Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
Fachgebiet Neuroinformatik und Kognitive Robotik

Intelligentes inkrementelles Lernen für Deep-Learning-Ansätze

Masterarbeit zur Erlangung des akademischen Grades Master of Science

Daniel Seichter

Betreuer: Dipl.-Inf. Markus Eisenbach
Dipl.-Inf. Ronny Stricker
FG Neuroinformatik und Kognitive Robotik

Verantwortlicher Hochschullehrer:
Prof. Dr. H.-M. Groß
FG Neuroinformatik und Kognitive Robotik

Die Masterarbeit wurde am 27.06.2017 bei der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau eingereicht.

Erklärung: „Hiermit versichere ich, dass ich diese Masterarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe. Alle von mir aus anderen Veröffentlichungen übernommenen Passagen sind als solche gekennzeichnet.“

Ilmenau, 27.06.2017 .....

Daniel Seichter

Inhaltsverzeichnis

1 Einleitung **1**

2 Theoretische Grundlagen **5**

2.1 Lernparadigmen 5

2.1.1 Supervised / Semi-Supervised / Unsupervised Learning 6

2.1.2 Incremental Learning 7

2.1.3 Active Learning 8

2.2 Deep-Learning-Techniken 9

2.2.1 Dropout 9

2.2.2 Softmax-Klassifikation 11

2.2.3 Adversarial Training 13

2.3 Bayesian Neural Networks 15

2.3.1 Markov-Chain-Monte-Carlo-Verfahren 16

2.3.2 Variational Inference 17

3 State of the Art **19**

3.1 Übersicht 19

3.2 Modellierung von Unsicherheit 20

3.2.1 Approximierende bayes'sche Ansätze 21

3.2.1.1 Markov-Chain-Monte-Carlo-Verfahren 21

3.2.1.2 Variational Inference 22

3.2.2 Nicht-bayes'sche Ansätze 27

3.2.3 Modifikation der Trainingsdaten 29

3.2.4 Nachtraining des Klassifikators 33
3.2.4.1 Supervised Learning 33
3.2.4.2 Semi-Supervised Learning 34
3.3 Zusammenfassung und Erkenntnisse 36

4 Daten und Vorverarbeitung 39
4.1 Zu unterscheidende Schadklassen 39
4.2 GAPs-Datensatz 40
4.3 Fahrbahnbilder 43
4.3.1 Zusammensetzung aus Einzelbildern 44
4.3.2 Labelinformationen 45
4.4 Vorverarbeitung der Fahrbahnbilder 47
4.4.1 Lokalisierung der Stitching-Kanten 47
4.4.1.1 Algorithmus 48
4.4.1.2 Experimente zur Parameteridentifikation 55
4.4.2 Bestimmung des Labelrasters 58
4.4.2.1 Algorithmus 58
4.4.2.2 Experimente zur Parameteridentifikation 69
4.5 Zusammenfassung 73

5 Intelligentes inkrementelles Lernen 75
5.1 Überblick 75
5.2 Basismodell des Klassifikators 78
5.3 Erweiterung des Klassifikators 79
5.3.1 Dropout-Ansatz 80
5.3.2 Ensemble-Ansatz 80
5.4 Inkrementeller Lernprozess 85
5.4.1 Anwendung auf unbekannte Daten 85
5.4.1.1 Verarbeitung von hochauflösenden Bildern 85
5.4.1.2 Extraktion von Patches 87
5.4.2 Auswahl neuer Trainingsbeispiele und Nachtraining 89

6 Experimentelle Untersuchungen 93

6.1 Eignung für inkrementellen Lernprozess 93

6.1.1 Dropout-Ansatz 95

6.1.2 Ensemble-Ansatz 97

6.2 Berücksichtigung der Unsicherheiten 100

6.2.1 Dropout-Ansatz 102

6.2.2 Ensemble-Ansatz 105

6.3 Anwendung auf Fahrbahnbilder 107

6.3.1 Zufällige Auswahl neuer Trainingsbeispiele 109

6.3.2 Berücksichtigung der Unsicherheiten 111

6.3.3 Zusätzliche Berücksichtigung der Klassifikatorausgabe 117

6.3.4 Ergebnisse für Test-Validierungsdatensatz 120

6.4 Zusammenfassung 122

7 Zusammenfassung und Ausblick 125

7.1 Zusammenfassung 125

7.2 Weiterführende Arbeiten 127

A Ergänzungen zu den Experimenten 129

A.1 Ensemble aus mehreren Netzwerken 129

A.2 Berücksichtigung der Adversarial Examples 131

A.3 Ergänzungen zu Abschnitt 6.3.2 134

A.4 Ergänzungen zu Abschnitt 6.3.3 140

Literaturverzeichnis 147

Abkürzungsverzeichnis

ASINVOS Assistierendes und interaktiv lernfähiges Videoinspektionssystem für Oberflächenstrukturen am Beispiel von Straßenbelägen und Rohrleitungen

AE Autoencoder

BCNN Bayesian Convolutional Neural Network

CNN Convolutional Neural Network

CPU Central Processing Unit

DL Deep Learning

FFT Fast Fourier Transform

GAPs German Asphalt Pavement Distress

GPU Graphics Processing Unit

HITL Human-in-the-loop

KL Kullback-Leibler-Divergenz

MCMC Markov Chain Monte Carlo

NIN Network in Network

RAL Richtlinien zur Anlage von Landstraßen

SGD Stochastic Gradient Descent

SVM Support Vector Maschine

ZEB Zustandserfassung und -bewertung von Fernstraßen

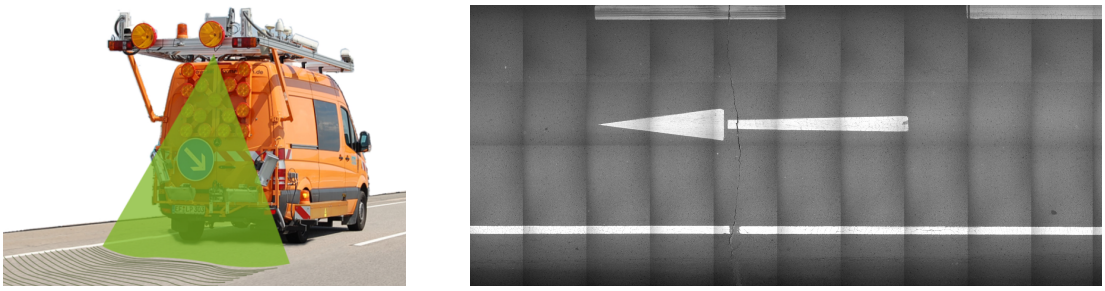
Kapitel 1

Einleitung

Deep Learning dominiert aktuell nahezu alle Bereiche des maschinellen Lernens. Deep-Learning-basierte Klassifikatoren übertreffen dabei teilweise menschliche Leistungen und erzielen Fortschritte, die lange noch nicht zu erwarten waren. Die erfolgreiche Anwendung setzt jedoch die Verfügbarkeit von umfangreichen und korrekt gelabelten Datensätzen voraus. Je umfassender und aussagekräftiger die präsentierten Beispiele sind, desto höher ist in vielen Fällen die letztliche Generalisierungsfähigkeit der trainierten Klassifikatoren. Große Datenbestände bilden somit einen wichtigen Grundstein für den Erfolg von Deep Learning in unterschiedlichsten Anwendungsbereichen.

Gleichermaßen bedeutet dies aber auch, dass die Anwendung von Deep Learning in abweichenden Einsatzfeldern nur eingeschränkt möglich ist. Oft existieren zwar große Datenbestände, jedoch ist nur ein Bruchteil dieser Daten mit Labelinformationen versehen und somit für das Training eines Klassifikators geeignet. Diese Tatsache ist hohen Labelkosten geschuldet. Die Erhebung der Labelinformationen ist oft sehr zeitaufwendig und kann zum Teil nur von menschlichen Operatoren durchgeführt werden. Für den Lernprozess bedeutet dies, dass die verfügbaren gelabelten Daten meist nicht alle Facetten der zugrunde liegenden Problemstellung repräsentieren. Bei der Anwendung auf unbekannte Daten kann dies vermehrt zu Fehlklassifikationen führen. Um diesem Problem begegnen zu können, müssen inkrementelle Lernverfahren in Betracht gezogen werden. Im Anschluss an eine initiale Lernphase mit den verfügbaren gelabelten Daten, soll der Klassifikator dabei schrittweise verbessert werden. Hierzu soll das System

möglichst selbstständig entscheiden, welche der unbekannten Datenpunkte informativ sind. Anschließend werden die Labelinformationen für die ausgewählten Beispiele bestimmt, der Trainingsdatensatz erweitert und der Klassifikator erneut trainiert. Durch diese iterative Vorgehensweise soll eine stetige Verbesserung erzielt werden.
Am Fachgebiet Neuroinformatik und Kognitive Robotik wird aktuell ein solcher Anwendungsfall von Deep Learning betrachtet. Im Rahmen des ASINVOS-Projektes[1] wird unter anderem an der visuellen Erkennung und Lokalisierung von Schadstellen auf Fernstraßen geforscht. Ziel ist es, das bereits seit vielen Jahren in Deutschland eingesetzte Verfahren zur Zustandserfassung und -bewertung (ZEB) von Fernstraßen durch den Einsatz von maschinellem Lernen zu verbessern. Der Zustand der Fahrbahnoberfläche wird dabei in regelmäßigen Abständen mit einem speziell ausgestatteten Fahrzeug aufgenommen. Zwei hochauflösende Kameras fertigen fortlaufend Einzelbilder an, welche anschließend zu kompletten Fahrbahnbildern zusammengesetzt werden. Jedes Bild repräsentiert hierbei einen zehn Meter langen Fahrbahnabschnitt. Abbildung 1.1 zeigt das verwendete Fahrzeug und ein Beispiel für ein aufgenommenes Fahrbahnbild.



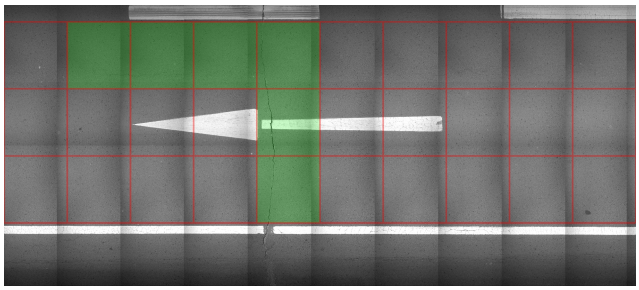
(a) Aufnahmefahrzeug **(b)** Zusammengesetztes Bild der Fahrbahn

Abbildung 1.1: Datenaufnahme im ASINVOS-Projekt
Dargestellt ist das Fahrzeug zur visuellen Aufnahme der Fahrbahnoberfläche und ein aus den Einzelaufnahmen zusammengesetztes Fahrbahnbild.

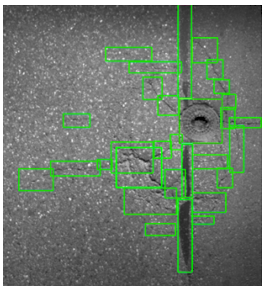
Während die Fahrbahn auf diese Weise verhältnismäßig schnell erfasst werden kann, ist die Beurteilung der Beschaffenheit sehr viel zeitintensiver und kann bisher nur von menschlichen Operatoren durchgeführt werden. Da eine pixelgenaue Begutachtung eine

[1] ASINVOS - Assistierendes und interaktiv lernfähiges Videoinspektionssystem für Oberflächenstrukturen am Beispiel von Straßenbelägen und Rohrleitungen

regelmäßige Zustandserfassung unmöglich machen würde, wird die Fahrbahn in jedem Bild in ein Raster, bestehend aus drei Zeilen und zehn Spalten, untergliedert. Jedes Rasterelement beschreibt folglich einen circa einen Meter langen Fahrbahnausschnitt, für welchen letztlich ein oder mehrere Label zur Bewertung des Zustandes vergeben werden. Abbildung 1.2a verdeutlicht die beschriebenen Labelinformationen für das zuvor in Abbildung 1.1b gezeigte Fahrbahnbild.



(a) Fahrbahn mit überlagerten Labelinformationen



(b) GAPs-Datensatz

Abbildung 1.2: Datengrundlage im ASINVOS-Projekt

*Dargestellt sind die Labelinformationen für den bereits in Abbildung 1.1b gezeigten Fahrbahnabschnitt. Das beschriebene Raster ist hierbei rot hervorgehoben. Der Fahrbahnabschnitt zeigt in allen drei Elementen der fünften Spalte einen sichtbaren Riss. Zusätzlich befinden sich im linken Teil der ersten Zeile weitere kleinere, kaum erkennbare Risse im Straßenbelag. Für den gesamten Fahrbahnbereich dieser Rasterelemente wird daher das Label für einen Schaden vergeben (grün hervorgehoben). Weiterhin ist ein Beispielbild aus dem GAPs-Datensatz zur Veranschaulichung der in [*Eisenbach *et al., 2017] angewendeten Labelvergabe dargestellt.*

Während der Zustand ganzer Straßenabschnitte auf diese Weise gut abgeschätzt werden kann, wird die Realisierung eines Klassifikators deutlich erschwert. Für das Training eines Neuronalen Netzes zur Erkennung und Lokalisierung von Schadstellen sind die verfügbaren Labelinformationen zu grob. Am Fachgebiet wurde daher zunächst ein neuer Datensatz erstellt. Der in [Eisenbach et al., 2017] publizierte GAPs-Datensatz[2] enthält, wie in Abbildung 1.2b veranschaulicht, exaktere Markierungen der Schadstellen und ermöglicht somit das Training eines bildausschnittbasierten Klassifikators.

[2] German Asphalt Pavement Distress (GAPs) Datensatz

Eine in [Eisenbach et al., 2017][3] durchgeführte systematische Evaluation verschiedener Netzwerkstrukturen hat jedoch gezeigt, dass bei der Anwendung auf unbekannte Daten deutliche Defizite bestehen. Dies ist vor allem der Tatsache geschuldet, dass die genau gelabelten Daten lediglich einen Bruchteil der verfügbaren Fahrbahnbilder umfassen. Somit besteht die Gefahr, dass die verwendeten Daten nicht alle Facetten der zugrunde liegenden Problemstellung repräsentieren.

In dieser Masterarbeit soll daher untersucht werden, ob die verfügbaren großflächig gelabelten Fahrbahnbilder für einen anschließenden inkrementellen Lernprozess genutzt werden können. Hierzu soll zunächst nach geeigneten Methoden zur Erkennung von Unsicherheit in den Entscheidungen des Klassifikators recherchiert werden. Aufbauend auf einem Modell für die Unsicherheit soll der Trainingsdatensatz anschließend systematisch erweitert werden. Neue Beispiele sollen dabei möglichst automatisiert ausgewählt werden. Ein Operator soll nur dann eingebunden werden, wenn die schadhaften Stellen durch den Klassifikator nicht lokalisiert werden können.

Im nachfolgenden Kapitel werden zunächst wichtige theoretische Grundlagen erläutert. Anschließend werden in Kapitel 3 bestehende Ansätze zur Extraktion von Unsicherheitsinformationen und zur Realisierung von inkrementellen Lernprozessen analysiert. In Kapitel 4 erfolgt die Betrachtung der in dieser Masterarbeit verwendeten Daten. Der Fokus liegt dabei auf dem Erstellungs- und Labelprozess der Fahrbahnbilder und daraus resultierenden notwendigen Vorverarbeitungsschritten. Aufbauend auf den Erkenntnissen der State-of-the-Art-Betrachtungen und den vorverarbeiteten Fahrbahnbildern wird in Kapitel 5 ein Gesamtsystem zur Realisierung des angestrebten intelligenten inkrementellen Lernprozesses konstruiert. Kapitel 6 umfasst die in dieser Masterarbeit durchgeführten Experimente. Neben Untersuchungen zu wichtigen Teilkomponenten des konstruierten Systems wird hierbei ebenfalls die Anwendung auf die verfügbaren Fahrbahnbilder betrachtet. Abschließend werden in Kapitel 7 die gewonnenen Erkenntnisse zusammengefasst und ein Ausblick auf zukünftige Arbeiten gegeben.

[3] Die Publikation enthält eigene Beiträge zur Bestimmung der verwendeten Netzwerktopologien und zur Festlegung der genutzten Lernparameter.

Kapitel 2

Theoretische Grundlagen

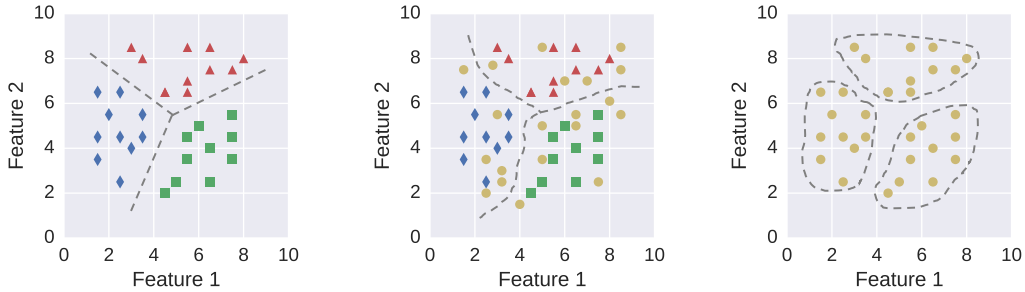
In diesem Kapitel erfolgt die Erläuterung von wichtigen theoretischen Grundlagen. Zunächst werden wesentliche Begrifflichkeiten für Lernparadigmen aus dem Bereich des maschinellen Lernens abgegrenzt und deren Kernaspekte zusammengefasst. Anschließend werden die für diese Masterarbeit relevanten Deep-Learning-Techniken vorgestellt. Das Kapitel schließt mit grundlegenden Betrachtungen zu Bayes'schen Neuronalen Netzen. Hierbei werden wichtige Unterschiede zu herkömmlichen Neuronalen Netzen herausgearbeitet und Verfahrensklassen zur Realisierung erläutert. Die Ausführungen in diesem Kapitel dienen dem Verständnis der anschließenden Abschnitte und können als eine Art Nachschlagewerk angesehen werden.

2.1 Lernparadigmen

Im Bereich des maschinellen Lernens werden viele unterschiedliche Bezeichnungen für die realisierten Lernprozesse unterschieden. Im Folgenden werden die für diese Masterarbeit relevanten Begrifflichkeiten eingeordnet und kurz zusammengefasst. Die Erläuterungen erfolgen hierbei ausschließlich im Hinblick auf Klassifikationsaufgaben und mit Bezug zu Deep-Learning-relevanter Literatur.

2.1.1 Supervised / Semi-Supervised / Unsupervised Learning

Die Verfahren des maschinellen Lernens können anhand der vorliegenden Daten in drei Hauptparadigmen untergliedert werden. Abbildung 2.1 stellt die Unterschiede dieser drei Paradigmen durch einfache Beispiele grafisch dar.



(a) Supervised **(b)** Semi-Supervised **(c)** Unsupervised

Abbildung 2.1: Grafische Darstellung der drei Hauptlernparadigmen
Dargestellt sind graphische Veranschaulichungen für die drei Hauptlernparadigmen: (a) Supervised Learning, (b) Semi-Supervised Learning und (c) Unsupervised Learning anhand eines Klassifikationsproblems mit den drei Klassen: grünes Rechteck, blauer Diamant und rotes Dreieck. Gelb hervorgehobene Punkte besitzen keine Klassenzugehörigkeit. Während Supervised Learning dem Lernen von korrekten Klassenzugehörigkeiten dient, soll bei Unsupervised Learning lediglich die den Daten zugrundeliegende Verteilung gelernt werden. Im Fall von Semi-Supervised Learning können zusätzliche ungelabelte Datenpunkte dazu beitragen, dass feinere Klassengrenzen gelernt werden und somit die Klassifikationsergebnisse verbessert werden können.

Existieren zu den Eingabedaten zusätzlich Zielinformationen, beispielsweise in Form von Labeln, und werden diese für das Training des Klassifikators genutzt, wird der Lernprozess als überwachtes Lernen (engl. Supervised Learning) bezeichnet. Liegen hingegen nur die Eingabedaten vor, wird von unüberwachtem Lernen (engl. Unsupervised Learning) gesprochen. In diesem Fall reduziert sich das betrachtete Problem oft auf das Einteilen der Eingabedaten in verschiedene Kategorien (engl. Clustering) [Gross, 2016]. In Deep-Learning-basierten Ansätzen werden beide Techniken jedoch oft auch kombiniert [Seichter, 2015b]. So erfolgt häufig zunächst ein unüberwachtes

Vortrainieren (engl. Pretraining) der Netzwerke mit Hilfe einer Vielzahl von ungelabelten Eingabedaten. Anschließend wird das Netzwerk in einem zweiten Schritt durch einen überwachten Lernprozess mit gelabelten Beispielen in einem Klassifikator überführt (engl. Fine-Tuning) [Bengio et al., 2007]. Auf diese Weise können die oft in geringer Anzahl vorliegenden Eingabedaten mit Labeln effektiver genutzt werden.
Neben diesen beiden typischen Lernparadigmen existiert zusätzlich eine Mischform, sogenanntes teil-überwachtes Lernen (engl. Semi-Supervised Learning). In diesem Fall können beide Arten von Eingabedaten gleichzeitig für das Training eines Klassifikators genutzt werden. Ähnlich zu der zuvor beschriebenen schrittweisen Kombination von Unsupervised und Supervised Learning setzt diese Art des Lernens voraus, dass Wissen über die zugrundeliegende Verteilung der Eingabedaten die Klassifikationsleistung verbessern kann. Ist dies nicht der Fall, führt Semi-Supervised Learning zu vergleichbaren Ergebnissen wie Supervised Learning [Chapelle et al., 2006].

2.1.2 Incremental Learning

Herkömmliche Lernverfahren gehen meist davon aus, dass der zum Trainingszeitpunkt vorliegende Datensatz abgeschlossen und aussagekräftig ist. Es wird angenommen, dass die enthaltenen Beispiele sämtliches Wissen umfassen, welches nötig ist, um einen Klassifikator im Rahmen eines einzigen Lernprozesses zu trainieren. Da dies oft nicht der Fall ist, bricht inkrementelles Lernen (engl. Incremental Learning) bewusst mit dieser Annahme. So liegt zum Trainingszeitpunkt oft nur ein Teil der benötigten Daten vor oder das verwendete Lernverfahren erlaubt nur dann ein zuverlässiges Training, wenn nicht alle Daten verwendet werden. Um dennoch zuverlässige Klassifikationen zu ermöglichen, soll der initial trainierte Klassifikator durch nachfolgende Lernprozesse inkrementell verbessert werden. Sobald neue Eingabedaten verfügbar sind, wird ein weiterer Trainingsprozess zur Optimierung des Klassifikators (Fine-Tuning) gestartet [Geng und Smith-Miles, 2009]. Incremental Learning kann daher als eine Form des on-line Lernens (engl. on-line Learning) bezeichnet werden. Während beim klassischen on-line Lernen jedoch davon ausgegangen wird, dass neue Eingabedaten nur für eine begrenzte Zeit verfügbar sind und das Training somit sofort erfolgen muss, arbeiten

viele Deep-Learning-Ansätze mit Puffern von begrenzter oder unbegrenzter Größe. Neue Daten werden darin zunächst zwischengespeichert. Erst, wenn eine bestimmte Anzahl an Beispielen erreicht ist, wird ein neuer Trainingsprozess initiiert [Hasan und Roy-Chowdhury, 2014]. In der Regel werden die neuen Beispiele zudem mit vorherigen Daten kombiniert und beide Teile gemeinsam für das Training genutzt. Weiterhin werden die neuen Daten nach Abschluss des Lernprozesses meist nicht verworfen. Durch diese beiden Maßnahmen soll katastrophales Vergessen [Goodfellow et al., 2013, Kirkpatrick et al., 2016] verhindert werden. Im Kontext von Neuronalen Netzen bezeichnet katastrophales Vergessen das Verdrängen von bisher aufgebautem problemspezifischen Wissen, sobald neue Trainingsbeispiele präsentiert werden. In der Folge ist das Neuronale Netz nicht mehr in der Lage, lange nicht gezeigte Beispiele korrekt zu klassifizieren. Wird der beschriebene inkrementelle Lernprozess kontinuierlich über einen längeren Zeitraum fortgeführt, kann er auch als kontinuierliches Lernen (engl. Continues Learning) bezeichnet werden.

2.1.3 Active Learning

Aktives Lernen (engl. Active Learning) kann als eine spezielle Realisierung von inkrementellem Lernen angesehen werden und basiert auf der Annahme, dass ein System bessere Klassifikationsergebnisse mit weniger Daten erzielen kann, wenn es selbst entscheiden kann, welche Daten als nächstes in den Trainingsdatensatz aufgenommen werden [Settles, 2010]. Active Learning geht daher meist von einem sehr kleinen initialen Trainingsdatensatz aus und verwendet anschließend ein interaktives inkrementelles Lernen um die Klassifikationsleistung schrittweise zu verbessern. Der bisherige Klassifikator wird dabei zunächst auf unbekannte Daten angewendet. Im Anschluss entscheidet das System auf Basis einer Akquirierungsfunktion, welche Daten möglichst informativ sind und in den Datensatz aufgenommen werden sollen. Die ausgewählten Beispiele werden im letzten Schritt einem sogenannten Orakel präsentiert. In der Regel ist dies ein Mensch, welcher mit hoher Sicherheit die korrekten Label für die unbekannten Daten zuordnen kann.

Viele der beschriebenen Kernaspekte von Active Learning verstoßen gegen wichtige Voraussetzungen für die Anwendung von Deep-Learning-Techniken. So besitzen Deep-Learning-basierte Klassifikatoren meist eine Vielzahl an zu lernenden Gewichten und benötigen daher für die korrekte Bestimmung enorm große Datenmengen. Ein sehr kleiner initialer Datensatz und die manuelle Bestimmung von Labeln für neue Daten sind daher undenkbar. Dennoch kann die iterative Vorgehensweise auf diese Ansätze übertragen werden. Im Rahmen der State-of-the-Art-Analyse in Kapitel 3 wird Active Learning daher erneut aufgegriffen.

Die in dieser Arbeit betrachteten Lernprozesse basieren hauptsächlich auf Supervised Learning. Optional soll ein vielversprechendes Verfahren auf Basis von Semi-Supervised Learning untersucht werden. Beide Ansätze nutzen Techniken aus dem Bereich Active Learning, um ein intelligentes inkrementelles Lernen zu realisieren.

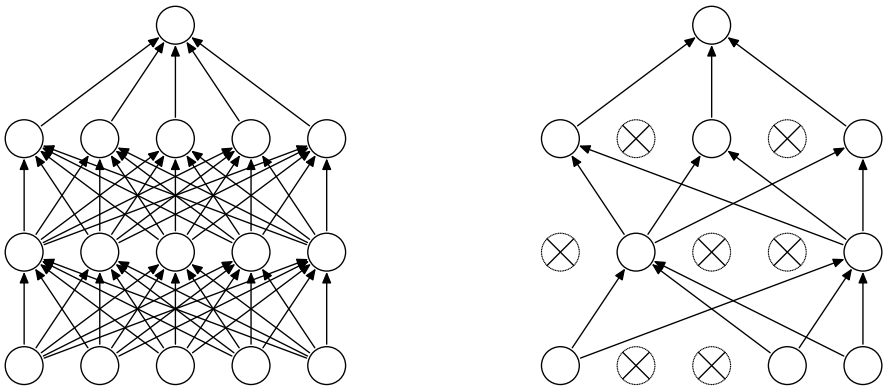
2.2 Deep-Learning-Techniken

Aktuell dominieren Deep-Learning-basierte Ansätze nahezu alle Bereiche des maschinellen Lernens. Klassifikatoren auf Basis von Convolutional Neural Networks (CNN, dt. „faltende“ Neuronale Netze) [Fukushima, 1980, LeCun et al., 1989] bilden den aktuellen State of the Art in vielen Bereichen der Bildverarbeitung. Die in dieser Arbeit verwendeten Klassifikatoren basieren aus diesem Grund ebenfalls auf Convolutional Neural Networks. Für ausführliche deutschsprachige Erläuterungen zur Funktionsweise von Convolutional Neural Network sei an dieser Stelle auf [Seichter, 2015a, Seichter, 2016] verwiesen. In diesem Abschnitt wird ausschließlich auf ausgewählte Techniken eingegangen, die für diese Masterarbeit von besonderer Relevanz sind.

2.2.1 Dropout

Dropout [Hinton et al., 2012, Srivastava et al., 2014] ist eine einfache, aber sehr effektive Methode zur Vermeidung von Koadaptionen mehrerer Neuronen in einem Neuronalen Netz. Infolgedessen kann „Auswendiglernen“ (engl. Overfitting) unterbun-

den und die Generalisierungsfähigkeit verbessert werden. Beim Einsatz von Dropout wird zwischen der Trainings- und der Anwendungsphase unterschieden. Während der Trainingsphase werden in jedem Lernschritt einzelne Neuronen entfernt und das Netzwerk ohne diese trainiert. In der Regel werden für jede Gruppe von Beispielen (engl. Mini-Batches) andere Neuronen entfernt. Abbildung 2.2 zeigt eine mögliche Veränderung der Netzwerkstruktur durch die Anwendung von Dropout.



(a) Ohne Dropout **(b)** Mit Dropout

Abbildung 2.2: Veränderung der Netzwerkstruktur durch Dropout
*Die linke Abbildung zeigt ein normales vollverschaltetes Neuronales Netz mit zwei Hidden-Schichten. Auf der rechten Seite ist eine mögliche Struktur des gleichen Netzwerks nach der Anwendung von Dropout dargestellt. Neuronen, welche mit einem „×“ gekennzeichnet sind, wurden hierbei aus dem Netzwerk entfernt. [*Srivastava *et al., 2014]*

Mit Hilfe eines Parameters $p^{(l)} \in (0, 1]$ kann bestimmt werden, mit welcher Wahrscheinlichkeit die Neuronen der l-ten Schicht in der Netzwerkstruktur enthalten sind. Ist ein Neuron nicht präsent, so wird dessen Ausgabe auf Null gesetzt. Dadurch werden ebenfalls alle Verbindungen, die von diesem Neuron ausgehen, eliminiert. Folglich entspricht diese Vorgehensweise der Multiplikation der Eingaben einer Schicht mit einer binären Maske oder dem Nullsetzen einzelner Zeilen in der Gewichtsmatrix einer Schicht. Durch die Anwendung von Dropout können während der Trainingsphase bis zu 2^n verschiedene Subnetzwerke betrachtet und gelernt werden, wobei n die Gesamtanzahl der Neuronen im betrachteten Netzwerk ist.

Für die Anwendungsphase schlagen die Autoren in [Srivastava et al., 2014] das Beibehalten aller Neuronen in Kombination mit der Multiplikation der Eingaben der l-ten Schicht mit dem Faktor $p^{(l)}$ vor. Laut [Srivastava et al., 2014] approximiert dies den Prozess der Mittelung über eine Vielzahl von kleineren Netzwerken in nur einem Vorwärtsprozess. Im Rahmen dieser Masterarbeit wird dieser Ansatz als Standard-Bernoulli-Dropout bezeichnet.
Neben dieser Form von Dropout können die Eingaben einer Schicht ebenso mit einer Maske multipliziert werden, deren Elemente aus einer Gaußverteilung gesampelt werden, sogenanntem Gaussian-Dropout [Srivastava et al., 2014]. In der Literatur gibt es inzwischen zudem zahlreiche Erweiterungen dieser Idee. In [Wan et al., 2013] wird gezeigt, dass es ebenfalls möglich ist, anstelle von Neuronen (Spalten der Gewichtsmatrix) einzelne Gewichte (Elemente der Gewichtsmatrix) zu eliminieren, sogenanntem DropConnect. Letztlich wird in [McClure und Kriegeskorte, 2017] die Kombination beider Vorgehensweisen in Form von Spike-and-Slab-Sampling vorgestellt.

Die Anwendung von Dropout hat sich auch für die Klassifikatoren im Rahmen des ASINVOS-Projektes bewährt. Dropout ist daher ein wichtiger Bestandteil der in dieser Masterarbeit verwendeten Netzwerke. In Abschnitt 3.2.1 (Seite 21) wird Dropout erneut aufgegriffen und um eine bayes'sche Interpretation ergänzt, welche eine mögliche theoretische Begründung für den Erfolg von Dropout bietet.

2.2.2 Softmax-Klassifikation

Die Softmax-Regression[1,2] ist neben der Support-Vector-Maschine (SVM) eine der am häufigsten genutzten Strukturen zur Realisierung von Klassifikationen und fester Bestandteil vieler Deep-Learning-Ansätze. Durch das Verallgemeinern der logistischen Regression können $C \in \mathbb{N}\backslash\{1\}$ verschiedene Klassen unterschieden werden. Zudem kann die Softmax-Klassifikation direkt als abschließende vollverschaltete Schicht in ein Neuronales Netz integriert werden.

[1] CS231n: `http://cs231n.github.io/linear-classify/` (Zuletzt aufgerufen: 19.03.2017)
[2] UFLDL: `http://deeplearning.stanford.edu/wiki/index.php/Softmax_Regression` (Zuletzt aufgerufen: 19.03.2017)

Mit Hilfe der Softmax-Funktion kann die Eingabe $\underline{\mathbf{x}}$ auf einen Vektor von Klassenzugehörigkeiten $\underline{\mathbf{y}}$ abgebildet werden. Jedes Element y_i dieses Vektors symbolisiert hierbei die Zugehörigkeit zur Klasse i mit $i \in [0, C-1]$ und lässt sich wie folgt berechnen:

$$y_i = p(y_i = i|\underline{\mathbf{x}}, \theta) := \frac{e^{f(\underline{\mathbf{x}},\theta_i)}}{\sum_{c=0}^{C-1} e^{f(\underline{\mathbf{x}},\theta_c)}} \tag{2.1}$$

Hierbei sei f allgemein die Funktion des Neuronalen Netzes mit den Gewichten θ bis zur abschließenden Softmax-Schicht. Für die Klassenzugehörigkeiten y_i gelten somit die folgenden Bedingungen:

$$y_i \in [0, 1] \quad \text{und} \quad \sum_{c=0}^{C-1} y_i = 1$$

Aus diesem Grund werden die Ausgaben der Softmax-Schicht oft als Zugehörigkeitswahrscheinlichkeiten interpretiert. Es muss jedoch beachtet werden, dass es sich an dieser Stelle um keine echten Wahrscheinlichkeiten und somit auch nicht um ein Maß für die Sicherheit der Klassifikation handelt. Im Folgenden wird diese Problematik näher erläutert.

Während des Lernprozesses wird meist die aus der Informationstheorie bekannte Kreuzentropie als Fehlermaß genutzt. Diese Kreuzentropie $\mathbb{H}(p, q)$ beschreibt die Qualität einer geschätzten Verteilung q im Vergleich zu der wirklichen Verteilung p und ist definiert durch:

$$\mathbb{H}[p, q] = -\sum_i p_i \cdot \log q_i \tag{2.2}$$

Ziel ist es, die Kreuzentropie zwischen den Netzwerkausgaben und der bekannten wirklichen Verteilung zu minimieren. Die wahre Verteilung wird hierbei durch einen Vektor $\underline{\mathbf{t}} \in \{0, 1\}^C$ mit 1-aus-n-Kodierung (engl. One-Hot) repräsentiert. Gehört ein Beispiel zur Klasse $c \in [0, C-1]$, so besitzt lediglich das c-te Element dieses Vektors einen Wert von 1, alle anderen Einträge sind 0. Ausgehend von Formel (2.2) lässt sich der während des Trainings des Netzwerks zu minimierende Fehler E formulieren als:

$$E(\underline{\mathbf{x}}, \underline{\mathbf{t}}, \theta) = -\sum_{i=0}^{C-1} t_i \cdot \log y_i(\underline{\mathbf{x}}, \theta) \tag{2.3}$$

Die Minimierung dieses Zielfunktionals, beispielsweise durch Gradientenabstiegsverfahren wie Stochastic Gradient Descent (SGD), führt zu Maximum-a-posteriori-Punktschätzungen für die Netzwerkgewichte θ. In der Folge kann keine Aussage zur Sicherheit einer Klassifikation getroffen werden. In Abschnitt 2.3 werden daher bayes'sche Erweiterungen für Neuronale Netze betrachtet, welche das Ableiten von Unsicherheitsinformationen ermöglichen.

2.2.3 Adversarial Training

In [Szegedy et al., 2014] wird gezeigt, dass sowohl flache als auch tiefe Netzwerkstrukturen bereits von leicht veränderten Eingabedaten getäuscht werden können, sodass es zu falschen Klassifikationen kommen kann. Oft genügt es dabei, die für das Training verwendeten Daten so minimal zu verändern, dass der Unterschied für einen Menschen kaum erkennbar ist. Derartige Beispiele werden in der Literatur als Adversarial Examples (dt. konfliktreiche Beispiele) bezeichnet. Abbildung 2.3 verdeutlicht den Zusammenhang anhand der Anwendung des GoogLeNet [Szegedy et al., 2015] auf ein Beispielbild des ImageNet-Datensatzes [Deng et al., 2009].

