UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
NEURO- UND BIOINFORMATIK

Semantische Instanzsegmentierung basierend auf Deep Learning im Kontext des automatisierten Fahrens unter Berücksichtigung der Anwendbarkeit auf medizinische Daten

Semantic Instance Segmentation based on Deep Learning in context of automated driving considering the applicability to medical data

Masterarbeit
im Rahmen des Studienganges Medizinische Informatik
der Universität zu Lübeck

vorgelegt von
Joshua Niemeijer

ausgegeben und betreut von
Prof. Dr.-Ing. Erhardt Barth

mit Unterstützung von
M. Sc. Paulin Pekezou Fouopi

Die Masterarbeit ist im Rahmen von Arbeiten am Institut für Verkehrssystemtechnik
am DLR Braunschweig entstanden

Lübeck, den 15. Dezember 2017

Ich versichere an Eides statt, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

Lübeck, den 15. Dezember 2017

Kurzfassung

Die Grundlage, auf der das Selbstfahrende Auto Aktionen plant, bildet ein umfassendes Modell der umgebenden Szene. Eine der Hauptinformationsquellen für semantische Informationen über die umgebende Szene sind Kamerabilder. In dieser Arbeit wurde ein Neuronales Netzwerk entwickelt, welches jedem Pixel eines Kamerabildes eine semantische Klasse zuweist und innerhalb der Klasse zwischen Instanzen unterscheiden kann (Instanzsegmentierung). Der Ansatz der Instanzsegmentierung basiert auf einer Architektur, die eine Semantische Segmentierung, eine Objektdetektion und eine pixelweise Bestimmung relativer Positionen innerhalb einer Instanz auf einer geteilten Feature-Map bestimmt, was zu einer besseren Ausnutzung der begrenzten Ressourcen führt. Eine Evaluation des Trainings der verschiedenen Aufgaben auf Basis desselben Netzwerkes zur Erstellung der Feature-Map ergab, dass hieraus eine Verschlechterung der Detektion im Verhältnis zum Einzelnetzwerk folgt. Durch neue Netzwerkstrukturen, die die pixelweise Semantische Segmentierung als weitere Informationsgrundlage verwenden, konnte dieser Nachteil ausgeglichen und die Detektion sogar verbessert werden. Die Qualität der Instanzsegmentierung liegt beim Cityscapes-Benchmark im Mittelfeld aller Ansätze. Verbesserungsmöglichkeiten liegen bei der Prädiktion der pixelweisen Informationen und beim Kriterium, das die Semantische Segmentierung, die relativen Positionen und die Objektdetektion zu einer Instanzsegmentierung vereint. Eine Anwendung des Netzwerks auf medizinische Daten erweist sich ebenfalls als sinnvoll.

Abstract

The autonomous car plans it's actions based on a model describing the surrounding scene. A main source for semantic information to create such a scene model are camera pictures. In this thesis a neural network was created, which assigns every pixel the semantic class and the instance label it is part of (Instance Segmentation). The approach is based on an architecture, that infers on a shared feature-map a semantic segmentation, an object detection and a pixelwise prediction of the relative positions inside an instance. This leads to a better use of the limited resources, the evaluation however showed that the detection quality suffers. By creating new network structures providing the Semantic Segmentation as an input for the detection this disadvantage was compensated and the detection quality was improved. Instance Segmentation offers average results on the Cityscapes Benchmark. The evaluation shows that especially the prediction of the pixelwise information and the criterion for combining the Semantic Segmentation, Object Detection and Relative Position prediction to an instance segmentation has to be improved. An application of the method in the field of medicine turned out to be useful.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Fähigkeit eines Selbstfahrenden Autos, sich sicher durch den Verkehr zu manövrieren, basiert in erster Linie auf einem möglichst korrekten und umfassenden Modell der umgebenden Szene. Ein solches Modell basiert auf Daten von Sensoren wie Lidar, Radar oder Kamerabildern. Hierbei werden semantische Informationen über die Szene, also Informationen über die in der Umgebung präsenten Objekte, vornehmlich aus Kamerabildern abgeleitet. Dabei ist es zunächst einmal von Bedeutung, für jedes Pixel zu bestimmen, welcher Klasse es zugehörig ist. Dadurch kann das Selbstfahrende Auto z.B. erkennen, dass es gerade über eine Straße fährt oder sich einem Hindernis nähert. Eine solche Karte, die jedem Pixel seine semantische Klasse zuweist, nennt man auch eine Semantische Segmentierung, wie sie in Abbildung 2.5 zu sehen ist. Solch eine Semantische Segmentierung kann durch ein neuronales Netzwerk aus dem gegebenen Kamerabild abgeleitet werden. Die generelle pixelweise Architektur, welche in [SLD16] präsentiert wurde, und deren Erweiterungen liefern in diesem Bereich im Moment die besten Ergebnisse.

Was eine solche Semantische Segmentierung jedoch nicht kann, ist, zwischen Instanzen einer generellen Klasse zu unterscheiden. Eine Unterscheidung zwischen verschiedenen Instanzen ist jedoch vor allem bei Klassen wichtig, welche instanzspezifische Bewegungsmuster aufweisen, wie Autos, Personen, Fahrradfahrer etc. Auf Basis solcher instanzspezifischen Informationen kann das Selbstfahrende Auto dann entscheiden, wie es sich gegenüber den einzelnen Verkehrsteilnehmern verhält. Diese Erweiterung der Semantischen Segmentierung bezeichnet man als Semantische Instanzsegmentierung, wie sie z.B. in Abbildung 2.14 zu sehen ist. Für diese gibt es wiederum zwei hauptsächliche Ansätze, welche noch näher in Kapitel 2.4 erläutert werden. Dabei handelt es sich zum einen um die pixelweise Formulierung der Aufgabe, was in Anbetracht der Tatsache, dass eine solche im Zusammenhang mit der Semantischen Segmentierung die besten Ergebnisse liefert, sinnvoll erscheint. Zudem sollten durch eine solche Formulierung globale Informationen in die lokalen Klassifikationen mit einbezogen werden können. Das hauptsächliche Problem dieser Ansätze liegt jedoch in ihrer Komplexität, die es schwer macht, ein gutes Modell zu implementieren und zu trainieren. Zum anderen ist es möglich, die Aufgabe der Instanzsegmentierung als zweistufigen Prozess zu betrachten, in dem zunächst durch eine Detektion der grobe Bereich einer Instanz bestimmt wird und anschließend in diesem Bereich die Instanz segmentiert wird. Hierauf basierende Ansätze lassen sich aufgrund der geringen Komplexität der Modelle leicht implementieren und trainieren und liefern auch deshalb im Moment die besten Ergebnisse. Ein

Hauptproblem dabei ist jedoch, dass sich diese Ansätze nicht von fehlenden Detektionen erholen können. Mithin ist ein guter Detektor für diese Ansätze von großer Bedeutung.

Die Idee, die in dieser Masterarbeit umgesetzt wurde, orientiert sich am letzteren Ansatz. Diese besteht nun darin, eine Detektion und eine Semantische Segmentierung auf derselben Feature-Map zu trainieren, wobei die resultierende Semantische Segmentierung mit als weitere Grundlage für die Detektion verwendet wird. Durch das gemeinsame Training der beiden verwandten Aufgaben sollen dabei bessere Abstraktionen gelernt werden. Die Einbeziehung der Semantischen Segmentierung als Features für die Detektion soll bei der Klassifikation einer Bounding-Box helfen und mithin zu einer besseren Detektion führen. Die Semantische Segmentierung innerhalb einer Detektion eines Objektes einer bestimmten Klasse repräsentiert dann dabei die Instanz. Dieser Ansatz wird in Kapitel 3 weiter ausgeführt. Dort wird auch eine Methode eingeführt, wie im Falle von überlappenden Instanzen innerhalb der Bounding-Box eine Unterscheidung zwischen diesen erreicht werden kann. Die hier beschriebenen Methoden sind in ihrer Anwendung nicht alleine auf das Selbstfahrende Auto beschränkt, sondern können vor allem auch auf die medizinische Bildverarbeitung übertragen werden. So kann die hier entwickelte Methode z.B. für die Detektion und Segmentierung von krankhaft verändertem Gewebe in CT/MRT/Röntgen-Bildern verwendet werden. Eine tiefergehende Analyse über diese Möglichkeiten der Semantischen Instanz-Segmentierung oder allgemein der Semantischen Segmentierung ist in Kapitel 7 zu finden.

Kapitel 2

Grundlagen

In diesem Kapitel soll nun zunächst ein Überblick über die für das Verständnis der Arbeit wichtigen Ansätze der Semantischen Segmentierung, der Objektdetektion und der Semantischen Instanzsegmentierung gegeben werden. In Abschnitt 2.2 wird auf den generellen Ansatz der pixelweisen Semantischen Segmentierung, wie er in [SLD16] eingeführt wurde, eingegangen. Darauf aufbauend wird die spezielle Architektur dieses Ansatzes erläutert, welche auch in dem in dieser Arbeit entwickelten Ansatz verwendet wird. Abschnitt 2.4 gibt dann einen Überblick über die zur Zeit vorhandenen Methoden der Semantischen Instanzsegmentierung. Um die Wahl des Detektors in unserem Ansatz nachvollziehen zu können, werden zudem in Abschnitt 2.3 die wichtigsten Ansätze zur Lösung der Objektdetektion veranschaulicht. All diese Methoden der Semantischen Segmentierung, Detektion und Semantischen Instanzsegmentierung basieren auf künstlichen Neuronalen Netzwerken, weshalb in 2.1 die Grundlagen der Neuronalen Netzwerke präsentiert werden. In Kapitel 2.5 wird auf die zum Verständnis der Arbeit wichtigen Datensätze eingegangen und Qualitätsmetriken präsentiert, die verwendet wurden, um die Ergebnisse auf den einzelnen Aufgaben zu analysieren.

2.1 Neuronale Netzwerke

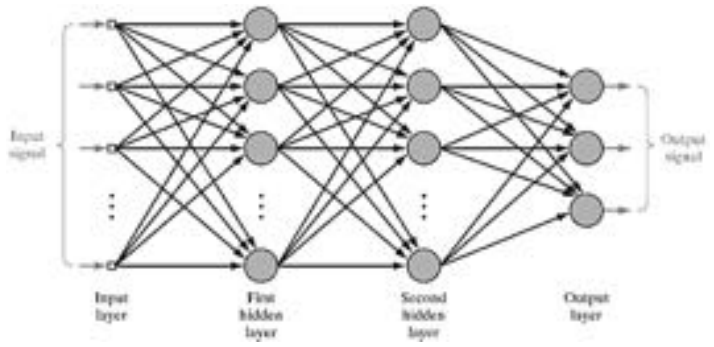

Abbildung 2.1: Die Abbildung zeigt die typische Struktur eines tiefen Neuronalen Netzwerks. Die Abbildung ist dabei entnommen aus http://adgefficiency.com/forecasting-uk-imbalance-price-using-a-multilayer-perceptron/ (abgerufen am 2.07.2017)

Die Verfahren, die in dieser Arbeit entwickelt wurden, basieren auf sogenannten tiefen Neuronalen Netzwerken. Die Informationen, die hier über diese Neuronalen Netzwerke

dargelegt werden, stammen aus [GBC16]. Diese sind an biologische Neuronale Netzwerke angelehnt und können allgemein zum Lösen von Klassifikations- und Regressionsaufgaben verwendet werden. Ein Beispiel für ein solches Neuronales Netzwerk ist z.B. in Abbildung 2.1 gegeben. Dieses kann generell als eine Transformation $f(x)$ eines Eingabevektors x auf einen gewünschten Wert y gesehen werden. Diese Transformation wird dabei durch viele Schichten von künstlichen Neuronen berechnet. Jeder Knoten in der Abbildung 2.1 repräsentiert dabei ein künstliches Neuron. Das Konzept eines solchen künstlichen Neurons ist in Abbildung 2.2 zu sehen. Dabei bekommt das Neuron (mit dem Index j) die Eingabe $x = [x_1, x_2, x_3, ..., x_n, x_{n+1}]$, welche mit dem Vektor $w_j = [w_{1j}, w_{2j}, w_{3j}..., w_{nj}, w_{(n+1)j}]$ gewichtet summiert wird. $w_{(n+1)j}$ bezeichnet dabei den Schwellenwert θ_j, welcher mit der gewichteten Summe der Eingabe durch die Multiplikation mit $x_{n+1} = 1$ addiert wird.

$$net_j = \sum_{k=0}^{n+1} w_{kj} x_k \tag{2.1}$$

Diese Summe wird weiterhin als net_j bezeichnet. Auf net_j wird dann eine nichtlineare Aktivierungsfunktion (oder auch Nichtlinearität) φ angewendet. Im Folgenden soll dabei $o_j = \varphi(net_j)$ die Aktivierung der gewichteten Summe bezeichnen.

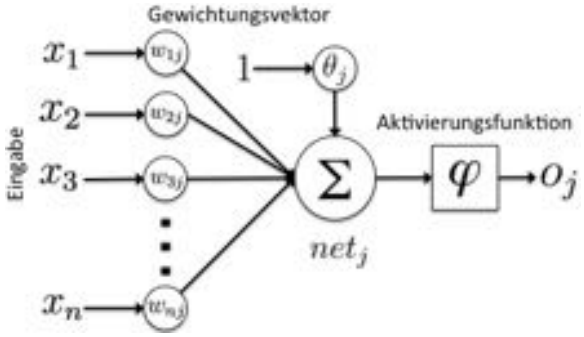

Abbildung 2.2: Diese Abbildung zeigt das Konzept eines künstlichen Neurons.

Das beschriebene Neuronenmodell hat in den tiefen Neuronalen Netzwerken, die in dieser Arbeit verwendet wurden, zwei Funktionen.

Versteckte Neuronen

In Abbildung 2.1 sind die versteckten Neuronen in den mit "hidden layer" bezeichneten Schichten zu finden. Jedes dieser Neuronen hat den Aufbau, welcher in Abbildung 2.2 beschrieben wurde. Dabei bekommt, wie man sieht, jedes Neuron einer Schicht die Ausgaben aller Neuronen der vorherigen Schicht als Eingaben. Eine solche Schicht wird auch als "Fully Connected Layer" bezeichnet. Bei der Nichtlinearität, die meist auf die Ausgaben der versteckten Neuronen angewendet wird, handelt es sich um eine sogenannte Rectified Linear Unit oder auch ReLU, welche in Abbildung 2.3 zu sehen und folgendermaßen definiert ist:

$$\varphi(net_j) = \max\{0, net_j\} \tag{2.2}$$

Die Anwendung der Nichtlinearität ist deshalb sinnvoll, da mehrere aufeinander folgende lineare Netzwerke (wie Fully Connected Layer) immer durch nur eine Fully Connected Layer ausgedrückt werden können. Durch die Nichtlinearität kann dann eine größere Variabilität an Transformationen dargestellt werden. Durch eine Aneinanderreihung von vielen dieser Fully Connected Layer werden dann Repräsentationen der Eingabe (Features) berechnet, auf deren Basis die Ausgabeneuronen das gegebene Problem lösen können.

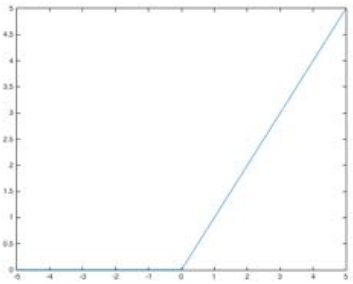

Abbildung 2.3: Diese Abbildung zeigt eine ReLU Aktivierungsfunktion.

Die Ausgabeneuronen

Auf Basis der Repräsentationen der Eingabe, die durch die versteckten Schichten berechnet wurde, kann dann die Transformation des Eingabevektors auf die gewünschte Ausgabe y berechnet werden. Wenn es sich bei der Aufgabe um ein Klassifikationsproblem handelt, so soll die Abbildung auf eine bestimmte Menge an n Klassen bestimmt werden. In diesem Fall existieren n Ausgabeneuronen, die jeweils einen Score-Wert für eine der n Klassen berechnen. Dabei dient jedes Neuron dazu, eine Hyperebene zu bestimmen, die eine binäre Klassifikation der ihm zugeordneten Klasse bestimmt (Klasse ja/nein). Diese Hyperebene wird durch den Gewichtungsvektor w_j (welcher den Schwellenwert θ_j beinhaltet) definiert. Der Score, den jedes Klassifikationsneuron für die Netzwerkeingabe x ausgibt, kann dann als eine Konfidenz gesehen werden, dass x zu der Klasse des Neurons gehört. Um nun die Wahrscheinlichkeitsverteilung über alle n Klassen zu berechnen, wird dann eine sogenannte Soft-Max Funktion verwendet. Dabei werden die Score-Werte aller Klassen auf den Wertebereich $[0, 1]$ abgebildet, sodass die Summe über alle Klassen 1 ergibt (eine mathematische Definition in Kapitel 2.1.4). Im Falle einer Regression kann y einen kontinuierlichen Wertebereich einnehmen. In diesem Fall wird nur ein Ausgabeneuron benötigt, welches durch die Gewichtungen w_j, die die Funktion net_j definieren, den Ausgabewert y approximiert.

2.1.1 Convolutional Neural Networks

Bei Convolutional Neural Networks (CNN) handelt es sich um eine spezielle Form der tiefen künstlichen Neuronalen Netzwerke, welche vom visuellen Kortex inspiriert ist. Diese Netzwerke sind folglich auf die Anwendung auf Bilder spezialisiert. Die generelle

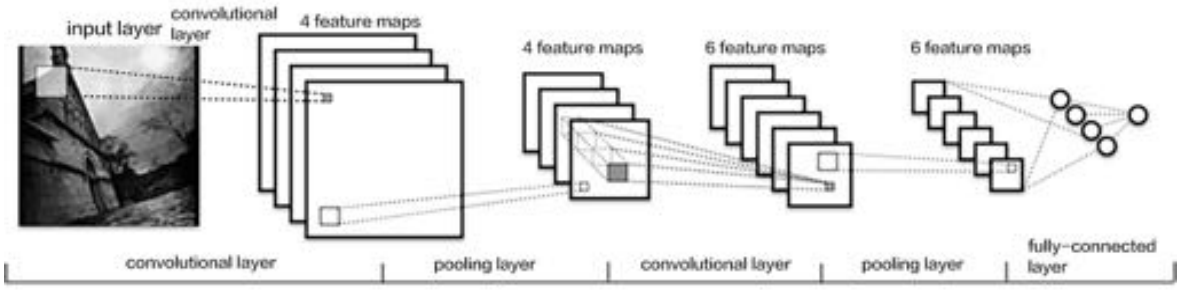

Abbildung 2.4: Die Abbildung zeigt die typische Struktur eines Convolutional Neural Networks. Die Abbildung ist dabei entnommen aus http://deeplearning.net/tutorial/lenet.html (abgerufen am 2.07.2017)

Struktur eines solchen Netzwerkes kann man in Abbildung 2.4 erkennen. Dabei besteht ein solches Netzwerk aus Faltungs- und Poolingschichten, durch die das Eingabebild verarbeitet wird.

Faltungsschichten

In Abbildung 2.4 ist die Anwendung von Faltungen (mit "convolutional layer" gekennzeichnet) dargestellt. Bei diesen Faltungen (oder auch Convolutions) handelt es sich um $k \times k$ Gitter, welche über die zweidimensionale Eingabe in einer bestimmten Schrittlänge geschoben werden. Dabei wird die gewichtete Summe aller Werte gebildet, die sich innerhalb des Gitters befinden. Wie man aus der Abbildung sehen kann, werden dabei die Werte aller Kanäle mit in die Berechnung einbezogen, sodass die Filterkerne die Form $k \times k \times AnzahlKanaele$ haben. Jeder Filterkern kann dabei als ein künstliches Neuron gesehen werden, bei dem die Gewichtungen des Filterkerns zu den Gewichtungen w_j des Neurons korrespondieren. Die Tiefe der Ausgabe einer Faltungsschicht entspricht dabei der Anzahl an Filterkernen, die auf die Eingabe angewendet wurden. Jeder Filterkern produziert dabei eine unterschiedliche Ausgabe auf Basis derselben Eingabe. Auf die Ausgabe der Faltungen wird dann eine Nichtlinearität angewendet, bei der es sich bei Nicht-Ausgabeneuronen meist um eine RELU (rectified linear unit) Funktion handelt.

Die Faltungsschichten unterscheiden sich von den Fully Connected Schichten dadurch, dass die Ausgabe der Faltungen nur von einem kleinen Teil der Eingabe abhängig ist und dadurch, dass für einen Filterkern die gleichen Gewichtungen auf die gesamte Eingabe angewendet werden. Diese spärlichen Verbindungen haben eine biologische Korrespondenz im Visuellen Kortex.

Durch eine Vielzahl an Faltungsschichten und Nichtlinearitäten werden die wichtigsten Eigenschaften (die Feature-Map) eines Bildes berechnet. Während des Trainings werden die Gewichtungen dabei so angepasst, dass die Features eine möglichst gute Datengrundlage für die Lösung des Problems bieten.

Pooling Schichten

In Abbildung 2.4 ist (mit "pooling layer" gekennzeichnet) die Anwendung von Pooling Operationen dargestellt. Diese bestehen aus einem $k \times k$ Gitter, welches in einer bestimmten Schrittlänge über die Feature-Maps geschoben wird. Dabei werden alle Werte, die in dieses Gitter fallen, zusammengefasst, indem z.B. nur der maximale Wert übernommen wird (als Max-Pooling bezeichnet) oder das Arithmetisches Mittel aller Werte berechnet wird (als Average-Pooling bezeichnet). Diese Operation wird getrennt auf allen Kanälen der Feature-Map ausgeführt, sodass das Pooling eine Reduktion der räumlichen Ausdehnung erreicht, die Anzahl der Kanäle jedoch konstant bleibt.

Aus dieser Reduktion folgt zunächst einmal eine Verringerung des Speicherbedarfes und eine schnellere Berechnungsgeschwindigkeit des Netzwerkes. Zudem wird durch das Pooling eine Erhöhung der Translationsinvarianz erreicht und die Komplexität der Feature-Map reduziert. Darüber hinaus folgt aus dem Pooling eine starke Vergrößerung des Rezeptiven Feldes.

Lösung der Aufgabe auf Basis der generierten Feature-Map

Durch die Anwendung der Pooling- und Faltungsschichten, wie sie in Abbildung 2.4 zu sehen ist, werden Feature-Maps erstellt, auf deren Basis ein Klassifikations- oder Regressions-Problem gelöst werden kann. Jedes räumliche Element wurde dabei auf Basis einer bestimmten Region im Eingabebild durch die Faltungen und Poolings berechnet. Diese Region nennt man das Rezeptive Feld. Nun gibt es zwei hauptsächliche Möglichkeiten, auf Basis der Feature-Maps das gegebene Problem zu lösen. Eine gängige Methode ist die Anwendung von Fully-Connected-Schichten, die die letzte Feature-Map als Eingabe erhalten. Die Klassifikation oder Regression wird dabei also auf Basis des gesamten Bildes inferiert. Ein Beispiel hierfür ist in der Abbildung 2.4 zu sehen. Eine andere Möglichkeit, welche häufig bei einer pixelweisen Klassifikation des Eingabebildes verwendet wird, ist, das Problem durch eine weitere Faltungsschicht zu lösen. Dabei werden die Entscheidungen also nur auf Basis der Rezeptiven Felder gefolgert. Solche Netzwerke, die keine Fully-Connected-Schichten besitzen, nennt man "fully convolutional".

2.1.2 Training Neuronaler Netzwerke

Das Training der Neuronalen Netzwerke wird über den Backpropagation Algorithmus realisiert. Dies geschieht auf Basis einer Menge an Trainingsdaten, bei der zu jedem Eingabevektor x die gewünschte Ausgabe y vorhanden ist (ground-truth Daten). Dabei wird auf Basis der Datengrundlage versucht, die Transformation, welche durch das Neuronale Netzwerk repräsentiert wird, so anzupassen, dass x auf y für alle Paare im Trainingsdatensatz abgebildet wird. Die Transformation soll dabei möglichst gut generalisieren, d.h. dass auch für Eingaben, die nicht Bestandteil der Trainingsmenge waren, die richtigen Ausgaben erzeugt werden.

Das Training wird durch die Minimierung einer Loss-Funktion E realisiert, welche den Fehler des Neuronalen Netzwerkes auf den Trainingsdaten quantifiziert. Beispiele für solche Loss-Funktionen sind z.B. in Kapitel 2.1.5 und 2.1.4 gegeben. Der Wert von E wird bestimmt, indem die Ausgabe des Netzwerkes auf Basis der Eingabe x berechnet und mit der Soll-Ausgabe y verglichen wird. Die durch das Netzwerk berechnete Transformation wird dabei durch die Gewichtungen aller Neuronen definiert. Durch einen Gradientenabstieg soll die Kombination der Gewichtungen gefunden werden, die E minimiert. Dabei wird der Wert der Loss-Funktion E durch das Netzwerk zurück propagiert, indem die partiellen Ableitungen für alle Gewichtungen der Neuronen gebildet werden. Darauf basierend wird dann die Änderung der Gewichtung ω_{ij} folgendermaßen berechnet:

$$\Delta\omega_{ij} = -\eta\frac{\partial E}{\partial\omega_{ij}} \tag{2.3}$$

Da der Gradient immer in die Richtung des größten Anstieges auf der Fehleroberfläche zeigt, wird also ein Schritt in die inverse Richtung des Gradienten gemacht. η wird als Lernrate bezeichnet und bestimmt die Länge dieses Schrittes, indem sie mit dem Gradienten multipliziert wird. Die Gewichtungen werden also abhängig von ihrem Einfluss auf das Ergebnis geändert. Der Backpropagations-Algorithmus findet dabei jedoch nur lokale Minima.

2.1.3 Caffe [JSD+14]

Das hier verwendete Frame-Work für die Konstruktion und Optimierung von Neuronalen Netzwerken ist Caffe [JSD+14]. Dabei wurde das Python-Interface benutzt. Caffe basiert auf C++ und bietet bereits eine Implementierung der wichtigsten Bausteine Neuronaler Netzwerk sowie der notwendigen Algorithmen zu deren Training. Dieses Framework bietet den Vorteil, dass bereits eine große Anzahl an Architekturen im Bereich der Convolutional Neural Networks zu Verfügung stehen. Für viele dieser Netzwerke stehen dabei auch bereits auf verschiedenen Datensätzen trainierte Modelle (die Gewichtungen) zur Verfügung, welche als Initialisierung für ein Finetuning der Gewichtungen in Bezug auf einen neuen Datensatz dienen können.

2.1.4 Klassifikation

Seien N eine Anzahl an Pixeln oder Objekten, die klassifiziert werden sollen, und K die Menge an Klassen. Die K Ausgabeneuronen produzieren für jede der K Klassen einen Score-Wert n_{nk} für die Klassifikationsaufgabe $n \in N$. Diese Score-Werte werden durch eine Soft-Max Funktion auf den Wertebereich $[0, 1]$ abgebildet, sodass die Summe über alle Klassen 1 ergibt. Die Soft-Max Funktion ist folgendermaßen definiert:

$$\hat{p}_{nk} = \frac{\exp(x_{nk})}{\sum_{k'}\exp(x_{nk'})} \tag{2.4}$$

Die sich so ergebenden Werte können als eine Art Wahrscheinlichkeit für die K Klassen interpretiert werden. Die Loss Funktion der Klassifikation, welche es zu minimieren gilt, ist dann auf dieser Basis der Kreuzentropie folgendermaßen definiert:

$$L = \frac{-1}{N} \sum_{n=1}^{N} log(\hat{p}_n, l_n) \tag{2.5}$$

$l_n \in [0, 1...K-1]$ bezeichnet dabei das ground-truth Label der Klassifikation n. $log(\hat{p}_n, l_n)$ ist nur für das Label l_n definiert und bezeichnet den Logarithmus zur Basis 10 der prädizierten Wahrscheinlichkeit $\hat{p}_n$ ($\hat{p}_{nk}$ wobei $k = l_n$) für das ground-truth Label bei der Klassifikation n. Dieser Wert ist also eine große negative Zahl, wenn die Wahrscheinlichkeit für l_n nahe 0 ist und eine kleine negative Zahl, wenn diese nahe 1 ist. $\frac{-1}{N}$ normiert diesen Loss über alle Klassifikationen und bildet ihn auf die positiven Zahlen ab, sodass die Minimierung der Loss-Funktion zu einer Maximierung der Wahrscheinlichkeit für das ground-truth Label führt. Die Informationen zu dieser Erklärung stammen aus [GBC16].

2.1.5 Regression

Durch Neuronale Netzwerke können auch Regressionsaufgaben gelöst werden. Diese zeichnen sich dadurch aus, dass anders als bei der Klassifikation nicht eine Zuordnung zu einer bestimmten Klasse bestimmt werden soll, sondern ein Wert in einem kontinuierlichen Wertebereich ermittelt wird. Der Wert des Ausgabeneurons soll dabei eine möglichst geringe Distanz zum ground-truth Wert haben, weshalb als Loss-Funktion eine Distanzfunktion dient, die den zu minimierenden Fehler der Regression bestimmt.

In dieser Arbeit wird die Anpassung einer Bounding-Box an ein Objekt als Regressionsaufgabe formuliert, in der die Höhe, die Breite und die Koordinaten des Bounding-Box-Zentrums ermittelt werden sollen. Insofern soll ein Vektor $t^u = (t^u_x, t^u_y, t^u_w, t^u_h)$ prädiziert werden, der diese Anpassungen repräsentiert. Der ground-truth Vektor wird dabei als $\upsilon = (\upsilon_x, \upsilon_y, \upsilon_w, \upsilon_h)$ bezeichnet und repräsentiert die nötigen Anpassungen der aktuellen Bounding-Boxen an die ground-truth Bounding-Box.

$L_{reg}(t, \upsilon)$ bezeichnet die Loss-Funktion der Regressionsaufgabe und ist definiert als die Distanzfunktion $smooth_{L_1}(t^u_i - \upsilon_i)$ summiert über alle Elemente des Anpassungsvektors.

$$L_{reg}(t, \upsilon) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t^u_i - \upsilon_i) \tag{2.6}$$

Die Distanzfunktion $smooth_{L_1}(x)$ ist folgendermaßen definiert.

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & \text{wenn } |x| < 1 \\ |x| - 0.5 & \text{sonst} \end{cases} \tag{2.7}$$

Diese Formulierung ist weniger anfällig für Ausreißer als die L_2 Distanzfunktion. Wenn die Regressionsziele nämlich unbegrenzt sind, kann das Trainieren mit dem

L_2 Loss nämlich eine sorgfältige Anpassung der Lernrate erfordern, um explodierende Gradienten zu verhindern. Der hier präsentierte Regressions-Loss stammt aus [Gir15].

2.2 Pixelweise Semantische Segmentierung

Abbildung 2.5: Die hier gezeigten Daten stammen aus [COR+16]. Dabei ist auf der linken Seite das Bild zu sehen, auf dessen Basis die Semantische Segmentierung auf der rechten Seite erstellt wurde. Dabei wurde jedem Pixel jeweils die Klasse der Struktur zugeordnet, von der dieses Pixel einen Teil bildet.

Im Bereich des Autonomen Fahrens bildet das Verständnis der gegebenen Szene die Grundlage, auf deren Basis das Auto seine Aktionen planen kann. Eine wichtige Informationsquelle hierfür bildet die Wahrnehmung der Umgebung durch Kameras, deren Bilder dahingehend analysiert werden, welche Objekte sich wo im Bild befinden. Eine solche Analyse wird auch Semantische Segmentierung genannt. Genauer versteht man darunter die pixelweise Klassifikation des Bildes zu einer definierten Menge an Klassen. Ein Beispiel hierfür kann man in Abbildung 2.5 sehen.

Um das Problem der Semantischen Segmentierung anzugehen, gibt es eine Vielzahl an Möglichkeiten. In letzter Zeit lieferten Ansätze, welche auf Neuronalen Netzen basieren, jedoch die besten Ergebnisse. Im Bereich der Neuronalen Netze gibt es zwei hauptsächliche Ansätze, die sich in ihrer Funktionalität unterscheiden. Der erste ist der sogenannte Bounding-Box-Ansatz, welcher als erstes von [Gir15] beschrieben wurde, und darauf basiert, zuerst das zu segmentierende Objekt zu detektieren und anschließend innerhalb der gefundenen Box zu segmentieren. Der zweite Ansatz, welcher sich in mehreren Challenges, z.B. Coco/Cityscapes, als überlegen herausgestellt hat, basiert auf einer pixelweisen Architektur, welche in [SLD16] als erstes vorgestellt wurde.

2.2.1 Generelle Architektur der pixelweisen Semantischen Segmentierung

Die generelle Architektur, wie sie in [SLD16] präsentiert wurde, kann man in der Abbildung 2.6 sehen. Die in der Abbildung gezeigte Architektur ist in 4 generelle

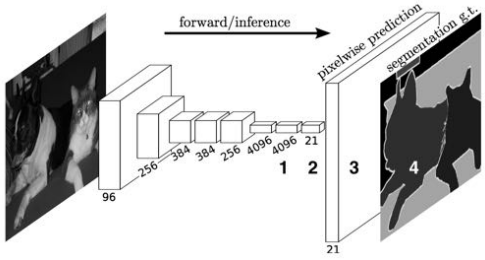

Abbildung 2.6: Hier ist die generelle Netzwerk-Architektur der pixelweisen Segmentierung präsentiert. Die Bildinformation wird zu einer herunterskalierten Feature-Map verarbeitet (1). Aus dieser Feature-Map wird eine Score-Map für jede der in diesem Fall 21 Klassen bestimmt (2). Diese Score-Maps werden dann wiederum auf die originale Bildgröße hochskaliert (3) Für jedes Pixel wird dann die Klasse mit dem größten Score bestimmt (4). Das wurde aus [SLD16] entnommen und bearbeitet.

Schritte aufgeteilt. Im ersten Schritt werden die Bilddaten so verarbeitet, dass ihre für die Segmentierung wichtigsten Features extrahiert werden. Die Ausgabe dieses Schrittes ist eine im Verhältnis zum Originalbild herunterskalierte Feature-Map, in welcher jedes Pixel ein bestimmtes Rezeptives Feld im Originalbild repräsentiert und mithin die Information dieser Region beinhaltet. Basierend auf der Feature-Map wird im nächsten Schritt dann eine sogenannte Score-Map Sc für jede Klasse berechnet, sodass die Ausgabe dieses Schrittes die folgende Form hat: $Sc := LaengeFeatureMap \times BreiteFeatureMap \times AnzahlKlassen$. Die Elemente dieser Score-Map, also die Scores einer Klasse in Bezug auf einen Pixel, können als eine Art Pseudo-Wahrscheinlichkeit interpretiert werden, dass das Pixel zur korrespondierenden Klasse gehört. Diese Score-Map, welche die gleiche Auflösung der Feature Map hat und mithin geringer aufgelöst als das Eingangsbild ist, wird nun im nächsten Schritt auf die Auflösung des Originalbildes skaliert. Das Skalieren wird in der Regel über eine Bilineare Interpolation ausgeführt. Dabei wird ein hochskalierter Score Wert durch die inverse Distanz zu seinen vier nächsten Nachbarn in der geringer aufgelösten Score-Map gewichtet. Im Anschluss existiert also für jedes Pixel im Eingangsbild ein Score für jede Klasse. Um das Label eines Pixels zu bestimmen, wird dann im vierten Schritt für jedes Pixel die Klasse mit dem maximalen Score ermittelt.

2.2.2 Spezielle Architektur des FCN-8

Abbildung 2.7 zeigt die spezielle Architektur des FCN-8, welche wir im Kontext der semantischen Instanzsegmentierung verwenden. Es handelt sich hierbei um ein Fully Convolutional Neural Network, was bedeutet, dass es nur Faltungsschichten beinhaltet und keine Fully Connected Schicht. Dies hat den Hintergrund, dass durch das Anwenden einer Fully Connected Schicht die räumliche Information beschädigt werden würde, welche also beim Fully Convolutional Network erhalten bleibt [SLD16].

Die Erstellung der Feature Map erfolgt von Schicht Conv1 bis Conv6-7 (siehe Abbildung 2.7). Dieser Teil des FCN-8 ist ein Netzwerk, das für die Klassifikation von ganzen Bildern konzipiert wurde. Beim Training kann das Modell dann mit den Gewichtungen,

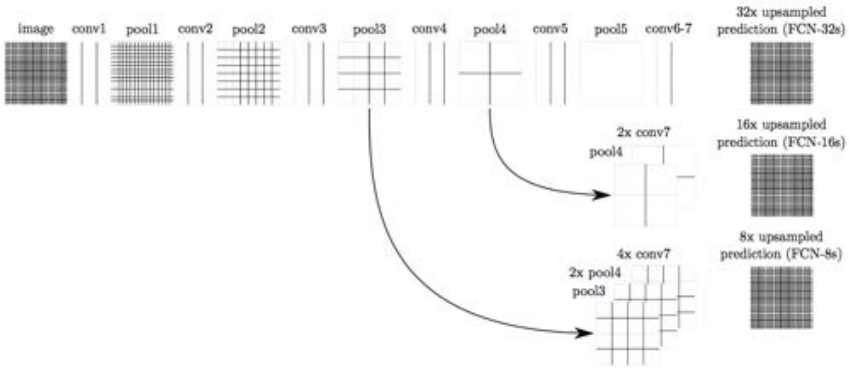

Abbildung 2.7: Die Abbildung, welche aus [SLD16] entnommen ist, zeigt die Netzwerkarchitektur des FCN-8. (Conv = convolutional layer/pool=pooling layer)

die beim Training der Klassifikation auf einem verwandten Datensatz bestimmt wurden, vorinitialisiert werden. Dabei werden jedoch die in diesen Klassifikationsnetzwerken üblicherweise vorhandenen Fully Connected Schichten in Faltungsschichten umgewandelt. In diesem Fall handelt es sich um das sogenannte VGG16, welches in [SZ14] beschrieben wurde.

Das FCN-8 kombiniert die Ausgabe von Score-Maps, welche auf Feature-Maps mit hohen und geringen Auflösungen bestimmt wurden, wie es in Abbildung 2.7 nachvollzogen werden kann. Dies geschieht, indem nach pool3, pool4 und conv6-7 die Scoremaps bestimmt werden, welche dann anschließend durch Skalierung und Zuschneiden auf die gleiche Größe gebracht werden. Hiernach werden diese summiert. Durch die Kombination von gering aufgelösten und hoch aufgelösten Score-Maps trifft das Netzwerk lokale Entscheidungen in Bezug auf globale Strukturen.

Die Interpolation ist als bilineare Interpolation initialisiert und wird im Zuge des Trainings zusammen mit dem Rest des Netzwerkes trainiert.

2.3 Detektion von Instanzen

Die Detektion von Instanzen im Bereich des Selbstfahrenden Autos bietet genauso wie die Semantische Segmentierung weitere semantische Informationen über die das Auto umgebende Szene. Dabei wird bei der Objektdetektion, anders als bei der Semantischen Segmentierung, zwischen den Objektinstanzen einer Klasse unterschieden. Dies geschieht dadurch, dass wie z.B. in Abbildung 2.8 zu sehen, eine Bounding-Box um das Objekt herum bestimmt wird, welche einer Semantischen Klasse zugeordnet wird. Bei der Bestimmung der Bounding-Box handelt es sich dabei um eine Regression, bei der die vier Koordinaten des Rechtecks innerhalb des Bildes bestimmt werden, die die maximalen und minimalen X und Y Koordinaten des Objektes repräsentieren. Die Bounding-Box umfasst also das Objekt so eng wie möglich. Bei der Zuordnung des von der Bounding-Box umfassten Objektes zu einer definierten Mengen an Klassen handelt es sich dann um eine Klassifikation.

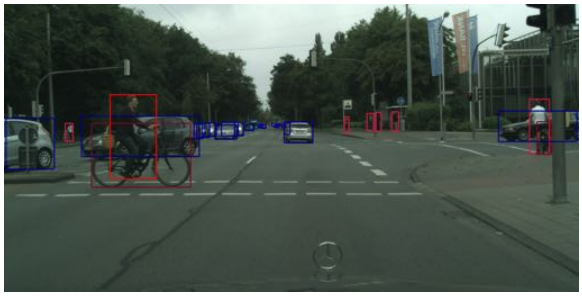

Abbildung 2.8: Die Daten für diese Abbildung stammen aus [COR+16]. Für jede Instanz einer Klasse im Bild existiert eine sogenannte Bounding-Box, die durch die maximalen und minimalen Koordinaten der Instanz in den beiden räumlichen Dimensionen definiert ist. Zudem ist jeder Bounding-Box die Klasse zugeordnet, zu der die umfasste Instanz gehört. Die Klassen sind in diesem Fall durch die Farben der Boxen kodiert.

