen Wissen gelegt. Im ersten Abschnitt wird die Aufgabe dieses Kapitels illustriert. Der zweite Abschnitt stellt den Lösungsansatz basierend auf dem Konzept des Kapitels 4 dar. In den Abschnitten drei und vier werden die Modellierung, die Implementierung und die Evaluation der Module des Lösungsansatzes veranschaulicht. Der Abschnitt fünf diskutiert die Ergebnisse des implementierten Systems. Der sechste Abschnitt fasst dieses Kapitel zusammen.

7.1 Vorstellung der Aufgabe

Gegeben sei ein Situationsmodell S_t^M zum Zeitpunkt t. Dieses Situationsmodell enthält eine Menge O_t von Verkehrsteilnehmern. Gesucht ist eine Funktion $O_{t+n} = f_\theta(O_t)$, die Menge O_{t+n} von Verkehrsteilnehmern zum Zeitpunkt $t + n$ prädiziert. Die Menge O_{t+n} enthält sowohl die Existenz als auch das Verhalten von Verkehrsteilnehmern zum Zeitpunkt $t + n$.

7.2 Lösungsansatz

Im Abschnitt 3.6.4 wurden Ansätze aus der Literatur vorgestellt, um das Verhalten von Verkehrsteilnehmern über die Zeit zu prädizieren. Manöverbasierte Ansätze für eine mittelfristige Vorhersage zeigten die besten Ergebnisse. Die Schätzung der Parameter der Prädiktionsfunktion wurde aufgrund der Komplexität von Situationen anhand von Trainingsdaten gelernt. Um mit seltenen Situationen umzugehen, wurden diese seltenen Situationen künstlich generiert und in die Trainingsdaten integriert. Der Nachteil dieses Verfahrens war der Aufwand bei der künstlichen Generierung von Situationen. Darüber hinaus könnte die künstliche Generierung von Situationen zum Fehlverhalten der gelernten Prädiktionsfunktion führen. Die Prädiktionsfunktion könnte dazu tendieren, künstlich generierte Situationen öfter zu prädizieren, als diese Situationen in der Realität auftreten würden. Ein weiterer Nachteil ist, dass einige seltenen Situationen sich nur mithilfe von allgemeinem Wissen prädizieren lassen, wobei dieses allgemeine Wissen anhand von Trainingsdaten schwer zu lernen ist.

Um mit seltenen Situationen umzugehen, wurde das in dieser Arbeit vorgestellte Konzept auf dem Beispiel „*Kind folgt Ball*“ angewendet. Das Beispiel „*Kind folgt Ball*“ beschreibt die seltene Situation, dass ein Kind in naher Zukunft hinter einem Ball laufen könnte, wenn dieser Ball auf die Straße rollt. Dabei kann eine potenzielle Kollision zwischen dem Kind und dem Ego-Fahrzeug entstehen. Die Abbildung 7-1 stellt die Schritte zur Prädiktion der zeitlichen Entwicklung der Situation „*Kind folgt Ball*“ dar. Zunächst wurde das Vorwissen über die Existenz und das Verhalten eines Kindes und den Ball mit einer WB bestehend aus einer Ontologie und logischen Regeln modelliert (Schritt eins der Abbildung 7-1). Die WB wurde dann mit Evidenzen inferiert, um neues Wissen zu generieren (Schritte zwei bis vier der Abbildung 7-1). Im nächsten Schritt wurde die WB als Input für die Modellierung der MLN verwendet (Schritte fünf und sechs der Abbildung 7-1). Das MLN-Modell beschrieb die Existenz und das Verhalten des Kindes in naher Zukunft, gegeben ein Ball unter Beachtung von Unsicherheiten.

Die Parameter des MLN-Modells wurden anhand von manuell generierten Daten gelernt (Schritte sieben bis neun der Abbildung 7-1). Gegeben das gelernte MLN-Modell und die detektierten Szenenobjekte (hier Ball und Kind), wurde die Existenz und das Verhalten des Kindes inferiert (Schritte zehn bis zwölf der Abbildung 7-1). TNN könnten verwendet werden, um Szenenobjekte zu detektieren. Diese wurden im Rahmen dieser Arbeit jedoch nicht adressiert. Eine ausführliche Beschreibung der Module aus der Abbildung 7-1 erfolgt in den nächsten Abschnitten. Eine frühere Version des in diesem Kapitel vorgestellten Lösungsansatzes wurde in [212] publiziert.

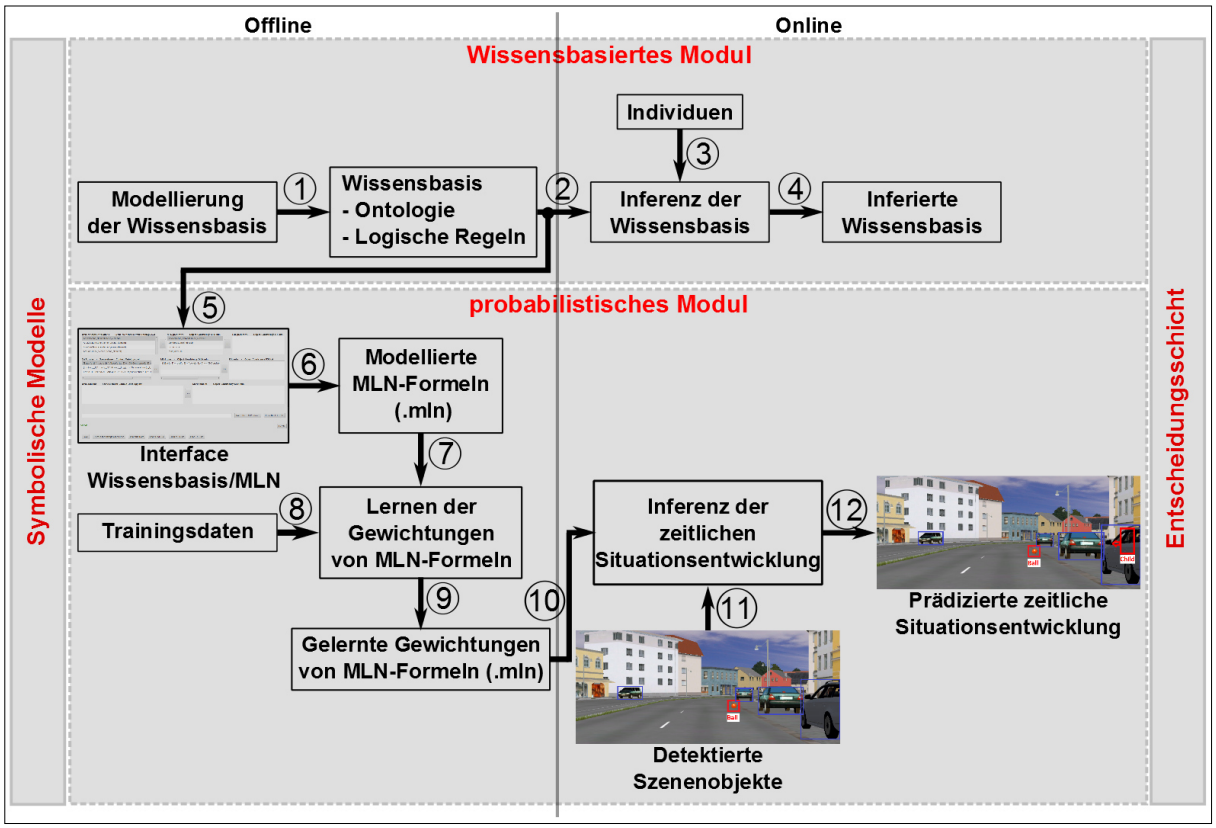


Abbildung 7-1: Konzeptuelle Sicht des Systems zur Prädiktion der Existenz und des Verhaltens von Szenenobjekten in seltenen Situationen anhand des Beispiels „Kind folgt Ball“

Der Vorteil der vorgeschlagenen Lösung lag darin, dass die WB gut geeignet war, um seltene Situationen zu modellieren, da die Existenz und das Verhalten von Szenenobjekten in seltenen Situationen meistens auf einer symbolischen Ebene einfach zu beschreiben waren. Darüber hinaus konnte das allgemeine Wissen, das für die Prädiktion der Existenz sowie des Verhaltens von Szenenobjekten relevant war, bestens in einer WB beschrieben werden. Die Erweiterung der WB mit MLN ermöglichte die Prädiktion unter Beachtung von Unsicherheiten, die in realen Situationen vorhanden waren. Die Komplexität des Gesamtsystems blieb überschaubar, da das zu modellierende Wissen gering war.

7.3 Wissensbasiertes Modul

In diesem Modul wurde zunächst das Vorwissen über die Situation „*Kind folgt Ball*“ modelliert. Danach wurde neues Wissen anhand von Evidenzen inferiert (siehe Schritte eins bis vier der Abbildung 7-1). Diese Schritte werden im Detail in den nächsten Abschnitten beschrieben.

7.3.1 Modellierung der WB

Analog zur WB des Abschnitts 6.3.1 bestand die die WB $\mathcal{WB} = \{\mathcal{O}, \mathcal{F}\}$ aus einer Ontologie $\mathcal{O}$ und einer Menge von logischen Regeln $\mathcal{F}$ (siehe Schritt eins der Abbildung 7-1). Diese WB wurde in der Offlinephase modelliert.

Die Ontologie $\mathcal{O} = \{\mathcal{B}, \mathcal{R}, \mathcal{T}Box, ABox, RBox\}$ enthielt als Begriffe die für diese Aufgaben relevanten Szenenobjekte sowie weitere abstrakte Begriffe. Diese Begriffe sind in der Menge $\mathcal{B}$ dargestellt:

$$\mathcal{B} = \{child, ball, hallucinated\_object\} \tag{55}$$

Die Begriffe *child* und *ball* standen jeweils für die Szenenobjekte der Klassen *Kind* und *Ball. hallucinated_object* war ein abstrakter Begriff zur Halluzination der Existenz eines Kindes, wenn ein Ball auf die Straße rollte.

Die Relationsmenge $\mathcal{R}$, die für diese Arbeit relevant war, wurde durch die folgende Gleichung definiert:

$$\mathcal{R} = \{follow\} \tag{56}$$

Die Relation *follow* modellierte die Tatsache, dass ein Kind in naher Zukunft einem Ball folgt.

Neben den Begriffen und Relationen wurden *TBox*- und *RBox*-Axiome verwendet, um die Ontologie zu beschreiben. Die Tabelle 7-1 stellt diese Axiome der Ontologie dar. *TBox*-Axiome beschrieben die Taxonomie zwischen den Begriffen der Ontologie. Die Begriffe *conceptual_object* und *physical_object* wurden analog zu der Taxonomie in der Tabelle 6-1 als Spezialisierungen des *Top*-Begriffs $\top$ definiert. Als weitere Elemente der Taxanomie wurden die Szenenelemente *child* und *ball* als Spezialisierungen von *physical_object* modelliert, während der Begriff *hallucinated_object* als Spezialisierung von *conceptual_object* modelliert wurde. Die *RBox*-Axiome A_5 und A_6 modellierten die Antisymmetrie und die Nichtreflexivität der Relation *follow*.

Tabelle 7-1: Beschreibung der TBox- und RBox-Axiome zur Modellierung der Ontologie zur Prädiktion der Existenz und des Verhaltens von Szenenobjekten für das Beispiel „Kind folgt Ball“

Name	Axiom	Typ	Bedeutung
A_0	$conceptual\_object \sqcup physical\_object \sqsubseteq \top$	*TBox*	Definition der Menge aller Individuen
A_1	$hallucinated\_object \sqsubseteq conceptual\_object$	*TBox*	Begriffe von konzeptuellen Objekten
A_2	$child \sqcup ball \sqsubseteq physical\_object$	*TBox*	Begriffe von physikalischen Objekten
A_3	$disjoint(child, ball)$	*TBox*	Disjunkte Begriffe
A_4	$disjoint(hallucinated\_object, ball)$	*TBox*	Disjunkte Begriffe
A_5	$follow \not\equiv follow^-$	*RBox*	Die Relation ist anti-symmetrisch
A_6	$irreflexiv(follow)$	*RBox*	Die Relation ist nicht reflexiv

Die Abbildung 7-2 stellt analog zur Abbildung 6-3 die mit *Protégé* [198] in der Beschreibungssprache *OWL*-2 [28] modellierte Taxonomie $\mathcal{T}$ der Ontologie aus der Tabelle 7-1 dar. Die Knoten dieses Graphs repräsentierten die Begriffe der Ontologie und die gerichteten Kanten modellierten die *Is-a*-Beziehung zwischen den Begriffen.

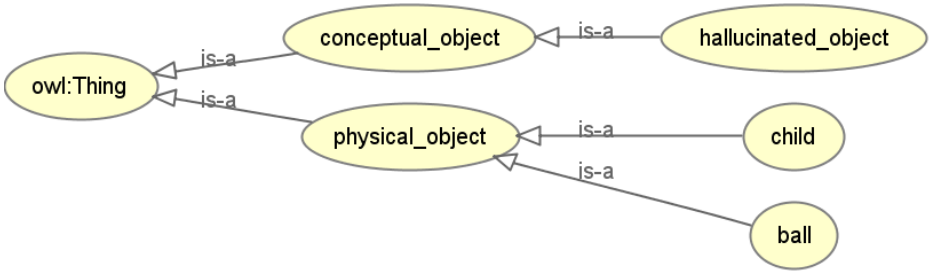


Abbildung 7-2: Taxonomie $\mathcal{T}$ der Ontologie zur Prädiktion der Existenz und des Verhaltens von Szenenobjekten für das Beispiel „Kind folgt Ball“. Die Grafik wurde mit dem OWLViz-Tool von Protégé [198] generiert.

Neben der Ontologie enthielt die WB logische Regeln. Diese Regeln modellierten das Wissen, das in der Ontologie nicht modelliert werden konnte. Hier wurden *FOL*-Formeln verwendet. Die Gleichung (57) zeigt die Menge der logischen Formel $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2\}$ zur Beschreibung der Existenz sowie des Verhaltens eines Kindes in der Situation „*Kind folgt Ball*“.

$$\begin{aligned} \mathcal{F}_1 &= \forall o_1, \forall o_2 : \big(ball(o_1) \wedge hallucinated\_object(o_2)\big) \Longrightarrow child(o_2) \\ \mathcal{F}_2 &= \forall o_1, \forall o_2 : \big(ball(o_1) \wedge child(o_2)\big) \Longrightarrow follow(o_2, o_1) \end{aligned} \tag{57}$$

Die Formel $\mathcal{F}_1$ *halluzinierte* die unmittelbare Existenz eines Kindes, wenn der Ball existierte. Mit dem Prädikat $hallucinated\_object$ war die Variable o_2 sowohl in der Vorbedingung als auch im Kopf der Formel $\mathcal{F}_1$ vorhanden. Damit wurden die Sicherheitsbedingung der Formel $\mathcal{F}_1$und die Entscheidbarkeit der Regelbasis gemäß der Syntax von *SWRL* gewährleistet [199, 213]. $\mathcal{F}_2$ beschrieb die Tatsache, dass ein Kind in naher Zukunft hinter einen Ball laufen wird, wenn der Ball und das Kind existierten. Damit wurden die zeitliche Abhängigkeit zwischen der Existenz des Kindes und des Balls zum Zeitpunkt t sowie das Verhalten des Kindes in naher Zukunft $t + n$ modelliert. Die Modellierung der logischen Regeln erfolgte auch hier in *Protégé* mit *SWRL*.

7.3.2 Inferenz der WB

Der *Pellet-Reasoner* [32], der in *Protégé* verfügbar war, wurde hier für die Inferenz verwendet. Dieser *Reasoner* wurde kurz im Abschnitt 6.5.1.2 beschrieben. Die Tabelle 7-2 stellt drei Beispiele der Inferenz für zwei Individuen i_1 und i_2 dar. Im ersten Fall (erste Zeile der Tabelle 7-2) waren die Individuen i_1 und i_2 Instanzen der Begriffe *ball* und *hallucinated_object*. In diesem Fall wurde die WB als konsistent inferiert. Dazu lieferten der Instanzstest und die Inferenz der logischen Regeln, dass i_2 eine Instanz des Begriffs *child* war und dass i_2 i_1 folgte (Prädikat $follow(i_2, i_1)$ erfüllt). Die Inferenzzeit betrug ca. 100 ms. Im zweiten Fall (zweite Zeile der Tabelle 7-2) war das Individuum i_1 Instanz des Begriffs *ball*. Ferner war i_2 Instanz der Begriffe *child* und *hallucinated_object*. In diesem Fall wurde auch die WB als konsistent inferiert. Die Inferenz der logischen Regeln ergab, dass i_2 i_1 folgte (Prädikat $follow(i_2, i_1)$ erfüllt). Die Inferenzzeit betrug ca. 20 ms. In der dritten Zeile der Tabelle 7-2 wurden die Inputs des ersten Falls mit $child(i_2)$, $follow(i_1, i_2)$ und $follow(i_1, i_2)$ erweitert. Die Inferenz ergab eine inkonsistente WB, da die Relation *follow* gemäß dem Axiom A_5 der Tabelle 7-1 als antisymmetrisch in der Ontologie definiert wurde. Somit durften die Relationen $follow(i_1, i_2)$ und $follow(i_1, i_2)$ nicht gleichzeitig den Wahrheitswert *wahr* haben. Die Begründung der Inkonsistenz wurde auch von *Protégé* als Text ausgegeben. Der *Reasoner* führte in diesem Fall keinen weiteren Test aus. Die Inferenz dauerte deswegen nur 10 ms.