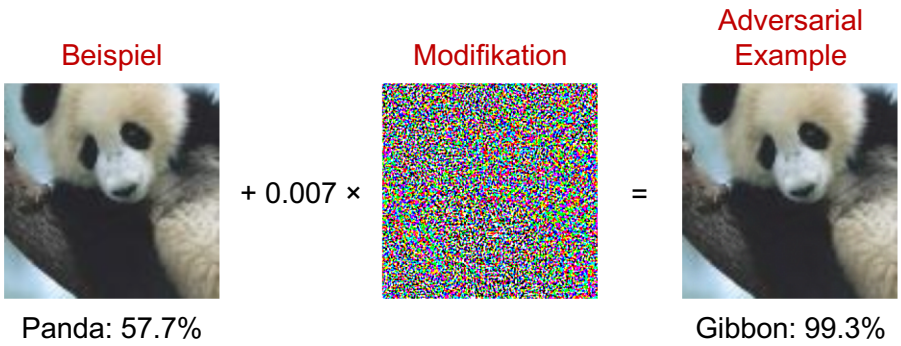


Abbildung 2.3: Beispiel für ein Adversarial Example

Dargestellt ist die Erzeugung eines Adversarial Examples für ein Beispielbild aus dem ImageNet-Datensatz. Durch eine minimale Veränderung kommt es bei der Klassifizierung durch das GoogLeNet zu einer Fehlklassifikation mit hoher Softmax-Ausgabe. [Goodfellow et al., 2015]

Ziel von Adversarial Training ist es, das Netzwerk robust gegen diese Art von Beispielen zu machen und somit die Generalisierungseigenschaften zu verbessern. Adversarial Training kann daher neben Dropout als eine weitere wichtige Form der Regularisierung angesehen werden.

In [Goodfellow et al., 2015] wird eine einfache, aber sehr effektive Methode zur Generierung von Adversarial Examples beschrieben. So kann ein Adversarial Example $\underline{\tilde{\mathbf{x}}}$ auf Basis eines vorhandenen Beispiels $\underline{\mathbf{x}}$ durch pixelweise Veränderung in Richtung der Gradienten $\nabla_{\underline{\mathbf{x}}} E(\underline{\mathbf{x}}, \underline{\mathbf{t}}, \theta)$ erzeugt werden (engl. Fast Gradient Sign Method)[3]:

$$\underline{\tilde{\mathbf{x}}} = \underline{\mathbf{x}} + \epsilon \operatorname{sign}\left(\nabla_{\underline{\mathbf{x}}} E(\underline{\mathbf{x}}, \underline{\mathbf{t}}, \theta)\right) \tag{2.4}$$

Hierbei bezeichnet $E(\underline{\mathbf{x}}, \underline{\mathbf{t}}, \theta)$ den Fehler, der bei der Verarbeitung der Eingabe $\underline{\mathbf{x}}$ durch ein Netzwerk mit den Gewichten θ unter Berücksichtigung des korrekten Labelvektors $\underline{\mathbf{t}}$ entsteht. Der Parameter ϵ definiert den Grad der Veränderung und sollte im Rahmen von Experimenten bestimmt werden.

Zur anschließenden Berücksichtigung während des Lernprozesses schlagen die Autoren in [Goodfellow et al., 2015] die folgende Modifikation des Zielfunktionals zu einer zweigeteilten Kostenfunktion vor:

$$E_{ges}(\underline{\mathbf{x}}, \underline{\mathbf{t}}, \theta) = \alpha\, E(\underline{\mathbf{x}}, \underline{\mathbf{t}}, \theta) + (1 - \alpha)\, E(\underline{\tilde{\mathbf{x}}}, \underline{\mathbf{t}}, \theta) \tag{2.5}$$

Hierbei beschreibt $E_{ges}(\cdot)$ die zu minimierenden Gesamtkosten. Der Parameter α ermöglicht die Festlegung einer Gewichtung beider Teilkosten. In [Goodfellow et al., 2015] wird hierfür der Wert 0,5 vorgeschlagen.

Im Vorfeld der Masterarbeit hat sich gezeigt, dass durch den Einsatz von Adversarial Training deutlich höhere Klassifikationsraten erzielt werden können. Aus diesem Grund soll in dieser Arbeit ebenfalls die Eignung für ein inkrementelles Lernverfahren untersucht werden.

[3] In einer Implementierung sollte zudem sichergestellt werden, dass $\underline{\tilde{\mathbf{x}}}$ nicht den festgelegten Eingabewertebereich verlässt. Weiterhin muss der Parameter ϵ unter Berücksichtigung dieses Wertebereiches gewählt werden.

2.3 Bayesian Neural Networks

Bayes'sche Neuronale Netze (engl. Bayesian Neural Networks) erweitern herkömmliche Neuronale Netze um wahrscheinlichkeitsorientierte Aspekte [Neal, 2012]. Erste Arbeiten in diesem Bereich gehen bis in die 90er Jahre zurück [MacKay, 1992]. Für umfassende Betrachtungen zur Geschichte und Entwicklung von Bayesian Neural Networks sei auf [Gal, 2016] verwiesen. Die Erläuterungen im Rahmen dieses Abschnitts beschränken sich auf die wichtigsten theoretischen Aspekte.

Bayesian Neural Networks zeichnen sich vor allem durch ihre Robustheit gegenüber Auswendiglernen (engl. Overfitting) aus und können bereits mit kleinen Datenmengen zielführend trainiert werden. Durch ihren wahrscheinlichkeitsorientierten Charakter ermöglichen sie zudem die Angabe von echten Klassifikationsunsicherheiten (im Vergleich zur Interpretation der Softmax-Ausgabe in Abschnitt 2.2.2, Seite 11). Hierzu werden die Gewichte θ eines Netzwerks in Form von Wahrscheinlichkeitsverteilungen definiert[4]. Während des Lernprozesses soll, ausgehend von einer A-priori-Verteilung über den Gewichten $p(\theta)$, auf die A-posteriori-Verteilung $p(\theta|\underline{\mathbf{X}}, \underline{\mathbf{T}})$ nach Betrachten der Eingabedaten $\underline{\mathbf{X}}$ mit den Labeln[5] $\underline{\mathbf{T}}$ geschlossen werden. Dieser Prozess erfolgt auf Basis der bayes'schen Formel:

$$p(\theta|\underline{\mathbf{X}}, \underline{\mathbf{T}}) = \frac{p(\underline{\mathbf{T}}|\underline{\mathbf{X}}, \theta)\, p(\theta)}{p(\underline{\mathbf{T}}|\underline{\mathbf{X}})} \tag{2.6}$$

Hierbei ist $p(\underline{\mathbf{T}}|\underline{\mathbf{X}}, \theta)$ die Wahrscheinlichkeit dafür, dass die Daten beobachtet werden, falls θ die korrekten Gewichte wären. Ferner beschreibt $p(\underline{\mathbf{T}}|\underline{\mathbf{X}})$ die Wahrscheinlichkeit dafür, dass die Daten durch das aktuelle Model erzeugt wurden. In der Literatur wird letztere oft als Evidenz (engl. Evidence) bezeichnet. Die anhand von Formel (2.6) bestimmte Verteilung über den Gewichten ermöglicht die Bestimmung der Klassenzugehörigkeit y für ein unbekanntes Beispiel $\underline{\mathbf{x}}^*$:

$$p(y = c|\underline{\mathbf{x}}^*, \underline{\mathbf{X}}, \underline{\mathbf{T}}) = \int p(y = c|\underline{\mathbf{x}}^*, \theta) p(\theta|\underline{\mathbf{X}}, \underline{\mathbf{T}})\, d\theta \tag{2.7}$$

[4] Oft werden nur die Gewichte in den Gewichtsmatrizen auf diese Weise definiert. Bias-Gewichte werden wegen ihrer geringeren Dimensionalität meist weiterhin als Punktschätzungen festgelegt.

[5] Die Labelangabe erfolgt in 1-aus-n-Kodierung (engl. One-Hot).

Für Klassifikationsaufgaben kann $p(y = c|\underline{\mathbf{x}}^*, \theta)$ hierbei beispielsweise zur Softmax-Funktion aus Abschnitt 2.2.2 (Seite 11) gewählt werden. Während Bayesian Neural Networks einfach zu beschreiben sind, ist die Realisierung mit vielen Herausforderungen verbunden. Das Hauptproblem ist dabei die Bestimmung der Evidenz:

$$p(\underline{\mathbf{T}}|\underline{\mathbf{X}}) = \int p(\underline{\mathbf{T}}|\underline{\mathbf{X}}, \theta)p(\theta)\, d\theta \tag{2.8}$$

Die in Formel (2.8) dargestellte Berechnungsvorschrift beinhaltet die Integration über alle Unsicherheiten in den Gewichten (Marginalisierung). Während für einfache einschichtige Netzwerke eine analytische Lösung für dieses Integral angegeben werden kann, ist die Bestimmung für komplexere Strukturen nicht möglich. Aus diesem Grund existieren in der Literatur verschiedene Ansätze zur Lösung dieses Problems. Im Folgenden werden zwei typische Verfahrensklassen beschrieben.

2.3.1 Markov-Chain-Monte-Carlo-Verfahren

Markov-Ketten-Monte-Carlo-Verfahren (engl. Markov Chain Monte Carlo)[Hastings, 1970, Gelfand und Smith, 1990] verfolgen das Ziel, die gesuchte A-posteriori-Verteilung $p(\theta|\underline{\mathbf{X}}, \underline{\mathbf{T}})$ mit Hilfe von Stichproben (engl. Samples) zu approximieren. Hierzu wird zunächst eine Markov-Kette konstruiert, deren stationäre Verteilung der zu approximierenden Verteilung entspricht. Anschließend können Stichproben gezogen und anhand dieser die gesuchte A-posteriori-Verteilung bestimmt werden. Aus dieser Vorgehensweise ergibt sich jedoch auch ein entscheidender Nachteil. So werden bei komplexen Netzwerken oder großen Datensätzen zum Teil sehr viele Schritte benötigt, bis die konstruierte Markov-Kette in ihrer stationären Verteilung konvergiert. Die genaue Anzahl der notwendigen Schritte ist dabei stets unbekannt [Blei et al., 2016]. Aus diesem Grund spielen Markov-Ketten-Monte-Carlo-Verfahren für Deep-Learning-Ansätze eher eine untergeordnete Rolle. Für weiterführende Literatur zu dieser Verfahrensklasse sei auf [Gal, 2016] verwiesen.

2.3.2 Variational Inference

Variational Inference (dt. Inferenz durch Variation) [Jordan et al., 1999] bildet eine Alternative zu Markov-Chain-Monte-Carlo-Verfahren. Ziel von Verfahren dieser Art ist es, die Berechnung der wahren A-posteriori-Verteilung $p(\theta|\underline{\mathbf{X}}, \underline{\mathbf{T}})$ durch eine Überführung in ein Optimierungsproblem zu umgehen. Hierzu wird eine approximierende Verteilung $q_\phi(\theta)$ mit den Parametern ϕ und einer einfacher zu evaluierenden Struktur definiert. Im Rahmen der Optimierung wird anschließend, ähnlich wie bei der Berechnung der Kreuzentropie in herkömmlichen Neuronalen Netzen[6] (Abschnitt 2.2.2, Seite 11), die Kullback-Leibler-Divergenz (KL) als Maß für Unterschiedlichkeit der beiden Verteilungen minimiert:

$$\begin{aligned} q_\phi^*(\theta) &= \arg\min_\phi \{KL(q_\phi(\theta) \,||\, p(\theta|\underline{\mathbf{X}}, \underline{\mathbf{T}}))\} \\ &= \arg\min_\phi \left\{ \int q_\phi(\theta) \log \frac{q_\phi(\theta)}{p(\theta|\underline{\mathbf{X}}, \underline{\mathbf{T}})} \, d\theta \right\} \end{aligned} \tag{2.9}$$

Hierbei ist $q_\phi^*(\theta)$ die approximierte A-posteriori-Verteilung. Da die in Formel (2.9) beschriebene Optimierung ebenfalls die Berechnung der Evidenz $p(\underline{\mathbf{T}}|\underline{\mathbf{X}})$ als Bestandteil von $p(\theta|\underline{\mathbf{X}}, \underline{\mathbf{T}})$ beinhaltet, wird bei der Optimierung auf die äquivalente Maximierung der sogenannten unteren Grenze der Evidenz (engl. Evidence Lower Bound)[7] zurückgegriffen:

$$q_\phi^*(\theta) = \arg\max_\phi \left\{ \int q_\phi(\theta) \log p(\underline{\mathbf{T}}|\underline{\mathbf{X}}, \theta) \, d\theta - KL(q_\phi(\theta) \,||\, p(\theta)) \right\} \tag{2.10}$$

Während die Maximierung des ersten Terms aus Formel (2.10) dafür sorgt, dass $q_\phi(\theta)$ die Daten möglichst gut erklärt, dient die Minimierung des zweiten Terms dazu, dass $q_\phi(\theta)$ möglichst ähnlich zur vorher bestimmten A-priori-Verteilung $p(\theta)$ wird.

Durch die Interpretation als Optimierungsproblem können Bayesian Neural Networks auch in Verbindung mit großen Datensätzen, wie sie im Deep-Learning-Bereich üblich

[6] Die Berechnungsvorschrift für die Kreuzentropie aus Abschnitt 2.2.2 (Seite 11) enthält neben der umgedrehten Kullback-Leibler-Divergenz $KL(p(\cdot) \,||\, q(\cdot))$ einen zusätzlichen Summanden für die Entropie von $p(\cdot)$.

[7] Anschauliche Erläuterung der Dualität: `https://hips.seas.harvard.edu/blog/2013/03/22/variational-inference-part-1/` (Zuletzt aufgerufen: 27.03.2017)

sind, trainiert werden. Die Bestimmung des ersten Terms in Formel (2.10) beinhaltet jedoch in der Regel Berechnungen über den kompletten Datensatz oder kann für sehr komplexe Netzwerkstrukturen wiederum nicht analytisch angegeben werden. Im Gegensatz zu Markov-Ketten-Monte-Carlo-Verfahren kann zudem keine exakte Lösung garantiert werden. So ist es möglich, dass die Maximierung in einem lokalen Maximum endet. Die genannten Gründe führen dazu, dass Anwendungen in Verbindung mit Deep Learning bisher eher untypisch und weniger populär sind. [Gal, 2016, Blei et al., 2016]

In den anschließenden State-of-the-Art-Betrachtungen werden sowohl Markov-Chain-Monte-Carlo-Verfahren als auch Ansätze auf Basis von Variational Inference erneut aufgegriffen und jeder Verfahrensklasse konkrete Realisierungen zur Extraktion von Unsicherheitsinformationen zugeordnet. Der Fokus liegt hierbei auf Publikationen aus dem Deep-Learning-Bereich.

Kapitel 3

State of the Art

Die Realisierung eines Systems zur Umsetzung von inkrementellem Lernen umfasst mehrere aufeinanderfolgende Teilschritte. In diesem Kapitel wird der State of the Art für jeden dieser Schritte betrachtet. Hierzu wird zunächst eine Systematik erarbeitet. Anschließend werden die einzelnen Unterpunkte beschrieben.

3.1 Übersicht

Der untersuchte State of the Art lässt sich anhand der zu realisierenden Schritte untergliedern. Abbildung 3.1 zeigt eine Systematisierung für die anschließenden State-of-the-Art–Betrachtungen.

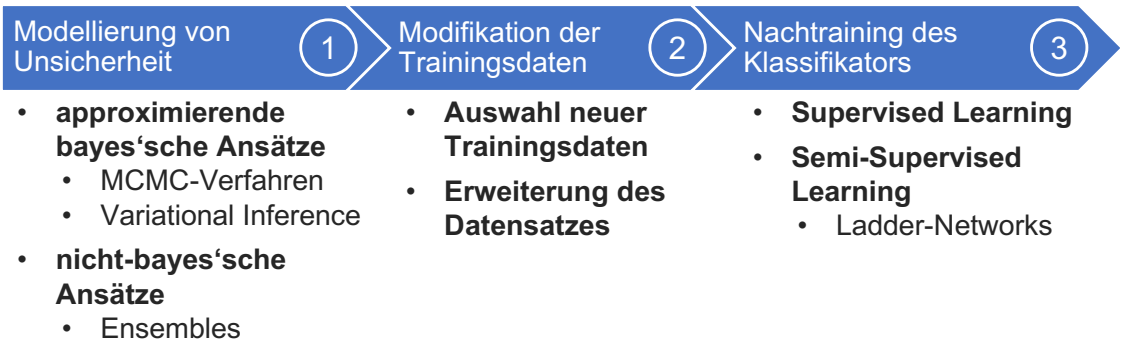


Abbildung 3.1: Überblick über State of the Art

Dargestellt ist die Systematik für die anschließenden State-of-the-Art-Betrachtungen.

Die Realisierung eines Systems zur Umsetzung von inkrementellem Lernen umfasst drei grundlegende Teilschritte. Zunächst müssen dem System unbekannte Daten präsentiert werden. Auf Basis der Klassifikationsergebnisse und der Unsicherheit bei den Klassifikationen können anschließend neue Daten für das Training ausgewählt werden. In den State-of-the-Art-Betrachtungen werden zwei Verfahrensklassen zur Bestimmung von Unsicherheiten untersucht. Hierbei handelt es sich um approximierende bayes'sche Ansätze und nicht-bayes'sche Ansätze. Rein bayes'sche Ansätze zur exakten Angabe von Unsicherheiten können aufgrund der Komplexität von Deep-Learning-Strukturen hingegen nicht angewendet werden[1]. Bei der anschließenden Auswahl neuer Trainingsdaten gilt es, möglichst informative Beispiele zu wählen. Hierfür werden verschiedene Maße auf Basis der bestimmten Unsicherheit vorgestellt und Ansätze zur effektiven Erweiterung der Trainingsdaten besprochen. Im letzten Teilschritt wird der betrachtete Klassifikator mit den angepassten Trainingsdaten erneut trainiert. An dieser Stelle lassen sich die Betrachtungen anhand des verwendeten Lernparadigmas untergliedern. Während der Klassifikator im Fall von Semi-Supervised Learning auch ohne Labelinformationen nachtrainiert werden kann, sind diese bei Supervised Learning zwingend erforderlich. Da im Rahmen dieser Masterarbeit nur sehr grob gelabelte unbekannte Daten verfügbar sind, werden beide Vorgehensweisen betrachtet. Der Schwerpunkt liegt jedoch auf der Umsetzung mittels Supervised Learning. Die Ausführungen zur Realisierung mit Hilfe von Semi-Supervised Learning beschränken sich daher auf einen ausgewählten vielversprechenden Ansatz.

3.2 Modellierung von Unsicherheit

Während Deep-Learning-basierte Ansätze aktuell viele Bereiche des maschinellen Lernens dominieren, müssen sie häufig als gut funktionierende Black-Box-Modelle betrachtet werden. Durch ihre komplexe Struktur ist es oft nicht möglich, die Prozesse im Netzwerk zu interpretieren oder nachzuvollziehen. In der Regel kann zudem kein Maß für die Sicherheit einer Klassifikation angegeben werden. Bei der Anwendung auf

[1] Für Strukturen mit mehr als einer Hidden-Schicht kann die bayes'sche Gleichung nicht analytisch gelöst werden (siehe Abschnitt 2.3, Seite 15)

unbekannten Daten kann somit nicht beurteilt werden, ob es sich um ein korrektes Ergebnis oder um eine Fehlklassifikation infolge von zu weit von den Trainingsdaten entfernten Eingaben handelt. In [Szegedy et al., 2014] wird zudem gezeigt, dass viele Netzwerkstrukturen bereits bei leicht veränderten Trainingsdaten, sogenannten Adversarial Examples (siehe Abschnitt 2.2.3, Seite 13), getäuscht werden können. Um diesen Problemen zu begegnen, gibt es in der Literatur aktuell vermehrt Bemühungen, das Verständnis der verwendeten Strukturen zu verbessern und Deep-Learning mit bekannten bayes'schen Verfahren zu kombinieren[2]. Im Folgenden werden geeignete Ansätze zur Modellierung von Klassifikationsunsicherheiten vorgestellt.

3.2.1 Approximierende bayes'sche Ansätze

Da exakte bayes'sche Inferenz für Deep-Learning-Ansätze nicht möglich ist, müssen approximierende Methoden zur Modellierung von Unsicherheiten genutzt werden. In Abschnitt 2.3 (Seite 15) wurden hierfür die zwei geläufigsten Verfahrensklassen: Markov-Chain-Monte-Carlo-Verfahren und Variational Inference vorgestellt. Nachfolgend werden wichtige Publikationen für beide Klassen erläutert.

3.2.1.1 Markov-Chain-Monte-Carlo-Verfahren

Markov-Chain-Monte-Carlo-Verfahren ermöglichen die Bestimmung der A-posteriori-Verteilung durch Stichproben. Die meisten klassischen Verfahren dieser Art können jedoch nicht auf Deep-Learning-Strukturen angewendet werden, da für die Bestimmung der Stichproben meist viele Iterationen durch den kompletten Datensatz nötig sind. In [Welling und Teh, 2011] wird erstmals ein anderer Weg vorgeschlagen. Die Autoren kombinieren Markov-Chain-Monte-Carlo-Verfahren und stochastische Optimierung auf Basis von sogenannten Langevin-Dynamiken [Neal, 2011]. Hierzu wird bewusst Rauschen in die Gewichtsänderungen induziert, sodass die komplette A-posteriori-Verteilung beim Training berücksichtigt werden kann. Auf diese Weise kann die Unsicherheit in den Gewichten auf die Ausgabe des Netzwerks übertragen werden.

[2] Bayesian Deep Learning Workshop (NIPS 2016): `http://bayesiandeeplearning.org/` (Zuletzt aufgerufen: 01.04.2017)

Es wird zudem gezeigt, dass Gradienten über Mini-Batches zur Approximation genügen und somit keine zeitaufwendigen Berechnungen über den kompletten Datensatz nötig sind. Seit den ersten Arbeiten von [Welling und Teh, 2011] gab es zahlreiche Erweiterungen dieser Grundidee. In [Li et al., 2016] und [Chen et al., 2016] werden diese übersichtlich zusammengefasst. In [Li et al., 2016] wird zudem eine optimierte Variante mit gewichtsspezifischen Lernraten in Anlehnung an das Optimierungsverfahren RMSprop [Tieleman und Hinton, 2012] vorgeschlagen. Während diese Ansätze im Vergleich zu herkömmlichen Markov-Chain-Monte-Carlo-Verfahren deutlich zeiteffizienter und somit skalierbarer sind, beinhalten die Updategleichungen noch immer rechenintensive Terme. Zudem müssen die Parameter des Algorithmus sehr sorgfältig gewählt werden, damit der Lernprozess nicht von dem zusätzlich eingebrachten Rauschen dominiert wird. Aus den genannten Gründen werden in dieser Masterarbeit keine derartigen Verfahren verfolgt.

3.2.1.2 Variational Inference

Der Großteil der Verfahren zur Bestimmung von Unsicherheiten basiert auf Variational Inference. Bei dieser Art von Ansätzen soll eine approximierte A-posteriori-Verteilung $q_\Phi(\theta)$ über den Gewichten θ durch das Lösen eines Optimierungsproblems bestimmt werden. In [Graves, 2011] wurde erstmals ein praktischer Weg für die Anwendung in Verbindung mit Neuronalen Netzen gezeigt. Auf Grund mangelnder Leistung konnte sich dieser jedoch nicht durchsetzen. Erst mit [Blundell et al., 2015] wurden Ansätze auf Basis von Variational Inference erneut aufgegriffen und erfolgreich angewendet. In dem in [Blundell et al., 2015] vorgestellten Bayes-by-Backprop-Algorithmus wird die A-posteriori-Verteilung $q_\Phi(\theta)$ über unabhängige Verteilungen $q_{\Phi_i}(\theta_i)$ mit den veränderlichen Parametern $\Phi_i = \{\mu_i, \sigma_i^2\}$ definiert:

$$q_{\Phi_i}(\theta_i) = N(\mu_i, \sigma_i^2) \quad \forall \theta_i \in \theta \tag{3.1}$$

Der Wert für jedes einzelne Gewicht θ_i wird somit nicht, wie sonst üblich, in Form einer festen Zahl definiert, sondern fortwährend aus einer Verteilung gesampelt:

$$\theta_i = \mu_i + \sigma_i \epsilon_i \quad \text{mit: } \epsilon_i \sim N(0, 1) \tag{3.2}$$

Während des Lernprozesses muss für jedes Gewicht θ_i ein Mittelwert μ_i und eine Standardabweichung σ_i gelernt werden[3]. Folglich verdoppelt sich die Anzahl der zu bestimmenden Parameter. Abbildung 3.2 verdeutlicht die Definition der Gewichte anhand eines einfachen vollverschalteten Netzwerks.

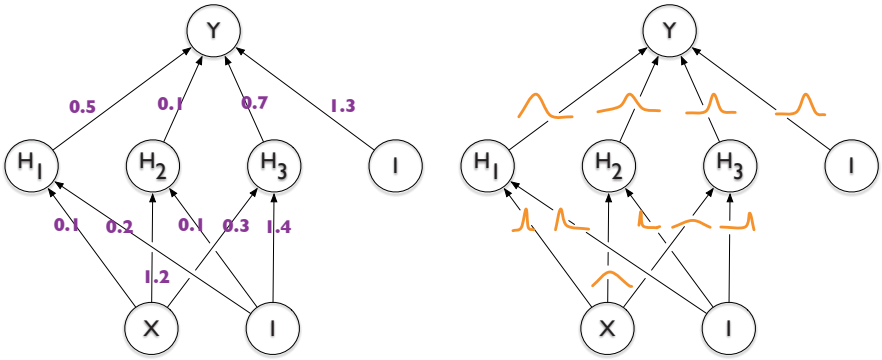


Abbildung 3.2: Definition der Gewichte

*Dargestellt ist die Definition der Gewichte in Form von Verteilungen nach [*Blundell *et al., 2015]. Hierbei bezeichnet X die Eingabe, I das Bias-Neuron und H beziehungsweise Y die Ausgabe einer Schicht. [*Blundell *et al., 2015]*

Durch die Wahl der Gewichte nach Formel (3.2) wird während des Lernprozesses nicht ein einzelnes Netzwerk, sondern eine Vielzahl von Netzwerken mit unterschiedlichen Gewichten, gesampelt aus gemeinsamen Verteilungen, betrachtet. Daher kann dieser Ansatz ebenfalls als eine Art Ensemble interpretiert werden, wobei die Anzahl der Gewichte im Vergleich zu einem echten Ensemble, bestehend aus mehreren Netzwerken, um ein Vielfaches kleiner ist. In einer Evaluation auf dem MNIST-Datensatz[4] zeigen die Autoren, dass ihr Verfahren zu ähnlichen Ergebnissen wie der Dropout-Ansatz nach [Srivastava et al., 2014] kommt. Allerdings werden an dieser Stelle nur vollverschaltete Netzwerke untersucht. Dies könnte der Tatsache geschuldet sein, dass aufbauend auf der beschriebenen theoretischen Grundlage nur dann eine exakte Approximation gewährleistet werden kann, wenn für jedes präsentierte Beispiel separate

[3] Formel (3.2) beschreibt eine vereinfachte Darstellung. Damit σ stets positiv ist, wird die Standardabweichung in [Blundell et al., 2015] zu $\sigma = \log(1 + \exp(\phi))$ gewählt und anschließend ϕ optimiert.

[4] The MNIST database of handwritten digits: `http://yann.lecun.com/exdb/mnist/` (Zuletzt abgerufen: 17.04.2017)

Gewichte gesampelt werden. Im Hinblick auf Convolutional Neural Networks wäre dies besonders rechenintensiv und würde bisherige Konzepte zur Parallelisierung [Chetlur et al., 2014] außer Kraft setzen. In der kürzlich erschienenen Publikation von [Oliveira et al., 2016] zeigen die Autoren jedoch, dass es möglich ist, die Gewichte nur einmalig für alle Beispiele eines Mini-Batches zu sampeln, ohne dass es dabei zu einem signifikanten Verlust bei der Schätzung der Unsicherheit kommt. Leider werden auch in dieser Publikation ausschließlich vollverschaltete Neuronale Netze betrachtet. Zudem liegt der Fokus einzig auf dem Vergleich der bestimmten Unsicherheiten. Eine Betrachtung zu den erreichten Klassifikationsergebnissen fehlt gänzlich.

Aufbauend auf dem beschriebenen Bayes-by-Backprop-Algorithmus [Blundell et al., 2015], wird in [Kingma et al., 2015] ein interessanter Weg beschrieben, mit dessen Hilfe das aufwendige Sampeln der Gewichte umgangen werden kann. So wird gezeigt, dass das Rauschen in den Gewichten durch geschickte Reparametrierung in ein Rauschen in den Aktivierungen überführt werden kann. Die Autoren bezeichnen dieses Vorgehen als „Local Reparameterization Trick“ und zeigen auffällige Parallelen zur Anwendung von Dropout nach [Srivastava et al., 2014] (siehe Abschnitt 2.2.1, Seite 9). Die Vorgehensweise lässt sich am einfachsten anhand einer vollverschalteten Schicht l erläutern. Hierzu seien $\underline{\mathbf{W}} \in \mathbb{R}^{N^{(l-1)} \times N^{(l)}}$ die nach Formel (3.2) definierten Gewichte der Schicht und $\underline{\mathbf{X}} \in \mathbb{R}^{M \times N^{(l-1)}}$ die Eingaben in Form eines Mini-Batches der Größe M. Mit diesen beiden Größen kann die Aktivierung $\underline{\mathbf{Z}} \in \mathbb{R}^{M \times N^{(l)}}$ der Schicht durch ein einfaches Matrixprodukt bestimmt werden:

$$\underline{\mathbf{Z}} = \underline{\mathbf{X}}\,\underline{\mathbf{W}} \tag{3.3}$$

Durch Nachvollziehen der dabei durchgeführten elementweisen Berechnungen kann gut gezeigt werden, dass die resultierenden Aktivierungen durch eine Summation normalverteilter Größen berechnet werden und somit ebenfalls einer Normalverteilung unterliegen:

$$q_\Phi(w_{i,j}) = N(\mu_{i,j}, \sigma_{i,j}^2) \;\; \forall w_{i,j} \in \underline{\mathbf{W}} \;\; \Rightarrow \;\; q_\Phi(z_{m,j} | \underline{\mathbf{X}}, \underline{\mathbf{W}}) = N(\gamma_{m,j}, \delta_{m,j}^2) \tag{3.4}$$

$$\text{wobei:} \;\; \gamma_{m,j} = \sum_{i=1}^{N^{(l-1)}} x_{m,i}\mu_{i,j} \;\; \text{und} \;\; \delta_{m,j} = \sum_{i=1}^{N^{(l-1)}} x_{m,i}^2 \sigma_{i,j}^2$$

Folglich ist es möglich, die einzelnen Aktivierungen $z_{m,j}$ direkt aus den auf diese Weise definierten Verteilungen zu sampeln. Für jede Aktivierung $z_{m,j}$ ergibt sich:

$$z_{m,j} = \gamma_{m,j} + \sqrt{\delta_{m,j}}\xi_{m,j} \quad \text{mit: } \xi_{m,j} \sim N(0,1) \tag{3.5}$$

Über alle Beispiele betrachtet ist $\underline{\xi}$ hierbei eine Matrix mit der gleichen Größe wie die Aktivierungen $\underline{\mathbf{Z}}$. An dieser Stelle sind deutliche Parallelen zu Dropout erkennbar. So basiert Dropout ebenfalls auf dem Hinzufügen von multiplikativem Rauschen mit dem Unterschied, dass dabei die Eingaben einer Schicht verändert werden und die Gewichte in der Regel nicht in Form von Verteilungen definiert sind. Da sich die Eingaben einer Schicht l jedoch nur durch die Anwendung einer Ausgabefunktion von den Aktivierungen der Schicht $l-1$ unterscheiden, kann Dropout ebenfalls als eine Art Variational Inference interpretiert werden. Für weitere mathematische Details sei an dieser Stelle auf Abschnitt 3 und Anhang B in [Kingma et al., 2015] verwiesen. Nach der Erläuterung dieses Zusammenhangs zeigen die Autoren, dass die Approximationseigenschaften von Dropout weiter verbessert werden können, indem adaptive Dropout-Parameter p verwendet werden. Hierfür ergeben sich mehrere Möglichkeiten. So kann eine individuelle Dropout-Rate für jede Schicht, für jedes Neuron oder für jedes Gewicht gelernt werden. Letztere Wahl kommt der Definition der Gewichte in Form von einer Verteilung sehr nahe. Auf Grund der Ähnlichkeit zu Methoden auf Basis von Variational Inference wird Dropout mit veränderlichem Parameter p in der Literatur als Variational Dropout bezeichnet.

Parallel zu den bisher beschriebenen Publikationen wurde die Eignung von Dropout zur Realisierung von Variational Inference von Yarin Gal im Rahmen seiner Dissertation [Gal, 2016] und den zugehörigen Publikationen [Gal und Ghahramani, 2015d, Gal und Ghahramani, 2015b, Gal und Ghahramani, 2015c] betrachtet. Anders als in [Kingma et al., 2015] erfolgt die Argumentation in diesen Arbeiten auf der Basis der Kostenfunktion. Anhand dieser wird gezeigt, dass die Verwendung von Dropout der bayes'schen Approximation in tiefen Gauß-Prozessen gleicht und somit als eine Form von Variational Inference interpretiert werden kann. In [Gal und Ghahramani, 2015d] werden hierfür zunächst beide Techniken vorgestellt und anschließend die

Dualität hergeleitet. Ergänzende mathematische Erläuterungen werden in [Gal und Ghahramani, 2015b] und [Gal und Ghahramani, 2015c] beschrieben. Der für diese Masterarbeit relevante Teil zur Extraktion von Unsicherheiten soll anhand von einem Netzwerk mit L vollverschalteten Schichten erläutert werden. Der Definition der approximierten Verteilung $q_\Phi(\theta)$ mit $\theta = \{\widehat{\underline{\mathbf{W}}}^{(l)}\}_{l=1}^{L}$ wird die Wirkung von Dropout auf die Gewichtsmatrizen zu Grunde gelegt. So entspricht beispielsweise die Anwendung von Bernoulli-Dropout der Multiplikation jeder Gewichtsmatrix $\underline{\mathbf{W}} \in \mathbb{R}^{N^{(l-1)} \times N^{(l)}}$mit einer binären Maske in Form einer Diagonalmatrix $\underline{\mathbf{M}}^{(l)}$, deren Elemente für jedes präsentierte Beispiel aus einer Bernoulli-Verteilung gezogen werden:

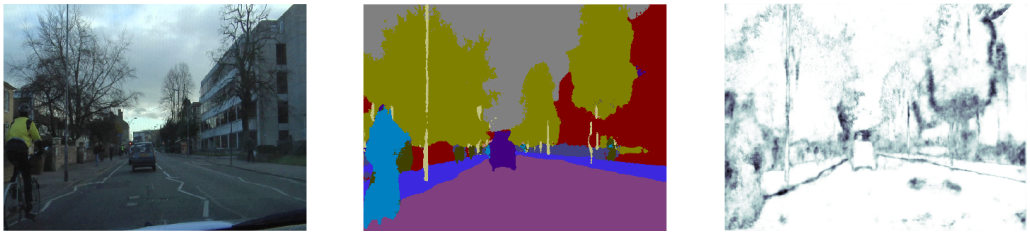
$$\widehat{\underline{\mathbf{W}}}^{(l)} = \underline{\mathbf{W}}^{(l)} \operatorname{diag}([z_i]_{i=1}^{N^{(l)}}) \quad \text{mit: } z_i \sim \text{Bernoulli}(p^{(l)}) \tag{3.6}$$

Die nicht-stochastischen Gewichtsmatrizen $\{\underline{\mathbf{W}}^{(l)}\}_{l=1}^{L}$ bilden hierbei die veränderlichen Parameter Φ der Verteilung $q_\Phi(\theta)$. Durch die in Abschnitt 2.3.2 (Seite 17) beschriebene Minimierung der Kullback-Leibler-Divergenz kann während des Trainingsprozesses mit den Daten $\underline{\mathbf{X}}, \underline{\mathbf{T}}$ eine möglichst gute Approximation der wahren A-postiori-Verteilung $p(\theta|\underline{\mathbf{X}}, \underline{\mathbf{T}})$ gelernt werden. Ausgehend von Formel (2.7) aus Abschnitt 2.3 (Seite 15) kann die Unsicherheit in den Gewichten letztlich durch Monte-Carlo-Integration auf die Klassifikation übertragen werden:

$$\begin{aligned} p(y = c|\underline{\mathbf{x}}^*, \underline{\mathbf{X}}, \underline{\mathbf{T}}) &= \int p(y = c|\underline{\mathbf{x}}^*, \theta) p(\theta|\underline{\mathbf{X}}, \underline{\mathbf{T}})\, d\theta \\ &\approx \int p(y = c|\underline{\mathbf{x}}^*, \theta) q_\Phi(\theta)\, d\theta \\ &\approx \frac{1}{K} \sum_{k=1}^{K} p(y = c|\underline{\mathbf{x}}^*, \{\widehat{\underline{\mathbf{W}}}_k^{(l)}\}_{l=1}^{L}) \end{aligned} \tag{3.7}$$

Hierbei ist K die Anzahl der zur Approximation verwendeten Sampel. Die in Formel (3.6) dargestellte Beschränkung auf Diagonalmatrizen ist nötig, um Dropout effizient auf die Eingabe einer Schicht zurückführen zu können. Während auf diese Weise für jedes Beispiel individuelle Gewichte betrachtet werden können, führt es jedoch auch dazu, dass nur komplette Zeilen in den Gewichtsmatrizen maskiert werden können und die Genauigkeit der Approximation infolgedessen eingeschränkt wird. In Abschnitt 3.3.2 in [Gal, 2016] wird daher angemerkt, dass Dropout Variational Inference,

ähnlich wie andere Ansätze auf diesem Gebiet, dazu neigt Unsicherheiten zu unterschätzen. Dennoch wird Dropout Variational Inference bereits vielseitig verwendet. In [Gal und Ghahramani, 2015a] wurde der Ansatz hierzu zunächst auf Convolutional Neural Networks, sogenannte Bayesian Convolutional Neural Networks (BCNN), übertragen. In der Folge wurden BCNN in den unterschiedlichsten Bereichen eingesetzt. So werden sie in [Kendall und Cipolla, 2016] zur Kamera-Lokalisierung im Rahmen von visionbasierten SLAM (Simultaneous Localization and Mapping) oder in [Leibig et al., 2016] zur Erkennung von Netzhauterkrankungen angewendet. Weiterhin wurde in [Kendall et al., 2015] ebenfalls eine der bekanntesten Netzwerkarchitekturen zur semantischen Segmentierung von Szenen SegNet [Badrinarayanan et al., 2015] um die Modellierung von Unsicherheiten erweitert. In der Folge konnte die Segmentierungsfähigkeit deutlich verbessert werden. Abbildung 3.3 verdeutlicht die mit Dropout Variational Inference extrahierten Unsicherheiten an einem konkreten Beispiel.