Ähnlich wie bei der Semantischen Segmentierung basieren die Ansätze, die die besten Ergebnisse für die Objektdetektion liefern, auf Neuronalen Netzwerken. Dabei existieren zwei hauptsächliche Architekturen. Zum einen existiert der Ansatz, dass zunächst über ein sogenanntes Region Proposal-Network (RPN) eine Menge an Bounding-Box Vorschlägen produziert werden, die dann im Nachhinein durch ein weiteres Netzwerk klassifiziert und verbessert werden. Zum anderen existieren einstufige Ansätze, die in einem Schritt eine Reihe von Bounding-Box Vorschlägen generieren und klassifizieren. Dabei ist der erste Ansatz meist in Bezug auf die Detektionsqualität überlegen und der letztere in Bezug auf die Geschwindigkeit, was beides in Hinsicht auf das Selbstfahrende Auto wichtige Eigenschaften sind, zwischen denen abzuwägen ist. Im Folgenden wird nun näher auf Architekturen aus beiden Ansätzen eingegangen. Ein genauerer Blick auf deren Leistungen bezüglich Geschwindigkeit, Detektionsqualität und Ressourcenverbrauch wird dann in Kapitel 3.3 im Zusammenhang mit der Auswahl des Detektors gegeben.

2.3.1 Faster-RCNN [RHGS15]

Eine der Architekturen, die zur Zeit mit die besten Ergebnisse liefern, ist das Faster-RCNN aus [RHGS15]. In Abbildung 2.9 ist dabei auf der linken Seite die generelle Architektur dieses Ansatzes zu sehen. Wie zu erkennen, wird zunächst durch ein Convolutional Neural Network eine Feature Repräsentation des Bildes erstellt, auf deren Basis die Aufgabe der Detektion gelöst werden kann.

Region-Proposal Network (RPN)

Zunächst wird durch das Region Proposal Network eine Reihe an möglichen Objekt-Detektionen generiert, welche keiner spezifischen Klasse zugehörig sind. In Abbildung 2.9 ist auf der rechten Seite die Funktionsweise dieses Netzwerks demonstriert. Dieses funktioniert dabei so, dass über jedes räumliche Element der Feature-Map k verschiedene Bounding-Box-Vorschläge, sogenannte Anchors, geprüft werden. Jedem dieser k Anchor

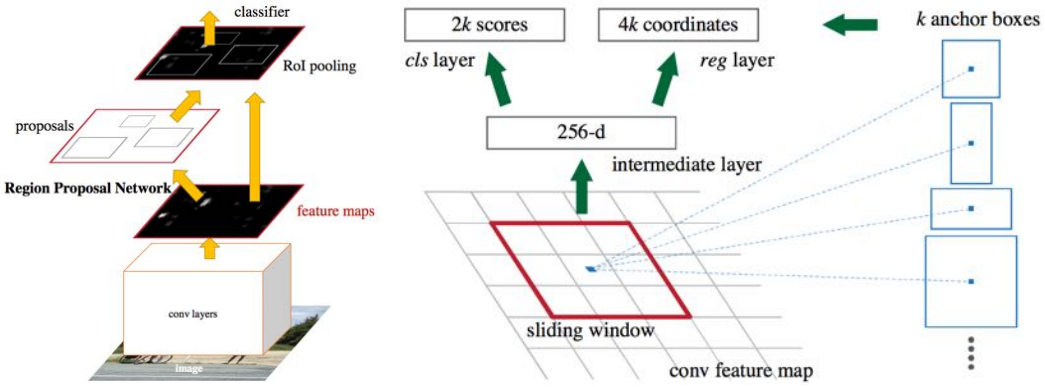

Abbildung 2.9: Auf der linken Seite der Abbildung ist die generelle Architektur des Faster-RCNN zu sehen. Dieser besteht aus einem Fully Convolutional Network ("conv layers") zur Erstellung einer Feature-Map ("feature maps "), einem "Region Proposal Network "zur Erstellung möglicher Objektdetektionen (proposals) und eines nachfolgenden Klassifikators dieser Objektdetektionen. Auf der rechten Seite ist die Architektur des RPN zu sehen. In dieser werden für k sogenannte "Anchor "Boxen an jeder räumlichen Position in der Feature-Map bestimmt, ob innerhalb ihrer Grenzen ein Objekt existiert ($2k$ Scores := Wahrscheinlichkeit für Hintergrund oder Vordergrund) und wie die Bounding-Box besser an das Objekt angepasst werden kann ($4k$:= neue Koordinaten).

wird dabei durch eine Reihe von Convolutions ("intermediate layer") zwei Konfidenz-Werte zugeordnet, dass sich innerhalb der Bounding-Box ein Objekt oder Hintergrund befindet. Deshalb sind in der Abbildung auch "$2k$ scores" als Ausgabe angegeben. Des Weiteren wird durch diese "intermediate layer" jeweils noch eine Bounding-Box-Regression bestimmt, die für alle k Anchors eine Anpassung der 4 Bounding-Box-Koordinaten auf das Objekt bestimmt, das diese umfasst. Insofern ergeben sich, wie in der Abbildung zu sehen, weitere $4k$ Ausgaben.

Da dieser Prozess für alle räumlichen Elemente einer Feature-Map erfolgt, wird zunächst eine Non-Local-Maximum Supression angewendet, welche verhindern soll, dass für ein Objekt mehrere Detektionen generiert werden. Für die weitere Bestimmung der klassenspezifischen Detektion werden dann die 300 Bounding-Boxen mit der höchsten Objekt-Konfidenz betrachtet.

ROI-Pooling

Diese 300 Bounding-Boxen dienen dann zunächst dazu, die Feature-Map zu quantisieren. Dabei wird der Teil der Feature-Map, der innerhalb des Bounding-Box Vorschlages liegt, auf eine immer gleich große Feature-Map abgebildet. Dieser Prozess wird als ROI-Pooling bezeichnet und besteht darin, dass der in der Feature-Map $h \times w$ Pixel große Bounding-Box Vorschlag in ein $H \times W$ großes Gitter aufgeteilt wird. Dabei werden die Feature-Map Pixel durch ein Max-Pooling ihrem korrespondierenden, $H/h \times W/w$ großen Gitterelement zugeordnet. In Abbildung 2.10 ist der Fall für $h = 5/w = 7$ und $H = 2/W = 2$ zu sehen. Wenn H/h oder W/w wie in der Abbildung keine ganzen Zahlen ergeben, wird jeweils zum nächsten Pixel im Gitterelement gerundet. Dieses

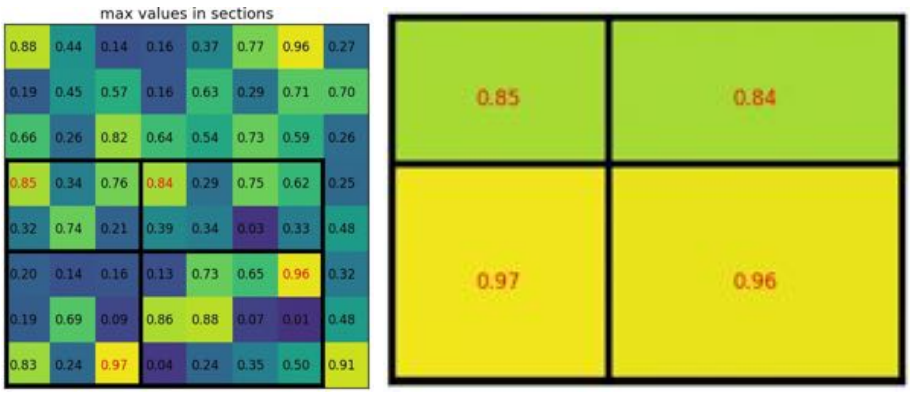

Abbildung 2.10: Diese Abbildung wurde entnommen und verändert aus https://github.com/deepsense-ai/roi-pooling.git (abgerufen am 3.10.2017). Links ist die Feature-Map zu sehen, über die ein 2×2 Gitter gelegt wird, sodass alle räumlichen Elemente einem Gitterelement zugeordnet sind. Rechts ist das Ergebnis Max-Poolings innerhalb der Gitterelemente zu sehen.

Pooling wird für alle Kanäle der Feature-Map einzeln erstellt, sodass die Ausgabe eine räumliche Dimension von $K \times W$ hat, und die Anzahl der Kanäle der Feature-Map.

RCNN

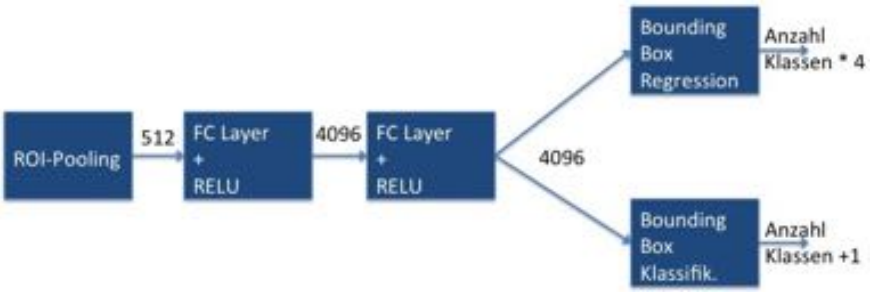

Abbildung 2.11: Die Abbildung zeigt den Aufbau des RCNN. Bei der Bounding-Box-Regression und Klassifikation handelt es sich dabei auch um Fully-Connected Layer.

Diese neue Repräsentation der Feature-Map innerhalb der Bounding-Box dient dann als Eingabe für das RCNN, wie es auch in Abbildung 2.11 zu sehen ist. Auf Basis der Ausgabe der beiden Fully-Connected Layer wird dann für jede der 300 Bounding-Box-Vorschläge des RPN eine Klassifikation bestimmt, die die Bounding-Box entweder zu einer der semantischen Klassen oder zum Hintergrund zuordnet. Insofern werden für jede Bounding-Box "Anzahl Klassen+1" Konfidenzen bestimmt. Zudem wird für jede der Semantischen Klassen eine spezifische Regression bestimmt, sodass die Ausgabe dieser Fully Connected Layer die Form "Anzahl Klassen * 4" hat. Die Detektion ist dann also über die Klassifikation der Bounding-Box und ihre speziell für ein Objekt dieser Klasse bestimmte Regression definiert.

Loss Funktionen des Faster RCNN

Zur Berechnung der Loss-Funktion für die Detektion des Faster-RCNN muss zunächst einmal eine Zuordnung der prädizierten Bounding-Boxen zu den ground-truth Bounding-Boxen geschehen. Dabei wird den Anchors (siehe Abschnitt über das RPN 2.3.1) ein positives oder negatives Label zugeordnet, welches dafür steht, dass diese zu einem Objekt korrespondieren oder nicht. Die Anchors werden im RPN und RCNN klassifiziert und durch die Regression angepasst, sodass aus einer Zuordnung dieser zu einer ground-truth Bounding-Box auch eine Zuordnung der im RPN und RCNN erstellten Bounding-Boxen folgt. Wenn die IoU (wird in Kapitel 2.5.3 eingeführt und ist ein Maß für die Überlappung der Bounding-Boxen) mit einer ground-truth Bounding-Box > 0.7, so wird dem Anchor ein positives Label zugeordnet. Zudem wird jedem Anchor ein positives Label zugeordnet, wenn er die maximale IoU unter allen Anchorn mit einer ground-truth Bounding-Box aufweist. Das negative Label wird dann einem Anchor zugewiesen, wenn die IoU des Anchors mit allen ground-truth Bounding-Boxen < 0.3 ist. Alle Anchors, auf die keines der Kriterien zutrifft, werden nicht mit in die Berechnung der Bounding-Box einbezogen.

Die Loss-Funktion für des Region Proposal Networks ist dabei folgendermaßen definiert:

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{2.8}$$

Dabei bezeichnet i den Index eines Bounding-Box-Vorschlags und p_i die vorhergesagte Wahrscheinlichkeit, dass i ein Objekt ist. p_i^* ist das ground-truth Label, welches 1 ist, wenn der Anchor positiv ist und 0, wenn der Anchor negativ ist. t_i bezeichnet den vierdimensionalen Vektor, welcher die Bounding-Box-Koordinaten umfasst und t_i^* bezeichnet die ground-truth-Koordinaten, welche zu der Bounding-Box des Objektes gehören. L_{cls} ist dabei der Loss aus 2.1.4 über zwei Klassen (Objekt vs. kein Objekt). L_{reg} ist definiert als $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$, wobei R die robuste Loss Funktion (smooth L_1) aus Kapitel 2.1.5 ist. Durch $p_i^* \cdot L_{reg}$ ist der Loss nur für tatsächliche Detektionen (positive Anchors) definiert und durch λ gegen L_{cls} gewichtet. N_{cls} ist dabei die Größe der Minibatch und N_{reg} die Anzahl der Bounding-Box-Vorschläge. $1/N_{cls}$ und $1/N_{reg}$ normalisieren dabei ihren jeweiligen Loss.

Der Gesamt-Loss des RCNN ist folgendermaßen definiert:

$$L(p, u, t^u, \upsilon) = L_{cls}(p, u) + \lambda[u \geq 1] L_{loc}(t^u, \upsilon) \tag{2.9}$$

Dabei ist $L_{cls} = -log(p_u)$ der Loss aus 2.1.4, welcher für die korrekte Klasse u definiert ist (p bezeichnet die prädizierte Klassenwahrscheinlichkeit). L_{loc} ist über ein Tupel von ground-truth Bounding-Boxen für die Klasse u, $\upsilon = (\upsilon_x, \upsilon_y, \upsilon_w, \upsilon_h)$ und dem für die Klasse u prädizierten Tupel $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$ definiert. $[u \geq 0]$ bedeutet dabei, dass die Hintergrundklasse bei der Berechnung des Loss ausgeschlossen wird. L_{loc} ist genauso wie L_{reg} als glatter L_1 Loss aus Kapitel 2.1.5 definiert.

2.3.2 R-FCN [DLHS16]

Diese Art einer zweistufigen Architektur, die auf der Generierung von Bounding-Box-Vorschlägen und der nachfolgenden Klassifikation basiert, findet auch im Ansatz aus [DLHS16] Anwendung. Durch eine Backbone-Architektur, bestehend aus Convolutions und Nichtlinearitäten, wird auch hier, wie in Abbildung 2.12 zu sehen, eine Feature-Map auf Basis des Eingabebildes bestimmt. Die Feature-Map dient dann dazu, wie in [RHGS15], zunächst einmal Instanzvorschläge in Form von Bounding-Boxen zu generieren.

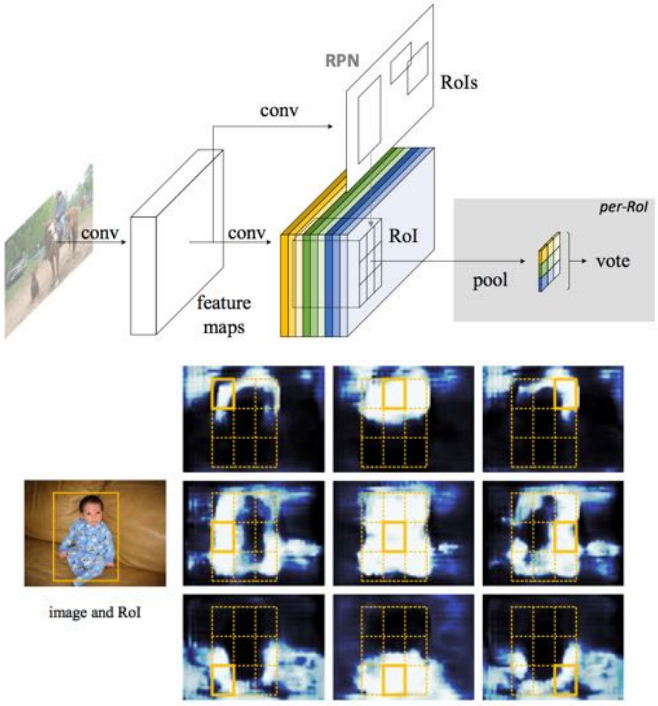

Abbildung 2.12: In dieser Abbildung ist oben die Architektur des R-FCN aus [DLHS16] zu sehen. Diese besteht aus einem Fully-Convolutional Netzwerk, das das Eingabebild zu einer Feature-Map verarbeitet. Diese Feature-Map dient als Basis für eine Region Proposal Network. Darüber hinaus wird pixelweise die Wahrscheinlichkeit bestimmt, dass das aktuelle Pixel eine bestimmte relative Position zu einer Instanz einer bestimmten Klasse hat (Kanäle mit verschiedenen Farben). Ein Beispiel für solch eine Score-Map mit 3×3 relativen Positionen ist im unteren Teil der Abbildung zu sehen. Auf Basis der Ausgabe des RPN wird dann pro Bounding-Box eine Score-Map bestimmt, indem die Wahrscheinlichkeiten für die relativen Positionen an den entsprechenden relativen Positionen der Bounding-Box übernommen werden. Durch eine Auswahlprozedur wird dann die Klasse bestimmt.

Gleichzeitig unterscheidet sich dieser Ansatz dadurch, dass, wie in Abbildung 2.12 zu sehen, eine positionssensitive Score-Map erstellt wird. Abhängig von der Position insofern, als dass für jedes Pixel die Wahrscheinlichkeit bestimmt wird, dass dieses sich in einer bestimmten relativen räumlichen Position zu einer Instanz der C Kategorien befindet. Dabei wird jeweils zwischen $k \times k$ relativen Positionen unterschieden, die als $k \times k$ Elemente in einem räumlichen Gitter interpretiert werden können. Ein Beispiel für den Fall eines 3×3 großen Gitters ist z.B. in Abbildung 2.12 auf der rechten

Seite zu sehen. Dabei repräsentieren die Einträge im unteren rechten Element z.B. die Wahrscheinlichkeiten dafür, dass das Pixel ein Element mit dieser relativen Position zu einer Instanz einer Klasse $\in C$ repräsentiert. Insgesamt wird auf Basis der Feature-Map also eine Score-Map für jede der $C + 1$ (+1 wegen des Hintergrundes) Kategorien und der k^2 relativen Positionen in Bezug auf die Instanzen dieser Kategorien berechnet, was zu einer Ausgabe mit $k^2(C + 1)$ Kanälen führt.

Die Bounding-Box Vorschläge des Region Proposal Networks werden dann von einem sogenannten ROI-Pooling Layer (eine andere als beim Faster-RCNN) verwendet, um in dem prädizierten Bounding-Box Vorschlag eine positionsabhängige Score-Map zu erstellen. Dabei wird der Bounding-Box Vorschlag durch ein räumliches $k \times k$ Gitter aufgeteilt, sodass für jedes Gitterelement die Scores gepoolt werden, die der relativen Position des Gitterelementes und der Kategorie entsprechen. Dieser Prozess kann wiederum in Abbildung 2.12 betrachtet werden, wo nach dem ROI-Pooling jedes Gitterelement die Scores aus dem jeweiligen Kanal entnommen hat.

Die so erstellte Score-Map für jede Bounding-Box dient dann wiederum als Grundlage für die Bestimmung der Klasse durch ein Auswahlverfahren. In diesem wird die Score-Map für jede Kategorie gemittelt und so für jede der $C + 1$ Kategorien ein Gesamt-Score bestimmt. Die Kategorie, die die maximale Score aufweist, wird dann für die Bounding-Box klassifiziert. Ähnlich wie in [Gir15] wird auch hier eine Bounding-Box Regression angewendet, um die Bounding-Boxen besser an die Objekte anzupassen.

Der Gedanke, der hinter der in diesem Ansatz eingeführten positionsabhängigen Score-Map steht, ist, dass die meisten Netzwerke, die als Basis zur Erstellung der Feature-Map genutzt werden, eigentlich für die Klassifikation erstellt wurden. Aus diesem Grund besitzen diese die Eigenschaft der Translationsinvarianz, welche für die Klassifikation wichtig ist. Bei der Detektion jedoch ist es von Bedeutung, mehrere Objekte voneinander zu unterscheiden und deren Positionen zu erkennen, weshalb die Features, auf deren Basis diese Aufgabe gelöst wird, translationsvariant sein müssen (die Verschiebung eines Objektes führt zu einem anderen Ergebnis). Die beschriebene Score-Map verschafft insofern Abhilfe, als dass jedes Pixel in der Score-Map seine relative Position zu einer Instanz beinhaltet und mithin abhängig von seiner räumlichen Position, also translationsvariant ist.

2.3.3 Einstufige Ansätze zur Detektion

Neben den bisher beschriebenen Ansätzen, welche auf Basis eines Region Proposal-Networks und einer nachfolgenden Klassifikation basieren, existieren auch Ansätze, die eine schnellere Detektion zu erreichen versuchen, indem sie die Bounding-Box und die zugehörigen Klassen simultan bestimmen. Ein solcher Ansatz, mit dem Namen YOLO - You Only Look Once - wurde z.B. in [RDGF16] vorgestellt. Dieser funktioniert dabei so, dass zunächst, wie in Abbildung 2.13 zu sehen, das Bild in ein $S \times S$ großes Gitter unterteilt wird. Wenn das Zentrum eines Objektes in einer der Gitterzellen liegt, ist diese für die Detektion dieser Instanz verantwortlich.

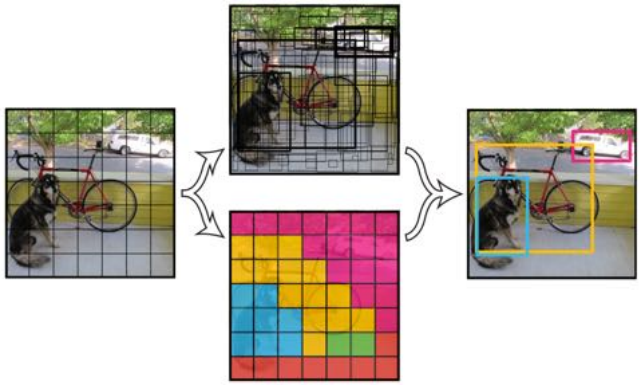

Abbildung 2.13: Die Abbildung (aus [RDGF16]) illustriert die Vorgehensweise des YOLO Detektors. Das Eingabebild wird in ein $S \times S$ großes Gitter unterteilt. Für jedes Gitterelement werden dann B Bounding-Boxen geprüft, für die jeweils ein Konfidenzwert für jede Klasse prädiziert wird. Die Bounding-Box mit dem besten Konfidenzwert wird dann dieser Klasse zugewiesen.

Jede Zelle prädiziert dabei B Bouding-Boxen, für die durch ein Neuronales Netzwerk ein Konfidenzwert prädiziert wird. Dieser beschreibt, wie gut die Box das Objekt umfasst und wie wahrscheinlich es ist, dass sich das Objekt in dieser Bounding-Box befindet. Die Konfidenz soll also eine Abschätzung für $P(Object) * IOU_{pred}^{truth}$ geben. Wenn also kein Objekt in der Bounding-box existiert, sollte die Konfidenz Null sein und wenn ein Objekt existiert, sollte die Konfidenz der IOU mit der ground-truth Bounding-Box entsprechen. Für jede Bounding-Box werden also fünf Parameter inferiert: (x, y) Koordinaten des Objektzentrums w, h Breite und Höhe der Box und die Konfidenz. Jede Bounding-Box prädiziert zudem noch die bedingte Wahrscheinlichkeit $P(Class_i \in C|Object)$ pro Gitterzelle. Die Gesamt-Konfidenz hat dann die Form:

$$P(Class_i|Object) * P(Object) * IOU_{pred}^{truth} = P(Class_i) * IOU_{pred}^{truth} \tag{2.10}$$

Die Ausgabe für das gesamte Bild hat dann also die Form $(S \times S * (B * 5 + C))$. Dieser Ansatz erreicht dabei mit der beschriebenen Architektur eine sehr schnelle Detektion, die jedoch weniger genau ist als die der zweistufigen Ansätze mit RPN.

Ein weiterer Ansatz, der sich die einstufige Architektur aus [RDGF16] zum Vorbild nimmt, ist der Ansatz aus [WIJK16]. Dabei ist zum einen der Gedanke hinter diesem Ansatz, durch die Verwendung des SqueezeNet Modells [IMA+16], einem ¡ 0.5 MB großen Modell mit ähnlicher Leistung wie wesentlich größere Netzwerke, eine ressourcenschonende Detektion zu ermöglichen, die zum anderen wie das YOLO Netzwerk sehr schnell ist. Die Detektionsarchitektur unterscheidet sich dabei von der des YOLO Netzwerkes vornehmlich dadurch, dass zusätzlich zu den Bounding-Boxen und der Confidenz noch eine Bounding-Box Regression ähnlich der in [Gir15] prädiziert wird. Als Klassifikator für die einzelnen Bounding-Boxen dient dabei ein Fully-Covolutional Network.

In [RCL+17] wird ebenfalls eine einstufige Architektur für die Detektion vorgeschlagen, welche in diesem Fall auf einem rekurrenten Neuronalen Netzwerk zur Generierung der

Feature-Map basiert. Bei diesem wird die Bildinformation nicht nur von Schicht zu Schicht vorwärts durch das Netzwerk propagiert, sondern auch in die entgegengesetzte Richtung. Dabei werden die räumlich detailreichen aber wenig semantisch abstrakten Informationen der hochaufgelösten Feature-Map mit den wenig räumlich detailreichen aber semantisch abstrakten Informationen kombiniert, sodass lokale Entscheidungen in Bezug auf den globalen Kontext getroffen werden können. Der hier präsentierte Ansatz liefert zwar gute Ergebnisse, ist jedoch auch sehr langsam.

2.4 Semantische Instanzsegmentierung

Abbildung 2.14: Die hier gezeigten Bilder basieren auf dem Datensatz aus [COR+16]. Auf der linken Seite ist das Eingabebild zu sehen, auf dessen Basis die Instanzsegmentierung auf der rechten Seite erstellt wurde. Dabei wird hier jedem Pixel neben der Klasse, zu der die Struktur gehört, deren Teil das Pixel ist, auch die Instanz innerhalb der Klasse zugewiesen. Um die Unterscheidung zwischen den Instanzen deutlich zu machen, wurde hier jeder Instanz eine unterschiedliche Farbe zugewiesen.

Ein Beispiel für die Semantische Instanzsegmentierung kann man z.B. in Abbildung 2.14 sehen. Dabei wird nicht nur die semantische Klasse jedes Pixel prädiziert, es wird, wie man sehen kann, auch zwischen den Instanzen einer Klasse unterschieden. Dies ist für die Erstellung eines Szenenmodells für das Selbstfahrende Auto wichtig, da sich unterschiedliche Instanzen unabhängig voneinander verhalten. Durch die Semantische Instanzsegmentierung kann dann pixelweise genau zwischen den Instanzen unterschieden werden und das Auto kann seine Reaktionen auf die einzelnen Objekte planen.

Die Methoden, die im Moment die besten Ergebnisse für die Semantische Instanzsegmentierung liefern, basieren auf Convolutional Neural Networks. Dabei kann grundsätzlich zwischen zwei hauptsächlichen Ansätzen unterschieden werden. Der erste (Ansatz 1) benutzt dabei eine überkomplette Menge an Objektinstanzen, die entweder verworfen oder als Instanzen einer bestimmten semantischen Klasse klassifiziert und verbessert werden. Die Leistung dieser Methoden ist abhängig von der Qualität der Instanzvorschläge, da sie keine fehlenden Vorschläge nach der Vorschlagsphase ersetzen können. Generell sind diese Ansätze eher langsam, da jeder Instanzvorschlag einzeln klassifiziert werden muss. Diese Eigenschaften begrenzen die Leistung dieser Methoden. Der zweite (Ansatz 2) kommt ohne Instanzvorschläge aus. Die Segmentierung und die semantische

Klasse von Objektinstanzen werden hier zusammen inferiert. Die pixelweise semantische Segmentierung ist sehr erfolgreich. Es ist also möglicherweise sinnvoll, das Problem der Instanzsegmentierung in eine Pixellabeling-Aufgabe umzuwandeln.

Allgemein besteht die größte Schwierigkeit beim Entwurf einer Architektur für die Semantische Instanzsegmentierung darin, dass man anders als bei der reinen Semantischen Segmentierung keine klar definierte Anzahl an Klassen hat, zu denen man jedes Pixel klassifizieren kann. Die Aufgabe der Semantischen Instanzsegmentierung hat vielmehr genau wie die Objektdetektion zusätzlich zu der Klassifikation Eigenschaften einer Regressionsaufgabe, bei der jedes Pixel zu einer unbekannten durch die Regression zu bestimmenden Menge an Instanzen und einer bekannten Menge an semantischen Klassen zugeordnet wird.

Im Folgenden wird näher auf die zwei hauptsächlichen Ansätze eingegangen mit dieser Dualität umzugehen. Dabei werden verschiedene konkrete Architekturen aus beiden Ansätzen näher betrachtet. In Bezug auf die Anwendbarkeit der Architekturen im Bereich des Selbstfahrenden Autos bietet Kapitel 3.1 dann einen weiteren Überblick.

2.4.1 Pixelweise Semantische Instanzsegmentierung

Eine pixelweise Formulierung der Instanzsegmentierung bietet die Herausforderung, dass um eine Instanz zu inferieren, der Klassifikator in jedem Pixel eine Regression und eine Klassifikation ausführen muss. Dabei müssen die Pixel zudem Informationen über die Zuordnung der anderen Pixel besitzen, da um eine Instanz zu segmentieren alle Pixel des Objektes das gleiche Klassen und Instanzlabel vorhersagen müssen und sich gleichzeitig von allen anderen vorhergesagten Instanzen unterscheiden müssen. Zudem besteht bei der Formulierung der Loss-Funktion das Problem, dass nicht genau definiert ist, welche prädizierte Instanz zu welcher GT-Instanz korrespondiert. Im folgenden Abschnitt wird nun ein Überblick über verschiedene Ansätze der pixelweisen Instanzsegmentierung gegeben, die diese Probleme angehen.

Unter diesen Ansätzen der pixelweisen Semantischen Instanzsegmentierung liefert die in [AT17] vorgestellte Methode im Moment die besten Ergebnisse. Dieser Ansatz basiert auf dem Gedanken, dass die Semantische Insanzsegmentierung als eine komplexere Art der Semantischen Segmentierung gesehen werden kann. Wie man aus Abbildung 2.15 erkennen kann, basiert dieser Ansatz auf zwei Sub-Netzwerken. Dabei wird im ersten eine Semantische Segmentierung mit einer an das in 2.2 vorgestellte FCN-8 angelehnte Netzwerkarchitektur inferiert. Als Klassifikator dieses Netzwerkes dient dabei ein Conditional Random Field (CRF). Dieses CRF verwendet die dicht verbundenen paarweisen Potentiale und formuliert diese als rekurrentes Neuronales Netzwerk. Zudem werden die Ergebnisse einer Instanz-Detektion, also die Bounding-Boxen in Form von Potentialen mit in das CRF einbezogen. Diese Detektionspotentiale verbessern die Semantische Segmentierung dadurch, dass sie Konsistenz zwischen der Semantischen Segmentierung und der Detektion erzwingen und die Detektionsscores rekalibrieren. Die Ausgabe dieses Netzwerkes sind dann die über eine Softmaxfunktion bestimmten

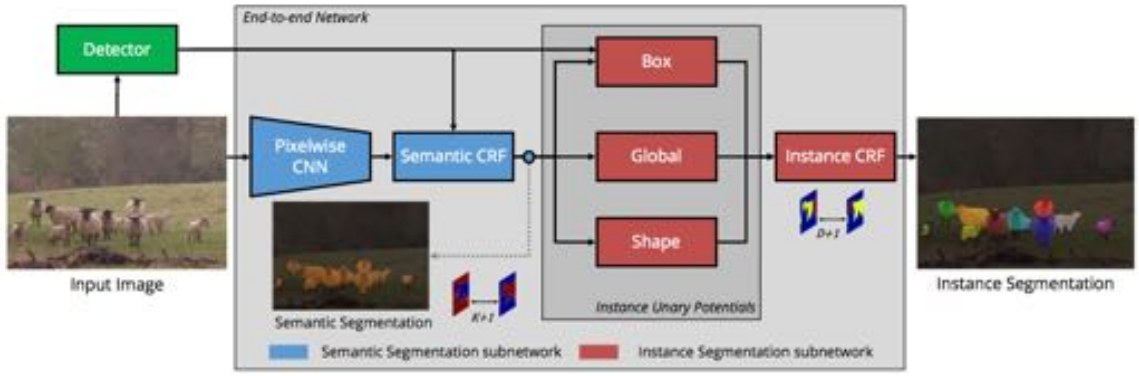

Abbildung 2.15: Die in der Abbildung (entnommen aus [AT17]) zu sehende Architektur zur pixelweisen Instanzsegmentierung basiert zunächst auf einem Subnetzwerk, das eine Semantische Segmentierung und eine Detektion inferiert. Dabei wird unter anderem auch eine Abschätzung der Anzahl der Instanzen im Bild bestimmt, nämlich die D Detektionen. Auf Basis dieser Informationen wird im zweiten Subnetzwerk ein Conditional Random Field konstruiert, das versucht, jedes Pixel einer der D Instanzen zuzuordnen.

Wahrscheinlichkeitskarten $Q_i(l) \in Q$, die für jeden Pixel i die Wahrscheinlichkeit beinhalten, dass dieses zur Klasse $l \in L$ gehört.

Das zweite Subnetzwerk erhält nun wiederum die Semantische Segmentierung Q und und die Detektionsergebnisse als Eingabe. Für jedes Bild existieren dabei D Detektionen der Form (l_i, s_i, B_i), wobei $l_i \in L$ das Klassenlabel bezeichnet, s_i den Konfidenzwert der Detektion und B_i die Menge an Pixeln in der Bounding-Box. Das Problem der Instanzsegmentierung kann nun als Zuordnung des Pixels zu den D Objektdetektionen oder dem Hintergrundlabel betrachtet werden. Jede der Objekt-Detektionen D, welche für jedes Eingangsbild unterschiedlich ist, repräsentiert also eine potentielle Instanz. Für jedes der N Pixel im Bild wird eine zufällige multinomiale Variable $V = [V_1, V_2, ..., V_N]^T$ definiert. Dabei definiert $V_i \in 0, 1, 2, ..., D$ die Instanz, zu der der Pixel i zugehörig ist (0 ist das Hintergrundlabel). Im Falle der Instanzsegmentierung ist das Label selber, dem ein Pixel zugeordnet wird, nicht von Bedeutung. Allein, dass unterschiedliche Instanzen auch unterschiedlichen Labels zugeordnet werden, ist das Ziel der Instanzsegmentierung. So ist z.B. in Abbildung 2.15 die Zuordnung der Farben zu den Schafen nicht wichtig, sondern nur, dass diese unterschiedliche Labels (Farben) aufweisen. Um auf Basis der Semantischen Segmentierung und der Detektion, welche im vorherigen Subnetzwerk bestimmt wurden, die Variable V_i für jedes Pixel i zu bestimmen, wird wiederum ein Conditional Random Field über die Varibale V definiert. Die Energiefunktion der Zuweisung von v zu zu den Variablen V ist definiert als:

$$E(V = v) = \sum_i U(v_i) + \sum_{i<j} P(v_i, v_j) \tag{2.11}$$

Dabei ist die unäre Energie die Summe dreier Terme, die die Objektdetektion (Bounding-Boxen), die Semantische Segmentierung und die auf Vorwissen basierenden Forminformationen berücksichtigen.

$$U(v_i) = -ln[w_1\psi_{Box}(v_i) + w_2\psi_{Global}(v_i) + w_3\psi_{Shape}(v_i)] \tag{2.12}$$

w_1, w_2, w_3 sind dabei die Gewichtungen, die durch die Backpropagation gelernt werden. Die paarweisen Potentiale $P(v_i, v_j)$ bestehen aus dicht verbundenen Gauß-Potentialen, welche zu räumlicher und Form-Konsistenz führen.

Das gesamte Netzwerk der Instanzsegmentierung kann dabei Ende zu Ende gelernt werden. Für das Training der Instanzsegmentierung wird nur eine Loss-Funktion angewendet, die durch beide Subnetzwerke zurückpropagiert wird um alle Parameter zu lernen. Wie schon zuvor erwähnt, ist bei der Semantischen Instanzsegmentierung nicht das Label einer Instanz selber entscheidend, sondern nur, dass sich die Label zwischen den Instanzen unterscheiden. Die ground-truth Instanzsegmentierung G sei nun durch eine Menge an $g_1, g_2, ..., g_r$ Segmenten bestimmt, wobei g_i jeweils die Menge der Pixel beschreibt, die einer Instanz zugehörig sind. g_i ist dabei auch die Semantische Klasse zugewiesen, zu der die Instanz gehört. Die Prädiktion sei ebenfalls eine Menge an Segmenten $P := p_1, p_2, ..., p_s$ der gleichen Struktur. r und s, also die Anzahl der Instanzen in der Detektion und Prädiktion können dabei unterschiedlich sein. M sei nun die Menge an Permutationen, der Ground Truth G. Um die Loss Funktion zu berechnen, wird nun diejenige Permutation G^* aus M genommen, die die IoU mit der Prädiktion P maximiert.

$$G^* = \arg\max_{m \in M} IoU(m, P) \tag{2.13}$$

Auf Basis der passenden Permutation, ist es nun möglich, jede beliebige Loss-Funktion anzuwenden. In diesem Fall handelt es sich dabei um eine normale Kreuzentropie Loss-Funktion (siehe Kapitel 2.1.4).

Zur Zeit liefert die präsentierte Methode unter den pixelweisen Semantischen Instanzsegmentierungen die besten Ergebnisse. Mit einer Average Arecision (AP) von 20.0% liefert es zur Zeit im Vergleich zu allen Methoden die viertbesten Ergebnisse auf dem Cityscapes Datensatz. Eine Schwierigkeit bei diesem Modell ist jedoch dessen enorme Komplexität. Wie in dem obigen Abschnitt erläutert, werden hier zwei Subnetzwerke trainiert, die eine Detektion, eine Semantische Segmentierung und Conditional Random Fields kombinieren. Dies führt dazu, dass dieses Modell viele Fehlerquellen aufweist und sehr schwierig zu optimieren ist.

Diese Komplexität ist auch eine Eigenschaft, die die meisten Modelle für die pixelweise Instanzsegmentierung teilen. So basieren die Modelle aus [KLA+16] und [UCFB16] ebenfalls auf Architekturen, die viele Unteraufgaben zu einer Semantischen Instanzsegmentierung vereinen.

Die Methode aus [UCFB16], ist in Abbildung 2.16 dargestellt. Diese Methode basiert zunächst einmal auf dem FCN-8, welches, wie in der Abbildung 2.16 zu sehen, zunächst

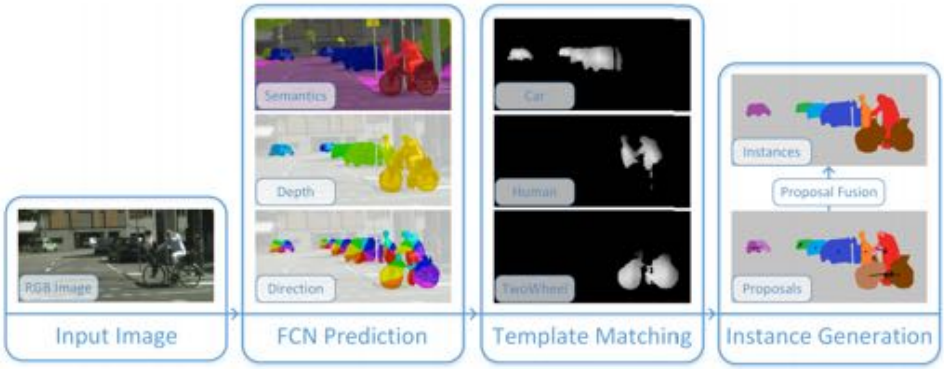

Abbildung 2.16: Die Abbildung (entnommen aus [UCFB16]) zeigt den Ablauf der Instanzsegmentierung aus [UCFB16]. Durch ein FCN-8 werden zunächst eine Tiefenkarte, eine Semantische Segmentierung und für jede Klasse, die Instanzen beinhaltet, pixelweise die diskretisierte Richtung zum korrespondierenden Instanzzentrum bestimmt ("FCN Prediction "). Auf Basis dieser Richtungskarte werden hiernach über ein klassenweises Templatematching die Instanzzentren bestimmt. Die Instanzen werden dann für jede Klasse generiert, indem alle Pixel dem Instanzzentrum zugeordnet werden, das in der prädizierten Richtung zum nächsten Instanzzentrum liegt. Über die Tiefeninformationen werden die Instanzvorschläge weiter verfeinert. Welcher Klasse eine Instanz zugeordnet wird, wird über die Semantische Segmentierung bestimmt (Klasse der Mehrheit der Pixel).

einmal eine Semantische Segmentierung, eine Tiefenkarte und eine Richtungskarte bestimmt. Die Semantische Segmentierung liefert dabei semantische Informationen über das Bild und ermöglicht es, Instanzen verschiedener Klassen voneinander und vom Hintergrund zu trennen. Die Tiefeninformationen werden bestimmt, da unterschiedliche Instanzen meist auch unterschiedliche Tiefen besitzen. Die Richtungskarte gibt dabei jeweils für jedes Pixel einer Klasse, die Instanzen beinhaltet, die diskretisierte Richtung zum Zentrum der Instanz zu der das Pixel gehört, an (siehe Abbildung 2.16). Dadurch werden implizit die Kanten einer Instanz kodiert, was vor allem bei sich überlappenden Instanzen derselben Klasse dazu führt, dass sich die Instanzen gut voneinander trennen lassen.