Tabelle 7-2: Beispielergebnisse der Inferenz anhand der modellierten WB zur Prädiktion der Existenz und des Verhaltens von Szenenobjekten für das Beispiel „Kind folgt Ball“

Fall	Input der WB	Inferenzoutput
Fall 1	**Individuen**: $ball(i_1)$ und $hallucinated\_object(i_2)$	**Konsistenztest**: konsistent **Instanzstest**: $child\ (i_2)$ **Relationstest**: $follow(i_2, i_1)$ **Laufzeit**: 100 ms
Fall 2	**Individuen**: $ball(i_1)$, $child\ (i_2)$ und $hallucinated\_object(i_2)$	**Konsistenztest**: konsistent **Instanzstest**: nicht zutreffend **Relationstest**: $follow(i_2, i_1)$ **Laufzeit**: 20 ms
Fall 3	**Individuen**: $ball(i_1)$, $child(i_2)$ und $hallucinated\_object(i_2)$ **Relationen**: $follow(i_1, i_2)$ und	**Inferenztest**: inkonsistent **Instanzstest**: nicht ausführbar

Fall	Input der WB	Inferenzoutput
	$follow(i_1, i_2)$	**Relationstest**: nicht ausführbar **Laufzeit**: 10 ms

7.4 Erweiterung der WB mit MLN

Die Inferenz der Tabelle 7-2 zeigte anhand der logischen Formel der Gleichung (57), dass eine Instanz des Begriffs *child* immer einer Instanz des Begriffs *ball* folgen wird, gegeben die Instanz des Begriffs *ball*. In der Realität war dies nicht immer wahr. Es gab Situationen, wo ein Ball auf die Straße rollte, ohne dass ein Kind in naher Zukunft hinterher lief. Dies bedeutete, dass die logischen Formeln der Gleichung (57) Unsicherheiten beinhalten sollten. Wie im Konzept im Abschnitt 4.2.2.1 beschrieben, stellten MLN ein passendes *Framework* dar, um *FOL*-Regeln mit Unsicherheiten zu modellieren. Die in der Abbildung 7-1 dargestellten Schritte fünf bis zehn zum Modellieren, Lernen und Inferieren von MLN-Modellen werden ausführlich in den nächsten Abschnitten vorgestellt.

7.4.1 Modellierung von MLN-*FOL*-Formeln

Zur Modellierung der MLN-*FOL*-Formeln wurden analog zu dem Modell im Abschnitt 6.4.1 die Begriffe, Relationen, Axiome und logischen Formeln der im Abschnitt 7.3.1 modellierten WB als Input verwendet. Als Schnittstelle zwischen der WB und dem MLN-Modell wurde der *FOL*-Editor aus der Abbildung 6-4 verwendet. Wie dieser Editor funktionierte, wurde im Abschnitt 6.4.1 erklärt.

Die Abbildung 7-3 stellt die modellierten logischen Formeln $\mathcal{F}_{mln}^{cfb}$ des MLN-Modells zur Prädiktion der Situationsentwicklung für das Beispiel „*Kind folgt Ball*“ dar. Die Prädikate von $\mathcal{F}_{mln}^{cfb}$ beinhalteten alle Begriffe und Relationen der Ontologie aus den Gleichungen (55) und (56), die in den *SWRL*-Regeln der WB als Prädikate verwendet wurden (siehe Zeilen zwei bis sechs der Abbildung 7-3). Darüber hinaus wurde eine Teilmenge der Ontologie-Axiome aus der Tabelle 7-1 als Axiome von $\mathcal{F}_{mln}^{cfb}$ in den Zeilen neun bis 14 der Abbildung 7-3 modelliert. Die Zeilen 18 bis 20 der Abbildung 7-3 beinhalteten die Formelmenge $\mathcal{F}$ der Gleichung (57). Die Formel $\mathcal{F}_1 \in \mathcal{F}$ enthielt das Prädikat $hallucinated\_object$, da dieses Prädikat während der Inferenz die logische Konstante, die für das Abfrageprädikat *child* notwendig war, bereitstellte. Ferner wurde die Formel $\mathcal{F}_2 \in \mathcal{F}$ mit der Und-Verknüpfung des Prädikats $!\,follow(o_1, o_2)$ erweitert. Die Formel $\mathcal{F}_2$ hatte durch die Erweiterung die Bedeutung, dass ein Kind in naher Zukunft einem Ball folgen wurde und ein Ball einem Kind nicht folgen wurde, wenn der Ball und das Kind existierten. Warum diese Erweiterung notwendig war wird während des Lernens der Gewichtungen der Formeln von $\mathcal{F}_{mln}^{cfb}$ im nächsten Abschnitt erklärt.

```
2  //predicates
3  ball(indivual)
4  child(indivual)
5  hallucinated_object(indivual)
6  follow(indivual,indivual)
7
8  //Axiom: Disjoint
9  !child(o1) v  !ball(o1).
10 !ball(o1) v !hallucinated_object(o1).
11 //Axiom: asymmetry
12 !follow(o1,o2) v !follow(o2,o1).
13 //Axiom: irreflexive
14 !follow(o1,o1).
15
16 //Rules
17 //{S1}
18 ball(o_1) ^ hallucinated_object(o_2) => child(o_2)
19 //{S2}
20 ball(o_1) ^ child(o_2) => (follow(o_2,o_1) ^ !follow(o_1,o_2))
```

Abbildung 7-3: Übersicht der modellierten MLN-FOL-Regeln $\mathcal{F}_{mln}^{cfb}$ mit dem in der Abbildung 6-4 dargestellten Interface. Diese Formeln dienten zur Prädiktion der Situationsentwicklung für das Beispiel „Kind folgt Ball“.

7.4.2 Lernen der Gewichtung der MLN-*FOL*-Formeln

Zum Lernen von Gewichtungen der logischen Formeln des MLN-Modells wurde analog zum Training im Abschnitt 6.4.2.4 das *Alchemy*-Tool [201] als *Framework* verwendet. Die Trainingsdaten enthielten die Wahrheitswerte der atomaren Formeln von $\mathcal{F}_{\text{mln}}^{cfb}$ für unterschiedliche Belegungen. Hier wurden die Trainingsdaten manuell generiert. Dafür wurde die Wahrscheinlichkeit, dass die Existenz eines Kindes *halluziniert* wird, wenn ein Ball existiert $P\big(child\ (i_2)\big|ball(i_1), hallucinated\_object(i_2)\big)$, sowie die Wahrscheinlichkeit, dass ein Kind einem Ball folgt $P\big(follow(i_2, i_1)\big|ball(i_1), child\ (i_2)\big)$, festgelegt. Anschließend wurden die Wahrheitswerte der Grundprädikate von $\mathcal{F}_{\text{mln}}^{cfb}$ für unterschiedliche logische Konstanten so generiert, dass die Anzahl der Erfüllungen der Formeln von $\mathcal{F}_{\text{mln}}^{cfb}$ proportional zu den o.g. Wahrscheinlichkeiten war. Die Tabelle 7-3 stellt beispielhaft generierte Trainingsdaten für

$$P\big(child\ (i_2)\big|ball(i_1), hallucinated\_object(i_2)\big) = 0{,}7 \tag{58}$$

und

$$P\big(follow(i_2, i_1)\big|ball(i_1), child\ (i_2)\big) = 0{,}9 \tag{59}$$

dar.

Die Samples eins bis sechs dieser Tabelle modellierten Situationen, in denen ein Kind in naher Zukunft einem Ball folgte. Im Sample sieben wurden Situationen modelliert, in denen ein Kind dem Ball nicht folgte. Die Samples acht bis zehn betrachteten Situationen, wo ein Ball auf die Straße rollte, aber Kinder in der Situation nicht existierten.

Tabelle 7-3: Beispielhafte Trainingssamples zum Lernen der Gewichtungen der Formeln von $\mathcal{F}_{mln}^{cfb}$ aus der Abbildung 7-3

```
// Samples 1
child(K1)
hallucinated_object(K1)
ball(B1)
follow(K1,B1)
follow(K1,B2)
follow(K1,B3)
follow(K1,B4)
follow(K1,B5)
follow(K1,B6)
follow(K1,B7)
follow(K1,B8)
follow(K1,B9)
follow(K1,B10)
// Samples 2
child(K2)
hallucinated_object(K2)
ball(B2)
follow(K2,B1)
follow(K2,B2)
follow(K2,B3)
follow(K2,B4)
follow(K2,B5)
follow(K2,B6)
follow(K2,B7)
follow(K2,B8)
follow(K2,B9)
follow(K2,B10)
```

```
// Samples 3
child(K3)
hallucinated_object(K3)
ball(B3)
follow(K3,B1)
follow(K3,B2)
follow(K3,B3)
follow(K3,B4)
follow(K3,B5)
follow(K3,B6)
follow(K3,B7)
follow(K3,B8)
follow(K3,B9)
follow(K3,B10)
// Samples 4
child(K4)
hallucinated_object(K4)
ball(B4)
follow(K4,B1)
follow(K4,B2)
follow(K4,B3)
follow(K4,B4)
follow(K4,B5)
follow(K4,B6)
follow(K4,B7)
follow(K4,B8)
follow(K4,B9)
follow(K4,B10)
```

```
// Samples 5
child(K5)
hallucinated_object(K5)
ball(B5)
follow(K5,B1)
follow(K5,B2)
follow(K5,B3)
follow(K5,B4)
follow(K5,B5)
follow(K5,B6)
follow(K5,B7)
follow(K5,B8)
follow(K5,B9)
follow(K5,B10)
// Samples 6
child(K6)
hallucinated_object(K6)
ball(B6)
follow(K6,B1)
follow(K6,B2)
follow(K6,B3)
follow(K6,B4)
follow(K6,B5)
follow(K6,B6)
follow(K6,B7)
follow(K6,B8)
follow(K6,B9)
follow(K6,B10)
```

```
// Samples 7
child(K7)
hallucinated_object(K7)
ball(B7)
follow(K7,B1)
follow(K7,B2)
follow(K7,B3)
```

```
// Samples 8
!child(K8)
hallucinated_object(K8)
ball(B8)
// Samples 9
!child(K9)
hallucinated_object(K9)
ball(B9)
```

```
// Samples 10
!child(K10)
hallucinated_object(K10)
ball(B10)
```

Basierend auf den Trainingsdaten der Tabelle 7-3 wurden die Gewichtungen der MLN-*FOL*-Formeln aus der Abbildung 7-3 gelernt. In dieser Anwendung wurde lediglich das generative Lernen verwendet, da diese Trainingsart sich in der Anwendung im Kapitel 6 kaum von dem diskriminativen Lernen unterschiedet. Dazu wurden die Prädikate *child* und *follow* als Nicht-Evidenz gesetzt, da diese Prädikate in der Inferenzphase abgefragt wurden. Alle Prädikate wurden beim Training mit der *Close-world-assumption*-Annahme (*CWA*) versehen. Die Grundprädikate der *CWA*-Prädikate, die nicht in den Trainingsdaten vorhanden waren, hatten deshalb den Wahrheitswert *falsch.* Alle weiteren Einstellungen des *Alchemy*-Tools blieben unverändert. Das *Alchemy*-Tool lernte die Gewichtungen generativ anhand des *L-BFGS*-Algorithmus, was im Abschnitt 2.3.2.1 erwähnt wurde. Für das Training und die Inferenz des MLN-Modells wurde dieselbe VM wie im Abschnitt 6.4.2.4 verwendet.

Was die Komplexität angeht, war das gelernte MLN-Modell MLN_{CfB} Im Vergleich zu den MLN-Modellen aus dem Abschnitt 6.4.2 sehr einfach. MLN_{CfB} hatte lediglich sieben Klauseln. Zum Trainieren wurden 103 Grundprädikate verwendet. Die Trainingsdauer lag bei ca. 1,5 s.

Die Abbildung 7-4 stellt die Ergebnisse des Lernens der Gewichtungen der MLN-*FOL*-Formeln von MLN_{CfB} dar. Wie im Abschnitt 2.3.2.3 beschrieben, konnten die gelernten Gewichtungen des MLN_{CfB}-Modells nicht einzeln betrachtet werden, da die logischen Formeln des MLN-Modells abhängig voneinander waren. Allerdings konnten die Gewichtungen der Formeln von MLN_{CfB} miteinander verglichen werden. In der Abbildung 7-4 enthielten die Zeilen sieben bis 17 die Axiome des MLN-Modells. Diese Axiome wurden in der Tabelle 7-1 erläutert.

In der Zeile 20 hatte die logische Formel F_{20} zum Halluzinieren der Existenz eines Kindes, wenn ein Ball sichtbar war, eine positive Gewichtung $w_{20} = 2{,}12223$. Dies bedeutete, dass diese Formel oft in den Trainingsdaten erfüllt wurde. Die Zeile 22 enthielt die Formel F_{22} zur Prädiktion des Verhaltens, dass ein Kind einem Ball folgen würde, wenn das Kind und der Ball existierten. Die Gewichtung $w_{22} = 6{,}3697$ der Formel F_{22} war deutlich grösser als w_{20}. Dies bedeutete, dass es wahrscheinlicher war, dass ein Kind in naher Zukunft einem Ball folgte und ein Ball einem Kind nicht folgte, als dass die Existenz eines Kindes halluziniert wurde. Die Ordnungsrelation zwischen den Gewichtungen w_{20} und w_{22} entsprach also der Ordnungsrelation zwischen den Wahrscheinlichkeiten aus den Gleichungen (58) und (59), die zur Generierung von Trainingsdaten verwendet wurden. Die logische Formel F_{22} wurde in zwei Klauseln F_{23} und F_{24} in den Zeilen 23 und 24 aufgeteilt, da die Inferenz mit Klauseln einfacher war. Die Summe der Gewichtungen der Formeln F_{23} und F_{24} ergab die Gewichtung der Formel F_{22}. Die Klause F_{24} entstand durch die Erweiterung der Formel $\mathcal{F}_2$ aus der WB (siehe Gleichung (57)) mit dem Prädikat $!\,follow(o_1, o_2)$ und hatte die Bedeutung, dass ein Ball in naher Zukunft einem Kind nicht folgen wurde, wenn der Ball und das Kind existierten. Die Gewichtung $w_{24} = 3{,}80617$ der Formel F_{24} war größer als die Gewichtung $w_{23} = 2{,}56353$ der Formel F_{23}, wobei die Formel F_{23} die Bedeutung hatte, dass ein Kind in naher Zukunft einem Ball folgte, wenn der Ball und das Kind existierten. Der Grund, warum w_{24} größer w_{23} war, war, dass die Formel F_{24} in *Samples 7* der Tabelle 7-3 zehn Grundprädikate des Prädikats *follow* hatte, die F_{24} erfüllten, während lediglich drei Grundprädikate F_{23} erfüllten. Die zehn Grundprädikate der Formel F_{24} waren in den Trainingsdaten der Tabelle 7-3 aufgrund der *CWA*-Annahme implizit vorhaben. Würde man das MLN-Modell MLN_{CfB} ohne die Klausel F_{24} trainieren, hätte die Formel F_{22} lediglich die Klausel F_{23} mit der Gewichtung $w_{23} = 2{,}56353$. Die Gewichtung w_{23} hätte nicht zu den Wahrscheinlichkeiten aus den Gleichungen (58) und (59), die für das Training von MLN_{CfB} verwendet wurden, gepasst.

```
//predicate declarations
hallucinated_object(indivual)
ball(indivual)
follow(indivual,indivual)
child(indivual)

// !child(o1) v !ball(o1).
!ball(a1) v !child(a1).

// !ball(o1) v !hallucinated_object(o1).
!ball(a1) v !hallucinated_object(a1).

// !follow(o1,o2) v !follow(o2,o1).
!follow(a1,a2) v !follow(a2,a1).

// !follow(o1,o1).
!follow(a1,a1).

// 2.12223  ball(o_1) ^ hallucinated_object(o_2) => child(o_2)
2.12223  !ball(a1) v child(a2) v !hallucinated_object(a2)

// 6.3697  ball(o_1) ^ child(o_2) => (follow(o_2,o_1) ^ !follow(o_1,o_2))
2.56353  !ball(a1) v !child(a2) v follow(a2,a1)
3.80617  !ball(a1) v !child(a2) v !follow(a1,a2)
```

Abbildung 7-4: Ergebnis der trainierten Gewichtungen der FOL-Formeln des MLN-Modells MLN_{CfB} zur Prädiktion der zeitlichen Entwicklung der Situation „Kind folgt Ball“

7.4.3 Evaluation des trainierten MLN-Modells

7.4.3.1 Vorstellung des simulierten Szenarios

Zur Evaluation des trainierten MLN-Modells MLN_{CfB} wurde die Situation „*Kind folgt Ball*“ in einer Simulation generiert. In dieser Simulation fuhr das Ego-Fahrzeug entlang einer zweispurigen Straße im urbanen Raum. Am Straßenrand waren Fahrzeuge geparkt. Damit war die Sicht des Ego-Fahrzeugs zum Fußgängerweg teilweise versperrt. Die Abbildung 7-5 stellt die drei wichtigen Szenen dieses Szenarios in der zeitlichen Reihenfolge dar. In der ersten Szene (erste Zeile) waren der Ball und das Kind nicht sichtbar. In der zweiten Szene der Abbildung 7-5 rollte der Ball auf die Straße. Das Kind war auch hier nicht sichtbar. In der dritten Szene waren der Ball und das Kind sichtbar. Die Detektion der Szenenobjekte wurde in dieser Anwendung vereinfacht, da die Simulation alle Szenenobjekte bereitstellte. Lediglich ein einfaches *Ray-Tracing*-Verfahren zur Schätzung der Sichtbarkeit von Objekten aus der Ego-Perspektive wurde implementiert. So wurden in jeder Szene nur sichtbare Objekte als Szenenobjekte betrachtet.