(a) Eingabe **(b)** Segmentierung **(c)** Unsicherheit

Abbildung 3.3: Bayesian SegNet

Dargestellt sind Ausgaben des Bayesian SegNet für ein konkretes Beispiel. Es ist zu erkennen, dass die Unsicherheiten besonders hoch (dunkel dargestellt in (c)) an Klassengrenzen und an Stellen mit teilweiser Verdeckung sind. [Kendall et al., 2015]

3.2.2 Nicht-bayes'sche Ansätze

Neben den Ansätzen von Verfechtern der bayes'schen Wahrscheinlichkeitstheorie gibt es, besonders von Google DeepMind, Bestrebungen, Unsicherheitsinformationen mit Hilfe von nicht-bayes'schen Verfahren zu extrahieren. In [Lakshminarayanan et al., 2016] und [Osband, 2016] werden die Ansätze auf Basis von Dropout Variational In-

ference scharf kritisiert. Die Autoren legen dar, dass die Interpretation von Dropout als Mittelung über eine Vielzahl kleinerer Netzwerke plausibler ist, solange nicht jeder Lernparameter und somit ebenfalls die Dropout-Raten hinsichtlich der Daten optimiert werden. In [Lakshminarayanan et al., 2016] wird als Alternative die Verwendung von Ensembles in Kombination mit Adversarial Training (siehe Abschnitt 2.2.3, Seite 13) vorgeschlagen. Die Autoren argumentieren, dass schon lange bekannt sei, dass die Klassifikationsleistung durch den Einsatz mehrerer Netzwerke gesteigert werden kann. Durch die Verwendung von Adversarial Training soll die Robustheit der Netzwerke zusätzlich gesteigert und somit die Genauigkeit der Unsicherheitsinformationen verbessert werden. Für das konkrete Training der Netzwerke wird eine unterschiedliche zufällige Initialisierung der Gewichte sowie eine abweichende Reihenfolge in den präsentierten Daten vorgeschlagen. Beide Punkte seien ausreichend, um Netzwerke mit genügend Diversität zu trainieren. Die Experimente bestätigen diese Annahmen und verdeutlichen, dass mit Ensembles deutlich höhere Klassifikationsergebnisse erzielt werden können. Weiterhin wird gezeigt, dass unbekannte Daten zu Abweichungen in den Ausgaben der einzelnen Netzwerke führen und dass auf deren Basis Unsicherheitsinformationen abgeleitet werden können. Im Rahmen der Experimente wird jedoch auch deutlich, dass hierfür mindestens fünf, besser sogar zehn, Netzwerke nötig sind. Obwohl diese parallel und unabhängig voneinander trainiert werden können, erfordert das Training eines solchen Ensembles sehr viel mehr Ressourcen in Form von Rechenleistung oder Zeit.

In [Osband et al., 2016] wird ein sehr ähnlicher Ensemble-Ansatz vorgestellt. Obwohl das Ziel dieser Publikation die Realisierung einer effizienten Explorationsstrategie für einen Reinforcement-Agenten ist, kann die Grundidee auf die Fragestellung in dieser Masterarbeit übertragen werden. Anstelle einer Vielzahl von Netzwerken wird in [Osband et al., 2016] lediglich ein Netzwerk, jedoch aber mit mehreren Ausgabeinstanzen trainiert. Abbildung 3.4 verdeutlicht die von den Autoren vorgeschlagene Netzwerkstruktur. Die Grundidee basiert hierbei auf der Tatsache, dass in frühen Schichten meist nur sehr allgemeine Merkmale gelernt werden, welche in den einzelnen Netzwerken des Ensembles ähnlich sind. Aus diesem Grund können die frühen Schichten zu einem geteilten Netzwerk (engl. Shared Network) zusammengefasst werden. Die an-

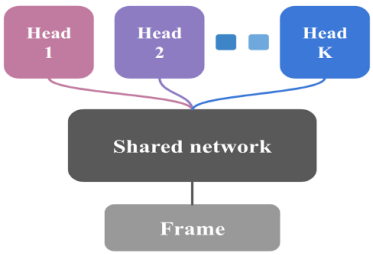


Abbildung 3.4: Netzwerkstruktur nach [Osband et al., 2016]
Dargestellt ist die Netzwerkstruktur für das in [Osband et al., 2016] realisierte Ensemble, bestehend aus einem geteilten Netzwerk (Shared Network) und K individuellen Ausgabeinstanzen (Heads). Die zu verarbeitende Eingabe wird durch ein einzelnes Bild (Frame) gebildet. [Osband et al., 2016]

schließenden Schichten werden für jedes Netzwerk des Ensembles getrennt in Form eines Kopfes (engl. Head) realisiert. Auf diese Weise kann der benötigte Ressourcenbedarf deutlich gesenkt werden. Für das Training schlagen die Autoren eine an Bagging [Breiman, 1996] angelehnte Vorgehensweise vor. Bagging sieht individuelle Datensätze für jedes Netzwerk vor, welche durch das Ziehen einzelner Datenpunkte mit Zurücklegen aus dem Trainingsdatensatz bestimmt werden. Zur Realisierung wird eine zusätzliche binäre Maske $\underline{\mathbf{m}} \in \{0,1\}^K$ eingeführt, deren Elemente fortwährend aus einer Bernoulli-Verteilung gezogen werden. Jedes Element m_k dieser Maske beschreibt, ob der Head k im aktuellen Trainingsschritt lernt. Über mehrere Epochen betrachtet, kann auf diese Weise klassisches Bagging simuliert werden. Die beschriebene Vorgehensweise ermöglicht das Trainieren von mehreren Netzwerken in einem Schritt, erfordert jedoch auch die Verrechnung der einzelnen Gradienten für die geteilten Gewichte. Die Autoren schlagen für ihre Problemstellung die Verwendung von zehn individuellen Heads ($K = 10$) und eine Skalierung der Gradienten mit $1/10$ vor.

3.2.3 Modifikation der Trainingsdaten

In diesem Abschnitt der State-of-the-Art-Betrachtungen werden verschiedene Maße zur Beschreibung von Unsicherheit vorgestellt, auf deren Grundlage möglichst informative, neue Trainingsbeispiele ausgewählt und anschließend in den Trainingsdatensatz über-

nommen werden können. Die dabei untersuchte Fragestellung ist ebenfalls entscheidender Bestandteil bei der Realisierung von Active Learning (siehe Abschnitt 2.1.3, Seite 8). Daher kann an dieser Stelle vor allem auf Publikationen aus diesem Bereich zurückgegriffen werden. Aufbauend auf [Islam, 2016] werden in [Gal et al., 2016] verschiedene Maße, sogenannte Akquirierungsfunktionen, zusammengetragen und verglichen. Grundlage für die folgende formale Definition dieser Größen sind stets K Ausgaben für ein unbekanntes Beispiel $\underline{\mathbf{x}}^*$ im Rahmen eines Klassifikationsproblems mit C zu unterscheidenden Klassen, den Trainingsdaten $\mathcal{D} = \{\underline{\mathbf{X}}, \underline{\mathbf{T}}\}$ und den Netzwerkgewichten θ. In [Gal et al., 2016] werden die Ausgaben durch Dropout Variational Inference erzeugt. Die Akquirierungsfunktionen sind aber allgemeingültig und können somit auch in Verbindung mit einem Ensemble-Ansatz angewendet werden. Im Detail werden in [Gal et al., 2016] die folgenden Maße verglichen:

1. Auswahl von Beispielen, welche die Entropie $\mathbb{H}[y|\underline{\mathbf{x}}^*, \mathcal{D}]$ maximieren

$$\mathbb{H}[y|\underline{\mathbf{x}}^*, \mathcal{D}] = -\sum_{c=1}^{C} p(y = c|\underline{\mathbf{x}}^*, \mathcal{D}) \log p(y = c|\underline{\mathbf{x}}^*, \mathcal{D}) \tag{3.8}$$

Die Entropie bildet das Standardmaß zur Beurteilung der Sicherheit. Für unsichere Klassifikationen wird ein hohes Maß an Unordnung im Ergebnis und folglich eine hohe Entropie erwartet.

2. Auswahl von Beispielen, welche die Mutual Information $\mathbb{I}(y, \theta|\underline{\mathbf{x}}^*, \mathcal{D})$ zwischen Klassifikation und Verteilung über den Ausgaben maximieren (Bayesian Active Learning by Disagreement - BALD)

$$\begin{aligned} \mathbb{I}[y, \theta|\underline{\mathbf{x}}^*, \mathcal{D}] &= \mathbb{H}[y|\underline{\mathbf{x}}^*, \mathcal{D}] - \mathbb{E}_{p(\theta|\mathcal{D})}[\mathbb{H}[y|\underline{\mathbf{x}}^*, \theta]] \\ &\approx \mathbb{H}[y|\underline{\mathbf{x}}^*, \mathcal{D}] + \frac{1}{K} \sum_{k=1}^{K} \sum_{c=1}^{C} p(y = c|\underline{\mathbf{x}}^*, \theta^k) \log p(y = c|\underline{\mathbf{x}}^*, \theta^k) \end{aligned} \tag{3.9}$$

Beispiele, die diese Akquirierungsfunktion maximieren, besitzen eine hohe Varianz in der Aktivierung der abschließenden Softmax-Schicht. Das Netzwerk ist sich für diese Beispiele im Durchschnitt unsicher, es existieren zudem Ausgaben, die mit fälschlicherweise hoher Sicherheit zu unterschiedlichen Klassifikationsergebnissen führen.

3. Auswahl von Beispielen, welche die Variationsrate $R(\underline{\mathbf{x}}^*)$ maximieren

$$R(\underline{\mathbf{x}}^*) = 1 - \frac{f}{K} \quad \text{mit: } f = \sum_{k=1}^{K} \delta(y^k) \text{ und } \delta(y^k) = \begin{cases} 1 & y^k = c^* \\ 0 & y^k \neq c^* \end{cases} \tag{3.10}$$

Hierbei ist c^* das am häufigsten vorkommende Label (der Modus) für das Beispiel $\underline{\mathbf{x}}^*$. Die Variationsrate beschreibt folglich die Abweichung von diesem Label.

4. Auswahl von Beispielen, welche die mittlere Standardabweichung $\bar{\sigma}(\underline{\mathbf{x}}^*)$ maximieren

$$\bar{\sigma}(\underline{\mathbf{x}}^*) = \frac{1}{C} \sum_{c=1}^{C} \sqrt{\mathbb{E}_{p(\theta|\mathcal{D})}[p(y = c|\underline{\mathbf{x}}^*, \theta)] - \mathbb{E}_{p(\theta|\mathcal{D})}[p(y = c|\underline{\mathbf{x}}^*, \theta)]^2} \tag{3.11}$$

Soll als Bewertungskriterium nicht die Breite, sondern die Stärke der Streuung um den Mittelwert angesetzt werden, kann an dieser Stelle ebenfalls die Varianz verwendet werden. Durch die Allgemeingültigkeit beider Größen kann dieses Maß zudem auch für Regressionsprobleme genutzt werden.

Die Eignung der unterschiedlichen Akquirierungsfunktionen wird in [Gal et al., 2016] anhand eines konstruierten Active-Learning-Szenarios auf dem MNIST-Datensatz untersucht. Für jede Funktion wird hierbei ein initialer Datensatz, bestehend aus zwei zufällig gewählten Beispielen pro Klasse, schrittweise um jeweils zehn neue Datenpunkte erweitert bis eine Gesamtgröße von 1.000 Beispielen erreicht wurde. Die genaue Implementierung[5] zeigt, dass, anders als erwartet, in jedem Akquirierungsschritt ein neues Convolutional Neural Network für 50 Epochen trainiert und anschließend auf den Testdaten evaluiert wird. Die verschiedenen Maße werden auf Basis von 100 Netzwerkausgaben berechnet und mit einer zufälligen Auswahl neuer Datenpunkte verglichen. Im Rahmen der Analyse des Quelltexts wurde zudem festgestellt, dass die Implementierung für eine Akquirierung mittels maximaler Standardabweichung einen Fehler aufweist[6]. Bedingt durch eine fehlerhafte Indizierung wird lediglich eine der

[5] `https://github.com/Riashat/Active-Learning-Bayesian-Convolutional-Neural-Networks/tree/master/ConvNets/FINAL_Averaged_Experiments/Final_Experiments_Run` (Zuletzt aufgerufen: 17.04.2017)

[6] `https://github.com/Riashat/Active-Learning-Bayesian-Convolutional-Neural-Networks/blob/master/ConvNets/FINAL_Averaged_Experiments/Final_Experiments_Run/N100_Dropout_Segnet.py#L237` (Zuletzt aufgerufen: 17.04.2017)

100 Netzwerkausgaben berücksichtigt. In der Folge geht die Aussagekraft verloren und die Akquirierung ähnelt, wie von den Autoren festgestellt, einer zufälligen Auswahl. Im Rahmen der State-of-the-Art-Analyse wurde daher versucht, das Experiment zu wiederholen. Anders als in der Publikation wurde der Klassifikator dabei nach jedem Akquirierungsschritt lediglich für 10 Epochen nachtrainiert und nicht komplett neu trainiert. Trotz dieser Modifikation in der Trainingsprozedur konnten die Ergebnisse sehr gut reproduziert werden. Abbildung 3.5 zeigt den Vergleich der Akquirierungsfunktionen für das vorgestellte Active-Learning-Szenario.

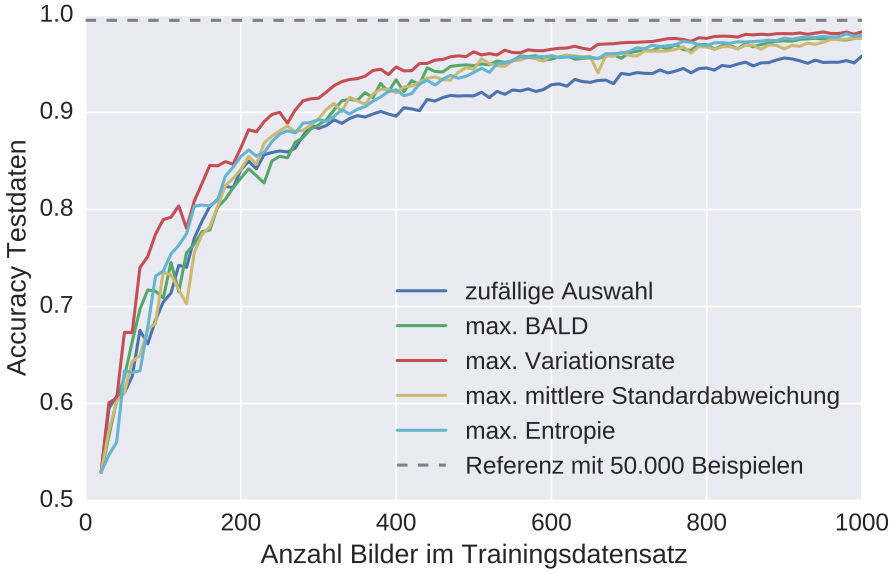


Abbildung 3.5: Vergleich verschiedener Akquirierungsfunktionen
Dargestellt sind die Ergebnisse für das reproduzierte Experiment aus [Gal et al., 2016]. Hierbei wird anhand eines konstruierten Active-Learning-Szenarios die Akquirierung neuer Trainingsdaten auf Basis der vorgestellten Funktionen mit der zufälligen Auswahl neuer Datenpunkte verglichen.

Es ist deutlich zu erkennen, dass die Akquirierung neuer Datenpunkte nach den vorgeschlagenen Funktionen effizienter als eine zufällige Auswahl ist. So kann bereits mit 1.000 der insgesamt 50.000 verfügbaren Datenpunkte des Trainingsdatensatzes annähernd die Leistung des Referenzklassifikators, welcher mit allen Beispielen trainiert wurde, erzielt werden. Die Autoren schlussfolgern aus dem Experiment, dass neue Daten auf Basis der Variationsrate ausgewählt werden sollten.

Neben den beschriebenen Größen existieren in der Literatur weitere Maße, beispielsweise für eine Auswahl auf Basis von Best-versus-Second-Best (BvsSB) [Joshi et al., 2009], der Informationsdichte [Li und Guo, 2013], der maximalen Reduzierung der Gesamtentropie [Holub et al., 2008] oder der Gradientenlänge [Hasan und Roy-Chowdhury, 2014]. Jedoch konnte sich keines dieser Maße in der Praxis durchsetzen, da entweder die Ergebnisse nicht auf andere Anwendungsbereiche übertragen werden können, es bei der Verwendung einer manuellen Nachkontrolle bedarf oder die Maße speziell für Support-Vector-Maschinen entworfen wurden.
Während neue Datenpunkte meist, wie in [Gal et al., 2016] beschrieben, einfach dem bisherigen Datensatz hinzugefügt werden, ist eine gezielte stärkere Gewichtung der neuen Beispiele ebenfalls denkbar. In [Islam, 2016] wird hierfür die Verwendung sogenannter Sample Weights andiskutiert.

3.2.4 Nachtraining des Klassifikators

Nachdem der Datensatz erweitert wurde, muss der Klassifikator im nächsten Schritt erneut trainiert werden. Hierfür ergeben sich grundsätzlich zwei Möglichkeiten. Wurden dem Datensatz ausschließlich gelabelte Beispiele hinzugefügt, so kann das Nachtraining mittels Supervised Learning erfolgen. Ist dies nicht der Fall, kann auf Verfahren aus dem Bereich von Semi-Supervised Learning zurückgegriffen werden. Im Anschluss werden beide Möglichkeiten vorgestellt.

3.2.4.1 Supervised Learning

Die meisten inkrementellen Lernansätze gehen davon aus, dass dem Datensatz ausschließlich gelabelte Daten hinzugefügt werden. Liegen keine gelabelten Daten vor, müssen die Label vor dem Training manuell durch einen Human-in-the-loop (HITL) bestimmt werden. Die genaue Realisierung des Nachtrainings ist in vielen Fällen stark von der vorliegenden Problemstellung abhängig. In [Käding et al., 2016] werden anhand des MS-COCO-Datensatzes [Lin et al., 2014] verschiedene Fragestellungen mit dem Ziel den Rechenaufwand möglichst gering zu halten, untersucht. Beginnend mit einem sehr kleinen initialen Datensatz werden hierbei schrittweise zufällig gewählte

Beispiele zu den Trainingsdaten hinzugefügt und das Netzwerk nachtrainiert. Die Autoren kommen zum Ergebnis, dass bereits wenige Iterationen durch den Datensatz genügen, um den Klassifikator sukzessive zu verbessern und eine zum normalen Lernprozess mit allen Daten vergleichbare Leistung zu erreichen. Die Autoren stellen zudem fest, dass keine alten Daten aus dem Datensatz entfernt werden dürfen und dass ein Labelfehler von bis zu 10% toleriert werden kann.

3.2.4.2 Semi-Supervised Learning

Semi-Supervised Learning kombiniert überwachtes und unüberwachtes Lernen, sodass der Klassifikator auch von ungelabelten Daten profitieren kann. Der State of the Art in diesem Bereich ist sehr umfangreich und geht bis in die 80er Jahre zurück. In [Chapelle et al., 2006] werden erste klassische Ansätze übersichtlich zusammengefasst. Konkrete Anwendungen im Deep-Learning-Bereich sind bisher jedoch sehr selten. In den meisten Ansätzen wird das Training in zwei aufeinanderfolgende Phasen untergliedert. Mit Hilfe von Autoencodern und ungelabelten Daten werden zunächst unüberwacht aussagekräftige Repräsentationen in den einzelnen Schichten gelernt. Anschließend erfolgt ein Fine-Tuning durch den Einsatz von Supervised Learning und gelabelten Daten. In [Rasmus et al., 2015a, Rasmus et al., 2015b] werden, aufbauend auf [Valpola, 2015], erstmals beide Phasen in Form von einem sogenannten Ladder Network kombiniert. Ermöglicht wird dies durch die spezielle Struktur dieser Netzwerke. Diese erlaubt die gleichzeitige Minimierung von Rekonstruktionsfehlern und Labelkosten. Die Autoren merken zudem an, dass nahezu jedes zuvor überwacht trainierte Netzwerk zu einem Ladder Network erweitert werden kann. Hierzu wird zunächst ein mehrschichtiger Encoder konstruiert. Dieser besitzt die gleiche Struktur wie das Ausgangsnetzwerk und verwendet die gleichen Gewichte. Der Encoder unterscheidet sich vom Ausgangsnetzwerk lediglich durch das Hinzufügen von Rauschen in jeder Schicht. Anschließend wird ein Decoder erstellt, welcher die Umkehrung der Funktion jeder Schicht des Encoders ermöglicht[7]. Im letzten Schritt werden zusätzliche laterale Verbindungen (engl. Skip Connections) zwischen korrespondierenden Schichten des

[7] Bis zu diesem Zeitpunkt gleicht die erstellte Struktur einem mehrschichtigen Denoising Autoencoder [Vincent et al., 2010, Seichter, 2015b].

En- und Decoders hinzugefügt. Diese ermöglichen die Konzentration tiefer gelegener Schichten auf abstraktere Merkmale und reduzieren den Druck zur Gewährleistung einer möglichen Rekonstruktion. Ähnlich wie beim herkömmlichen Autoencoder wird die rekonstruierte Repräsentation in jeder Schicht mit Hilfe eines Fehlers bewertet. Während des Trainingsprozesses wird letztlich die Gesamtsumme dieser Rekonstruktionskosten und der Kreuzentropie minimiert. Abbildung 3.6 veranschaulicht den schematischen Aufbau von einem Ladder Network.

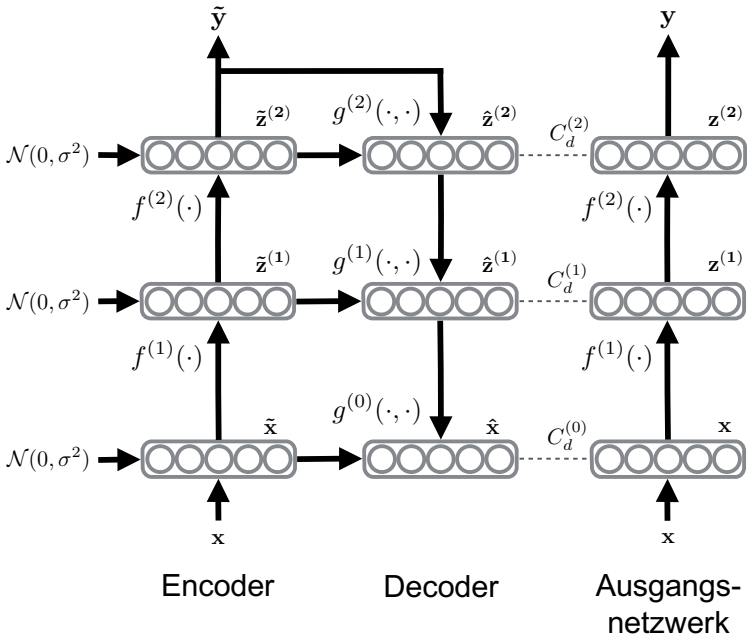


Abbildung 3.6: Struktur eines Ladder Networks nach [Rasmus et al., 2015a]
Dargestellt ist die Struktur eines Ladder Networks mit $L = 2$ vollverschalteten Schichten. Der Encoder wird auf Basis des Ausgangsnetzwerks durch das Kopieren jeder Schicht l und deren Abbildung $f^{(l)}(\cdot)$ gebildet. Im Encoder werden sowohl die Eingaben $\underline{\mathbf{x}}$ als auch die gelernten Repräsentationen $\underline{\mathbf{z}}^{(l)}$ durch das Hinzufügen von normalverteiltem Rauschen $N(0,\sigma^2)$ zu $\underline{\tilde{\mathbf{x}}}$ beziehungsweise $\underline{\tilde{\mathbf{z}}}^{(l)}$ verändert. Die einzelnen Decoderfunktionen $g(\cdot,\cdot)$ erzeugen die Rekonstruktionen $\underline{\hat{\mathbf{x}}}$ beziehungsweise $\underline{\hat{\mathbf{z}}}^{(l)}$, welche anschließend mit je einem Kostenterm $C_d^{(l)}$ bewertet werden. Zur Erzeugung der dafür benötigten unverrauschten Referenzgrößen ist ein zweiter Vorwärtsprozess nötig. Die verrauschte Ausgabe des Netzwerks $\underline{\tilde{\mathbf{y}}}$ wird zur Berechnung der Kreuzentropie genutzt. Die Klassifikation erfolgt anhand der unverrauschten Ausgabe $\underline{\mathbf{y}}$. [Rasmus et al., 2015a]

Für weiterführende Erläuterungen zu der Bedeutung der einzelnen Bestandteile sei neben der Hauptpublikation auf [Pezeshki et al., 2016] verwiesen. In dieser Veröffentlichung wird der Einfluss der einzelnen Komponenten systematisch evaluiert.
Der Aufbau von Ladder Networks ermöglicht lokales Lernen in den einzelnen Schichten des Netzwerks, auch wenn keine Labelinformationen verfügbar sind und macht Pretraining überflüssig. Anhand von umfangreichen Experimenten zeigen die Autoren, dass auf diese Weise verschiedene State-of-the-Art-Netzwerke verbessert werden können. Als besonders effektiv hat sich der Einsatz von Ladder Networks herausgestellt, wenn die Zahl der Gewichte im Netzwerk die Anzahl der gelabelten Trainingsbeispiele um ein Vielfaches übersteigt.

3.3 Zusammenfassung und Erkenntnisse

Grundvoraussetzung für die Realisierung von intelligentem inkrementellem Lernen ist die Modellierung von Unsicherheit. Nur so können möglichst informative, neue Trainingsbeispiele ausgewählt und redundante Daten vermieden werden. Die Analyse des State of the Art hat gezeigt, dass es hierfür zwei Optionen in Form von approximierenden bayes'schen und nicht-bayes'schen Ansätzen gibt. Konkrete Vergleiche zwischen beiden Ansatzklassen, welche das Ableiten von Schlussfolgerungen zur Eignung für die Problemstellung in dieser Masterarbeit ermöglichen könnten, existieren jedoch nicht. Vielmehr spaltet die Fragestellung die Forscher in zwei Lager. Verfechter der bayes'schen Wahrscheinlichkeitstheorie versuchen Deep-Learning mit bekannten bayes'schen Verfahren zu kombinieren. Jedoch werden viele dieser approximierenden Ansätze ausschließlich auf kleinen Datensätzen präsentiert und skalieren unzureichend mit großen Datenmengen. Einzig die Modellierung von Unsicherheiten durch Dropout Variational Inference hat sich in mehreren Anwendungsbereichen als guter Kompromiss zwischen Aufwand und Ergebnis erwiesen. Viele Vertreter aus dem Deep-Learning-Bereich berufen sich daher auf den bisherigen Erfolg von Deep-Learning ohne bayes'sche Komponenten und verweisen auf alternative Methoden, wie etwa den Einsatz von Ensembles. Das Training mehrerer unabhängiger Netzwerke ist jedoch aufgrund der begrenzten Ressourcen und des limitierten Zeitrahmens nicht möglich.

Im Rahmen dieser Masterarbeit wird daher die Extraktion von Unsicherheitsinformationen auf Basis von Dropout Variational Inference und ergänzend durch den Einsatz eines Ensembles mit geteiltem Basisnetzwerk verfolgt. Es bleibt in dieser Arbeit zu zeigen, welche Vorgehensweise zur Modellierung von Unsicherheit besser geeignet ist.
In Abschnitt 3.2.3 (Seite 29) wurden verschiedene Akquirierungsfunktionen vorgestellt. Hierbei hat sich gezeigt, dass die vorgeschlagenen Maße die Unsicherheit in der Ausgabe repräsentieren können und dass das Netzwerk infolge einer Auswahl neuer Datenpunkte nach den vorgeschlagenen Funktionen deutlich effizienter als mit einer zufälligen Akquirierung lernt. In dieser Masterarbeit bleibt zu zeigen, ob die Anwendung auch auf Daten aus dem ASINVOS-Projekt übertragen werden kann und welche Akquirierungsfunktion hierfür am besten geeignet ist.
Im letzten Abschnitt der State-of-the-Art-Betrachtungen wurde recherchiert, welche Möglichkeiten sich für das Nachtraining des Klassifikators ergeben. Für den Fall von überwachtem Nachtraining wurden Empfehlungen aus [Käding et al., 2016] zusammengefasst. Diese sollten jedoch ausschließlich als Richtwerte angesehen werden. So ist es möglich, dass geeignete Parameter infolge von verschiedenartigen Netzwerkstrukturen und durch die Verwendung eines anderen Datensatzes abweichen. Zum Schluss der Betrachtungen wurde der aktuell erfolgversprechendste Ansatz auf Basis von Semi-Supervised Learning vorgestellt. Während die Grundidee von Ladder Networks in wenigen Sätzen beschrieben werden kann, ist die Realisierung mit sehr vielen Herausforderungen verbunden. So müssen beispielsweise sowohl die Parameter für das additive Rauschen als auch die Decoderfunktionen sorgfältig bestimmt werden, um Triviallösungen vorzubeugen. Zudem erfordert die Anwendung weitreichende Veränderungen an dem bisher im ASINVOS-Projekt verwendeten Klassifikatormodell. So müsste auf den Einsatz von Dropout verzichtet und das Netzwerk um zusätzliche Schichten zur Realisierung von Batch Normalization [Ioffe und Szegedy, 2015] erweitert werden. Aus [Eisenbach et al., 2017] ist jedoch bekannt, dass sich beide Modifikationen negativ auf die Klassifikationsleistung auswirken. Aus diesem Grund wird die Realisierung eines Nachtrainings mit Ladder Networks in dieser Masterarbeit nicht weiter verfolgt.

Kapitel 4

Daten und Vorverarbeitung

In diesem Teil der Masterarbeit werden die verwendeten Daten und deren Vorverarbeitung beschrieben. Im Rahmen eines standardisierten Verfahrens zur Zustandserfassung und -bewertung (ZEB) von Fernstraßen werden mit einem speziell ausgestatteten Fahrzeug der Firma LEHMANN+PARTNER GmbH verschiedenste Parameter der Straße bestimmt. Unter anderem wird die Oberflächenbeschaffenheit der Fahrbahn durch zwei hochauflösende monochrome Kameras aufgenommen. Durch den Einsatz von zusätzlichen Blitzlichtern soll sichergestellt werden, dass die Aufnahmen stets ähnlich ausgeleuchtet sind. Die mit diesem Fahrzeug fortwährend aufgenommenen Full-HD-Grauwertbilder bilden die Datengrundlage für das am Fachgebiet bearbeitete ASINVOS-Projekt[1] und für diese Masterarbeit.

4.1 Zu unterscheidende Schadklassen

Im ASINVOS-Projekt sollen anhand der aufgenommenen Bilder fünf verschiedene Schadklassen von intakten Straßenausschnitten abgegrenzt werden. Neben typischen Schäden wie Rissen oder großflächigen Ausbrüchen sollen ebenfalls offene Arbeitsnähte erkannt werden. Diese können an den Übergängen zwischen einzelnen Fertigungsabschnitten entstehen und verlaufen daher meist geradliniger als Risse und stets parallel

[1] ASINVOS - Assistierendes und interaktiv lernfähiges Videoinspektionssystem für Oberflächenstrukturen am Beispiel von Straßenbelägen und Rohrleitungen

oder senkrecht zum Fahrbahnrand. Weiterhin sollen bereits ausgebesserte Stellen, sogenannte Flickstellen, erfasst werden. Hierbei wird zwischen aufgelegten und eingelegten Flickstellen unterschieden. Letztere zeichnen sich dadurch aus, dass Bereiche der Straße mit geradlinigen Begrenzungen entfernt und anschließend wieder aufgefüllt wurden. Die Stöße werden zudem meist mit flüssigem Bitumen ausgegossen. In Abbildung 4.1 werden die zu unterscheidenden Klassen anhand von Beispielen veranschaulicht.

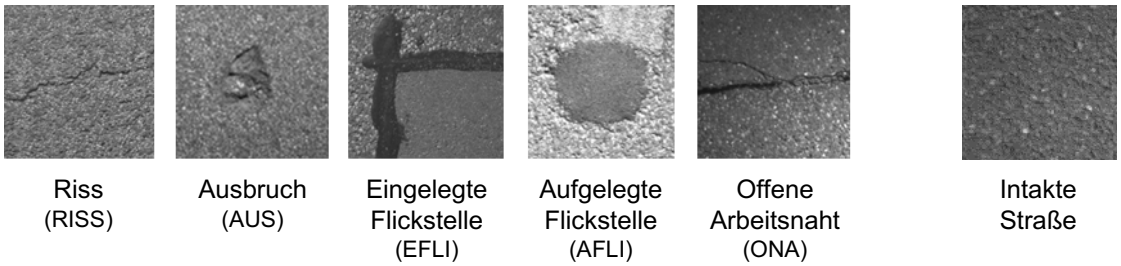


Abbildung 4.1: Schadklassen

Die Abbildung zeigt die im ASINVOS-Projekt von intakten Straßenausschnitten zu unterscheidenden Schadklassen sowie deren Kurzbezeichnung.

Der Fokus in dieser Masterarbeit liegt jedoch nicht auf der korrekten Klassifizierung einzelner Schadensmuster, sondern auf der Erkennung und Lokalisierung von Schäden. Aus diesem Grund werden ausschließlich Zweiklassen-Probleme betrachtet: intakter Straßenausschnitt (Klasse 0) versus Straßenausschnitt mit Schaden (Klasse 1).

4.2 GAPs-Datensatz

Am Fachgebiet Neuroinformatik und Kognitive Robotik wurde im Rahmen der Projektbearbeitung der sogenannte GAPs-Datensatz[2] erstellt und in [Eisenbach et al., 2017] publiziert. Dieser Datensatz basiert auf exakten Schadstellenmarkierungen und ermöglicht somit sowohl das Training als auch den Vergleich von bildausschnittbasierten Klassifikatoren für die Erkennung von Schadstellen auf Fernstraßen. Im Anschluss an die Publikation wurde die Genauigkeit des Datensatzes nochmals verbessert. In dieser Masterarbeit wird ausschließlich diese überarbeitete Version verwendet.

[2] German Asphalt Pavement Distress (GAPs) Datensatz
`https://www.tu-ilmenau.de/neurob/data-sets-code/gaps/` (Zuletzt aufgerufen: 01.05.2017)

Für die Erstellung des GAPs-Datensatzes wurden insgesamt 2.469 Full-HD-Bilder von vier Bundesstraßen händisch mit Labelinformationen versehen. Schadhafte Stellen wurden dabei möglichst genau mit Hilfe von Bounding-Boxen markiert. Auf dieselbe Weise wurden ebenfalls zu ignorierende Randbereiche und Straßenmarkierungen gekennzeichnet. Abbildung 4.2 zeigt zwei dieser händisch gelabelten Full-HD-Bilder. Diese bilden die Grundlage für die Extraktion kleiner Bildausschnitte (engl. Patches), welche für das Training eines Klassifikators zur Erkennung und Lokalisierung von Schäden verwendet werden können.