Im nächsten Schritt werden dann auf Basis der gegebenen Informationen die Instanzzentren bestimmt. Dies wird durch ein "template Matching" auf der Richtungskarte gemacht. Dies ist deshalb sinnvoll, da die Instanzzentren klare visuelle Muster in der Richtungskarte haben. Die Templates sind dabei rechteckig und haben dieselben visuellen Muster. Die Normalisierte Kreuzentropie wird dabei benutzt, um für jedes Pixel mit seinem entsprechenden Template zu korrelieren. Dabei wird für jede Kategorie eine solche Score-Map erstellt, wobei die Einträge pro Pixel jeweils die Wahrscheinlichkeit angeben, dass das Pixel ein Instanzzentrum ist. Um nun die temporären Instanzzentren zu lokalisieren, wird in der Score-Map nach Maxima gesucht (Mehrfach-Detektionen durch Non Maximum Suppression verhindert). Hiernach werden für jede Kategorie getrennt die Instanzen vorhergesagt. Jedes Pixel mit einer vorhergesagten Richtung wird dem nächsten temporären Instanzzentrum zugewiesen, wo die relative Position und die vorhergesagten Richtungen mit dem Instanzzentrum übereinstimmen. Dadurch werden erste Instanzvorschläge generiert. Durch ein Postprocessing, basierend auf einer Tiefen-

karte werden die Instanzzugehörigkeiten noch weiter verfeinert. Nachdem alle Instanzen gefunden wurden wird diesen das Klassenlabel zugewiesen, zu dem die Mehrheit der Pixel in der Instanz laut Semantischer Segmentierung gehören.

Dieser Ansatz liefert auf dem Cityscapes Datensatz mit eine Average Precission von 8.9% keine ausreichend guten Ergebnisse. Zudem kann man auch an dieser Architektur die hohe Komplexität, die die meisten pixelweisen Instanzsegmentierungen haben, beobachten.

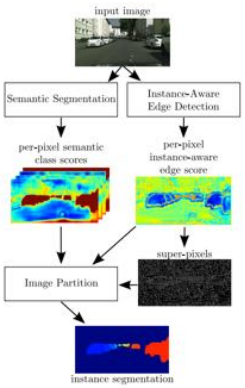

Abbildung 2.17: Die Abbildung (entnommen aus [KLA+16]) zeigt den generellen Ablauf der Instanzsegmentierung aus [KLA+16]. Dabei werden auf Basis des Eingabebildes zunächst pixelweise eine Semantische Segmentierung und eine instanzbezogene Kantendetektion prädiziert. Auf Basis letzterer wird das Bild dann in Superpixel aufgeteilt. Alle diese Informationen dienen dann einem Conditional Random Field als Eingabe, das die Superpixel Instanzen zuordnet und somit die Instanzsegmentierung bestimmt.

Ein weiterer Ansatz, die pixelweise Semantische Instanzsegmentierung zu realisieren, ist, eine Semantische Segmentierung zusammen mit Instanzgrenzen (Kanten) abzuleiten. Die Instanzen werden dann dadurch bestimmt, dass die Semantische Segmentierung durch die prädizierten Kanten der Instanzen aufgeteilt wird. In [vdBOM17] wurde ein derartiger Ansatz vorgestellt, welcher allerdings im Vergleich sehr schlechte Ergebnisse liefert. So belegt er bei Cityscapes Benchmark mit einer AP von 2.3% den letzten Platz. Dass das alleinige Prädizieren von Instanzkanten nicht funktioniert, ist damit zu begründen, dass es sich bei den Kanten um sehr feine Strukturen handelt (nur einen Pixel breit), die mithin beim Gradientenabstieg ein geringes Gewicht haben und bei der Prädiktion zudem häufig Lücken aufweisen.

In [KLA+16], dessen Architektur in Abbildung 2.17 zu sehen ist, wurde deshalb eine Architektur präsentiert, die die Informationen einer Semantischen Segmentierung mit den instanzbezogenen Kanteninformationen über ein komplexes graphisches Modell kombiniert. Die Semantische Segmentierung wird dabei über eine vom FCN-8 abgeleitete Architektur bestimmt. Auf Basis der am geringsten aufgelösten Feature-Map dieser Architektur wird die instanzbezogene Kantendetektion über zwei Schichten mit 1x1 Faltungen bestimmt. Dies geschieht deshalb auf der am geringsten aufgelösten Feature-Map, da durch die Interpolations-Schichten die Kanten andernfalls zu breit und zu glatt dargestellt würden (gleichzeitig viele Semantische Informationen). Die Ausgabe dieser Schicht ist eine Wahrscheinlichkeitskarte, die für jedes Pixel angibt, wie wahrscheinlich

es ist, dass dieses Pixel an eine Kante einer Instanz grenzt. Diese beiden Informationen werden nun in einem CRF miteinander kombiniert, um die Instanzsegmentierung zu bestimmen. Zur Reduktion der Komplexität des Problems wird das Bild dabei zunächst einmal durch die instanzbezogenen Kanteninformationen (Wahrscheinlichkeitskarte) in Superpixel aufgeteilt. Diese Superpixel repräsentieren nun die Knoten V im Graphen $G = V, E$. Zwischen zwei örtlich benachbarten Superpixeln werden dann die Kanten E gezogen. Dabei werden jedem Superpixel die Wahrscheinlichkeiten $\alpha_{u,l}$ und $b_{u,v}$ zugewiesen. Dabei ist $u \in$ der Menge der Knoten V und $l \in$ der Menge an Labeln L. $\alpha_{u,l}$ gibt mithin die Wahrscheinlichkeit an, dass der Superpixel u zum Label l gehört und $b_{u,v}$ die Wahrscheinlichkeit, dass zwischen den Superpixeln u und v eine Instanzkante verläuft. $\alpha_{u,l}$ wird dabei bestimmt, indem die Score-Map der Labels im Superpixel u gemittelt wird. $b_{u,v}$ wird bestimmt, indem die b Werte der Pixel in den beiden Superpixeln u und v, die an der Grenze zwischen u und v liegen, gemittelt werden. Die Wahrscheinlichkeit $\beta_{l,\bar{l}}$ ist dabei ein a-priori Term, der angibt, wie wahrscheinlich es ist, zwischen l und $\bar{l}$ eine Kante zu haben.

Die Formulierung der Istanzsegmentierung wird dann in zwei graphenbasierte Sub-Probleme aufgeteilt. Zu einen wird ein CRF auf Basis der beschriebenen Wahrscheinlichkeiten konstruiert, der die Semantische Segmentierung verbessert. Zum anderen wird, ausgehend vom Zustand, dass alle Superpixel bereits ein semantisches Klassen-Label zugewiesen bekommen haben, versucht, die Partitionierung des Graphens zu finden, sodass alle Superpixel, die zu einer Instanz gehören, zu einer zusammenhängenden Region gehören. Dabei werden nur Partitionierungen zwischen benachbarten Superpixeln der gleichen Größe bestimmt, da unterschiedliche Klassen auch gleichzeitig unterschiedliche Instanzen bedeuten. Diese Aufgabe wird als Multicut Problem formuliert und zusammen mit dem CRF optimiert.

Diese Methode liefert unter den pixelweisen Instanzsegmentierungen auf dem Cityscapes Datensatz mit einer AP von 13.0% die drittbesten Ergebnisse.

2.4.2 Stufenweise Semantische Instanzsegmentierung

Eine der größten Schwierigkeiten bei der pixelweisen Instanzsegmentierung ist sicherlich die Definition einer Loss Funktion, die dazu führt, dass die Semantische Segmentierung zwischen Instanzen unterscheiden kann. Dies ist vor allem deshalb so schwierig, da die Anzahl der Instanzen in einem Bild nicht a priori bekannt ist. Insofern ist es möglich, die Semantische Instanzsegmentierung als eine Art Hybrid zwischen einer Regression und einer Klassifikation zu betrachten. Eine Regression insofern, als dass ein jedes Bild eine nicht definierte Anzahl an Instanzen beinhaltet und eine Klassifikation, als dass die im Bild gegebenen Instanzen zu einer klar definierten Menge an Klassen zugeordnet werden.

Die im Folgenden näher erklärten Ansätze für die Semantische Instanzsegmentierung beinhalten nun genau diese Struktur. Dabei handelt es sich um eine zweistufige Architektur, die zunächst über einen Detektor versucht, alle Instanzen in einem Bild durch eine Bounding-Box zu lokalisieren und anschließend innerhalb der Boundig-Box das

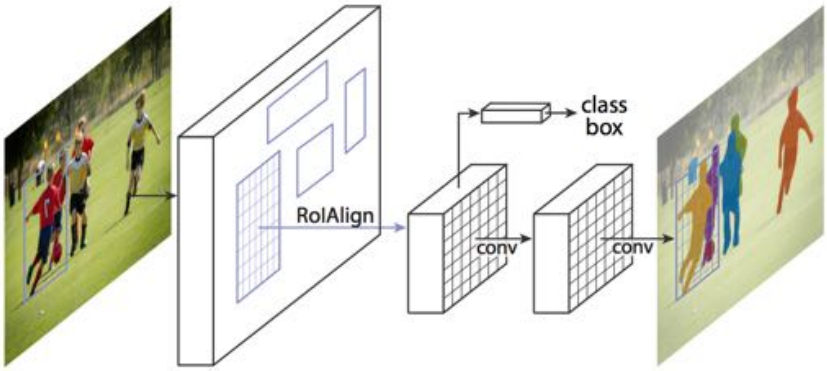

Abbildung 2.18: In der Abbildung, welche aus [HGDG17] entnommen ist, ist die generelle Architektur des Ansatzes aus [HGDG17] zu sehen. Dabei werden zunächst die Instanzvorschläge durch ein RPN generiert. Diese werden dann nachfolgend klassifiziert und es wird eine Instanzmaske in der ROI generiert. Dieser grobe Ablauf ist repräsentativ für die meisten zweistufigen Instanzsegmentierungen.

Objekt segmentiert. Diese generelle Architektur ist in Abbildung 2.18 zu sehen. Bei der Objektdetektion handelt es sich dabei um eine Regressions- und bei der Segmentierung um eine pixelweise Klassifikationsaufgabe. Die Definition des Loss ist dabei wesentlich einfacher als bei der pixelweisen Instanzsegmentierung, da innerhalb einer Bounding Box nur noch eine zu segmentierende Instanz vorhanden ist, weshalb es sich hierbei wiederum nur noch um eine pixelweise Klassifikation handelt, für die die Definition eines Loss recht einfach ist.

Eine andere Sichtweise auf diese Art der Architektur ist es, die Aufgabe der Semantischen Instanzsegmentierung als eine Aufgabe zu betrachten, die vom Generellen zum Speziellen gelöst wird. Dabei wird also durch die Detektion der Instanzen zunächst eine generelle Information über den Ort und die Größe der im Bild existierenden Objekte generiert und durch die Segmentierung dieser Instanzen die spezielle Form dieser Objekte innerhalb der Bounding-Box definiert. Eine derartige Vorgehensweise ergibt vielleicht auch deshalb Sinn, da die Wahrnehmung von Instanzen beim Menschen auf ähnliche Art und Weise abläuft.

Die besten Ergebnisse für die Instanzsegmentierung und damit auch die besten Ergebnisse unter diesen zweistufigen Ansätzen liefert zur Zeit das Mask-RCNN [HGDG17]. Dieses erreicht z.B. auf dem Cityscapes-Datensatz mit einer AP von 32.0 den ersten Platz im Benchmark. Das Mask-RCNN basiert auf dem Faster-RCNN, einem Detektor, der in Abschnitt 2.3 noch näher erläutert wird. Generell besteht dieser Detektor aus einem FCN, das eine Feature-Map für die Detektion erstellt und normalerweise aus einem, Klassifikationsnetzwerk entnommen wurde. Durch ein sogenanntes Region-Proposal-Network werden potentielle Bounding-Boxen für die gegebenen Instanzen detektiert. Innerhalb dieser Kandidaten-Bounding-Boxen wird dann die Feature-Map gepoolt, indem ein immer gleich großes räumliches $m \times m$ Gitter (z.B. 7x7) über die Bounding-Box gelegt wird. Alle Werte der Feature-Map innerhalb eines Elementes dieses Gitters werden dann zusammengefasst, sodass die räumlichen Eigenschaften der Feature-Map erhalten bleiben. Diesen Vorgang nennt man ROI-Pooling. Auf Basis dieser neuen normierten

Feature-Map wird dann die Klasse der Bounding-Box bestimmt und die Bounding-Box besser an die vorhandene Instanz angepasst (Bounding-Box Regression). Dies ist auch im grau unterlegten Bereich von Abbildung 2.19 abgebildet.

Um eine Instanz Segmentierung zu generieren, beinhaltet die Architektur des Mask-RCNN einen weiteren Zweig, welcher auf der Feature-Map nach dem ROI-Pooling basiert. Wie in Abbildung 2.19 zu sehen (Zweig mit Label "mask ") wird dabei durch ein beliebiges (austauschbares) Fully Convolutional Netzwerk für jede Semantische Klasse unabhängig eine Maske für die Instanzsegmentierung erstellt. Als ground truth während

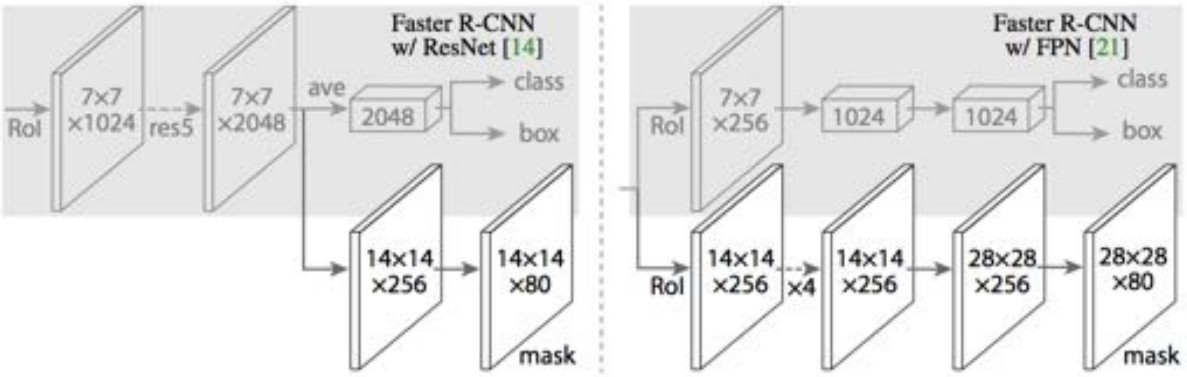

Abbildung 2.19: Die Abbildung, welche aus [HGDG17] stammt, zeigt die Architektur des Mask-RCNN nach der Erstellung der Instanzvorschläge und dem ROI-Pooling. Dabei existieren, wie zu sehen, pro Bounding-Box (ROI) zwei Zweige, zum einen ein austauschbares FCN (hier z.B. ResNet links und FPN rechts) um für jede Klasse eine Instanzmaske zu prädizieren und zum anderen ein Klassifikator, der die Klasse der Bounding-Box und die Bounding-Box-Regression bestimmt. Am Ende wird die Instanzmaske der Klasse übernommen, welche die Klassifikation der Bounding-Box ergeben hat

.

des Trainings dient dabei die Instanzsegmentierung, die zur entsprechenden Boundig-Box korrespondiert. Bei dem Loss handelt es sich um einen über die Pixel gemittelten binären Kreuzentropie Loss, der jeweils einzeln und damit unabhängig für die Segmentierungs-Maske jeder Klasse bestimmt wird. Daraus folgt, dass die einzelnen Masken der Klassen während des Trainings nicht miteinander konkurrieren. Als Instanzsegmentierung wird dann die Maske der Klasse gewählt, welche vom Klassifikationszweig vorhergesagt wird. Die Pädiktion der Klasse und der Maske sind dadurch folglich voneinander getrennt. Dies ist ein entscheidender Unterschied gegenüber den meisten Verfahren zur Semantischen Segmentierung / Instanzsegmentierung und ist ein Grund für den Erfolg des Mask-RCNN.

Ein weiterer Grund für die guten Ergebnisse des Mask-RCNN ist zudem die verwendete Methode zum ROI-Pooling. Beim Poolen der Feature-Map ist es wichtig, dass dessen räumliche Eigenschaften erhalten bleiben. Die einzelnen Elemente der Feature-Map repräsentieren nämlich rezeptive Felder im Originalbild. Über diese Korrespondenzen kann man dann schließlich die Instanzsegmentierung, welche auf der gering aufgelösten Feature-Map bestimmt wurde, zurück auf das Originalbild (durch Interpolation) abbilden. Wenn nun beim ROI-Pooling die Feature Map durch das Gitter räumlich aufgeteilt

wird, kann es sein, dass die Aufteilung nicht glatt aufgeht, z.B. im Fall einer Aufteilung einer 13×13 Feature-Map durch ein 7×7 Gitter. Im originalen Faster-RCNN wird in solchen Fällen auf- oder abgerundet um die Elemente der Feature-Map auf das Gitter abzubilden. Es wird also, um die Abbildung der Feature-Map auf das Gitter zu bestimmen, über jede räumliche Position x in der Feature-Map $[x/13]$ (bei einem 13×13 Gitter) bestimmt. Das Mask-RCNN bestimmt hier eine Bilineare Interpolation (also $x/13$ anstatt von $[x/13]$) und erhält dadurch die räumlichen Korrespondenzen besser.

Das Mask-RCNN liefert zur Zeit nicht nur die besten Ergebnisse im Bereich der Semantischen Instanzsegmentierung, sondern auch in der Detektion. Ein Grund dafür ist ebenfalls das ROI-Pooling durch die Bilineare Interpolation. Wie in [HGDG17] Mask RCNN dargelegt, führen auch das gemeinsame Training der Detektion und der pixelweisen Segmentierung zu Verbesserungen. Da die Fähigkeit dieser zweistufigen Architekturen, ein Objekt zu segmentieren, davon abhängt, dass dieses Objekt zunächst einmal detektiert wurde, ist diese Verbesserung der Detektionseigenschaften elementar wichtig für die Instanzsegmentierung.

Allgemein ist der zugrunde liegende Detektor bei dieser Art von Ansätzen deshalb ein wichtiges Unterscheidungsmerkmal. In 2.3 wird auf mögliche und schon im Zuge der Instanzsegmentierung verwendete Detektions-Architekturen noch näher eingegangen. In [LQD+16] z.B. wird ein Detektor verwendet, der versucht, das Problem anzugehen, dass die Score-Maps in FCNs nur die Wahrscheinlichkeit jedes Pixels zu einer Klasse zu gehören, bestimmt. Dabei ist diese Score-Map translationsinvariant und generiert keine Informationen über unterschiedliche Objekt-Kategorien. In [LQD+16] wird deshalb eine Architektur vorgeschlagen (siehe Abbildung 2.20), die positions- und instanzabhängige Scores erstellt. Die Erstellung dieser Score-Map ist in 2.3 noch näher erklärt. Allgemein werden hierbei, wie in Abbildung 2.20 zu sehen, nach dem Region Proposal Network, die Bounding-Box Vorschläge durch ein gleichmäßiges $k \times k$ Gitter räumlich unterteilt. In jedem Element dieser Unterteilung wird dann pixelweise die Wahrscheinlichkeit (Score) berechnet, dass das Pixel zu einer bestimmten Kategorie ($\in C$) oder dem Hintergrund gehört und die relative Position des Gitterelementes im Verhältnis zum Objekt hat. Dies wäre z.B. in Abbildung 2.20 für das Gitterelement links unten die pixelweise Wahrscheinlichkeit zur Kategorie Person zu gehören und relativ zur Person links unten zu sein.

Um die gesamte Score-Map für einen Bounding-Box Vorschlag zu berechnen, wird die unterteilte Feature-Map dann wieder zusammengesetzt. Hiernach hat die Score-Map der Bounding-Box die Form $2 * (C + 1)$, da für jede Kategorie $\in C$ (einschließlich des Hintergrundes also $\in C + 1$) sowohl die Wahrscheinlichkeit bestimmt, wurde Vordergrund oder Hintergrund in Bezug auf diese Kategorie zu sein. Der Gedanke dahinter ist, dass für eine ROI in jedem Score-Paar (Vordergrund,Hintergrund) immer der eine Wert hoch und der jeweils andere Wert niedrig sein sollte, wenn die ROI zur aktuellen Kategorie zugehörig ist. Für eine negative ROI, die also nicht zu der aktuellen ROI gehört, sollten alle Scores im Score-Paar klein sein. Sowohl der Hintergrund- als auch der Vordergrund-Score werden also kategoriespezifisch bestimmt und ergeben

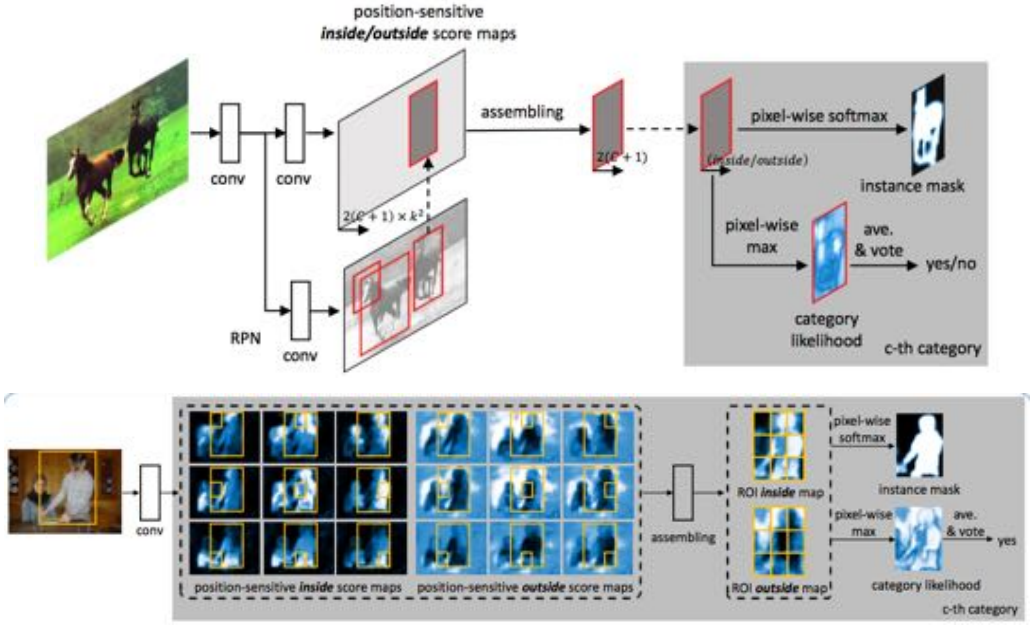

Abbildung 2.20: Die Abbildungen sind aus [LQD+16] entnommen. Die obige zeigt dabei, wie die generelle Architektur der Instanzsegmentierung aufgebaut ist. Auf Basis der Feature-Map werden durch ein RPN Bounding-Box-Vorschläge generiert und zudem pixelweise für das ganze Bild eine Score-Map prädiziert, die die Wahrscheinlichkeiten dafür enthält, dass der aktuelle Pixel eine bestimmte relative Position zu einer Instanz einer bestimmten Klasse einnimmt. Diese Score-Map wird innerhalb der ROIs der Bounding-Box-Vorschläge gepoolt und auf deren Basis die Klasse und Segmentierung der aktuellen Instanz bestimmt. Im unteren Bild ist ein Beispiel für diese pixelweise Score-Map zu sehen. Dabei sind die relativen Positionen durch ein 3×3 Gitter bestimmt.

zusammen Informationen über das Vorhandensein einer Instanz dieser Kategorie und deren Maske.

Um die Segmentierungsmaske einer Kategorie zu bestimmen, wird dann eine Softmax-Funktion auf die Vordergrund-Scores angewendet, die die Wahrscheinlichkeit pro Pixel bestimmt, dass er dem Vordergrund also der Instanz zugehörig ist. Um die Klasse der gesamten Maske zu bestimmen, wird auf den Scores zunächst pixelweise das Maximum aus Hintergrund- und Vordergrund-Wahrscheinlichkeit der gesamten Bounding-Box berechnet und über die Bounding-Box durch ein Average-Pooling gemittelt. Dadurch werden auch die Pixel, die zum Hintergrund gehören, mit in die Klassifikation einbezogen, sodass auch der Kontext, in dem die Instanz sich befindet, berücksichtigt wird. Um schließlich die Kategorie zu bestimmen, zu der die Detektion gehört, wird diejenige genommen, die den größten Kategorie-Score besitzt. Die Maske der Instanz wird dadurch bestimmt, dass alle anderen Bounding-Boxen, die mit der aktuellen eine IOU > 0.5 haben gemeinsam betrachtet werden. Dabei werden die Vordergrund-Scores der einzelnen Detektionen pixelweise gemittelt und mit ihrer Kategorie-Score gewichtet. Die gemittelte Maske wird dann als binärer Output verwendet.

Jede ROI hat während des Trainings einen Soft-Max Klassifikations-Loss, einen Bounding-Box Regressions-Loss (trainiert eine nachträgliche Anpassung der Bounding-Box an das

Objekt) und einen Soft-max Segmentierungs-Loss für die Vordergrundklasse der Ground-Truth. Ähnlich wie beim Mask-RCNN [HGDG17] stehen die einzelnen Masken der Kategorien während des Trainings also nicht in Konkurrenz miteinander.

Ein häufiges Problem bei diesen Ansätzen, die auf einer Detektion und einer nachfolgenden Segmentierung basieren, ist, dass bei einer schlechten Detektion, die das Objekt nicht gänzlich umfasst, auch die Segmentierung des Objektes notwendigerweise unvollständig ist. Ein Beispiel hierfür ist z.B. in Abbildung 2.21 abgebildet. Um mit diesem Problem umzugehen, wird in [HHS16] ein Ansatz eingeführt, Objekten, die nicht gänzlich von der Bounding-Box umfasst wurden, durch Informationen über ihre Form über die Bounding-Box hinaus zu segmentieren. Dabei wird durch ein FCN eine pixelweise Distanz-Transformation $D(p)$ berechnet, sodass der Wert in jedem Pixel p die Distanz zum nächstgelegenen Objektkantenpixel Q repräsentiert, oder für alle nicht Objektpixel, die Tatsache, dass sie zum Hintergrund gehören. Die Distanztransformation ist entsprechend folgendermaßen definiert:

$$D(p) = \min(\min_{\forall q \in Q}(\lceil d(p,q) \rceil, R)) \tag{2.14}$$

Dabei bezeichnet $d(p,q)$ die euklidische Distanz zwischen dem Randpixeln $q \in Q$ und dem aktuellen Objektpixel p (durch $\lceil d(p,q) \rceil$ auf ganzzahlige Werte aufgerundet). Durch den Schwellenwert R wird die Distanz dabei begrenzt, sodass sie besser durch ein FCN vorhergesagt werden kann. Allgemein bietet diese pixelweise Kodierung den Vorteil, dass die Distanzwerte redundant und damit robust gegenüber Rauschen sind und gleichzeitig Informationen über die Lage der Objektgrenze beinhalten.

Diese Distanztransformation kann zudem leicht in eine pixelweise Klassifikationsaufgabe umgewandelt werden. Dabei werden die möglichen Distanzwerte zwischen 0 und dem Schwellenwert R in $n \in K$ Stufen quantisiert. Diese Stufen können dann als Label interpretiert werden, die von einem FCN pixelweise prädiziert werden können. Die Distanztransformation ist dann für jeden Pixel p folgendermaßen definiert:

$$D(p) = \sum_{n=1}^{K} r_n \cdot b_n(p), \sum_{n=1}^{K} b_n(p) = 1 \tag{2.15}$$

b bezeichnet dabei den prädizierten binären K dimensionalen Vektor mit den quantisierten Distanzstufen bei Pixel p und r_n ist der Distanzwert, der zum Element b_n korrespondiert. Eine derartige durch R begrenzte Distanztransformation kann z.B. in Abbildung 2.21 im rechten Bild betrachten werden. Diese Distanztransformation wird dann für jede prädizierte Bounding-Box bestimmt. In der rechten Spalte der linken Abbildung aus Abbildung 2.21 ist zu sehen, was passiert, wenn die Transformation für eine Bounding-Box bestimmt wird, die das Objekt nicht gänzlich umfasst. In den Bereichen, in denen die Bounding-Box das Objket durchtrennt, beinhaltet die Distanztransformation die Information, dass die Objektgrenze über den Rand der Bounding-Box hinaus liegt. Um nun das gesamte Objekt zu segmentieren, wird nun die inverse Distanztransformation angewendet. Dabei wird jedem Pixel ein binärer Kreis mit Radius $D(p)$

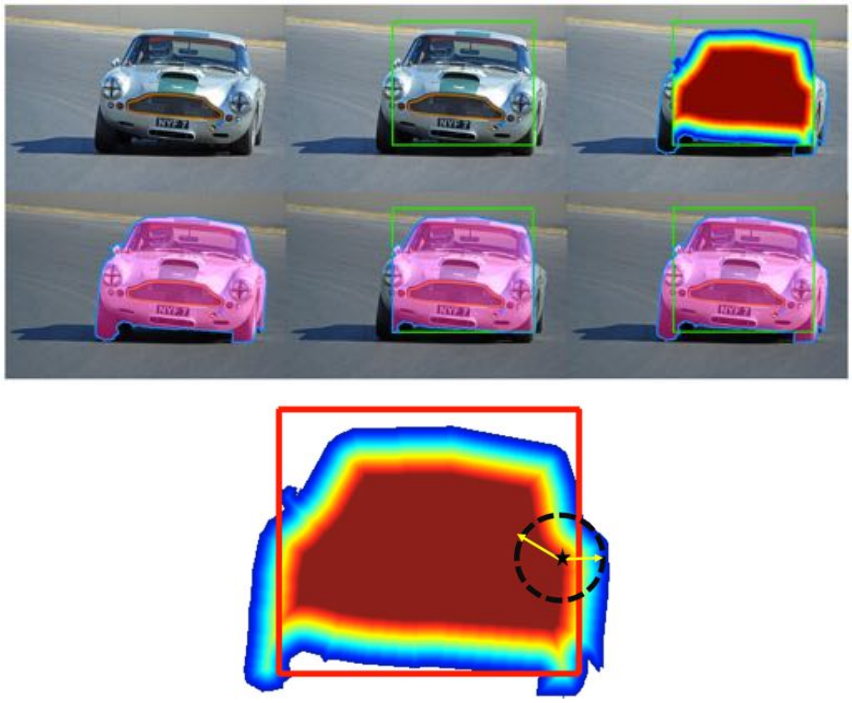

Abbildung 2.21: Die Abbildung stammt aus [HHS16] und zeigt oben, wie die Distanztransformation eine Segmentierung über die Instanzgrenzen erweitern kann und unten, wie die Distanztransformation definiert ist.

zugeordnet. Die Objektmaske M wird dann berechnet, indem die Vereinigung aller Kreise berechnet wird. $T(p, r)$ sei der Kreis von Radius r an Pixel p, die Objektmaske kann dann folgendermaßen ausgedrückt werden:

$$M = \bigcup_p T(p, D(p)) = \bigcup_p T(p, \sum_{n=1}^{K} r_n \cdot b_n(p)) = \bigcup_{n=1}^{K} \bigcup_p T(p, r_n \cdot b_n(p)) = \bigcup_{n=1}^{K} T(\cdot, r_n) * B_n \tag{2.16}$$

Dabei bezeichnet B_n die binäre pixelweise Karte für die n-te binäre Distanzquantisierung. Dies sind also alle Pixel für die die n-te Distanz im Vektor b vorhergesagt wurde. $*$ steht dabei für die Faltungsoperation, welche durch ein CNN implementierbar ist. $T(\cdot, r_n) * B_n$ kann dabei auch als die Anwendung eines morphologischen Operators betrachtet werden, der bei jedem Pixel mit der Distanzquantisierung b_n mit einem Radius der zu b_n korrespondierenden Distanz r_n zur Objektgrenze angewendet wird. So werden alle Pixel im Radius r_n mit in das Objekt einbezogen. Durch die Vereinigung über alle Distanzquantisierungen kann so folglich die inverse Distanztransformation berechnet werden.

Das in [HHS16] beschriebene Netzwerk besteht aus drei Subnetzwerken. Diese sind den drei Unteraufgaben, der Generierung von Bounding-Box-Vorschlägen, der Objektmaskenprädiktion und der Objektklassifikation zugeordnet und laufen in dieser Reihenfolge

ab. Dabei wird nach der ersten Phase, also der Generierung der Bounding-Box Vorschläge, im Subnetzwerk zur Generierung der Objektmaske die beschriebene inverse Distanztransformation angewendet. Auf Basis der Objektmaske und der Bounding-Box (Pooling der Feature-Map in diesen Bereichen) wird dann eine Bounding-Box Regression zur Anpassung der Bounding-Box an das Objekt und die Objektklassifikation durchgeführt. Die Instanzsegmentierung besteht dann also aus der Objektmaske mit der prädizierten Klasse.

2.5 Wichtige Datensätze und Qualitätsmetriken

2.5.1 Der COCO und Pascal Datensatz

Bei den Datensätzen COCO [LMB+14] und Pascal (2012 [EVGW+b] / 2007 [EVGW+a]) handelt es sich um Datensätze, die Bilddaten von alltägliche Szenen enthalten. Dabei wird eine breite Variabilität an Objektklassen abgedeckt, die in diesen Bildern zu sehen sind. Diese Datensätze bieten Annotationen für die Semantische Segmentierung, die Detektion und die Semantische Instanzsegmentierung. Für beide Datensätze existieren öffentlich einsehbare Ranglisten, die die Leistung verschiedener Methoden auf den jeweiligen Testdatensätzen vergleichen. In dieser Arbeit wurden diese Vergleiche benutzt, um einen Überblick über die besten Ansätze für die einzelnen Aufgaben zu gewinnen.

2.5.2 Der Cityscapes Datensatz

Der Cityscapes Datensatz, der für die Tests verwendet wurde, umfasst 5000 Bilder von urbanen Straßenszenen. Ein Beispiel für eine typische Straßenszene im Datensatz kann man z.B. in Abbildung 2.22 sehen. Die Bilder sind in 50 verschiedenen Städten unter verschiedenen Konditionen und Tageszeiten aufgenommen und haben eine HD Auflösung von 2048×1024 Pixeln.

Für die einzelnen Bilder sind jeweils ground-truth Daten, also von Menschen erstellte Daten, für die Semantische Segmentierung und die Semantische Instanzsegmentierung gegeben. Beispiele für diese ground-truth Daten sind für die Semantische Segmentierung in Abbildung 2.5 und für die Semantische Instanzsegmentierung in Abbildung 2.14 zu sehen. Die Annotationen beinhalten 30 Klassen, welche in die Kategorien „Fläche“, „Menschen“, „Fahrzeuge“, „Konstruktion“, „Objekte“, „Natur“ und „Himmel“ aufgeteilt sind. Ein näherer Überblick über die Beschaffenheit der Annotationen wird in Kapitel 4.3 gegeben. Die Trainingsmenge umfasst 2975 Bilder, die Validierungsmenge 500 und eine Testmenge von 1525 Bildern. Aufgrund der Tatsache, dass die ground-truth Bilder der Testmenge nicht vorhanden waren, wurde die Validierungsmenge zum Testen verwendet.

Abbildung 2.22: Die hier gezeigten Bilder stammen aus dem Cityscapes-Datensatz [COR+16]. Diese Abbildung gibt einen Überblick über typische Bilder dieses Datensatzes.

2.5.3 Semantische Segmentierung Qualitätsmetriken

Um die Ergebnisse der Segmentierung zu evaluieren, werden die Qualitätsmetriken IoU und iIoU verwendet, wie es auch im Cityscapes Benchmark gemacht wird. Diese Metriken sind skalare Werte $\in [0, 1]$, die pro Klasse über die gesamte Testmenge (oder in diesem Fall Validierungsmenge) berechnet werden. Die Intersection over Union (IoU) wird dabei folgendermaßen berechnet:

$$IoU = TP/(TP + FP + FN) \tag{2.17}$$

Wobei pro Klasse TP für True Positive also richtig positiv, FP für False Positive also für Falsch Richtig und FN für Falsch Negativ steht. Diese Metrik kann auch so interpretiert werden, dass man die Überlappung der bestimmten Segmentierung mit seiner ground-truth mit der Vereinigung der Segmentierung mit der ground-truth teilt. Die IoU ist jedoch insofern gebiast, als dass große Instanzen einen größeren Einfluss auf die IoU haben als kleine. Um dieses Problem anzugehen, wird zusätzlich die Instance Intersection over Union (iIoU) berechnet:

$$iIoU = iTP/(iTP + FP + iFN) \tag{2.18}$$

iTP und iFN werden dabei berechnet, indem der Beitrag jedes Pixels mit dem Verhältnis der durchschnittlichen Klassengröße zu der korrespondierenden ground-truth Instanzgröße gewichtet wird.

2.5.4 Detektion Qualitätsmetriken

Die Ergebnisse der Detektion liegen in der Form vor, dass jede Bounding-Box jeweils einen Konfidenzwert besitzt, welcher vom Netzwerk bestimmt wurde und als Wahrscheinlichkeit interpretiert werden kann, dass diese Bounding-Box ein Objekt der prädizierten Klasse umfasst. Man betrachtet für einen bestimmten Konfidenzschwellenwert alle Bounding-Boxen, deren Konfidenz diesen übersteigt, als eine Detektion (als positiven Fall). Für einen solchen Schwellenwert kann man dann den Recall und die Precission berechnen. Der Recall bezeichnet das Verhältnis der Richtig Positiven TP Detektionen zu allen richtigen Detektionen, also den Richtig Positiven addiert mit den Falsch Negativen FN.

$$Recall = TP/(TP + FN) \tag{2.19}$$

Die Precission bezeichnet das Verhältnis der Richtig Positiven zu allen Positiven Prädiktionen (Falsch Positive+ Richtig Positive $FP + TP$).

$$Precission = TP/(TP + FP) \tag{2.20}$$

Die Recall Precission Kurve ergibt sich dann, indem über viele Konfidenzschwellenwerte die Precission und der korrespondierende Recall-Wert berechnet wird. Eine große Fläche unterhalb der Kurve ergibt sich genau dann, wenn für jeden Konfidenzschwellenwert sowohl für den Recall als auch für die Precission ein hoher Wert existiert.

Als Richtig Positive Prädiktion gilt eine Bounding-Box dann, wenn die IoU dieser Bounding-Box mit der korrespondierenden ground-truth Bounding-Box einen Schwellenwert übersteigt und die Klasse der ground-truth Bounding-Box prädiziert wurde. Wenn nicht anders angegeben, ist dieser Schwellenwert in dieser Arbeit 0.5. Wenn keine korrespondierende Bounding-Box zu der ground-truth Bounding-Box existiert, so handelt es sich um eine Falsch Negative FN Bounding-Box. Wenn eine falsche Klasse für eine Bounding-Box prädiziert wurde oder die IoU geringer als der Schwellenwert ist, so handelt es sich um einen Falsch Positiven FP Fall.