Abbildung 7-5: Übersicht der drei wichtigsten Szenen des simulierten Szenarios (erste Spalte) und Ergebnisse der Situationsprädiktion (zweite Spalte). In der ersten Zeile sind sowohl der Ball als auch das Kind nicht sichtbar. In der zweiten Zeile rollt der Ball auf die Straße und wird als Szenenobjekt detektiert. In dieser Szene ist das Kind nicht sichtbar und somit für die Interpretation nicht relevant. In der dritten Zeile werden der Ball und das Kind detektiert.

Die Ergebnisse der Situationsprädiktion wurden verwendet, um die Automation des Ego-Fahrzeugs zu steuern. Weitere Details zur Automation sind bei Lapoehn et al. [212] zu finden.

7.4.3.2 Evaluationsergebnisse

Zur Inferenz des trainierten MLN-Modells MLN_{CfB} wurde das *Alchemy*–Tool verwendet. Objekte der Simulation wurden als Input verwendet, um die Situationsentwicklung zu prädizieren, gegeben das trainierte MLN_{CfB}-Modell (siehe Schritte zehn bis zwölf der Abbildung 7-4). Als Inferenzverfahren wurde das *MC-SAT*-Verfahren [204] verwendet. Die Tabelle 7-4 stellt die Ergebnisse der Inferenz des MLN_{CfB}-Modells mit den Szenen aus der Abbildung 7-5 als Input dar. Die erste Spalte der Tabelle 7-4 beschreibt die Inputszene. In der zweiten Spalte sind die Evidenz-Prädikate dargestellt. Inferenzergebnisse sind in der dritten Spalte zu finden. Die vierten und fünften Spalten stellen die *User*- und Gesamtinferenzlaufzeit in ms dar.

In der zweiten Zeile der Tabelle 7-4 wurden die Ergebnisse für die erste Szene dargestellt. In dieser Szene waren der Ball und das Kind unsichtbar (siehe oberstes linkes Bild der Abbildung 7-5). So wurde mit einer Wahrscheinlichkeit von 0,48 inferiert, dass ein Kind o_2 existieren konnte und dass dieses Kind mit einer Wahrscheinlichkeit von 0,33 in naher Zukunft hinter dem Objekt o_1 folgen konnte, wobei das Objekt o_1 kein Ball war. Darüber hinaus wurde mit einer Wahrscheinlichkeit von 0,32 inferiert, dass das Objekt o_1 in naher Zukunft dem Kind folgen konnte. Alle die geschätzten Wahrscheinlichkeiten waren kleiner als den empirisch bestimmten Schwellenwert von 0,55 und somit für die Automation nicht relevant. Die Szene mit den Inferenzergebnissen blieb deshalb der Inputszene gleich (siehe Abbildung 7-5, Bild oben rechts). Die Ergebnisse der Inferenz für diese Szene entsprachen der Realität, da eine der wichtigsten Prämisse des MLN_{CfB}-Modells $ball(o_1)$ nicht erfüllt war. In solchen Fällen sollte das MLN_{CfB}-Modell keine aussagekräftigen Wahrscheinlichkeiten inferieren.

Die Ergebnisse für die zweite Szene wurden in der dritten Zeile der Tabelle 7-4 dargestellt. In dieser Szene wurde der Ball als Szenenobjekt detektiert. Das Kind war nicht sichtbar und somit für die Inferenz nicht relevant. Diese Szene wurde im mittleren linken Bild der Abbildung 7-5 dargestellt. Mit dieser Szene als Input wurde mit einer Wahrscheinlichkeit von 0,75 inferiert, dass ein Kind o_2 existieren konnte und dass dieses Kind mit einer Wahrscheinlichkeit von 0,78 in naher Zukunft dem Ball o_1 folgen konnte. Da diese geschätzten Wahrscheinlichkeiten größer als der Schwellenwert von 0,55 waren, waren die Inferenzergebnisse für die Automation relevant. So entschied sich die Automation, in dieser Szene langsamer zu fahren, um potenzielle Kollisionen mit einem Kind zu vermeiden [212]. Die Inferenzergebnisse dieser Szene sind im mittleren rechten Bild der Abbildung 7-5 dargestellt. Die halluzinierte Existenz des Kindes ist in dieser Szene mit einem roten Viereck sowie den Beschriftungen o_2 und *Child* dargestellt. Der rote Pfeil an diesem Viereck beschreibt die Absicht des halluzinierten Kindes, dem Ball zu folgen. Die Ergebnisse der Inferenz für diese Szene entsprachen wie bei der ersten Szene auch der Realität, da die Prämisse des MLN_{CfB}-Modells $ball(o_1)$ erfüllt war. In einem

solchen Fall sollte das MLN-Modell aussagekräftige Wahrscheinlichkeiten für die Existenz eines Kindes und seines Verhaltens inferieren.

In der vierten Zeile der Tabelle 7-4 wurden die Ergebnisse für die dritte Szene dargestellt. In dieser Szene wurden der Ball und das Kind als Szenenobjekt detektiert (siehe unteres linkes Bild der Abbildung 7-5). Die Inferenz ergab mit einer Wahrscheinlichkeit von 0,91, dass das Kind o_2 dem Ball o_1 in naher Zukunft folgen würde. Diese Wahrscheinlichkeit war größer als der Schwellenwert von 0,55. Die Automation entschied sich deshalb, eine Notbremsung einzuleiten, um vor dem Kind stillzustehen [212]. Die Inferenzergebnisse für diese Szene sind im unteren rechten Bild der Abbildung 7-5 dargestellt. Das detektierte Kind ist in dieser Szene mit einem roten Viereck sowie der Beschriftung o_2 und *Child* dargestellt. Darüber hinaus wird die Absicht des Kindes, dem Ball zu folgen, durch einen roten Pfeil dargestellt. Die Ergebnisse der Inferenz entsprachen analog zu den Inferenzergebnissen der ersten und zweiten Szene ebenfalls der Realität, da die Prämissen $ball(o_1)$ und $child(o_2)$ des MLN_{CfB}-Modells erfüllt waren. In diesem Fall sollte das MLN-Modell aussagenkräftige Wahrscheinlichkeiten für das Verhalten des Kindes inferieren.

Die fünfte Zeile der Tabelle 7-4 stellt die Ergebnisse der vierten Szene dar. In dieser Szene wurde lediglich das Kind detektiert. Der Ball war unsichtbar und somit kein Szenenobjekt. Diese Szene wurde nicht in der Simulation abgebildet, aber in dieser Arbeit zur Vollständigkeit betrachtet. Für diese Szene wurde mit einer Wahrscheinlichkeit von 0,35 inferiert, dass das Kind o_2 dem Objekt o_1 in naher Zukunft folgen würde, wobei das Objekt o_1 kein Ball war. Darüber hinaus wurde mit einer Wahrscheinlichkeit von 0,32 inferiert, dass das Objekt o_1 dem Kind in naher Zukunft folgen könnte. Alle diese Wahrscheinlichkeiten waren kleiner als der Schwellenwert von 0,55 und somit für die Automation nicht relevant. Auch für diesen Fall entsprachen die Ergebnisse der Inferenz der Realität. Da die Prämisse $ball(o_1)$ nicht erfüllt war, sollte das MLN_{CfB}-Modell keine aussagekräftigen Wahrscheinlichkeiten inferieren.

Die vierte und fünfte Spalte der Tabelle 7-4 stellen die *User*- und Gesamtinferenzlaufzeit in ms dar. Die *User*-Inferenzlaufzeit lag zwischen 20 und 60 ms. Je einfacher die Inferenzabfrage war, desto kleiner war die *User*-Inferenzlaufzeit. So erreichte die Inferenz der dritten Szene die kleinste *User*-Inferenzlaufzeit von 20 ms, da in der dritten Szene lediglich die Wahrscheinlichkeit des Prädikats *follow* inferiert wurde. In der ersten und zweiten Szene wurden die Wahrscheinlichkeiten der Prädikate *child* und *follow* inferiert. Die Gesamtinferenzlaufzeit war deutlich höher als die *User*-Inferenzlaufzeit und lag zwischen 190 ms und 290 ms. Der Grund, warum die Gesamtinferenzlaufzeiten höher als die *User*-Inferenzlaufzeiten waren, lag darin, dass der Inferenzprozess in der VM auf die CPU warten musste.

Tabelle 7-4: Ergebnisse der Inferenz des MLN_{CfB}-Modells aus der Abbildung 7-4 mit den Inputs aus den Szenen der Abbildung 7-5

Szene	Input	Output	User-Laufzeit (ms)	Gesamt-laufzeit (ms)
Szene#1: Ball und Kind unsichtbar	$!\,ball(o_1)$ $hallucinated\_object(o_2)$ $!\,child(o_1)$	$P(child(o_2)) = 0{,}48$ $P(follow(o_1, o_1)) = 0$ $P(follow(o_1, o_2)) = 0{,}32$ $P(follow(o_2, o_1)) = 0{,}33$ $P(follow(o_2, o_2)) = 0$	60	290
Szene#2: Ball sichtbar und Kind unsicht-	$ball(o_1)$ $hallucinated\_object(o_2)$ $!\,child(o_1)$	$P(child(o_2)) = 0{,}75$ $P(follow(o_1, o_1)) = 0$	50	290

Szene	Input	Output	User-Laufzeit (ms)	Gesamt-laufzeit (ms)
bar		$P(follow(o_1, o_2)) = 0{,}08$ $P(follow(o_2, o_1)) = 0{,}78$ $P(follow(o_2, o_2)) = 0$		
Szene#3: Ball und Kind sichtbar	$ball(o_1)$ $child(o_2)$	$P(follow(o_1, o_1)) = 0$ $P(follow(o_1, o_2)) = 0$ $P(follow(o_2, o_1)) = 0{,}91$ $P(follow(o_2, o_2)) = 0$	20	230
Szene#4: Ball unsichtbar und Kind sichtbar	$!\, ball(o_1)$ $child(o_2)$ $!\, child(o_1)$	$P(follow(o_1, o_1)) = 0$ $P(follow(o_1, o_2)) = 0{,}32$ $P(follow(o_2, o_1)) = 0{,}35$ $P(follow(o_2, o_2)) = 0$	30	190

7.5 Diskussion

In diesem Abschnitt werden die Ergebnisse des vorgeschlagenen Systems zur Prädiktion der zeitlichen Entwicklung der Situation *„Kind folgt Ball“* diskutiert. Ferner werden mögliche Lösungen zur Verbesserung des Systems präsentiert.

7.5.1 Wissensbasiertes Modul

7.5.1.1 Modellierung

Zur Modellierung der WB wurden eine Ontologie und logische Regeln verwendet. Diese WB hatte den Vorteil, dass das Wissen anhand von Symbolen und logischen Regeln hierarchisch modelliert wurde. Damit war die Inferenz der WB leicht zu verstehen. Die manuelle Modellierung der WB war mit minimalem Aufwand verbunden, da das zu modellierende Wissen auf das Wesentliche reduziert wurde.

7.5.1.2 Inferenz

Die Korrektheit der Ontologie wurde durch die Inferenz anhand von drei Beispielen gezeigt. Allerdings waren die *SWRL*-Regeln zur Prädiktion der zeitlichen Entwicklung der Situation *„Kind folgt Ball“* nicht passend. Der Grund dafür war, dass diese Situation Unsicherheiten beinhaltete, die in den *SWRL*-Regeln nicht modellierte werden konnten. Darum erfolgte die Inferenz der WB in dieser Arbeit nur offline. Die Modellierung von Unsicherheiten der Situation *„Kind folgt Ball“* wurde mit PGM-Modellen außerhalb der WB vorgenommen. Die PGM-Modelle wurden dann online verwendet, um die Situation zu prädizieren. Diese PGM-Modelle basierten auf der modellierten WB und nutzten somit die Korrektheit der WB.

Die Inferenzzeit lag aufgrund der geringeren Komplexität der WB zwischen 10 ms und 100 ms. Damit war die Inferenz der WB echtzeitfähig, da die Inferenzzeit maximal 100 ms ergab. Für Anwendungen mit komplexeren Wissensbasen wäre die Inferenzzeit sicherlich höher als 100 ms gewesen, weil die Inferenzzeit abhängig von der Größe der WB im schlimmsten Fall exponentiell steigen wird (siehe Abschnitt 2.1.2.3). Für solche Anwendungen könnten zukünftige Arbeiten zur Verringerung der Inferenzzeit relevant werden, wenn die Inferenz online erfolgen sollte.

7.5.2 MLN zur Prädiktion der zeitlichen Entwicklung der Situation „*Kind folgt Ball*“

7.5.2.1 Modellierung

Das MLN-Modell diente zur Modellierung des Vorwissens zur Prädiktion der zeitlichen Entwicklung der Situation „*Kind folgt Ball*“ unter Beachtung von Unsicherheiten. Die MLN-*FOL*-Formeln basierten auf der modellierten WB. Das MLN-Modell war deshalb korrekt, bezogen auf die WB. Der Aufwand der Modellierung der MLN-*FOL*-Regeln war minimal, da die Komplexität der zugrunde liegenden WB gering war.

7.5.2.2 Training

Der Aufwand bei der Generierung von Trainingsdaten war minimal, da die Trainingsdaten auf den Wahrscheinlichkeiten der Gleichungen (58) und (59) basierten. Diese Wahrscheinlichkeiten ließen sich leicht durch menschliche Experten schätzen.

Das Training war wegen der geringeren Komplexität der MLN-*FOL*-Regeln und der überschaubaren Anzahl von Trainingssamples sehr schnell. Das trainierte MLN-Modell hatte eine handhabbare Komplexität. Darüber hinaus waren die gelernten Gewichtungen der *FOL*-Regeln mit geringem Aufwand interpretierbar, was ein Vorteil im Vergleich zu den Verfahren aus bspw. [125, 126, 128], in denen das Wissen nicht explizit und symbolisch modelliert wurde, war. Durch die Integration der Unsicherheiten in das MLN-Modell war das vorgeschlagenen System im Vergleich zu den Ansätzen aus [131–133] vorteilhaft, da diese Ansätze zwar das Wissen explizit und symbolisch modellierten, aber die Unsicherheiten nicht betrachteten. Lediglich Souza et al. [130] verwendeten MLN-Modelle zur Situationsinterpretation. Allerdings lag der Fokus auf der Prädiktion der Manöver des Ego-Fahrzeugs.

7.5.2.3 Inferenz

Die Inferenzergebnisse des MLN-Modells zeigten, dass das trainierte Modell in der Lage war, die Situation richtig und in Echtzeit zu prädizieren. Die geschätzten Wahrscheinlichkeiten entsprachen der Realität, obwohl das MLN-Modell mit manuell generierten Daten trainiert wurde. Da die Prämisse der Regeln des MLN-Modells explizit modelliert wurde, lieferte die Inferenz brauchbare Wahrscheinlichkeiten nur, wenn die Prämisse erfüllt war. Dies war ein Vorteil im Vergleich zu den Systemen, bei denen diese explizite Modellierung fehlte. Wurden solche Systeme mit künstlich generierten Situationen trainiert, konnten solche Systeme diese Situationen aufgrund ihrer Häufigkeiten in den Trainingsdaten falsch prädizieren.

Aufgrund der geringeren Komplexität des MLN-Modells waren die Inferenzzeiten mit maximal 60 ms kleiner als 100 ms. Somit war das MLN-Modell echtzeitfähig. Allerdings deckte das vorgeschlagene Modell nur eine von mehreren seltenen Situationen oder Situationen mit allgemeinem Wissen ab, die im Verkehr relevant sein können. Einige dieser Situationen werden explizit in der Fahrschule unterrichtet [214]:

1. An Haltestellen mit anhaltenden Bussen oder Straßenbahnen könnten Fahrgäste plötzlich die Straße betreten.
2. Kinder sowie ältere Menschen könnten unerwartet die Straße betreten.
3. Fahrzeuge mit fremden Kennzeichen könnten unerwartete Manöver durchführen.
4. Bei Dämmerung könnten wilde Tiere Straßen, die durch Wälder oder Felder führen, überqueren.
5. Bei Fahrzeugen, die am Straßenrand geparkt sind, könnte eine Tür zur Fahrbahnseite geöffnet werden.
6. Radfahrer mit Last auf dem Gepäckträger könnten eine schwankende Fahrweise haben.