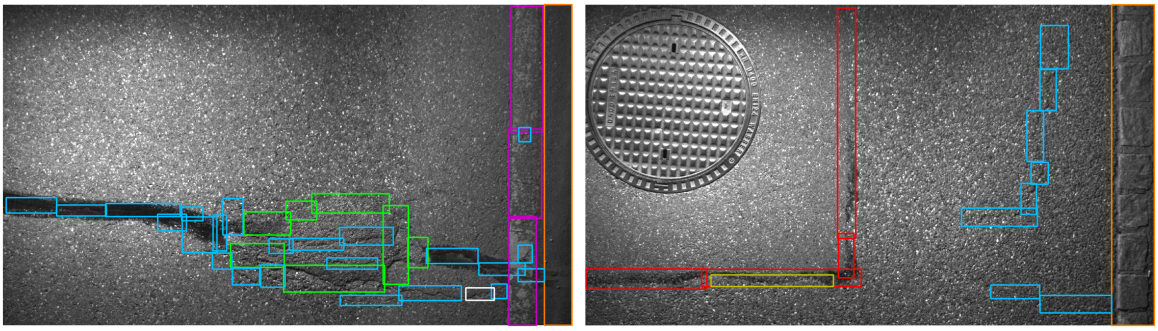


Abbildung 4.2: Gelabelte Full-HD-Bilder für den GAPs-Datensatz
Dargestellt sind zwei gelabelte Full-HD-Bilder, welche für die Extraktion von Patches herangezogen werden. Die Labelinformationen sind durch farbige Bounding-Boxen gekennzeichnet, wobei diese die folgenden Bedeutungen haben: aufgelegte Flickstelle (grün), eingelegte Flickstelle (rot), Ausbruch (weiß), offene Arbeitsnaht (gelb), Riss (blau), Straßenmarkierung (violett) und zu ignorierender Randbereich (orange).

Der Algorithmus zur Extraktion der Bildausschnitte unterstützt verschiedene Patchgrößen. In Anlehnung an [Eisenbach et al., 2017][3] werden für diese Masterarbeit Patches mit einer Größe von 64x64 Pixeln verwendet. Während Beispiele für intakte Straßenverhältnisse an zufälligen Positionen außerhalb der markierten Bereiche und mit halber Überlappung (32 Pixel) gesampelt werden, erfolgt die Extraktion von Beispielen für Schäden, ausgehend von der oberen linken Ecke der Bounding-Boxen, mit einer Verschiebung von acht Pixeln. Ist eine Bounding-Box kleiner als die verwendete Patchgröße, wird diese mittig im Patch platziert und der fehlende Bereich aus dem Bild

[3] Die Publikation enthält eigene Beiträge zur Festlegung der Patchgröße.

übernommen. An den Rändern des Bildes wird dabei mit einem mittleren Grau aufgefüllt. Überlappende Markierungen werden nach vorher definierten Prioritäten anhand der Gesamthäufigkeiten der Schadklassen abgearbeitet.

In Anlehnung an [Ng, 2016] werden die Patches nach der Extraktion anhand der Ausgangsbilder in vier Teildatensätze unterteilt. Das Bildmaterial der ersten beiden Bundesstraßen wird für die Erstellung eines Trainings- und eines Validierungsdatensatzes genutzt. Der Großteil des Materials entfällt hierbei mit 1.418 Bildern auf den Trainingsdatensatz. Der Validierungsdatensatz wird aus 51 Bildern von einem separaten Abschnitt von einer der beiden Bundesstraßen gebildet. Nach [Ng, 2016] dient dieser Validierungsdatensatz ausschließlich der Erkennung von Overfitting und folglich zur Festlegung eines geeigneten Abbruchzeitpunktes. Das Material der verbleibenden beiden Bundesstraßen wird zur Generierung von Testdaten genutzt. Diese werden in einen Test-Validierungsdatensatz und einen Testdatensatz untergliedert. Beide Teildatensätze basieren auf 500 Bildern von jeweils einer der beiden Bundesstraßen. Während der Test-Validierungsdatensatz für Vergleiche von unterschiedlichen Ansätzen genutzt werden kann, bleibt der Testdatensatz meist unter Verschluss und soll während der Erstellung des Klassifikators nicht genutzt werden.

Da der komplette GAPs-Datensatz eine enorm große Anzahl an Patches umfasst und somit das Experimentieren mit verschiedenen Architekturen und Parametern erschwert, wurde für jeden Teildatensatz mit Ausnahme des Testdatensatzes eine Version mit geringerem Umfang erstellt. Um zusätzlich die Repräsentationskraft zu erhöhen, wurde dabei auf eine bessere Balancierung zwischen den Klassen für Schäden und intakter Straße geachtet. In Anlehnung an den MNIST-Datensatz wurde ein Trainingsdatensatz mit 50.000 zufälligen Beispielen definiert. Für die beiden Validierungsdatensätze wurde ein Umfang von 10.000 Beispielen gewählt. Im Rahmen dieser Masterarbeit werden ausschließlich die im Umfang reduzierten Datensätze für das Training und die Bewertung eines Klassifikators verwendet. Tabelle 4.1 zeigt die genauen Klassenverteilungen in den genutzten Teildatensätzen.

Vor dem Training erfolgt zudem eine Vorverarbeitung in Form einer Skalierung des Wertebereichs der einzelnen Pixel. In Anlehnung an [Eisenbach et al., 2016] wird der ursprüngliche Wertebereich von $[0, 255]$ hierfür auf das Intervall $[-1, 1]$ skaliert. Mit

einem mittleren Grau aufgefüllte Randbereiche führen so, ähnlich wie die Anwendung von Dropout, zu einer Eingabe von Null.

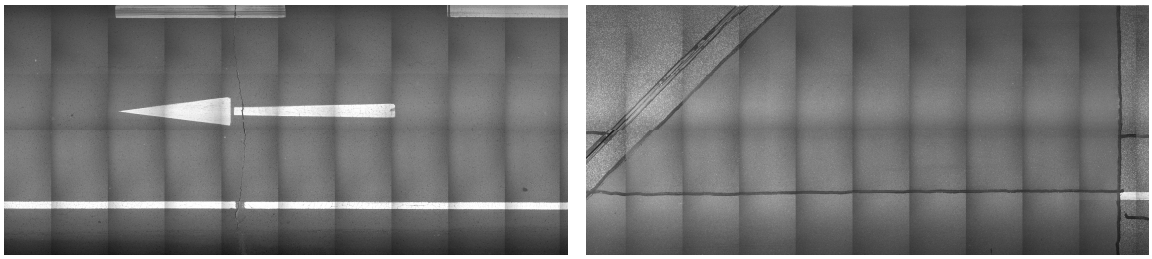
Klasse	Teildatensatz		
	Training (50k)	Validierung (10k)	Test-Validierung (10k)
Intakt	30.000	6.000	6.000
Schaden	20.000	4.000	4.000
∟ AFLI	5.000	1.000	1.000
∟ AUS	1.500	300	300
∟ EFLI	2.500	500	500
∟ ONA	1.000	200	200
∟ RISS	10.000	2.000	2.000

Tabelle 4.1: GAPs-Datensatz

Dargestellt sind die Klassenverteilungen in den verwendeten Teildatensätzen. Für eine Erläuterung der Kurzbezeichnungen sei auf Abbildung 4.1 verwiesen.

4.3 Fahrbahnbilder

Das bundesweit verwendete Verfahren zur Zustandserfassung und -bewertung (ZEB) sieht eine regelmäßige visuelle Beurteilung der Fahrbahnoberfläche vor. Die mit Hilfe des Aufnahmefahrzeuges erstellten Full-HD-Bilder werden hierfür zu kompletten Fahrbahnbildern zusammengesetzt und von geschulten Operatoren durch die Vergabe von Labeln bewertet. Die Fahrbahn wird dabei zunächst anhand eines groben Rasters in mehrere Abschnitte untergliedert. Anschließend wird jeder Bildausschnitt den in Abschnitt 4.1 vorgestellten Klassen zugeordnet. Auf diese Weise kann der Zustand der Fernstraßen mit vertretbarem Aufwand bestimmt und, falls nötig, die Instandsetzung einzelner Abschnitte eingeleitet werden. Durch die permanente Erfassung der Straßen wurden seit der Einführung des ZEB-Verfahrens große Datenbestände aufgenommen und archiviert, welche für die Realisierung eines inkrementellen Lernverfahrens herangezogen werden könnten. In dieser Masterarbeit wird anhand von 1.637 Fahrbahnbildern exemplarisch untersucht, ob diese Bestände zur Verbesserung eines Klassifikators genutzt werden können. Abbildung 4.3 zeigt zwei der verfügbaren Fahrbahnbilder.



(a) Fahrbahnbild mit erkennbarem Riss **(b)** Fahrbahnbild mit erkennbarer Flickstelle

Abbildung 4.3: Beispielfahrbahnbilder

Dargestellt sind zwei der für diese Masterarbeit verfügbaren 1.637 Fahrbahnbilder.

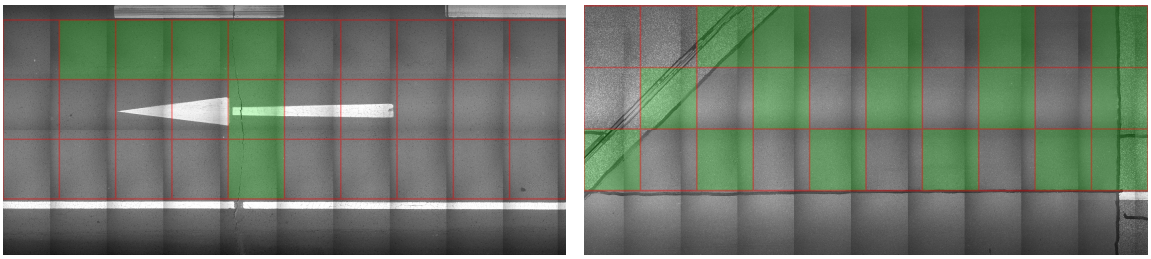
Im Folgenden wird noch einmal genauer auf das Verfahren zur Erstellung der Fahrbahnbilder eingegangen. Anschließend wird der Prozess der Labelvergabe näher betrachtet. Hierbei werden vor allem Besonderheiten herausgearbeitet und daraus resultierende Probleme abgeleitet.

4.3.1 Zusammensetzung aus Einzelbildern

Die mit Hilfe der beiden Kameras aufgenommenen Einzelbilder werden für die Erstellung der Fahrbahnbilder fortlaufend in einer zweireihigen Anordnung aneinander gesetzt. Da sich die einzelnen Bilder geringfügig überlappen, können gleiche Bildanteile gesucht und das komplette Bild anhand dieser durch einen Stitching-Algorithmus zusammengesetzt werden. Im letzten Schritt wird die so erstellte Abbildung der kompletten Fahrbahn in feste Abschnitte mit einer Länge von zehn Metern untergliedert. In der Folge entstehen die für die Beurteilung und in dieser Masterarbeit verwendeten Fahrbahnbilder. Die Anzahl der für ein Fahrbahnbild benötigten Einzelbilder variiert mit der Geschwindigkeit des Aufnahmefahrzeuges. Die für diese Masterarbeit verfügbaren Bilder basieren auf 22 bis 24 Einzelbildern und besitzen eine Auflösung von 7883x3498 Pixeln. Das verwendete Verfahren besitzt jedoch einen Makel. Bedingt durch eine ungleichmäßige Ausleuchtung der einzelnen Full-HD-Bilder entstehen bei der Zusammensetzung deutlich sichtbare Kontrastkanten. Für die in Abbildung 4.3 gezeigten Beispielbilder sind diese trotz reduzierter Auflösung gut erkennbar. Um Fehlklassifikationen infolge einer starken Ähnlichkeit zu Rissen vorzubeugen, sollten diese Bildbereiche im Vorfeld maskiert und von der Verarbeitung ausgeschlossen werden.

4.3.2 Labelinformationen

Die Vergabe von Labelinformationen erfolgt in zwei Schritten. Zunächst platziert der Operator ein aus drei Zeilen und zehn Spalten bestehendes Raster über dem fahrbahnrelevanten Teil des Bildes. Das Raster wird dabei primär anhand von Fahrbahnmarkierungen und Fahrbahnbegrenzungen, wie Bordsteinen, Geh- und Radwegen oder deutlich erkennbaren Fahrbahnrändern, ausgerichtet. Enthält das Bild keine derartigen Strukturen, wird vom Bildrand als Begrenzung ausgegangen. Anschließend werden für den kompletten Bildausschnitt jedes Rasterelementes Label vergeben. Jeder Ausschnitt kann dabei einer oder mehreren Schadklassen zugeordnet werden. Zeigt der betrachtete Bildbereich keinen Schaden, wird kein Label vergeben. Abbildung 4.4 verdeutlicht die Labelvergabe für die zuvor in Abbildung 4.3 vorgestellten Fahrbahnbilder.

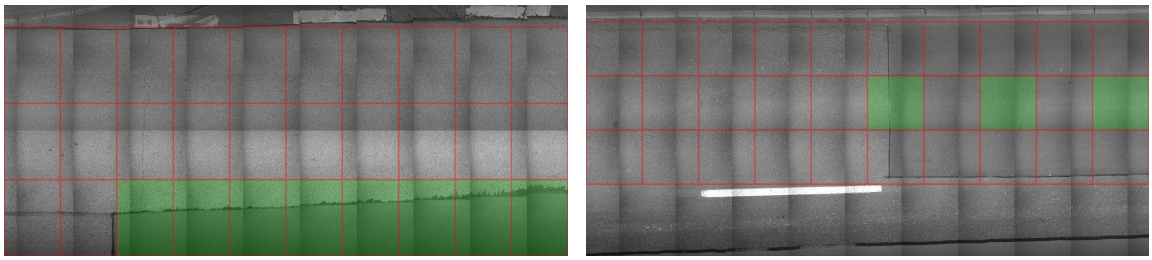


(a) Labelinformationen für Klasse RISS **(b)** Labelinformationen für Klasse EFLI

Abbildung 4.4: Beispielfahrbahnbilder mit Labelinformationen
Die Abbildung zeigt die Labelinformationen für die zuvor in Abbildung 4.3 präsentierten Fahrbahnbilder. Das beschriebene Labelraster ist in beiden Bildern rot überlagert dargestellt. Weiterhin sind die von den Operatoren markierten Bereiche für ausgewählte Schadklassen grün überlagert hervorgehoben. Für eine Erläuterung der Abkürzungen der Schadklassen sei auf Abbildung 4.1 (Seite 40) verwiesen.

In dem in Abbildung 4.4a dargestellten Fahrbahnbild befindet sich neben dem sichtbaren vertikalen Riss ein weiterer, in dieser Auflösung nicht erkennbarer, horizontaler Riss, der sich über drei Rasterelemente erstreckt. Die vergebenen Label sind daher korrekt und nachvollziehbar. Die markierten Bereiche für die Flickstelle in Abbildung 4.4b wirken hingegen unintuitiv und werfen Fragen auf. An dieser Stelle ist eine Besonderheit bei der Labelvergabe für Flickstellen dargestellt. Das ZEB-Verfahren sieht für

diese beiden Schadklassen nicht die Markierung der Übergänge, sondern die Abbildung der Fläche der Flickstellen mit Hilfe von Rasterelementen vor. In der Folge kann es zu sogenannten Flächenausgleichen zwischen einzelnen Rasterelementen mit dem Ziel kommen. Diese haben das Ziel, dass die markierte Gesamtfläche möglichst genau der Fläche der Flickstelle entspricht. Die letztlich markierten Bereiche sind jedoch in vielen Fällen dennoch nicht nachvollziehbar. Abbildung 4.5 verdeutlicht dies an zwei weiteren Beispielen. Während die Labelvergabe für das linke Fahrbahnbild plausibel erscheint, sind die markierten Rasterelemente im rechten Bild nicht nachvollziehbar und lassen sich auch nicht durch einen Flächenausgleich erklären. Die für diese beiden Klassen verfügbaren Labelinformationen sind daher nur bedingt für ein inkrementelles Lernverfahren nutzbar.



(a) Labelinformationen für Klasse AFLI **(b)** Labelinformationen für Klasse EFLI

Abbildung 4.5: Beispielfahrbahnbilder mit Labelinformationen
Dargestellt sind die Labelinformationen für zwei weitere Fahrbahnbilder. Das beschriebene Labelraster ist in beiden Bildern rot überlagert dargestellt. Weiterhin sind die von den Operatoren markierten Bereiche für ausgewählte Schadklassen grün überlagert hervorgehoben. Für eine Erläuterung der Abkürzungen der Schadklassen sei auf Abbildung 4.1 (Seite 40) verwiesen.

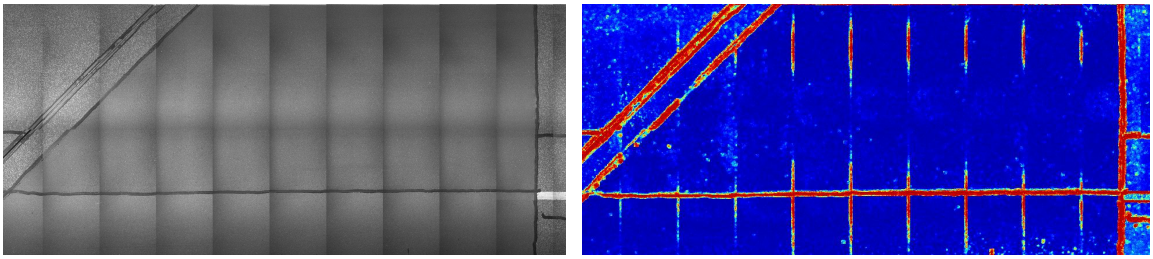
Ein weiteres Problem besteht in dem verwendeten Speicherformat der Labelinformationen. Da mit Hilfe des ZEB-Verfahrens nur allgemeine Aussagen zum Zustand der Fahrbahn getroffen werden sollen, sieht das genutzte Format nicht die Speicherung der Position des Labelrasters vor. In der Folge muss für eine erneute Verwendung der Labelinformationen zunächst die Lage des Rasters bestimmt werden. Im Rahmen dieser Masterarbeit wurde hierfür ein Algorithmus entwickelt, welcher im Anschluss als Teil der Vorverarbeitung vorgestellt wird.

4.4 Vorverarbeitung der Fahrbahnbilder

Bevor die verfügbaren Fahrbahnbilder für einen inkrementellen Lernprozess genutzt werden können, müssen die beschriebenen Probleme im Rahmen einer Vorverarbeitung gelöst werden. So müssen die ungewollten Kontrastkanten im Bild erkannt und deren Bereiche maskiert werden. Zudem können die verfügbaren Labelinformationen nur dann genutzt werden, wenn die Lage des Labelrasters zuvor bestimmt wurde. In diesem Abschnitt werden die hierfür entwickelten Algorithmen beschrieben.

4.4.1 Lokalisierung der Stitching-Kanten

Bedingt durch eine ungleichmäßige Ausleuchtung entstehen bei der Zusammensetzung (engl. Stitching) zu kompletten Fahrbahnbildern deutlich sichtbare Kontrastkanten, welche vom Klassifikator fälschlicherweise als Schaden erkannt werden könnten. Zu Beginn der Masterarbeit wurde daher der Einfluss der Kanten anhand verschiedener Fahrbahnbilder untersucht. Abbildung 4.6 zeigt die Klassifikatorausgabe für ein ausgewähltes Beispiel.



(a) Fahrbahnbild **(b)** Klassifikatorausgabe Klasse Schaden

Abbildung 4.6: Fehldetektionen infolge von Stitching-Kanten
Dargestellt ist die Klassifikatorausgabe für ein Beispielfahrbahnbild zur Untersuchung des Einflusses der Stitching-Kanten. Rot eingefärbte Bereiche symbolisieren eine Detektion als Schaden.

Es ist deutlich zu erkennen, dass vor allem Kanten in dunklen Bildbereichen zu Fehlklassifikationen führen. In der Folge könnten die verfügbaren Labelinformationen falsch interpretiert und der Datensatz mit inkorrekten Beispielen erweitert werden. Daher müssen diese Stitching-Kanten detektiert und die betreffenden Bildbereiche vorab

maskiert werden. Im Folgenden wird der hierfür umgesetzte Algorithmus beschrieben. Anschließend werden die durchgeführten experimentellen Untersuchungen zur Parameteridentifikation vorgestellt.

4.4.1.1 Algorithmus

Das in dieser Masterarbeit realisierte Verfahren zur Erkennung der Stitching-Kanten basiert auf der Anwendung eines Sobel-Kantenfilters und einer anschließenden problemspezifischen Auswertung der Kantenbilder. Der verfolgte Ansatz setzt eine konstante Aufnahmegeschwindigkeit voraus, unterstützt jedoch beliebige absolute Geschwindigkeiten. Nachfolgend wird der Algorithmus in Textform und anhand der in Abbildung 4.7 dargestellten Beispielbildern beschrieben. Genaue Implementierungsdetails können der anschließenden Formulierung als Pseudocode in Abbildung 4.10 (Seite 54) entnommen werden.

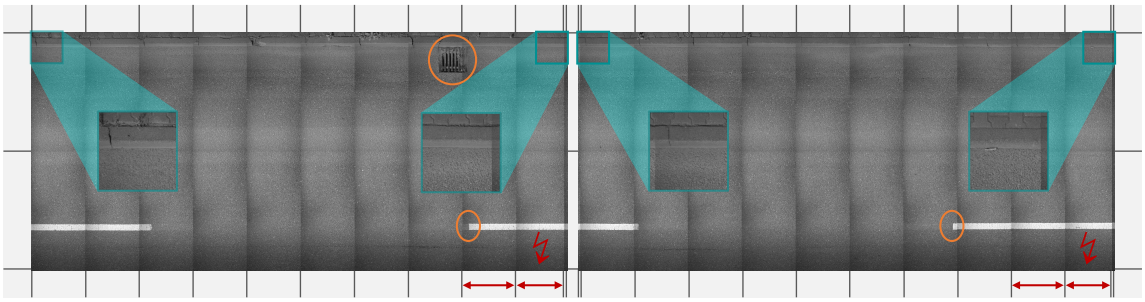


Abbildung 4.7: Beispielfahrbahnbilder
Dargestellt sind zwei der für diese Masterarbeit verfügbaren Fahrbahnbilder zur Erläuterung des realisierten Algorithmus zur Erkennung der Stitching-Kanten. Die zusätzlichen Markierungen werden Schritt für Schritt im Fließtext erklärt.

Teilung in zwei Bildhälften

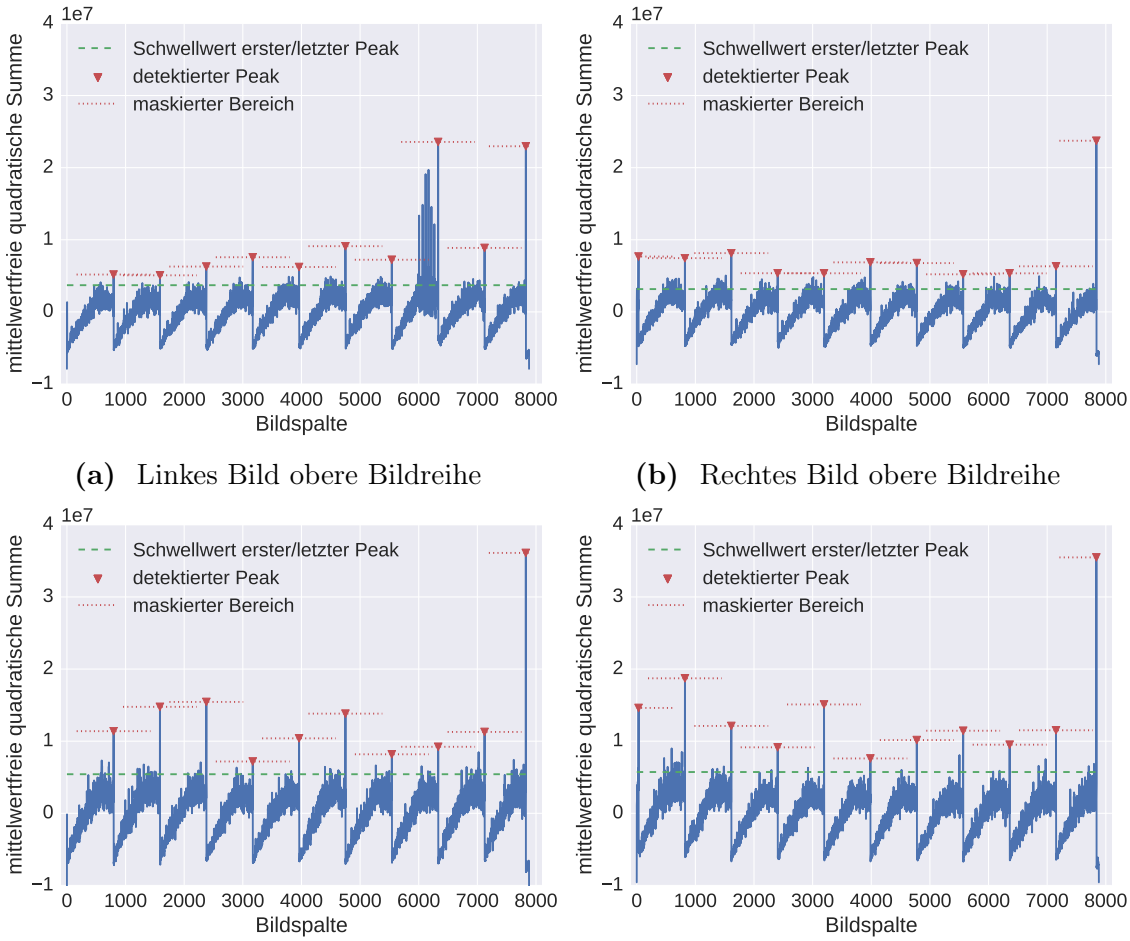
Die Fahrbahnbilder werden aus fortlaufend aufgenommenen Einzelbildern, angeordnet in zwei Reihen, zusammengesetzt. Während der Erstellung des Algorithmus hat sich gezeigt, dass das vertikale Raster in beiden Bildreihen getrennt voneinander erkannt werden sollte. Durch eine anschließende Verrechnung beider Ergebnisse können auf

diese Weise fehlerhafte Kantendetektionen infolge von Artefakten, wie zum Beispiel den in Abbildung 4.7 orange markierten Abfluss oder Straßenmarkierungen, erkannt und ausgebessert werden. Dies erhöht die Fehlertoleranz, setzt jedoch auch eine vorab erfolgte horizontale Teilung des Bildes voraus. Für die in dieser Masterarbeit verwendeten Fahrbahnbilder hat sich gezeigt, dass sich die horizontale Stitching-Kante stets in der Bildmitte befindet und das Bild daher sehr leicht geteilt werden kann. Alternativ könnte jedoch auch eine Detektion mit Hilfe eines horizontalen Sobel-Filters angewendet werden.

Getrennte Kantenerkennung für jede Bildhälfte

Im Anschluss an die horizontale Teilung erfolgt für jede Bildhälfte getrennt die Lokalisierung der vertikalen Stitching-Kanten. Hierzu wird zunächst ein entsprechender Sobel-Filter angewendet. Anschließend werden die Summen der einfachen und die der quadrierten Elemente in jeder Spalte des Kantenbildes berechnet und in eine mittelwertfreie Form überführt. Diese Summen bilden die Grundlage für den weiteren Lokalisierungsprozess und werden im Folgenden vereinfacht als Summen beziehungsweise als quadratische Summen bezeichnet. Aufgrund des verwendeten Erstellungsprozesses ist vorab weder die Anzahl noch die Anordnung der Kanten bekannt. Wie in Abbildung 4.7 anhand eines Rasters im Hintergrund und vergrößerten Ausschnitten verdeutlicht, verschieben sich die Kanten von Bild zu Bild, sodass, je nach Lage, zehn oder elf Stitching-Kanten existieren. Der realisierte Algorithmus sieht daher eine Approximation des mittleren Kantenabstandes mit Hilfe der Fourier-transformierten Darstellung der einfachen Summen vor. Durch die Bestimmung der dominierenden Frequenz und einen zuvor festgelegten Skalierungsfaktor kann ein minimaler und ein maximaler Abstand abgeleitet werden. Bei der Festlegung dieses Skalierungsfaktors muss berücksichtigt werden, dass der Kantenabstand am rechten Bildrand aus nicht nachvollziehbaren Gründen abweichen kann (rot markierte Segmente in Abbildung 4.7). Die Lokalisierung der Kanten erfolgt letztlich durch eine Peak-Erkennung in den quadratischen Summen. Beginnend beim globalen Maximum werden dabei, unter Zuhilfenahme des minimalen Abstandes, die Bereiche um alle nach dem Funktionswert absteigend sortierten Maxima maskiert, bis alle Bildspalten bearbeitet wurden. Da auf diesem Wege

möglicherweise Bereiche am Bildrand nicht korrekt maskiert werden können, erfolgt sowohl für den ersten als auch für den letzten Peak eine abschließende Plausibilitätsprüfung anhand des aus den quadratischen Summen an den übrigen Kantenpositionen gebildeten Medians und unter Zuhilfenahme des approximierten maximalen Abstandes. Sollten sich hierbei Unstimmigkeiten ergeben, wird das komplette Raster für diese Bildhälfte verworfen. In diesem Fall verringert sich die Anzahl der verfügbaren Raster für die anschließende Verrechnung. Abbildung 4.8 verdeutlicht die zuletzt beschriebene Peak-Erkennung anhand der Fahrbahnbilder aus Abbildung 4.7.



(a) Linkes Bild obere Bildreihe

(b) Rechtes Bild obere Bildreihe

(c) Linkes Bild untere Bildreihe

(d) Rechtes Bild untere Bildreihe

Abbildung 4.8: Visualisierung der Peak-Erkennung

Die Abbildung visualisiert die Peak-Erkennung für die in Abbildung 4.7 gezeigten Fahrbahnbilder.

Die durch den Abfluss im letzten Drittel des linken Fahrbahnbildes erzeugten vertikalen Kanten zeigen sich deutlich in den quadratischen Summen (Abbildung 4.8a). In diesem Fall besitzen sie jedoch keine Auswirkung, da die angrenzende Stitching-Kante einen minimal höheren Peak erzeugt. Weiterhin wird deutlich, dass der erste Peak in Abbildung 4.8a und 4.8c das Potenzial für eine Fehldetektion bietet (siehe vergrößerter Ausschnitt für diesen Bereich in Abbildung 4.7, Seite 48), welche jedoch durch die abschließende Plausibilitätsprüfung vermieden werden kann.

Verrechnung der erkannten Raster

Nachdem die Stitching-Kanten für beide Bildhälften detektiert wurden, erfolgt in diesem Schritt die Verrechnung zu einem finalen Raster. Je nach Ausgangspunkt müssen hierbei drei Fälle unterschieden werden. Wurden zuvor alle Raster verworfen, endet der Prozess ohne Ergebnis. Steht nur noch ein Raster zur Verfügung, wird dieses als finales Detektionsergebnis angenommen. Andernfalls erfolgt eine Verrechnung. Dafür wird zuerst die Anzahl der detektierten Stitching-Kanten verglichen. Unterscheiden sich diese, wird die Welligkeit[4] als ein Maß für die Unterschiedlichkeit der Abstände zwischen den detektierten Kantenpositionen analysiert. Hierzu wird eine maximale Welligkeit definiert und das Raster, welches diese am stärksten übersteigt, eliminiert. Sollte dieser Grenzwert in keinem der beiden Fälle überschritten werden, können beide Möglichkeiten als plausibel angesehen werden und somit automatisiert kein finales Raster abgeleitet werden. Wurde ein Raster verworfen, endet der Prozess und das Detektionsergebnis wird durch das verbleibende Raster definiert. Andernfalls ist nun sichergestellt, dass die verbleibenden Raster die gleiche Länge besitzen. Folglich können die einzelnen Kantenpositionen verglichen werden. Für jede Kante wird hierzu die Standardabweichung aus den beiden Position berechnet. Überschreitet eine dieser Standardabweichungen einen vorher definierten Maximalwert, wird das Raster mit der höchsten Welligkeit verworfen. Sollten auch nach dieser Überprüfung zwei Raster verbleiben, wird jeweils der Mittelwert aus den beiden Kantenpositionen gebildet und so das finale Raster bestimmt. Abbildung 4.9 zeigt die finalen Detektionsergebnisse für die in Abbildung 4.7 (Seite 48) präsentierten Fahrbahnbilder.

[4] Die Welligkeit kann als Summe der absoluten Werte der zweiten Ableitung berechnet werden.

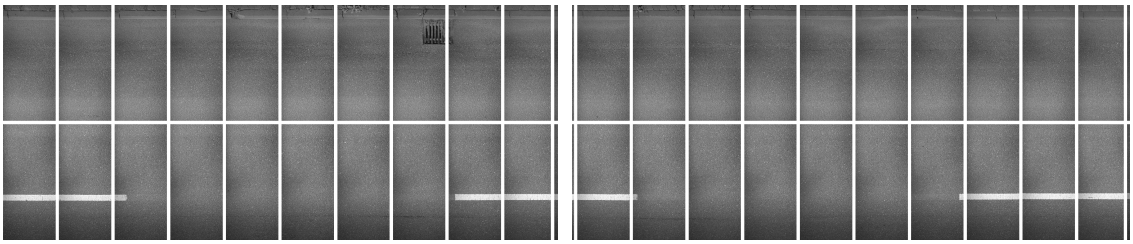


Abbildung 4.9: Detektionsergebnisse

Dargestellt sind die Detektionsergebnisse für die Fahrbahnbilder aus Abbildung 4.7.

Angabe der Sicherheit bei der Detektion

Der realisierte Algorithmus unterstützt neben der Lokalisierung der Stitching-Kanten auch das Ableiten einer Detektionssicherheit. Auf Basis dieser können Operatoren eingebunden oder bestimmte Bilder für eine automatisierte Weiterverarbeitung ausgeschlossen werden. Die Sicherheit wird dabei in die folgenden drei Stufen eingeteilt:

Stufe 0

Die Lage der Stitching-Kanten konnte nicht bestimmt werden. In diesem Fall könnten die detektierten Raster der einzelnen Bildhälften als Ausgangspunkt für das Einbinden eines Operators genutzt werden.

Stufe 1

Die Lage der Stitching-Kanten konnte bestimmt werden, jedoch übersteigt die Welligkeit des finalen Rasters den vordefinierten Maximalwert. Es handelt sich folglich um eine unsichere Detektion, deren Ergebnis einem Operator zur Validierung präsentiert werden könnte.

Stufe 2

Die Lage der Stitching-Kanten konnte bestimmt werden und die Welligkeit des finalen Rasters ist kleiner als der vordefinierte Maximalwert. Detektionen dieser Art können als sicher angesehen werden und bedürfen keiner Validierung.

Für den kompletten Lokalisierungsprozess werden pro Fahrbahnbild unter Verwendung einer mittelpreisigen Grafikkarte, wie beispielsweise einer NVIDIA GTX 1080, durchschnittlich 204 Millisekunden benötigt.

Formulierung als Pseudocode

Nachfolgend wird der vorgestellte Algorithmus in Form von Pseudocode beschrieben. Die dabei verwendete Notation ist an Python-Syntax angelehnt. Indizierungen beginnen daher stets bei 0.