Zur Evaluation der Detektionsarchitektur, die in dieser Arbeit entwickelt wurde, diente die Average Precission (AP), wie sie auch im Zusammenhang mit der Pascal-VOC 2007 Challange [EVGW+a] verwendet wurde, und wird jeweils klassenweise berechnet. Die Average Precission fasst dabei die Form der Precission Recall Kurve zusammen. Die AP ist nämlich ein Maß für die Fläche unterhalb dieser Kurve und ist folgendermaßen definiert:

$$AP = \frac{1}{AnzahlKlassen} \sum_{r \in \{0, 0.1, ..., 1\}} p_{interp}(r) \tag{2.21}$$

r bezeichnet dabei den Recallschwellenwert, welches Element der elf Schwellenwerte $\{0, 0.1, ..., 1\}$ ist. $p_{interp}(r)$ ist dabei die interpolierte Precission am Recallschwellenwert

r und ist definiert als die maximale Precission, für die der korrespondierende Recallwert den Schwellenwert r übersteigt:

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r}\geq r} p(\tilde{r}) \tag{2.22}$$

Das Ziel hinter dieser Interpolation der Recall Precission Kurve ist es, den Einfluss von Rauschen zu reduzieren, welches durch Variationen in der Konfidenz von Instanzen entsteht. Um eine hohe Bewertung zu erreichen, ist es erforderlich, dass ein Precissionwert an allen Recallschwellenwerten $r \in \{0, 0.1, ..., 1\}$ existiert, sodass Methoden, die nur eine Untermenge an Bounding-Boxen mit einer hohen Precission betrachten, bestraft werden.

2.5.5 Semantische Instanzsegmentierung Qualitätsmetriken

Die hier verwendete Average Precission stammt aus [HAGM14]. Die durch das Netzwerk prädizierten Daten haben dabei die Form, dass anstatt von Bounding-Boxen pixelweise Masken der Instanz mit einem entsprechenden Konfidenzwert vorliegen. Dabei unterscheidet sich diese Form der AP von der aus Kapitel 2.5.4 zunächst einmal dadurch, dass die IoU zwischen der pixelweisen ground-truth Maske der Instanz und der prädizierten Maske berechnet wird, anstatt diese auf Basis von Bounding-Boxen zu bestimmen. Zudem werden IoU Schwellenwerte zwischen 0.5 und 0.95 in Schritten von 0.05 betrachtet, für die jeweils die AP berechnet wird. Der über alle Schwellenwerte gemittelte AP-Wert wird dann als AP bezeichnet. Die Metrik $AP50\%$ repräsentiert die AP bei einem IoU Schwellenwert von 0.5. Die mehrmalige Prädiktion einer ground-truth Instanz wird als Falsch Positiver Fall bestraft.

Kapitel 3

Methoden

Die Idee, die in dieser Masterarbeit verfolgt wurde, ist nun die, ein Modell zu entwerfen, das sich gut in die Erstellung des Szenen-Modells im Selbstfahrenden Auto einfügt. Dabei besteht neben der Qualität der Instanzsegmentierung auch die Maxime, dass die Architektur möglichst ressourcenschonend arbeiten sollte. Das bedeutet eine möglichst effiziente Speicherauslastung, einen möglichst geringen Energieverbrauch und eine möglichst schnelle Inferenz, die eine Ausführung in Echtzeit, also mindestens 10 Frames per Second (FPS) ermöglicht. Im nächsten Abschnitt wird nun zunächst einmal darauf eingegangen, wie die vorhandenen in Abschnitt 2.4 vogestellten Methoden sich in Bezug auf diese Kriterien verhalten.

3.1 Abwägung zwischen der pixelweisen und der zweistufigen Architektur

Die in dieser Arbeit entwickelte Methode baut auf den in Kapitel 2.4 vorgestellten Verfahren für die Instanzsegmentierung auf. Tabelle 3.1 gibt einen Überblick über diese bereits vorhandenen Methoden, deren Eigenschaften und Leistung auf verschiedenen Datensätzen basieren. Allgemein liefert die stufenweise Instanzsegmentierung im Moment die besseren Ergebnisse. So sind die Ansätze, die auf dem Cityscapes Datensatz (Mask-RCNN [HGDG17]) und auf dem COCO Datensatz (R-FCN [LQD+16]) die besten Ergebnisse liefern, wie man aus der Tabelle 3.1 entnehmen kann, beide stufenweise Ansätze. Diese generelle Architektur bietet zudem den Vorteil weniger komplexer Modelle im Vergleich zu den pixelweisen Ansätzen. Diese Ansätze haben zwar den Nachteil, dass sie sich nicht von nicht vorhandenen Obkjektvorschlägen erholen können, aber, wie das Mask-RCNN zeigt, führt das gemeinsame Training der Instanzsegmentierung und der Objektdetektion zu einer Verbesserung der Objektdetektion. Wie zudem der Ansatz aus [HHS16] zeigt, gibt es Ansätze, die es ermöglichen, bei schlechten Objektvorschlägen die Segmentierung über die Bounding-Box-Grenzen hinweg zu erweitern.

Die Entscheidung gegen einen Ansatz der pixelweisen Instanzsegmentierung fiel auch deshalb, da es sich bei diesen meist um sehr komplexe Architekturen handelt, die meistens CRFs verwenden, um die Regression und Klassifikation der Instanzsegmentierung zu bewältigen. So kann man in Tabelle 3.1 sehen, dass die beiden Beispiele für die pixelweise Semantische Instanzsegmentierung, welche zur Zeit die besten Ansätze aus

Ansatz	Leistung	Verfügbarkeit	Urteil
Mask-RCNN [HGDG17] 3.1	Es handelt sich hierbei um einen zweistufigen Ansatz 2.4.2. Dieser liefert die besten Ergebnisse auf dem Cityscapes Datensatz: 32%. Die Evaluation auf Basis von Beispielbildern zeigt, dass auch bei schwierigen Fällen, wie überlappenden Instanzen der gleichen Klasse, gute Ergebnisse erzielt werden.	Keine offen zugängliche Caffe Implementierung, auf der aufgebaut werden kann.	Geringe Komplexität des Modells. Bietet Ansatzpunkte im Bereich des ROI-Pooling und des gemeinsamen Trainings.
Relative Position [LQD+16] 3.1	Eine zweistufige Architektur, welche auf einer pixelweisen Prädiktion von relativen Positionen basiert. liefert auf dem COCO Datensatz (Standard/2016 Challenge) mit einer AP von 37.3 % die besten Ergebnisse. Schnelle Inferenz 0.24 Sekunden auf COCO Datensatz.	Es gab zur Zeit der Entscheidung noch keine offen zugängliche Caffe-Implementierung.	Geringe Komplexität des Modells. Die pixelweise Vorhersage der relativen Positionen kann als Ansatzpunkt für die Trennung der Instanzen nach der Detektion dienen.
Bounding Box Augmentation [HHS16]	Eine Methode um die zweistufige Detektion robuster gegenüber fehlerhaften Detektionen zu machen. Geringe Komplexität. Liefert auf dem Cityscapes Datensatz mit einer AP von 17.4 % die viertbesten Ergebnisse.	Keine offen zugängliche Caffe Implementierung, auf der aufgebaut werden kann.	Der Ansatz ist von geringer Komplexität und kann auf fast jedes zweistufige Modell angewendet werden.
SemSeg+ Detekt+CRF [AT17]	Eine pixelweise Instanzsegmentierung, welche auf einer Detektion, einer Semantischen Segmentierung und einem CRF basiert. Sehr komplexes Modell, schwierig umzusetzen. Liefert die zweitbesten Ergebnisse auf dem Cityscapes Datensatz: AP 20.0 %	Keine offen zugängliche Caffe Implementierung, auf der aufgebaut werden kann.	Aufgrund der hohen Komplexität (vor allem wegen der CRFs) und dem fehlenden Code schwierig zu implementieren.
Instance Cut from Edges [KLA+16]	Ein pixelweiser Ansatz mit hoher Komplexität, welcher auf Basis von instanzspezifischen Kanteninformationen die Instanzsegmentierung in ein graphentheoretisches Problem umwandelt, welches durch CRFs gelöst wird. 5 Platz Cityscapes Datensatz: AP 13.0 %	Keine offen zugängliche Caffe Implementierung, auf der aufgebaut werden kann.	Sehr komplexer Ansatz, auf dem nur schwer aufgebaut werden kann, da kein Code vorhanden ist.

Tabelle 3.1: In dieser Tabelle wird ein Überblick über die in Kapitel 2.4 erläuterten Verfahren gegeben. Diese dient als Basis für den weiteren Vergleich und die Argumentation für unseren Ansatz

dieser Kategorie sind, beide auf Conditional Random Fields basieren, die zudem auf dem Ergebnis von Unteraufgaben wie der Detektion oder der Prädiktion von Instanzkanten basieren. Über den Ressourcenverbrauch der vorgestellten Methoden gibt es nur wenige verfügbare Informationen, man kann jedoch davon ausgehen, dass aus der höheren

Komplexität auch eine schlechtere Ausnutzung der Ressourcen folgt. Darüber hinaus ist die Code-Basis für die stufenweise Segmentierung wesentlich besser, da anders als bei den pixelweisen Architekturen, die Ansätze aus einer Detektion und einer nachfolgenden Segmentierung bestehen. Es existieren nämlich bereits eine Vielzahl an Detektionsarchitekturen mit Caffe-Implementierung, aus denen ausgewählt werden kann, sodass der Regressions-Teil der Instanzsegmentierung bereits vorhanden ist. Darüber hinaus ist es durch die recht große Auswahl an Detektionsansätzen möglich, durch die Auswahl des Ansatzes Parameter wie Schnelligkeit, Speicherbedarf etc. steuern zu können, was vor allem in Bezug auf das Selbstfahrende Auto wichtig ist. Auf die Auswahl des Detektors wird im Abschnitt 3.3 noch weiter eingegangen.

Es wurde also entschieden, einen auf einer Detektion basierenden zweistufigen Ansatz zu entwickeln, da dieser im Moment die besten Ergebnisse liefert und eine bessere Code Basis liefert. Im Folgenden wird nun näher darauf eingegangen, welche Architektur in dieser Arbeit entwickelt wurde.

3.2 Generelle Beschreibung des neuen Ansatzes

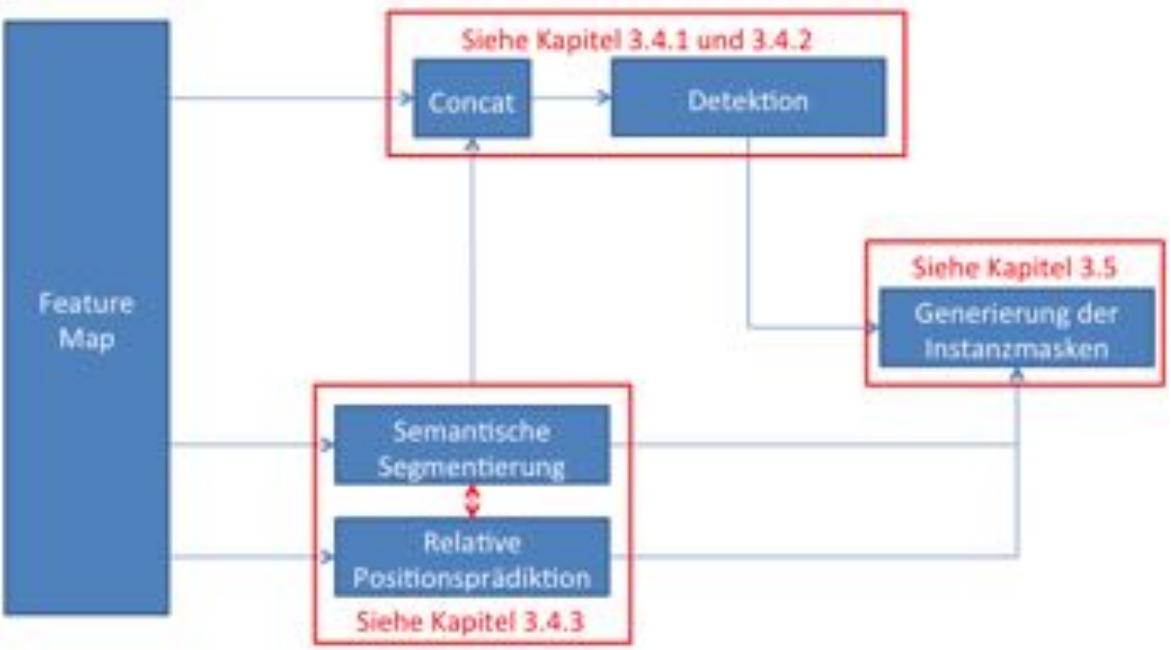

Abbildung 3.1: Die Abbildung illustriert die generelle Architektur, die in dieser Masterarbeit umgesetzt wurde. Dabei werden die Detektion und die Semantischen Segmentierung sowie eine Relative Positionsbestimmung auf derselben Feature-Map bestimmt. Auf Basis dieser Informationen kann dann eine Instanzsegmentierung bestimmt werden. Die einzelnen Elemente werden dabei in den in der Abbildung genannten Kapiteln näher betrachtet.

Wie man aus der Tabelle 3.1 sieht, ist es so, dass das Mask-RCNN die besten Ergebnisse auf Basis des Cityscapes Datensatz, also einen im Zusammenhang mit dem Autonomen

Fahren relevanten Datensatz in Bezug auf die semantische Instantzsegmentierung liefert. Dieser Erfolg basiert zu einem nicht geringen Teil auf der Tatsache, dass durch das gemeinsame Training der Objektsegmentierung und der Objektdetektion auf derselben Feature-Map beide Unteraufgaben profitieren ([HGDG17]). Die beiden Aufgaben sind eng verwandt, weshalb ähnliche Abstraktionen benötigt werden, um diese Aufgaben zu lösen. Diese Effekte des gemeinsamen Trainings sollen in dem Ansatz der Masterarbeit auch genutzt werden.

Den grundsätzlichen Gedanken für die Semantische Instanzsegmentierung, der in dieser Masterarbeit verfolgt wurde, kann man anhand von Abbildung 3.1 nachvollziehen. Auf die Implementierung der einzelnen Module wird jeweils in den angegebenen Kapiteln näher eingegangen. Der Ansatz besteht zunächst einmal darin, eine Objektdetektion zusammen mit einer Semantischen Segmentierung auf Basis derselben Feature-Map zu trainieren. In der Anwendung des Selbstfahrenden Autos ist es von großer Bedeutung, auch die Klasse von Pixeln zu kennen, die nicht zu einer Instanz gehören, was durch eine Semantische Segmentierung, wie in Kapitel 2.2 beschrieben, erreicht werden kann. Durch das Training der Semantischen Segmentierung und der Detektion auf Basis derselben Feature-Map sollen dann beide Aufgaben durch ihre enge Korrelation ähnlich wie beim Mask-RCNN [HGDG17] profitieren. Vor allem durch die Verbesserung der Detektion soll dadurch ein hauptsächlicher Nachteil der zweistufigen Ansätze, nämlich die Tatsache, dass sich diese nicht von fehlenden Detektionen erholen können, ausgeglichen werden. Im Kontext des Selbstfahrenden Autos betrachtet, bietet die gemeinsame Vorhersage von einer Semantischen Instanzsegmentierung und einer Semantischen Segmentierung innerhalb nur eines Modells zudem den Vorteil, dass weniger Ressourcen verbraucht werden (Geschwindigkeit, Speicher, Energie, ...).

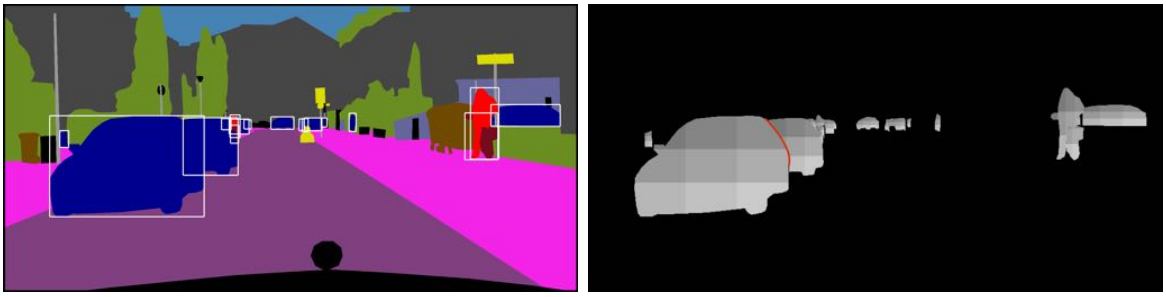

Abbildung 3.2: Die Daten für die Abbildung sind entnommen aus [COR+16]. Auf der linken Seite ist die Semantische Segmentierung zusammen mit der Detektion der Instanzen zu sehen. In den überlappenden Bereichen der Bounding-Boxen gleicher Klassen ist keine eindeutige Zuordnung der Segmentierung zu den Bounding-Boxen gegeben. Durch die relativen Positionen auf der rechten Seite ist dies wie zu sehen in den überlappenden Bereichen möglich. Es ergibt sich durch die relativen Positionen die in rot gekennzeichnete Kante zwischen den beiden Instanzen der Klasse Auto.

Durch die Objektdetektion wird der Regressionsteil der Instanzsegmentierung gelöst, indem für die im Bild existierenden Instanzen Bounding-Boxen generiert werden. Die Semantische Instanzsegmentierung wird dann dadurch generiert, dass für die prädizierte Klasse der Instanz der Bounding-Box die Semantische Segmentierung übernommen wird. Dieser Vorgang ist in Abbildung 3.2 zu sehen. Diese illustriert dabei auch das Problem

dieses Ansatzes. Durch diese Methode kann innerhalb des Bereiches der Bounding-Boxen nur zwischen Instanzen verschiedener Klassen unterschieden werden. So wäre es z.B. in Abbildung 3.2 möglich, innerhalb der Bounding-Box des Fahrradfahrers die Instanz der Klasse Fahrradfahrer von der Instanz der Klasse Fahrrad zu trennen, indem nur die Pixel der Semantischen Segmentierung mit der Klasse der Bounding-Box (also der Klasse Fahrradfahrer) übernommen werden. Wie man im Falle der beiden Autos im Vordergrund sieht, ergibt sich im Fall von sich überlappenden Instanzen der gleichen Klasse im Bereich der Überlappung der Bounding-Boxen jedoch keine eindeutige Zuordnung zu den Instanzen. Um dieses Problem anzugehen, wird neben der Semantischen Segmentierung durch das gleiche Netzwerk pixelweise eine Score-Map abgeleitet, die für jeden Pixel einer Instanz bestimmt, in welcher relativen Position er sich zu seiner Instanz befindet. Dieser Ansatz ist an die Architektur von [LQD+16] angelehnt, welche auf dem COCO Datensatz die besten Ergebnisse im Bereich der Instanzsegmentierung erzielen. Wie in Abbildung 3.2 zu sehen, ist es auf Basis dieser Positionsinformationen möglich, die Instanzen in überlappenden Bereichen voneinander zu trennen. In Kapitel 3.5 wird dann weiter darauf eingegangen, wie auf Basis der relativen Positionen die Trennung der Instanzen bestimmt wird. Hier wird auch gezeigt, wie die relativen Positionen der einzelnen Instanzen generiert werden. In Kapitel 3.4.3 wird erläutert, wie die relativen Positionen durch das Neuronale Netz pixelweise bestimmt werden können.

Die Semantische Segmentierung des Bildes und die relativen Positionsinformationen sind zudem Informationen, die auch für die Detektion von Bedeutung sind. Deshalb wurde in dieser Arbeit, wie in Abbildung 3.1 zu sehen, die Möglichkeit geprüft, der Detektion diese Informationen als weitere Eingabe zur Verfügung zu stellen (siehe Kapitel 3.4.1). Andererseits liefert auch die Detektion Ergebnisse, die für die Semantische Segmentierung und die Relative Positionsbestimmung von Bedeutung sind. Eine wechselseitige Abhängigkeit zu modellieren, würde jedoch ein rekurrentes Netzwerk voraussetzen, was eine höhere Komplexität aufweist. Deshalb wäre ein solches Netzwerk schwerer zu trainieren und würde gleichzeitig weniger schnelle Prädiktionen als ein nicht rekurrentes Netzwerk liefern. Die Abhängigkeit der Detektion von der Semantischen Segmentierung wurde der Abhängigkeit der Semantischen Segmentierung von der Detektion deshalb vorgezogen, da wie in Kapitel 2.3 näher erläutert, fehlende Objektvorschläge durch nachfolgende Schritte nicht mehr kompensiert werden können, sodass nur Objekte segmentiert werden, die auch eine Detektion haben.

Im nächsten Kapitel 3.3 wird nun darauf eingegangen, welcher der in 2.3 beschriebenen Ansätze für die Detektion ausgewählt wurde und welches Netzwerk für die Semantische Segmentierung verwendet wird. Nähere Einzelheiten in Bezug auf die genaue Implementierung der Architektur werden dann in 3.4 gegeben.

3.3 Begründung für die Auswahl der Detektion und Semantischen Segmentierung

In Kapitel 3.2 wurde der grundlegende Ablauf der Instanzsegmentierung, die in dieser Arbeit verwirklicht wurde, beschrieben. Dabei handelt es sich um einen Ansatz, der auf einer pixelweisen Semantischen Segmentierung und einem Detektor basiert. In diesem Kapitel soll nun näher auf die Umsetzung des Ansatzes eingegangen werden, indem erläutert wird, welcher Detektor und welche Architektur für die Semantische Segmentierung ausgewählt wurden. Da dieser Ansatz im Zusammenhang mit dem Selbstfahrenden Auto entwickelt wurde, sind neben der Qualität der Verfahren auch deren Ressourcenverbrauch wichtig. Ein weiterer wichtiger Punkt bei der Auswahl der Verfahren stellt das Vorhandensein einer benutzbaren Implementierung dar. Aus der gegeben Infrastruktur am DLR folgte zudem, dass diese nach Möglichkeiten auf dem Python Interface von Caffe basieren sollte.

Eine Übersicht über die bisher vorhandenen Architekturen zur Instanzdetektion bietet Kapitel 2.3. In Tabelle 3.2 werden diese Ansätze näher auf Basis ihrer Eigenschaften verglichen. Dabei zeigt sich zunächst einmal, dass die Basis an Implementierungen, die benutzt werden können, recht klein ist. Nur das “Faster-RCNN“ [Gir15] und “Recurr“ [RCL+17] bieten eine Caffe- Implementierung unter dem Python Interface. Dabei bietet das “Recurr“ Netzwerk zwar eine sehr gute Detektionsleistung auf Basis des KITTI-Datensatzes, es ist nämlich die beste Architektur mit Paper. Mit einer Ausführungszeit von 3.6 Sekunden pro Bild ist dieses jedoch zu langsam für unseren Anwendungsfall. Eine weitere Beobachtung, die man auf Basis der Tabelle machen kann, ist, dass die Detektoren “YOLO-V2“ [RF16] und “Squeeze“ [WIJK16], welche auf einer Architektur basieren, die innerhalb eines Schrittes die Klasse und die Bounding-Box prädiziert, zwar sehr wenig Ressourcen in Bezug auf Geschwindigkeit Speicher und Energie verbrauchen, dass diese jedoch eine vergleichsweise geringe Detektionsqualität bieten. Eine gute Detektionsgenauigkeit ist jedoch sehr wichtig für meinen Ansatz der Instanzsegmentierung, da sich dieser nicht von fehlenden Objektdetektionen erholen kann (siehe Kapitel 3.2). In Bezug auf die Objektdetektion bieten die Ansätze “Faster-RCNN“, “Mask-RCNN“, “R-FCN“ und “Fully Conv Inst-Aware“ die besten Ergebnisse. Diese basieren auf einer Architektur, die zuerst Bounding-Box Vorschläge generiert, welche darauf folgend zu den Klassen zugeordnet und verfeinert werden. Wie in Kapitel 2.3 erklärt, bildet das “Faster-RCNN“ den Detektor, auf dem die Instanzsegmentierung des “Mask-RCNN“ aufbaut, und das “R-FCN“ den Basisdetektor der “Fully Conv Inst-Aware“ Instanzsegmentierung. In beiden Fällen scheint das gemeinsame Training einen positiven Einfluss zu haben. Die Detektionsgenauigkeit und Geschwindigkeit der Architekturen liegen dabei ungefähr im selben Bereich. Da zur Zeit der Entscheidung über den Detektor nur für das “Faster-RCNN“ eine verwendbare Implementierung (siehe Tabelle 3.2) vorlag, wurde beschlossen, diesen Detektor zu verwenden und durch das gemeinsame Training mit der Semantischen Segmentierung zu verbessern. Dabei wurde auf die Implementierung aus “https://github.com/rbgirshick/py-faster-rcnn“ (abgerufen am 15.06.2017) zurückgegriffen, welche auf dem VGG-16 [SZ14] als Basisnetzwerk zur Generierung der Feature-Map aufbaut.

Ansatz	Leistung	Code	Ressourcen
Faster-RCNN [Gir15]	Ansatz basiert auf Generierung von Region Proposals und nachfolgender Klassifikation und Regression. Erreicht zur Zeit Platz 6 auf dem COCO Leader Board (Standard) AP: 37.1 % (das beste Modell des Faster RCNN).	Ja	GPU: Tesla K40; Auflösung:Tesla K40; FPS: 5;
Mask-RCNN [HGDG17]	Ansatz basiert auf dem Faster RCNN und profitiert vom gemeinsamen Training der Detektion mit der Semantischen Instanzsegmentierung und einer neuen ROI-Pooling Methode. Auf Basis des selben Backbones wird eine Steigerung von einer AP:36.8 % auf eine AP: 39.8 % erreicht (auf dem COCO test-dev Set; Laut Autoren auf diesen Testdaten der beste verfügbare Detektor).	Nein	GPU: Nvidia Tesla M40; FPS: ≈ 5)
R-FCN [DLHS16]	Basiert auf der pixelweisen Vorhersage von relativen Positionen im Verhältnis zur Instanz, welche in den Regionenvorschlägen eines RPN betrachtet werden. Erreicht auf dem Pascal-VOC+Additional Data mit einer AP von 85% die besten Ergebnisse einer Basis-Variante eines Detektors.	Ja aber im Matlab Caffe Interface	GPU: Nvidia K40; FPS: ≈ 6
Fully Conv Inst-Aware [LQD+16]	Basiert auf dem R-FCN. Zusätzlich werden aus den pixelweisen relativen Positionen innerhalb der Bounding-Box Vorschläge für jede Klasse Instanzmasken prädiziert. Erreicht auf dem COCO Standard-Test Set mit einer AP von 0.394 % die besten Ergebnisse einer Basis Variante eines Detektors (noch kein Vergleich mit Mask-RCNN). Bessere Leistung als Erweiterungen des Faster-RCNN.	Zur Zeit der Entscheidung nicht	GPU: Nvidia K40; FPS: ≈ 4 (COCO Datensatz)
Recurr [RCL+17]	In einem Schritt werden sowohl Bounding-Boxen als auch Klassen bestimmt. Basiert auf einer rekurrenten Feature-Map. Erreicht auf dem KITTI-Datensatz eine ("Moderate" Evaluation) mAP von Car (Platz 7): 90.22%; Pedestrian (Platz 3): 75.33 %; Cyclist (Platz 3): 76.47% bester Ansatz mit Paper.	Ja	GPU: 2.5 Ghz (Python + C/C++); Pro Frame:3.6 s
YOLO V2 [RF16]	In einem Schritt werden sowohl Bounding-Boxen als auch Klassen bestimmt. Aufteilung des Bildes in ein Gitter. Auf Basis der Gitterelemente wird Bounding-Box und Klasse vorhergesagt. Erreicht auf dem Pascal-VOC+Additional Data Test Set eine mean AP von 78.2 % (Platz 21), was 6.8 % weniger sind als das R-FCN.	Nicht für Caffe	GPU: Geforce GTX Titan X; Auflösung: 544×544 (Pascal 2007); FPS: 40
Squeeze [WIJK16]	In einem Schritt werden sowohl Bounding-Boxen als auch Klassen bestimmt. Grundarchitektur basiert auf YOLO. Sehr ressourcensparendes Netzwerk. Auf dem Kitti Datensatz ("Moderate" Evaluation) AP von Car (Platz 29): 88.83%; Pedestrian (Platz 7): 73.62 %; Cyclist (Platz 7): 74.45%.	Nicht für Caffe	GPU:NVIDIA TITAN X; Auflösung: 1242×375; FPS: $31 - 57$; Speicher: 65 bis 252 MB: Energie: 0.84-4.0 Joule

Tabelle 3.2: In dieser Tabelle wird ein Überblick über die in Kapitel 2.3 erläuterten Verfahren gegeben. Diese dient als Basis für den weiteren Vergleich und die Argumentation für die Auswahl des Detektors in diesem Ansatz. Allgemein werden nur die Basisvarianten der Detektoren betrachtet (aufbauende Varianten meist zu komplex).

Bei der Architektur, die in dieser Arbeit für die Semantische Segmentierung und die Prädiktion der relativen Positionen verwendet wurde, handelt es sich um das "FCN-8" aus [SLD16], welches in Kapitel 2.2 näher erklärt ist. Dabei ergänzt sich diese gut mit dem "Faster-RCNN", da beide auf dem VGG-16 als Backbone aufbauen. Die generelle Architektur des "FCN-8" (siehe Kapitel 2.2) bildet im Bereich der Semantischen Segmentierung die Basis, auf der die besten Architekturen aufbauen. Die Erweiterungen haben jedoch meistens einen höheren Ressourcenverbrauch und eine höhere Komplexität. Zudem bietet das "FCN-8" mit einer IOU von 65.3 auf dem Cityscapes-Datensatz bereits eine ausreichend gute Leistung. Die Implementierung, die als Basis für die Semantische Segmentierung diente, ist unter "https://github.com/shelhamer/fcn.berkeleyvision.org" (abgerufen am 15.06.2017) zu finden.

3.4 Beschreibung der Architekturen

Nun, da sowohl die Verwendung des Detektionsansatzes und des Ansatzes für die Semantische Segmentierung geklärt ist, soll näher auf die in dieser Masterarbeit umgesetzte Architektur eingegangen werden. In 3.4.1 wird dabei zunächst erklärt, wie das gemeinsame Training des Faster-RCNN und des FCN-8 auf der gleichen Feature-Map umgesetzt wurde. In Kapitel 3.4.2 wird dann auf eine neu erdachte ROI-Pooling Methode eingegangen und in Kapitel 3.4.3 wird erläutert, wie die Prädiktion der Relativen Positionen umgesetzt wurde und wie diese sich mit der Semantischen Segmentierung ergänzen kann. Allgemein kann anhand von Abbildung 3.1 nachvollzogen werden, an welcher Stelle die in diesen Abschnitten erklärten Architekturen in der Gesamtarchitektur zu finden sind.

3.4.1 Architekturen für das gemeinsame Training

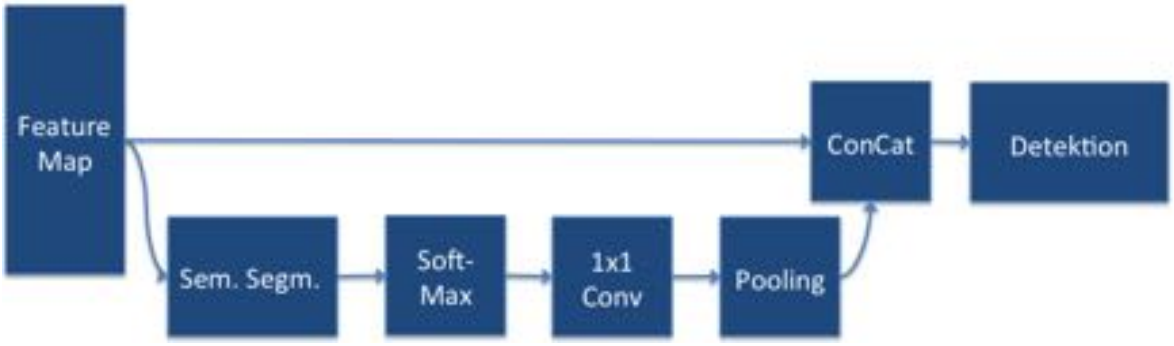

Abbildung 3.3: Die Abbildung illustriert, wie die durch das FCN-8 erzeugte Ausgabe verwendet wird, um die Detektion zu verbessern. Diese wird durch eine 1×1 Faltung und ein Pooling auf die Größe der Eingabe der Detektion gebracht, sodass diese konkateniert werden können.

Das gemeinsame Training kann so angegangen werden, dass zwei Aufgaben mit zwei Loss-Funktionen auf einer gemeinsamen Feature-Map (in diesem Fall dem VGG-16)

trainiert werden. Wenn man davon ausgeht, dass diese Aufgaben wie in unserem Fall sehr eng korreliert sind, so kann man davon ausgehen, dass beide Aufgaben auf einer ähnlichen Feature-Map gelöst werden können und sich gegebenenfalls beim Training positiv beeinflussen. Solche positiven Wechselwirkungen können z.B. daraus entstehen, dass Features durch die jeweils andere Loss-Funktion entstehen, die nicht durch die Loss-Funktion der eigentlichen Aufgabe hätten entstehen können, die sich jedoch positiv auswirken. Im übertragenden Sinne kann man das vielleicht als eine andere Perspektive auf das eigentliche Problem betrachten. Wenn man zudem davon ausgeht, dass Features generiert werden, welche für beide Aufgaben von Vorteil sind, so handelt es sich bei diesen Features also um generellere Features (oder Sichtweisen), die dazu führen sollten, dass das Netzwerk besser generalisiert, also bessere Ergebnisse auf unbekannten Bilder liefert.

Bei den im vorherigen Abschnitt erläuterten Folgen des Trainings auf einer gemeinsamen Feature-Map muss man jedoch auch beachten, dass es sich bei der pixelweisen Prädiktion von semantischen Informationen (Semantische Segmentierung/ Relative Positionsvorhersage) und der Detektion, auch wenn diese viele Eigenschaften teilen, um unterschiedliche Aufgaben handelt, deren Loss-Funktionen unterschiedliche Optima haben. Darüber hinaus kann es sein, dass der Loss einer Aufgabe, bezogen auf das Gesamtsystem, ein stärkeres Gewicht hat, oder schneller konvergiert und somit die Feature-Map hauptsächlich für eine Aufgabe optimiert wird. Eine Gewichtung der Loss-Funktion ist sehr schwierig, da diese empirisch bestimmt werden müsste, da bisher keine logische Grundlage gefunden wurde, diese zu bestimmen (siehe Vortrag: http://on-demand.gputechconf.com/gtc/2017/presentation/s7347-joechen-a-deep-hierarchial-model-for-joint-object-detection.pdf , entnommen am 24. 7. 2017).

Um auf diese Schwierigkeiten des Trainings auf einer gemeinsamen Feature-Map einzugehen, wurde die in Abbildung 3.3 zu sehende Architektur konstruiert. Hierbei werden die durch das FCN-8 bestimmten semantischen Informationen als weitere Eingabe für das Faster-RCNN, also die Detektion, zur Verfügung gestellt. Dies geschieht dadurch, dass, wie in der Abbildung zu sehen, die pixelweisen Scores der semantischen Informationen zunächst durch eine Soft-Max Funktion (siehe Kapitel 2.1.4) auf pixelweise Pseudo-Wahrscheinlichkeiten normiert werden. Eine 1×1 Faltung ermöglicht eine weitere lernbare Anpassung der Scores für die Aufgabe der Detektion und führt ein Padding (um 100) aus, das nötig ist, damit nach dem Pooling die Scores und die Feature-Map konkateniert werden können. Die sich ergebende Tiefe der Ausgabe ist dann wiederum identisch mit der der Eingabe der Faltung also gleich der Anzahl der Klassen. Das Pooling dient dazu, die pixelweisen Scores, welche dieselbe Auflösung wie das Eingabebild haben, auf die Größe der Feature-Map zu bringen. Es handelt sich hierbei also um eine Skalierung auf 1/16 der Größe des Eingabebildes, weshalb auch ein 16×16 Average-Pooling mit einer Schrittweite von 16 angewendet wird. Durch die Konkatenation mit der Feature-Map ergibt sich eine Detektionsgrundlage, die sowohl für das RPN als auch für das RCNN des Faster-RCNN zu Verfügung gestellt werden kann.

Diese Abhängigkeit der Detektion von den semantischen Ausgaben des FCN-8 führen dazu, dass die Detektion bereits pixelweise semantische Informationen über das Bild hat, sodass die Klassifikation und Regression der Bounding-Boxen davon profitieren. Daraus folgt zudem, dass eine möglichst optimale pixelweise Klassifikation des FCN-8 zu einer guten Informationsgrundlage für das Faster-RCNN also die Detektion führt, sodass sich die beiden Optima der Loss-Funktionen nicht mehr widersprechen. Darüber hinaus werden nun die Gewichtungen durch den Back-Propagation Algorithmus so angepasst, dass auch der Einfluss des Pfades von der Semantischen Segmentierung zur Detektion mit in die Berechnung der Gradienten einfließt.

Die Abhängigkeit der Detektion von der pixelweisen semantischen Ausgabe des FCN-8 wurde der umgekehrten Abhängigkeit des FCN-8 von der Detektion deshalb vorgezogen, da die Detektion in der Gesamtarchitektur für die semantische Instanzsegmentierung eine wichtigere Rolle übernimmt. Wie bereits in Kapitel 3.2 ausgeführt, können nämlich nur Instanzen segmentiert werden, die auch eine Detektion haben. Ideal wäre eine wechselseitige direkte Abhängigkeit des Faster-RCNN und des FCN-8, indem die Ausgabe der einen Aufgabe jeweils mit als Eingabe der anderen dienen würde. Hierbei ergibt sich jedoch das Problem, dass eine derartige Architektur rekurrent wäre. Das heißt zum einen, dass diese einen erheblich höheren Aufwand in Bezug auf die Implementierung und das Training bedeuten würde, was in Konflikt mit den in der Masterarbeit gesetzten Prioritäten treten würde. Zum anderen würde eine solche Rekurrenz zu einem höheren Ressourcenverbrauch in Bezug auf Geschwindigkeit und Speicherverbrauch führen, was im Kontext des Selbstfahrenden Autos einen erheblichen Nachteil darstellt.

3.4.2 Die neue ROI Pooling Methode

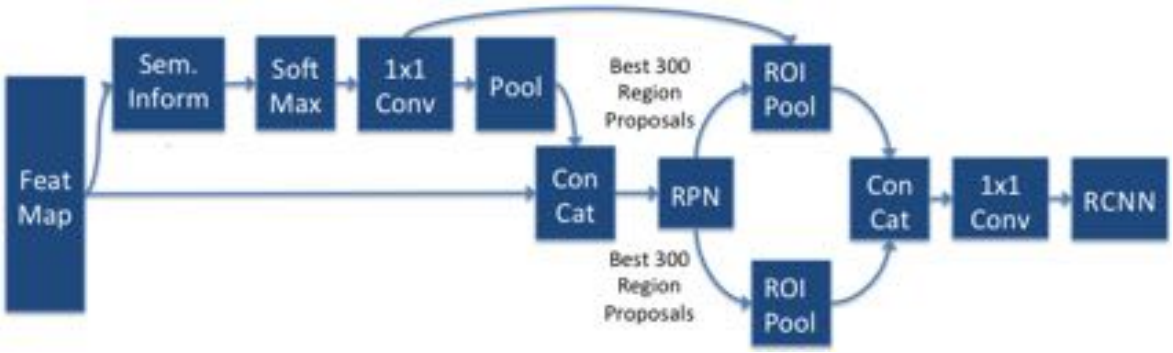

Abbildung 3.4: Die Abbildung illustriert die neue Methode des ROI-Poolings. Dabei werden die 300 durch das RPN generierte Bounding-Boxen mit der höchsten Objekt-Konfidenz verwendet, um die semantischen Informationen in der Auflösung des Originalbildes und die gering aufgelöste Feature-Map auf eine konstante Feature-Map abzubilden. Diese dient dann als Eingabe für das RCNN.

Das Faster-RCNN, wie es in Kapitel 2.3.1 beschrieben wurde, sieht eine zweistufige Architektur vor, bei der zunächst durch das Region Proposal Network (RPN) Bounding-Box-Vorschläge generiert werden. Die 300 Bounding-Boxen mit den höchsten Konfiden-

zen (Scores) dienen dann dem RCNN als Eingabe. Innerhalb dieser Bounding-Boxen wird dafür zunächst das ROI-Pooling bestimmt, bei dem die Feature-Map innerhalb der Bounding-Box-Vorschläge auf ein immer gleich großes $k \times k$ Gitter abgebildet wird. Diese Feature-Map hat hierbei im Verhältnis zum Eingabebild eine stark heruntersskalierte räumliche Ausdehnung (1/16 der Größe des Eingabebildes), was dazu führt, dass wenig räumliche Details repräsentiert werden können. Eben solche Details sind jedoch für die Regression und die Klassifikation der Bounding-Box-Vorschläge des RPN wichtig.