Die Prädiktion der zeitlichen Entwicklung der o.g. Situationen mit MLN setzt voraus, dass diese Situationen sich anhand von einfachen FOL-Regeln modellieren lassen, da die Inferenzkomplexität von MLN-Modellen im schlimmsten Fall *P*- bzw. *NP-vollständig* sein könnte (siehe Abschnitt 2.3.3). Zukünftige Arbeiten sollten weitere Situationen im Verkehr anhand von MLN-Modellen inferieren und

den Ressourcenbedarf der Inferenz analysieren. Dabei sollten MLN-Modelle, die unabhängig zueinander sind, parallel inferiert werden, um die gesamte Inferenzzeit zu minimieren.

7.6 Zusammenfassung

In diesem Kapitel wurde ein System zur Schätzung der zeitlichen Entwicklung von seltenen Situationen, die meistens nur mithilfe von allgemeinem Wissen interpretiert werden können, vorgestellt. Dieses System wurde anhand des Beispiels „*Kind folgt Ball*“ prototypisch implementiert und evaluiert. In diesem Anschnitt werden die wesentlichen Ergebnisse dieses Kapitels zusammengefasst

7.6.1 Lösungsansatz

Zur Schätzung der zeitlichen Entwicklung von seltenen Situationen wurde das im Kapitel 4 vorgestellte Konzept verwendet. Der Lösungsansatz umfasste ein wissensbasiertes Modul und ein probabilistisches Modul. Die Kombination einer WB mit PGM hatte den Vorteil im Vergleich zum Stand der Technik, dass das allgemeine Vorwissen, das für die Prädiktion relevant war, sich einfach modellieren lassen konnte. Darüber hinaus konnten die Inferenzergebnisse aufgrund des expliziten, symbolisch und formal modellierten Vorwissens mit wenig Aufwand erklärt werden. Weiterhin wurde durch die Verwendung von probabilistischen Modellen zur Modellierung von Unsicherheiten die Inferenzergebnisse verbessert. Die Komplexität des vorgeschlagenen Systems blieb aufgrund der geringeren Komplexität des Vorwissens überschaubar. Der modulare Entwurf des Systems sowie die explizite und symbolische Modellierung der WB ermöglichten die Erweiterung der Softwarekomponente des Systems mit minimalem Aufwand.

7.6.2 Wissensbasiertes Modul

Das wissensbasierte Modul diente zur Modellierung und Inferenz der WB. Die WB wurde mit einer Ontologie und einer Regelbasis in einer Offlinephase modelliert. Die Ontologie enthielt Begriffe und Relation, die für die Situation „*Kind folgt Ball*“ relevant waren. Die Begriffe der Ontologie bestanden sowohl aus physikalischen Szenenelementen (z. B. Kind, Ball) als auch weiteren abstrakten Elementen. Diese Begriffe wurden anhand von *TBox*-Axiomen in einer Taxonomie angeordnet. Die Relation der Ontologie beschrieb das Verhalten des Kindes abhängig vom Ball. Die Eigenschaften dieser Relation wurden durch *RBox*-Axiome modelliert. Die Regelbasis bestand aus einer Menge von *FOL*-Regeln, die die zeitliche Entwicklung der Situation in naher Zukunft logisch beschrieben. Die Modellierung der WB war mit einem geringeren Aufwand verbunden, da das für die Prädiktion relevante allgemeine Vorwissen wenig komplex war.

Die Inferenz der modellierten WB prüfte vor allem die Konsistenz dieser WB und die Erfüllbarkeit von Regeln der Regelbasis. So wurde anhand von Evidenzen sichergestellt, dass die WB konsistent war. Aufgrund fehlender Unsicherheiten der Regelbasis lieferte die Inferenz der Regelbasis keine brauchbaren Ergebnisse für die Schätzung der zeitlichen Entwicklung der betrachteten Situation. Dies stellte allerdings kein Problem dar, da ein MLN-Modell verwendet wurde, um Unsicherheiten in der WB zu modellieren. Dieses MLN-Modell wird im nächsten Abschnitt beschrieben. Die maximale Inferenzzeit war mit 100 ms gleich der Inferenzzeit, die oft als obere Grenze für Onlineanwendungen im Kontext des automatisierten Fahrens angegeben wird. Damit war die Inferenz der WB echtzeitfähig.

7.6.3 Probabilistisches Modul

Das wissensbasierte Modul aus dem Abschnitt 7.6.2 war aufgrund der fehlenden Unsicherheiten nicht in der Lage, die zeitliche Entwicklung der Situation „*Kind folgt Ball*“ zu prädizieren. Deshalb wurde die WB anhand eines MLN-Modells mit Unsicherheiten erweitert. Zur Generierung der MLN-*FOL*-Formeln wurde in der Offlinephase die modellierte WB verwendet. So wurde die Konsistenz der MLN-*FOL*-Formeln sichergestellt, da die modellierte WB konsistent war. Die logischen MLN-*FOL*-Formeln wurden mit wenig Mühe modelliert, weil die zugrunde liegende WB eine geringere Komplexität hatte.

Zum Lernen der Gewichtungen der Formeln des MLN-Modells wurden Trainingssamples basierend auf der Wahrscheinlichkeit, dass die Existenz eines Kindes *halluziniert* wird, wenn ein Ball existiert,

sowie auf der Wahrscheinlichkeit, dass ein Kind in naher Zukunft einem Ball folgt, generiert. Diese Wahrscheinlichkeiten wurden mithilfe von Expertenwissen festgesetzt. Der Aufwand der Generierung von Trainingsdaten war aufgrund der geringen Komplexität des MLN-Modells überschaubar. Ferner war das Training sehr schnell und endete nach ca. 1,5 s. Durch den Vergleich der gelernten Gewichtung der Formeln des MLN-Modells konnte gezeigt werden, dass die gelernten Gewichtungen bezogen auf die Trainingsdaten konsistent waren.

In der Onlinephase wurde das gelernte MLN-Modell verwendet, um die zeitliche Entwicklung der Situation „*Kind folgt Ball*“ in einem simulierten Szenario zu prädizieren. Die Evaluationsergebnisse zeigten, dass das gelernte MLN-Modell die Situation richtig und in Echtzeit prädizieren konnte. Die maximale Inferenzzeit betrug 60 ms. Die Prädiktionsergebnisse wurden verwendet, um die Automation bei der Entscheidungsfindung zu unterstützen. So konnte die Automation die richtige Entscheidung frühzeitig treffen und vorausschauend fahren.

Da das vorgeschlagene MLN-Modell nur eine einzige Situation betrachtete, sollten in der Zukunft die Inferenzzeit der MLN-Modelle für weitere seltene Situationen oder Situationen mit allgemeinem Wissen, die im Verkehr relevant sind, untersucht werden.

8 Zusammenfassung und Ausblick

Die Erfassung von Szenen und Situationen ist eine primäre Aufgabe, die von automatisierten Fahrsystemen gelöst werden soll, um eine kollisionsfreie Trajektorie des Ego-Fahrzeugs zu ermöglichen. Obwohl viele Arbeiten in der Literatur diese Aufgabe adressiert haben, sind immer noch einige Punkte offen. Das Ziel dieser Arbeit war, ein System zu entwerfen, das WB, PGM und TNN zur kontextkonsistenten und holistischen Erfassung von Szenen und Situationen kombinierte. Das System sollte prototypisch implementiert und evaluiert werden.

In diesem Kapitel werden die wesentlichen Ergebnisse dieser Arbeit im ersten Abschnitt zusammengefasst. Im zweiten Abschnitt werden die Ergebnisse dieser Arbeit bezogen auf die in der Einleitung formulierten Ziele evaluiert. Der letzte Abschnitt wird sich mit Themen beschäftigen, die in der Zukunft im Zusammenhang mit dieser Arbeit angegangen werden könnten.

8.1 Zusammenfassung

8.1.1 Konzept

Eine Szene wurde in der Literatur als eine Momentaufnahme des Umfelds beschrieben. Diese beinhaltete vor allem die Szenerie und die dynamischen Objekte. In der Situation wurden Szenenelemente, die für die Fahraufgabe relevant waren, betrachtet. Dazu wurden Aspekte der Situation interpretiert, um weitere Informationen, die für die Fahraufgabe relevant waren, zu gewinnen. So wurden bspw. die zeitliche Entwicklung und die Kritikalität der Situation analysiert. Die Ansätze zur Erfassung von Szenen und Situationen in der Literatur zeigten zwei grundsätzliche Probleme:

1. Schichtbasierte Architekturen mit einer holistischen Betrachtung von Szenen und Situationen, die zu robusten und kontextkonsistenten Szenen- und Situationserfassungen führten, waren aufgrund der holistischen Modellierung teilweise sehr komplex.
2. Ansätze, die auf PGM, WB, und TNN basierten, lieferten zwar für einzelne Aufgaben der Szenen- und Situationserfassung die besten Ergebnisse, aber die Kombination von PGM, WB, und TNN wurde selten untersucht.

In dieser Arbeit wurde deshalb ein Lösungsansatz vorgeschlagen, der auf schicht-basierten Architekturen mit einer holistischen Betrachtung von Szenen und Situationen basierte. Gleichzeitig wurden die PGM-, WB-, und TNN-basierten Module der Schichten der vorgeschlagenen Architektur zugeordnet, wo diese Module aus der Literatur die besten Ergebnisse lieferten. So wurde eine modulare Architektur, die das *PF2*-Datenfusionsmodell um ein *Top-down*-Verfahren und mit Vorwissen erweitert, vorgeschlagen. Das Vorwissen enthielt relevante Kontextinformationen zur holistischen Betrachtung von Szenen und Situationen. Dieses Vorwissen wurde in WB- und PGM-basierten Modulen explizit, formal und symbolisch modelliert und von den *PF2*-Schichten der Szenen- und Situationserfassung getrennt. Darüber hinaus wurde das WB-basierte Modul gemäß dem Stand der Technik mehr der Aufgaben der Situationserfassung zugeordnet. Im Gegensatz dazu war das TNN-basierte Modul mehr für die Aufgaben der Szenenerfassung verantwortlich. Das probabilistische Modul beschäftigte sich sowohl mit den Aufgaben der Szenen- als auch der Situationserfassung.

Das vorgeschlagene Konzept nutzte also die Vorteile von PGM, WB, und TNN zur holistischen Erfassung von Szenen und Situationen. Gleichzeitig wurden Lösungen, um die Nachteile der PGM, WB, und TNN zu verringern, erläutert. Diese Lösungen adressierten die Probleme der Modellierung, des Lernens und der Inferenz von PGM, WB, und TNN. Die Modellierung und das Lernen der Modelle fanden in der Offlinephase statt, während die Inferenz der Modelle online erfolgte. Ferner wurde ein Konzept zum Umgang mit Inkonsistenzen zwischen der modellierten WB und den Evidenzen sowie zwischen den Outputs der Modelle des Systems vorgeschlagen. Die entstandenen Inkonsistenzen wurden gespeichert und von Experten in der Offlinephase bewertet. Nach der Bewertung der Inkonsistenzen wurden die Modelle bei Bedarf angepasst, getestet und validiert, bevor sie wieder für die Onlinephase weiter verwendet wurden. Das vorgeschlagene System hatte folgende Vorteile:

1. Das System konnte gleichzeitig symbolische und subsymbolische Daten verarbeiten.

2. Das System konnte das Vorwissen mit Unsicherheiten modellieren.
3. Teile des Systems, die explizit und symbolisch modelliert wurden, ließen sich einfach erklären und inhaltlich erweitern.
4. Die softwaretechnische Erweiterung der Komponente des Systems war aufgrund des modularen Aufbaus mit wenig Aufwand verbunden.
5. Die Korrektheit des Systems wurde durch die Handlung von Inkonsistenzen des Systems garantiert.

8.1.2 Semantische Segmentierung von Kamerabildern

Die Aufgabe der semantischen Segmentierung von Kamerabildern war die Klassifikation von einzelnen Pixeln in den Kamerabildern. Um diese Aufgabe zu lösen, wurden in der Literatur oft TNN verwendet, da TNN die besten Ergebnisse lieferten. Das *FCN-8s*-Modell war ein Modell, das als Meilenstein im Bereich der semantischen Segmentierung betrachtet wurde, da dieses Modell anhand von TNN deutliche Verbesserungen der semantischen Segmentierung brachte. In dieser Arbeit wurde das vorgeschlagene Konzept verwendet, um das *FCN-8s*-Modell unter Beachtung von Kontextinformationen zu bewerten. Dafür wurde zunächst die Struktur des *FCN-8s*-Modells im Detail vorgestellt. Danach wurde das *FCN-8s*-Modell anhand des frei verfügbaren *DCS*-Datensatzes in der Offlinephase trainiert. Das trainierte *FCN-8s*-Modell wurde in der Onlinephase auf dem *DCS*-Validierungsdatensatz inferiert und evaluiert. Mit dem Klassen-*mIOU*-Wert von 60,7 % und dem Kategorien-*mIOU*-Wert von 78 % war das trainierte *FCN-8s*-Modell weit von dem optimalen *IOU*-Wert von 100 % entfernt. Allerdings schwankten die einzelnen Klassen- und Kategorien-*IOU*-Werte zwischen 36 % und 92 %. Die Evaluation des trainierten *FCN-8s*-Modells unter Beachtung von Pixeln, die bezogen auf den Kontext erreichbar waren, zeigte eine kleine relative Verbesserung des Klassen-*mIOU*-Werts um bis zu 3,5 % im Vergleich zu dem Fall, in dem alle Pixel bei der Evaluation betrachtet wurden. Allerdings verschlechterte sich der Kategorien-*mIOU*-Wert um 10,6 %. Diese Ergebnisse zeigten, dass die Genauigkeit des *FCN-8s*-Modells nicht immer von der Entfernung von Punkten zur Kamera abhing. Eine weitere Evaluation der Klassen *traffic light* und *traffic sign,* wobei nur Pixel dieser Klassen, die bezogen auf den Kontext einen Einfluss auf das Manöver des Ego-Fahrzeugs hatten, betrachtet wurden, führte zu *IOU*-Werten von bis zu 98,5 %. Damit wurde gezeigt, dass das *FCN-8s*-Modell für bestimmte Klassen unter Beachtung des Kontexts brauchbare Informationen für die Fahraufgabe liefern konnte. Die *IOU*-Metrik, die einzelne Pixel bewertete, hatte den Nachteil, dass Fehler der Segmentierung gleich bestraft wurden, unabhängig davon, ob diese Fehler für die Fahraufgabe kritisch waren. Es wurde deshalb eine weitere Evaluation basierend auf *OGM*-Karten, die mit der semantischen Segmentierung des *FCN-8s*-Modells berechnet wurden, ausgeführt. Diese Evaluation der *OGM*-Karten anhand der *MSE*- und die *THW*-Metriken zeigte, dass die *OGM*-Karten für die Planung der Trajektorien von automatisierten Fahrsystemen geeignet waren.

8.1.3 Erkennung von Inkonsistenzen der semantischen Segmentierung

Die semantische Segmentierung, die vom *FCN-8s*-Modell generiert wurde, hatte den Nachteil, dass der Einfluss von Kontextinformationen auf diese Segmentierung schwer nachzuvollziehen war. Deswegen wurde das in dieser Arbeit vorgeschlagene Konzept verwendet, um zu untersuchen, ob das *FCN-8s*-Modell Kontextinformationen bei der Segmentierung verwendete. Weiterhin sollte untersucht werden, ob eine Korrelation zwischen inkonsistenten und falschen Segmentierungen vorlag. Semantisch segmentierte Regionen wurden als konsistent definiert, wenn diese Regionen das Vorwissen in Form von Kontextinformationen erfüllten. Zur Konsistenzschätzung wurden zwei Module des vorgeschlagen Konzepts verwendet: das wissensbasierte und das probabilistische Modul. Der Lösungsansatz hatte im Vergleich zu anderen Ansätzen der Literatur den Vorteil, dass das Vorwissen symbolisch, explizit und formal unter Beachtung von Unsicherheiten modelliert wurde. Damit konnten die Entscheidungen des Systems einfach erklärt werden. Dazu ließen sich die Softwarekomponente des Systems aufgrund der Trennung der modellierten Kontextinformationen vom *FCN-8s*-Modell sowie der symbolischen und expliziten Modellierung der WB mit geringerem Aufwand erweitern.

Die WB des wissensbasierten Moduls modellierte das Wissen über die Szenenelemente, ihre Relationen und die Konsistenz dieser Szenenelemente mit einer Ontologie und einer Regelbasis. Das probabilistische Modul verwendete MLN, um die WB mit Unsicherheiten in der Offlinephase zu erweitern. Darüber hinaus wurden die Gewichtungen der modellierten MLN-*FOL*-Regeln mithilfe der Annotationen der semantischen Segmentierung des *DCS*-Trainingsdatensatzes gelernt. Die Evaluationsergebnisse mit den segmentierten Regionen des *FCN-8s*-Modells auf dem *DCS*-Validierungsdatensatz zeigten, dass bis zu 13,5 % der falsch segmentierten Pixel inkonsistent waren, während nur 4,4 % der richtig segmentierten Regionen inkonsistent waren. Damit wurde die Abhängigkeit zwischen den vom *FCN-8s*-Modell falsch segmentierten Regionen und inkonsistenten Regionen gezeigt.