Eingaben

1 $\underline{\mathbf{X}} \in \{0, 1, \ldots, 255\}^{3498 \times 7883}$ *// Fahrbahnbild*

2 $c_{peaks} = 0{,}80$ *// Skalierungsfaktor für approx. Abstand zwischen zwei Peaks*

3 $th_{peaks} = 0{,}50$ *// relative Mindesthöhe für Peaks in Plausibilitätsprüfung*

4 $w_{max} = 10{,}00$ *// maximale Welligkeit in Pixeln*

5 $\sigma_{max} = 2{,}00$ *// maximale Standardabweichung in Pixeln*

Algorithmus

Teilung in zwei Bildhälften:

6 `Horizontale Teilung des Fahrbahnbildes:`
$\underline{\mathbf{X}}^{(oben)}, \underline{\mathbf{X}}^{(unten)} \in \{0, 1, \ldots, 255\}^{1749 \times 7883}$

Getrennte Kantenerkennung für jede Bildhälfte:

7 `Faltung mit Sobel-Filter` $\underline{\mathbf{F}}$`:` $\underline{\mathbf{Y}}^{(\cdot)} = \mathrm{conv}(\underline{\mathbf{X}}^{(\cdot)}, \underline{\mathbf{F}})$

8 `Elementweise Bestimmung der mittelwertfreien Summen` $\underline{\mathbf{s}}, \underline{\mathbf{s}}^2 \in \mathbb{N}^{1749}$`:`
$\underline{\mathbf{s}} = \underline{\tilde{\mathbf{s}}} - \mathrm{mean}(\underline{\tilde{\mathbf{s}}})$ `mit:` $\underline{\tilde{\mathbf{s}}}[j] = \mathrm{sum}(\underline{\mathbf{X}}[:, j]) \quad \forall j \in [0, \ldots, 7883 - 1]$
$\underline{\mathbf{s}}^2 = \underline{\tilde{\mathbf{s}}}^2 - \mathrm{mean}(\underline{\tilde{\mathbf{s}}}^2)$ `mit:` $\underline{\tilde{\mathbf{s}}}^2[j] = \mathrm{sum}(\underline{\mathbf{X}}^2[:, j]) \quad \forall j \in [0, \ldots, 7883 - 1]$

9 `Approximation des minimalen und maximalen Rasterabstandes` d_{min} `bzw.` d_{max}
`mittels reelwertiger FFT unter Vernachlässigung der DC-Komponente:`
1. `Anwendung der reelwertigen FFT:` $\underline{\mathbf{u}} = \mathrm{abs}(\mathrm{rfft}(\underline{\mathbf{s}}))$
2. `Bestimmung der dominierenden Frequenz:` $i_{max} = \mathrm{argmax}(\underline{\mathbf{u}}[1:])$
3. `Ableitung eines minimalen Rasterabstandes:` $d_{min} = c_{peaks} \cdot {}^{7883}/_{i_{max}}$
4. `Ableitung eines maximalen Rasterabstandes:` $d_{max} = {}^{1}/_{c_{peaks}} \cdot {}^{7883}/_{i_{max}}$

10 `Peak-Erkennung zur Bestimmung der Kantenpositionen` $\underline{\mathbf{p}} \in \mathbb{N}^{\{10,11\}}$ `anhand der Maxima in` $\underline{\mathbf{s}}^2$ `beginnend bei globalem Maximum und unter Berücksichtigung von` d_{min} `zur Maskierung umliegender Elemente`

11 `Plausibilitätsprüfung für ersten und letzten Peak anhand des Wertes:`
1. `Berechnung des Medians:` $m = \mathrm{med}(\underline{\mathbf{s}}^2[\underline{\mathbf{p}}[1{:}{-1}]])$
2. `Falls` $\underline{\mathbf{p}}[0] < th_{peaks} \cdot m$ `verwerfen von` $\underline{\mathbf{p}}[0]$`:` $\underline{\mathbf{p}} = \underline{\mathbf{p}}[1{:}]$
3. `Falls` $\underline{\mathbf{p}}[-1] < th_{peaks} \cdot m$ `verwerfen von` $\underline{\mathbf{p}}[-1]$`:` $\underline{\mathbf{p}} = \underline{\mathbf{p}}[{:}{-1}]$

12 `Plausibilitätsprüfung für ersten und letzten Peak anhand des Abstandes zum Bildrand:`
1. `Falls` $\underline{\mathbf{p}}[0] > d_{max}$ `verwerfen des erkannten Rasters:` $\underline{\mathbf{p}}[:] = -1$
2. `Falls` $(7883 - \underline{\mathbf{p}}[-1]) > d_{max}$ `verwerfen des erkannten Rasters:` $\underline{\mathbf{p}}[:] = -1$

Verrechnung der erkannten Raster $\underline{\mathbf{p}}^{(oben)}$ und $\underline{\mathbf{p}}^{(unten)}$ zu $\underline{\mathbf{p}}^{(final)}$:

```
13      Berechnung der Welligkeiten: w^(•) = sum(abs(diff(diff(p^(•)))))
14      Falls all(p^(oben)[:] == −1) und all(p^(unten)[:] == −1): goto Fehler
15      Falls all(p^(oben)[:] == −1): goto Final_unten
16      Falls all(p^(unten)[:] == −1): goto Final_oben
17      Sonst:
18         Falls size(p^(oben)) != size(p^(unten)):
19            Falls w^(oben) < w_max und w^(unten) < w_max: goto Fehler
20            Falls w^(oben) > w^(unten) und w^(oben) > w_max: goto Final_unten
21            Falls w^(unten) > w^(oben) und w^(unten) > w_max: goto Final_oben
22         Sonst:
23            Berechnung der Standardabweichungen σ ∈ ℝ^size(p^(•)):
              σ[j] = std({p^(oben)[j], p^(unten)[j]})   ∀j ∈ [0, ..., size(p^(•)) − 1]
24            Falls any(σ[:] > σ_max):
25               Falls w^(unten) > w^(oben): goto Final_oben
26               Falls w^(oben) > w^(unten): goto Final_unten
27            Sonst:
28               p^(final)[:] = 1/2 · (p^(oben) + p^(unten))
29               goto Ende
Fehler:
30      p^(final)[:] = −1; goto Ende
Final_oben:
31      p^(final)[:] = p^(oben); w^(final) = w^(oben); goto Ende
Final_unten:
32      p^(final)[:] = p^(unten); w^(final) = w^(unten)
Ende:
```

Angabe der Detektionssicherheit c:

```
33      Falls p^(final)[:] == −1: c = 0
34      Falls w^(final) > w_max: c = 1
35      Falls w^(final) < w_max: c = 2
```

Rückgaben

```
36      p^(final)                    // Rasterpositionen der Stitching-Kanten
37      c                            // Detektionssicherheit
```

Abbildung 4.10: Pseudocode: Algorithmus zur Lokalisierung der Stitching-Kanten

Dargestellt ist der an Python angelehnte Pseudocode für den realisierten Algorithmus zur Lokalisierung der Stitching-Kanten. Ausführliche Erläuterungen können der textuellen Beschreibung in diesem Abschnitt entnommen werden.

4.4.1.2 Experimente zur Parameteridentifikation

In diesem Abschnitt werden die durchgeführten experimentellen Untersuchungen zur Parameteridentifikation beschrieben. Sowohl der Algorithmus als auch die verwendeten Parameter wurden auf iterative Weise bestimmt. In jeder Iteration wurden die Ergebnisse von allen 1.637 Fahrbahnbildern betrachtet und bestehende Probleme analysiert. Anschließend wurden entweder die Parameter entsprechend adaptiert, oder es erfolgte eine Anpassung des Algorithmus. Tabelle 4.2 fasst alle Konfigurationen zusammen, welche nach der Festlegung des Grundalgorithmus untersucht wurden.

	Konfiguration			
	I	II	III	IV
Allgemeine Parameter				
Skalierungsfaktor für Peakabstand c_{peaks}	0,7	0,7	0,7	0,8
maximale Welligkeit w_{max} [Pixel]	10	10	10	10
maximale Standardabweichung σ_{max} [Pixel]	2	2	2	2
Plausibilitätsprüfung erster/letzter Peak				
relative Mindesthöhe th_{peaks}	0,7	0,6	0,5	0,5
Funktion zur Bestimmung der mittleren Höhe	Mittelwert	Median	Median	Median
Überprüfung des Abstandes zum Bildrand	Nein	Nein	Nein	Ja

Tabelle 4.2: Parameteridentifikation Stitching-Kanten-Lokalisierung
Dargestellt sind die nach der Festlegung des Grundalgorithmus getesteten Konfigurationen zur finalen Parameteridentifikation. Für eine Erläuterung der einzelnen Parameter sei auf den Pseudocode in Abbildung 4.10 verwiesen

Neben den bereits vorgestellten Parametern wurden zwei verschiedene Funktionen zur Bestimmung der mittleren Höhe für die Validierung des ersten und des letzten Peaks betrachtet (Pseudocode Zeile 11). Weiterhin wurde die für diese Peaks durchgeführte Plausibilitätsprüfung anhand des Abstandes zum Bildrand erst in der letzten Iteration ergänzt (Pseudocode Zeile 12) und ist daher nur Teil von Konfiguration IV.

Zur Bewertung der Parameterkonfigurationen wurden die Ergebnisse der Kantenlokalisierung in acht Gruppen untergliedert. Diese umfassen neben einer Klasse für korrekte Detektionen sieben beobachtete Probleme, welche eine fehlerhafte Kantenbestimmung

zur Folge haben. In Tabelle 4.3 werden die einzelnen Gruppen benannt und die zugehörigen Quantitäten für jede Konfiguration zugeordnet.

	Konfiguration			
	I	II	III	IV
Korrekt [%]	87,35	94,07	96,03	98,35
Fehlerhaft [%]	12,65	5,93	3,97	1,65
Korrekt	1.430	1.540	1.572	1.610
Fehlerhaft	207	97	65	27
∟ Kante am linken Bildrand fehlt	100	34	8	1
∟ Kante am rechten Bildrand fehlt	71	24	6	2
∟ Kante am linken und rechten Bildrand fehlt	10	3	1	-
∟ Kante am linken Bildrand zuviel	2	2	3	2
∟ Kante am rechten Bildrand zuviel	1	5	15	1
∟ Fehler im Inneren	23	29	32	20
∟ alle Raster wurden verworfen	-	-	-	1

Tabelle 4.3: Ergebnisse Stitching-Kanten-Lokalisierung

Die Tabelle zeigt die Ergebnisse der Untersuchungen zur Parameteridentifikation.

Die für Konfiguration I gewählten Parameter führten zu einer Vielzahl fehlerhafter Ergebnisse, bei denen die letzte Kante unmittelbar vor den Bildrändern nicht detektiert wurde. Daher wurden die quadratischen Summen der zugehörigen Peaks und die jeweils bestimmten mittleren Höhen analysiert. Dabei hat sich herausgestellt, dass der genutzte Mittelwert zu stark von einzelnen deutlich ausgeprägten Kanten beeinflusst wird (siehe letzter Peak in den Abbildungen 4.8a-4.8d, Seite 50) und dass der davon verwendete prozentuale Anteil die Detektion der beiden Kanten in vielen Fällen ausschließt. Für das anschließende Experiment wurde daher der Median und eine leicht reduzierte Mindesthöhe genutzt. Die Ergebnisse in Tabelle 4.3 zeigen, dass die für Konfiguration II gewählten Parameter zu einer deutlichen Verbesserung, aber noch nicht zu zufriedenstellenden Ergebnissen geführt haben. Daher wurden erneut die berechneten Größen für die verbleibenden fehlerhaft bestimmten Bilder analysiert. In der Konsequenz wurde die erforderliche Mindesthöhe für Konfiguration III nochmals durch eine Anpassung des prozentualen Anteils gesenkt. Während damit das Nicht-

detektieren dieser Kanten weitestgehend vermieden werden konnte, wurden nun aber auch vermehrt zusätzliche, nicht existierende Kanten erkannt. Aus diesem Grund wurde für die abschließende Konfiguration der erforderliche Mindestabstand zwischen den Kanten erhöht. Dadurch erreicht die Maskierung in vielen Fällen den Bildrand und es kann die Detektion zusätzlicher Kanten ausgeschlossen werden. Gleichzeitig hat sich gezeigt, dass die meisten verbleibenden nicht detektierten Kanten lediglich in einer der beiden Bildhälften nicht korrekt bestimmt wurden. In der anschließenden Verrechnung wurde das korrekte Raster aufgrund höherer Welligkeit verworfen. Die erhöhte Welligkeit ist dabei aber nicht einer fehlerhaften Lokalisierung, sondern abweichenden Segmentgrößen an den Bildrändern geschuldet. Daher wurde der Algorithmus um eine zusätzliche Überprüfung des Abstandes zum Bildrand erweitert. In der Folge konnten die beschriebenen, fälschlicherweise für korrekt gehaltenen Raster vor der Verrechnung verworfen und somit das Ergebnis korrigiert werden.

Der finale Algorithmus liefert für 1.610 der 1.637 Fahrbahnbilder das korrekte Ergebnis. Lediglich für 27 Bilder ist die Bestimmung entweder nicht möglich oder fehlerhaft. Der Großteil der Fehler ist hierbei durch falsch lokalisierte Kanten im Inneren des Bildes bedingt, welche auch nicht durch die getrennte Betrachtung der beiden Bildhälften behoben werden können. Infolge von Artefakten, wie Abflüssen, Straßenmarkierungen oder Pflastersteinen auf Geh- und Radwegen, enthält das Bild in diesen Fällen noch stärker ausgeprägte Kanten, welche durch den Algorithmus bevorzugt detektiert werden. Da fehlerhafte Kanten im Inneren aber in der Regel zu einem ungleichmäßigen Raster führen, können die meisten dieser Fälle durch die angegebene Detektionssicherheit identifiziert werden. Tabelle 4.4 zeigt die bestimmten Sicherheiten.

	Sicherheit		
	Stufe 0	Stufe 1	Stufe 2
Absolut	1	24	1.612
⊢ davon korrekt lokalisiert	-	2	1.608
⊢ davon fehlerhaft lokalisiert	1	22	4

Tabelle 4.4: Sicherheiten Stitching-Kanten-Lokalisierung

Dargestellt ist die Verteilung der bestimmten Detektionssicherheiten.

Insgesamt 25 der 1.637 Fahrbahnbilder werden einer von Stufe 2 abweichenden Detektionssicherheit zugeordnet. Unter diesen befinden sich 23 der 27 Bilder mit fehlerhaft lokalisierten Kanten. Durch den Einsatz eines Operators und der Präsentation aller Raster mit einer Stufe kleiner als 2 könnte folglich bei minimalem Aufwand eine nahezu fehlerfreie Detektion garantiert werden.

Im Anschluss an die experimentellen Untersuchungen wurden die verbleibenden Fehler händisch korrigiert, sodass im weiteren Verlauf von fehlerfrei lokalisierten Stitching-Kanten ausgegangen werden kann.

4.4.2 Bestimmung des Labelrasters

Da mit Hilfe des ZEB-Verfahrens nur flächenmäßige Aussagen zum Zustand der Fahrbahn getroffen werden sollen, sieht das verwendete Format nicht die Speicherung des zur Beurteilung genutzten Labelrasters vor. Folglich muss für eine erneute Verwendung der Labelinformationen zunächst die Lage des Rasters bestimmt werden. Nur bei korrekter Ausrichtung über der Fahrbahn kann sichergestellt werden, dass die markierten Bereiche auch wirklich schadhafte Stellen beschreiben. In diesem Abschnitt wird der für die Lagebestimmung entwickelte Algorithmus präsentiert. Anschließend werden die durchgeführten experimentellen Untersuchungen zur Parameteridentifikation vorgestellt.

4.4.2.1 Algorithmus

Die Bestimmung der Lage des Labelrasters setzt die Lokalisierung der Fahrbahn im Fahrbahnbild voraus. Hierfür muss das Bild auf fahrbahnbegrenzende Strukturen, wie Fahrbahnmarkierungen, Geh- und Radwege oder andersartige Randbereiche, untersucht werden. Da die Gestalt dieser Strukturen sehr vielseitig sein kann, sieht der realisierte Algorithmus hierfür die Anwendung eines trainierten Klassifikators vor. Anschließend können die fahrbahnbegrenzenden Strukturen im Bild maskiert und die Lage der Fahrbahn abgeleitet werden. Nachfolgend werden alle drei Teile des Algorithmus in Textform beschrieben. Genaue Implementierungsdetails können der anschließenden Formulierung als Pseudocode in Abbildung 4.15 (Seite 69) entnommen werden.

Detektion fahrbahnbegrenzender Strukturen

Für die Detektion von fahrbahnbegrenzenden Strukturen wird ein eigens dafür trainierter Klassifikator verwendet, dessen Aufbau sich an dem in [Eisenbach et al., 2017][5] präsentierten Convolutional Neural Network zur Erkennung von Schadstellen orientiert. Das verwendete Netzwerk besteht aus einer Vielzahl von Faltungs- und Pooling-Schichten, gefolgt von drei abschließenden vollverschalteten Schichten[6]. Für den Einsatz in dieser Masterarbeit wurde die letzte vollverschaltete Schicht so modifiziert, dass drei Klassen unterschieden werden können. So sollen Patches mit Straßenmarkierungen (Klasse 0) von solchen mit zu ignorierenden Randbereichen (Klasse 1) und von regulären Straßenausschnitten (Klasse 2) abgegrenzt werden. Da bei der Erstellung des GAPs-Datensatzes neben schadhaften Stellen ebenfalls bei der Extraktion auszuschließende Bereiche in Form von Bildregionen mit Straßenmarkierungen und zu ignorierenden Randbereichen markiert wurden, können diese Bildteile für die beiden zuerst genannten Klassen verwendet werden. Der komplette GAPs-Datensatz[7] kann zur Präsentation von regulären Straßenausschnitten genutzt werden. Der letztlich verwendete Datensatz umfasste neben 4,9 Millionen Beispielen aus dem GAPs-Datensatz weitere 1,46 Millionen Patches mit Straßenmarkierungen und 2,66 Millionen Bildausschnitte, welche zu ignorierende Randbereiche zeigen. Der Klassifikator wurde über einen Zeitraum von einem halben Monat trainiert und anschließend anhand der balancierten Accuracy auf Validierungsdaten die beste Epoche ausgewählt.

Für die Anwendung des Klassifikators auf komplette Fahrbahnbilder wurde die bereits für [Eisenbach et al., 2016][8] entwickelte Umwandlung in ein Netzwerk, welches ausschließlich aus Faltungs- und Pooling-Schichten besteht, genutzt. In der Folge kann das Netzwerk auf beliebige Bildgrößen angewendet werden und die deutlich zeitaufwendigere patchweise Verarbeitung des Inputbildes entfällt[9].

[5] Die Publikation enthält eigene Beiträge zur Bestimmung der verwendeten Netzwerktopologien.

[6] Die gleiche Netzwerkstruktur wird ebenfalls für den zu realisierenden inkrementellen Lernprozess verwendet. Weitere Details zum Aufbau des Klassifikators können daher den Ausführungen in Abschnitt 5.2 (Seite 78) entnommen werden.

[7] Da der Klassifikator parallel zur Überarbeitung des GAPs-Datensatzes trainiert wurde, konnte nicht die überarbeitete Version genutzt werden.

[8] Die Publikation enthält eigene Beiträge zur Anwendung von Klassifikatoren auf ganze Bilder.

[9] Die genaue Vorgehensweise wird in Abschnitt 5.4.1.1 (Seite 85) durch ein Beispiel veranschaulicht.

Nach erfolgter Verarbeitung durch das Netzwerk wird für jede Klasse eine Maske zur Beschreibung der Bereiche im Eingabebild erstellt. Abbildung 4.11 veranschaulicht den Erstellungsprozess der Masken anhand eines Beispielbildes.

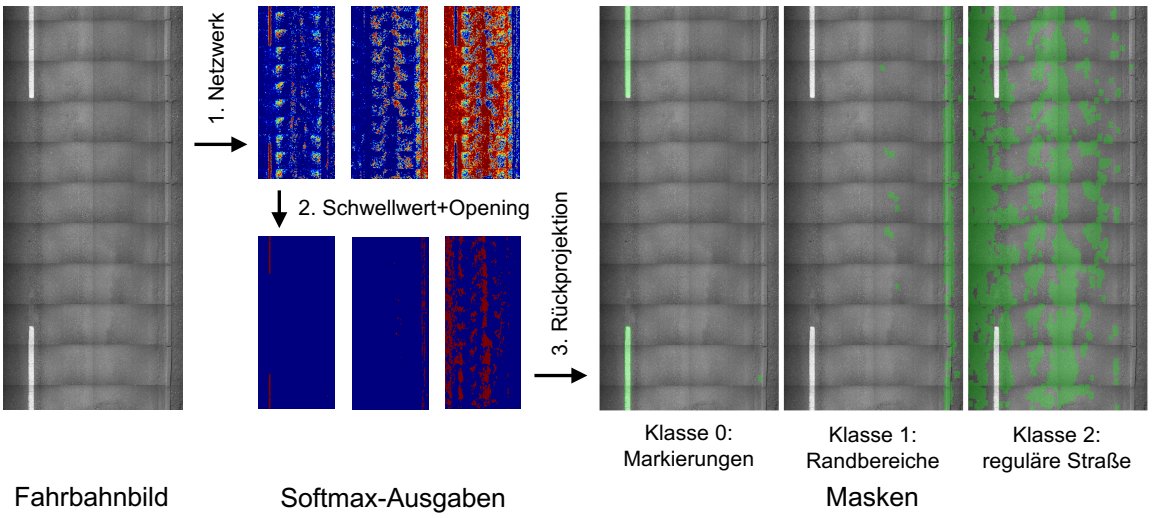


Abbildung 4.11: Detektion fahrbahnbegrenzender Strukturen
Dargestellt ist der Prozess zur Detektion fahrbahnbegrenzender Strukturen. Für ausführliche Erläuterungen sei auf den nachfolgenden Fließtext verwiesen.

Zunächst wird ein Schwellwert auf die Softmax-Ausgabe der jeweiligen Klasse angewendet. Anschließend werden vereinzelte Fehlklassifikationen durch die Anwendung eines Openings mit einem rechteckigen morphologischen Element unterdrückt. Da die maskierten Ausgaben noch immer sehr hochauflösend sind, wird dieser Teil der Verarbeitung durch zwei aufeinanderfolgende Faltungsoperationen realisiert und kann somit sehr effizient mit einer Grafikkarte berechnet werden. Abschließend werden die auf diese Weise bestimmten binären Masken auf die Netzwerkeingabe zurück projiziert. Da das Netzwerk ausschließlich aus Faltungs-und Pooling-Schichten besteht und bei der Festlegung der Netzwerktopologie auf eventuell auftretende Randeffekte[10] geachtet wurde, kann der komplette Vorwärtsprozess durch das Netzwerk ebenfalls als die Anwendung

[10] Randeffekte entstehen, wenn der Filter beziehungsweise der Pooling-Bereich bei der Verarbeitung in einer entsprechenden Schicht nicht mehr komplett auf der Eingabe aufgesetzt werden kann. In der Regel werden diese Randbereiche bei der Verarbeitung einfach ignoriert. Nach der Konvertierung des Netzwerks für eine Anwendung auf beliebige Bildgrößen werden diese Bereiche jedoch berücksichtigt und können zu Problemen führen.

einer einzelnen Faltungs-Schicht interpretiert werden. Die Filtergröße entspricht dabei der für das Training gewählten Patchgröße und die Verschiebung der Filter ergibt sich als Produkt aus den einzelnen, in den Faltungs- und Poolingschichten definierten Verschiebungen. Die Rückprojektion kann letztlich anhand dieser Ersatzschicht durch die Bestimmung der Gradienten der Ausgaben hinsichtlich der Eingaben berechnet werden. Die auf diese Weise zurückgerechneten Ausgaben berücksichtigen die teilweise Überlappung bei der Netzwerkanwendung und führen damit zu sehr genauen Masken. Die für das Beispielbild in Abbildung 4.11 berechneten Masken zeigen, dass die Netzwerkausgabe nicht fehlerfrei ist. Der anschließende Algorithmus zur Bestimmung der Fahrbahnlage muss daher tolerant gegen vereinzelte Fehlklassifikationen sein.

Verknüpfung zu einer Gesamtmaske für unzulässige Bereiche

Nachdem die Masken für alle Klassen extrahiert wurden, werden diese nun zu einer Gesamtmaske zur Beschreibung unzulässiger Bereiche verrechnet. Die Ergebnisse in Abbildung 4.11 zeigen, dass reguläre Straßenausschnitte nicht zuverlässig genug detektiert werden können und deshalb für die Berechnung der Gesamtmaske nicht auf die Inverse der Maske für Klasse 2 zurückgegriffen werden kann. Es erfolgt daher eine Verrechnung der verbleibenden beiden Masken. Abbildung 4.12 verdeutlicht die beschriebene Vorgehensweise zur Verknüpfung der beiden Masken.

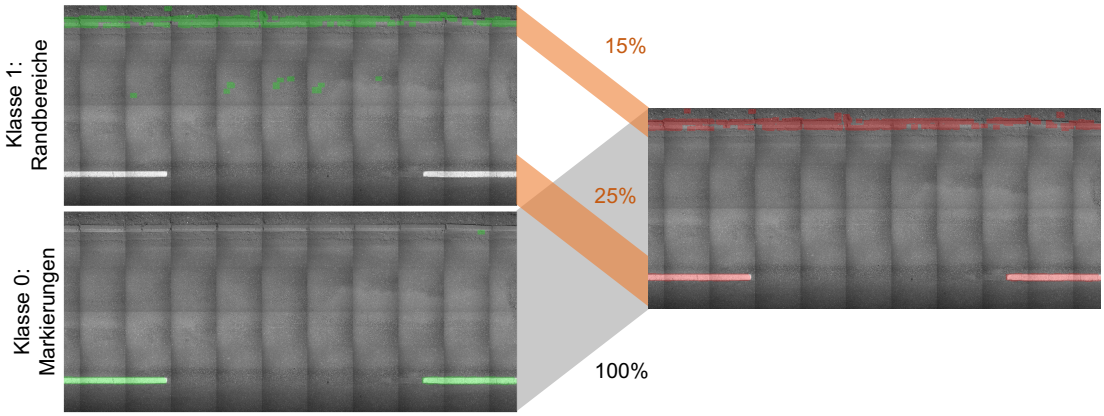


Abbildung 4.12: Verknüpfung zu einer Gesamtmaske für unzulässige Bereiche
Die Abbildung verdeutlicht die Verrechnung der Masken für die Klassen 0 und 1 zu einer Gesamtmaske zur Beschreibung unzulässiger Bereiche.

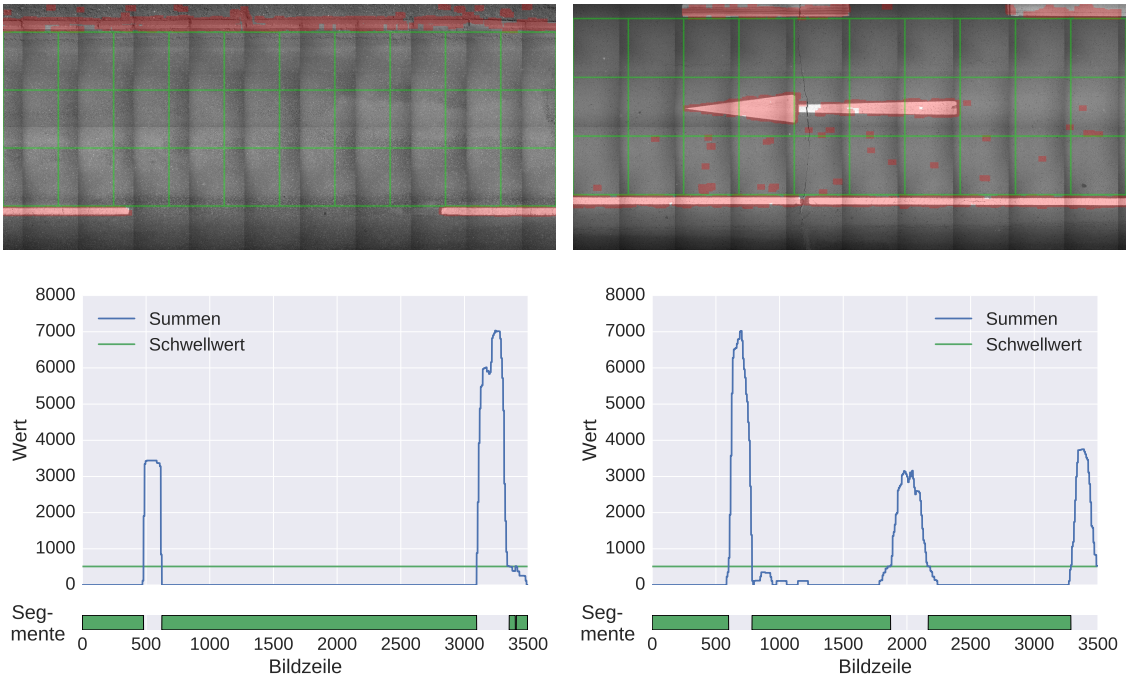
Da relevante Straßenmarkierungen im gesamten Bild auftreten können, werden für diese Klasse alle Bereiche in die Ergebnismaske übernommen. Aufgrund der Kalibrierung der Kameras auf den rechten Fahrbahnrand vor Aufnahmebeginn kann für Randbereiche hingegen davon ausgegangen werden, dass sich diese ausschließlich am Rand der Bilder befinden. Aus diesem Grund werden von der entsprechenden Maske nur 25% vom linken und 15% vom rechten Bildrand in das Ergebnis übernommen. Auf diese Weise kann der Einfluss von Fehldetektionen innerhalb der Fahrbahn, wie sie in Abbildung 4.11 für Klasse 1 zu sehen sind, verringert werden.

Bestimmung der Lage der Fahrbahn

Die berechnete Gesamtmaske bildet die Grundlage für die nun erfolgende Lagebestimmung der Fahrbahn. Hierfür werden zuerst alle Maskenelemente entlang der Fahrtrichtung aufsummiert. Der auf diese Weise erstellte Querschnitt wird anschließend mit Hilfe eines Schwellwertes in Segmente unterteilt. Abbildung 4.13a veranschaulicht diesen Vorgang für das bisher zur Erläuterung verwendete Fahrbahnbild. Jedes der grün dargestellten Segmente markiert hierbei einen möglichen Bereich für die Fahrbahn. Die Schwierigkeit besteht darin, in allen Bildern stets das fahrbahnbildende Segment beziehungsweise die fahrbahnbildenden Teilsegmente zu identifizieren. Während beispielsweise die Entscheidung für Abbildung 4.13a visuell sehr leicht getroffen werden kann, verdeutlicht das Beispiel in Abbildung 4.13b, dass Straßenmarkierungen oder Störungen infolge von Fehlklassifikationen die Wahl erschweren können. Der Algorithmus orientiert sich daher an den Richtlinien zur Anlage von Landstraßen (RAL)[11] und definiert eine Mindestfahrbahnbreite von 2,5 Metern. Diese metrische Angabe kann mit Hilfe der bekannten Relation zwischen der Bildgröße und der Länge des dargestellten Fahrbahnausschnittes in eine Pixelangabe umgerechnet und für die Validierung der Segmente genutzt werden. Die Segmentanalyse beginnt stets beim größten zentralen Segment. Sollte dieses die definierte Mindestgröße überschreiten, wird davon ausgegangen, dass die Fahrbahn gefunden wurde und der Prozess endet. Andernfalls wird zusätzlich das zweitgrößte Segment bestimmt und dessen relative Lage zum ersten Seg-

[11] RAL: `http://www.vsvi-mv.de/fileadmin/Medienpool/Seminarunterlagen/Seminare_2012/Vortrag_1_-_neue_RAL_Frau_Vetters.pdf` (Zuletzt aufgerufen: 14.05.2017)

ment analysiert. Liegen beide Segmente in der unmittelbaren Umgebung und werden lediglich durch eine Struktur nicht breiter als zwei Fünftel der Mindestfahrbahnbreite getrennt, erfolgt eine Verknüpfung zu einem Ersatzsegment. Auf diese Weise können mittige Straßenmarkierungen, wie in Abbildung 4.13b dargestellt, korrekt behandelt werden. Übersteigt das bestimmte Ersatzsegment die geforderte Mindestgröße, konnte die Fahrbahn lokalisiert werden. Ansonsten endet der Prozess ohne Ergebnis.



(a) Beispielbild 1 **(b)** Beispielbild 2

Abbildung 4.13: Bestimmung der Lage der Fahrbahn

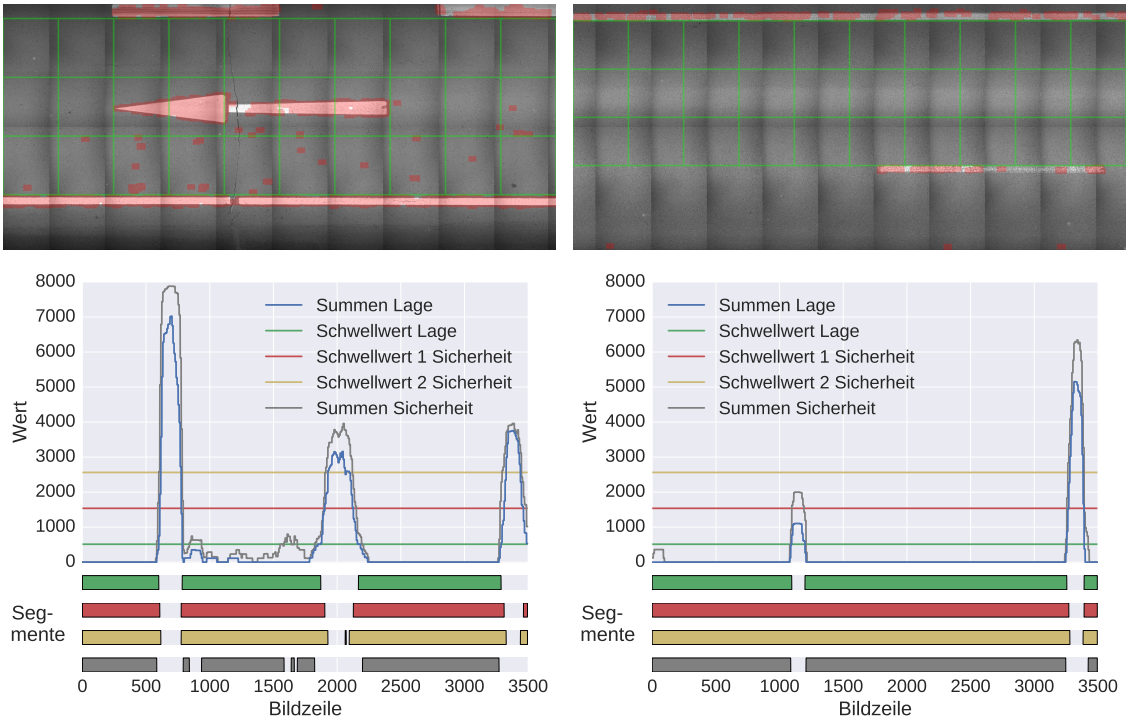
Die Abbildung zeigt zwei Fahrbahnbilder mit rot überlagerten unzulässigen Bereichen und grün eingezeichnetem Ergebnis der Lagebestimmung. Weiterhin werden zwei unterstützende Visualisierungen zur Erläuterung der Vorgehensweise dargestellt. Für weiterführende Details sei auf den vorstehenden Fließtext verwiesen.

Angabe der Sicherheit bei der Detektion

Der realisierte Algorithmus unterstützt neben der eigentlichen Lagebestimmung der Fahrbahn ebenfalls das Ableiten einer Detektionssicherheit, auf deren Basis Operatoren eingebunden oder bestimmte Fahrbahnbilder für eine automatisierte Weiterverar-

beitung ausgeschlossen werden können. Bei der Angabe der Detektionssicherheit wird sowohl die Größe der fahrbahnbegrenzenden Bildstrukturen als auch die Entscheidungskraft des eingesetzten Klassifikators berücksichtigt.
Zur Beurteilung der entscheidungsgebenden Bildstrukturen wird die Segmenteinteilung und die Lagebestimmung der Fahrbahn zunächst mit dreifachem und fünffachem Schwellwert wiederholt. Anschließend werden die bestimmten Fahrbahnpositionen auf Gleichheit geprüft. Hierfür wird die Standardabweichung aus den einzelnen Start- und Endpunkten berechnet und mit einem vorher festgelegten, maximal zu tolerierenden Wert verglichen. Dieser Maximalwert orientiert sich an den in den Richtlinien zur Anlage von Landstraßen festgelegten Bestimmungen zur Fahrbahnkennzeichnung. In diesen wird eine minimale Linienstärke von 10cm definiert. Da bei der Validierung auf keinen Fall vorhandene Linien übersprungen werden dürfen, wird für die maximale Standardabweichung ein Drittel dieser Minimalbreite verwendet. Im Fall von identisch bestimmten Fahrbahnpositionen kann geschlussfolgert werden, dass das Ausgangsbild sehr großflächige fahrbahnbegrenzende Strukturen enthält und dass diese vom Klassifikator auch entsprechend erkannt werden. Folglich kann davon ausgegangen werden, dass die Lagebestimmung auf einer soliden Basis beruht. Entsprechend kann dem Ergebnis eine höhere Sicherheit zugeordnet werden. Abbildung 4.14 verdeutlicht den verfolgten Ansatz an zwei Beispielen. Während sich die fusionierten Fahrbahnsegmente in Abbildung 4.14a auch bei erhöhten Schwellwerten (rot beziehungsweise gelb dargestellt) kaum verändern, unterscheiden sich die fahrbahndefinierenden Segmente in Abbildung 4.14b bereits bei doppeltem Schwellwert. Der verfolgte Ansatz ordnet daher dem ersten Beispiel eine deutlich höhere Sicherheit zu. Ausgehend von der initialen Stufe 1 für eine erfolgreiche Lagebestimmung der Fahrbahn, kann die Sicherheit durch diese Überprüfung für jeden zusätzlichen Schwellwert um eine Stufe erhöht werden.
Zusätzlich zu den Bildstrukturen wird ebenfalls die Entscheidungskraft der Klassifikatorausgabe beurteilt. Hierfür wird die Bestimmung der Fahrbahnposition unter Verwendung eines gesenkten Schwellwertes für die Softmax-Ausgabe ein weiteres Mal wiederholt. Ziel ist es, zu überprüfen, ob sich die detektierten fahrbahnbegrenzenden Strukturen auch bei gesenktem Schwellwert in ähnlicher Weise von der Umgebung abgrenzen lassen. In diesem Fall kann davon ausgegangen werden, dass die bestimmte La-

ge kaum von Fehlklassifikationen beeinflusst wurde. Für die Beispiele in Abbildung 4.14 ist diese zusätzliche Lagebestimmung grau dargestellt. Abbildung 4.14b zeigt, dass auf diese Weise fälschlicherweise ausmaskierte korrekte Klassifikationen mit in die Positionsfindung einbezogen werden können. Abbildung 4.14a hingegen verdeutlicht, dass der gesenkte Schwellwert ebenfalls den Einfluss von Fehldetektionen verstärkt. So entstehen für dieses Bild deutlich abweichende Segmente mit erheblichen Folgen für die Lagebestimmung, welche für diese Konstellation nicht möglich wäre.