Um dieses Problem zu lösen, wurde im Zuge dieser Masterarbeit eine neue Pooling Methode erdacht. Diese Architektur ist in Abbildung 3.4 dargestellt. Die neue ROI-Pooling Methode wird neben dem herkömmlichen ROI-Pooling ausgeführt. Wie in der Abbildung zu sehen, dienen dabei die pixelweisen semantischen Informationen, welche dieselbe räumliche Auflösung wie das Originalbild haben, als Eingabe für ein weiteres ROI-Pooling. Diese werden also ebenfalls auf das konstante $k \times k$ Gitter abgebildet. Dies kann ebenfalls als eine räumliche Skalierung der Semantischen Bildinformation gesehen werden, welche jedoch geringer ist als bei der Ausführung des gleichen ROI-Poolings auf der geringer aufgelösten Feature-Map. Die Ergebnisse des ROI-Poolings auf der gering aufgelösten Feature-Map und auf den hoch aufgelösten semantischen Informationen werden dann konkateniert, sodass die Tiefe der Ausgabe der Konkatenation $AnzahlDerKlassen + TiefeDerFeatureMap$ entspricht. Diese dient dann als Eingabe für die erste Fully-Connected Schicht des Faster-RCNN, deren Form (Anzahl der Gewichtungen) von der Form der Eingabe abhängig ist. Durch die Konkatenation ist die Tiefe der Eingabe jedoch größer als beim Faster-RCNN. Beim Training soll jedoch ein Modell des Faster-RCNN zur Initialisierung der Gewichtungen verwendet werden, dessen Form der Fully-Connected Schicht auf der Form der Feature-Map nach dem ROI-Pooling basiert. Diese besitzt eine geringere Tiefe ($TiefeDerFeatureMap$) als die Konkatenation der Ergebnisse beider ROI-Poolings ($AnzahlDerKlassen + TiefeDerFeatureMap$). Deshalb muss die Tiefe der Eingabe der Fully-Connected Schicht angepasst werden. Hierfür wird eine 1×1 Faltung verwendet, deren Ausgabe die Tiefe der Feature-Map hat.

Durch die zusätzlichen pixelweisen semantischen Informationen in der Auflösung des Eingabebildes erhält das RCNN also eine Daten-Grundlage, auf der die Klassifikation kleiner Objekte und eine genauere Regression der Bounding-Boxen möglich sind. Zudem bieten die Semantische Segmentierung und die Relative Positionsvorhersage bereits vor der Klassifikation/Regression durch das RCNN semantische Informationen über die Bounding-Box. Die Konkatenation kann auch als eine Verbindung von eher lokalen semantischen Informationen der Ausgabe des FCN-8 und eher globalen Informationen der gering aufgelösten Feature-Map betrachtet werden.

3.4.3 Semantischer Segmentierung und Relative Positionsbestimmung

Die Abbildung 3.5 illustriert, wie mithilfe der Architektur des FCN-8 neben der Semantischen Segmentierung auch die Relative Positionsbestimmung (also die Prädiktion der

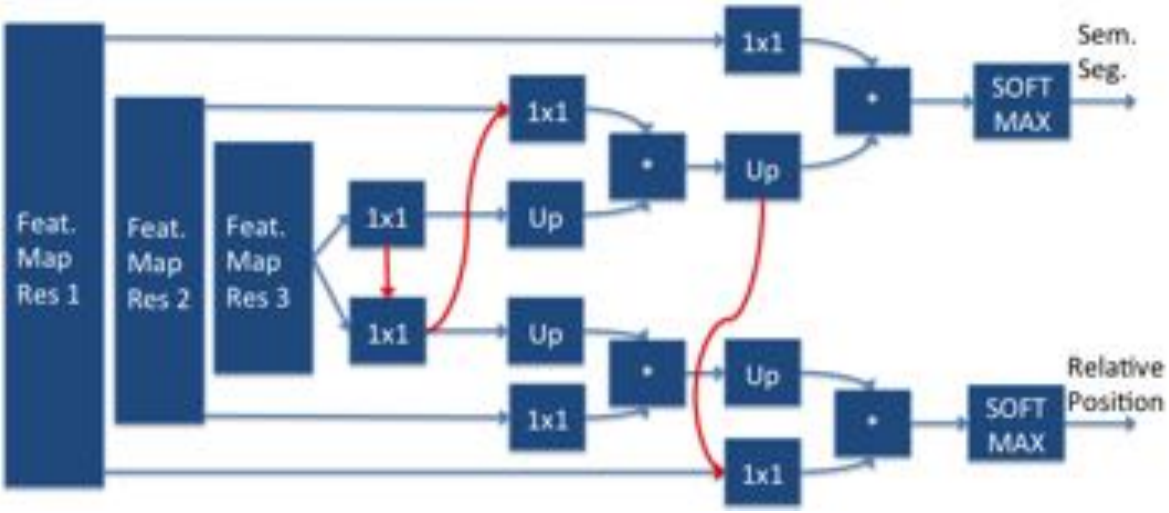

Abbildung 3.5: Die Abbildung illustriert, wie die semantischen Informationen, also die relative Positionsbestimmung und die Semantische Segmentierung durch das FCN-8 bestimmt werden und voneinander abhängig gemacht werden. Dabei symbolisieren die roten Pfade einen Fluss von semantischen Informationen. Diese werden jeweils dem anderen Zweig zur Verfügung gestellt, bevor dieser auf einer der Feature-Maps die Scores bestimmt. Mit "Up" wird das Upsampling aus Kapitel 2.2.1 gekennzeichnet.

relativen Positionen) generiert wird. Beide folgen dabei der Struktur, wie sie in Kapitel 2.2 beschrieben wurde, auf Basis von hoch und niedrig aufgelösten Feature-Maps die pixelweisen Klassifikationen zu bestimmen, um lokale detailreiche Entscheidungen in Bezug auf globale Strukturen zu treffen. Die Scores für die einzelnen Klassen werden dabei jeweils durch die 1×1 Faltungen auf Basis der verschiedenen Feature-Maps bestimmt.

Bestimmung der Relativen Positionsprädiktion

Die Prädiktion der relativen Positionen kann dabei ebenso als Klassifikationsproblem betrachtet werden. Wie in Kapitel 3.2 eingeführt und in Kapitel 3.5 weiter ausgeführt, werden die relativen Positionen nämlich in eine bestimmte Menge an Klassen (in diesem Fall 9 Klassen) unterteilt, die jedem Pixel einer Instanz zugeordnet werden. Dabei wird die gleiche pixelweise Klassifikations Loss-Funktion aus Kapitel 2.1.4 verwendet, die auch bei der Semantischen Segmentierung angewendet wurde. Zusätzlich zu den (9) Klassen der relativen Positionen sollen die Pixel, die nicht zu einer Instanz gehören, als Hintergrund klassifiziert werden. Dabei gibt es zwei Arten, dieses Problem zu lösen. Zum einen kann man das Hintergrundlabel als weitere Klasse im Zusammenhang mit der Relativen Positionsprädiktion mitbestimmen. Zum anderen ist es so, dass durch die Semantische Segmentierung bereits eine pixelweise Information über den Hintergrund im Zusammenhang mit der Relativen Positionsprädiktion gegeben ist. Alle Pixel, denen in der Semantischen Segmentierung eine Klasse zugewiesen wurde, die keine Instanzen enthält (z.B. Klassen wie Straße oder Vegetation), gelten im Zusammenhang mit der Relativen Positionsprädiktion nämlich als Hintergrund. Eine weitere Möglichkeit,

das Hintergrundlabel zu bestimmen, ist also, während des Trainings der Relativen Positionsprädiktion die Loss-Funktion nur für die Pixel zu bestimmen, denen eine relative Position zugewiesen ist, und die Hintergrundpixel zu ignorieren. Bei der Inferenz der relativen Positionen wird dann für jedes Pixel ein Klassifikationsproblem zwischen den (9) relativen Positionen gelöst. Das Hintergrundlabel wird dann wie beschrieben für alle Pixel angenommen, denen bei der Semantischen Segmentierung eine Klasse zugewiesen wurde, die keine Instanzen enthält.

Abhängigkeiten zwischen Relativer Positionsbestimmung und Semantischer Segmentierung

Diese beiden Aufgaben der Relativen Positionsprädiktion und der Semantischen Segmentierung generieren jeweils wiederum pixelweise semantische Informationen. Die Relative Positionsbestimmung generiert die relativen Positionen in einer Instanz in Bezug zum Betrachter auf Basis eines $k \times k$ Gitters (wird in Kapitel 3.5 näher ausgeführt) und die Semantische Segmentierung bestimmt die Klassenzugehörigkeit jedes Pixels. Diese Informationen hängen voneinander ab. Dabei liefert die Semantische Segmentierung semantische Informationen, um zwischen Objekten verschiedener Klassen zu unterscheiden, und die Relativen Positionsprädiktion, um zwischen verschiedenen Objekten derselben Klasse zu unterscheiden. Diese beiden Sichtweisen sind für die jeweils andere Aufgabe wichtig. Die Unterscheidung zwischen Objekten verschiedener Klassen erleichtert die Bestimmung der relativen Positionen in Grenzregionen von Objekten unterschiedlicher Klassen. Eine genauere Unterscheidung zwischen den Instanzen einer Klasse kann zu einer besseren Segmentierung der Formen der einzelnen Instanzen bei der Semantischen Segmentierung führen. Deshalb ist es sinnvoll, auch im Netzwerk Abhängigkeiten zwischen diesen Aufgaben zu schaffen. Solche Abhängigkeiten sind in der Abbildung durch die roten Pfeile gekennzeichnet. Diese sind dabei so konstruiert, dass die semantischen Informationen des anderen Zweiges jeweils als weitere Datengrundlage neben der Feature-Map für die Bestimmung der Scores durch die 1×1 Faltungen dient. Dabei werden vor der 1×1 Faltung die Feature-Map und die pixelweise Score-Map konkateniert. Da die Bestimmung der Relativen Positionen für die Trennung der Instanzen in der Instanzsegmentierung von großer Bedeutung ist und Rekurrenzen vermieden werden sollten, fließen die semantischen Informationen des Segmentierungszweiges zweimal für die Relative Positionsbestimmung ein und die Scores der Relativen Positionsbestimmung nur einmal in den Zweig der Semantischen Segmentierung.

3.5 Trennung der Instanzen

Wie in Kapitel 3.2 bereits eingeführt, wird die Instanzsegmentierung bestimmt, indem innerhalb der prädizierten Bounding-Boxen die Semantische Segmentierung und die Prädiktion der Relativen Positionen betrachtet werden. Dabei dient die Semantische Segmentierung als Quelle dafür, zu bestimmen, in welchem Bereich in der Bounding-Box sich das Objekt der durch die Detektion bestimmten Klasse befindet. Wie in Abbildung

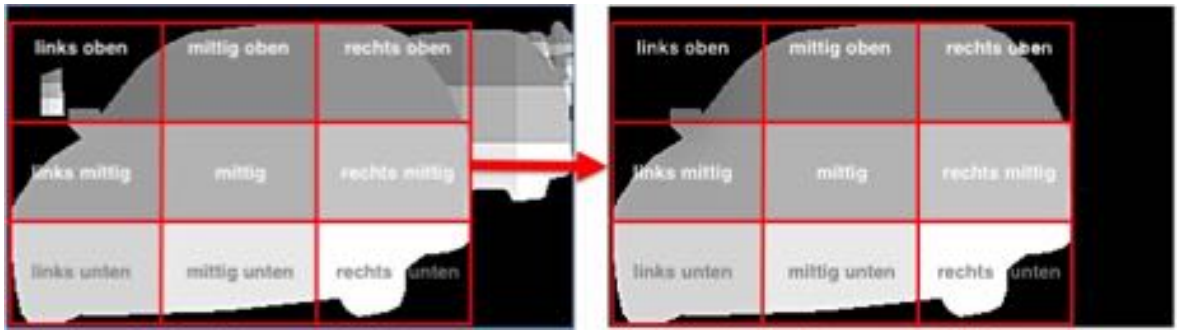

Abbildung 3.6: Die Daten für die Abbildung stammen aus [COR+16]. Hier ist die Erstellung der pixelweisen relativen Positionen anhand des Autos im Vordergrund zu sehen. Es wird ein 3×3 Gitter innerhalb der Bounding-Box der Instanz definiert. Jedes Gitterelement hat dabei ein bestimmtes Label. Dieses Label wird allen Pixeln der zur Bounding-Box korrespondierenden Instanz, die sich im Gitterelement befinden, zugewiesen (siehe linken Teil). Auf der rechten Seite kann man sehen, dass die vordere Autoinstanz von der hinteren getrennt werden kann, wenn nur die Pixel der Klasse Auto übernommen werden, die das relative Positionslabel des Gitterelementes haben, in dem sie sich befinden.

3.2 zu sehen, führt dies jedoch bei sich überlappenden Instanzen derselben Klasse dazu, dass Teile der anderen Instanz mit in die Instanzsegmentierung des zu der Bounding-Box korrespondierenden Objektes aufgenommen werden. Um in solchen Fällen die Instanzen voneinander trennen zu können, wird eine relative Positionsinformation wie in Abbildung 3.6 prädiziert. Wie man hier sieht, sind die Relativen Positionen durch ein gleichmäßiges $k \times k$ großes Gitter definiert, das innerhalb der Bounding-Box der Instanz bestimmt ist. In dieser Arbeit wurde dabei ein 3×3 Gitter verwendet, wie es auch in Abbildung 3.6 zu sehen ist. Jedem Pixel der zu der Bounding-Box korrespondierenden Instanz wird dann das Label des Gitterelementes zugewiesen, in dem er sich befindet. In Abbildung 3.6 ist auf der linken Seite ein Beispiel für diese Zuweisung anhand des Autos im Vordergrund zu sehen. Jedes Gitterelement definiert, wie zu sehen, ein Label, das zu einer relativen Position korrespondiert: “links oben“, “mittig oben“, “rechts-oben“,..., “rechts unten“. Diese Labels können dann durch das FCN-8 auf die gleiche Weise wie die Semantische Segmentierung prädiziert werden. Dabei wird für diese Aufgabe eine weitere Loss-Funktion definiert. Bei dieser handelt es sich wie bei der Semantischen Segmentierung um die pixelweise Klassifikations-Loss-Funktion aus Kapitel 2.1.4.

Wenn nun eine Bounding-Box prädiziert wird, so kann diese wiederum durch dasselbe 3×3 große Gitter aufgeteilt werden. Nun wird für jedes Pixel der prädizierten Klasse davon ausgegangen, dass er zu der Bounding-Box gehört, wenn für ihn das relative Positionslabel prädiziert wurde, welches zum Gitterelement korrespondiert, in dem sich das Pixel befindet. Dies führt, wie in Abbildung 3.6 zu sehen, dazu, dass sich die Instanzen in den meisten Fällen trennen lassen. Je nach der Dichte des Gitters gibt es auch Fälle, in denen die Trennung auf Basis der relativen Positionen nicht möglich ist, wie in Abbildung 3.7 auf der rechten Seite zu sehen. Bei diesen Fällen handelt es sich jedoch meistens um ohnehin sehr komplexe Situationen, die häufig eine geringe Relevanz

haben. Gegebenenfalls muss dann das Gitter so fein gewählt werden, dass alle relevanten Fälle richtig voneinander getrennt werden können. Allgemein ist jedoch bereits eine grobe Abgrenzung der Instanzen durch die vorhergesagte Bounding-Box gegeben, was im Kontext des Automatisierten Fahrens häufig ausreichend ist.

3.5.1 Kriterium zur Trennung der Instanzen

Man kann davon ausgehen, dass die Bounding-Boxen und die prädizierten relativen Positionen nicht immer gänzlich übereinstimmen, sodass vor allem in Regionen zwischen Labelgrenzen im Gitter der prädizierten Bounding-Box bei einer strikten Einhaltung des Gitters Teile der Instanz nicht übernommen würden. Ein Beispiel hierfür ist z.B. in Abbildung 3.7 auf der linken Seite zu sehen. Hier kann man auch sehen, dass generell die Prädiktion der relativen Positionen in Bereichen, wo Label aneinander grenzen unsicher ist.

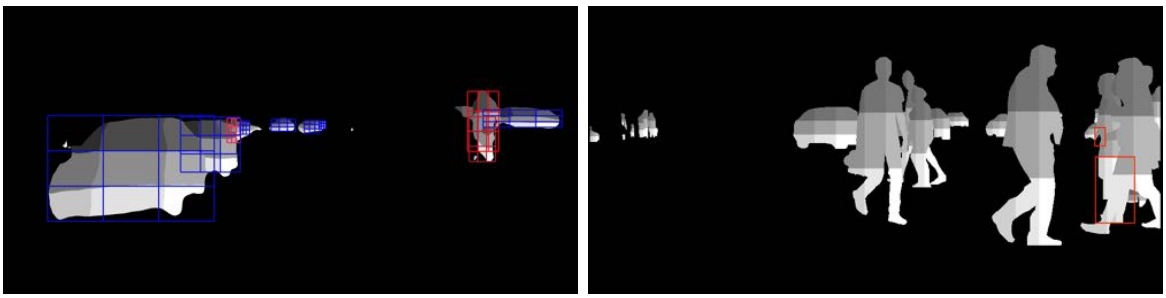

Abbildung 3.7: Die hier dargestellten Daten basieren auf [COR+16]. Auf der linken Seite ist ein Beispiel für die prädizierten pixelweisen Positionen und Bounding-Boxen zu sehen. Dabei zeigt sich, dass die relativen Positionen des durch die Bounding-Box bestimmten Gitters nicht immer mit den pixelweisen Relativen Positionen übereinstimmen. Vor allem an den Rändern der Gitterelemente gibt es Probleme. Auf der rechten Seite sind in rot umrandet Bereiche zu sehen, in denen durch die auf einem 3×3 Gitter basierenden Relativen Positionen die Instanzen nicht voneinander getrennt werden können.

Es ist nun also wichtig, ein Kriterium zur Trennung der Instanzen auf Basis der Semantischen Segmentierung, der relativen Positionen und der Bounding-Boxen zu finden. Ein erster Ansatz ist es, zunächst einmal (wie bereits beschrieben) nur Pixel in der Bounding-Box zu betrachten, für die durch die Semantische Segmentierung dieselbe Klasse bestimmt wurde, welche durch die Bounding-Box vorhergesagt wurde. Um nun innerhalb der Bounding-Box die Instanzen der Klasse zu trennen, wird die Euklidische Distanz zwischen der pro Pixel prädizierten relativen Position und der relativen Position innerhalb der Bounding-Box bestimmt (dem Gitterelement). Mit dieser Distanz ist dabei die Distanz innerhalb des Gitters gemeint. So hat z.B. das Gitterelement "links oben" eine Euklidische Distanz von $\sqrt{2} = \sqrt{1^2 + 1^2}$ zu dem Gitterelement in der Mitte (bezogen auf ein 3×3 Gitter), da diese beiden Elemente um den Vektor $[1, 1]$ in Bezug auf das Gitter translatiert sind. Man kann auf Basis dieser Distanz nun davon ausgehen, dass ein Pixel zu der Bounding-Box gehört, wenn die Euklidische

Distanz zwischen dem Gitterelement, in dem es sich befindet, und seiner prädizierten relativen Position $\leq\sqrt{2}$ ist. Eine Distanz $\leq\sqrt{2}$ deutet nämlich darauf hin, dass die relative Position in der Bounding-Box und die prädizierte relative Position entweder übereinstimmen oder direkte Nachbarn sind, was also eine gewisse Fehlausrichtung zwischen prädizierten pixelweisen relativen Positionen und den Gitterelementen der Bounding-Boxen erlaubt. Ist die euklidische Distanz $>\sqrt{2}$, so kann man davon ausgehen, dass es sich hierbei nicht um einen Teil der Instanz handelt, die durch die Bounding-Box beschrieben wird, da eine solche Distanz darauf hinweist, dass das aktuelle Pixel sich relativ zu seiner Instanz mindestens in einer Dimension auf der anderen Seite der Instanz befindet (im Falle eines 3×3 Gitters), als es die Bounding-Box der aktuellen Instanz prädiziert.

Auf Basis der Informationen der Semantischen Segmentierung, der Relativen Positionsprädiktion und der Detektion können auch andere Kriterien verwendet werden um die Instanzsegmentierung zu erstellen. Darüber hinausgehende Kriterien werden in Kapitel 6.2 erläutert.

Kapitel 4

Experimente

In den Experimenten, welche ich in meiner Masterarbeit durchführte, sollten drei hauptsächliche Fragestellungen beantwortet werden. Erstens, welchen Einfluss das gemeinsame Training der verschiedenen Unteraufgaben auf die Leistung hat und zweitens, was die maximal nötige Komplexität der Architektur ist, und drittens, welche Ergebnisse man mit der beschriebenen Architektur in Bezug auf die Semantische Instanzsegmentierung erreichen kann. Im Abschnitt 3.4 wurde auf die zu testenden Architekturen eingegangen. Im Abschnitt 4.3 werden dann die Gegebenheiten des Trainings, unter denen die Modelle zustande gekommen sind, näher erläutert. In Kapitel 5 werden dann die Ergebnisse präsentiert, welche aus den beschriebenen Experimenten resultierten.

4.1 Ziel der Experimente

Wie bereits erwähnt, sollten in den Experimenten hauptsächlich drei Eigenschaften der Architektur untersucht werden: Der Einfluss des gemeinsamen Trainings auf die einzelnen Unteraufgaben, die maximal nötige Komplexität des Netzwerkes und die Leistung der Gesamtarchitekturen für die Semantische Instanzsegmentierung. Da die Leistung der Gesamtarchitektur für die Instanzsegmentierung elementar von den Leistungen der Unteraufgaben der Semantischen Segmentierung, der Relativen Positionsprädiktion und der Objektdetektion abhängig ist, lag zunächst das Augenmerk darauf, den Einfluss des gemeinsamen Trainings auf diese zu evaluieren. Dabei ergaben sich folgende Maximen, nach denen die Experimente gestaltet wurden:

1. Werden durch das gemeinsame Training bessere Abstraktionen gelernt?
2. Welchen Einfluss hat die Abhängigkeit der einzelnen Unteraufgaben voneinander?
3. Welchen Einfluss hat die Komplexität der Architekturen aus Kapitel 3.4 ?: Ressourcen vs. Qualitäts-Tradeoff

Letzteres Kriterium zielt darauf ab, eine möglichst ressourcenschonende Architektur im Zusammenhang mit dem Selbstfahrenden Auto zu konstruieren, welche gleichzeitig gute Ergebnisse liefert. Durch die Kriterien 1. und 2. wird überprüft, ob und wie durch das gemeinsame Training Verbesserungen in Verhältnis zu den auf die Einzelaufgaben spezialisierten Netzwerken erreicht werden können und welchen Einfluss das Einführen von direkten Abhängigkeiten zwischen den einzelnen Aufgaben hat.

Um die maximal nötige Komplexität zu ermitteln, wurden die in den Abschnitten 3.4.3, 3.4.2 und 3.4.1 erläuterten Ideen auf ihren Einfluss geprüft. Die hier zugrundeliegende Maxime ist der Komplexitäts-Leistungs-Tradeoff.

Die Systeme, die sich aus den ersten zwei Maximen als die am besten geeigneten hervorgetan haben, wurden dann auf ihre Leistung in Bezug auf die Instanzsegmentierung geprüft. Allgemein werden die zu diesen Maximen durchgeführten Experimente in Abschnitt 4.2 erläutert.

4.2 Überblick über die Experimente

In den Abschnitten 3.4.3, 3.4.2 und 3.4.1 wurden die in dieser Masterarbeit erdachten Netzwerkstrukturen erläutert. Um diese zu testen und zu schauen, welche Gesamtarchitektur sich am besten für die Instanzsegmentierung in Bezug auf Qualität und Ressourcenverbrauch eignet, wurden die im Folgenden beschriebenen Experimente durchgeführt. Zunächst lag der Fokus darauf, herauszufinden, auf welche Art und Weise das gemeinsame Training der Einzelaufgaben angegangen werden sollte. Zu diesem Zwecke wurden also die in Abschnitt 3.4.1 beschriebenen Konzepte implementiert und trainiert.

Experimente aus Kapitel 5.2:

1. Gemeinsames Training des Faster-RCNN und FCN-8 auf derselben Feature-Map.
2. Gemeinsames Training des Faster-RCNN und FCN-8 auf derselben Feature-Map und Konstruktion der Abhängigkeiten der Detektion von Semantischen Informationen.

Der Einfluss der neu erdachten und in Abschnitt 3.4.2 ausgeführten ROI-Pooling Methode wurde dadurch geprüft, dass die Leistung von Architekturen verglichen wurden, die sich nur durch die Pooling Methode unterschieden.

Experimente aus Kapitel 5.3:

1. (Altes Pooling/ Neues Pooling) sowohl RPN als auch das RCNN bekommen semantische Informationen als Eingabe.
2. (Altes Pooling/ Neues Pooling) nur das RCNN bekommt semantische Informationen als Eingabe.

Durch diese Experimente konnte zudem geprüft werden, ob es notwendig ist, die pixelweisen semantischen Informationen sowohl beim RCNN als auch beim RPN einfließen zu lassen (siehe Kapitel 5.3.1).

Um zu prüfen, ob die Abhängigkeit der Semantischen Segmentierung und der Relativen Positionsbestimmung, wie sie in 3.4.3 beschrieben ist, einen positiven Einfluss hat, oder nur die Komplexität erhöht, wurden auch hier jeweils Architekturen verglichen, die sich nur in Bezug auf diese Eigenschaft unterschieden.

Experimente aus dem Kapitel 5.4:

1. (Abhängigkeit JA/NEIN) Neue Pooling Methode; sowohl RPN als auch RCNN bekommen Semantische Informationen als Eingabe; Prädiktion des Hintergrundlabels für die Relative Positionsbestimmung.
2. (Abhängigkeit JA/NEIN) Neue Pooling Methode; sowohl RPN als auch RCNN bekommen semantische Informationen als Eingabe; keine Prädiktion des Hintergrundlabels für die Relative Positionsbestimmung.

Auf Basis dieser Experimente konnte auch ermittelt werden, ob es sinnvoll ist, das Hintergrundlabel zusammen mit der Relativen Positionsbestimmung zu ermitteln, oder ob es besser ist, dieses durch die semantische Segmentierung zu bestimmen, wie es in Kapitel 3.4.3 beschrieben wurde.

Um zu ermitteln, ob es sinnvoll ist, die Relative Positionsbestimmung der Detektion als weitere Eingabe zur Verfügung zu stellen, wurden folgende Experimente unternommen:

Experimente aus Kapitel 5.4.2:

1. (Einbeziehung JA/NEIN) Neue Pooling Methode sowohl RPN als auch RCNN bekommen semantische Informationen als Eingabe; Prädiktion des Hintergrundlabels für die Relative Positionsbestimmung; keine Abhängigkeit der Relativen Positionsbestimmung und der Semantischen Segmentierung
2. (Einbeziehung JA/NEIN) Neue Pooling Methode; sowohl RPN als auch RCNN bekommen semantische Informationen als Eingabe; Prädiktion des Hintergrundlabels für die Relative Positionsbestimmung; Abhängigkeit der Relative Positionsbestimmung und der Semantischen Segmentierung.

Die sich nun aus dieser Reihe an Experimenten ergebenden besten Modelle in Bezug auf die Einzelaufgaben wurden dann mit den jeweiligen einzeln trainierten Modellen des FCN-8 und des Faster-RCNN in Kapitel 5.6 verglichen. Dieser Vergleich gibt Auskunft darüber, welche Vorteile ein gemeinsames Modell in Bezug auf die Detektion, die Semantische Segmentierung und den Ressourcenverbrauch bieten. Schließlich werden in Kapitel 5.8 die besten Modelle auf ihre Leistung in Bezug auf die Semantische Instanzsegmentierung geprüft. Dabei werden die Ergebnisse durch das in 3.5 dargelegte Kriterium zur Trennung der Instanzen voneinander generiert.

4.3 Training

4.3.1 Vorbereitung des Datensatzes

Das Training und Testen der beschriebenen Modelle erfolgte auf dem Cityscapes Datensatz, auf den in Kapitel 2.5 näher eingegangen wurde. Aus dem oberen Teil der Abbildung 4.1 ergibt sich ein Überblick über die in diesem Datensatz enthaltenen pixelweisen Informationen, indem die Anzahl der Pixel pro Klasse über den gesamten Datensatz hinweg dargestellt wird. Hieraus kann man also entnehmen, wie gut die

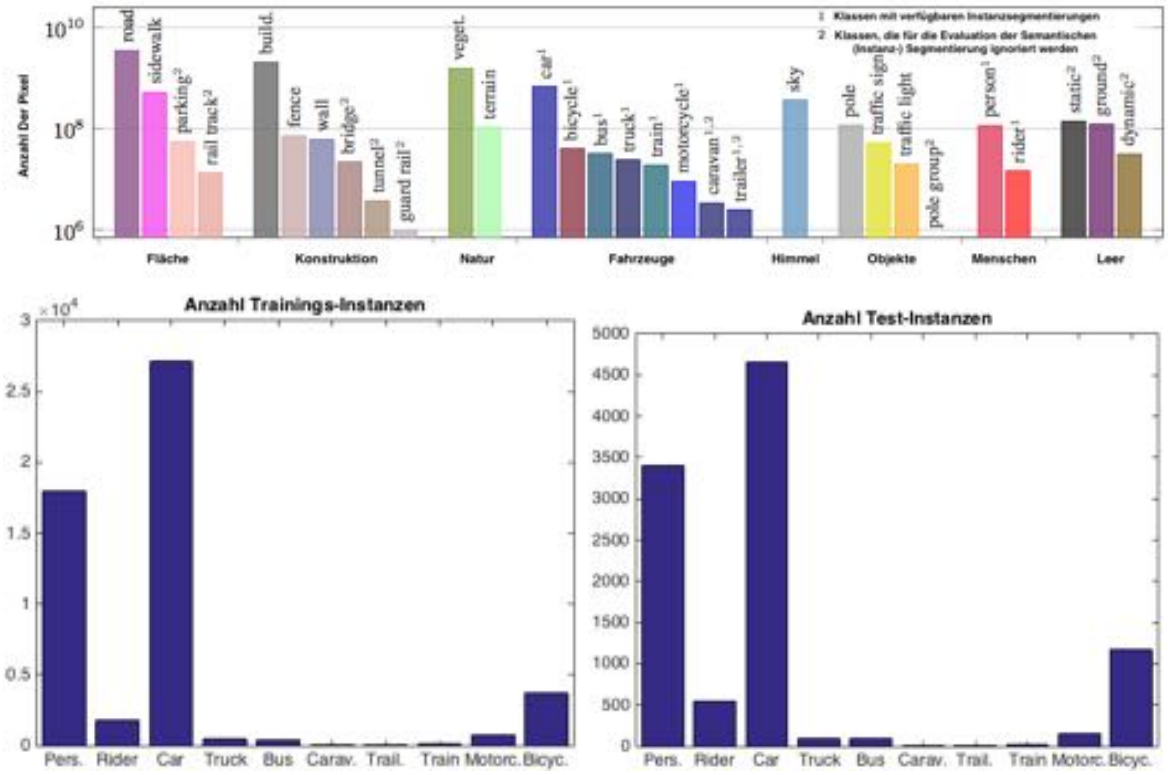

Abbildung 4.1: Auf der rechten Seite ist die Anzahl der Instanzen der einzelnen Klassen auf dem Testdatensatz zu sehen, auf der linken Seite die Verteilung auf dem Trainingsdatensatz. In der oberen Abbildung (entnommen aus [COR+16]) ist die Verteilung der Pixel pro Klasse im Cityscapes Datensatz zu sehen. Zudem kann man hieraus entnehmen, welche Klassen Instanzen besitzen und welche nicht.

Datengrundlage für die Bestimmung der pixelweisen Informationen, also der Semantischen Segmentierung und der Relativen Positionsprädiktion bezogen auf die einzelnen Klassen ist. Dabei werden für das Training der Semantischen Segmentierung alle Klassen verwendet, für die in der Abbildung nicht angegeben ist, dass sie für die Evaluation ignoriert werden. Die während der Evaluation ignorierten Klassen haben, wie in der Abbildung zu sehen, zu wenig Trainingsdaten oder stellen keine wichtigen Klassen dar, wie z.B. die Klassen der Kategorie “Leer“. Diese Klassen wurden als Hintergrundlabel zusammengefasst und auf diese Weise in das Training der Semantischen Segmentierung miteinbezogen.

Zudem kann aus dem oberen Teil der Abbildung auch entnommen werden, welche Klassen Instanzsegmentierungen beinhalten. Die ground-truth Daten für die Relativen Positionsprädiktion wurden jeweils für alle diese Klassen, wie in Kapitel 3.4.3 beschrieben, generiert. Im unteren Teil der Abbildung 4.1 sind unabhängig von der Anzahl der Pixel die Anzahl der Instanzen pro Klassen aufgeführt. Diese Statistik ist vor allem in Bezug auf das Training der Detektion und der Relativen Positionsprädiktion von Bedeutung (aber auch für die Semantische Segmentierung). Bei der Relativen Positionsprädiktion ist diese Statistik deshalb wichtig, da jedes Positionslabel genau einer Instanz zugeordnet ist. Bei der Detektion bilden die Bounding-Boxen der Instanzen die Daten-Grundlage des Trainings. Eine Vielzahl an Instanzen in einer Klasse bietet also eine

gute Repräsentation der Variabilität, in der sich die Instanzen bewegen. Allgemein kann man aus den Abbildungen entnehmen, dass die Verteilung der Pixel auf die einzelnen Klassen gleichmäßiger ist als die Verteilung der Anzahl der Instanzen auf die Klassen. Es zeigt sich zudem, dass die Klasse sowohl im Test als auch im Trainingsdatensatz eine ähnliche Verteilung aufweisen. Dabei haben die Klassen “Person“, “Rider“, “Car“ und “Bicycle“ sehr viele Instanzen und die Klassen “Truck“, “Bus“, “Caravan“, “Trailer“, “Train“ und “Motorcycle“ im Verhältnis zu diesen sehr wenige.

Für das Training der Detektion wurden auch die Klassen Caravan und Trailer für die Detektion mit einbezogen. Im Zusammenhang mit der Semantischen Segmentierung wird im Cityscapes-Datensatz empfohlen, diese nicht mit zu trainieren. Für diese Aufgabe wurden die Klassen auch nicht mit einbezogen. Da jedoch bei der Detektion weniger Informationen bezüglich der Form eines Objektes gelernt werden müssen, wurde versucht, diese hier mit einzubeziehen.

Initial sind keine Detektionsdaten im Cityscapes Datensatz vorhanden. Da diese jedoch aus den maximalen/ minimalen Koordinaten der Instanzen und deren Klasse bestehen, konnten diese aus den Instanzsegmentierungen generiert werden. Der über alle Bilder hinweg bestimmte RGB-Mittelwert aller Pixel ist $[72.38, 82.9, 73.15]$. Dieser wurde zum Normieren der Eingabebilder des Netzwerkes durch die Subtraktion von allen Eingabepixeln verwendet.

4.3.2 Hardware

Bei der zur Verfügung stehenden Hardware handelte es sich für den größten Teil der Masterarbeit um eine Tesla K-20 GPU (Grafikkarte), welche nur $\approx 5GB$ Arbeitsspeicher bietet. Dies begrenzte die maximale Auflösung der Bilder während des Trainings auf 600×300 Pixel und während des Testens auf 1000×500 Pixel. Das Training über 70000 Iterationen dauerte mit dieser GPU ca.4 Tage. Gegen Ende der Masterarbeit stand dann ein Server mit acht GTX-1080 Ti GPUs zur Verfügung, welche mit $11GB$ RAM und schnellerer Taktfrequenz das Training auf einer Größe von 1500×750 Pixel großen Bildern und das Testen auf der vollen Auflösung des Cityscapes-Datensatzes ermöglichten, sodass die Trainingsdauer auf dieser Auflösung auf ca.2 Tage verringert wurde.

4.3.3 Initialisierung des Modells

Die Initialisierung der Architekturen vor dem Training bestand darin, dass die Feature-Map des VGG-16 mit Gewichtungen initialisiert wurde, welche aus einem Faster-RCNN stammten, das auf dem Pascal-VOC Datensatz trainiert wurde. Dieses Modell wird auch verwendet, um die übrigen Schichten des RCNN und RPN des Faster-RCNN zu initialisieren. Die Upsampling-Layer der Semantischen Segmentierung und Relativen Positionsbestimmung wurden in Form einer bilinearen Interpolation initialisiert. Dabei wird der hochskalierte Score-Wert durch die Summe seiner vier nächsten Nachbarn in der geringer aufgelösten Score-Map bestimmt, in welcher jeder dieser vier Werte mit

der inversen Distanz zum aktuellen Score-Wert gewichtet wird. Für die Initialisierung der zusätzlichen 1×1 Faltungen, welche eingeführt wurden, um die semantischen Informationen dem Faster-RCNN zur Verfügung zu stellen, wurden dabei die sogenannte Xavier Initialisierung aus [GB10] verwendet. Diese sorgen dafür, dass die Gewichtungen sich so verhalten, dass Signale in sehr tiefen Netzwerken wie diesem weder zu groß noch zu klein werden.

4.3.4 Hyperparameter und Lerndauer

Die Hyperparameter wurden aus der Implementierung des Faster-RCNN aus https://github.com/rbgirshick/py-faster-rcnn.git (abgerufen am 26.06.2017) übernommen. Dabei wurden für die Interpolationsschichten der Zweige, die pixelweise semantische Informationen prädizieren, jeweils sehr kleine Lernraten von 10^{-15} gewählt, wie es im Zusammenhang mit dem FCN-8 Netzwerk empfohlen wird (siehe "https://github.com/shelhamer/fcn.berkeleyvision.org.git" abgerufen am 26.06.2017). Die Anzahl der Iterationen wurde mit 700000 ebenfalls aus der Implementierung des Faster-RCNN übernommen, da, wie man in Abbildung 5.1 sieht, nach dieser Lerndauer die Loss-Funktonen bereits gegen einen bestimmten Wert konvergierten und kaum noch Änderungen aufwiesen. Diese Iterationsanzahl entspricht ≈ 25.5 Epochen, sodass also jedes Bild ≈ 25 mal im Lernprozess verwendet wurde. Pro Iteration wurde dabei nur ein Bild betrachtet, da so der zur Verfügung stehende Speicher dafür verwendet werden konnte, auf möglichst hoch aufgelösten Bildern zu trainieren. Dies ist vor allem für die Semantische Segmentierung und Relative Positionsprädiktion wichtig, da hier räumliche Details von großer Bedeutung sind, die bei einer Skalierung auf eine kleine Auflösung verloren gehen.

4.3.5 Loss-Funktionen

Die hier verwendeten Loss-Funktionen für die Regression und Klassifikation können in Kapitel 2.1.4 und 2.1.5 nachvollzogen werden. Dabei hat das RPN und das RCNN jeweils einen Loss für die Regression und Klassifikation. Bei RPN handelt es sich dabei um die Klassifikation zwischen Objekt und Hintergrund und die Regression einer Bounding-Box bezüglich einer generellen Instanz ohne besondere Klassenzugehörigkeit. Beim RCNN handelt es sich dabei um die Klassifikation zwischen den Semantischen Klassen und die Regression einer Bounding-Box speziell für diese Klassen. Die Semantische Segmentierung und die Relative Positionsprädiktion haben jeweils einen pixelweisen Klassifikations-Loss (siehe Kapitel 2.1.4). Dieser wird im Zusammenhang mit dem gemeinsamen Training mit der Anzahl der Pixel normiert, da der summierte Loss andernfalls einen größeren Einfluss als die Loss-Funktionen der Detektion hätte. All diese 6 Loss-Funktionen sind dabei während des Trainings gleich gewichtet.

Kapitel 5

Ergebnisse

In diesem Kapitel soll nun auf die Ergebnisse der Experimente eingegangen werden. Dabei ist dieses so strukturiert, dass zunächst einmal analysiert wird, welchen Einfluss das gemeinsame Training und die Erweiterungen der Architektur aus Kapitel 3.4.3, 3.4.2 und 3.4.1 haben und welche dieser Erweiterungen im Verhältnis zur Steigerung des Ressourcenverbrauches Sinn ergeben. Die besten dieser Modelle werden dann näher mit den jeweiligen Einzelmodellen verglichen. Bis zum Kapitel Hiernach wird das Ergebnis der am besten geeigneten Modelle für die Instanzsegmentierung analysiert.