8.1.4 Kontextabhängige Prädiktion der zeitlichen Situationsentwicklung

Die Prädiktion der zeitlichen Situationsentwicklung war eine Aufgabe der Situationsinterpretation und stellte die Basis für die Erkennung von potenziellen Konflikten zwischen dem Ego-Fahrzeug und anderen Verkehrsteilnehmern dar. Das in dieser Arbeit vorgeschlagene Konzept wurde auf dieser Aufgabe der Situationsinterpretation angewendet. Dabei lag der Fokus auf seltenen Situationen, die zur Interpretation allgemeines Wissen benötigten. Der Lösungsansatz bestand aus einem wissensbasierten und einem probabilistischen Modul. Durch die Kombination von WB und PGM hatte das System im Vergleich zum Stand der Technik den Vorteil, dass das allgemeine Vorwissen, das für die Prädiktion der zeitlichen Entwicklung der Situation entscheidend war, einfach zu modellieren war. Ferner konnte durch die Verwendung von PGM das Wissen mit Unsicherheiten modelliert werden. Ein weiterer Vorteil war, dass die Inferenzergebnisse des Systems mit wenig Aufwand erklärbar waren, da das Vorwissen explizit, symbolisch und formal modelliert wurde. Weiterhin war das System wenig komplex, da das Vorwissen, das für die Prädiktion der zeitlichen Situationsentwicklung notwendig war, überschaubar war. Teile des Systems ließen sich aufgrund des modularen Aufbaus des Systems sowie der expliziten und symbolischen Modellierung der WB mit wenig Aufwand erweitern.

Der vorgeschlagene Lösungsansatz wurde mit dem Beispiel „*Kind folgt Ball*“ prototypisch implementiert und evaluiert. Das wissensbasierte Modul modellierte das Vorwissen über die Situation „*Kind folgt Ball*“ mit einer WB bestehen aus einer Ontologie und einer Regelbasis in der Offlinephase. Zur Modellierung von Wissen mit Unsicherheiten wurde die WB mit MLN erweitert. Die MLN-*FOL*-Formeln basierten auf der modellierten Wissensbasis. Zum Lernen der Gewichtung der modellierten MLN-*FOL*-Formeln wurden anhand von Expertenwissen Trainingssamples, die bestmöglich die Situation „*Kind folgt Ball*“ abdeckten, mit minimalem Aufwand generiert. Das trainierte MLN-Modell konnte in der Onlinephase in Echtzeit die Situation „*Kind folgt Ball*“ richtig prädizieren. Eine Automation verwendete die Prädiktionsergebnisse zum vorausschauenden Fahren in der Simulation.

8.2 Vergleich der Ziele mit den erreichten Ergebnissen

Die in der Einleitung dieser Arbeit festgelegten Ziele (siehe Kapitel 1) wurden aufgrund des vorgeschlagenen Konzepts sowie des implementierten und evaluierten Systemprototyps erreicht.

Das Konzept stellte eine Lösung vor, die WB, PGM und TNN als Module verwendete. Diese drei Module wurden den Aufgaben der Szenen- und Situationserfassung zugeordnet, die sie am besten lösen konnten. Dazu wurden Lösungen vorgeschlagen, um die Probleme dieser Module zu mindern. Damit wurden WB, PGM und TNN in dem Lösungsansatz optimal kombiniert. Die holistische Betrachtung von Szenen und Situationen wurde im Konzept durch die Erweiterung des *PF2*-Modells mit einem *Top-down*-Verfahren und mit Vorwissen in Form von Kontextinformationen sichergestellt. Sowohl das Konzept als auch der implementierte Prototyp waren in der Lage, das Vorwissen mit Unsicherheiten anhand von WB und *OGM* zu modellieren. Durch die explizite, symbolische und formale Modellierung des Vorwissens waren die Entscheidungen des Systems erklärbar. Lediglich das TNN-basierte Modul, das subsymbolische Daten verarbeitete, blieb eine *Black-Box*. Allerdings wurden die Outputs des TNN-basierten Moduls auf einer symbolischen Ebene mit MLN weiterverarbeitet, wobei die MLN besser erklärbar waren. Der modulare Aufbau des Systems sowie die explizite und symbolische Modellierung der WB ermöglichten eine einfache Erweiterung der Softwarekomponente des Systems. Da das System aus unterschiedlichen Modulen bestand, gab es die Möglichkeit, dass die Outputs

dieser Module sich widersprachen. Ferner konnten Inkonsistenzen zwischen dem modellierten Vorwissen und den Evidenzen entstehen. Um mit diesen Widersprüchen und Inkonsistenzen umzugehen, wurde ein Konzept vorgeschlagen. Dieses Konzept stellte sicher, dass menschliche Experten diese Widersprüche und Inkonsistenzen bewerteten, die betroffenen Module anpassten und validierten, bevor diese Module im System weiterverwendet wurden. So wurde die Korrektheit des Systems sichergestellt.

Der implementierte und validierte Prototyp lieferte folgende Ergebnisse:

1. Das trainierte und evaluierte *FCN-8s*-Modell zur semantischen Segmentierung: Während der Evaluation wurden Kontextinformationen durch die Relevanz von Pixeln für die Fahraufgabe betrachtet. Da diese Kontextinformationen Elemente der Situation waren und die semantische Segmentierung Element der Szene war, wurden hier die Szene und die Situation holistisch betrachtet.
2. Trainierte und evaluierte MLN-Modelle zur Schätzung von Inkonsistenzen der semantischen Segmentierung sowie zur Prädiktion der zeitlichen Situationsentwicklung. Dabei basierten die MLN-Modelle auf Wissensbasen, die Kontextinformation über relevante Szenenelemente und Situationsaspekte in Form von Vorwissen explizit, formal und symbolisch modellierten. Die Inferenz der MLN-Modelle lieferten Informationen, die die holistische Betrachtung von Szenen und Situationen ermöglichten, da die Abhängigkeit von Situationen zu Szenen dadurch modelliert wurde.

Obwohl viele Maßnahmen getroffen wurden, um die Komplexität des Systems zu reduzieren, wurde die Echtzeitanforderung zum Teil nicht erfüllt. Vor allem MLN und TNN waren rechenaufwendig. Jedoch war noch Optimierungspotenzial für MLN und TNN vorhanden. Kombiniert mit den Fortschritten der HW zum parallelen Rechnen, könnte die Echtzeitanforderung in der nahen Zukunft erfüllt werden.

8.3 Ausblick

Das in dieser Arbeit vorgeschlagene System stellte eine gute Lösung für die holistische Modellierung von Szenen und Situationen dar. Allerdings wurden Probleme bei der Implementierung und der Evaluation des Systems festgestellt. Zukünftige Arbeiten, um diese Probleme zu lösen, wurden in den Abschnitten 5.7.6, 6.5 und 7.5 präsentiert. In diesem Abschnitt werden Ideen zur Weiterführung des in dieser Arbeit vorgeschlagenen Systems vorgestellt.

8.3.1 Integration der Konsistenzschätzung beim Trainieren der semantischen Segmentierung

In dieser Arbeit wurde gezeigt, dass es eine Korrelation zwischen Regionen, die inkonsistent waren, und Regionen, die vom *FCN-8s*-Modell falsch segmentiert wurden, gab. Zukünftige Arbeiten könnten als Beitrag zur holistischen Erfassung von Szenen und Situationen die Konsistenzschätzung beim Trainieren des *FCN-8s*-Modells verwenden, um die zu minimierende Fehlerfunktion der Gleichung (21) mit Kontextinformationen aus der Konsistenzschätzung zu erweitern. Die neue Fehlerfunktion

$$L_{cons} = L + \lambda \frac{1}{K} \sum_{k=1}^{K} \left(1 - P_\theta \left(\left(K\left(R_k^{I_S}\right) \right) \middle| R_k^{I_S}, KB \right) \right) \qquad (60)$$

wird minimiert, wenn die Fehlerfunktion L aus der Gleichung (21) minimiert wird, während die Konsistenzwahrscheinlichkeiten der segmentierten Regionen $P_\theta \left(\left(K\left(R_k^{I_S}\right) \right) \middle| R_k^{I_S}, KB \right)$ maximiert wird. λ ist ein Parameter, der den Einfluss der Konsistenzschätzung auf die semantische Segmentierung steuert. Hu et al. [215] schlugen eine ähnliche Methode zur Integration von Wissen in Form von gewichteten *FOL*-Formeln in TNN-Modellen und zeigten, dass diese Integration des Wissens die Erkennungsrate der TNN-Modelle insbesondere bei teil-überwachtem Lernen und bei dünnbesetzten Daten verbesserte. Diese Arbeit könnte als Referenz für die Weiterentwicklung der Fehlerfunktion aus der Gleichung (60) verwendet werden.

Eine weitere Möglichkeit, Kontextinformationen aus der Konsistenzschätzung zur Verbesserung der semantischen Segmentierung zu verwenden, wäre, mit MLN die Klassenwahrscheinlichkeiten von inkonsistenten Regionen zu schätzen, gegeben die konsistenten Regionen und das Vorwissen. Die MLN-*FOL*-Formeln zur Schätzung der Klassenwahrscheinlichkeiten von segmentierten Regionen könnten durch eine leichte Anpassung der MLN-*FOL*-Formeln zur Konsistenzschätzung aus der Abbildung 6-14 generiert werden. Die Formel F_{19} der Abbildung 6-14 könnte z. B. durch F_{19}^{c}: $\forall o_1 \forall o_2 \forall c_1 \; \forall c_2 \; ObjectClass(+c_1, o_1) \wedge Above(o_1, o_2) \Rightarrow ObjectClass(+c_2, o_2)$ ersetzt werden. F_{19}^{c} modelliert die semantische Klasse der Region o_2, gegeben die Region o_1, ihre semantische Klasse und die räumliche Relation $Above$ zwischen den Regionen o_1 und o_2. Die Gewichtung von F_{19}^{c} könnte analog zu der Gewichtung von F_{19} gelernt werden. Die MLN-Klassenwahrscheinlichkeiten könnten dann mit den Klassenwahrscheinlichkeiten der semantischen Segmentierung aus dem *FCN-8s*-Modell fusioniert werden, wobei der MLN-Klassifikator eine Art Prior wäre. Die Parameter der Fusion könnten während des Trainings des *FCN-8s*-Modells oder anhand eines trainierten *FCN-8s*-Modells gelernt werden.

Die o. g. Vorschläge zur Kombination von Kontextinformationen aus MLN mit dem *FCN-8s*-Modell setzen voraus, dass die MLN im Vorfeld modelliert und trainiert wurden. So können die MLN-Modelle bei der Kombination mit dem *FCN-8s*-Modell nicht mehr angepasst werden. Es sollten deshalb in der Zukunft Mechanismen entwickelt werden, in denen Modelle mit Kontextinformationen gleichzeitig mit TNN-Modellen gelernt werden könnten. Die Modelle mit Kontextinformationen müssen nicht zwangsläufig symbolisch repräsentiert werden. Die sogenannten *Kapsel*-Netze [181, 216] bieten die Möglichkeit an, Relationen zwischen *Low-Level*-Merkmalen wie Teile von Objekten und *High-Level*-Merkmalen wie Objekte zu modellieren. Diese Idee könnte verwendet werden, um Kontextabhängigkeiten zwischen semantisch segmentierten Regionen zu modellieren. Ferner sollten Mechanismen vorgeschlagen werden, um den Einfluss der gelernten Modelle mit Kontextinformationen auf TNN zu untersuchen. Die positiven Einflüsse der Modelle mit Kontextinformationen auf TNN werden sehr wahrscheinlich hervorgehoben, wenn wenige Trainingsdaten vorhanden sind. Dazu können einfachere TNN mit wenigen Iterationen trainiert werden. TNN, die mit Hilfe von Kontextinformationen trainiert werden, werden robustere Inferenzergebnisse bei seltenen Situationen und *Adversarial*-Beispielen liefern.

8.3.2 Explizite Modellierung von Verkehrsregeln zur Prädiktion der zeitlichen Entwicklung von Situationen

Verkehrsregeln sind wichtige Informationsquellen, um die zeitliche Entwicklung von Verkehrssituationen zu prädizieren, da Verkehrsteilnehmer inklusive das Ego-Fahrzeug diesen Regeln folgen müssen. Eine WB kann zur manuellen Modellierung dieser Regeln verwendet werden, da die Regeln in Form von Texten vorhanden sind. Der Vorteil der manuellen und symbolischen Modellierung der Verkehrsregeln im Vergleich zum Verfahren, bei dem diese Regeln aus den Daten gelernt werden sollten, ist, dass mit der WB die Korrektheit und Konsistenz der Regeln formal bewiesen werden kann. Geht man davon aus, dass Verkehrsteilnehmer die Verkehrsregeln manchmal verletzen, können diese Regeln anhand von MLN mit Unsicherheiten erweitert werden. Die MLN mit gewichteten Verkehrsregeln können dann verwendet werden, um eine *a priori* zeitliche Situationsentwicklung zu generieren. Diese *A-priori*-Situationsentwicklung könnte dann mit datengetriebenen Verfahren zur Prädiktion von zeitlichen Situationsentwicklungen kombiniert werden. So ein System wäre vergleichbar mit einem Fahrer, der von der Fahrschule die Verkehrsregeln explizit lernt und diese mit datengetriebenen Verfahren kombiniert, um die Situation zu interpretieren. So ein Fahrer gewinnt mit der Zeit durch datengetriebene Verfahren mehr Erfahrung und kann somit besser vorausschauend fahren.

Eine WB der Verkehrsregel könnte auch verwendet werden, um sicherzustellen, dass die von dem automatisierten Fahrsystem geplanten und gefahrenen Manöver gemäß den Verkehrsregeln zulässig sind. Dies wäre ein wesentlicher Vorteil in Bezug auf das Testen und Absichern von automatisierten Fahrsystemen.

8.3.3 Vorschläge für weitere zukünftige Arbeiten

- Die Konsistenzschätzung könnte verwendet werden, um die räumlichen Relationen der vom *FCN-8s*-Modell semantisch segmentierten Regionen in der Trainingsphase zu lernen. Während der Inferenz der semantischen Segmentierung werden Regionen, die die gelernten räumlichen Relationen verletzen, als inkonsistent geschätzt. Die inkonsistenten Regionen deuten auf Änderungen des Verhaltens der semantischen Segmentierung zwischen der Trainings- und der Inferenzphasen hin. Diese Änderungen des Verhaltens könnten mit falschen Segmentierungen oder seltenen Situationen verbunden sein. So könnte das Verhalten der semantischen Segmentierung in der Inferenzphase überwacht werden.
- Die Prädiktion der zeitlichen Situationsentwicklung könnte als Aufmerksamkeitsmodell für die Erfassung von Szenenelementen verwendet werden. Die Halluzination über die Existenz des Kindes in der der Situation „*Kind folgt Ball*“ könnte z. B. von der Objekterkennung verwendet werden, um besonders auf die Detektion eines Objekts der Klasse Kind aufmerksam zu machen. Damit wäre ein weiterer Beitrag zur holistischen Modellierung von Szenen und Situationen geleistet.
- Mechanismen zur automatischen Lösung von Inkonsistenzen und Unstimmigkeiten, die in dem vorgeschlagenen System in der Onlinephase gespeichert wurden (siehe Abschnitt 4.2.4), sollten untersucht werden. Damit sollte der menschliche Experte entlastet werden. Diese Mechanismen sollten im optimalen Fall das System in der Offlinephase anpassen und validieren, bevor es wieder für die Onlinephase verwendet wird.
- Die Erweiterung von MLN um Evidenzen mit Wahrscheinlichkeiten sollte untersucht werden. So könnte z. B. in dem Beispiel „Kind folgt Ball“ die Evidenz $ball(o_1)$ mit der Wahrscheinlichkeit $P = 90\ \%$ eingegeben werden. Damit würde die MLN-Inferenz die Unsicherheiten der Objekterkennung betrachten. Nyga [217] schlug mit dem *pracmln*-Tool die sogenannten Fuzzy-Evidenzen vor, um den Wahrheitswert von MLN-Evidenzen als Wahrscheinlichkeiten zu modellieren. Das *pracmln*-Tool sollte in der Zukunft betrachtet werden.
- Mechanismen zur Kombination von WB, TNN und PGM sowohl beim Modellieren als auch beim Lernen und Inferieren sollten weiterhin untersucht werden, da keine dieser Methoden aus der Literatur alle Aufgaben der Szenen- und Situationserfassung am besten lösen konnte. Schnittstellen zwischen diesen drei Methoden stellen eine große Herausforderung dar, da diese Methoden unterschiedliche Algorithmen verwenden. Richardson und Domingos [25], Pearl [218, 219], Selvaraju et al. [220], Battaglia et al. [221], Hu et al. [215], Diligenti et al. [222] Dong et al [223] sind einige Autoren, die mindestens zwei der drei Methoden kombinieren. Diese Arbeiten könnten als Inspiration auf dem Weg zur allgemeinen künstlichen Intelligenz verwendet werden.