(a) Beispielbild 1 (Sicherheit: 3,0) **(b)** Beispielbild 2 (Sicherheit: 1,5)

Abbildung 4.14: Angabe der Sicherheit bei der Detektion

Die Abbildung zeigt zwei Fahrbahnbilder mit rot überlagerten unzulässigen Bereichen und grün eingezeichnetem Ergebnis der Lagebestimmung. Weiterhin werden zwei unterstützende Visualisierungen zur Erläuterung der Vorgehensweise zur Ableitung der Sicherheit dargestellt. Für weiterführende Details sei auf den Fließtext verwiesen.

Aus diesem Grund wird diese Überprüfung ausschließlich zur Ableitung von Sicherheitsinformationen genutzt. Der Vergleich der beiden bestimmten Fahrbahnpositionen erfolgt auf die bereits beschriebene Weise. Sollten sich hierbei identische Fahrbahn-

positionen ergeben, wird die zugeordnete Sicherheit um eine halbe Stufe erhöht. Auf diese Weise kann trotz der Verwendung von nur einem Zahlenwert für die Sicherheit eine nachträgliche Unterscheidung ermöglicht werden. Zusammenfassend ergeben sich für die Sicherheit die folgenden Stufen:

Stufe 0

Die Lage der Fahrbahn konnte nicht bestimmt werden. In diesem Fall könnten einem Operator verschiedene Möglichkeiten auf Basis der unterschiedlichen Schwellwerte präsentiert werden.

Stufe 1

Die Lage der Fahrbahn konnte bestimmt werden, jedoch existieren keine großflächigen fahrbahnbegrenzenden Strukturen oder diese konnten nicht detektiert werden. (Stufe 1,5: fahrbahnbegrenzende Strukturen konnten aber deutlich abgegrenzt werden)

Stufe 2

Die Lage der Fahrbahn konnte bestimmt werden. Zudem gibt es Anzeichen, dass die Detektion auf großflächigen fahrbahnbegrenzenden Strukturen basiert. (Stufe 2,5: fahrbahnbegrenzende Strukturen konnten zudem deutlich abgegrenzt werden)

Stufe 3

Die Lage der Fahrbahn konnte bestimmt werden und basiert auf großflächigen fahrbahnbegrenzenden Strukturen. (Stufe 3,5: fahrbahnbegrenzende Strukturen konnten zudem deutlich abgegrenzt werden)

Für den kompletten Lokalisierungsprozess einschließlich der Ableitung der Sicherheit werden pro Fahrbahnbild unter Verwendung einer mittelpreisigen Grafikkarte, wie beispielsweise einer NVIDIA GTX 1080, durchschnittlich 2,923 Sekunden benötigt.

Formulierung als Pseudocode

Nachfolgend wird der Algorithmus in Form von Pseudocode beschrieben. Da die verwendete Notation an Python-Syntax angelehnt ist, beginnen Indizierungen stets bei 0.

Eingaben

$\underline{\mathbf{X}} \in \{0, 1, \ldots, 255\}^{3498 \times 7883}$ // *Fahrbahnbild*

Lagebestimmung:

$\underline{\mathbf{s}}_{Lage} = [0.98, 0.98, 0.98]$ // *Softmax-Schwellwerte*

$h_{Opening} = 3, w_{Opening} = 7$ // *Größe morphologisches Element Opening in Pixeln*

$th_{Lage} = \text{meter2pixel}(0{,}65)$ // *Schwellwert für Segmenteinteilung*

$d_{Fahrbahn} = \text{meter2pixel}(2{,}50)$ // *Mindestfahrbahnbreite*

$d_{Segment} = {}^{2}/_{5} \cdot \text{meter2pixel}(2{,}50)$ // *maximaler Abstand zwischen zwei Segmenten*

Sicherheitsbestimmung:

$\underline{\mathbf{s}}_{Sicherheit} = [0.96, 0.96, 0.96]$ // *Softmax-Schwellwerte*

$th_{Sicherheit1} = 3 \cdot th_{Lage}$ // *1. Schwellwert für Segmenteinteilung*

$th_{Sicherheit2} = 5 \cdot th_{Lage}$ // *2. Schwellwert für Segmenteinteilung*

$\sigma_{Sicherheit} = {}^{1}/_{3} \cdot \text{meter2pixel}(0{,}10)$ // *maximale Standardabweichung für gleiche Positionen*

Algorithmus

Hilfsfunktionen:

function `meter2pixel`(x): **return** $x \cdot {}^{7883}/_{10}$ // *meter2pixel(Angabe in Metern)*

function `segmentbestimmung`$(\underline{\mathbf{s}}, th)$: // *segmentbestimmung(Summen, Schwellwert)*

  $\mathbb{S} = \emptyset$, $start = -1$

  for $i = 0$ **to** $\text{size}(\underline{\mathbf{s}}) - 1$

    if $\underline{\mathbf{s}}[i] > th$ **and** $start == -1$ **then** $start = i$

    if $\underline{\mathbf{s}}[i] <= th$ **and** $start \mathrel{!=} -1$ **then** $start = -1$; $\mathbb{S} = \mathbb{S} \cup \{[start, i]\}$

  if $start \mathrel{!=} -1$ **then** $\mathbb{S} = \mathbb{S} \cup \{[start, \text{size}(\underline{\mathbf{s}})]\}$

  return $\mathbb{S}$

function `lagebestimmung`$(\mathbb{S})$: // *lagebestimmung(Menge der Segmente)*

  $id = \text{argsort}(\mathbb{S}, key = segment[1] - segment[0])$

  if $(\mathbb{S}[id[0]][1] - \mathbb{S}[id[0]][0]) > d_{Fahrbahn}$ **then**

    return $\mathbb{S}[id[0]]$

  $\underline{\mathbf{s}}_{links} = \mathbb{S}[\min(id[0], id[1])]$

  $\underline{\mathbf{s}}_{rechts} = \mathbb{S}[\max(id[0], id[1])]$

  if $(\underline{\mathbf{s}}_{rechts}[0] - \underline{\mathbf{s}}_{links}[1]) > d_{Segment}$ **then**

    if $(\underline{\mathbf{s}}_{rechts}[1] - \underline{\mathbf{s}}_{links}[0]) > d_{Fahrbahn}$ **then**

      return $[\underline{\mathbf{s}}_{links}[0], \underline{\mathbf{s}}_{rechts}[1]]$

  return $[-1, -1]$

function `gleicheposition`$(\mathbb{P})$: // *gleicheposition(Menge der Positionen)*

  $\sigma_{links} = \text{std}(\{\underline{\mathbf{p}}[0]\}_{\underline{\mathbf{p}} \in \mathbb{P}})$

  $\sigma_{rechts} = \text{std}(\{\underline{\mathbf{p}}[1]\}_{\underline{\mathbf{p}} \in \mathbb{P}})$

  return $\sigma_{links} < \sigma_{Sicherheit}$ **and** $\sigma_{rechts} < \sigma_{Sicherheit}$

Detektion fahrbahnbegrenzender Strukturen:

33 `Anwendung des Klassifikators zur Generierung der Ausgabe` $\underline{\mathbf{Y}} \in \mathbb{R}^{3\times430\times978}$:
$\underline{\mathbf{Y}} = \text{network}(\underline{\mathbf{X}})$

34 `Maskierung der Ausgaben mit den Softmax-Schwellwerten` $\underline{\mathbf{s}} = \underline{\mathbf{s}}_{Lage}$:
$\underline{\mathbf{Y}}[i] = \underline{\mathbf{Y}}[i] >= \underline{\mathbf{s}}[i] \quad \forall i \in [0,1,2]$

35 `Anwendung des Openings mit morphologischem Element` $\underline{\mathbf{E}} \in \{1\}^{h_{Opening}\times w_{Opening}}$:
$\underline{\mathbf{Y}}[i] = \text{opening}(\underline{\mathbf{Y}}[i], \underline{\mathbf{E}}) \quad \forall i \in [0,1,2]$

36 `Rückprojektion der Ausgaben auf die Eingaben zur Generierung der Masken` $\underline{\mathbf{M}} \in \{0,1\}^{3\times3496\times7880}$ `durch transponierte Faltung mit den Filtern` $\underline{\mathbf{F}} \in \{1\}^{1\times1\times64\times64}$ `und den Verschiebungen` $s_y = s_x = 8$:
$\underline{\mathbf{M}}[i] = \text{transposedconv}(\underline{\mathbf{Y}}[i], \underline{\mathbf{F}}, s_y, s_x) > 0{,}0 \quad \forall i \in [0,1,2]$

Verknüpfung zu einer Gesamtmaske für unzulässige Bereiche:

37 `Bestimmung der Gesamtmaske` $\underline{\mathbf{U}} \in \{0,1\}^{3496\times7880}$ `durch Übernahme der Maske für Klasse 0: Markierungen:`
$\underline{\mathbf{U}} = \underline{\mathbf{M}}[0]$

38 `Hinzufügen der Anteile der Maske für Klasse 1: Randbereiche:`
$\underline{\mathbf{U}}[:n,:] = \underline{\mathbf{U}}[:n,:] \vee \underline{\mathbf{M}}[1,:n,:]$ `mit` $n = \lceil 0{,}15 \cdot 3496 \rceil$
$\underline{\mathbf{U}}[:-n,:] = \underline{\mathbf{U}}[:-n,:] \vee \underline{\mathbf{M}}[1,:-n,:]$ `mit` $n = \lceil 0{,}25 \cdot 3496 \rceil$

Bestimmung der Lage der Fahrbahn:

39 `Summation aller Maskenelemente entlang der Fahrtrichtung zur Bestimmung der Summen` $\underline{\mathbf{q}} \in \mathbb{N}^{3496}$:
$\underline{\mathbf{q}}[i] = \text{sum}(\underline{\mathbf{U}}[i,:]) \quad \forall j \in [0,\ldots,3496-1]$

40 `Einteilung der Summen in Segmente` $\mathbb{S}$: $\mathbb{S} = \text{segmentbestimmung}(\underline{\mathbf{q}}, th_{Lage})$

41 `Bestimmung der Fahrbahnposition` $\underline{\mathbf{p}} \in \mathbb{N}^2$: $\underline{\mathbf{p}} = \text{lagebestimmung}(\mathbb{S})$

Angabe der Detektionssicherheit c:

42 `Falls` $\text{all}(\underline{\mathbf{p}}[:] == -1)$: $c = 0$; **goto** *Ende*

43 `Festlegung des Standardwertes:` $c = 1$

44 `Beurteilung der entscheidungsgebenden Bildstrukturen:`

45 `Bestimmung der Fahrbahnpositionen mit erhöhten Schwellwerten:`
$\underline{\mathbf{p}}_{Sicherheit1} = \text{lagebestimmung}(\text{segmentbestimmung}(\underline{\mathbf{q}}, th_{Sicherheit1}))$
$\underline{\mathbf{p}}_{Sicherheit2} = \text{lagebestimmung}(\text{segmentbestimmung}(\underline{\mathbf{q}}, th_{Sicherheit2}))$

46 `Falls` $\text{gleicheposition}(\{\underline{\mathbf{p}}, \underline{\mathbf{p}}_{Sicherheit1}\})$: $c = c + 1$

47 `Falls` $\text{gleicheposition}(\{\underline{\mathbf{p}}, \underline{\mathbf{p}}_{Sicherheit1}, \underline{\mathbf{p}}_{Sicherheit2}\})$: $c = c + 1$

48 `Beurteilung der Entscheidungskraft der Klassifikatorausgabe:`

49 `Wiederhole den Prozess ab der Detektion fahrbahnbegrenzender Strukturen in Zeile 34 unter Nutzung des Schwellwertes` $\underline{\mathbf{s}} = \underline{\mathbf{s}}_{Sicherheit}$ `bis zur Angabe der Detektionssicherheit zur Bestimmung der Fahrbahnposition` $\underline{\mathbf{p}}_{Sicherheit3}$

50 `Falls` $\text{gleicheposition}(\{\underline{\mathbf{p}}, \underline{\mathbf{p}}_{Sicherheit3}\})$: $c = c + 0{,}5$

Ende:

Rückgaben

```
51  p                    // Position der Fahrbahn beziehungsweise des Labelrasters
52  c                    // Detektionssicherheit
```

Abbildung 4.15: Pseudocode: Algorithmus zur Bestimmung des Labelrasters
Dargestellt ist der an Python angelehnte Pseudocode für den realisierten Algorithmus zur Bestimmung des Labelrasters. Ausführliche Erläuterungen können der textuellen Beschreibung in diesem Abschnitt entnommen werden.

4.4.2.2 Experimente zur Parameteridentifikation

In diesem Abschnitt werden die durchgeführten experimentellen Untersuchungen zur Parameteridentifikation zusammengefasst. Da eine Betrachtung aller Experimente den vorgegebenen Rahmen übersteigen würde, wird an dieser Stelle hauptsächlich auf die für die Lagebestimmung relevanten Parameter eingegangen, welche nicht auf einer metrischen Angabe basieren oder deren metrische Interpretation nicht sofort erkennbar ist[12]. Tabelle 4.5 zeigt die in diesem Abschnitt genauer betrachteten Parameterkonfigurationen.

	Konfiguration			
	I	II	III	IV
Detektion fahrbahnbegrenzender Strukturen				
Softmax-Schwellwerte $\underline{\mathbf{s}}_{Lage}[:]$	0,96	0,96	0,98	0,98
Höhe des morphologischen Elements $h_{Opening}$ [Pixel]	3	3	3	3
Breite des morphologischen Elements $w_{Opening}$ [Pixel]	3	3	3	7
Lagebestimmung der Fahrbahn				
Schwellwert für Segmenteinteilung th_{Lage} [m]	0,50	0,65	0,65	0,65

Tabelle 4.5: Parameterkonfigurationen Labelraster-Bestimmung
Dargestellt sind ausgewählte Parameterkonfigurationen, welche im Anschluss genauer betrachtet werden. Für Details zur Bedeutung und Verwendung der einzelnen Parameter sei auf den Pseudocode in Abbildung 4.15 verwiesen.

[12] Es wurden zusätzlich mehrere Experimente zur Bestimmung der Parameter zur Ableitung von aussagekräftigen Detektionssicherheiten durchgeführt.

Die Festlegung der initialen Parameter erfolgte anhand von 20 zufällig ausgewählten Fahrbahnbildern. In jedem Bild wurde zuerst die Länge der kürzesten Fahrbahnmarkierung bestimmt. Hierbei hat sich gezeigt, dass bereits 0,5m lange Markierungen für die Lagebestimmung relevant sein können (siehe Abbildung 4.6a, Seite 47). Folglich wurde der Schwellwert für die Segmenteinteilung auf den entsprechenden Wert gesetzt. Um geeignete Softmax-Schwellwerte ableiten zu können, wurden zudem die zugehörigen Softmax-Ausgaben betrachtet. Damit die fahrbahnbegrenzenden Strukturen in den ausgewählten Bildern durchgehend detektiert werden konnten, musste der Schwellwert für alle Klassen bis auf einen Wert von 0,96 reduziert werden. Um vereinzelten Fehlklassifikationen vorbeugen zu können, wurde weiterhin ein Opening mit einem quadratischen Element verwendet, wobei die Seitenlänge zunächst zu drei Pixeln gewählt wurde.

Anschließend wurde die Parameterkonfiguration I auf alle 1.637 Fahrbahnbilder angewendet. Die Ergebnisse in Tabelle 4.6 zeigen, dass die korrekte Fahrbahnposition mit diesen Parametern nur für etwas mehr als die Hälfte der Bilder bestimmt werden konnte. Die Analyse der fehlerhaften Labelraster hat ergeben, dass es besonders oft in der unmittelbaren Umgebung von deutlich sichtbaren Markierungen zu Fehlklassifikationen kommt. In der Folge wird ein zu kleines fahrbahnbildendes Segment bestimmt, welches letztlich zu einer abweichenden Fahrbahnposition führt. Um diesem Verhalten entgegenzuwirken, wurde der Schwellwert für die Segmenteinteilung in Konfiguration II auf einen Wert von 0,65m erhöht. Die Ergebnisse zeigen jedoch, dass dies nur bedingt zu einer Verbesserung geführt hat. Daher wurde für Konfiguration III zusätzlich der Softmax-Schwellwert für alle Klassen erhöht und auf einen Wert von 0,98 gesetzt. Durch beide Änderungen konnte die Anzahl der fehlerhaft bestimmten Fahrbahnpositionen immerhin um die Hälfte reduziert werden. Da die Lage jedoch noch immer für etwas mehr als ein Fünftel der Fahrbahnbilder fehlerhaft bestimmt wurde, wurden die Ergebnisse für diese Bilder ein weiteres Mal analysiert. Hierbei hat sich gezeigt, dass die Fehlklassifikationen durch die Erhöhung der Softmax-Schwellwerte auf nur noch sehr kleine und größtenteils nicht zusammenhängende Bereiche reduziert werden konnten. Aus diesem Grund wurde für die finale Konfiguration die Größe des morphologischen Elements verändert. Da bei der Festlegung der Abmessungen

darauf geachtet werden muss, dass die minimale Linienstärke von 10cm im Ausgangsbild nicht überschritten wird, wurde das morphologische Element lediglich entlang der Fahrtrichtung auf sieben Pixel erweitert. Markierungen müssen somit im Fahrbahnbild auf einer Länge von mindestens 112 Pixeln beziehungsweise 14,2cm durchgehend erkannt werden, damit sie von Fehlklassifikationen unterschieden werden können. Die finalen Ergebnisse in Tabelle 4.6 zeigen, dass durch diese Modifikation der Einfluss von Fehldetektionen in allen Gruppen noch einmal deutlich reduziert und die Genauigkeit der Lagebestimmung wesentlich verbessert werden konnte.

	Konfiguration			
	I	II	III	IV
Korrekt [%]	56,75	63,10	78,68	94,14
Fehlerhaft [%]	43,25	36,90	21,32	5,86
Korrekt	929	1.033	1.288	1.541
Fehlerhaft	708	604	349	96
∟ leicht abweichend	451	396	219	29
∟ stark abweichend	213	173	96	60
∟ keine Bestimmung möglich	44	35	34	7

Tabelle 4.6: Ergebnisse Labelraster-Bestimmung

Dargestellt sind die Ergebnisse der experimentellen Untersuchungen zur Parameteridentifikation. Um die bestimmten Positionen automatisiert beurteilen zu können, wurde die korrekte Fahrbahnlage in allen 1.637 Bildern zuvor händisch markiert. Durch den Algorithmus fehlerhaft bestimmte Fahrbahnpositionen werden für die Auswertung zusätzlich in drei Gruppen unterteilt. Eine Fahrbahnposition wird dabei als stark abweichend bezeichnet, wenn deren Lage an mindestens einer Seite um mehr als 80 Pixel (entspricht der minimalen Linienbreite von 10cm) abweicht.

Mit Hilfe des letztlich realisierten Verfahrens kann die korrekte Fahrbahnposition für 1.541 der 1.637 Fahrbahnbilder ermittelt werden. Dies entspricht einer Genauigkeit von 94,14%. Die in Tabelle 4.7 dargestellten Detektionssicherheiten zeigen, dass mit vertretbarem Aufwand 18 fehlerhaft bestimmte Fahrbahnpositionen korrigiert werden können. Hierfür müssten lediglich 30 Bilder (Detektionssicherheit kleiner als 1,5)

durch einen Operator validiert werden. Die Genauigkeit könnte in diesem Fall um weitere 1,1 Prozentpunkte verbessert werden. Die in Tabelle 4.7 dargestellten Sicherheiten verdeutlichen jedoch auch, dass für eine weitere Fehlerminimierung deutlich mehr Aufwand notwendig ist. Zudem werden vielen falsch bestimmten Positionen fälschlicherweise hohe Sicherheiten zugeordnet. Abbildung 4.16 stellt zwei der acht Fahrbahnbilder dar, für welche das Labelraster trotz höchster Detektionssicherheit nicht korrekt bestimmt wurde. Die den Bildern überlagerten Masken zeigen, dass die falschen Ergebnisse vor allem durch fehlende Detektionen für bereits verblasste Fahrbahnmarkierungen entstehen. Untersuchungen hierzu haben gezeigt, dass diese Bereiche selbst bei einem Softmax-Schwellwert von 0,6 nicht zuverlässig detektiert werden würden. Dieses Problem könnte daher lediglich durch ein gezieltes Nachtraining des Klassifikators behoben werden.

	Sicherheitsstufe						
	0	1	1,5	2	2,5	3	3,5
Absolut	7	23	230	15	484	61	817
∟ davon korrekt bestimmt	-	12	199	13	476	32	809
∟ davon fehlerhaft bestimmt	7	11	31	2	8	29	8

Tabelle 4.7: Sicherheiten Labelraster-Bestimmung
Dargestellt ist die Verteilung der Detektionssicherheiten über alle Fahrbahnbilder.

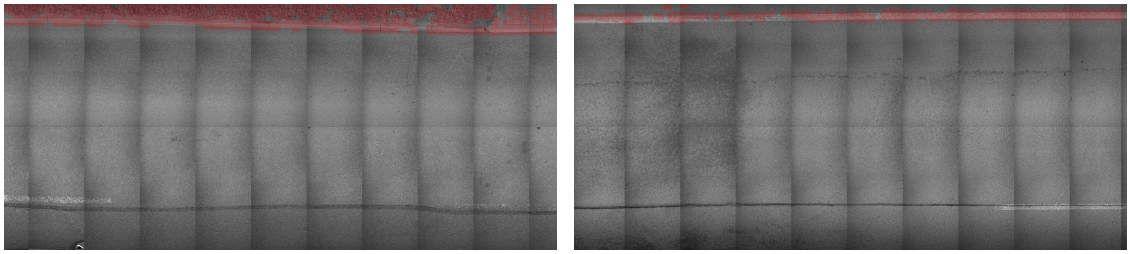


Abbildung 4.16: Beispielbilder zur Erläuterung ausstehender Probleme
Dargestellt sind zwei der acht Fahrbahnbilder, für welche trotz höchster Sicherheitsstufe die Fahrbahnposition falsch bestimmt wurde. Die rot überlagerten Masken zeigen, dass dies vor allem ausbleibenden Detektionen für Fahrbahnmarkierungen im unteren Bildteil geschuldet ist.

Im weiteren Verlauf der Masterarbeit werden ausschließlich die händisch markierten Fahrbahnpositionen verwendet, sodass ein Einfluss von eventuell falsch bestimmten Labelrastern ausgeschlossen werden kann.

4.5 Zusammenfassung

In diesem Kapitel wurden die in dieser Masterarbeit verwendeten Daten vorgestellt. Zum einen stehen sehr exakt gelabelte Einzelbilder zur Verfügung, aus denen bereits im Vorfeld der sogenannte GAPs-Datensatz erstellt wurde. Durch das schon seit vielen Jahren angewendete Verfahren zur Zustandserfassung und -bewertung von Fernstraßen (ZEB) existieren zum anderen sehr große Datenbestände an bereits zusammengesetzten Fahrbahnbildern, welche jedoch nur nach einer geeigneten Vorverarbeitung verwendet werden können. So müssen die in den Bildern enthaltenden Stitching-Kanten erkannt und maskiert werden. Weiterhin muss die genaue Position der Fahrbahn in jedem Bild bestimmt werden, da die verfügbaren Labelinformationen nur dann konkreten Bildbereichen in den Fahrbahnbildern zugeordnet werden können. Für beide Probleme wurden in diesem Kapitel Verfahren vorgestellt, welche diese Vorverarbeitungsschritte realisieren.
In den experimentellen Untersuchungen in Kapitel 6 werden sowohl der GAPs-Datensatz und dessen Ausgangsbilder (im weiteren Verlauf als GAPs-Basisbilder bezeichnet) als auch ein Teil des verfügbaren Datenbestandes an Fahrbahnbilder im Umfang von 1.647 Bildern verwendet.

Kapitel 5

Intelligentes inkrementelles Lernen

Aufbauend auf den Erkenntnissen der State-of-the-Art-Betrachtungen aus Kapitel 3 wird in diesem Teil der Masterarbeit ein Verfahren zur Realisierung von intelligenten inkrementellen Lernprozessen konstruiert. Hierfür werden die anhand der State-of-the-Art-Analyse ausgewählten Ansätze zu einem vollständigen Prozessablauf zusammengesetzt. Obwohl dieser speziell auf den im ASINVOS-Projekt betrachteten Anwendungsfall zugeschnitten ist, können viele der enthaltenen Kernaspekte auch auf andere Anwendungsfelder übertragen werden.
Im nachfolgenden Abschnitt wird zunächst ein Überblick über das realisierte Verfahren gegeben. Anschließend werden die einzelnen Teilprozesse separat betrachtet und genauer beschrieben. Entgegen der theoretisch gehaltenen State-of-the-Art-Ausführungen steht hierbei vor allem eine Betrachtung aus anwendungsorientierter Sicht im Vordergrund. So werden formale Beschreibungen durch anschaulichere textuelle Formulierungen ersetzt und zusätzliche bei der Implementierung zu beachtende Details erläutert.

5.1 Überblick

Die Realisierung des angestrebten intelligenten inkrementellen Lernprozesses umfasst mehrere aufeinanderfolgende Teilschritte. In der in Abbildung 5.1 dargestellten Übersichtsgrafik werden alle benötigten Schritte benannt und in den Prozessablauf eingeordnet.

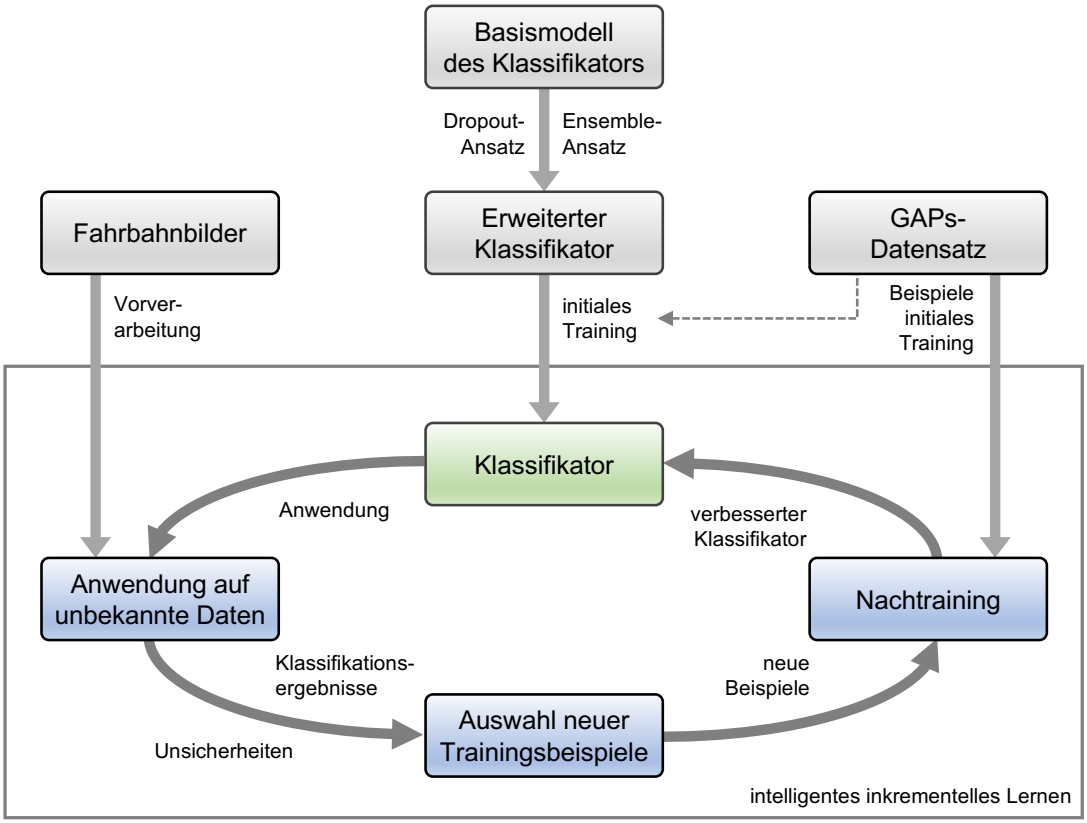


Abbildung 5.1: Übersicht intelligentes inkrementelles Lernen

Dargestellt ist der Prozessablauf zur Realisierung von intelligentem inkrementellem Lernen. Das festgelegte Basismodell des Klassifikators muss zuerst mit einem geeigneten Ansatz um eine Extraktionsmethode für Unsicherheitsinformationen erweitert und initial trainiert werden. Im Anschluss kann in den inkrementellen Lernprozess übergegangen werden. Hierbei wird der aktuelle Klassifikator zunächst auf unbekannte Daten angewendet. Anschließend werden auf Basis der Klassifikationsergebnisse und der extrahierten Unsicherheiten informative, neue Trainingsbeispiele ausgewählt. Abschließend wird der Klassifikator mit dem erweiterten Datensatz erneut trainiert und auf diese Weise in eine verbesserte Form überführt.

Bevor die zur Realisierung von intelligentem inkrementellem Lernen vorgesehene Vorgehensweise angewendet werden kann, muss zunächst ein Basismodell für den zu verwendenden Klassifikator festgelegt werden. Für die in dieser Masterarbeit betrachtete Problemstellung wird das Basismodell anhand des in [Eisenbach et al., 2017][1] präsen-

[1] Die Publikation enthält eigene Beiträge zur Bestimmung der verwendeten Netzwerktopologien.

tierten ASINVOS-Netzwerks definiert. Das verwendete Convolutional Neural Network sowie vorgenommene Modifikationen werden in Abschnitt 5.2 vorgestellt. Weiterhin müssen neben einem gelabelten Datensatz für das initiale Training zusätzliche Daten für den sich anschließenden inkrementellen Lernprozess zur Verfügung stehen. In dieser Masterarbeit werden hierfür der in Abschnitt 4.2 (Seite 40) präsentierte GAPs-Datensatz und die im Abschnitt 4.3 (Seite 43) vorgestellten und entsprechend vorverarbeiteten Fahrbahnbilder verwendet.

Damit für den angestrebten inkrementellen Lernprozess möglichst informative neue Trainingsbeispiele ausgewählt und redundante Daten vermieden werden können, muss neben der eigentlichen Klassifikatorausgabe ebenfalls die Unsicherheit bei der Entscheidungsfindung berücksichtigt werden. Da das betrachtete Basismodell, wie die meisten anderen Deep-Learning-basierten Klassifikatoren, keine Möglichkeit bietet, derartige Unsicherheitsinformationen anzugeben, muss das Modell zunächst auf geeignete Weise erweitert werden. Während der State-of-the-Art-Analyse konnten hierfür zwei mögliche Ansätze identifiziert werden. Zum einen kann die sonst nur für das Training verwendete Dropout-Technik ebenfalls in der Anwendungsphase angewendet werden (Dropout-Ansatz). Zum anderen kann auf Grundlage des definierten Basismodells ein Ensemble konstruiert werden (Ensemble-Ansatz). Die für jeden Ansatz erforderlichen Modifikationen am Basismodel und die Vorgehensweise zur Extraktion von Unsicherheitsinformationen werden in Abschnitt 5.3 beschrieben. In den anschließenden experimentellen Untersuchungen in Kapitel 6 werden beide Ansätze hinsichtlich ihrer Eignung zur Extraktion von Unsicherheitsinformationen untersucht.

Nachdem der Klassifikator erweitert wurde, muss dieser mit Hilfe des initialen Datensatzes trainiert werden. Ziel ist es, die Gewichte des Netzwerks stabil zu initialisieren, sodass bei der Anwendung des Klassifikators aussagekräftige Unsicherheitsinformationen extrahiert werden können. Anschließend kann der Übergang in den angedachten inkrementellen Lernprozess erfolgen. Hierbei wird der Klassifikator in jeder Iteration zunächst auf mehrere Fahrbahnbilder angewendet. Auf Basis der Klassifikatorausgaben und der extrahierten Unsicherheiten werden anschließend neue Trainingsbeispiele ausgewählt und ein weiterer Lernprozess initiiert. In Abschnitt 5.4 (Seite 85) werden diese drei Teilschritte des inkrementellen Lernprozesses erneut aufgegriffen und zunächst die

genaue Realisierung der Anwendungsphase betrachtet. Anschließend werden konkrete Szenarien für die Erweiterung des Trainingsdatensatzes vorgestellt und wichtige Aspekte für das Nachtraining des Klassifikators beschrieben.
Der vorgestellte inkrementelle Lernprozess kann beliebig oft wiederholt werden. Im Idealfall kann die Klassifikationsleistung in jeder Iteration verbessert werden. Sollten fortlaufend ähnliche Daten präsentiert werden, ist von einer Abnahme der Unsicherheit auszugehen.

5.2 Basismodell des Klassifikators

Das in dieser Masterarbeit verwendete Basismodell beruht auf dem in [Eisenbach et al., 2017] vorgestellten ASINVOS-Netzwerk. Die Struktur dieses Convolution Neural Networks ist in Abbildung 5.2 dargestellt.

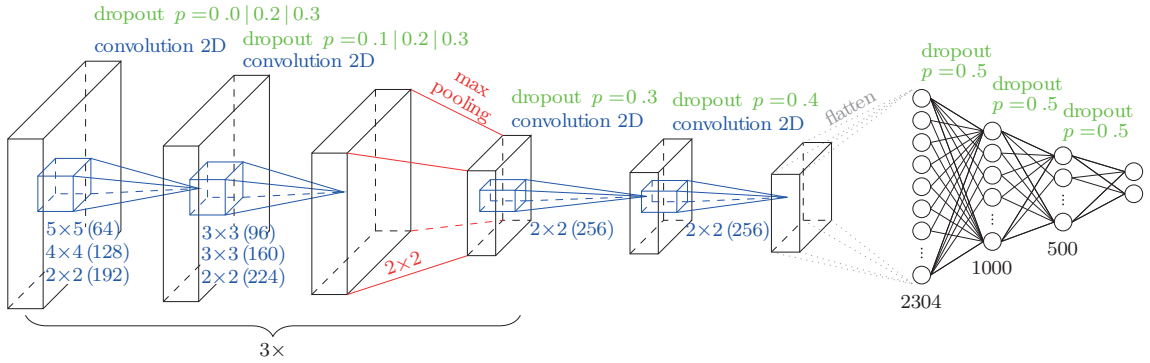


Abbildung 5.2: Aufbau des ASINVOS-Netzwerk

Die Abbildung zeigt die Struktur des in [Eisenbach et al., 2017] vorgestellten ASINVOS-Netzwerks, welches das in dieser Masterarbeit verwendete Klassifikator-Basismodell definiert. Das Netzwerk besteht aus einer Vielzahl von Faltungs- und Pooling-Schichten gefolgt von drei vollverschalteten Schichten.

Die Netzwerkstruktur lässt sich anhand der verwendeten Schichten in zwei Teile untergliedern. Der erste Teil des Netzwerks besteht aus mehreren Faltungs- und Pooling-Schichten, welche in einer alternierenden Form angeordnet sind. Mit Ausnahme des letzten Blocks folgt hierbei auf zwei Faltungs-Schichten stets eine Pooling-Schicht. Durch diesen lokalverschalteten Teil des Netzwerks sollen wichtige Features aus den

präsentierten Straßenausschnitten extrahiert und gleichzeitig die Bildgröße reduziert werden. Für jede Eingabe werden 256 Featuremaps mit einer Größe von 7x7 Pixeln extrahiert und an die anschließenden drei vollverschalteten Schichten im zweiten Teil des Netzwerks übergeben. Dieser besitzt den größten Anteil der Gewichte und realisiert die letztliche Klassifikation[2]. Zur Vermeidung von Overfitting wird während der Trainingsphase in allen Schichten Dropout eingesetzt, wobei die Dropout-Wahrscheinlichkeit zum Ende des Netzwerks schrittweise bis zu einem Wert von 0,5 zunimmt.
Für diese Masterarbeit wird das beschriebene Modell um eine zusätzliche Regularisierung mittels Weight Decay [Krogh und Hertz, 1992] erweitert. Diese wurde bereits in [Eisenbach et al., 2017] untersucht, wobei das Hinzufügen von Weight Decay an dieser Stelle stets zu geringfügig schlechteren Klassifikationsergebnissen geführt hat. Der Grund, warum Weight Decay dennoch eingesetzt wird, ist eine in [Gal, 2016] präsentierte probabilistische Interpretation, nach welcher diese Form der Regularisierung unmittelbaren Einfluss auf die den Gewichten unterstellte Verteilung besitzt und diese auf eine Normalverteilung mit einer Breite indirekt proportional zum verwendeten Weight-Decay-Faktor beschränkt. Diese Beschränkung wirkt sich nicht zuletzt positiv auf den Optimierungsprozess aus. Die Autoren zeigen weiterhin, dass die Verwendung von Dropout die optimale Wahl der Weight-Decay-Faktoren beeinflusst und dass diese mit der Dropout-Wahrscheinlichkeit skaliert werden sollten. Aus diesem Grund wurde der typischerweise verwendete Wert von 0,0005 entsprechend der kleinsten Dropout-Wahrscheinlichkeit skaliert und ein Faktor von 0,00005 für alle Schichten verwendet, was sich in allen Untersuchungen stets positiv ausgewirkt hat.