Allgemein werden die Ergebnisse in Form von Tabellen analysiert. Die Tabellen zur Analyse der Detektion sind dabei so aufgebaut, dass sie klassenweise die Average Precission (AP) aus Kapitel 2.5.4 aufführen. Zudem wird der nach Anzahl der Instanzen pro Klasse im Testdatensatz gewichtete Durchschnitt der AP gebildet, welcher mit AVG bezeichnet wird. Die Verteilung der Anzahl der Instanzen im Testdatensatz ist in Abbildung 4.1 unten links zu sehen. Ein Beispiel für eine solche Tabelle findet sich z.B. in 5.1.

Die Semantische Segmentierung wird ausgewertet, indem zunächst jeweils die Klassen mit und ohne Instanzen getrennt voneinander betrachtet werden. Aus Abbildung 4.1 kann man entnehmen, welche Klassen Instanzen besitzen und welche nicht. Als Qualitätsmetrik dient die IoU aus Kapitel 2.5.3. Zusätzlich wird für die Klassen mit Instanzen die iIoU bestimmt, welche eine Betrachtung der Segmentierung ermöglicht, die nicht durch die Größe der Instanzen beeinflusst wird (siehe ebenfalls Kapitel 2.5.3). Die IoU über alle Klassen der Semantischen Segmentierung dient als Maß für das gesamte Ergebnis der Semantischen Segmentierung. Das Maß der IoU wird dann auch verwendet, um jeweils die Segmentierungen der relativen Positionen und des Hintergrundes der Relativen Positionsprädiktion zu bestimmen. Ein Beispiel für eine solche Tabelle findet sich z.B. in 5.2.

Der Ressourcenverbrauch ist jeweils in Bezug auf die Ausführung der Netzwerke mit einer Auflösung von 1000×500 Pixeln auf einer GTX-1080 Ti GPU gegeben und in der Geschwindigkeit und dem Speicherverbrauch während der Auflösung bemessen. Ein Beispiel für eine solche Tabelle zeigt sich anhand Tabelle 5.3.

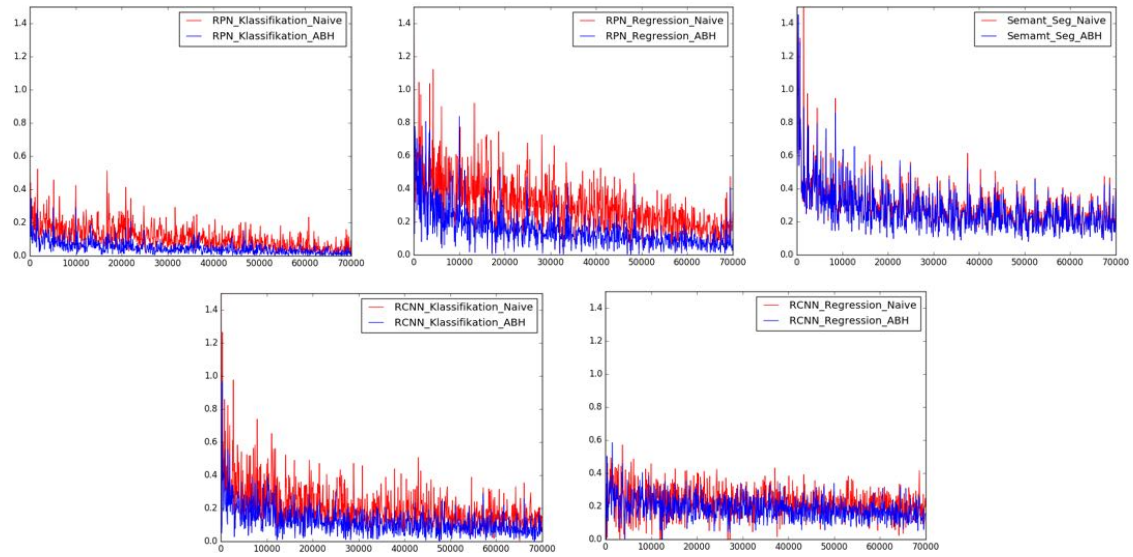

Abbildung 5.1: Hier sind die Trainings-Loss-Funktionen der Modelle aus Kapitel 3.4.1 zu sehen. Die Auflösung der Trainingsbilder betrug 600×300 Pixel. Dabei bezeichnet "Naive" das Modell, bei dem die Detektion und die Semantische Segmentierung auf derselben Feature-Map trainiert werden. "ABH" bezeichnet das Modell, bei dem die Abhängigkeiten der Detektion von der Semantischen Segmentierung aus Kapitel 3.4.1 vorhanden sind. Man kann erkennen, dass die Loss Funktionen von "ABH" schneller und gegen einen geringeren Wert konvergieren.

5.1 Verlauf der Loss-Funktionen

In Abbildung 5.1 ist der Trainings-Loss der Modelle aus Kapitel 3.4.1 zu sehen. Die einzelnen Loss-Funktionen sind dabei zum Vergleich jeweils gemeinsam in einer Abbildung dargestellt. Die Loss-Funktionen des einfachen Trainings der Semantischen Segmentierung und der Detektion auf derselben Feature-Map (Modell mit "Naive" bezeichnet) sind dabei in rot gekennzeichnet. Die Loss-Kurven des Modells (als "ABH" bezeichnet), in welchem die Detektion von der Semantischen Segmentierung abhängig ist (sowohl RPN als auch RCNN, siehe Kapitel 3.4.1 ohne die neue ROI-Pooling Methode), sind jeweils in blau gekennzeichnet. Dabei war der generelle Verlauf der Loss-Funktion für alle Modelle, bei denen die Detektion von den pixelweisen semantischen Informationen abhängig ist, sehr ähnlich, weshalb die Abbildung als repräsentativ für alle diese Modelle angesehen werden kann. Man kann sehen, dass die Einbeziehung der pixelweisen semantischen Informationen dazu führt, dass alle Loss-Kurven der Detektion schneller konvergieren und generell einen geringeren Wert aufweisen. Dies gilt insbesondere für die Loss-Funktionen des Region Proposal Networks (RPN). Dabei fällt auf, dass vor allem der Regressions-Loss stark von der Einbeziehung der Semantischen Segmentierung in die Detektion profitiert. Wenn man den Regressions- und Klassifikations-Loss des RCNN vergleicht, so fällt auf, dass das "ABH" Modell auch hier zu einer geringeren Loss-Funktion führt. Hier ist es jedoch der Klassifikations-Loss, der stärker von der Einbeziehung der Semantischen Informationen profitiert als der Regressions-Loss. Die Loss-Funktion der semantischen Segmentierung hingegen ist jedoch beim Training beider Modelle ungefähr gleich.

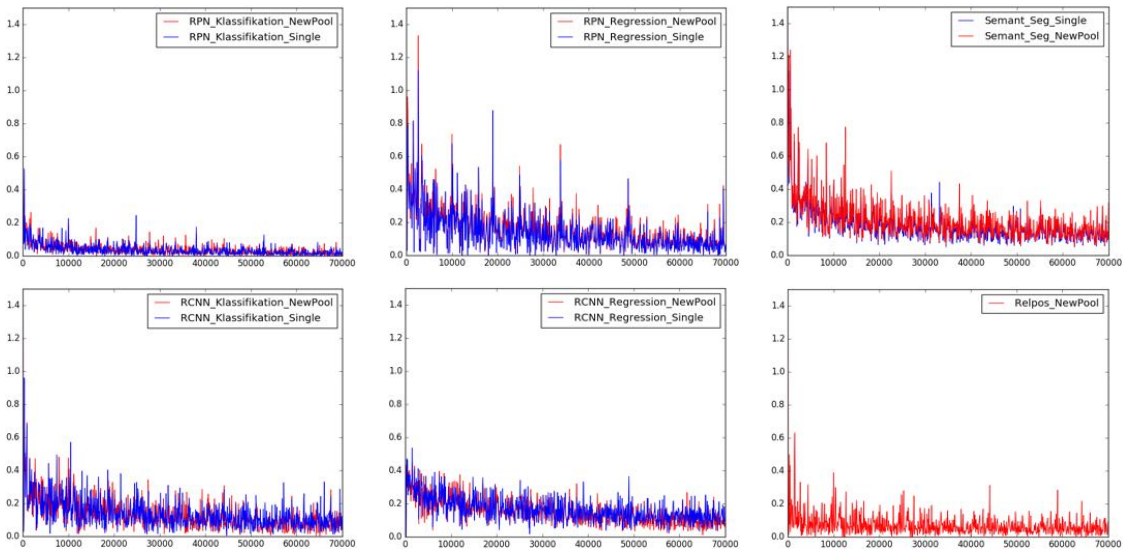

Abbildung 5.2: Hier sind die Loss-Funktionen der Einzelnetzwerke (Faster-RCNN und FCN-8) und eines Modells mit der neuen ROI-Pooling Methode aus Kapitel 3.4.2 dargestellt (Modell "3." aus Kapitel 5.3). Die Auflösung der Bilder betrug 1500×750 Pixel. Wie man sieht, sind die Funktionen allgemein sehr ähnlich. Beim Regressions-Loss des RCNN ergibt sich jedoch ein leichter Vorteil für das Modell mit neuer ROI-Pooling Methode.

In Abbildung 5.3 kann man zudem die Ergebnisse der Unteraufgaben, also der Semantischen Segmentierung, der Relativen Positionsprädiktion und der Detektion über die Iterationen hinweg betrachten. Dabei sind diese Kurven durch eine Architektur zu Stande gekommen, die auch die Relative Positionsprädiktion mitbestimmt und die neue ROI-Pooling Methode verwendet (Modell "3." aus Kapitel 5.3). Der generelle Verlauf der Kurven ist dabei wiederum repräsentativ für alle Modelle mit gemeinsamer Feature-Map. Man kann sehen, dass die Ergebnisse konvergierten und bis zur letzten Iteration nicht wieder schlechter wurden. Insofern kann man davon ausgehen, dass kein Overfitting während des Training der Modelle aufgetreten ist. Zudem kann man in Abbildung 5.2 den Verlauf der Trainings-Loss-Funktionen im Vergleich zu den Loss-Funkionen der Einzelnetzwerke sehen. Die Kurven sind sich dabei sehr ähnlich. Nur beim Regressions-Loss des RCNN ist ein leichter Vorteil des gemeinsamen Modells mit neuer ROI-Pooling Methode gegenüber dem einzelnen Faster-RCNN zu sehen.

5.2 Evaluation des gemeinsamen Trainings nach Kapitel 3.4.3

Wie bereits in Kapitel 3.4.1 angedeutet, sind die Effekte einer gemeinsamen Feature-Map auf das Training und die Inferenz nicht offensichtlich. Durch die Experimente, die in diesem Abschnitt erläutert werden, sollte zunächst geklärt werden, wie eine solche Architektur, die auf einer geteilten Feature-Map basiert, aussehen sollte. Dabei

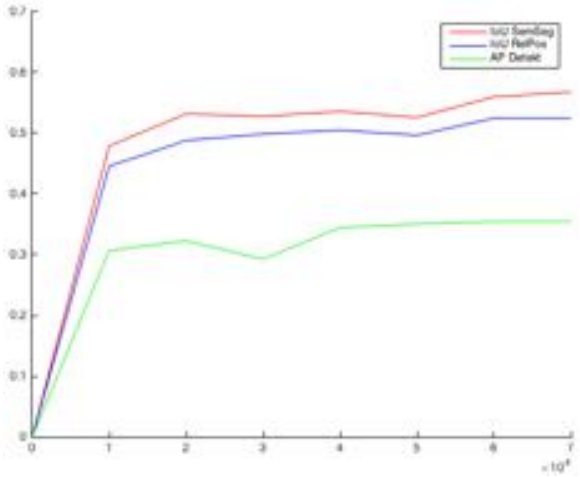

Abbildung 5.3: Hier sind die Ergebnisse auf dem Validierungsdatensatz des Cityscapes Datensatzes in Schritten von 10000 Iterationen zu sehen. Mit AP wird bei der Detektion die über alle Klassen gemittelte AP bezeichnet, wobei die klassenweise AP jeweils mit der Anzahl der Instanzen im Validierungsdatensatz gewichtet ist. Die Auflösung der Testbilder betrug 1000×500. Das Modell ist das Modell "3." aus Kapitel 5.3

Ansatz	Person	Rider	Car	Truck	Bus	Caravan	Trailer	Train	Motorcycle	Bicycle	AVG
Naive	0.2359	0.2841	0.4276	0.3274	0.4241	0.0730	0.0056	0.2087	0.1913	0.2026	0.3240
ABH.	0.237	0.318	0.430	0.270	0.512	0.205	0	0.274	0.178	0.216	0.3292
Einzel	0.2531	0.3535	0.4377	0.2745	0.4678	0.0256	0.0089	0.3986	0.2751	0.2341	0.3433

Tabelle 5.1: Die Struktur der Tabelle, die die Ergebnisse der Detektion zeigt, ist in der Einführung des Kapitels 5 beschrieben. "ABH" und "Naive" sind die Modelle aus Kapitel 3.4.1, wobei "ABH" das Modell bezeichnet, bei dem die Semantische Segmentierung als weitere Basis für die Detektion dient. "Einzel" bezeichnet das "Faster-RCNN".

wurden die beiden Ansätze aus 3.4.3 auf Bildern der Größe 600×300 unter den in Kapitel 4.3 erläuterten Bedingungen trainiert und auf einer Größe von 1000×500 getestet.

In Tabelle 5.1 sind dabei die Ergebnisse der Detektion zu sehen und in Tabelle 5.2 die Ergebnisse der Semantischen Segmentierung. Das Modell, welches durch ein einfaches Training der beiden Aufgaben auf derselben Feature-Map zustande gekommen ist, wird dabei mit dem Namen "Naive" bezeichnet. Die Architektur, welche Abhängigkeiten zwischen der Semantischen Segmentierung und der Detektion konstruiert, indem es dem RPN und RCNN des Faster RCNN die Score-Maps zusätzlich zur Feature-Map zur Verfügung stellt, wird mit dem Namen "ABH" bezeichnet. Durch den Vergleich mit dem Faster-RCNN und dem FCN-8, welche jeweils einzeln für die jeweiligen Aufgaben trainiert wurden, werden die Leistung der neuen Architekturen ins Verhältnis zum jeweiligen alleinigen Modell gesetzt. Diese Architekturen werden dann jeweils mit "Einzel" bezeichnet.

Wenn man die Detektions-Ergebnisse aus Tabelle 5.1 betrachtet, so kann man sehen, dass das einfache Training der Detektion und der Semantischen Segmentierung auf derselben Feature-Map im Verhältnis zu den Ergebnissen des Einzelmodells zu einer

Ansatz	IoU Klassen ohne Inst.	IoU Klassen mit Inst.	iIoU Klassen mit Inst.	IoU Gesamt
Naive	0.5283	0.4574	0.354	0.50
ABH	0.5262	0.4449	0.341	0.494
Einzel	0.5386	0.4345	0.335	0.497

Tabelle 5.2: Die Struktur der Tabelle, die die Ergebnisse der Semantischen Segmentierung zeigt, ist in der Einführung des Kapitel 5 beschrieben. “ABH“ und Naive sind die Modelle aus Kapitel 3.4.1, wobei “ABH“ das Modell bezeichnet, bei dem die Semantische Segmentierung als weitere Basis für die Detektion dient. Einzel bezeichnet das “FCN-8“.

um absolut gesehen 1.93% kleineren Average Precission (AVG-AP) führt. Die durchschnittliche Intersection over Union (IoU) des FCN-8 und des “Naive“ Modells hingegen unterscheiden sich kaum. Beim Training der Semantischen Segmentierung und der Detektion auf einer geteilten Feature-Map verhält sich die Aufgabe der Semantischen Segmentierung, wie auch schon im Kapitel 5.1 zu beobachten, also robuster als die der Detektion.

Durch die in 3.4.3 beschriebene Konstruktion der Abhängigkeit der Detektion von der Semantischen Segmentierung wurde ein Anstieg der gewichteten Average Precission von absolut gesehen 0.52% im Vergleich zum “Naive“ Modell erreicht. Das einzelne Faster-RCNN erreicht mit einer AVG-AP von 0.3433 also immer noch eine absolut gesehen um 1.41% höhere AP. Betrachtet man jedoch die nicht mit der Anzahl der Instanzen der Klasse gewichtete AP, so ergibt sich absolut gesehen ein Anstieg von Modell “Naive“ (0.2380 AVG-AP) zu “ABH“ (0.2640 AVG-AP) von 2.6%AP. Auch der Abstand zum “Einzel“ Modell (0.2729 AVG-AP) verringert sich absolut gesehen bei dieser Betrachtungsweise auf 0.89%. Die Einbeziehung der pixelweisen Semantischen Informationen führt also zu einer Verbesserung bei Klassen, die weniger Instanzen aufweisen. Die Gesamt-IoU ist dabei über alle Modelle (“Einzel“=0.497 ,“Naive“=0.50 und “ABH“=0.494) ähnlich gut.

Generell lässt sich beobachten, dass die Ergebnisse der Semantischen Segmentierung für die Modelle, die zusammen mit der Detektion trainiert wurden, leicht bessere Ergebnisse bei der durchschnittlichen IoU und iIoU der Klassen mit Instanzen haben (“Naive“=0.4574, “ABH“=0.4449 vs. “Einzel“=0.4345 IoU und “Naive“=0.354, “ABH“=0.341 vs. “Einzel“=0.335 iIoU). Die durchschnittliche IoU der Klassen ohne Instanzen ist jedoch beim “Einzel“-Modell leicht besser als bei den mit der Detektion zusammen trainierten Modellen (“Naive“=0.5283, “ABH“=0.5262 vs. “Einzel“=0.5386 IoU).

Die Tabelle 5.3 stellt einen Vergleich zwischen dem Ressourcenverbrauch der gemeinsamen Modelle “Naive“, “ABH“ und den addierten Ressourcen der Einzelmodelle dar. Dabei zeigt sich, dass sowohl das “Naive“ Modell als auch das “ABH“ Modell einen Geschwindigkeitsvorteil bei einer sequentiellen Ausführung der Einzelnetzwerke aufweisen und weniger Speicher bei einer parallelen Ausführung in Anspruch nehmen. Das “ABH“ Modell funktioniert dabei um $\approx 22.1\%$ schneller und verbraucht $\approx 30.69\%$ weniger Speicher. Die Steigerung des “ABH“ Modells zum “Naive“ Modell beträgt $\approx 8.81\%$ in Bezug auf die Geschwindigkeit und $\approx 5.13\%$ im Verhältnis zum Speicher. Die Einbeziehung der pixelweisen Semantischen Segmentierung als weite-

Ansatz	Geschwindigkeit	Speicher
Naive	0.1345s	2.885GB
ABH	0.1426s	3.041GB
Einzel	0.1829s	4.388GB

Tabelle 5.3: Die Struktur der Tabelle ist in der Einführung des Kapitel 5 beschrieben. “ABH“ und “Naive“ sind die Modelle aus Kapitel 3.4.1, wobei “ABH“ das Modell bezeichnet, bei dem die Semantische Segmentierung als weitere Basis für die Detektion dient. In der Zeile “Einzel“ ist der addierte Ressourcenverbrauch des “Faster-RCNN“ und “FCN-8“ zu sehen.

.

Ansatz	Person	Rider	Car	Truck	Bus	Caravan	Trailer	Train	Motorcycle	Bicycle	AVG
16.	0.249	0.384	0.418	0.344	0.469	0.083	0.007	0.143	0.229	0.251	0.3360
15. New	0.281	0.391	0.441	0.321	0.510	0.070	0.004	0.209	0.274	0.266	0.3603
17.	0.264	0.380	0.425	0.289	0.467	0.053	0.017	0.226	0.252	0.246	0.3434
3. New	0.285	0.390	0.442	0.311	0.480	0.233	0.027	0.243	0.275	0.272	0.3626

Tabelle 5.4: Mit dem Suffix “New“ sind die Architekturen bezeichnet, die die neue Pooling Methode aus Kapitel 3.4.2 verwenden. Bei “15.“/“16.“ fließen die pixelweisen Semantischen Informationen nur beim RCNN ein und bei “3.“/“17.“ sowohl beim RPN als auch beim RCNN.

re Grundlage für die Detektion führt also bei einer verbesserten Detektionsleistung und einer ähnlichen Semantischen Segmentierung nur zu einer leichten Steigerung des Ressourcenverbrauches.

5.3 Evaluation der neuen Pooling Methode 3.4.2

Die in diesem Abschnitt präsentierten Ergebnisse dienen dazu, die in Kapitel 3.4.2 beschriebene neue ROI-Pooling Methode zu evaluieren. Die Auflösung, mit der die Architekturen trainiert wurden, beträgt dabei 1500×750 Pixel und die Auflösung, auf der diese Architekturen getestet wurden beträgt 1000×500. Zu diesem Zweck wurden jeweils Architekturen trainiert und getestet, die sich bis auf die ROI-Pooling Methode nicht voneinander unterscheiden. Dabei werden neben der Semantischen Segmentierung auch die Relativen Positionen jeder Instanz prädiziert. Das Hintergrundlabel wird dabei zusammen mit den relativen Positionen vorhergesagt.

Bei der Architektur mit der Nummer “16.“ handelt es sich um ein Modell, das die alte ROI-Pooling Methode verwendet und bei dem die pixelweisen semantischen Informationen nur beim RCNN einfließen. Die Architektur “15.New“ unterscheidet sich zu “16.“ nur dadurch, dass die neue ROI-Pooling Methode verwendet wird. “15.New“ und “16.“ bieten also ein Vergleichspaar, anhand dessen der Einfluss der neuen Pooling-Methode nachvollzogen werden kann. Ein weiteres Vergleichspaar bilden dabei “17.“ und “3.New“, wobei “3.New“ wiederum die neue Pooling Methode besitzt und “17.“ nicht. Bei beiden Modellen fließen dabei die pixelweisen semantischen Informationen sowohl beim RPN als auch beim RCNN ein.

Ansatz	IoU Klassen ohne Inst.	IoU Klassen mit Inst.	iIoU Klassen mit Inst.	IoU Gesamt
16.	0.5792	0.5369	0.354	0.562
15.New	0.5799	0.5273	0.337	0.559
17.	0.5805	0.5713	0.361	0.577
3.New	0.5788	0.5348	0.339	0.561

Tabelle 5.5: Mit dem Suffix "New" sind die Architekturen bezeichnet, die die neue Pooling Methode aus Kapitel 3.4.2 verwenden. Bei "15."/"16." fließen die pixelweisen Semantischen Informationen nur beim RCNN ein und bei "3."/"17." sowohl beim RPN als auch beim RCNN.

Tabelle 5.5 und 5.4 stellen die Ergebnisse der Architekturen "15.New", "3.New", "16." und "17." in Bezug auf die Detektion und die Semantische Segmentierung dar. Bezogen auf die Detektion kann man aus Tabelle 5.4 erkennen, dass sich durch die neue Pooling-Methode sowohl beim Vergleichspaar "15.New"/ "16." (absolut gesehen um 2.43% AVG-AP) als auch beim Vergleichspaar "3.New"/ "17." (absolut gesehen um 1.92% AVG-AP) eine Verbesserung durch die Einführung der neuen ROI-Pooling Methode einstellt. Gleichzeitig geht aus Tabelle 5.5 hervor, dass in beiden Vergleichspaaren die Einführung der neuen Pooling Methode zu einer leichten Verschlechterung der Semantischen Segmentierung führt (absolut gesehen "16." vs. "15.New":=0.3% Gesamt IoU, "17." vs. "3.New":=1.6% Gesamt IoU).

In Tabelle 5.6 kann man den Ressourcenverbrauch der Modelle betrachten. Es ist zu sehen, dass die Modelle mit der neuen Pooling Methode im Durchschnitt $\approx$ 24.0% langsamer sind. Der Speicherverbrauch hingegen erhöht sich im Durchschnitt jedoch nur um $\approx$ 0.81%. Durch die Einführung der neuen ROI-Pooling Methode ergibt sich also der Vorteil einer Verbesserung der Detektionsleistung, welcher jedoch zum Preis einer leichten Verschlechterung der Qualität der Semantischen Segmentierung und einer um $\approx$ 24.0% höheren Ausführungszeit (bei fast gleicher Speicherauslastung) kommt.

5.3.1 Einbeziehung der Semantischen Informationen für das RPN und RCNN

"3.New" und "15.New" ebenso wie "17." und "16." unterscheiden sich nur dadurch, dass bei "3.New" und "17." sowohl beim RCNN als auch beim RPN die pixelweisen semantischen Informationen für die Detektion als Eingabe gestellt werden und bei "15.New" und "16." nur für das RCNN. Folglich kann man, wenn man "3.New" mit "15.New" und "17." mit "16." vergleicht, ermitteln, ob es sinnvoll ist, die semantischen Informationen sowohl für das RPN als auch für das RCNN zur Verfügung zu stellen.

Wie man aus Tabelle 5.4 sehen kann, bietet "3.New" eine um 0.23% bessere Detektion als sein Vergleichsmodell "15.New". Wie aus Tabelle 5.5 zu erkennen, erreichen die beiden Modelle in der Semantischen Segmentierung eine ähnliche Gesamt-IoU. Modell "3.New" bietet jedoch in Bezug auf die Klassen mit Instanzen eine absolut gesehen um 0.75% bessere IoU. Das Modell "17." liefert eine absolut gesehen um 0.74% bessere

Ansatz	Geschwindigkeit	Speicher
16.	0.1672s	3.335GB
15.New	0.2s	3.323GB
17.	0.1692s	3.335GB
3.New	0.217s	3.401GB

Tabelle 5.6: Mit dem Suffix "New" sind die Architekturen bezeichnet, die die neue Pooling Methode aus Kapitel 3.4.2 verwenden. Bei "15."/"16." fließen die pixelweisen semantischen Informationen nur beim RCNN ein und bei "3."/"17." sowohl beim RPN als auch beim RCNN.

Detektion und eine um 1.5% bessere IoU bei der Semantischen Segmentierung als das Vergleichsmodell, in dem keine semantischen Informationen für das RPN als Eingabe dienen.

Gleichzeitig unterscheiden sich der Ressourcenverbrauch, wie man in Tabelle 5.6 sehen kann, zwischen den Modellen ("15.New" vs. "3.New" und "16." vs. "17.") kaum. Man kann also aufgrund der leicht besseren Ergebnisse aus der Semantischen Segmentierung und Detektion folgern, dass es sinnvoll ist, sowohl dem RPN als auch dem RCNN die pixelweisen semantischen Informationen zur Verfügung zu stellen.

5.4 Einfluss der Abhängigkeiten aus 3.4.3

Die hier gezeigten Ergebnisse wurden auf Basis von Modellen generiert, welche auf einer Bildgröße von 600×300 trainiert und auf einer Bildgröße von 1000×500 getestet wurden. In diesem Abschnitt soll nun auf die in Kapitel 3.4.3 eingeführte Konstruktion der Abhängigkeiten zwischen der pixelweisen Semantischen Segmentierung und der pixelweisen Relativen Positionsbestimmung eingegangen werden. In den Tabellen 5.7 und 5.8 sind dabei jeweils die Ergebnisse von Architekturen aufgeführt, die sich nur in Bezug auf diese Abhängigkeiten unterscheiden. Allen Architekturen ist dabei gemein, dass die neue ROI-Pooling Methode aus 3.4.2 verwendet wird, und semantische Informationen (Semantische Segmentierung und Relative Positionsbestimmung) sowohl dem RPN als auch dem RCNN zur Verfügung gestellt werden. "ABH" weist darauf hin, dass die Abhängigkeiten aus 3.4.3 in der jeweiligen Architektur implementiert sind. Das Vergleichspaar "5.ABH"/ "6." repräsentiert dabei Architekturen, die das Hintergrundlabel im Zusammenhang mit der Relativen Positionsbestimmung nicht mit prädiziert. Das Vergleichspaar "4.ABH"/ "3." repräsentiert Architekturen, die das Hintergrundlabel für die Relative Positionsbestimmung mit prädiziert.

Bezogen auf die Detektion führt die Implementierung der Abhängigkeiten nach Kapitel 3.4.3 beim Vergleichspaar "5.ABH"/ "6." absolut gesehen zu einer Verbesserung von 0.95% AVG-AP und beim Vergleichspaar "4.ABH"/ "3." absolut gesehen zu einer Verbesserung von 0.89% AVG-AP. Wenn man Tabelle 5.8 betrachtet, so kann man erkennen, dass in Bezug auf die Semantische Segmentierung oder die Relative Positionsprädiktion keine Unterschiede existieren, die auf eine Überlegenheit der Methoden

Ansatz	Person	Rider	Car	Truck	Bus	Caravan	Trailer	Train	Motorcycle	Bicycle	AVG
5.ABH	0.269	0.330	0.446	0.287	0.458	0.209	0.014	0.319	0.207	0.228	0.3495
6.	0.247	0.304	0.444	0.265	0.415	0.241	0.001	0.281	0.193	0.238	0.3400
4.ABH	0.273	0.339	0.444	0.283	0.418	0.390	0.001	0.290	0.206	0.235	0.3508
3.	0.246	0.316	0.444	0.276	0.485	0.120	0.006	0.389	0.265	0.234	0.3419

Tabelle 5.7: Der Suffix "ABH" weist darauf hin, dass das Modell die Abhängigkeiten zwischen Semantischer Segmentierung und Relativer Positionsprädiktion aus Kapitel 3.4.3 implementiert. Die Modelle "5."/"6." prädizieren bei der Relativen Positionsprädiktion das Hintergrundlabel nicht mit. Das Hintergrundlabel wird wie in Kapitel 3.4.3 beschrieben ermittelt. "4."/"3." prädizieren das Hintergrundlabel in der Relativen Positionsprädiktion mit.

Ansatz	IoU Klassen ohne Inst.	IoU Klassen mit Inst.	iIoU Klassen mit Inst.	IoU Gesamt	mean IoU RelPos — HG IOU
5.ABH	0.5187	0.4213	0.326	0.480	0.4592—HG:=0.980
6.	0.5198	0.4238	0.318	0.481	0.4597—HG:=0.980
4.ABH	0.5199	0.445	0.333	0.490	0.4668—HG:=0.981
3.	0.521	0.4419	0.337	0.490	0.4608—HG:=0.981

Tabelle 5.8: Der Suffix "ABH" weist darauf hin, dass das Modell die Abhängigkeiten zwischen Semantischer Segmentierung und Relativer Positionsprädiktion aus Kapitel 3.4.3 implementiert. Die Modelle "5."/"6." prädizieren bei der Relativen Positionsprädiktion das Hintergrundlabel nicht mit. Das Hintergrundlabel wird wie in Kapitel 3.4.3 beschrieben ermittelt. "4."/"3." prädizieren das Hintergrundlabel in der Relativen Positionsprädiktion mit.

mit oder ohne Abhängigkeiten schließen lässt (alle Unterschiede absolut gesehen $< 1\%$). Die Abhängigkeiten zwischen den pixelweisen semantischen Informationen haben also für diese Aufgaben selber keinen Vorteil ergeben.

Durch die Einführung der Abhängigkeiten zwischen der Semantischen Segmentierung und der Relativen Positionsbestimmung erhöht sich, wie aus Tabelle 5.9 zu entnehmen, die Ausführungszeit um durchschnittlich $\approx 13.05\%$. Der Speicherverbrauch hingegen steigt nur um $\approx 11.29\%$. Die Konstruktion der Abhängigkeiten nach Kapitel 3.4.3 führt also zwar zu einer leichten Verbesserung der Detektion, allerdings ergibt sich auch eine Erhöhung der Ausführungszeiten von $\approx 13.05\%$.

5.4.1 Einbeziehung des Hintergrundlabels in die Relative Positionsbestimmung

Aus den Tabellen 5.7 und 5.8 lässt sich zudem schließen, welchen Einfluss die Einbeziehung des Hintergrundlabels in die Relative Positionsprädiktion hat. Dabei ergeben jeweils "3."/ "6." und "4.ABH"/ "5.ABH" die Vergleichspaare, da diese sich jeweils nur dadurch unterscheiden, dass bei der einen Architektur das Hintergrundlabel für die Relative Positionsprädiktion mitbestimmt wird und bei den anderen nicht.

Es ergibt sich bezüglich der Detektion durch die zusätzliche Prädiktion des Hintergrundlabels bei der Relativen Positionsprädiktion eine leichte Verbesserung um absolut gesehen 0.13% in der AVG-AP beim Vergleichspaar "4.ABH"/ "5.ABH" und beim Vergleichspaar "3."/ "6." um absolut gesehen 0.19%. Wenn man zudem die nicht

Ansatz	Geschwindigkeit	Speicher
5.ABH	0.243s	3.405GB
6.	0.213s	3.365GB
4.ABH	0.2431s	3.405GB
3.	0.217s	3.369GB

Tabelle 5.9: Der Suffix "ABH" weist darauf hin, dass das Modell die Abhängigkeiten zwischen Semantischer Segmentierung und Relativer Positionsprädiktion aus Kapitel 3.4.3 implementiert. Die Modelle "5."/"6." prädizieren bei der Relativen Positionsprädiktion das Hintergrundlabel nicht mit. Dieses wird wie in Kapitel 3.4.3 beschrieben ermittelt. "4."/"3." prädizieren das Hintergrundlabel in der Relativen Positionsprädiktion mit.

nach der Anzahl der Instanzen gewichtete durchschnittliche AP berechnet, so ist die Steigerung noch ausgeprägter. Für das Vergleichspaar "4.ABH"/ "5.ABH" ergibt sich eine Steigerung von absolut gesehen 1.12% und für das Vergleichspaar "3."/ "6." eine Steigerung von absolut gesehen 1.52% in der AVG-AP, sodass man davon ausgehen kann, dass vor allem bei Klassen mit wenig Instanzen die Modelle mit Prädiktion des Hintergrundlabels Vorteile haben. Die durchschnittliche Gesamt-IoU der Semantischen Segmentierung ist um 0.95% leicht besser, genauso wie die Relative Positionsvorhersage mit einer um 0.435% besseren mean IoU der Positionslabel.

Was den Ressourcenverbrauch angeht, so kann man aus Tabelle 5.9 entnehmen, dass durch die zusätzliche Prädiktion des Hintergrundlabels ein durchschnittlicher Anstieg der Ausführungszeit um $\approx 0.8991\%$ folgt und auch der Speicherverbrauch nur um $\approx 0.59\%$ ansteigt. In Anbetracht der Tatsache, dass alle Qualitätsmetriken eine leichte Verbesserung aufweisen und nur eine leichte Erhöhung des Ressourcenverbrauches stattfindet, ist die zusätzliche Prädiktion eines Hintergrundlabels für die Relative Positionsbestimmung also positiv zu bewerten.

5.4.2 Einbeziehung der Relativen Positionsbestimmung in die Detektion

Die Tabellen 5.11 und 5.10 zeigen die Ergebnisse der Experimente, die die Frage klären sollten, ob die Einbeziehung der pixelweisen Relativen Positionsbestimmung in die Detektion notwendig ist. Diese Frage beinhaltet zudem Aspekte des gemeinsamen Trainings. Wie wir bereits in Abschnitt 5.2 gesehen haben, kann nämlich die pixelweise Prädiktion von semantischen Informationen ohne direkte Abhängigkeiten der Detektion von dieser zu schlechteren Ergebnissen bei der Detektion führen. Um diese Frage nun zu klären, wurden wiederum Architekturen trainiert (auf einer Bildgröße von 600×300) und getestet (auf einer Bildgröße von 1000×500), bei denen Abhängigkeiten zwischen der Detektion und der Relativen Positionsprädiktion existieren, welche mit dem Anhang "ABH" gekennzeichnet sind. Die Modelle ohne diesen Suffix haben keine direkten Abhängigkeiten der Detektion von der Relativen Positionsprädiktion. Die Abhängigkeiten der Detektion von der Semantischen Segmentierung existieren jedoch schon. Bei allen hier betrachteten Modellen handelt es sich um Architekturen, die die

Ansatz	Person	Rider	Car	Truck	Bus	Caravan	Trailer	Train	Motorcycle	Bicycle	AVG
4.ABH	0.273	0.339	0.444	0.283	0.418	0.390	0.001	0.290	0.206	0.235	0.3508
2.	0.2462	0.314	0.443	0.279	0.457	0.299	0.005	0.292	0.198	0.234	0.3401
3.ABH	0.246	0.316	0.444	0.276	0.485	0.120	0.006	0.389	0.265	0.234	0.3419
1.	0.267	0.314	0.443	0.265	0.441	0.065	0.004	0.336	0.207	0.231	0.3464

Tabelle 5.10: Die Modelle mit dem Suffix "ABH" beziehen neben der Semantischen Segmentierung auch die Relative Positionsprädiktion mit in die Detektion ein. Die anderen Modelle stellen nur die Semantische Segmentierung als weitere Grundlage für den Detektor zur Verfügung.

Ansatz	IoU Klassen ohne Inst.	IoU Klassen mit Inst.	iIoU Klassen mit Inst.	IoU Gesamt	mean IoU RelPos — HG IOU
4.ABH	0.5199	0.445	0.333	0.490	0.4668—HG:=0.981
2.	0.5203	0.4573	0.341	0.495	0.4636—HG:=0.981
3.ABH	0.521	0.4419	0.337	0.490	0.4608—HG:=0.981
1.	0.5288	0.4581	0.329	0.501	0.461—HG:=0.981

Tabelle 5.11: Die Modelle mit dem Suffix "ABH" beziehen neben der Semantischen Segmentierung auch die Relative Positionsprädiktion mit in die Detektion ein. Die anderen Modelle stellen nur die Semantische Segmentierung als weitere Grundlage für den Detektor zur Verfügung.

neue ROI-Pooling Methode aus 3.4.2 verwenden, und semantische Informationen sowohl dem RPN als auch dem RCNN zur Verfügung stellen.

Das Vergleichspaar "4.ABH" und "2." hat dabei die Besonderheit, dass die in Kapitel 3.4.3 eingeführten Abhängigkeiten zwischen der Relativen Positionsprädiktion und der Semantischen Segmentierung existieren. Hierdurch existieren in der Architektur "2.", auch wenn die pixelweisen Ergebnisse der Relativen Positionsprädiktion nicht direkt mit in die Detektion einfließen, transitive Pfade über den Zweig der Semantischen Segmentierung hin zur Detektion. Das andere Vergleichspaar "3.ABH"/ "1." besitzt die in Kapitel 3.4.3 eingeführten Abhängigkeiten nicht, was auch den Unterschied zum Vergleichspaar "4.ABH"/ "2." darstellt.

Wenn man die Detektionsergebnisse aus Tabelle 5.10 betrachtet, so ist zu beobachten, dass "4.ABH" eine absolut gesehen um 1.7% bessere AVG-AP als der Ansatz "2.ABH" erreicht, bei dem die Relative Positionsprädiktion nicht mit als Informationsgrundlage für die Detektion einfließt. Beim Vergleichspaar "3.ABH"/ "1." ist zu beobachten, dass der Ansatz "1." ohne Einbeziehung der Relativen Positionsprädiktion in die Detektion eine absolut gesehen um 0.45% bessere mit der Anzahl der Instanzen gewichtete AP aufweist (AVG-AP). Wenn man jedoch die ungewichteten durchschnittlichen AP-Werte berechnet, so ergibt sich ein absolut gesehen 2.08% besserer Wert für das Modell "3.ABH", sodass man davon ausgehen kann, dass diese Architektur Instanzen seltener Klassen besser detektiert. Tabelle 5.11 zeigt, dass die Semantische Segmentierung bei beiden Vergleichspaaren etwas besser beim Modell ohne die direkte Einbeziehung der Relativen Positionsprädiktion funktioniert.

Den Einfluss der Transitiven Abhängigkeiten der Detektion von der Relativen Positionsprädiktion kann man anhand des Vergleichspaares "1."/ "2." erkennen. Wie man sieht, erreicht das Modell "1." ohne diese transitiven Abhängigkeiten etwas bessere

Ansatz	Geschwindigkeit	Speicher
4.ABH	0.2431s	3.405GB
2.	0.1822s	3.323GB
3.ABH	0.217s	3.401GB
1.	0.1838s	3.287GB

Tabelle 5.12: Die Modelle mit dem Suffix "ABH" beziehen neben der Semantischen Segmentierung auch die Relative Positionsprädiktion mit in die Detektion ein. Die anderen Modelle stellen nur Semantische Segmentierung als weitere Grundlage für den Detektor zur Verfügung.