8.3.4 Integration des vorgeschlagenen Systems ins Versuchsfahrzeug

Im Rahmen dieser Arbeit wurden erste Schritte zur Integration des vorgeschlagenen Systems ins Versuchsfahrzeug adresseiert. Die Abbildung 8-1 stellt das Versuchsfahrzeug *VIEWCar II* [224] des Instituts TS, welches für die Integration verwendet wurde, dar. Dieses Versuchsfahrzeug war mit (Stereo)Kameras-, Radar-, Ultraschall- und Laserscanner-Sensoren ausgestattet. Insbesondere das auf das Dach des Fahrzeugs montierte Stereokamerasystem, welches von Börner et al. [225] unter dem Namen (*IPS*) entwickelt wurde, lieferte Inputbilder für das vorgeschlagene System.

Abbildung 8-1: Versuchsfahrzeug VIEWCar II des Instituts TS.

Zur Integration der semantischen Segmentierung wurde das *FCN-8s*-Modells aus dem Kapitel 5 gemäß den Hinweisen von Lee [162] angepasst. Ferner wurde die Anzahl der semantischen Klassen zur Verringerung der Inferenzzeit auf sechs reduziert. Dafür wurden die Kategorien des *DCS*-Datensatzes als Referenz verwendet. Die zusammengefassten semantischen Klassen sind in der Tabelle 8-1 zu finden.

Tabelle 8-1: Tabellarische Darstellung der zusammengefassten semantischen Klassen des DCS-Datensatzes sowie weitere Metainformationen.

Klassenname	**Klassen-ID**	**R**	**G**	**B**
unknow	0	0	0	0
flat	7	128	64	128
background	11	70	70	70
sky	23	70	130	180
human	24	220	20	60
vehicle	26	0	0	142

Das angepasste *FCN-8s*-Modell wurde dann, wie im Abschnitt 5.4 beschrieben, mit dem *DCS*-Datensatz vortrainiert. Weiterhin wurden ca. zehn *Ground-Truth*-Bilder aus den Kamerabildern des Versuchsfahrzeugs generiert. Die geringe Anzahl der *Ground-Truth*-Bilder stellte kein Problem für das Training dar, da die Statistik der *DCS*-Bilder mit der Statistik der Kamerabilder des Versuchsfahrzeugs ähnlich war. Die generierten *Ground-Truth*-Bilder wurden im weiteren Schritt verwendet, um das angepasste *FCN-8s*-Modell neu zu trainieren. Dabei wurde das mit dem *DCS*-Datensatz trainierten *FCN-8s*-Modell zur Initialisierung des Trainings verwendet. Das Training konvergierte nach ca. 3000 Iterationen.

Vor der Integration des trainierten *FCN-8s*-Modells ins Versuchsfahrzeug, wurde dieses Modells mit Hilfe der *TensorRT*-Tools [163] optimiert, um die Inferenzzeit zu verbessern [162]. Die Integration des optimierten *FCN-8s*-Modells ins Versuchsfahrzeug erfolgte mit Hilfe des *NVIDIA-DrivePX-AutoChauffeur*-Rechners [164]. Dieser Rechner stellte *GPU*-Einheiten mit minimalen Stromverbraucht zur Inferenz von TNN in Fahrzeugen zur Verfügung. Die Inferenzzeit lag mit ca. 220 ms bei der Bildauflösung 1024×512 über 100 ms und war somit nicht echtzeitfähig.

Zukünftige arbeiten sollten das angepasste *FCN-8s*-Modell validieren und die Inferenzzeit verbessern. Weiterhin sollten die wissensbasierten und probabilistischen Module aus den Kapiteln 6 und 7 ins Versuchsfahrzeug integriert werden.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AUC ..... *Area Under the Curve*
BoW ..... *Border over Width*
CamVid ..... *Cambridge-driving Labelled Video Database*
CNN ..... *Convolutional Neural Networks*
Conv ..... *Convolutional layer*
CRF ..... *Conditional Random Fields*
CWA ..... *Close World Assumption*
DCS ..... *Daimler Cityscapes Dataset*
DoA ..... *Difference over Area*
DRL ..... *Deep Reinforcement Learning*
DUC ..... Dense Upsampling Convolution
FC ..... *Fully-Connected layer*
FCN ..... *Fully Convolutional Networks*
FN ..... *False Negative*
FOL ..... *First Order Logic*
FP ..... *False Positive*
FPR ..... *False Positive Rate*
GDHM ..... *gewichtete Differenz der Histogrammmittelwerte*
GPU ..... *Graphics Processing Unit*
HOG ..... *Histogram of Oriented Gradients*
ILSVRC ..... *ImageNet Large Scale Visual Recognition Challenge*
InPlace-ABN ..... *In-Place Activated Batch Normalization*
IOU ..... *Intersection Over Union*
IPS ..... *Integrated Positioning System*
JDL ..... *Joint Directors of Laboratories*
KNF ..... konjunktive Normalform
L-BFGS ..... *memory-limited-Broyden-Fletcher-Goldfarb-Shanno*
Lidar ..... *Light Detection and Ranging*
MAP ..... *Maximum-A-Posteriori*
MCMC ..... *Markov Chain Monte Carlo*
mIOU ..... *mean Intersection Over Union*
MLN ..... Markov-Logik-Netze
MPE ..... *Most Probable Explanation*
MSE ..... *Mean Squared Error*
OGM ..... *Occupancy Grid Map*
OWA ..... *Open World Assumption*
OWL ..... *Web Ontology Language*
PF2 ..... *ProFusion2*
PGM ..... Probabilistische Graphische Modelle
PKW ..... Personenkraftwagen
Pool ..... *Max-pooling layer*
Radar ..... *Radio Detection and Ranging*
ReLU ..... *Rectified Linear Unit*
RNN ..... Rekurrente Neuronale Netze
ROC ..... *Receiver Operating Characteristic*
SAE ..... *Society of Automotive Engineers*
SVM ..... *Support Vector Machine*
SWRL ..... *Semantic Web Rule Language*
THW ..... *Time Headway*
TNN ..... Tiefe Neuronale Netze
TP ..... *True Positive*
TS ..... *Institut of Transportation Systems*
VM ..... virtuelle Maschine
WB ..... Wissensbasis

Quellen-/Literaturverzeichnis

[1] Stuckenschmidt, H.: Ontologien. Konzepte, Technologien und Anwendungen. Berlin, Heidelberg 2009.

[2] Gawron, J. H.; Keoleian, G. A.; De Kleine, R. D.; Wallington, T. J.; Kim, H. C.: Life Cycle Assessment of Connected and Automated Vehicles: Sensing and Computing Subsystem and Vehicle Level Effects. In: Environmental science & technology 52 (2018) 5, S. 3249–56.

[3] Spieser, K.; Treleaven, K.; Zhang, R.; Frazzoli, E.; Morton, D.; Pavone, M.: Toward a Systematic Approach to the Design and Evaluation of Automated Mobility-on-Demand Systems A Case Study in Singapore 2013.

[4] Ahlgrimm, J.: Lexikon: Automatisiertes Fahren. Bonn 2018.

[5] NHTSA - United States Department of Transportation: Automated Vehicles for Safety. SAE Automation Levels. URL: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety. Abrufdatum 02.11.2018.

[6] Audi MediaCenter: Der neue Audi A8 – hochautomatisiertes Fahren auf Level 3.

[7] Dickmanns, E. D.; Mysliwetz, B.; Christians, T.: An integrated spatio-temporal approach to automatic visual guidance of autonomous vehicles. In: IEEE Transactions on Systems, Man, and Cybernetics 20 (1990) 6, S. 1273–84.

[8] Franke, U.; Mehring, S.; Suissa, A.; Hahn, S.: The Daimler-Benz steering assistant: a spin-off from autonomous driving: Intelligent Vehicles ’94 Symposium, Proceedings of the 1994.

[9] DARPA Urban Challenge. URL: http://archive.darpa.mil/grandchallenge/. Abrufdatum 22.07.2016.

[10] Thrun, S.; Montemerlo, M.; Dahlkamp, H.; Stavens, D.; Aron, A.; Diebel, J.; Fong, P.; Gale, J.; Halpenny, M.; Hoffmann, G.; Lau, K.; Oakley, C.; Palatucci, M.; Pratt, V.; Stang, P.; Strohband, S.; Dupont, C.; Jendrossek, L.-E.; Koelen, C.; Markey, C.; Rummel, C.; van Niekerk, J.; Jensen, E.; Alessandrini, P.; Bradski, G.; Davies, B.; Ettinger, S.; Kaehler, A.; Nefian, A.; Mahoney, P.: Stanley: The Robot That Won the DARPA Grand Challenge. In: Buehler, M.; Iagnemma, K.; Singh, S. (Hrsg.): The 2005 DARPA Grand Challenge: The Great Robot Race. Berlin, Heidelberg 2007.

[11] HAVEit. URL: http://www.haveit-eu.org/. Abrufdatum 22.07.2016.

[12] AdaptiVe. Automated Driving. URL: https://www.adaptive-ip.eu/index.php/objectives.html. Abrufdatum 22.07.2016.

[13] Stadtpilot. URL: https://www.tu-braunschweig.de/stadtpilot. Abrufdatum 22.07.2016.

[14] AutoMate. Automation as accepted and Trusted TeamMate to enhance traffic safety and efficiency. URL: http://www.automate-project.eu/.

[15] Google Self-Driving Car Project. URL: https://www.google.com/selfdrivingcar/. Abrufdatum 27.06.2016.

[16] Ziegler, J.; Bender, P.; Schreiber, M.; Lategahn, H.; Strauss, T.; Stiller, C.; Dang, T.; Franke, U.; Appenrodt, N.; Keller, C. G.; Kaus, E.; Herrtwich, R. G.; Rabe, C.; Pfeiffer, D.; Lindner, F.; F., S.; Erbs, F.; Enzweiler, M.; Knoppel, C.; Hipp, J.; Haueis, M.; Trepte, M.; Brenk, C.; Tamke, A.; Ghanaat, M.; Braun, M.; Joos, A.; Fritz, H.; Mock, H.; Hein, M.; Zeeb, E.: Making Bertha Drive—An Autonomous Journey on a Historic Route. In: IEEE Intelligent Transportation Systems Magazine 6 (2014) 2, S. 8–20.

[17] DAIMLER: On the road in self-driving vehicles. Autonomous driving for more comfort, more safety and efficiency. URL: https://www.daimler.com/innovation/autonomous-driving/special/changes.html. Abrufdatum 22.07.2016.

[18] 360c. Volvo Cars. URL: https://www.volvocars.com/de/modelle/concept-cars/360c. Abrufdatum 02.11.2018.

[19] AUDI: Audi piloted driving.
URL: http://www.audi.de/de/brand/de/vorsprung_durch_technik/content/2014/10/piloted-driving.html. Abrufdatum 22.06.2016.

[20] Waymo. URL: https://waymo.com/. Abrufdatum 02.11.2018.
[21] Uber. Self-Driving Ubers, The world's first Self-Driving Ubers are on the road in the Steel City. URL: https://www.uber.com/cities/pittsburgh/self-driving-ubers/. Abrufdatum 02.11.2018.
[22] Zoox. The art of mobility. URL: https://zoox.com/. Abrufdatum 02.11.2018.
[23] Aurora. We do self-driving cars. URL: https://aurora.tech/. Abrufdatum 02.11.2018.
[24] Meyer-Fujara, J.; Puppe, F.; Wachsmuth, I.: Expertensysteme und Wissensmodellierung: Einführung in die künstliche Intelligenz.
[25] Richardson, M.; Domingos, P.: Markov Logic Networks. In: Mach. Learn. 62 (2006) 1-2, S. 107–36.
[26] Ertel, W. (Hrsg.): Grundkurs Künstliche Intelligenz. Wiesbaden 2016.
[27] Özsu, M. T.; LIU, L.: Encyclopedia of database systems. New York, London 2009.
[28] Motik, B.; Patel-Schneider, P. F.; Parsia, B.; Bock, C.; Fokoue, A.; Haase, P.; Smith, M.: OWL 2 web ontology language: Structural specification and functional-style syntax. In: W3C recommendation 27.65 (2009).
[29] Krötzsch, M.; Simancik, F.; Horrocks, I.: Description Logics. In: IEEE Intelligent Systems 29 (2014) 1, S. 12–19.
[30] Baader, F.: The description logic handbook. Theory, implementation, and applications. Cambridge 2005.
[31] Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. In: Journal of Artificial Intelligence Research 12 (2000), S. 199–217.
[32] Sirin, E.; Parsia, B.; Grau, B. C.; Kalyanpur, A.; Katz, Y.: Pellet: A Practical OWL-DL Reasoner. In: Web Semant 5 (2007) 2, S. 51–53.
[33] Koller, D.; Friedman, N.: Probabilistic graphical models. Principles and techniques. Cambridge, Massachusetts 2009.
[34] Koller, D.; Friedman, N.; Getoor, L.; Taskar, B.: Graphical Models in a Nutshell. In: Getoor, L.; Taskar, B. (Hrsg.): Introduction to Statistical Relational Learning 2007.
[35] Huynh, T. N.; Mooney, R. J.: Discriminative structure and parameter learning for Markov logic networks. In: Cohen, W.; McCallum, A.; Roweis, S. (Hrsg.): the 25th international conference.
[36] Loshchilov, I.: LM-CMA: An alternative to L-BFGS for large-scale black Box optimization. In: Evolutionary computation 25 (2015).
[37] Singla, P.; Domingos, P.: Discriminative Training of Markov Logic Networks: Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2 2005.
[38] Lowd, D.; Domingos, P.: Efficient Weight Learning for Markov Logic Networks: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases. Berlin, Heidelberg 2007.
[39] Kruse, R.; Borgelt, C.; Braune, C.; Klawonn, F.; Moewes, C.; Steinbrecher, M.: Computational Intelligence. Wiesbaden 2015.
[40] Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le V, Q.; Hinton, G. E.; Dean, J.: Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In: CoRR abs/1701.06538 (2017).
[41] He, K.; Zhang, X.; Ren, S.; Sun, J.: Identity Mappings in Deep Residual Networks. In: CoRR abs/1603.05027 (2016).
[42] Le, H.; Borji, A.: What are the Receptive, Effective Receptive, and Projective Fields of Neurons in Convolutional Neural Networks? In: CoRR abs/1705.07049 (2017).
[43] Li, F.-F.; Karpathy, A.; Johnson, J.: CS231n Convolutional Neural Networks for Visual Recognition.
[44] Ulbrich, S.; Menzel, T.; Reschka, A.; Schuldt, F.; Maurer, M.: Definition der Begriffe Szene, Situation und Szenario für das automatisierte Fahren: Workshop Fahrerassistenzsysteme.

[45] Balzer, P.: Fahrzeugumfeldsensorik: Überblick und Vergleich zwischen Lidar, Radar, Video. URL: http://www.cbcity.de/fahrzeugumfeldsensorik-ueberblick-und-vergleich-zwischen-lidar-radar-video. Abrufdatum 13.03.2018.

[46] Barnard, M.: Tesla & Google Disagree About LIDAR — Which Is Right? URL: https://cleantechnica.com/2016/07/29/tesla-google-disagree-lidar-right/. Abrufdatum 13.03.2018.

[47] Steinberg, A. N.; Bowman, C. L.; White, F. E.: Revisions to the JDL Data Fusion Model 1999.

[48] Polychronopoulos, A.; Amditis, A.; Scheunert, U.; Tatschke, T. (Hrsg.): Revisiting JDL model for automotive safety applications: the PF2 functional model, 2006 9th International Conference on Information Fusion 2006.

[49] Hosang, J.; Benenson, R.; Dollar, P.; Schiele, B.: What Makes for Effective Detection Proposals? In: IEEE Trans. Pattern Anal. Mach. Intell. 38 (2016) 4, S. 814–30.

[50] Uijlings, J. R. R.; van de Sande, K. E. A.; Gevers, T.; Smeulders, A. W. M.: Selective Search for Object Recognition. In: International Journal of Computer Vision 104 (2013) 2, S. 154–71.

[51] Krähenbühl, P.; Koltun, V.: Geodesic Object Proposals. In: Fleet, D.; Pajdla, T.; Schiele, B.; Tuytelaars, T. (Hrsg.): Computer Vision - ECCV 2014. Cham 2014.

[52] Alexe, B.; Deselaers, T.; Ferrari, V.: Measuring the Objectness of Image Windows. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (2012) 11, S. 2189–202.

[53] Zitnick, C. L.; Dollár, P.: Edge Boxes: Locating Object Proposals from Edges. In: Fleet, D.; Pajdla, T.; Schiele, B.; Tuytelaars, T. (Hrsg.): Computer Vision - ECCV 2014. Cham 2014.

[54] Dalal, N.; Triggs, B.: Histograms of Oriented Gradients for Human Detection: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01. Washington, DC, USA 2005.

[55] Wang, X.; Han, T. X.; Yan, S. (Hrsg.): An HOG-LBP human detector with partial occlusion handling, 2009 IEEE 12th International Conference on Computer Vision 2009.

[56] Felzenszwalb, P. F.; Girshick, R. B.; McAllester, D.; Ramanan, D.: Object Detection with Discriminatively Trained Part-Based Models. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (2010) 9, S. 1627–45.