5.3 Erweiterung des Klassifikators

Nachdem das Basismodell vorgestellt wurde, werden in diesem Abschnitt die beiden ausgewählten Erweiterungen zur Extraktion von Unsicherheitsinformationen vorgestellt. Hierbei wird zunächst auf eventuell nötige strukturelle Veränderungen eingegangen und anschließend die Vorgehensweise zur Extraktion der Unsicherheiten beschrieben. Vergleichende Untersuchungen werden in Abschnitt 6.2 (Seite 100) vorgestellt.

[2] Für weitere Details zur Netzwerkstruktur sei auf Tabelle 2 in [Eisenbach et al., 2017] verwiesen.

5.3.1 Dropout-Ansatz

Der in [Gal, 2016] präsentierte Dropout-Ansatz zur Extraktion von Unsicherheitsinformationen basiert auf einer probabilistischen Interpretation von Dropout. Folglich muss das Basismodell für die Anwendung dieses Ansatzes mit Dropout-Schichten erweitert werden. Für eine möglichst gute Approximation wird in [Gal, 2016] der Einsatz von Dropout vor jeder Schicht mit Gewichten empfohlen. Die Anwendung auf die Netzwerkeingabe ist jedoch umstritten und wird in der Regel nicht durchgeführt. Da das in dieser Masterarbeit verwendete Basismodel die genannten Voraussetzungen bereits erfüllt, sind für die Verwendung dieses Ansatzes keinerlei Modifikationen an der Netzwerkstruktur erforderlich.

Zur Extraktion von Unsicherheitsinformationen werden in der Anwendungsphase, entgegen der in [Srivastava et al., 2014] vorgeschlagenen Vorgehensweise, ebenfalls Dropout-Masken gesampelt und die Eingaben der einzelnen Schichten mit diesen multipliziert. Durch eine wiederholte Verarbeitung eines Inputbildes können auf diese Weise verschiedene Netzwerkausgaben gesammelt und für die Ableitung von Unsicherheitsinformationen verwendet werden. Für gute Approximationsergebnisse wird in [Gal, 2016] die Verwendung von mindestens zehn, besser jedoch 100, Vorwärtsprozessen durch das Netzwerk empfohlen. An dieser Stelle wird der größte Nachteil dieses Ansatzes deutlich. So verzehnfacht beziehungsweise verhundertfacht sich die benötigte Zeit in der Anwendungsphase. In Abschnitt 6.2.1 (Seite 102) der experimentellen Untersuchungen wird daher analysiert, wie viele Iterationen für den in dieser Masterarbeit verwendeten Klassifikator benötigt werden.

5.3.2 Ensemble-Ansatz

Der zweite betrachtete Ansatz basiert auf der Publikation von [Osband et al., 2016] und nutzt ein Ensemble zur Approximation von Unsicherheitsinformationen. Hierfür werden jedoch nicht mehrere gleichartige Netzwerke trainiert, sondern das betrachtete Basismodell in zwei Teile untergliedert und lediglich die Struktur des zweiten Teils zur Erstellung von mehreren Ausgabeinstanzen repliziert. Die Motivation für diese Vorge-

hensweise beruht auf der Beobachtung, dass in den frühen Schichten der Netzwerke eines Ensembles meist sehr ähnliche Features gelernt werden. Folglich können diese Schichten von den einzelnen Ausgabeinstanzen geteilt werden. Der erste Netzwerkteil wird aus diesem Grund auch als Shared Network bezeichnet. Die einzelnen Ausgabeinstanzen bilden anschaulich betrachtet Köpfe auf diesem geteilten Netzwerk und werden daher Heads genannt. Abbildung 5.3 veranschaulicht die am Basismodell erforderliche strukturelle Modifikation.

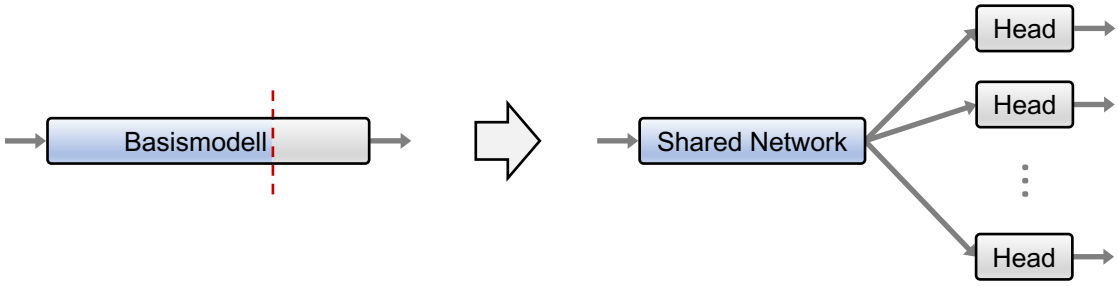


Abbildung 5.3: Strukturelle Modifikation für Ensemble-Ansatz
Dargestellt ist die Überführung des Basismodells in ein Ensemble, bestehend aus einem geteilten Netzwerk (Shared Network) und mehreren Ausgabeinstanzen (Heads).

Während durch diese Vorgehensweise der Berechnungsaufwand im Vergleich zu einem herkömmlichen Ensemble deutlich reduziert werden kann, muss bei der Teilung des Basismodells jedoch auch auf die Individualität der einzelnen Heads geachtet werden. So muss sichergestellt werden, dass die Heads einen unterschiedlichen Lernprozess durchlaufen und nicht gegen die gleiche Parameterkonfiguration konvergieren. In [Osband et al., 2016] werden hierfür eine unterschiedliche Initialisierung und eine an Bagging angelehnte Trainingsprozedur vorgeschlagen. Bei dieser Vorgehensweise werden in jedem Schritt einzelne Heads ausgewählt und für den Lernprozess aktiviert. Die verbleibenden deaktivierten Heads erfahren hingegen keine Parameteränderungen. Die Autoren argumentieren, dass dieses Vorgehen über mehrere Epochen betrachtet der Anwendung von Bagging gleicht. Zur Realisierung wird in [Osband et al., 2016] für jeden Lernschritt eine binäre Maske gesampelt, wobei jedes der enthaltenen Elemente angibt, ob der zugehörige Head im aktuellen Schritt lernt. Genaue Implementierungsdetails werden jedoch nicht zur Verfügung gestellt.

Im Folgenden wird daher kurz auf die in dieser Masterarbeit realisierte Implementierung eingegangen. Zur Umsetzung der beschriebenen Vorgehensweise wird ausschließlich der Rückwärtsprozess durch das Netzwerk modifiziert. Dadurch kann in jeder Iteration trotzdem der Fehler des gesamten Ensembles bestimmt und somit eine Aussage zur Klassifikationsleistung auf den Trainingsdaten abgeleitet werden. Abbildung 5.4 veranschaulicht die veränderten Gradientenflüsse durch das Netzwerk.

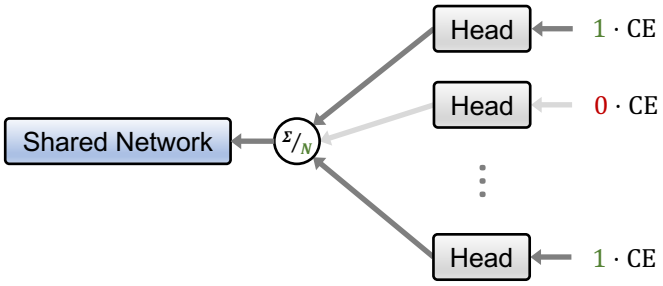


Abbildung 5.4: Veranschaulichung der implementierten Trainingsprozedur
*Dargestellt sind die Gradientenflüsse in der Lernphase zur Realisierung der in [*Osband *et al., 2016] vorgestellten Trainingsprozedur. Die an den Ausgabeinstanzen bestimmten Fehler (Kreuzentropien) werden mit den entsprechenden Elementen der binären Maske multipliziert. Da deaktivierte Heads somit keinen Fehler erzeugen, werden weder Gradienten durch diese Heads noch in das geteilte Netzwerk propagiert. Für die aktivierten Heads werden hingegen Gradienten erzeugt und die Gewichte entsprechend angepasst. Alle zum geteilten Netzwerk propagierten Fehler werden mit der Anzahl der aktivierten Heads skaliert und zu einem Gesamtgradienten aufsummiert.*

Die zur Maskierung der Heads verwendete multiplikative Verknüpfung bietet den Vorteil, dass keine bedingten Verzweigungen in den Berechnungsgraph eingebracht werden. Trotz eventuell zusätzlich durchgeführten Berechnungen resultiert dies in einer deutlich schnelleren Ausführungsgeschwindigkeit. Die von den Autoren vorgeschlagene Gradientenskalierung mit der Gesamtanzahl der Heads führte in dieser Masterarbeit stets zu sehr instabilen Lernprozessen. Die zum geteilten Netzwerk propagierten Gradienten werden daher stattdessen in jedem Lernschritt mit der Anzahl der aktivierten Heads normiert. Auf diese Weise ist der in das geteilte Netzwerk propagierte Gesamtgradient stets proportional zur Anzahl der aktivierten Heads. Zur zusätzlichen Erhöhung der

Diversität können zudem binäre Gewichte für die Beispiele eines Mini-Batches verwendet werden. Diese ebenfalls in jedem Lernschritt analog zur Head-Maske und für jeden Head getrennt gesampelten Gewichte ermöglichen das Ausblenden einzelner Beispiele bei der Berechnung des mittleren Fehlers und sorgen folglich dafür, dass die aktivierten Heads nicht exakt mit den gleichen Daten eines präsentierten Mini-Batches lernen. Da sich durch diese Vorgehensweise jedoch die effektive Batchgröße verringert, sollte dies bei der Festlegung der Lernrate beachtet und diese proportional zur resultierenden, effektiven Batchgröße verkleinert werden. Die Verwendung dieser zusätzlichen Gewichte hat sich in dieser Masterarbeit stets positiv ausgewirkt, erhöht jedoch gegebenenfalls die Anzahl der für das Training benötigten Epochen.

Ergänzend sei angemerkt, dass die Klassifikationsleistung eines auf diese Weise gebildeten Ensembles nicht mit der eines herkömmlichen Ensembles verglichen werden kann. Bei einem aus mehreren Netzwerken bestehenden Ensemble wird die beste Epoche in der Regel für jedes Einzelnetzwerk getrennt bestimmt und der Gesamtklassifikator aus den Netzwerken in ihrer jeweils besten Konfiguration gebildet. Für das vorgestellte Ensemble ist dies nicht möglich, da die Gewichte der Heads stets an eine bestimmte Parameterkonfiguration der geteilten Schichten gebunden sind[3]. Folglich ist es sehr wahrscheinlich, dass der normalerweise beobachtete Zugewinn in der Klassifikationsleistung ausbleibt. Im Fokus dieser Arbeit steht jedoch auch nicht die Verbesserung der Klassifikationsleistung durch einen Ensemble-Ansatz, sondern die Extraktion von Unsicherheitsinformationen.

Nachdem der Ensemble-Ansatz und dessen Implementierung vorgestellt wurden, können nun die sich für das in dieser Masterarbeit verwendete Basismodell ergebenden Konsequenzen betrachtetet werden. So wird das Basismodell vor dem Übergang in die vollverschalteten Schichten geteilt. Diese Entscheidung kann durch die enorm hohe Anzahl an Gewichten in der ersten vollverschalteten Schicht begründet werden. In dieser befinden sich mehr als 57% aller Gewichte. Die beiden darauffolgenden Schichten

[3] Für die Realisierung einer zum herkömmlichen Ensemble vergleichbaren Parameterauswahl müssten die Gewichte des geteilten Netzwerks zu einem bestimmten Zeitpunkt festgesetzt und anschließend nur noch die Heads trainiert werden. In dieser Masterarbeit hat sich jedoch gezeigt, dass auch in späten Epochen betragsmäßig relevante Gradienten in das geteilte Netzwerk propagiert werden. Folglich kann ein geeigneter Zeitpunkt nur sehr schwer abgeschätzt werden.

machen zusammen lediglich einen Anteil von weniger als 13% aus. Zur Gewährleistung von Diversität muss die Netzwerkstruktur daher vor der ersten vollverschalteten Schicht geteilt werden. Abbildung 5.5 verdeutlicht die sich ergebende Netzwerkstruktur des mit Hilfe des Ensemble-Ansatzes erweiterten Klassifikators.

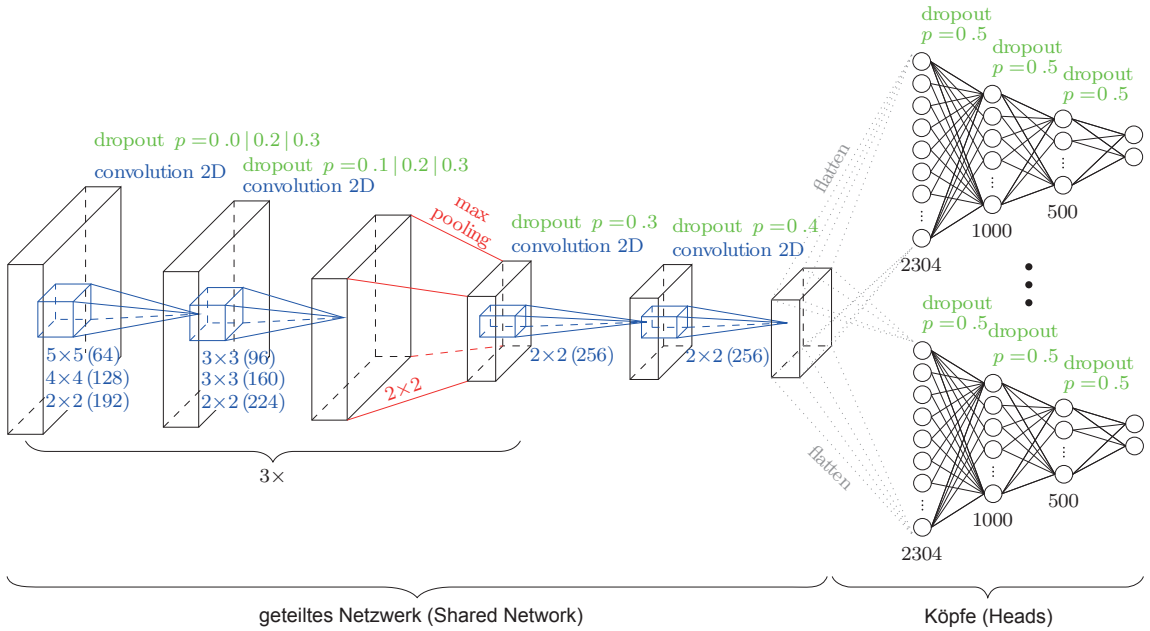


Abbildung 5.5: Modifikationen am Basismodell für Ensemble-Ansatz
Dargestellt ist die erweiterte Netzwerkstruktur für das betrachtete Basismodell. Die vollverschalteten Schichten werden für die Erstellung der Heads mehrfach repliziert. Die verbleibenden Schichten bilden das geteilte Netzwerk.

Während beim Dropout-Ansatz zur Extraktion von Unsicherheitsinformationen mehrere Vorwärtsprozesse durch das Netzwerk benötigt werden und es somit zu einer deutlich zeitaufwendigeren Anwendungsphase kommt, genügt bei einer Erweiterung mittels Ensemble-Ansatz die einmalige Anwendung des erweiterten Klassifikators. Die Ausgaben der einzelnen Heads bilden die Grundlage für die Ableitung von Unsicherheitsinformationen. In Abschnitt 6.2.2 (Seite 105) der experimentellen Untersuchungen wird daher untersucht, wie viele Heads für das in dieser Masterarbeit verwendete Basismodell benötigt werden.

5.4 Inkrementeller Lernprozess

Im Folgenden werden die einzelnen Teilschritte der iterativen Vorgehensweise zur Realisierung eines intelligenten inkrementellen Lernprozesses genauer betrachtet und wichtige Implementierungsdetails beschrieben. Hierzu wird zunächst auf die Anwendungsphase des erweiterten Klassifikators und auf die Vorgehensweise zur Extraktion von neuen Bildausschnitten eingegangen. Anschließend werden mögliche Szenarien für die Erweiterung der Trainingsdaten vorgestellt und wichtige Aspekte für das Nachtraining des Klassifikators genannt.

5.4.1 Anwendung auf unbekannte Daten

Damit neue Trainingsbeispiele ausgewählt werden können, muss der erweiterte Klassifikator auf unbekannte Daten angewendet werden. In der Regel handelt es sich dabei nicht um bereits in Patches geschnittene Bilder, sondern um hochauflösende Aufnahmen, wie die in dieser Masterarbeit untersuchten Fahrbahnbilder. Im nachfolgenden Abschnitt wird daher beschrieben, wie diese hochauflösenden Bilder verarbeitet werden können. Im Anschluss wird erläutert, wie unter Berücksichtigung der in der Vorverarbeitung extrahierten Masken neue Patches extrahiert werden können.

5.4.1.1 Verarbeitung von hochauflösenden Bildern

Für die Verarbeitung von hochauflösenden Bildern ergeben sich generell zwei mögliche Vorgehensweisen. Entweder wird das Ausgangsbild in entsprechend große Patches geschnitten (Sliding-Window-Ansatz), oder die Struktur des Klassifikators wird ein weiteres Mal modifiziert, sodass dieser auf beliebige Bildgrößen angewendet werden kann. Für die zuletzt genannte Vorgehensweise müssen, wie schon in Abschnitt 4.4.2.1 (Seite 58) im Rahmen der Vorverarbeitung der Fahrbahnbilder erwähnt, alle vollverschalteten Schichten in äquivalente Faltungs-Schichten umgewandelt werden. Bereits für [Eisenbach et al., 2016][4] wurde hierfür ein Verfahren entwickelt, bei welchem die Netzwerkstruktur, ausgehend von der Eingabe in Richtung der Ausgabe, analysiert

[4] Die Publikation enthält eigene Beiträge zur Anwendung von Klassifikatoren auf ganze Bilder.

und beginnend ab der ersten vollverschalteten Schicht die restliche Netzwerkstruktur mit Faltungs-Schichten neu konstruiert wird. Abbildung 5.6 verdeutlicht, dass durch eine Neuanordnung der Eingabeelemente und eine geschickte Überführung der Kantengewichte in Filtermatrizen jede vollverschaltete Schicht in eine äquivalente Faltungs-Schicht umgewandelt werden kann[5].

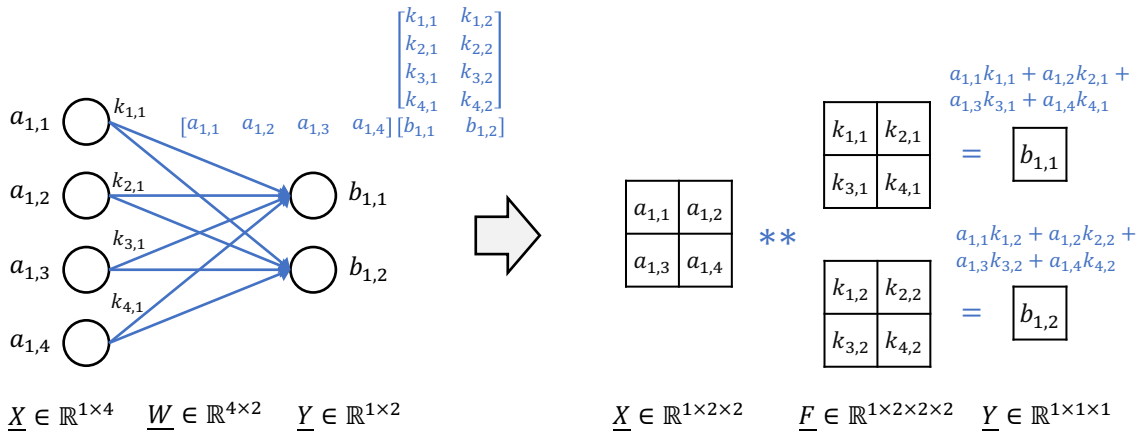


Abbildung 5.6: Umwandlung vollverschaltete Schicht in Faltungs-Schicht
Die Abbildung zeigt, wie eine vollverschaltete Schicht in eine äquivalente Faltungs-Schicht umgewandelt werden kann. Hierbei sind $\underline{\mathbf{X}}$ *die Eingaben,* $\underline{\mathbf{Y}}$ *die Ausgaben und* $\underline{\mathbf{W}}$ *beziehungsweise* $\underline{\mathbf{F}}$ *die Gewichte der jeweiligen Schicht.*

Die für die erste vollverschaltete Schicht bestimmten Ersatzfilter besitzen folglich immer exakt die gleiche Größe wie die Feature-Maps des letzten Output-Volumes. Weiterhin wird deutlich, dass sich die Anzahl der zu verwendenden Filter nach der Neuronenanzahl in der zu ersetzenden vollverschalteten Schicht richtet und dass sich aufgrund von einelementigen Output-Feature-Maps für alle anschließenden Faltungs-Schichten 1x1 Filter ergeben. Seit dem Erfolg sogenannter Network-in-Network-Architekturen[6] [Lin et al., 2013] existieren auch für derart kleine Filter sehr effiziente Implementierungen.

[5] Je nach Implementierung der Faltungsoperation, kann es erforderlich sein, die Filter zusätzlich entlang beider Dimensionen zu spiegeln.

[6] Network-in-Network-Architekturen (NIN) nutzen vollverschaltete Schichten, um positionsgleiche Elemente in verschiedenen Feature-Maps neu zu kombinieren. Damit diese Schichten zwischen Faltungs-Schichten eingebracht werden können, müssen diese mit Hilfe von 1x1-Faltungen umgesetzt werden.

Im Rahmen dieser Masterarbeit wurden die Verarbeitungszeiten für beide Vorgehensweisen (Sliding-Window-Ansatz und Strukturmodifikation) anhand des Basismodells verglichen. Bei einer Patchgröße von 64x64 Pixeln und lediglich halbseitiger Überlappung (32 Pixel) zur Realisierung einer nur sehr groben Lokalisierungsgenauigkeit ergeben sich für die Verarbeitung eines Fahrbahnbildes mittels Sliding-Window-Ansatz bereits 26.460 Patches, welche mit Hilfe einer NVIDIA GTX 1080 in 6,436 Sekunden abgearbeitet werden können. Da bei einer Verarbeitung des gesamten Bildes[7] redundante Berechnungen vermieden werden können, ist die Verarbeitungszeit in diesem Fall mit 3,345 Sekunden deutlich kürzer. Zudem ergibt sich bei der Verarbeitung mit Hilfe der modifizierten Netzwerkstruktur eine deutlich feinere Überlappung im Ausgangsbild von lediglich 8 Pixeln[8]. Für einen äquivalenten Sliding-Window-Ansatz mit der gleichen Lokalisierungsgenauigkeit müssten für jedes einzelne Fahrbahnbild insgesamt 420.540 Patches verarbeitet werden, was in einer Verarbeitungszeit von 102,290 Sekunden pro Fahrbahnbild resultieren würde. Gerade im Hinblick auf wiederholte Netzwerkanwendungen zur Extraktion von Unsicherheitsinformationen bei der Anwendung des Dropout-Ansatzes zeigt dieser Vergleich den Stellenwert dieser Modifikation und dass für diese Masterarbeit ausschließlich die Verarbeitung der kompletten Fahrbahnbilder in Frage kommt.

5.4.1.2 Extraktion von Patches

Infolge der Modifikation der Netzwerkstruktur besteht die Klassifikatorausgabe ebenfalls aus einem Output-Volume. Die positionsgleichen Elemente in den einzelnen Feature-Maps beinhalten dabei jeweils die Softmax-Ausgabe für einen bestimmten Teilbereich der Fahrbahnbilder. Für die angestrebte Extraktion ausgewählter Patches, ausgehend von der Klassifikatorausgabe, müssen die den einzelnen Ausgabeelementen zugrunde liegenden Bildbereiche bestimmt werden. Aufgrund der Struktur des Basismodells ist bekannt, dass ein kompletter Patch auf eine einzige Softmax-Ausgabe

[7] Ein Fahrbahnbild kann selbst mit 8GB Grafikspeicher nicht komplett verarbeitet werden und wird daher in mehrere Blöcke mit einer Größe von 1024x3496 Pixeln unterteilt. Zur Gewährleistung von exakter Ergebnisgleichheit müssen sich diese Blöcke an den Schnittkanten um 64-8 Pixel (Patchgröße - Produkt aller Verschiebungen in der jeweiligen Dimension) überlappen.

[8] Details hierzu werden im nachfolgenden Abschnitt 5.4.1.2 erläutert.

abgebildet wird. Die Größe der einzelnen Teilbereiche entspricht daher der in der Trainingsphase verwendeten Patchgröße von 64x64 Pixeln. Für die Berechnung der genauen Positionen der Teilbereiche im Eingabebild müssen die in den einzelnen Schichten genutzten Strukturparameter berücksichtigt werden. So wirkt sich sowohl der Einsatz von Zero-Padding als auch die Verwendung von mehrelementigen Verschiebungen (engl. Strides) auf die Positionen der Bereiche aus. Da die Faltungs- und Pooling-Schichten in dem für diese Masterarbeit verwendeten Klassifikator kein zusätzliches Zero-Padding nutzen, entfällt die Betrachtung dieser Größen. Mehrelementige Verschiebungen werden zudem nur in den drei Pooling-Schichten des Netzwerks eingesetzt. In jeder Pooling-Schicht werden jeweils 2x2 Eingabeelemente zu einem Ausgabeelement zusammengefasst, wobei sich die einzelnen Pooling-Bereiche nicht überlappen sollen. Folglich wird für diese Schichten eine zweielementige Verschiebung verwendet. Für die Berechnung der Positionen bedeutet dies, dass sich der Aufpunkt[9] im Ausgangsbild zwischen zwei Elementen in der Ausgabe infolge dieser drei Schichten um insgesamt $2 \cdot 2 \cdot 2 = 8$ Pixel verschiebt. Die Lage des einer Softmax-Ausgabe zugrunde liegenden Bildbereiches kann somit durch eine einfache Multiplikation der in der Ausgabe betrachteten Position mit der aus den einzelnen Schichten abgeleiteten Gesamtverschiebung berechnet werden.

Bevor der für eine Ausgabe bestimmte Bildausschnitt extrahiert und verwendet werden kann, müssen weitere Randbedingungen überprüft werden. So darf ein Patch nur dann in die Trainingsdaten aufgenommen werden, wenn sich dieser innerhalb des Fahrbahnbereiches und nicht über einer Straßenmarkierung oder einer Stitching-Kante befindet. Zur Überprüfung werden die in der Vorverarbeitung bestimmten verbotenen Bereiche (siehe Abschnitt 4.4.2 (Seite 58)) in einer Gesamtmaske zusammengefasst und für jeden zu extrahierenden Patch ebenfalls der zugehörige Teilbereich in dieser Maske betrachtet. Ist kein Element in diesem Bereich der Maske gesetzt, erfüllt der betrachtete Bildausschnitt die genannten Anforderungen und kann verwendet werden. Weiterhin definiert der für den GAPs-Datensatz genutzte Extraktionsprozess für jede der beiden Klassen eine bestimmte Mindestverschiebung zwischen zwei benachbarten

[9] Der Aufpunkt befindet sich stets oben links.

Patches. So dürfen Ausschnitte von Schadstellen nur dann extrahiert werden, wenn sich diese in mindestens acht Bildzeilen oder -spalten unterscheiden. Da sich durch die verwendete Vorgehensweise stets eine Verschiebung von acht Pixeln ergibt, müssen Patches für die Schadklasse nicht zusätzlich überprüft werden. Für Bildausschnitte von intakten Straßenverhältnissen wird hingegen eine halbseitige Überlappung (32 Pixel) definiert. Diese Anforderung kann realisiert werden indem das zentrale Pixel von jedem bereits extrahierten Patch in die beschriebene Gesamtmaske der verbotenen Bereiche übernommen wird. Abbildung 5.7 verdeutlicht die bei der Extraktion zu beachtenden Randbedingungen an einem Fahrbahnbild.

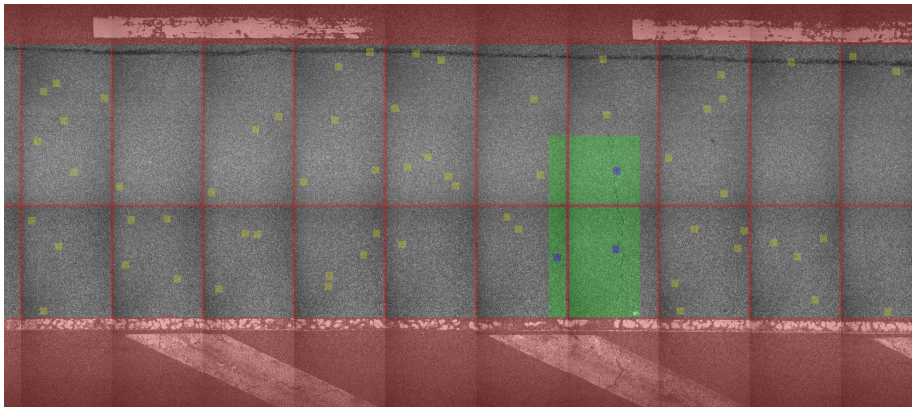


Abbildung 5.7: Extraktion von Patches am Beispiel
Die Abbildung zeigt ein Fahrbahnbild, für welches an zufälligen Positionen neue Patches extrahiert wurden. Die für Klasse 0 (intakt) ausgewählten Bildbereiche sind gelb und die für Klasse 1 (Schaden) extrahierten Patches blau überlagert dargestellt. Der grün markierte Bereich verdeutlicht die verfügbaren Labelinformationen. Es ist deutlich zu erkennen, dass keine Patches aus den verbotenen Bereichen (rot überlagert) extrahiert werden.

5.4.2 Auswahl neuer Trainingsbeispiele und Nachtraining

Nachdem die allgemeine Vorgehensweise zur Extraktion von Patches beschrieben wurde, thematisiert dieser Abschnitt konkrete Szenarien zur Auswahl neuer Trainingsbeispiele und das anschließende Nachtraining des Klassifikators.
In herkömmlichen inkrementellen Lernprozessen werden den Trainingsdaten zufälli-

ge oder nur vom Netzwerk falsch klassifizierte Patches hinzugefügt. Im Rahmen dieser Masterarbeit wird diese Vorgehensweise um eine zusätzliche Berücksichtigung der Unsicherheit in den Klassifikatorentscheidungen erweitert. Auf Basis der Netzwerkausgaben des erweiterten Klassifikators wird hierzu unter Zuhilfenahme der in Abschnitt 3.2.3 (Seite 29) vorgestellten Maße für jeden in Frage kommenden Patch eine zusätzliche Aussage zur Unsicherheit der Klassifikation abgeleitet und mit der von anderen Patches verglichen. Auf diese Weise sollen dem Datensatz nur möglichst informative Beispiele hinzugefügt und redundante Daten vermieden werden.
Neben einer Erweiterung der herkömmlich verwendeten Auswahlprozesse, können neue Trainingsbeispiele auch nur auf Basis der bestimmten Unsicherheit ausgewählt werden. Für diese Masterarbeit ergeben sich daher insgesamt vier mögliche Szenarien zur Erweiterung der Trainingsdaten:

Zufällige Auswahl

Dem Trainingsdatensatz werden ausschließlich zufällig ausgewählte Patches hinzugefügt.

Auswahl anhand des Klassifikationsergebnisses

Dem Trainingsdatensatz werden ausschließlich die Beispiele hinzugefügt, welche falsch klassifiziert wurden.

Auswahl anhand der Unsicherheit

Dem Trainingsdatensatz werden ausschließlich die Beispiele mit der höchsten Unsicherheit hinzugefügt.

Auswahl anhand des Klassifikationsergebnisses und der Unsicherheit

Dem Trainingsdatensatz werden ausschließlich sehr sichere, aber dennoch falsch klassifizierte Beispiele hinzugefügt. Zur Verfeinerung der Klassengrenzen können auch falsch klassifizierte Patches mit höchster Unsicherheit gewählt werden.

Die Labelzuordnung erfolgt in allen Szenarien anhand der in der Vorverarbeitung bestimmten Labelinformationen. Da diese jedoch sehr großflächig sind, werden die beschriebenen Szenarien sowohl über beide Klassen (intakter Straßenausschnitt, Straßenausschnitt mit Schaden) als auch für jede Klasse getrennt betrachtet.

Nachdem den Trainingsdaten neue Beispiele hinzugefügt wurden, wird ein neuer Lernprozess initiiert und der erweiterte Klassifikator mit dem modifizierten Datensatz erneut trainiert. Im Rahmen der in dieser Masterarbeit realisierten Vorgehensweise werden die Trainingsdaten stetig erweitert. Ein Entfernen oder eine abklingende Gewichtung alter Beispiele führte stets zu einer Überspezialisierung auf die zuletzt hinzugefügten Patches und wird daher nicht verwendet.
In Abschnitt 6.2 (Seite 100) der experimentellen Untersuchungen werden zunächst die verschiedenen Maße zur Modellierung von Unsicherheiten verglichen. Anschließend wird das Gesamtsystem in Abschnitt 6.3 (Seite 107) auf die verfügbaren Fahrbahnbilder angewendet und untersucht, ob die Klassifikationsleistung mit diesen verbessert werden kann.

Kapitel 6

Experimentelle Untersuchungen

In diesem Teil der Masterarbeit werden die durchgeführten experimentellen Untersuchungen für die in Kapitel 5 vorgestellte Vorgehensweise zur Realisierung von intelligentem inkrementellem Lernen präsentiert. Hierzu werden zunächst die beiden zur Extraktion von Unsicherheitsinformationen genutzten Ansätze hinsichtlich ihrer Eignung für einen inkrementellen Lernprozess untersucht. Anschließend werden die in den State-of-the-Art-Betrachtungen vorgestellten Maße zur Modellierung von Unsicherheit auf beide Ansätze angewendet und ein für diese Masterarbeit geeignetes Gesamtsystem abgeleitet. Abschließend wird dieses auf die verfügbaren Fahrbahnbilder angewendet und untersucht, ob die Klassifikatorleistung mit der angedachten Vorgehensweise inkrementell verbessert werden kann.

6.1 Eignung für inkrementellen Lernprozess

Bevor die beiden zur Erweiterung des Basismodells verwendeten Ansätze für die Extraktion von Unsicherheitsinformationen genutzt wurden, wurde zunächst überprüft, ob die erweiterten Basis-Klassifikatoren die Realisierung eines inkrementellen Lernprozesses zulassen. Um einen Einfluss von ungenauen Labelinformationen auf die Ergebnisse ausschließen zu können, wurden die Experimente ausschließlich auf dem GAPs-Datensatz durchgeführt. Für die Untersuchungen wurden die verfügbaren 50.000 Trainingsbeispiele gemischt und in fünf Teile untergliedert. Tabelle 6.1 zeigt die genaue

Aufteilung des Trainingsdatensatzes und fasst die Klassenverteilungen zusammen.

	Datensatzteil				
	I	II	III	IV	IV
Anzahl an Beispielen	10.240	10.240	10.240	10.240	9.040
⊢ Klasse 0: Intakt	6.172	6.044	6.192	6.143	5449
⊢ Klasse 1: Schaden	4.068	4.196	4.048	4.097	3.591
kumulative Anzahl an Beispielen	10.240	20.480	30.720	40.960	50.000

Tabelle 6.1: Teilung GAPs-Trainingsdatensatz

*Die Tabelle zeigt die Teilung des GAPs-Trainingsdatensatzes für die durchgeführten experimentellen Untersuchungen. Mit Ausnahme des letzten Datensatzteils wurden die Größen so bestimmt, dass diese ganzzahlige Vielfache der in [*Eisenbach *et al., 2017] vorgeschlagenen Batchgröße von 256 Beispielen sind.*

Für beide Ansätze wurden zunächst fünf Referenz-Klassifikatoren trainiert, wobei die für das Training verwendeten Daten, beginnend beim ersten Datensatzteil, für jeden Klassifikator um den jeweils nächsten Teil erweitert wurden (siehe letzte Zeile in Tabelle 6.1). Die Ergebnisse dieser Referenz-Klassifikatoren ermöglichen die Einschätzung und Bewertung der im Anschluss inkrementell trainierten Klassifikatoren.

Die Gewichte der inkrementell trainierten Klassifikatoren wurden mit Hilfe der Parameter des ersten Referenz-Klassifikators initialisiert. Während des Lernprozesses wurden dem Trainingsdatensatz, ausgehend vom ersten Datensatzteil, in jeder Iteration 1.024 zufällig gewählte Beispiele hinzugefügt bis die Gesamtanzahl von 50.000 Beispielen erreicht wurde. Um einen fairen Vergleich mit den einzelnen Referenz-Klassifikatoren zu ermöglichen, konnten neue Beispiele dabei nur aus dem jeweils nächsten Datensatzteil gezogen werden. Die inkrementellen Lernprozesse wurden stets dreimal wiederholt und die Ergebnisse über alle Durchläufe gemittelt.

Für die Evaluation wurde, entsprechend der Empfehlung in [Eisenbach et al., 2017], der sich aus der Precision-Recall-Kurve maximal ergebende F1-Score auf den Validierungsdaten genutzt. Für dessen Berechnung wird für alle sich aus den Softmax-Ausgaben ergebenden Schwellwerte der F1-Score ermittelt und anschließend der Maximalwert ausgewählt.