Detektionsergebnisse bezüglich der gewichteten AP (absolut gesehen 0.63% besser). Das Modell "2." hat jedoch Vorteile bei seltenen Klassen wie "Bus", "Truck" oder "Caravan", was sich auch in der Differenz der nicht gewichteten gemittelten AP zeigt, bei der das Modell einen Vorteil von absolut gesehen 1.95% hat. Die Semantische Segmentierung und Relative Positionsprädiktion funktionieren dabei bei beiden Modellen ungefähr gleich gut.

Aus Tabelle 5.12 kann man erkennen, dass der Ressourcenverbrauch bezüglich Geschwindigkeit und Speicher bei einer direkten Einbeziehung der Relativen Positionsprädiktion deutlich ansteigt. Die transitive Abhängigkeit der Detektion von der relativen Positionsprädiktion führt bei einer ungefähr gleichen Ausführungsgeschwindigkeit nur zu einem leichten Anstieg des Speicherverbrauches ($0.0360GB$). Generell lässt sich also sagen, dass die direkte oder transitive Einbeziehung der Relativen Positionsprädiktion zu einer leichten Verbesserung der Detektionsleistung bei seltenen Klassen führt. Bei der transitiven Abhängigkeit wird dieser Vorteil ohne starke Steigerung des Ressourcenverbrauches erreicht.

5.5 Bestimmung der besten gemeinsamen Modelle

In diesem Anschnitt soll nun aus den vorangegangenen Experimenten die Kombination an Merkmalen ermittelt werden, die die besten Ergebnisse für die Semantische Segmentierung und die Detektion liefern und gleichzeitig einen möglichst geringen Ressourcenverbrauch aufweisen.

Zunächst einmal ist zu konstatieren, dass bei der Konstruktion eines Modells, bei dem die pixelweisen semantischen Informationen (die Semantische Segmentierung oder die Relative Positionsvorhersage) und die Detektion auf der gleichen Feature-Map basieren, Abhängigkeiten zwischen der pixelweisen Prädiktion und der Detektion von Vorteil sind, da sich andernfalls die Detektionsergebnisse verschlechtern (siehe Kapitel 5.2). Die Abhängigkeiten der Detektion von der Relativen Positionsprädiktion können dabei auch transitiv sein, wenn bereits direkte Abhängigkeiten zur Semantischen Segmentierung existieren (siehe Kapitel 5.4.2).

In Kapitel 5.3 wurde gezeigt, dass die neue ROI-Pooling Methode aus Kapitel 3.4.2 zu besseren Ergebnissen führt, jedoch auch im Vergleich zur originalen ROI-Pooling Methode einen höheren Ressourcenverbrauch aufweist. Gleichzeitig konnte durch diese Experimente ermittelt werden, dass die Einbeziehung der pixelweisen semantischen Informationen beim RPN und RCNN Sinn ergibt, da dadurch die Ergebnisse der Detektion und der Semantischen Segmentierung besser werden, was nur mit einer leichten Erhöhung des Speicheraufwandes einhergeht. Aus dem Kapitel 5.4 lässt sich entnehmen, dass die Konstruktion der Abhängigkeiten aus Kapitel 3.4.3 zu einer leichten Verbesserung der Detektion führt, sich jedoch gleichzeitig die Ausführungszeit stark erhöht. Zudem konnte keine starke Verbesserung der Relativen Positionsprädiktion oder der Semantischen Segmentierung erzielt werden, weshalb diese Steigerung der Komplexität eher negativ zu bewerten ist. Die zusätzliche Prädiktion des Hintergrundlabels führt, wie in Kapitel 5.4.1 untersucht, zu einer Verbesserung der Detektion bei einem gleichzeitig nur geringfügig erhöhtem Ressourcenverbrauch.

Aus den Ergebnissen kann man also folgern, dass eine Architektur für die Semantische Instanzsegmentierung am sinnvollsten ist, welche die pixelweisen semantischen Informationen für das RPN und das RCNN nutzt, die neue ROI-Pooling Methode verwendet und im Zusammenhang mit der Relativen Positionsbestimmung das Hintergrundlabel mitprädiziert. Ein solches Modell ist durch “3.“ gegeben. Eine Alternative zu “3.“ wäre, um weiter Ressourcen zu sparen, auch das Modell “2.“, in dem die Detektion nur transitiv über die Abhängigkeiten aus Kapitel 3.4.3 von der Relativen Positionsprädiktion abhängig ist. Die Erhöhung des Ressourcenverbrauchs durch die Abhängigkeiten wird dabei durch die nicht vorhandene Einbeziehung der Relativen Positionsprädiktion in die Detektion ausgeglichen.

Sollte keine Instanzsegmentierung von Nöten sein, so ist die Konstruktion einer Architektur nach diesen Prinzipien möglich, die die Relative Positionsvorhersage nicht prädiziert und so weitere Ressourcen spart. Das Modell “18“ aus Tabelle 5.13 und 5.17 repräsentiert so ein System. Wenn man zudem die Ergebnisse aus Kapitel 5.2 betrachtet, so kann man sehen, dass auch ohne die neue ROI-Pooling Methode bereits ein Modell entsteht, dessen Ergebnisse in Bezug auf die nicht gewichtete durchschnittliche AP auf dem Niveau der Einzelnetzwerke liegen. Insofern ergibt die Konstruktion einer Architektur, die das Ergebnis der Semantischen Segmentierung für das RPN und das RCNN als Eingabe erhält und die alte ROI-Pooling Methode verwendet, Sinn, da diese nochmals weniger Ressourcen verbraucht. So eine Architektur ist durch das Modell “14“ aus den Tabellen 5.13 und 5.17 gegeben.

5.6 Vergleich der besten gemeinsamen Modelle mit den Einzelmodellen

In diesem Kapitel sollen nun die besten Modelle, welche auf einer geteilten Feature-Map basieren, mit den Einzelmodellen verglichen werden. Dabei beschäftigte sich das vorherige Kapitel damit, das Modell zu finden, welches unter diesen neu entwickelten Architekturen die besten Eigenschaften hat und somit das beste Modell darstellt. Hieraus

Ansatz	Person	Rider	Car	Truck	Bus	Caravan	Trailer	Train	Motorcycle	Bicycle	AVG
14.	0.263	0.377	0.424	0.302	0.479	0.233	0.023	0.222	0.255	0.242	0.3424
18.	0.275	0.377	0.440	0.308	0.505	0.252	0.009	0.350	0.294	0.259	0.3569
2.	0.269	0.372	0.441	0.331	0.473	0.198	0.016	0.310	0.242	0.263	0.3545
3.	0.285	0.390	0.442	0.311	0.480	0.233	0.027	0.243	0.275	0.272	0.3626
Faster-RCNN	0.259	0.366	0.431	0.320	0.480	0.061	0.018	0.328	0.273	0.264	0.3467

Tabelle 5.13: Die Definition der Modelle ist in Kapitel 5.5 gegeben. Anhand dieser Tabelle kann dann die Detektionsleistung der Architekturen mit dem einzelnen Faster-RCNN verglichen werden.

Ansatz	IoU Klassen ohne Inst.	IoU Klassen mit Inst.	iIoU Klassen mit Inst.	IoU Gesamt	mean IoU RelPos — HG IOU
14.	0.5798	0.5622	0.363	0.573	
18.	0.5798	0.54	0.3436	0.5638	
2.	0.5786	0.5513	0.354	0.568	0.5243—HG:=0.982
3.	0.5788	0.5347	0.339	0.561	0.5243—HG:=0.982
FCN-8	0.592	0.5593	0.390	0.579	

Tabelle 5.14: Die Definition der Modelle ist in Kapitel 5.5 gegeben. Anhand dieser Tabelle kann dann die Qualität der Semantischen Segmentierung der Architekturen mit dem einzelnen FCN-8 verglichen werden.

ergaben sich die Architekturen “14.“ und “18.“, welche die Relative Positionsbestimmung nicht mitprädizieren und somit einen Vergleich mit den Einzelmodellen zulassen, der nicht von dieser zusätzlichen Aufgabe beeinflusst wird. Die Modelle unterscheidet dabei, dass “18.“ die neue ROI-Pooling Methode verwendet und “14.“ nicht. Allen hier betrachteten Architekturen ist gemein, dass das Ergebnis der Semantischen Segmentierung als Eingabe für das RPN und das RCNN dient. “3.“ und “2.“ sind die Modelle, die auch die Relative Positionsbestimmung ermitteln, und somit als Grundlage für eine Instanzsegmentierung dienen können. Diese unterscheidet dabei, dass “2.“ die Relative Positionsbestimmung nicht direkt als Eingabe für die Detektion bereitstellt, sondern nur transitive Abhängigkeiten der Detektion von der Relativen Positionsbestimmung durch die Abhängigkeiten aus Kapitel 3.4.3 existieren. Die Modelle wurden alle unter den gleichen Bedingungen trainiert und getestet. Die Größe der Trainingsbilder war dabei 1500×750 und die Größe der Testbilder war 1000×500.

Zunächst einmal lässt sich aus den Tabellen 5.13 und 5.17 ersehen, dass durch das Modell “14.“ sowohl bei der Semantischen Segmentierung als auch bei der Detektion ähnliche Ergebnisse erzielt werden konnten wie bei den Einzelmodellen. Der Ressourcenverbrauch des Modells ist jedoch wesentlich geringer als der addierte Ressourcenverbrauch der Einzelnetzwerke. So ergibt sich für Modell “14.“ eine Ausführungszeit, die $0.0403s$ geringer ist bei einem um $1.315GB$ kleineren Speicherverbrauch.

Die Einführung der neuen ROI-Pooling Methode führt dabei dazu, dass die Methoden “18.“,“3.“ und “2.“ bessere Ergebnisse bezüglich der AP (AVG Wert durchschnittlich um absolut gesehen $\approx 1.13\%$ besser) der Detektion aufweisen. Dies führt jedoch im Durchschnitt zu einer absolut gesehen um $\approx 1.4\%$ etwas schlechteren Gesamt-IoU der Semantischen Segmentierung. Die Modelle “18.“ und “2.“ bieten hierbei ungefähr die

Ansatz	Geschwindigkeit	Speicher
14.	0.1426s	3.073GB
18.	0.1846s	3.147GB
2.	0.1822s	3.323GB
3.	0.217s	3.401GB
Einzel	0.1829s	4.388GB

Tabelle 5.15: Die Definition der Modelle ist in Kapitel 5.5 gegeben. Anhand dieser Tabelle kann dann der Ressourcenverbrauch der Architekturen mit dem addierten Ressourcenverbrauch des einzelnen FCN-8 und Faster-RCNN verglichen werden

gleiche Ausführungszeit wie die addierten Einzelmodelle, verbrauchen jedoch beide mehr als einen GB weniger Speicher. Dabei verbraucht "18." $\approx 0.176GB$ weniger Speicher als "2." und erreicht ungefähr die gleiche gewichtete AP. "2." hat dabei jedoch den Vorteil, dass es zusätzlich die Relative Positionsprädiktion bestimmt und somit die Grundlagen für eine Instanzsegmentierung bietet. Dieses erreicht dabei bezüglich der gewichteten AP der Detektion ein um absolut gesehen 0.81% etwas schlechteres Ergebnis als das Modell "3.", hat jedoch einen um $\approx 0.078GB$ etwas geringeren Speicherverbrauch und eine um $0.0348s$ geringere Ausführungszeit, welche beim Modell "3." höher als bei den addierten Einzelmodellen ist. Zudem bietet das Modell "2." eine etwas bessere Leistung in der Segmentierung von Klassen mit Instanzen. Für die Bestimmung der Instanzsegmentierung bietet das Modell "2." also das beste Gesamtsystem. Die transitiven Abhängigkeiten der Detektion von der Relativen Positionsprädiktion reichen also aus, um ähnliche Ergebnisse zu erzielen wie bei direkten Abhängigkeiten.

Allgemein lässt sich über den Vergleich der gemeinsamen Modelle mit den Einzelmodellen folgendes sagen: Bei einer parallelen Ausführung des FCN-8 und des Faster-RCNN ergibt sich durch alle gemeinsamen Modelle eine Verbesserung bezüglich der Speicherauslastung. Die Ausführungszeit der Einzelmodelle ist dann jedoch geringer (sowohl FCN-8 als auch Faster RCNN haben eine Ausführungszeit $\approx 0.09145s$). Bei einer sequentiellen Ausführung des FCN-8 und des Faster-RCNN ist es möglich, durch das Modell "14." Ergebnisse ähnlicher Qualität in 77.9% der Zeit zu bekommen oder bessere Ergebnisse in ähnlicher Zeit durch die Modelle "18." und "2.". Die Speicherauslastung der sequentiellen Ausführung der Einzelmodelle ist dann jedoch geringer (sowohl FCN-8 als auch Faster RCNN belegen ≈ 2.194 GB).

5.6.1 Qualitativer Vergleich auf der vollen Auflösung

Dieser Abschnitt beschäftigt sich nun mit dem qualitativen Vergleich der Ergebnisse der Detektion und Semantischen Segmentierung. Dabei liegt ein besonderes Augenmerk auf der Detektion, da diese von der Semantischen Segmentierung abhängig ist und durch das neue ROI-Pooling beeinflusst wird. Die Modelle wurden auf einer Größe von 1500×750 trainiert und auf der vollen Auflösung der Cityscapes Bilder von 2048×1024 getestet. Das Modell "2." soll als das beste gemeinsame Modell (bezüglich Verhältnis von Ressourcenverbrauch und Qualität), das die neue Pooling Methode implementiert

Ansatz	Person	Rider	Car	Truck	Bus	Caravan	Trailer	Train	Motorcycle	Bicycle	AVG
TH:0.5;G	0.424	0.542	0.608	0.413	0.620	0.197	0.025	0.424	0.451	0.415	0.5151
TH:0.5;E	0.421	0.533	0.598	0.379	0.593	0.188	0.029	0.390	0.403	0.419	0.5082
TH:0.7;G	0.237	0.310	0.439	0.325	0.446	0.185	0.015	0.209	0.098	0.190	0.3285
TH:0.7;E	0.237	0.310	0.426	0.220	0.424	0.000	0.002	0.210	0.136	0.192	0.3220
TH:0.8;G	0.128	0.128	0.328	0.152	0.296	0.000	0.000	0.018	0.023	0.091	0.2151
TH:0.8;E	0.134	0.082	0.303	0.134	0.293	0.000	0.002	0.000	0.091	0.091	0.2040

Tabelle 5.16: Mit TH ist der IoU Schwellenwert abgekürzt. G steht für das Modells "2." und E für das einzelne"Faster-RCNN". Man kann aus der Tabelle die Leistung über die IoU-Schwellenwerte 0.5, 0.7 und 0.8 ablesen, was einen Rückschluss auf die Qualität der Bounding-Boxen zulässt.

und gleichzeitig als Grundlage für die Relative Positionsbestimmung dienen kann, stellvertretend zum Vergleich dienen.

Tabelle 5.16 dient dabei als Indikator dafür, ob "2." eine verbesserte Regression der Bounding-Boxen erreicht. Dabei wurden die Average Precission-Werte jeweils für die IoU-Schwellenwerte von 0.5, 0.7 und 0.8 generiert. Der Zusatz "E" weist auf das einzelne "Faster-RCNN" hin und der Zusatz "G" auf das Modell "2.". Wenn man nun die Tabelle 5.16 analysiert, so lässt sich erkennen, dass das Modell "2." in den AP Werten über alle IoU-Schwellenwerte überlegen ist. Der Unterschied zwischen den beiden Architekturen ist dabei beim höchsten IoU-Schwellenwert mit absolut gesehen 1.11% in der durchschnittlichen AP am größten. Insofern kann man davon ausgehen, dass das Modell "2." in der Tat Bounding-Boxen generiert, die etwas besser an die Objekte angepasst sind. Eine Klasse, in der dieser Unterschied besonders groß ist, ist die Klasse "Truck". In Abbildung 5.4 ist ein Beispiel dafür gezeigt, wie das Modell "2." (rechts) bessere Bounding-Boxen um die Klasse "Truck" im Vergleich zum "Faster-RCNN" (links) zieht.

Abbildung 5.4: Die Bilddaten für diese Abbildung stammen aus [COR+16]. Auf der linken Seite ist das Ergebnis der Detektion des einfachen Faster-RCNN zu sehen und auf der rechten Seite das Ergebnis des Modells "2.". Man kann sehen, dass für die beiden LKWs und das weiße Auto durch "2." bessere Bounding-Boxen generiert werden.

Der Vergleich der Kurven des Modells "2." und des "Faster-RCNN" aus Abbildung 5.5 soll als Maß dafür dienen, ob das gemeinsame Modell kleinere Objekte besser detektiert. Diese zeigen die durchschnittliche Anzahl der Falschen Negativen pro Bild und den negativen Prädikativen Wert über für Konfidenz-Schwellenwerte von $[0, 1]$. Die oberen Kurven sind dabei über alle Klassen hinweg bestimmt, sodass die Klassen gemäß der

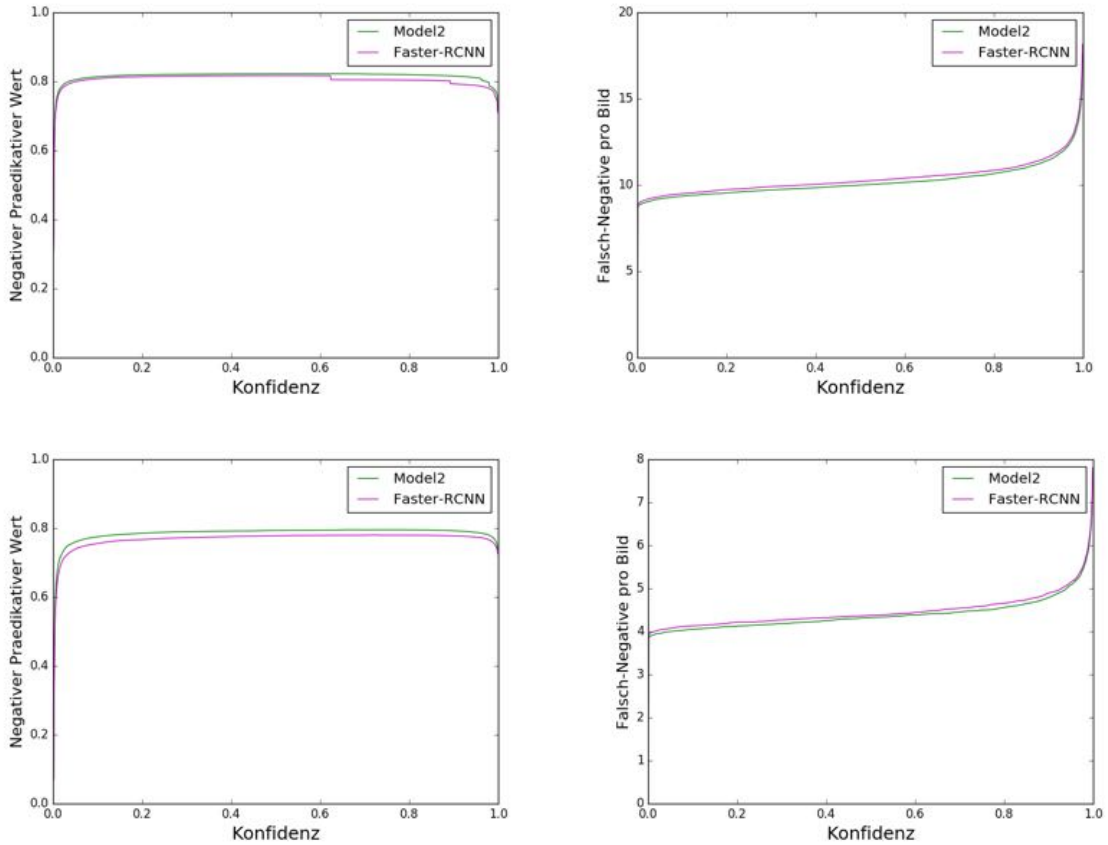

Abbildung 5.5: Die oberen Kurven sind auf Basis aller Klassen bestimmt worden. Die unteren beiden Abbildungen nur auf Basis der Klassen Rider und Person. Dabei sind die Klassen mit der Anzahl der Instanzen gewichtet.

Anzahl ihrer Instanzen gewichtet werden. Die unteren Kurven sind auf die gleiche Art speziell über die Klassen "Rider" und "Person" gemittelt bestimmt worden. Da die Netzwerke bei großen Objekten meist keine Probleme haben, eine Bounding-Box mit einer IoU von mindestens 0.5 zu generieren, sind die falsch negativen Fälle meist durch fehlende Detektionen kleiner Objekte bedingt. Durch den negativen Prädikativen Wert $(RichtigNegative/(RichtigNegative + FalschNegative))$ wird geprüft, ob bei einem geringen Wert an falsch negativen Detektionen auch gleichzeitig viele richtig negative Detektionen existieren.

In den oberen Kurven kann man sehen, dass Modell "2." in Bezug auf alle Klassen und Konfidenz-Schwellenwerte hinweg einen höheren negativen Prädikativen Wert aufweist und weniger falsch negative Detektionen pro Bild generiert. Dabei ist der Unterschied der Negativen Prädikativen Werte bei hohen Konfidenz-Schwellenwerten zwischen $[0.6, 0.9]$ am höchsten. Auch wenn dieser Unterschied allgemein eher klein ist, so kann man doch von einer leichten Verbesserung in der Detektion kleiner Objekte ausgehen. Bei den Instanzen der Klassen "Rider" und "Person" ist es im Zusammenhang mit dem Selbstfahrenden Auto besonders wichtig, dass eine Detektion vorhanden ist, da diese Menschen repräsentieren, deren Verhalten zur schwer vorhersagbar ist und die keinen physischen Schutz haben. Das Modell "2." weist auch für diese Klassen leicht weniger

Ansatz	IoU Klassen ohne Inst.	IoU Klassen mit Inst.	iIoU Klassen mit Inst.	IoU Gesamt
2.	0.6287	0.5348	0.438	0.5907
FCN-8	0.6469	0.5635	0.432	0.6118

Tabelle 5.17: Hier ist ein Vergleich der Semantischen Segmentierung des Modells "2." und des FCN-8 zu sehen.

falsch negative Detektionen auf. Die negativen Prädikativen Werte weisen dabei einen besonders großen Unterschied zum einzelnen "Faster-RCNN" auf, sodass man bei einer fehlenden Detektion beim Modell "2." eher davon ausgehen kann, dass tatsächlich kein Fußgänger oder Fahrradfahrer vorhanden ist.

Wie man aus der Tabelle 5.17 entnehmen kann, erreicht das einzelne "FCN-8" eine absolut gesehen um 2.11% bessere Gesamt-IoU und ist auch in der speziell für die Klassen mit und ohne Instanzen bestimmten IoU überlegen. In der iIoU, welche ein Maß für die Segmentierung von Klassen mit Instanzen unabhängig von deren Größe ist, erreichen beide Modelle ein ähnliches Ergebnis.

Generell kann man also sagen, das "2." eine bessere Detektion aufweist, die in der Lage ist, genauere Bounding-Boxen zu bestimmen und leicht weniger falsch negative Detektionen erzeugt. Dies kommt jedoch zum Preis einer schlechteren Semantischen Segmentierung.

5.7 Zwischenfazit der Experimente

Es lassen sich zunächst einmal folgende Erkenntnisse aus den vorangegangenen Experimenten konstatieren:

1. Die pixelweise Prädiktion Semantischer Informationen ist robuster als die Detektion in Bezug auf das gemeinsame Training; siehe Kapitel 5.2 und 5.1.

2. Durch die Einbeziehung der pixelweisen semantischen Informationen konvergieren alle Loss-Funktionen schneller und gegen einen geringeren Wert als ohne die Abhängigkeiten der Detektion von den pixelweisen semantischen Informationen; siehe Kapitel 5.1.

3. Die Einbeziehung der pixelweisen semantischen Informationen führt beim RPN vor allem zu einem geringeren Loss der Bounding-Box Regression und beim RCNN zu einem geringeren Loss der Bounding-Box Klassifikation; siehe Kapitel 5.1.

4. Die Konstruktion von direkten Abhängigkeiten zwischen der pixelweisen Prädiktion semantischer Informationen und der Detektion ist wichtig, da sonst die Detektion schlechter wird; siehe Kapitel 5.2.

5. Die Konstruktion von transitiven Abhängigkeiten zwischen der Relativen Positionsbestimmung und der Detektion reicht bei einer gleichzeitigen direkten Abhängigkeit der Detektion von der Semantischen Segmentierung aus; siehe Kapitel 5.4.2.

6. Durch die neue ROI-Pooling Methode aus Kapitel 3.4.2 ergeben sich bessere Detektioneigenschaften im Verhältnis zu den gemeinsamen Modellen, welche zum Preis einer etwas schlechteren Semantischen Segmentierung und einem höheren Ressourcenverbrauch kommen; siehe Kapitel 5.3 und 5.6.

7. Durch die Abhängigkeiten zwischen Semantischer Segmentierung und Relativer Positionsbestimmung aus Kapitel 3.4.3 ergeben sich leicht bessere Detektioneigenschaften, jedoch auch ein wesentlich höherer Ressourcenverbrauch. Die Qualität der Semantischer Segmentierung und Relativer Positionsbestimmung ändert sich kaum; siehe Kapitel 5.4.

8. Es ist sinnvoll, sowohl dem RCNN als auch dem RPN die pixelweisen semantischen Informationen zur Verfügung zu stellen; siehe 5.4.2.

9. Die zusätzliche Prädiktion eines Hintergrundlabels bei der Relativen Positionsbestimmung führt zu einer leichten Verbesserung der Detektionseigenschaften bei einer leichten Erhöhung des Ressourcenverbrauches; siehe Kapitel 5.4.1.

10. Auch im Vergleich zu den Einzelmodellen ergeben sich durch die neue ROI-Pooling Methode bessere Detektionsergebnisse und eine leicht schlechtere Semantische Segmentierung; siehe Kapitel 5.6.

11. Ausgehend von einer sequentiellen Ausführung der Einzelnetzwerke ergibt sich bei der neuen ROI-Pooling Methode in etwa die gleiche Ausführungszeit; siehe Kapitel 5.6.

12. Bei Verwendung der alten ROI-Pooling Methode und der Einbeziehung der Semantischen Segmentierung in die Detektion (beim RPN und RCNN) ergeben sich ähnliche Ergebnisse wie bei den Einzelnetzwerken. Ausgehend von einer sequentiellen Ausführung ist das gemeinsame Modell um 22.1% schneller; siehe Kapitel 5.6.

13. Ausgehend von einer parallelen Ausführung der Einzelnetzwerke verbrauchen alle gemeinsamen Modelle mindestens 1GB weniger Speicher; siehe Kapitel 5.6.

14. Bei einer sequentiellen Ausführung nehmen die Einzelnetzwerke weniger Speicher in Anspruch und bei einer parallelen Ausführung ergibt sich eine geringere Ausführungszeit als bei den gemeinsamen Modellen; siehe Kapitel 5.6.

15. Durch die neue ROI-Pooling Methode werden die Bounding-Boxen besser an die Objekte angepasst; siehe Kapitel 5.6.1.

Ansatz	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle	AVG
5.	0.118	0.051	0.284	0.109	0.182	0.086	0.062	0.067	0.120
2.	0.104	0.043	0.278	0.109	0.156	0.044	0.051	0.049	0.104
Rang	8	10	5	8	8	11	9	7	8
5. AP 50%	0.343	0.208	0.516	0.198	0.309	0.228	0.194	0.246	0.281
2. AP 50%	0.303	0.191	0.496	0.237	0.289	0.217	0.179	0.184	0.262
Rang	7	10	4	8	7	9	10	6	7

Tabelle 5.18: Hier sind die Ergebnisse des Modells "2." und "5." (siehe Kapitel 5.4) aufgeführt. Die Metriken können in Kapitel 2.5.5 nachvollzogen werden. AVG ist der ungewichtete Mittelwert der AP Werte über alle Klassen.

5.8 Ergebnisse für die Instanzsegmentierung

In diesem Abschnitt soll nun darauf eingegangen werden, wie sich die in Kapitel 5.5 bestimmten besten Architekturen im Zusammenhang mit der Semantischen Instanzsegmentierung verhalten. Dabei werden in Tabelle 5.18 die einzelnen Ansätze anhand der Metriken aus 2.5 miteinander verglichen. Zudem wird mithilfe von Bildern die Leistung der Architekturen auf dem Cityscapes Datensatz vermittelt. Dabei werden durch visuelle Beispiele die Fehler und Leistungen der Methode verdeutlicht. Die Experimente, die hier gezeigt werden, wurden mit Modellen erstellt, die auf einer Bildgröße von 1500×750 trainiert wurden. Bei der Größe der Bilder, auf denen die Ergebnisse erstellt wurden, handelt es sich um die Originalgröße von 2048×1024. Allgemein muss man zudem beachten, dass die Ergebnisse auf dem Validierungsdatensatz bestimmt wurden und nicht auf der vollen Auflösung der Bilder trainiert werden konnten, da der RAM der GPUs nicht ausreichend war. Zudem beinhaltete die Detektion auch die Klassen "Caravan" und "Trailer", welche bei den anderen Modellen nicht mittrainiert wurden.

Aus der Tabelle 5.18 lässt sich zunächst einmal erkennen, dass die entwickelte Methode mit einer Average Precission von $10.4 - 12.0\%$ und einer "AP 50%" von $26.2 - 28.1\%$ ungefähr die Leistung des Ansatzes aus [KLA+16] (siehe Kapitel 3.1) auf dem Cityscapes-Testdatensatz erreicht. Damit würde die Methode, wenn man also die Leistung auf dem Validierungsdatensatz zu Grunde legt, auf dem 8. Platz der Rangliste des Cityscapes Datensatzes landen (von 12 der strukturell unterschiedlichen Methoden). Die Klasse Auto (Rang 5 in Metrik $AP/$ 4 $AP50\%$ basierend auf Modell "5.") ist dabei besonders gut im Verhältnis zu den anderen Ansätzen auf der Cityscapesrangliste. Diese Klasse ist dabei natürlich von großer Bedeutung im Bereich des Autonomen Fahrens und besitzt in den Testdaten viele Instanzen.

Wenn man die Modelle untereinander vergleicht, so fällt auf, dass der Ansatz "5." die besten Ergebnisse liefert und in beiden Metriken (absolut gesehen um $1.6\% AP / 1.9\% AP50\%$) besser funktioniert als der Ansatz "2.". Dies ist so, obwohl, wie in Tabelle 5.19 zu sehen,

Ansatz	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle	AVG
5.	0.426	0.541	0.609	0.384	0.609	0.341	0.402	0.421	0.4666
2.	0.424	0.542	0.608	0.413	0.620	0.424	0.451	0.415	0.4871

Tabelle 5.19: AVG bezeichnet in diesem Fall den nicht gewichteten Mittelwert über alle AP-Werte der Klassen. Die hier gezeigten Detektionsergebnisse sind dabei auf der vollen Auflösung von 2048×1024 entstanden.

das Modell "2." eine etwas bessere Detektion erreicht. Der hauptsächliche Unterschied des Modells "5." gegenüber "2." in Bezug auf die Instanzsegmentierung ist dabei, dass das Hintergrundlabel nicht als weitere Klasse bei der Relativen Positionsbestimmung mitprädiziert wurde. Der Hintergrund wurde in "5.", wie in 3.4.3 beschrieben, dadurch bestimmt, dass Pixel, die in der Semantischen Segmentierung einer Klasse ohne Instanzen zugeordnet wurden, in der Relativen Positionsbestimmung als Hintergrund angenommen werden. Wie man aus Tabelle 5.20 sieht, ergeben sich für das Modell "5." für fast alle relativen Positionen sowie für die Semantische Segmentierung ("Objekt") bessere Ergebnisse. Dabei ist zu beachten, dass jeweils alle Pixel in der Semantischen Segmentierung und Relativen Positionsprädiktion als Hintergrund betrachtet wurden, die in einem der beiden Ergebnisse als Hintergrund (nicht als Klasse mit Instanzen) klassifiziert wurden. In Anbetracht der Tatsache, dass "5." bei der Relativen Positionsprädiktion kein Hintergrundlabel prädiziert, werden hier also weniger Pixel ausgeschlossen.

5.8.1 Analyse zur Entstehung der Ergebnisse

In diesem Abschnitt sollen nun die für die Erstellung der Semantischen Instanzsegmentierung wichtigen Unteraufgaben analysiert werden. Auf Basis dieser Betrachtung kann dann analysiert werden, wie die Ergebnisse zu Stande gekommen sind und wo Verbesserungen vorzunehmen sind, damit bessere Ergebnisse für die Instanzsegmentierung erreicht werden können. Dabei wird dies anhand des Modells "5." gemacht, da dieses wie im vorherigen Abschnitt erläutert, die besten Ergebnisse liefert.

Tabelle 5.19 zeigt die Detektionsergebnisse der Klassen, für die die Instanzsegmentierung generiert wurde. In Tabelle 5.20 sind die klassenweisen Ergebnisse der Relativen Positionsprädiktion aufgeführt. Dabei sind die Klassen jeweils so kodiert, dass "o" für oben, "m" für Mitte, "u" für unten, "l" für links und "r" für rechts steht, sodass "o+r" z.B. die relative Position oben rechts kodiert. In der Spalte "Objekt" ist dabei das Ergebnis der Semantischen Segmentierung für diese Klasse in Form der IoU mit der ground-truth-Segmentierung angegeben. Indem man diese Ergebnisse gemeinsam betrachtet, kann man die Ergebnisse der Instanzsegmentierung erklären.

Allgemein müssen bei dieser Methode für eine gute Instanzsegmentierung sowohl eine gute Semantische Segmentierung der Klasse als auch eine gute Detektion und eine gute Relative Positionsbestimmung, die mit der Detektion übereinstimmt, vorhanden sein. Für die Klasse "car" sind sowohl eine gute Detektion als auch eine gute Semantische Segmentierung gegeben. Dabei erreicht auch die Relative Positionsprädiktion über alle

Ansatz	o+l	o+m	o+r	m+l	m+m	m+r	u+l	u+m	u+r	Objekt
5.Person	0.299	0.455	0.275	0.326	0.436	0.318	0.296	0.336	0.293	0.670
5.Rider	0.160	0.316	0.257	0.222	0.269	0.286	0.088	0.139	0.063	0.380
5.Car	0.539	0.609	0.571	0.620	0.604	0.619	0.616	0.592	0.600	0.883
5.Truck	0.125	0.135	0.154	0.189	0.214	0.215	0.263	0.224	0.231	0.291
5.Bus	0.235	0.317	0.296	0.286	0.304	0.256	0.319	0.356	0.247	0.507
5.Train	0.047	0.219	0.127	0.093	0.113	0.136	0.142	0.120	0.212	0.390
5.Motorc.	0.083	0.124	0.100	0.210	0.226	0.190	0.228	0.209	0.242	0.359
5.Bicycle	0.085	0.147	0.111	0.339	0.261	0.294	0.387	0.257	0.354	0.513
5.AVG	0.1966	0.2903	0.2364	0.2856	0.3034	0.2893	0.2924	0.2791	0.2802	0.4991
2.AVG	0.1833	0.2538	0.2048	0.2682	0.2771	0.2807	0.2924	0.2687	0.2838	0.4916

Tabelle 5.20: Die Tabelle zeigt die Auswertung der relativen Positionen für jede Klasse in Form der IoU mit den ground-truth Daten.“o“ steht für oben, “m“ für mittig, “u“ für unten, “l“ für links und “r“ für rechts. “o+l“ kodiert also z.B. das Label “oben links“. “Objekt“ gibt das Ergebnis der Semantischen Segmentierung an, wobei diese so generiert wird, dass sowohl die Relative Positionsprädiktion einen Pixel als Vordergrund definieren muss als auch die eigentliche Semantische Segmentierung.

Label hinweg im Verhältnis zu den anderen Klassen gute Ergebnisse. Insofern ergibt sich für diese Klasse auch mit einer AP der Instanzsegmentierung von (0.120/0.281) ein gutes Ergebnis. In Abbildung 5.6 ist ein Beispiel für die einzelnen Ergebnisse der Unteraufgaben zu sehen. Dabei zeigt sich, dass auch überlappende Instanzen der Klasse “car“ gut durch das Kriterium aus Kapitel 3.5 voneinander getrennt werden. Auffällig ist hierbei auch die hohe Übereinstimmung der inferierten pixelweisen relativen Positionen mit denen der prädizierten Bounding-Box.

Klassen, die ein ähnliches Aussehen und mithin eine ähnliche Breite an Formvariationen haben, jedoch im Durchschnitt wesentlich größere Instanzen aufweisen, sind “Truck“, “Bus“ und “Train“. Wenn man sich jedoch die Ergebnisse der Relativen Positionsprädiktion anschaut, so fällt auf, dass diese im Verhältnis zur Klasse “car“ bei “Truck“ (durchschnittlich 0.4525 schlechtere IoU), “Bus“ (durchschnittlich 0.3442 schlechtere IoU) und “Train“ (durchschnittlich 0.5201 schlechtere IoU) wesentlich schlechter sind. Bei “Truck“ und bei “Train“ ist dabei auch die Semantische Segmentierung am schlechtesten, was dann auch dazu führt, dass diese Klassen ein schlechteres Ergebnis bei der Instanzsegmentierung erreichen. Die Klasse “Bus“ erreicht zudem mit einer durchschnittlichen IoU von 0.3270 die beste relative Positionsbestimmung unter diesen Klassen, was dann zu einer Instanzsegmentierung von 0.182/0.309 führt.

Zwei weitere Klassen, die sich in ihrer Gestalt und Formvariabilität sehr ähneln, sind “Person“ und “Rider“. Dabei erreicht die Klasse “Person“ jedoch mit $0.118AP/0.343AP50\%$ ein besseres Ergebnis als die Klasse “Rider“ mit $0.051AP/0.208AP50\%$. Dies ist vornehmlich der besseren Semantischen Segmentierung (eine um 0.29 bessere IoU) und der besseren Relativen Positionsprädiktion (eine im Durchschnitt 0.1542 bessere IoU) zuzuschreiben. Allgemein erreichen Klassen, deren Instanzen eher kleine Strukturen darstellen, eher schlechte Ergebnisse bei der Relativen Positionsprädiktion.

Dies lässt sich auch bei den Klassen “Motorcycle“ und “Bicycle“ feststellen. In Abbildung 5.7 z.B. ist zu sehen, dass für die Fahrräder keine gute Relative Positionprädiktion erstellt wird. Diese Klassen erreichen mit einer AP von 0.062 und 0.067 sehr schlechte Instanzsegmentierungen, welche jedoch bei der AP50% einen starken Anstieg aus 0.194 und 0.246 erfahren. Dies kann darauf zurück geführt werden, dass bei einer Detektions-AP von 0.402 (“Motorcycle“) und 0.421 (“Bicycle“) zwar Detektionen für die Instanzen existieren, diese jedoch meistens eine geringe Überlappung mit der ground-truth Instanz aufweisen, da die Semantischen Segmentierungen mit 0.359 und 0.513 eher schlecht sind und die Instanzen selber nur schwer über die relativen Positionen voneinander getrennt werden können.

Allgemein scheint das größte Problem der Semantischen Instanzsegmentierung nach dem hier angewendeten Prinzip die pixelweise Prädiktion der semantischen Informationen zu sein. Insbesondere funktioniert die Bestimmung der relativen Positionen, wie man an fast allen Klassen außer “Car“ sieht, noch nicht gut genug. Die Detektion hingegen scheint einen geringeren negativen Einfluss zu haben, wie man an dem großen Anstieg von der AP auf die $AP50\%$ in allen Klassen sieht. $AP50\%$ hat nämlich nur einen Schwellenwert von 50% bezüglich der IoU der einzelnen Instanzen mit den ground-truth Instanzen und kann somit eher als Maß für eine grobe Lokalisation der Instanzen gesehen werden, welche ja durch die Detektion gegeben wird.