[57] Kowsari, T.; Beauchemin, S. S.; Cho, J. (Hrsg.): Real-time vehicle detection and tracking using stereo vision and multi-view AdaBoost, 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC) 2011.

[58] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks. In: F. Pereira; C. J. C. Burges; L. Bottou; K. Q. Weinberger (Hrsg.): Advances in Neural Information Processing Systems 25 2012.

[59] Simonyan, K.; Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: CoRR abs/1409.1556 (2014).

[60] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S. E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A.: Going Deeper with Convolutions. In: CoRR abs/1409.4842 (2014).

[61] He. K.; Zhang, X.; Ren, S.; Sun, J.: Deep Residual Learning for Image Recognition. In: arXiv preprint arXiv:1512.03385 (2015).

[62] Jie Hu; Li Shen; Gang Sun: Squeeze-and-Excitation Networks. In: CoRR abs/1709.01507 (2017).

[63] Fridman, L.: MIT 6.S094: Deep Learning for Self-Driving Cars. Lecture 2: Self-Driving Cars.

[64] Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y.: OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In: CoRR abs/1312.6229 (2013).

[65] Girshick, R.; Donahue, J.; Darrell, T.; Malik, J.: Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. Washington, DC, USA 2014.

[66] Ren, S.; He. K.; Girshick, R. B.; Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: CoRR abs/1506.01497 (2015).

[67] Redmon, J.; Divvala, S. K.; Girshick, R. B.; Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection. In: CoRR abs/1506.02640 (2015).

[68] Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A. C.: SSD: Single Shot MultiBox Detector. In: Leibe, B.; Matas, J.; Sebe, N.; Welling, M. (Hrsg.): Computer Vision - ECCV 2016. Cham 2016.

[69] Huang, J.; Rathod, V.; Sun, C.; Zhu, M.; Korattikara, A.; Fathi, A.; Fischer, I.; Wojna, Z.; Song, Y.; Guadarrama, S.; Murphy, K.: Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[70] Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C. L.: Microsoft COCO: Common Objects in Context. In: Fleet, D.; Pajdla, T.; Schiele, B.; Tuytelaars, T. (Hrsg.): Computer Vision - ECCV 2014. Cham 2014.

[71] Janai, J.; Güney, F.; Behl, A.; Geiger, A.: Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art. In: CoRR abs/1704.05519 (2017).

[72] Shotton, J.; Winn, J.; Rother, C.; Criminisi, A.: TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context. In: International Journal of Computer Vision 81 (2009) 1, S. 2–23.

[73] Ladicky, L.; Russell, C.; Kohli, P.; Torr, Philip H. S.: Graph Cut Based Inference with Co-occurrence Statistics. In: Daniilidis, K.; Maragos, P.; Paragios, N. (Hrsg.): Computer Vision - ECCV 2010. Berlin, Heidelberg 2010.

[74] Rabinovich, A.; Vedaldi, A.; Galleguillos, C.; Wiewiora, E.; Belongie, S.: Objects in Context: 2007 IEEE 11th International Conference on Computer Vision.

[75] Long, J.; Shelhamer, E.; Darrell, T.: Fully convolutional networks for semantic segmentation: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[76] Everingham, M.; Van Gool, L.; Williams, C.; Winn, J.; Zisserman, A.: The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results.

[77] Everingham, M.; Van Gool, L.; Williams, C.; Winn, J.; Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.

[78] Garcia-Garcia, A.; Orts-Escolano, S.; Oprea, S.; Villena-Martinez, V.; García Rodríguez, J.: A Review on Deep Learning Techniques Applied to Semantic Segmentation. In: CoRR abs/1704.06857 (2017).

[79] Badrinarayanan, V.; Kendall, A.; Cipolla, R.: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. In: CoRR abs/1511.00561 (2015).

[80] Brostow, G. J.; Fauqueur, J.; Cipolla, R.: Semantic Object Classes in Video: A High-definition Ground Truth Database. In: Pattern Recogn. Lett. 30 (2009) 2, S. 88–97.

[81] Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B.: The Cityscapes Dataset for Semantic Urban Scene Understanding: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016.

[82] Yu, F.; Koltun, V.: Multi-Scale Context Aggregation by Dilated Convolutions. In: CoRR abs/1511.07122 (2015).

[83] Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J.: Pyramid Scene Parsing Network. In: CoRR abs/1612.01105 (2016).

[84] Chen, L.-C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H.: Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In: ArXiv e-prints (2018).

[85] Wang, P.; Chen, P.; Yuan, Y.; Liu, D.; Huang, Z.; Hou, X.; Cottrell, G.: Understanding Convolution for Semantic Segmentation. In: ArXiv e-prints (2017).

[86] Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B.: Cityscapes Dataset - Semantic Understanding of Urban Street Scenes. Bench-

mark Suite - Pixel-Level Semantic Labeling Task. URL: https://www.cityscapes-dataset.com/benchmarks/#pixel-level-results. Abrufdatum 11.05.2018.

[87] Gal, Y.; Ghahramani, Z.: Dropout As a Bayesian Approximation: Representing Model Uncertainty in Deep Learning: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 2016.

[88] Kendall, A.; Badrinarayanan, V.; Roberto Cipolla: Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding. In: CoRR abs/1511.02680 (2015).

[89] Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A. L.: DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. In: CoRR abs/1606.00915 (2016).

[90] Lin, G.; Shen, C.; Reid, I. D.; van den Hengel, A.: Efficient piecewise training of deep structured models for semantic segmentation. In: CoRR abs/1504.01013 (2015).

[91] Mousavian, A.; Pirsiavash, H.; Kosecka, J.: Joint Semantic Segmentation and Depth Estimation with Deep Convolutional Networks. In: CoRR abs/1604.07480 (2016).

[92] Liu, Z.; Li, X.; Luo, P.; Loy, C.-C.; Tang, X.: Semantic Image Segmentation via Deep Parsing Network: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). Washington, DC, USA 2015.

[93] Zheng, S.; Jayasumana, S.; Romera-Paredes, B.; Vineet, V.; Su, Z.; Du, D.; Huang, C.; Torr, Philip H. S.: Conditional Random Fields As Recurrent Neural Networks: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). Washington, DC, USA 2015.

[94] Fan, H.; Ling, H.: Dense Recurrent Neural Networks for Scene Labeling. In: ArXiv e-prints (2018).

[95] Arnab, A.; Zheng, S.; Jayasumana, S.; Romera-Paredes, B.; Larsson, M.; Kirillov, A.; Savchynskyy, B.; Rother, C.; Kahl, F.; Torr, P. H.: Conditional Random Fields Meet Deep Neural Networks for Semantic Segmentation: Combining Probabilistic Graphical Models with Deep Learning for Structured Prediction. In: IEEE Signal Processing Magazine 35 (2018) 1, S. 37–52.

[96] Liu, F.; Lin, G.; Shen, C.: CRF learning with CNN features for image segmentation. In: Pattern Recognition 48 (2015) 10, S. 2983–92.

[97] Rota Bulò, S.; Porzi, L.; Kontschieder, P.: In-Place Activated BatchNorm for Memory-Optimized Training of DNNs. In: ArXiv e-prints (2017).

[98] Chen, L.-C.; Papandreou, G.; Schroff, F.; Adam, H.: Rethinking Atrous Convolution for Semantic Image Segmentation. In: CoRR abs/1706.05587 (2017).

[99] Atul Kanaujia; Ping Wang; Niels Haering: Markov Logic Networks for Scene Interpretation and Complex Event Recognition in Videos 2015.

[100] Ruder, S.: An Overview of Multi-Task Learning in Deep Neural Networks. In: ArXiv e-prints (2017).

[101] Marvin Teichmann; Michael Weber; J. Marius Zöllner; Roberto Cipolla; Raquel Urtasun: MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving. In: CoRR abs/1612.07695 (2016).

[102] Kendall, A.; Gal, Y.; Cipolla, R.: Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In: ArXiv e-prints (2017).

[103] Liao, Y.; Kodagoda, S.; Wang, Y.; Shi, L.; Liu, Y.: Understand scene categories by objects: A semantic regularized scene classifier using Convolutional Neural Networks: 2016 IEEE International Conference on Robotics and Automation (ICRA).

[104] Liang, M.; Hu, X. (Hrsg.): Recurrent convolutional neural network for object recognition, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015.

[105] Gatta, C.; Romero, A.; van de Veijer, J.: Unrolling Loopy Top-Down Semantic Feedback in Convolutional Deep Networks: 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).

[106] Chen, B.-k.; Gong, C.; Yang, J.: Importance-Aware Semantic Segmentation for Autonomous Driving System. In: Bacchus, F.; Sierra, C. (Hrsg.): Twenty-Sixth International Joint Conference on Artificial Intelligence.

[107] Hendrik Metzen, J.; Chaithanya Kumar, M.; Brox, T.; Fischer, V.: Universal Adversarial Perturbations Against Semantic Image Segmentation. In: ArXiv e-prints (2017).

[108] Reichel, M.; Botsch, M.; Rauschecker, R.; Siedersberger, K.-H.; Maurer, M.: Situation aspect modelling and classification using the Scenario Based Random Forest algorithm for convoy merging situations: 2010 13th International IEEE Conference on Intelligent Transportation Systems - (ITSC 2010).

[109] Hull, R. D.; Jenkins, D.; McCutchen, A.: Semantic Enrichment and Fusion Of Multi-Intelligence Data. In: White Paper 27 (2006).

[110] S. Ulbrich; T. Nothdurft; M. Maurer; P. Hecker: Graph-based context representation, environment modeling and information aggregation for automated driving: 2014 IEEE Intelligent Vehicles Symposium Proceedings 2014.

[111] Hülsen, M.; Zöllner, J. M.; Weiss, C. (Hrsg.): Traffic intersection situation description ontology for advanced driver assistance, Intelligent Vehicles Symposium (IV), 2011 IEEE 2011.

[112] Hummel, B.: Description Logic for Scene Understanding at the Example of Urban Road Intersections. Karlsruhe 2009.

[113] Nienhüser, D.; Zöllner, J. M.: Kontextsensitive Erkennung und Interpretation fahrrelevanter statischer Verkehrselemente, Karlsruhe, Karlsruher Institut für Technologie (KIT), Diss., 2014. Karlsruhe 2014.

[114] Johnson, J.; Krishna, R.; Stark, M.; Li, L.-J.; Shamma, D. A.; Bernstein, M. S.; Fei-Fei, L.: Image retrieval using scene graphs: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[115] Galleguillos, C.; Belongie, S.: Context Based Object Categorization: A Critical Survey. In: Comput. Vis. Image Underst. 114 (2010) 6, S. 712–22.

[116] Choi, M. J.; Torralba, A.; Willsky, A. S.: Context models and out-of-context objects. In: Pattern Recognition Letters 33 (2012) 7, S. 853–62.

[117] Ruiz-Sarmiento, J. R.; Galindo, C.; Gonzalez-Jimenez, J.: Probability and Common-Sense: Tandem Towards Robust Robotic Object Recognition in Ambient Assisted Living. In: García, C. R.; Caballero-Gil, P.; Burmester, M.; Quesada-Arencibia, A. (Hrsg.): Ubiquitous Computing and Ambient Intelligence. Cham 2016.

[118] Evans, L.; Schwing, R. C. (Hrsg.): Human Behavior and Traffic Safety. Boston, MA 1986.

[119] Abramson, Y.; Steux, B.: Hardware-friendly pedestrian detection and impact prediction: IEEE Intelligent Vehicles Symposium, 2004.

[120] Bertozzi, M.; Broggi, A.; Fascioli, A.; Tibaldi, A.; Chapuis, R.; Chausse, F.: Pedestrian localization and tracking systen with kalman filtering: IEEE Intelligent Vehicles Symposium, 2004.

[121] Chen, S. Y.: Kalman Filter for Robot Vision. A Survey. In: IEEE Transactions on Industrial Electronics 59 (2012) 11, S. 4409–20.

[122] Quintero, R.; Parra, I.; Llorca, D. F.; Sotelo, M. A.: Pedestrian Intention and Pose Prediction through Dynamical Models and Behaviour Classification: 2015 IEEE 18th International Conference on Intelligent Transportation Systems - (ITSC 2015).

[123] Gindele, T.; Brechtel, S.; Dillmann, R.: Learning context sensitive behavior models from observations for predicting traffic situations: 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013) 2013.

[124] Nagel, H.-H.: A vision of 'Vision and language' comprises action: An example from road traffic. In: Artificial Intelligence Review 8 (1994) 2-3, S. 189–214.
[125] Rehder, E.; Kloeden, H.: Goal-Directed Pedestrian Prediction: 2015 IEEE International Conference on Computer Vision Workshop (ICCVW).
[126] Schreier, M.; Willert, V.; Adamy, J.: An Integrated Approach to Maneuver-Based Trajectory Prediction and Criticality Assessment in Arbitrary Road Environments. In: IEEE Transactions on Intelligent Transportation Systems 17 (2016) 10, S. 2751–66.
[127] Kuhnt, F.; Schulz, J.; Schamm, T.; Zollner, J. M.: Understanding interactions between traffic participants based on learned behaviors: 2016 IEEE Intelligent Vehicles Symposium (IV).
[128] Wiest, J.; Karg, M.; Kunz, F.; Reuter, S.; Kresel, U.; Dietmayer, K.: A probabilistic maneuver prediction framework for self-learning vehicles with application to intersections: 2015 IEEE Intelligent Vehicles Symposium (IV).
[129] Bahram, M.; Hubmann, C.; Lawitzky, A.; Aeberhard, M.; Wollherr, D.: A Combined Model- and Learning-Based Framework for Interaction-Aware Maneuver Prediction. In: IEEE Transactions on Intelligent Transportation Systems 17 (2016) 6, S. 1538–50.
[130] Souza, C. R. C.; Santos, P. E.: Probabilistic Logic Reasoning About Traffic Scenes: Proceedings of the 12th Annual Conference on Towards Autonomous Robotic Systems. Berlin, Heidelberg 2011.
[131] Arens, M.; Ottlik, A.; Nagel, H.-H.: Using Behavioral Knowledge for Situated Prediction of Movements. In: Hutchison, D.; Kanade, T.; Kittler, J.; Kleinberg, J. M.; Mattern, F.; Mitchell, J. C.; Naor, M.; Nierstrasz, O.; Pandu Rangan, C.; Steffen, B.; Sudan, M.; Terzopoulos, D.; Tygar, D.; Vardi, M. Y.; Weikum, G.; Biundo, S.; Frühwirth, T.; Palm, G. (Hrsg.): KI 2004: Advances in Artificial Intelligence. Berlin, Heidelberg 2004.
[132] Geng, X.; Liang, H.; Yu, B.; Zhao, P.; He, L.; Huang, R.: A Scenario-Adaptive Driving Behavior Prediction Approach to Urban Autonomous Driving. In: Applied Sciences 7 (2017) 4, S. 426.
[133] Armand, A.; Filliat, D.; Ibanez-Guzman, J.: Ontology-based context awareness for driving assistance systems: 2014 IEEE Intelligent Vehicles Symposium (IV).
[134] Choi, M. J.; Torralba, A.; Willsky, A. S.: A Tree-Based Context Model for Object Recognition. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (2012) 2, S. 240–52.
[135] Torralba, A.; Murphy, K. P.; Freeman, W. T.: Using the forest to see the trees: exploiting context for visual object detection and localization. In: Commun. ACM (2010).
[136] Gould, S.; Fulton, R.; Koller, D. (Hrsg.): Decomposing a scene into geometric and semantically consistent regions, 2009 IEEE 12th International Conference on Computer Vision 2009.
[137] Yao, J.; Fidler, S.; Urtasun, R. (Hrsg.): Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation, Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on 2012.
[138] Geiger, A.; Lauer, M.; Urtasun, R. (Hrsg.): A generative model for 3D urban scene understanding from movable platforms, Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on 2011.
[139] Gould, S.: Probabilistic Models for Region-based Scene Understanding 2010.
[140] Hensel, I.: Probabilistisch-logische Inferenz relationaler Situationsbeschreibungen aus Verkehrsbildfolgen: DNB 2013.
[141] Stiller, C.; Kammel, S.; Lulcheva, I.; Ziegler, J.: Probabilistische Methoden in der Umfeldwahrnehmung Kognitiver Automobile Probabilistic Methods for Environment Perception of Cognitive Automobiles. In: at-Automatisierungstechnik Methoden und Anwendungen der Steuerungs-, Regelungs-und Informationstechnik 56 (2008) 11, S. 563–74.
[142] Mariusz Bojarski; Davide Del Testa; Daniel Dworakowski; Bernhard Firner; Beat Flepp; Prasoon Goyal; Lawrence D. Jackel; Mathew Monfort; Urs Muller; Jiakai Zhang; Xin Zhang; Jake Zhao; Karol Zieba: End to End Learning for Self-Driving Cars (2016).

[143] Mariusz Bojarski; Yeres, P.; Choromanska, A.; Choromanski, K.; Bernhard Firner; Lawrence D. Jackel; Urs Muller: Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car. In: CoRR abs/1704.07911 (2017).
[144] Pei, K.; Cao, Y.; Yang, J.; Jana, S.: DeepXplore: Automated Whitebox Testing of Deep Learning Systems: Proceedings of the 26th Symposium on Operating Systems Principles. New York, NY, USA 2017.
[145] Hubschneider, C.; Bauer, A.; Doll, J.; Weber, M.; Klemm, S.; Kuhnt, F.; Zollner, J. M.: Integrating end-to-end learned steering into probabilistic autonomous driving: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC).
[146] Shalev-Shwartz, S.; Shammah, S.; Shashua, A.: Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving. In: CoRR abs/1610.03295 (2016).
[147] Pekezou Fouopi, P.; Srinivas, G.; Knake-Langhorst, S.; Köster, F.; Niemeijer, J.: Holistische Szenenmodellierung und -Interpretation basierend auf subsymbolischen, symbolischen und probabilistischen Methoden: VDI-Fachkonferenz Umfelderfassung im Fahrzeug 2018.
[148] Bohlken, W.; Menzel, W.: Realzeit-Szeneninterpretation mit ontologiebasierten Regeln, Universität Hamburg, FB Informatik, Diss.--Hamburg, 2012. Hamburg 2013.
[149] Yang, Y.; Calmet, J.: OntoBayes: An Ontology-Driven Uncertainty Model: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-1 (CIMCA-IAWTIC'06) - Volume 01. Washington, DC, USA 2005.
[150] Carvalho, R. N.: Probabilistic Ontology: Representation and Modeling Methodology 2011.
[151] Neuhold, G.; Ollmann, T.; Bulò, S. R.; Kontschieder, P.: The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes: 2017 IEEE International Conference on Computer Vision (ICCV) 2017.
[152] Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R.: Vision Meets Robotics: The KITTI Dataset. In: Int. J. Rob. Res. 32 (2013) 11, S. 1231–37.
[153] Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; Darrell, T.: Caffe: Convolutional Architecture for Fast Feature Embedding: Proceedings of the 22Nd ACM International Conference on Multimedia. New York, NY, USA 2014.
[154] Shelhamer, E.: Github Project FCN. URL: https://github.com/shelhamer/fcn.berkeleyvision.org.
[155] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M. S.; Berg, A. C.; Li, F.: ImageNet Large Scale Visual Recognition Challenge. In: CoRR abs/1409.0575 (2014).
[156] GEFORCE GTX 1080 Ti. GEFORCE IN PERFEKTION. URL: http://www.nvidia.de/graphics-cards/geforce/pascal/gtx-1080-ti/.
[157] URL: https://www.cityscapes-dataset.com/method-details/?submissionID=38.
[158] Althoff, M.; Hess, D.; Gambert, F.: Road occupancy prediction of traffic participants: 2013 16th International IEEE Conference on Intelligent Transportation Systems - (ITSC 2013).
[159] Pekezou Fouopi, P.; Thomaidis, G.; Knake-Langhorst, S.; Köster, F.: Unified situation modeling and understanding using hierarchical graphical Model: 6. VDI Tagung Optische Technologien in der Fahrzeugtechnik 2014.
[160] Thrun, S.: Robotic Mapping: A Survey. Exploring Artificial Intelligence in the New Millennium. In: Lakemeyer, G.; Nebel, B. (Hrsg.). San Francisco, CA, USA 2003.
[161] Badino, H.; Franke, U.; Mester, R.: Free Space Computation Using Stochastic Occupancy Grids and Dynamic: Programming," Proc. Int'l Conf. Computer Vision, Workshop Dynamical Vision 2007.
[162] Lee, J.: Fast INT8 Inference for Autonomous Vehicles with TensorRT 3. URL: https://devblogs.nvidia.com/int8-inference-autonomous-vehicles-tensorrt/.

[163] NVIDIA TensorRT. Programmable Inference Accelerator.
URL: https://developer.nvidia.com/tensorrt. Abrufdatum 20.09.2018.
[164] NVIDIA DRIVE PX. Scalable AI Supercomputer For Autonomous Driving.
URL: https://www.nvidia.com/en-au/self-driving-cars/drive-px/. Abrufdatum 20.09.2018.
[165] Chen, L.-C.; Zhu, Y.; Papandreou, G.; Hui, H.: TensorFlow DeepLab Model Zoo. DeepLab models trained on Cityscapes.
URL: https://github.com/tensorflow/models/tree/master/research/deeplab. Abrufdatum 22.11.2018.
[166] Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q. V.: Learning Transferable Architectures for Scalable Image Recognition. In: ArXiv e-prints (2017).
[167] Wu, B.; Iandola, F. N.; Jin, P. H.; Keutzer, K.: SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving: CVPR Workshops 2017.
[168] Liu, C.; Chen, L.-C.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A.; Fei-Fei, L.: Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation. In: arXiv preprint arXiv:1901.02985 (2019).
[169] Chen, L.-C.; Collins, M.; Zhu, Y.; Papandreou, G.; Zoph, B.; Schroff, F.; Adam, H.; Shlens, J.: Searching for efficient multi-scale architectures for dense image prediction: Advances in Neural Information Processing Systems 2018.
[170] Frankle, J.; Carbin, M.: The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks: International Conference on Learning Representations 2019.
[171] Zhou, H.; Lan, J.; Liu, R.; Yosinski, J.: Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask. In: ArXiv e-prints (2019), S. arXiv:1905.01067.
[172] Luc, P.; Neverova, N.; Couprie, C.; Verbeek, J.; LeCun, Y.: Predicting Deeper into the Future of Semantic Segmentation: 2017 IEEE International Conference on Computer Vision (ICCV).
[173] Mathieu, M.; Couprie, C.; LeCun, Y.: Deep multi-scale video prediction beyond mean square error. In: CoRR abs/1511.05440 (2015).
[174] Lotter, W.; Kreiman, G.; Cox, D.: A neural network trained to predict future video frames mimics critical properties of biological neuronal responses and perception. In: ArXiv e-prints (2018).
[175] Villegas, R.; Yang, J.; Zou, Y.; Sohn, S.; Lin, X.; Lee, H.: Learning to Generate Long-term Future via Hierarchical Prediction. In: CoRR abs/1704.05831 (2017).
[176] Tsung-Yi Lin; Goyal, P.; Girshick, R.; Kaiming He; Piotr Dollár: Focal Loss for Dense Object Detection. In: 2017 IEEE International Conference on Computer Vision (ICCV) (2017), S. 2999–3007.
[177] Salehi, Seyed Sadegh Mohseni; Erdogmus, D.; Gholipour, A.: Tversky Loss Function for Image Segmentation Using 3D Fully Convolutional Deep Networks. In: Wang, Q.; Shi, Y.; Suk, H.-I.; Suzuki, K. (Hrsg.): Machine Learning in Medical Imaging. Cham 2017.
[178] Bulò, S.; Neuhold, G.; Kontschieder, P.: Loss Max-Pooling for Semantic Image Segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017), S. 7082–91.
[179] Rahman, M. A.; Wang, Y.: Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation. In: Bebis, G.; Boyle, R.; Parvin, B.; Koracin, D.; Porikli, F.; Skaff, S.; Entezari, A.; Min, J.; Iwai, D.; Sadagic, A.; Scheidegger, C.; Isenberg, T. (Hrsg.): Advances in Visual Computing. Cham 2016.
[180] Kampffmeyer, M.; Salberg, A.-B.; Jenssen, R.: Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks: Computer Vision and Pattern Recognition Workshops (CVPRW), 2016 IEEE Conference on 2016.

[181] Sabour, S.; Frosst, N.; Hinton, G. E.: Dynamic Routing Between Capsules. In: I. Guyon; U. V. Luxburg; S. Bengio; H. Wallach; R. Fergus; S. Vishwanathan; R. Garnett (Hrsg.): Advances in Neural Information Processing Systems 30 2017.
[182] Ratner, A.; Bach, S. H.; Ehrenberg, H. R.; Fries, J. A.; Wu, S.; Ré, C.: Snorkel: Rapid Training Data Creation with Weak Supervision. In: CoRR abs/1711.10160 (2017).
[183] Watters, N.; Tacchetti, A.; Weber, T.; Pascanu, R.; Battaglia, P.; Zoran, D.: Visual Interaction Networks. In: ArXiv e-prints (2017).
[184] Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; Hassabis, D.; Clopath, C.; Kumaran, D.; Hadsell, R.: Overcoming catastrophic forgetting in neural networks. In: Proceedings of the National Academy of Sciences of the United States of America 114 (2017) 13, S. 3521–26.
[185] Jo, J.; Bengio, Y.: Measuring the tendency of CNNs to Learn Surface Statistical Regularities. In: CoRR abs/1711.11561 (2017).
[186] Ilyas, A.; Santurkar, S.; Tsipras, D.; Engstrom, L.; Tran, B.; Madry, A.: Adversarial Examples Are Not Bugs, They Are Features. In: ArXiv e-prints (2019), S. arXiv:1905.02175.
[187] Hosseini, H.; Poovendran, R.: Deep Neural Networks Do Not Recognize Negative Images. In: CoRR abs/1703.06857 (2017).
[188] Evtimov, I.; Eykholt, K.; Fernandes, E.; Kohno, T.; Li, B.; Prakash, A.; Rahmati, A.; Song, D.: Robust Physical-World Attacks on Machine Learning Models. In: CoRR abs/1707.08945 (2017).
[189] Raghunathan, A.; Steinhardt, J.; Liang, P.: Certified Defenses against Adversarial Examples. In: CoRR abs/1801.09344 (2018).
[190] Yuan, X.; He, P.; Zhu, Q.; Bhat, R. R.; Li, X.: Adversarial Examples: Attacks and Defenses for Deep Learning. In: CoRR abs/1712.07107 (2017).
[191] Carlini, N.; Wagner, D.: Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. New York, NY, USA 2017.
[192] Tian, Y.; Pei, K.; Jana, S.; Ray, B.: DeepTest: Automated Testing of Deep-neural-network-driven Autonomous Cars: Proceedings of the 40th International Conference on Software Engineering. New York, NY, USA 2018.
[193] Tjeng, V.; Xiao, K. Y.; Tedrake, R.: Evaluating Robustness of Neural Networks with Mixed Integer Programming: International Conference on Learning Representations 2019.
[194] Zhang, Q.-s.; Zhu, S.-C.: Visual interpretability for deep learning: a survey. In: Frontiers of Information Technology & Electronic Engineering 19 (2018) 1, S. 27–39.
[195] Zhang, Q.; Wu, Y. N.; Zhu, S.-C.: Interpretable Convolutional Neural Networks. In: CoRR abs/1710.00935 (2017).
[196] Bau, D.; Zhu, J.-Y.; Strobelt, H.; Zhou Bolei; Tenenbaum, J. B.; Freeman, W. T.; Torralba, A.: GAN Dissection: Visualizing and Understanding Generative Adversarial Networks. In: arXiv preprint arXiv:1811.10597 (2018).
[197] Rahwan, I.; Cebrian, M.; Obradovich, N.; Bongard, J.; Bonnefon, J.-F.; Breazeal, C.; Crandall, J. W.; Christakis, N. A.; Couzin, I. D.; Jackson, M. O.; Jennings, N. R.; Kamar, E.; Kloumann, I. M.; Larochelle, H.; Lazer, D.; McElreath, R.; Mislove, A.; Parkes, D. C.; Pentland, A. '.; Roberts, M. E.; Shariff, A.; Tenenbaum, J. B.; Wellman, M.: Machine behaviour. In: Nature 568 (2019) 7753, S. 477–86.
[198] Musen, M. A.; the Protégé Team: The Protégé Project: A Look Back and a Look Forward. In: AI matters 1 (2015) 4, S. 4–12.
[199] Horrocks, I.; F. Patel-Schneider, P.; Boley, H.; Tabet, S.; Grosof, B.; Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. In: W3C Member submission 21 (2004).
[200] Frederick Sander: Object Classification using Probabilistic Graphical Model 2016.

[201] Kok, S.; Sumner, M.; Richardson, M.; Singla, P.; Poon, H.; Lowd, D.; Domingos, P.: The Alchemy system for statistical relational AI. Technical report (2007).
URL: http://alchemy.cs.washington.edu. Abrufdatum 24.08.2016.

[202] Galleguillos, C.; Rabinovich, A.; Belongie, S.: Object categorization using co-occurrence, location and appearance: 2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[203] Galleguillos, C.: Beyond Appearance Features: Contextual Modeling for Object Recognition. La Jolla, CA, USA 2011.

[204] Poon, H.; Domingos, P.: Sound and Efficient Inference with Probabilistic and Deterministic Dependencies: Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1 2006.

[205] Aguiar, E. F.; Ladeira, M.; Carvalho, R. N.; Matsumoto, S.: A GUI for MLN: Proceedings of the 9th International Conference on Uncertainty Reasoning for the Semantic Web - Volume 1073. Aachen, Germany, Germany 2013.

[206] Niu, F.; Ré, C.; Doan, A.; Shavlik, J.: Tuffy: Scaling Up Statistical Inference in Markov Logic Networks Using an RDBMS. In: Proc. VLDB Endow. 4 (2011) 6, S. 373–84.

[207] Beedkar, K.; Corro, L. D.; Gemulla, R.: Fully Parallel Inference in Markov Logic Networks.

[208] Wittek, P.; Gogolin, C.: Quantum Enhanced Inference in Markov Logic Networks. In: Scientific Reports 7 (2017), S. 45672 EP -.

[209] Singla, P.; Mooney, R. J.: Abductive Markov Logic for Plan Recognition: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence 2011.

[210] Bodart, A.; Evrard, K.; Ortiz, J.; Schobbens, P.-Y.: ArThUR: A Tool for Markov Logic Network. In: Meersman, R.; Panetto, H.; Mishra, A.; Valencia-García, R.; Soares, A. L.; Ciuciu, I.; Ferri, F.; Weichhart, G.; Moser, T.; Bezzi, M.; Chan, H. (Hrsg.): On the Move to Meaningful Internet Systems: OTM 2014 Workshops. Berlin, Heidelberg 2014.

[211] Noessner, J.; Niepert, M.; Stuckenschmidt, H.: RockIt: Exploiting Parallelism and Symmetry for MAP Inference in Statistical Relational Models: Proceedings of the 16th AAAI Conference on Statistical Relational Artificial Intelligence 2013.

[212] Lapoehn, S.; Pekezou Fouopi, P.; Löper, C.; Knake-Langhorst, S.; Hesse, T.: Semantische Netze als Wissensbasis automatisierter Fahrzeuge: VDI/VW-Gemeinschaftstagung: Fahrassistenzsysteme und automatisiertes Fahren 2016.

[213] Krötzsch, M.: Description logic rules. Heidelberg, Germany 2010.

[214] Plewka, M.: Lehrbuch Fahrschule. Klassen , B, BE & A, A1, M, S + L, T. Das Grund- und Zusatzwissen für die theoretische Prüfung, 8. Auflage 2009.

[215] Hu, Z.; Ma, X.; Liu, Z.; Hovy, E.; Xing, E.: Harnessing deep neural networks with logic rules. In: arXiv preprint arXiv:1603.06318 (2016).

[216] Hinton, G. E.; Krizhevsky, A.; Wang, S. D.: Transforming Auto-Encoders. In: Honkela, T.; Duch, W.; Girolami, M.; Kaski, S. (Hrsg.): Artificial Neural Networks and Machine Learning - ICANN 2011. Berlin, Heidelberg 2011.

[217] Nyga, D.: pracmln. Markov logic networks in Python. URL: http://www.pracmln.org/. Abrufdatum 17.11.2018.

[218] PEARL, J.: Theoretical Impediments to Machine Learning With Seven Sparks from the Causal Revolution: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. New York, NY, USA 2018.

[219] PEARL, J.: The Seven Tools of Causal Inference with Reflections on Machine Learning (2018).

[220] Selvaraju, R. R.; Chattopadhyay, P.; Elhoseiny, M.; Sharma, T.; Batra, D.; Parikh, D.; Lee, S.: Choose Your Neuron: Incorporating Domain Knowledge through Neuron-Importance. In: ArXiv e-prints (2018).

[221] Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V. F.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; Gülçehre, Ç.; Song, F.; Ballard, A. J.; Gilmer, J.; Dahl, G. E.; Vaswani, A.; Allen, K. R.; Nash, C.; Langston, V.; Dyer, C.; Heess, N.; Wierstra, D.; Kohli, P.; Botvinick, M.; Vinyals, O.; Li, Y.; Pascanu, R.: Relational inductive biases, deep learning, and graph networks. In: CoRR abs/1806.01261 (2018).

[222] Diligenti, M.; Roychowdhury, S.; Gori, M.: Integrating Prior Knowledge into Deep Learning: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA).

[223] Dong, H.; Mao, J.; Lin, T.; Wang, C.; Li, L.; Zhou, D.: Neural Logic Machines. In: ArXiv e-prints (2019), S. arXiv:1904.11694.

[224] VIEWCar II. Forschungsfahrzeug zur Analyse von Wahrnehmungsprozessen und Verhalten. URL: https://www.dlr.de/ts/desktopdefault.aspx/tabid-11367/19950_read-46559/. Abrufdatum 27.03.2019.

[225] Börner, A.; Baumbach, D.; Buder, M.; Choinowski, A.; Ernst, I.; Funk, E.; Grießbach, D.; Schischmanow, A.; Wohlfeil, J.; Zuev, S.: IPS – a vision aided navigation system. In: Advanced Optical Technologies 6 (2017) 2, S. 674.