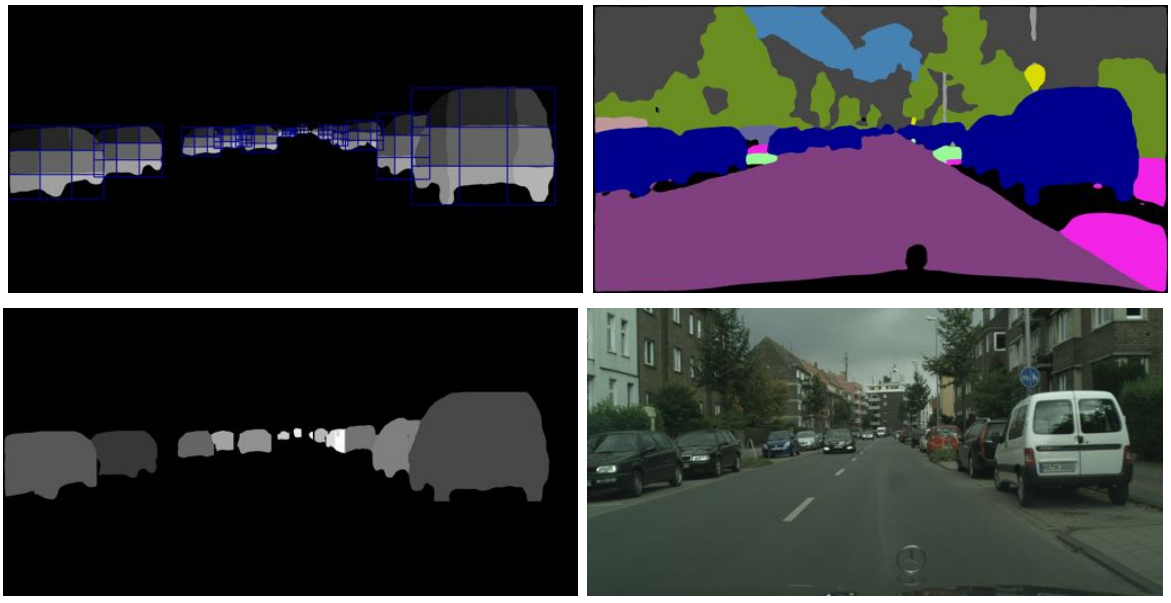

Abbildung 5.6: Die Daten zur Generierung der Abbildung stammen aus [COR+16]. Die Netzwerkausgaben wurden mit Modell “5.“ generiert. Oben links ist das Ergebnis der Relativen Positionsprädiktion und der Detektion zu sehen. Man kann erkennen, dass für beide Aufgaben gute Ergebnisse generiert wurden, und dass die relativen Positionen innerhalb der Bounding-Boxen meist gut mit den pixelweisen relativen Positionen übereinstimmen. Oben rechts ist die prädizierte Semantische Segmentierung zu erkennen, welche ebenfalls gute Ergebnisse liefert. Die Kombination der Unteraufgaben durch das Kriterium aus Kapitel 3.5.1 zu einer Instanzsegmentierung ist unten links zu sehen, wobei für unterschiedliche Instanzen jeweils unterschiedliche Grauwerte zugeordnet wurden. Es ergibt sich ein gutes Ergebnis, bei dem auch überlappende Autos gut voneinander getrennt wurden.

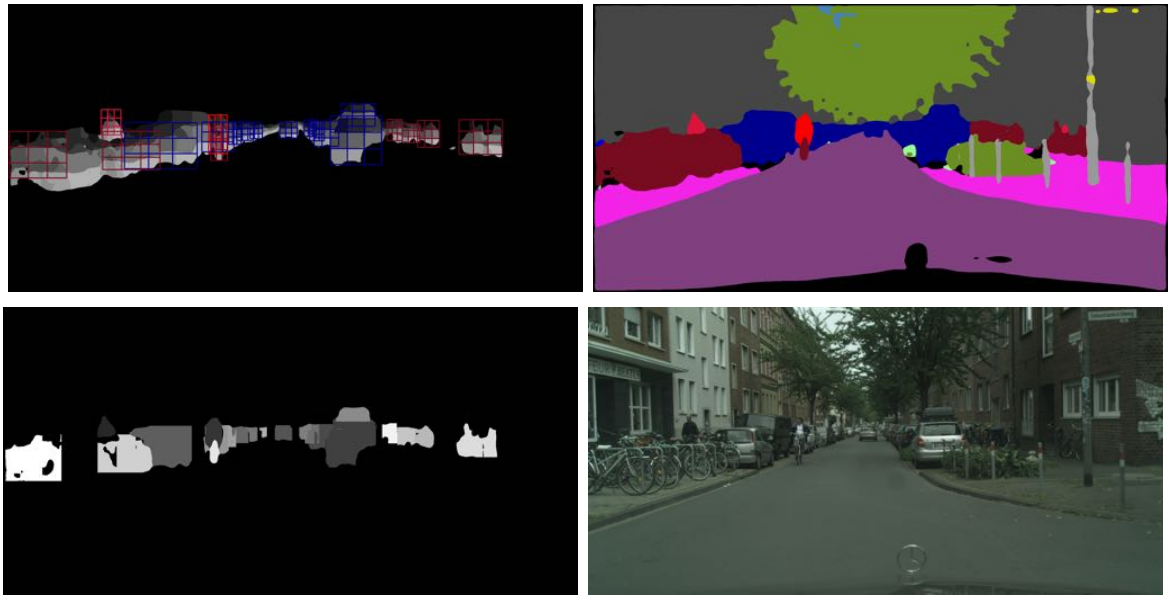

Abbildung 5.7: Die Daten zur Generierung der Abbildung stammen aus [COR+16]. Die Netzwerkausgaben wurden mit Modell "5." generiert. Oben links ist das Ergebnis der Relativen Positionsprädiktion und der Detektion zu sehen. Man kann sehen, dass die Detektion gute Ergebnisse liefert. Die Relative Positionsprädiktion ist jedoch vor allem bei den Fahrrädern unsicher und kann nicht mehr zur Trennung der Instanzen verwendet werden. Dies kann man auch in der Instanzsegmentierung unten links sehen. Unten rechts ist das Bild zu sehen, auf dem die Ausgaben bestimmt wurden.

Kapitel 6

Diskussion der Experimente

In diesem Kapitel sollen nun die hauptsächlichen Erkenntnisse aus den vorangegangenen Experimenten näher betrachtet werden. Dabei ist diese Diskussion in drei grundsätzliche Teile unterteilt. In Abschnitt 6.1 soll das Zwischenfazit aus Kapitel 5.7 näher betrachtet werden, welches sich mit den Leistungen des Netzwerkes auf den einzelnen Unteraufgaben beschäftigt. Der Abschnitt 6.2 beschäftigt sich dann mit den Ergebnissen, die bezüglich der Semantischen Instanzsegmentierung in Kapitel 5.8 ermittelt wurden. Eine generelle Betrachtung des Einsatzes der entwickelten Netzwerke im Selbstfahrenden Auto findet sich dann in Kapitel 6.3.

6.1 Leistung der Architekturen für die Semantische Segmentierung und Detektion

6.1.1 Effekte des gemeinsamen Trainings

In Kapitel 5.7 sind die Erkenntnisse aufgeführt, die nun in diesem Abschnitt bezüglich des gemeinsamen Trainings auf derselben Feature-Map besprochen werden sollen. Wie man aus Aussage 4. entnehmen kann, werden durch das naive Training der Detektions-und Segmentierungsaufgabe auf derselben Feature-Map nicht bessere, sondern schlechtere Ergebnisse für die Detektion generiert. Dies kann neben der Initialisierung oder der Wahl der Hyperparameter auch an der unterschiedlichen Struktur der Aufgaben liegen. Es ist möglich, dass sich die Features, die für die Semantische Segmentierung benötigt werden, sich von denen der Detektion auf eine Weise unterscheiden, dass die jeweils andere Aufgabe von diesen nicht profitieren kann. Im Falle der Detektion kann dies dann sogar zu schlechteren Ergebnissen führen. Die Optima, die die einzelnen Loss-Funktionen der pixelweisen Semantischen Segmentierung und der Detektion haben, scheinen jedoch nicht gänzlich unterschiedlich zu sein, da beide Aufgaben immer noch gute Ergebnisse liefern. Allgemein scheint die Semantische Segmentierung, wie aus Aussage 1. zu entnehmen, weniger stark durch das gemeinsame Training beeinflusst zu werden, was möglicherweise daran liegt, dass es sich hierbei um eine lokale Aufgabe handelt, bei der für jedes Pixel einer Struktur eine unabhängige Klassifikation bestimmt wird. Bei der Detektion hingegen werden Strukturen, also Instanzen, in ihrer Gesamtheit betrachtet, sodass bei einer auf einer schlechten Feature-Map basierenden fehlerhaften Detektion gleich eine ganze Instanz falsch detektiert wird.

Bei der Konstruktion von Abhängigkeiten der Detektion von der Semantischen Segmentierung wird, wie aus Aussage 4. abzulesen, eine Verbesserung der Detektion erreicht, ohne dass die Segmentierung darunter leidet. Dabei wird der Unterschied der Optima dadurch ausgeglichen, dass eine gute Semantische Segmentierung nun auch zu einer guten Detektion führt. Vor allem bei Klassen, die eher weniger Instanzen im Test und Trainings-Datensatz haben, lässt sich dabei ein Anstieg in der Detektionsleistung beobachten. Dies liegt wahrscheinlich daran, dass die Semantische Segmentierung für diese Instanzen wirklich neue Informationen liefert, wohingegen bei Klassen mit vielen Instanzen die Semantische Segmentierung zusätzlich zur Feature-Map eher redundante Informationen liefert. Die direkte Einbeziehung der relativen Positionen in die Detektion führt auch deshalb, wie aus Aussage 4. abzuleiten, zu keiner starken Verbesserung, da derartige Informationen bereits durch die Loss-Funktion der Regression der Bounding-Boxen in der Feature-Map erzeugt werden. Allgemein muss man zudem noch beachten, dass der Pfad zwischen der Semantischen Segmentierung und der Detektion auch dazu führt, dass diese Abhängigkeit der Detektion von der Semantischen Segmentierung auch bei der Berechnung der Gradienten während des Backpropagation Algorithmus einen Einfluss hat. Folglich wird jede Änderung an den Gewichtungen in der Feature-Map vereinfacht ausgedrückt so bestimmt, dass eine gute Detektion entsteht und eine gute Semantische Segmentierung, die als gute Grundlage für die Detektion dienen kann. Dieser Einfluss während der Backpropagation kann auch der Grund dafür sein, dass die transitiven Abhängigkeiten der Detektion von der Relativen Positionsprädiktion, wie in Aussage 5. ausgeführt, zu einer Verbesserung der Detektion von Klassen mit wenig Instanzen im Trainings und Testdatensatz führt.

6.1.2 Effekte der neuen Netzwerkstrukturen

Wie in Aussage 6. zu sehen, wird durch die Einbeziehung der pixelweisen semantischen Informationen durch die neue ROI-Pooling Methode aus Kapitel 3.4.2 eine Verbesserung der Detektion erreicht. Im Gegensatz zur Einbeziehung der skalierten pixelweisen Informationen wie in Kapitel 3.4.1 ergeben sich bei diesem Ansatz neue Detailinformationen über die im Bild anwesenden Objekte, da die pixelweisen semantischen Informationen in der Größe des Eingabebildes vorhanden sind. Dies führt, wie in Aussage 15. zu sehen, zu einer genaueren Generierung von Bounding-Boxen. Allerdings wird, wie in Aussage 6. zu lesen, durch diese Methode auch die Semantische Segmentierung etwas schlechter. Dies ist wahrscheinlich darin begründet, dass die pixelweisen Konfidenzen auf zwei Weisen in das RPN und das RCNN einfließen. Beim RPN werden diese erst durch eine Reihe von Average-Poolings herunterskaliert und beim RCNN fließen diese in der Originalauflösung durch das ROI-Pooling mit in die Detektion ein. Wie man aus Kapitel 5.2 entnehmen kann, führt das Zurverfügungstellen der pixelweisen Semantischen Informationen nur durch das Herunterskalieren durch die Poolings nicht zu einer Verschlechterung der Semantischen Segmentierung. Der Einfluss, den die Detektion auf die Semantische Segmentierung hat, liegt in der Berechnung der Gradienten während des Backpropagations-Algorithmus. Man kann also davon ausgehen, dass die zusätzliche Einbeziehung der pixelweisen semantischen Informationen in Größe des Eingabebildes

zu einer für die Semantische Segmentierung schlechteren Berechnung der Gradienten führt.

Die Abhängigkeiten der Semantischen Segmentierung und der Relativen Positionsbestimmung aus Kapitel 3.4.3 führen nicht zu einer Verbesserung der Semantischen Segmentierung oder der Relativen Positionsprädiktion, jedoch zu einer leichten Verbesserung der Detektion (Aussage 7.). Anscheinend entsteht auf Basis der Kombination der Feature-Map und der Semantischen Informationen des jeweils anderen Zweiges keine bessere Datengrundlage für die Semantische Segmentierung oder die Relative Positionsbestimmung. Es scheinen jedoch jeweils durch diese Verbindungen bessere pixelweise Scores für die Eingabe in die Detektion produziert zu werden.

6.1.3 Effekte der Modelle auf den Ressourcenverbrauch

Was die Verlangsamung der Ausführungszeit angeht, so hat diese folgende hauptsächliche Gründe:

1. Zusätzliche Faltungen

2. Durch Abhängigkeiten muss auf das Ergebnis anderer Zweige gewartet werden

Diese treffen z.B. auf die Einbeziehung der pixelweisen semantischen Informationen in die Detektion zu. Hier wird zum einen, wie in Kapitel 3.4.1 näher erläutert, eine 1×1 Faltung eingeführt. Zum anderen muss der Detektionszweig neben der Feature-Map auf die Erstellung der Semantische Segmentierung warten, was zu einer weiteren Verzögerung führt. Wenn zusätzlich noch die relative Positionsprädiktion mit einbezogen wird, so müssen sowohl diese als auch die Semantische Segmentierung sowie die Feature-Map bereits berechnet worden sein, damit die Detektion berechnet werden kann. Wenn zusätzlich noch die Semantische Segmentierung und die Relative Positionsbestimmung, wie in Kapitel 3.4.3 ausgeführt, gegenseitig von den Ergebnissen des jeweils anderen Zweiges abhängen, so führt dies zu weiteren Verzögerungen. Im Falle der neuen ROI-Pooling Methode aus Kapitel 3.4.2 wird die Steigerung der Ausführungszeit zudem dadurch verursacht, dass neben dem ROI-Pooling auf der gering aufgelösten Feature-Map auch ein ROI-Pooling auf den pixelweisen semantischen Informationen in der Auflösung des Eingabebildes ausgeführt wird. Dieses zusätzliche ROI-Pooling wird neben der größeren räumlichen Ausdehnung der Eingabe auch auf zusätzlichen Kanälen ausgeführt.

Was die Speicherauslastung angeht, so wird diese dann erhöht, wenn zusätzliche Faltungen eingefügt werden oder wenn die Anzahl der Kanäle der zu verarbeitenden Eingabe größer wird. Dies ist z.B. der Fall bei der Einbeziehung der Semantischen Segmentierung in die Detektion, bei der, wie in Kapitel 3.4.3 beschrieben, eine zusätzliche 1×1 Faltung eingeführt wird. Oder bei der Erhöhung der Kanaltiefe, wenn sowohl die Relativen Positionen als auch die Semantische Segmentierung als zusätzliche Eingabe für die Detektion dienen.

6.2 Leistung für die Semantische Instanzsegmentierung

Hier sollen nun die Erkenntnisse aus dem Kapitel 5.8 näher betrachtet werden. Allgemein ergab sich aus diesem Kapitel, dass für die Semantische Segmentierung schon ein Ergebnis erzielt werden konnte, welches im Verhältnis zu den anderen verfügbaren Aufgaben auf dem Cityscapes Datensatz in der jetzigen Form bereits eine durchschnittliche Leistung erbringt. Bei den hier präsentierten Ergebnissen muss jedoch beachtet werden, dass die Netzwerkarchitektur, welche für die Erstellung der Feature-Map verwendet wurde, also das VGG-16, nicht mehr dem aktuellen Stand der Technik entspricht. Die Netzwerke, die auf diesem Datensatz die besten Ergebnisse erzielen, verwenden nämlich alle neuere Netzwerkarchitekturen, welche besser als als VGG-16 funktionieren. Zudem wurden diese Netzwerke auf der vollen Auflösung des Cityscapes-Datensatzes trainiert, was vor allem für die pixelweise Semantische Segmentierung oder Relative Positionsprädiktion nochmal eine starke Verbesserung bringen würde. Da diese, wie in Kapitel 5.8 beschrieben, noch das größte Problem der Methode darstellen, ergäbe sich hieraus auch eine starke Verbesserung der Instanzsegmentierung. Allgemein ist es ein Vorteil der hier entwickelten Methode, dass man die einzelnen Elemente der Semantischen Instanzsegmentierung einzeln analysieren kann, sodass man feststellen konnte, dass der Regressionsteil der Aufgabe der Instanzsegmentierung in Form der Detektion bereits gut funktioniert. Neben der Anzahl der vorhandenen Trainingsdaten beeinflusst die Leistung der Prädiktion der pixelweisen semantischen Informationen dabei die Form der Instanzen der Klassen. Vor allem die Prädiktion der relativen Positionen ist für Klassen, die sehr feine Strukturen repräsentieren, wie z.B. Fahrräder, schwierig. Dies ist leicht nachvollziehbar, da es für solche Klassen sehr schwierig ist, das Objektzentrum zu finden sowie allgemein diese pixelweise zu segmentieren. Bei Klassen, wie z.B. Züge oder Lastwagen, deren Instanzen im Verhältnis zu den anderen Klassen eine hohe Größe aufweisen, ergeben sich jedoch ebenfalls Probleme mit der Prädiktion der Semantischen Segmentierung und Relativen Positionsprädiktion. Der Grund hierfür ist in der Größe des Rezeptiven Feldes zu suchen, also dem konstant großen Bereich im Bild, der die Basis für die Klassifikation jedes Pixels im Bild darstellt. Bei sehr großen Instanzen beinhaltet dieses nicht das gesamte Objekt, was es für das Netzwerk schwierig macht, die Pixel der richtigen semantischen Klasse zuzuordnen und insbesondere zu Schwierigkeiten führt, die relative Position in Bezug zum Betrachter zu bestimmen.

Das Kriterium, welches hier verwendet wurde, um auf Basis der Semantischen Segmentierung, der Detektion und der Relativen Positionsprädiktion die Semantische Instanzsegmentierung zu generieren, ist in Kapitel 3.5 beschrieben. Wie aus Kapitel 5.8 hervorgeht, wirkt es sich positiv auf die Ergebnisse aus, wenn das Hintergrundlabel nicht für die Relative Positionsprädiktion mitbestimmt wird. Der Grund hierfür ist darin zu finden, dass innerhalb der prädizierten Bounding-Boxen nur die Pixel betrachtet werden, die dieselbe Klasse haben wie die der Bounding-Box. Wenn man nun zusätzlich noch alle Pixel ausschließt, für die das Hintergrundlabel bei der Relativen Positionsprädiktion bestimmt wurde, so führt dies in den meisten Fällen zu einer Untersegmentierung

der Instanzen. Weiterhin ist zu sagen, dass das hier zunächst verwendete Kriterium nicht zwingend verwendet werden muss, um auf Basis der vorhandenen Informationen das Problem der Instanzsegmentierung zu lösen. Auch hier liegt sicherlich eine Stärke des gewählten Ansatzes. Weitere denkbare Wege wären z.B. die Formulierung eines Graph-Cut Problems oder die Formulierung eines Nearest-Neighbour Klassifikators auf Basis der Relativen Position. So könnte man z.B. bei sich überlappenden Bounding-Boxen derselben Klasse jedes Pixel der Bounding-Box zuordnen, zu dessen relativer Position innerhalb der Bounding-Box die pixelweise relative Position die geringste Distanz aufweist. Wenn keine Überlappung von Bounding-Boxen der gleichen Klasse vorhanden ist, könnte man dann ohne ein weiteres Kriterium anzuwenden, alle Pixel mit der prädizierten Klasse der Bounding-Box übernehmen.

6.3 Nutzen für die Generierung des Szenenmodells des Selbstfahrenden Autos

Für die Anwendung des Netzwerkes im Zusammenhang mit dem Selbstfahrenden Auto muss das Netzwerk zum einen möglichst ressourcenschonend arbeiten und zum anderen gute Ergebnisse liefern. Dabei ist das hier entwickelte Netzwerk nicht nur für die Instanzsegmentierung anwendbar. Wie in den vorherigen Kapiteln beschrieben, werden durch das Netzwerk gleichzeitig eine Semantische Segmentierung und eine Detektion bestimmt. Nun haben sich aus den Experimenten wie in Kapitel 5.5 eine bestimmte Menge an Modellen hervorgetan, die in Bezug auf die Qualität der Ergebnisse oder des Ressourcenverbrauches besonders gut funktionieren. Das Modell, welches das alte ROI-Pooling verwendet und die Semantische Segmentierung sowohl beim RCNN als auch beim RPN einfließen lässt, erreicht bei der Semantischen Segmentierung und der Detektion ungefähr dieselbe Qualität wie die Einzelnetzwerke. Wenn man jedoch von einer sequentiellen Ausführung der Einzelnetzwerke ausgeht, so bietet dieses Netzwerk eine deutlich geringere Ausführungszeit. Aufgrund der Tatsache, dass auf der Hardware des Selbstfahrenden Autos viele Prozesse gleichzeitig ausgeführt werden, so ist eine derartige Einschränkung des Arbeitsspeichers durchaus wahrscheinlich. Bei einer derartigen Ausführung würden die Einzelnetzwerke jedoch eine etwas geringere Auslastung des Arbeitsspeichers bieten. Ausgehend von einer parallelen Ausführung der Netzwerke wird durch dieses Netzwerk wie bei allen hier entwickelten Architekturen eine geringere Auslastung des Arbeitsspeichers erreicht. Allerdings bietet diese Ausführung der Detektion und Semantischen Segmentierung durch die Einzelnetzwerke eine geringere Ausführungszeit. Indem die neue ROI-Pooling Methode eingeführt wird, wird eine weitere Steigerung in der Qualität der Detektionsergebnisse erzielt, welche jedoch mit einer Steigerung der Ausführungszeit einhergeht, die einer sequentiellen Ausführung der Einzelnetzwerke entspricht. Ein weiterer Nebeneffekt ist eine etwas schlechtere Semantische Segmentierung. Im Zusammenhang mit der Fahraufgabe kommt der Detektion von anderen Verkehrsteilnehmern eine größere Bedeutung zu, da diese, anders als die Semantische Segmentierung, auch zwischen Instanzen einer Klasse unterscheidet, welche sich unabhängig voneinander verhalten. Für die Fahraufgabe ist es dabei häufig nicht notwendig, die exakten Umrisse dieser Objekte zu kennen. Aus diesem Grund ist das

Netzwerk, auch wenn die Ergebnisse der Instanzsegmentierung für mache Klassen noch nicht gut genug für das Selbstfahrende Auto sind, dennoch in Form der Ergebnisse der Detektion und Semantischen Segmentierung für den Einsatz im Auto geeignet. Indem man nämlich die Semantische Segmentierung als Informationsgrundlage für die Klassen ohne Instanzen wie z.B. die Straße nimmt und durch die Detektion Informationen über die Position und Klasse der anderen Verkehrsteilnehmer generiert, ist das Auto in den meisten Fällen schon in der Lage, sich durch den Verkehr zu navigieren. Die Semantische Instanzsegmentierung könnte dann z.B. im Nahbereich des Autos zum Einsatz kommen, in der eine genauere Unterscheidung zwischen den Instanzen wichtiger wird. Bei solchen sehr gut sichtbaren Instanzen bietet die Instanzsegmentierung auch schon häufig gute Ergebnisse, weshalb man die hier entwickelte Methode hierfür einsetzen könnte.

Kapitel 7

Medizinische Anwendung der Methoden

Das Netzwerk in seiner jetzigen Form, kann auch im Bereich der Verarbeitung von medizinischen Daten eingesetzt werden. Ein Hindernis, das für die direkte Übertragung des Netzwerkes auf medizinische Daten existiert, ist jedoch sicherlich, dass im Bereich der Medizin meist ein Mangel an Trainingsdaten herrscht. Allerdings sind die Aufgaben, die es im Zusammenhang mit der Medizin zu lösen gilt, meist auch wesentlich spezieller als die generelle Erstellung eines Szenenmodells im Straßenverkehr. Daraus folgt zum einen, dass die Daten in Bezug auf die generelle Zusammensetzung des Bildes weniger stark variieren. Bei der Segmentierung von Organen im menschlichen Körper in 3D-CT Datensätzen ist z.B. immer die gleiche Anzahl der Organe vorhanden, die immer eine ähnliche Position im Körper haben. Bei solchen Aufgaben spielt dann eher die Qualität der Segmentierung der einzelnen Objekte eine größere Rolle als im Anwendungsfall des Straßenverkehrs, in welchem eher ein Überblick über die gesamte Szene wichtig ist, welcher wie erläutert bei medizinischen Daten meist schon vorhanden ist.

Die Eigenschaften des Netzwerkes können mithin in der Medizin vornehmlich dort von Vorteil sein, wo a priori wenig Informationen über die wichtigen Informationen der Szene vorhanden sind, wo also z.B. Krankheiten detektiert werden sollen. Einen solchen Anwendungsfall stellt z.B. die Detektion und Segmentierung von Tumoren oder Metastasen in medizinischen Bildern dar. In [CEE+16] wurde die Unteraufgabe der Semantischen Segmentierung darauf verwendet, eine gleichzeitige Segmentierung der Leber und von Leberläsionen in 3D-CT Bilddaten zu ermöglichen. Diese Informationen stellen wichtige Biomarker für das Fortschreiten von Leberkrebs dar. In [MST+13] wurden durch die Semantische Segmentierung Morbus Crohn-Regionen in 3D-MRT Bilddaten detektiert. Für die Anwendung des hier entwickelten Netzwerkes auf dreidimensionale Daten kann jede Voxel-Schicht einzeln dem Netzwerk als Eingabe dienen, was jedoch bedeuten würde, dass die Informationen der dritten Dimension des $3D$-Bilddatensatzes nicht mit in die Prädiktion einbezogen würde. Insofern wäre es für einen solchen Fall sinnvoll die Faltungen und die Poolings auf die dritte Dimension zu erweitern um so alle Informationen des Bildvolumens nutzen zu können.

Ein Beispiel, wo im Bereich der Medizin die Unterscheidung zwischen Instanzen einer Klasse eine Rolle spielt, ist z.B. die Analyse von Zellbildern, welche in Zusammenhang mit der Diagnose von Krebs von Bedeutung ist (siehe [SSzH98]). Hierbei entscheidende Zellmerkmale sind z.B. die Form und Beschaffenheit, welche visuell wahrnehmbar sind und mithin von CNNs verarbeitet werden können. In [RFB15] wurde eine auf der CNNs basierende Methode vorgestellt, welche durch Data-Augmentation auf Basis von wenigen

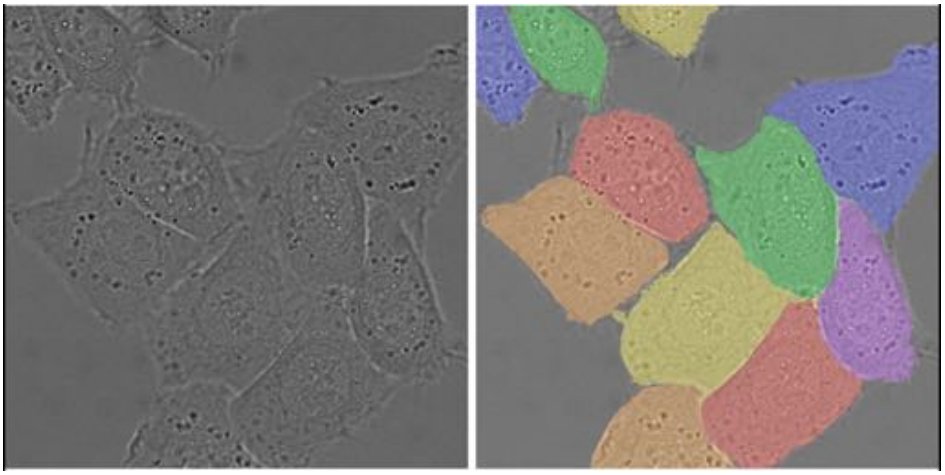

Abbildung 7.1: Das Bild ist entnommen aus [RFB15] und zeigt auf der linken Seite HeLa Zellen, welche durch DIC-Mikroskopie aufgenommen wurden. Auf der rechten Seite ist die manuell annotierte Instanzsegmentierung der Zellen zu sehen.

Trainingsdaten gute Ergebnisse für die Segmentierung von einzelnen Zellen erreicht. Für einen derartigen Anwendungsfall wäre auch die hier entwickelte Instanzsegmentierung geeignet. Bei den Zellen handelt es sich nämlich um Strukturen, deren Form wie in Abbildung 7.1 zu sehen, ähnlich wie die eines Autos, eher konvex sind und deren Instanzen (abhängig vom Bild) meistens nicht größer als das Rezeptive Feld sind. Insofern sollte hier die Prädiktion der pixelweisen semantischen Informationen (ähnlich wie bei der Klasse Auto) gut funktionieren, sodass eine gute Semantische Segmentierung möglich sein sollte und die Instanzen auf Basis der relativen Positionen gut voneinander getrennt werden können.

Die Eigenschaft des geringen Ressourcenverbrauches bezüglich Speicher und Geschwindigkeit ist bei der Anwendung auf medizinische Daten eher von geringer Bedeutung. In diesem Feld handelt es sich bei den meisten Anwendungen um eine Nachverarbeitung von Daten, die also nicht zur gleichen Zeit wie die Aufnahme geschehen muss. Ein Anwendungsfeld in der Medizin, in der die performante Ausführung des Netzwerkes jedoch von Vorteil sein könnte, findet sich in der Verarbeitung von Bilddaten während Operationen. So können durch dieses Netzwerk während der Operation aufgenommene Bilddaten schnell analysiert werden. Bei solchen Bilddaten könnte es sich z.B. um Gewebeschnitte handeln, welche wie oben beschrieben auf die Klassen, Anzahl und Form von Zellen untersucht werden sollen, um z.B. festzustellen, ob ein bestimmtes Gewebe Krebszellen enthält.

Kapitel 8

Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Architektur für die Detektion, die Semantische Segmentierung und die Instanzsegmentierung entwickelt, die auf dem gemeinsamen Training der Aufgaben auf der selben Feature-Map beruhten. Das einfache Training der Detektion und Semantischen Segmentierung auf derselben Feature-Map führte dabei zu einer Verschlechterung der Detektionsleistung. Diese konnte ausgeglichen werden, wenn die Detektion die pixelweisen semantischen Informationen neben der Feature-Map als zusätzliche Eingabe erhalten. Durch die in dieser Arbeit neu erdachte ROI-Pooling Methode konnte zudem die Qualität der Detektion im Verhältnis zum normalen Detektor verbessert werden. Allerdings ergab sich hierdurch eine im Verhältnis zu den Netzwerken ohne das neue ROI-Pooling etwas schlechtere Semantische Segmentierung. Der Ressourcenverbrauch der Methoden ist dabei bei einer sequentiellen Ausführung der einzelnen Detektions- und Segmentierungsnetzwerke bezüglich der Geschwindigkeit geringer, wenn die neue ROI-Pooling Methode nicht verwendet wird und gleich, wenn diese verwendet wird. Bei einer parallelen Ausführung ist die Ausführungszeit der Einzelnetzwerke (für die Detektion und Semantische Segmentierung) geringer, jedoch bieten in diesem Fall alle Modelle mit einer geteilten Feature-Map einen geringeren Speicherverbrauch. Die hier entwickelte Semantische Instanzsegmentierung basiert neben der Detektion und der Semantischen Segmentierung auf der Prädiktion von Relativen Positionen. Die Instanzen einer Klasse werden dann voneinander getrennt, indem die Übereinstimmung der pixelweisen relativen Positionen mit den relativen Postionen innerhalb der Bounding-Boxen bestimmt wird. Die Ergebnisse, die mit dieser Methode erreicht werden, liegen dabei ungefähr im Mittelfeld der Rangliste des Cityscapes Datensatz. Dabei liegen die größten Verbesserungsmöglichkeiten in der Prädiktion der pixelweisen semantischen Informationen, also der Relativen Positionsprädiktion und der Semantischen Segmentierung. Dies kann z.B. geschehen, indem ein besseres Netzwerk als das VGG-16 verwendet wird um die Feature-Map zu generieren. Ein weiterer Ansatzpunkt, die Methode zu verbessern, ist die Wahl eines besseren Kriteriums, das auf Basis der Detektion der Semantischen Segmentierung und der Relativen Positionsprädiktion die Instanzsegmentierung bestimmt.

Für den Einsatz im Selbstfahrenden Auto bieten die hier entwickelten Methoden ein Gesamtsystem für die Detektion, die Semantische Segmentierung und die Instanzsegmentierung, welche eine wichtige Grundlage für die Erstellung eines Szenenmodells darstellen. Dabei bieten vor allem die Detektion und die Semantische Segmentierung eine qualitativ gute Grundlage für die Erstellung eines Szenenmodells. Die Instanzsegmentierung in ihrer jetzigen Form kann im Nahbereich, in der eine genauere Trennung von Instanzen

erst wichtig wird, wichtige Informationen bieten. In diesen Fällen erreicht diese nämlich bereits gute Ergebnisse. Im Falle eines begrenzten Arbeitsspeichers können die Modelle einen Vorteil bezüglich der Ausführungszeit bieten und im Falle einer sehr schnellen Hardware einen Vorteil bezüglich des Speicherverbrauches.

In Kapitel 7 wurde zudem gezeigt, dass die Netzwerkarchitektur auch für die Anwendung auf medizinische Daten nutzbar ist. Als mögliche Anwendungsfelder zeigten sich hier z.B. die Instanzsegmentierung von Zellen oder die Detektion von krankhaft verändertem Gewebe.

Als weiteren Schritt, mit dem eine Verbesserung der pixelweisen semantischen Informationen und der Detektion erreicht werden könnte, ist die Konstruktion von zusätzlichen Abhängigkeiten der Semantischen Segmentierung von der Detektion zu nennen. Indem auch die Semantische Segmentierung die Detektionsergebnisse als zusätzliche Eingabe erhält, würde eine Datengrundlage für die Semantische Segmentierung entstehen, die für jedes Pixel angibt, zu welcher globalen Struktur es gehört. Dadurch würde eine effektive Vergrößerung des Rezeptiven Feldes entstehen und die pixelweise Klassifikation von Klassen mit großen Instanzen würde sich verbessern. Dies in Kombination mit einer besseren Netzwerkarchitektur würde zu einer besseren Semantischen Segmentierung und Relativen Positionsvorhersage führen und mithin die Semantische Segmentierung stark verbessern. Auch die Detektion würde dann durch die besseren pixelweisen semantischen Informationen profitieren. Hierbei ist jedoch eine ressourcenschonende Implementierung wichtig, da eine solche Architektur sonst das Echtzeitkriterium im Zusammenhang mit dem Selbstfahrenden Auto nicht erfüllen würde.

Literaturverzeichnis

[AT17] Arnab, Anurag und Philip H. S. Torr: *Pixelwise Instance Segmentation with a Dynamically Instantiated Network*. CoRR, abs/1704.02386, 2017.

[CEE+16] Christ, Patrick Ferdinand, Mohamed Ezzeldin A. Elshaer, Florian Ettlinger, Sunil Tatavarty, Marc Bickel, Patrick Bilic, Markus Rempfler, Marco Armbruster, Felix Hofmann, Melvin D'Anastasi, Wieland H. Sommer, Seyed-Ahmad Ahmadi und Bjoern H. Menze: *Automatic Liver and Lesion Segmentation in CT Using Cascaded Fully Convolutional Neural Networks and 3D Conditional Random Fields*. CoRR, abs/1610.02177, 2016.

[COR+16] Cordts, Marius, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth und Bernt Schiele: *The Cityscapes Dataset for Semantic Urban Scene Understanding*. CoRR, abs/1604.01685, 2016.

[DLHS16] Dai, Jifeng, Yi Li, Kaiming He und Jian Sun: *R-FCN: Object Detection via Region-based Fully Convolutional Networks*. CoRR, abs/1605.06409, 2016.

[EVGW+a] Everingham, M., L. Van Gool, C. K. I. Williams, J. Winn und A. Zisserman: *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[EVGW+b] Everingham, M., L. Van Gool, C. K. I. Williams, J. Winn und A. Zisserman: *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[GB10] Glorot, Xavier und Yoshua Bengio: *Understanding the difficulty of training deep feedforward neural networks*. In: Teh, Yee Whye und Mike Titterington (Herausgeber): *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Band 9 der Reihe *Proceedings of Machine Learning Research*, Seiten 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[GBC16] Goodfellow, Ian, Yoshua Bengio und Aaron Courville: *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[Gir15] Girshick, Ross: *Fast R-CNN*. In: *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[HAGM14] Hariharan, Bharath, Pablo Arbeláez, Ross Girshick und Jitendra Malik: *Simultaneous Detection and Segmentation*, Seiten 297–312. Springer International Publishing, Cham, 2014.

[HGDG17] He, Kaiming, Georgia Gkioxari, Piotr Dollár und Ross B. Girshick: *Mask R-CNN*. CoRR, abs/1703.06870, 2017.

[HHS16] Hayder, Zeeshan, Xuming He und Mathieu Salzmann: *Shape-aware Instance Segmentation*. CoRR, abs/1612.03129, 2016.

[IMA+16] Iandola, Forrest N., Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally und Kurt Keutzer: *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size*. CoRR, abs/1602.07360, 2016.

[JSD+14] Jia, Yangqing, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama und Trevor Darrell: *Caffe: Convolutional Architecture for Fast Feature Embedding.* arXiv preprint arXiv:1408.5093, 2014.

[KLA+16] Kirillov, Alexander, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy und Carsten Rother: *InstanceCut: from Edges to Instances with MultiCut.* CoRR, abs/1611.08272, 2016.

[LMB+14] Lin, Tsung-Yi, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár und C. Lawrence Zitnick: *Microsoft COCO: Common Objects in Context.* CoRR, abs/1405.0312, 2014.

[LQD+16] Li, Yi, Haozhi Qi, Jifeng Dai, Xiangyang Ji und Yichen Wei: *Fully Convolutional Instance-aware Semantic Segmentation.* CoRR, abs/1611.07709, 2016.

[MST+13] Mahapatra, D., P. J. SchÃ$\frac{1}{4}$ffler, J. A. W. Tielbeek, F. M. Vos und J. M. Buhmann: *Crohn's disease tissue segmentation from abdominal MRI using semantic information and graph cuts.* In: *2013 IEEE 10th International Symposium on Biomedical Imaging*, Seiten 358–361, April 2013.

[RCL+17] Ren, Jimmy S. J., Xiaohao Chen, Jianbo Liu, Wenxiu Sun, Jiahao Pang, Qiong Yan, Yu-Wing Tai und Li Xu: *Accurate Single Stage Detector Using Recurrent Rolling Convolution.* CoRR, abs/1704.05776, 2017.

[RDGF16] Redmon, Joseph, Santosh Divvala, Ross Girshick und Ali Farhadi: *You Only Look Once: Unified, Real-Time Object Detection.* In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[RF16] Redmon, Joseph und Ali Farhadi: *YOLO9000: Better, Faster, Stronger.* CoRR, abs/1612.08242, 2016.

[RFB15] Ronneberger, Olaf, Philipp Fischer und Thomas Brox: *U-Net: Convolutional Networks for Biomedical Image Segmentation*, Seiten 234–241. Springer International Publishing, Cham, 2015.

[RHGS15] Ren, Shaoqing, Kaiming He, Ross B. Girshick und Jian Sun: *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* CoRR, abs/1506.01497, 2015.

[SLD16] Shelhamer, E., J. Long und T. Darrell: *Fully Convolutional Networks for Semantic Segmentation.* IEEE Transactions on Pattern Analysis and Machine Intelligence, PP(99):1–1, 2016.

[SSzH98] Stamatiadis-Smidt, Hilke und Harald zur Hausen: *Mikroskopische Diagnostik*, Seiten 213–217. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[SZ14] Simonyan, Karen und Andrew Zisserman: *Very Deep Convolutional Networks for Large-Scale Image Recognition.* CoRR, abs/1409.1556, 2014.

[UCFB16] Uhrig, Jonas, Marius Cordts, Uwe Franke und Thomas Brox: *Pixel-level Encoding and Depth Layering for Instance-level Semantic Labeling.* CoRR, abs/1604.05096, 2016.

[vdBOM17] Brand, Jan van den, Matthias Ochs und Rudolf Mester: *Instance-Level Segmentation of Vehicles by Deep Contours*, Seiten 477–492. Springer International Publishing, Cham, 2017.

[WIJK16] Wu, Bichen, Forrest N. Iandola, Peter H. Jin und Kurt Keutzer: *SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving.* CoRR, abs/1612.01051, 2016.