Da sich im Vorfeld der Masterarbeit gezeigt hat, dass durch den Einsatz von Adversarial Training[1] bessere Klassifikationsergebnisse erzielt werden können, wurden die Experimente ebenfalls unter Verwendung von Adversarial Training durchgeführt. Die Adversarial Examples wurden dabei mittels der Fast-Gradient-Sign-Methode mit einer Schrittweise von $\epsilon = {}^{2\cdot 4}/_{255}$ (entspricht vier Graustufen) erstellt und deren Fehler gleichgewichtet ($\alpha = 0,5$) in der Kostenfunktion berücksichtigt.

6.1.1 Dropout-Ansatz

In Anlehnung an [Eisenbach et al., 2017][2] wurde für das Training der Referenz-Klassifikatoren eine konstante Lernrate von 0,01, ein unveränderliches Momentum von 0,9 und eine Batchgröße von 256 Beispielen verwendet. Während des Trainings hat sich gezeigt, dass sich die Klassifikatoren spätestens nach 350 Epochen in der Sättigung befinden und der F1-Score auf den Validierungsdaten rückläufig ist. Das Training wurde daher für alle Referenz-Klassifikatoren nach maximal 400 Epochen abgebrochen und jeweils das Ergebnis der besten Epoche ausgewählt. Tabelle 6.2 zeigt die erzielten Klassifikationsergebnisse.

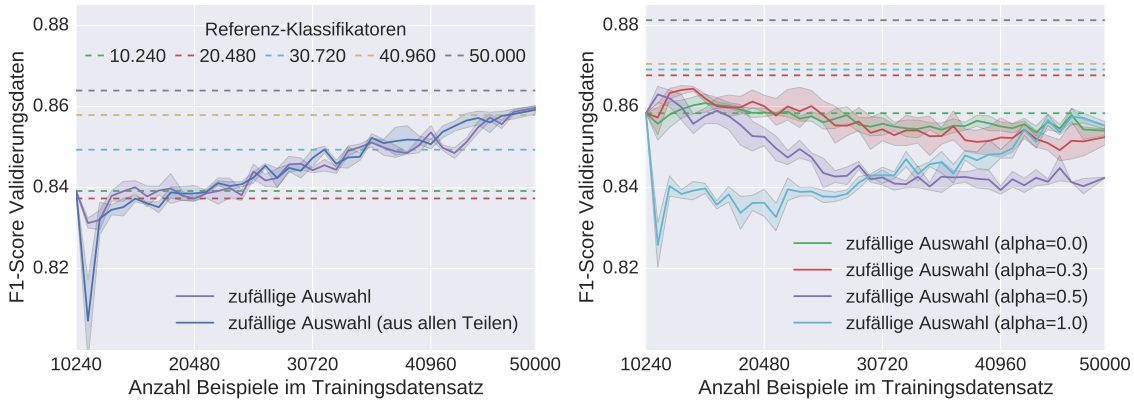
	Anzahl Trainingsbeispiele Referenz-Klassifikator				
	10.240	20.480	30.720	40.960	50.000
F1-Score	0,8392 (190)	0,8373 (171)	0,8494 (206)	0,8579 (342)	0,8640 (183)
F1-Score (AT)	0,8584 (300)	0,8677 (138)	0,8692 (165)	0,8705 (103)	0,8813 (346)

Tabelle 6.2: Erzielte F1-Scores der Referenz-Klassifikatoren
Dargestellt sind die von den Referenz-Klassifikatoren erzielten F1-Scores. Die jeweils beste Epoche ist in Klammern angegeben. „AT“ zeigt die Verwendung von Adversarial Training an.

Die Parameter für die im Anschluss inkrementell trainierten Klassifikatoren wurden aufbauend auf den Erkenntnissen der Referenz-Klassifikatoren festgelegt. Während die Batchgröße und das Momentum unverändert übernommen wurden, wurde für die Lernrate zusätzlich ein Zehntel des bei den Referenz-Klassifikatoren verwendeten Wertes

[1] Adversarial Training wurde in theoretischen Grundlagen in Abschnitt 2.2.3 (Seite 13) vorgestellt.
[2] Die Publikation enthält eigene Beiträge zur Festlegung der genutzten Lernparameter.

getestet, was jedoch in allen Fällen zu geringfügig schlechteren Ergebnissen geführt hat. Aus diesem Grund wurde erneut eine konstante Lernrate von 0,01 verwendet[3]. Nach jeder Erweiterung der Trainingsdaten wurde der Klassifikator für 40 zusätzliche Epochen trainiert. Durch die Wahl dieser Epochenanzahl wird der Klassifikator für jeden komplett hinzugefügten Datensatzteil weitere 400 Epochen trainiert. In jeder Iteration wurde die beste Parameterkonfiguration bestimmt und die Gewichte vor der nächsten Erweiterung auf diese Konfiguration zurückgesetzt. Diese Vorgehensweise hat zur Folge, dass auch Verschlechterungen toleriert werden, welche jedoch möglicherweise für das Überwinden lokaler Minima in der Fehlerfunktion erforderlich sind. Abbildung 6.1 stellt die Ergebnisse der Untersuchungen dar.



(a) Ohne Adversarial Training **(b)** Mit Adversarial Training

Abbildung 6.1: Ergebnisse Eignung Dropout-Ansatz

Dargestellt sind die Ergebnisse (mittlerer F1-Score und einfache Standardabweichung) der experimentellen Untersuchungen zur Eignung des Dropout-Ansatzes für einen inkrementellen Lernprozess.

Die in Abbildung 6.1a dargestellten Ergebnisse (violette Kurve) zeigen, dass der angestrebte inkrementelle Lernprozess erfolgreich realisiert werden konnte. Die Klassifikationsleistung nimmt durch das Hinzufügen neuer Trainingsbeispiele kontinuierlich zu, liegt jedoch stets leicht unter der der entsprechenden Referenz-Klassifikatoren. Die Leistung des mit dem kompletten Datensatz trainierten Referenz-Klassifikators kann

[3] Konfigurationen mit abnehmenden Lernraten konnten aufgrund des begrenzten Zeitrahmens in dieser Masterarbeit nicht untersucht werden.

zu keinem Zeitpunkt erreicht werden. Um auszuschließen, dass die Ergebnisse durch die gewählte zufällige Teilung des Datensatzes beeinflusst werden, wurde der inkrementelle Lernprozess ein zweites Mal wiederholt, mit dem Unterschied, dass neue Beispiele für diese zusätzliche Untersuchung zufällig aus allen Datensatzteilen gezogen werden konnten[4]. Der Verlauf der in Abbildung 6.1a blau dargestellten Kurve zeigt jedoch, dass ein derartiger Einfluss ausgeschlossen werden kann.

Die in Abbildung 6.1b dargestellten Ergebnisse bestätigen, dass mit Hilfe von Adversarial Training deutlich höhere F1-Scores erzielt werden können, zeigen jedoch auch, dass kein inkrementeller Lernprozess realisiert und nicht zu der Leistung der Referenz-Klassifkatoren aufgeschlossen werden kann (violette Kurve). Die zusätzlich durchgeführten Untersuchungen mit stärkerer Gewichtung der Adversarial Examples ($\alpha < 0,5$; siehe Formel (2.5), Seite 14) legen nahe, dass die dem Datensatz neu hinzugefügten Beispiele dem bisherigen Lernprozess schaden und dass Verbesserungen ausschließlich mit erzeugten Adversarial Examples erzielt werden können. Der Lernprozess befindet sich jedoch in allen Fällen in einem stark ausgeprägten Minimum, welches nicht überwunden werden kann. Untersuchungen mit halbierter Batchgröße oder abweichenden Lernraten führten zu ähnlichen Ergebnissen. Die hellblau dargestellte Kurve zeigt jedoch, dass ohne Adversarial Training ($\alpha = 1,0$) aus diesem lokalen Minimum ausgebrochen und nach einer gewissen Stabilisierungsphase erneut zu einem funktionierenden inkrementellen Lernprozess übergegangen werden kann. Für diese Masterarbeit kann daher geschlussfolgert werden, dass der Dropout-Ansatz nicht in Kombination mit Adversarial Training verwendet werden sollte.

6.1.2 Ensemble-Ansatz

Im Anschluss wurde die Eignung der mit Hilfe des Ensemble-Ansatzes erweiterten Basis-Klassifikatoren überprüft. Da in diesem Fall vor dem Training eine Headanzahl festgelegt werden muss, wurden die Untersuchungen an dieser Stelle für verschiedene Anzahlen durchgeführt, wobei sich die Maximalanzahl an dem auf der jeweilig genutz-

[4] Die Vorgehensweise entspricht einer erneuten zufälligen Zuordnung der verfügbaren Beispiele zu den Datensatzteilen II bis V.

ten Grafikkarte verfügbaren Speicher gerichtet hat. Ohne Adversarial Training konnten auf einer NVIDIA GTX 1080 (8GB) Netzwerke mit bis zu zehn Heads trainiert werden. Da sich die Speicherauslastung durch den Einsatz von Adversarial Training jedoch ungefähr verdoppelt, konnten für die Untersuchungen mit Adversarial Training trotz Verwendung einer NVIDIA Titan X (Maxwell, 12 GB) nicht mehr als sieben Heads genutzt werden.

Analog zu den vorherigen Untersuchungen wurden zuerst jeweils fünf Referenz-Klassifikatoren trainiert. Die dabei verwendeten Parameter orientieren sich an denen aus den Experimenten zum Dropout-Ansatz, wurden jedoch entsprechend des Ensemble-Ansatzes angepasst. Um überhaupt ein Training mit bis zu zehn Heads ermöglichen zu können, wurde zunächst die Batchgröße halbiert und auf einen Wert von 128 reduziert. Zusätzlich wurden die in Abschnitt 5.3.2 (Seite 80) vorgestellten Sample Weights verwendet, wobei die Wahrscheinlichkeit, dass ein einzelnes Beispiel in die Berechnung des Gesamtfehlers eingeht, zu einem Wert von $^{100}/_{128} \approx 0{,}78125$ gewählt wurde. Folglich ergibt sich eine effektive Batchgröße von 100 Beispielen. Um diese Veränderung in den Lernparametern zu kompensieren, wurde eine Lernrate von $^{0{,}01}/_{256} \cdot 100 \approx 0{,}0039$ verwendet. Die Wahrscheinlichkeit, dass ein einzelner Head in einem Lernschritt Gewichtsänderungen erfährt, wurde in Anlehnung an [Osband et al., 2016] zu 0,5 gewählt und die Epochenzahl entsprechend auf 800 verdoppelt. Tabelle 6.3 zeigt die auf Basis der mittleren Ausgabe erzielten Klassifikationsergebnisse.

Die in Tabelle 6.3 dargestellten Ergebnisse für die Referenz-Klassifikatoren ohne Adversarial Training verdeutlichen, dass die Klassifikationsleistung mit mehr Daten zunimmt und dass sich die Verwendung zusätzlicher Heads ebenfalls leicht positiv auswirkt. Der mit allen Trainingsbeispielen und zehn Heads trainierte Klassifikator erzielt einen F1-Score, welcher nur geringfügig unter dem Ergebnis des Dropout-Ansatzes (siehe Tabelle 6.2, Seite 95) liegt. In Anhang A.1 (Seite 129) wird die Leistung zusätzlich mit der eines Ensembles aus mehreren eigenständigen Netzwerken verglichen. An dieser Stelle wird deutlich, dass der normalerweise von einem Ensemble erwartete Zugewinn in der Klassifikationsleistung wohl infolge der geteilten Netzwerkarchitektur ausbleibt[5].

[5] Die Thematik wurde bereits in Abschnitt 5.3.2 (Seite 80) ausführlich erläutert.

	Anzahl Trainingsbeispiele Referenz-Klassifikator				
	10.240	20.480	30.720	40.960	50.000
4 Heads	0,8318 (220)	0,8569 (160)	0,8467 (677)	0,8571 (521)	0,8580 (091)
7 Heads	0,8333 (227)	0,8580 (260)	0,8477 (746)	0,8553 (067)	0,8583 (183)
10 Heads	0,8337 (234)	0,8575 (160)	0,8483 (503)	0,8578 (123)	0,8607 (095)
4 Heads (AT)	0,8415 (651)	-	-	-	0,8638 (463)
7 Heads (AT)	0,8323 (606)	-	-	-	0,8567 (788)

Tabelle 6.3: Erzielte F1-Scores der Referenz-Klassifikatoren

Dargestellt sind die von den Referenz-Klassifikatoren erzielten F1-Scores. Die jeweils beste Epoche ist in Klammern angegeben. „AT“ zeigt die Verwendung von Adversarial Training an. Mit „-“ gekennzeichnete Klassifikatoren wurden nicht trainiert.

Die im unteren Teil von Tabelle 6.3 dargestellten Ergebnisse zeigen, dass die Verwendung von Adversarial Training nur zu minimal besseren Ergebnissen führt. Die für den Dropout-Ansatz beobachtete deutliche Verbesserung bleibt aus. Im Gegenteil scheint die Leistung bei der Verwendung von zusätzlichen Heads sogar rückläufig zu sein, was auf Probleme bei der Erzeugung der Adversarial Examples hindeutet. Möglicherweise widersprechen sich die von den einzelnen Heads in das geteilte Netzwerk zurück propagierten Gradienten und führen so zu fehlerhaften Adversarial Examples. Aufgrund der beobachteten Probleme wurden nicht alle Klassifikatoren trainiert und auf die Verwendung von Adversarial Training für weitere Untersuchungen verzichtet.

Da mit Hilfe des Ensembles mit zehn Heads die besten Ergebnisse erzielt werden konnten, wurde diese Anzahl für den anschließenden inkrementellen Lernprozess ausgewählt[6]. Die Parameter für den inkrementell trainierten Klassifikator wurden in Anlehnung an die Referenz-Klassifikatoren definiert. Nach jeder Erweiterung der Trainingsdaten wurde der Klassifikator für 80 weitere Epochen trainiert, sodass für jeden komplett hinzugefügten Datensatzteil erneut eine Gesamtanzahl von 800 Epochen erreicht wurde. In jeder Iteration wurde erneut die beste Parameterkonfiguration bestimmt und die Gewichte vor der nächsten Erweiterung auf diese Konfiguration zurückgesetzt.

[6] Die Betrachtung aller Headanzahlen ist aus zeitlichen Gründen nicht möglich gewesen. Die Trainingszeit für die im Anschluss dargestellten Ergebnisse betrug unter Verwendung einer NVIDIA Titan X (Maxwell) bereits insgesamt $3 \cdot 10$ Tage (sequentiell alle drei Durchläufe).

Abbildung 6.2 stellt die Ergebnisse der Untersuchungen dar.

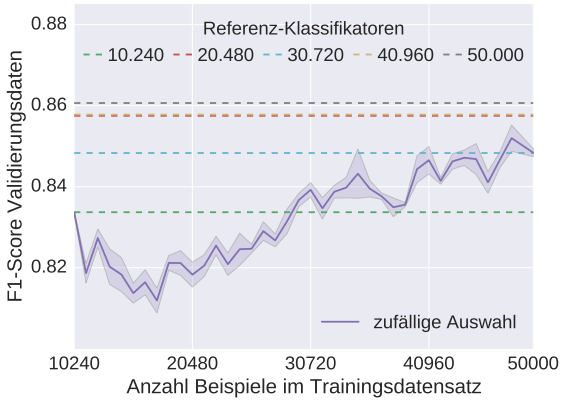


Abbildung 6.2: Ergebnisse Eignung Ensemble-Ansatz
Dargestellt sind die Ergebnisse (mittlerer F1-Score und einfache Standardabweichung) der experimentellen Untersuchungen zur Eignung des Ensemble-Ansatzes für einen inkrementellen Lernprozess.

Die in Abbildung 6.2 dargestellte Lernkurve zeigt, dass der angestrebte inkrementelle Lernprozess erfolgreich realisiert werden konnte. Die Leistung des Dropout-Ansatzes kann jedoch nicht erreicht werden. In allen drei Durchläufen existiert zudem ein starker Einbruch, welcher erst mit Erreichen von etwa 30.000 Beispielen überwunden werden kann. Während des Trainings wurde beobachtet, dass der Fehler auf den Trainingsdaten in diesem Bereich infolge neuer Beispiele besonders stark zunimmt, was auf verstärktes Auswendiglernen und schlechte Generalisierungseigenschaften hindeutet. Dieses Verhalten könnte eventuell mit anders gewählten Lernparametern verbessert werden.

6.2 Berücksichtigung der Unsicherheiten

In diesem Teil der experimentellen Untersuchungen wurden die zuvor betrachteten inkrementellen Lernprozesse um eine Berücksichtigung der extrahierten Unsicherheiten erweitert. Für beide Ansätze wurde hierzu zunächst untersucht, wie viele Dropout-Vorwärtsprozesse beziehungsweise wie viele Heads zur Extraktion von aussagekräf-

tigen Unsicherheitsinformationen benötigt werden. Anschließend wurden die in Abschnitt 3.2.3 (Seite 29) vorgestellten Maße zur Unsicherheitsmodellierung angewendet und hinsichtlich ihrer Eignung zur Realisierung von intelligenten inkrementellen Lernprozessen analysiert. Aufbauend auf den gewonnenen Erkenntnissen wurde abschließend ein für diese Masterarbeit geeignetes Gesamtsystem abgeleitet.
Im Folgenden werden die in State-of-the-Art-Betrachtungen vorgestellten Maße aus [Gal et al., 2016] noch einmal wiederholt und deren Kerngedanken zur Auswahl neuer Trainingsbeispiele zusammengefasst. Jedes der folgenden Maße kann auf Basis von mehreren Netzwerkausgaben berechnet werden.

Auswahl von Beispielen, welche die Entropie maximieren

Die Entropie bildet das Standardmaß zur Beurteilung von Sicherheit. Für unsichere Klassifikationen wird ein hohes Maß an Unordnung in der mittleren Klassifikatorausgabe und folglich eine hohe Entropie erwartet.

Auswahl von Beispielen, welche die Mutual Information zwischen Klassifikation und Verteilung über den Ausgaben maximieren

Beispiele, die dieses Maß maximieren, besitzen eine hohe Varianz in der Aktivierung der Softmax-Schicht. Der Klassifikator ist sich für diese Beispiele im Durchschnitt unsicher, es existieren zudem Ausgaben, die mit fälschlicherweise hoher Sicherheit zu unterschiedlichen Klassifikationsergebnissen führen.

Auswahl von Beispielen, welche die Variationsrate maximieren

Die Variationsrate quantifiziert die Abweichungen von dem am häufigsten vorkommenden Label (Modus). Für die betrachteten Zweiklassenprobleme liegt diese im Intervall $[0, 1/2]$.

Auswahl von Beispielen, welche die mittlere Varianz maximieren

Die Varianz ist ebenfalls ein Standardmaß zur Beschreibung von Unregelmäßigkeiten, welches zudem auch auf Regressionsprobleme angewendet werden kann. Soll als Bewertungskriterium nicht die Stärke sondern die Breite der Streuung um den Mittelwert angesetzt werden, kann an dieser Stelle ebenfalls die Standardabweichung verwendet werden.

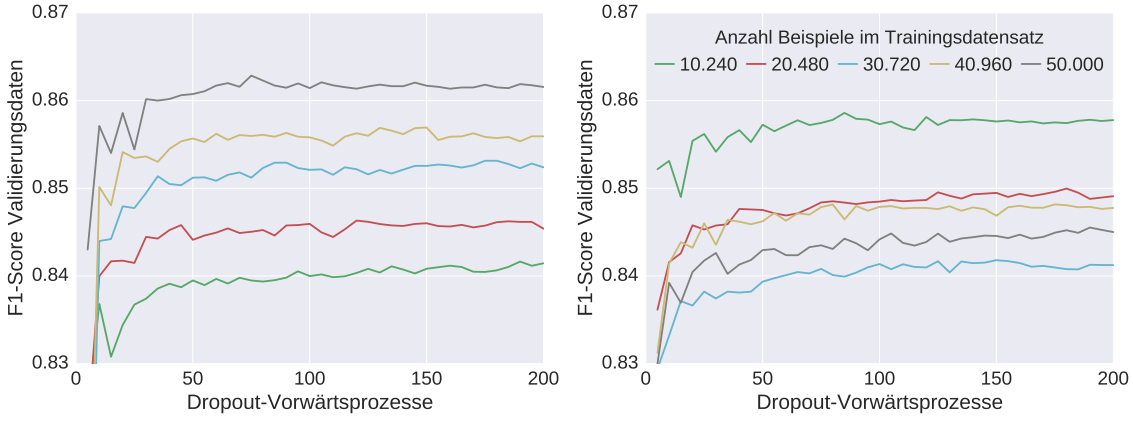
Entgegen der Untersuchungen in Abschnitt 6.1 (Seite 93) wurden die Trainingsdaten der inkrementell trainierten Klassifikatoren in zehn Iterationen lediglich bis zu einer Gesamtanzahl von 20.480 Beispielen erweitert. Neue Beispiele konnten zudem aus allen vier verbleibenden Teilen des GAPs-Datensatzes gezogen werden. Durch diese Modifikationen sollte sichergestellt werden, dass bei der Bestimmung neuer Trainingsbeispiele auf Basis der modellierten Unsicherheiten stets genügend Beispiele zur Verfügung stehen und nur möglichst informative Bildausschnitte gewählt werden. Analog zu den Untersuchungen in Abschnitt 6.1 (Seite 93) wurden die Lernprozesse dreimal wiederholt und der maximale F1-Score auf den Validierungsdaten für die Evaluierung genutzt.
Für den Dropout-Ansatz wurde zusätzlich erneut die Verwendung von Adversarial Training betrachtet. Hierbei sollte überprüft werden, ob das bisher beobachtete Verhalten durch die Berücksichtigung der Unsicherheiten bei der Auswahl neuer Trainingsbeispiele verbessert werden kann. Für den Ensemble-Ansatz entfällt eine vergleichbare Untersuchung, da die beobachteten Probleme in diesem Fall bereits beim Training der Referenz-Klassifikatoren aufgetreten sind und auf Unstimmigkeiten beim Erstellungsprozess der Adversarial Examples zurückgeführt werden können.

6.2.1 Dropout-Ansatz

Bei der Extraktion von Unsicherheitsinformationen mittels Dropout-Ansatz muss jede Eingabe mehrmals unter Verwendung verschiedener Dropout-Masken durch das Netzwerk propagiert werden. Da dieser Prozess sehr zeitintensiv ist, wurde zunächst untersucht, wie viele Vorwärtsprozesse durch das Netzwerk zur Extraktion von aussagekräftigen Unsicherheitsinformationen benötigt werden. Für die Bestimmung einer geeigneten Anzahl wurde der in Abschnitt 6.1.1 (Seite 95) betrachtete inkrementelle Lernprozess an den fünf Punkten, an denen komplette Datensatzteile erreicht wurden, probabilistisch evaluiert[7]. Hierzu wurden die mittleren Ausgaben für die Validierungsdaten auf Basis von bis zu 200 Vorwärtsprozessen bestimmt und mittels F1-Score bewertet. Mit Erreichen einer geeigneten Anzahl an Vorwärtsprozessen ist davon auszugehen, dass sich die bestimmten F1-Scores stabilisieren und sich nur noch geringfügig

[7] Es wurde stets nur der erste der dreimal wiederholten Lernprozesse probabilistisch evaluiert.

verändern. Abbildung 6.3 zeigt die Ergebnisse der Untersuchungen.

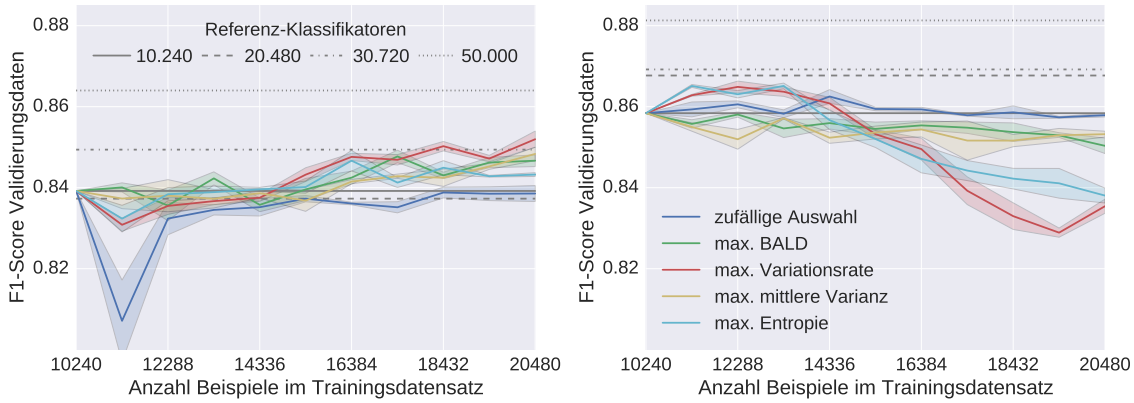


(a) Ohne Adversarial Training **(b)** Mit Adversarial Training

Abbildung 6.3: Ergebnisse Bestimmung der Anzahl der Vorwärtsprozesse
Dargestellt sind die Ergebnisse der Untersuchungen zur Bestimmung einer geeigneten Anzahl an Vorwärtsprozessen durch das Netzwerk[8].

Die in Abbildung 6.3 dargestellten Ergebnisse zeigen, dass sich der F1-Score sowohl ohne als auch mit Adversarial Training bei ungefähr 100 Vorwärtsprozessen stabilisiert. Dieses Ergebnis deckt sich mit der von den Autoren in [Gal, 2016] empfohlenen Anzahl an Iterationen. Im Folgenden werden daher zur Extraktion von Unsicherheitsinformationen stets 100 Vorwärtsprozesse durch das Netzwerk verwendet.
Nach der Festlegung einer geeigneten Anzahl, wurden die vorgestellten Maße zur Modellierung von Unsicherheiten auf die zuvor betrachteten inkrementellen Lernprozesse angewendet und neue Trainingsbeispiele ausschließlich auf Basis der bestimmten Unsicherheiten ausgewählt. Die in den Untersuchungen genutzten Lernparameter wurden analog zu denen in Abschnitt 6.1.1 (Seite 95) gewählt. Abbildung 6.4 stellt die erweiterten inkrementellen Lernprozesse dar und vergleicht die Leistung der einzelnen Maße mit der zuvor verwendeten zufälligen Auswahl.

[8] Die dargestellten Ergebnisse können nicht mit den inkrementellen Lernprozessen in Abbildung 6.1 verglichen werden, da diese ohne Dropout in der Anwendungsphase und durch Mittelung über drei Durchläufe erzeugt wurden. Der Verlauf der Kurven bestätigt jedoch, dass die in [Srivastava et al., 2014] (Hauptpublikation Dropout) für die Anwendungsphase vorgeschlagene Vorgehensweise einer Mittelung über eine Vielzahl von Vorwärtsprozessen durch das Netzwerk ähnelt und dass in dieser Masterarbeit für die Evaluierung auf mehrfache Vorwärtsprozesse verzichtet werden kann.



(a) Ohne Adversarial Training **(b)** Mit Adversarial Training

Abbildung 6.4: Ergebnisse Dropout-Ansatz

Dargestellt sind die Ergebnisse (mittlerer F1-Score und einfache Standardabweichung) der experimentellen Untersuchungen zur Eignung der verschiedenen Maße zur Verbesserung der betrachteten inkrementellen Lernprozesse.

Die in Abbildung 6.4a dargestellten Kurven zeigen, dass das Lernverhalten durch die Berücksichtigung der extrahierten Unsicherheiten deutlich verbessert werden kann. Bereits nach dem Hinzufügen von weiteren 10.240 Trainingsbeispielen wird die Leistung des mit 30.720 Beispielen trainierten Referenz-Klassifikators übertroffen. Ähnlich wie in [Gal et al., 2016] eignet sich die Variationsrate am besten zur Modellierung der Unsicherheiten.

Die Ergebnisse in Abbildung 6.4b verdeutlichen hingegen, dass bei Verwendung von Adversarial Training auch unter Berücksichtigung der Unsicherheiten kein inkrementeller Lernprozess realisiert werden kann. Bei einer Modellierung der Unsicherheiten mit Hilfe der Variationsrate und der Entropie nimmt die Klassifikationsleistung zwar zunächst zu, bricht jedoch anschließend, ähnlich wie in den vorherigen Untersuchungen, ein. Dieses Verhalten deutet weiterhin auf ein stark ausgeprägtes lokales Minimum hin, welches nicht überwunden werden kann und in dessen Folge keine aussagekräftigen Adversarial Examples erzeugt werden können. Im Rahmen dieser Masterarbeit wurde daher zusätzlich untersucht, ob durch eine Berücksichtigung der Adversarial Examples bei der Auswahl neuer Trainingsbeispiele ein besseres Lernverhalten realisiert werden kann. Die Ergebnisse dieser Untersuchung sind in Anhang A.2 (Seite 131) dargestellt.

Diese zeigen, dass das Lernverhalten zwar verbessert werden kann, die Probleme jedoch nicht vollständig überwunden werden können. Zudem führt die zusätzliche Berücksichtigung der Adversarial Examples zu einer weiteren deutlichen Verlängerung der Verarbeitungszeit. Aus diesem Grund wird von der Verwendung von Adversarial Training in weiteren Experimenten abgesehen.

6.2.2 Ensemble-Ansatz

Bei der Extraktion von Unsicherheitsinformationen mittels Ensemble-Ansatz muss zunächst die zu verwendende Headanzahl festgelegt werden. In Abschnitt 6.1.2 (Seite 97) wurde bereits festgestellt, dass die maximal mögliche Anzahl von zehn Heads zu den besten Klassifikationsergebnissen führt. Für die Ableitung einer ebenfalls zur Extraktion von Unsicherheitsinformationen geeigneten Headanzahl wurde in diesem Teil der experimentellen Untersuchungen analysiert, bei welcher Headanzahl sich die Klassifikationsergebnisse stabilisieren. Hierzu wurde der in Abschnitt 6.1.2 (Seite 97) betrachtete inkrementelle Lernprozess ebenfalls an den fünf Punkten, an denen komplette Datensatzteile erreicht wurden, evaluiert. Mit Erreichen einer geeigneten Headanzahl sollten sich auch in diesem Fall die bestimmten F1-Scores stabilisieren und nur noch geringfügig verändern. Abbildung 6.5 zeigt die Ergebnisse der Untersuchungen.

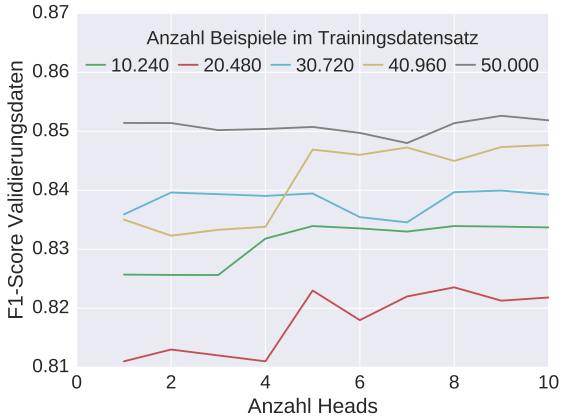


Abbildung 6.5: Ergebnisse Bestimmung der Headanzahl

Dargestellt sind die Ergebnisse der Untersuchungen zur Bestimmung einer geeigneten Headanzahl.

Die in Abbildung 6.5 dargestellten Evaluationsergebnisse zeigen, dass sich die F1-Scores bis zu einer Verwendung von fünf Heads sprunghaft ändern können und sich erst danach langsam stabilisieren. Da durch die genutzte Trainingsprozedur nicht ausgeschlossen werden kann, dass einzelne Heads schlechter als andere sind, existieren auch für größere Headanzahlen noch kleinere Schwankungen. Der Verlauf der Kurven zeigt jedoch, dass diese immer kleiner werden. Aus diesem Grund wurden für die anschließenden Experimente zu den Maßen zur Modellierung von Unsicherheiten erneut zehn Heads verwendet. Abbildung 6.6 stellt die erweiterten inkrementellen Lernprozesse dar und vergleicht die Leistung der einzelnen Maße mit der zuvor verwendeten zufälligen Auswahl neuer Trainingsbeispiele.

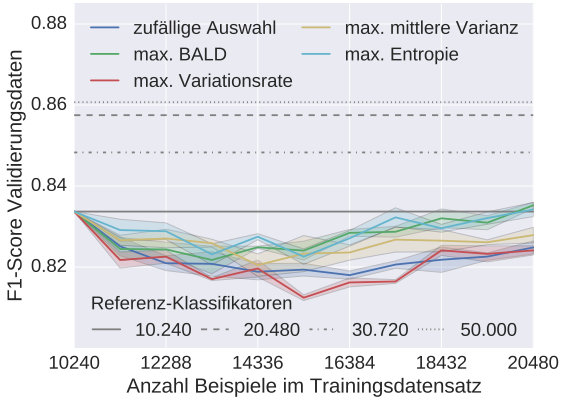


Abbildung 6.6: Ergebnisse Ensemble-Ansatz

Dargestellt sind die Ergebnisse (mittlerer F1-Score und einfache Standardabweichung) der experimentellen Untersuchungen zur Eignung der verschiedenen Maße zur Verbesserung der betrachteten inkrementellen Lernprozesse.

Die in Abbildung 6.5 dargestellten Lernprozesse zeigen, dass sich auch unter Berücksichtigung der Unsicherheiten ein zu Abbildung 6.2 (Seite 100) vergleichbares Lernverhalten einstellt. Der zuvor beobachtete Einbruch in den Klassifikationsergebnissen kann jedoch durch die Berücksichtigung der modellierten Unsicherheiten (mit Ausnahme der Variationsrate) gedämpft werden, sodass sich der Lernprozess deutlich schneller erholt. Es ist daher davon auszugehen, dass auch die mittels Ensemble-Ansatz extrahierten Unsicherheiten aussagekräftig sind und dass sich das zusätzliche Einbringen in den Entscheidungsprozess positiv auf den Lernprozess auswirkt.

Die auf dem GAPs-Datensatz durchgeführten experimentellen Untersuchungen haben gezeigt, dass sich beide Erweiterungsansätze (Dropout- und Ensemble-Ansatz) zur Realisierung von inkrementellen Lernprozessen eignen. Die mit Hilfe des Ensemble-Ansatzes erweiterten Basis-Klassifikatoren sind jedoch deutlich schwerer zu trainieren und erreichen nicht die Leistung des Dropout-Ansatzes[9]. Die zusätzliche Verwendung von Adversarial Training führt zwar zum Teil zu besseren Klassifikationsergebnissen, macht die Realisierung eines inkrementellen Lernprozesses jedoch unmöglich und wird daher nicht weiter verfolgt. Die zuletzt durchgeführten Experimente haben weiterhin gezeigt, dass mit Hilfe beider Ansätze aussagekräftige Unsicherheitsinformationen extrahiert werden können, welche durch die Modellierung mit den vorgestellten Maßen zusätzlich in den Entscheidungsprozess eingebracht werden und so den inkrementellen Lernprozess verbessern können.

Das beste inkrementelle Lernverhalten konnte mit Hilfe des Dropout-Ansatzes in Verbindung mit einer Unsicherheitsmodellierung mittels Variationsrate erzielt werden. Daher wird diese Konfiguration des Gesamtsystems für die anschließenden experimentellen Untersuchungen auf den verfügbaren Fahrbahnbildern genutzt. Weiterhin werden die in Abschnitt 6.1.1 (Seite 95) festgelegten Lernparameter übernommen.

6.3 Anwendung auf Fahrbahnbilder

Nachdem ein geeignetes Gesamtsystem abgeleitet werden konnte, wurde dieses auf die in dieser Masterarbeit verfügbaren Fahrbahnbilder angewendet. Da die Berücksichtigung aller 1.637 Bilder den zeitlichen Rahmen der Experimente übersteigen würde, wurden vorab mehrere Maßnahmen getroffen, um die Anzahl der Bilder einzuschränken. So wurden für die betrachteten inkrementellen Lernprozesse ausschließlich die Fahrbahnbilder verwendet, in denen sich laut Labelinformationen schadhafte Stellen (Klasse 1) befinden. Aufgrund der in Abschnitt 4.3.2 (Seite 45) beschriebenen

[9] Mit den in dieser Masterarbeit für den Ensemble-Ansatz gewählten Lernparametern konnte die Leistung des Dropout-Ansatzes nicht erreicht werden. Es ist jedoch nicht ausgeschlossen, dass andere Parameter zu besseren Lernprozessen geführt hätten. Durch den begrenzten zeitlichen Rahmen der Masterarbeit konnten jedoch keine ausgiebigen Parametertests durchgeführt werden.

Probleme bei der Labelvergabe für Flickstellen, wurden zudem alle 27 Bilder ausgeschlossen, welche aufgelegte oder eingelegte Flickstellen beinhalten. Durch diese Einschränkungen konnte die Anzahl auf 499 Fahrbahnbilder reduziert werden, wobei die in den verbleibenden Bildern als Schaden markierte Gesamtfläche dennoch lediglich nur einen Anteil von 17,45% ausmacht. Somit existieren trotz Einschränkung ausreichend viele Bereiche für intakte Straßenausschnitte (Klasse 0). In den betrachteten inkrementellen Lernprozessen wurden in jeder Iteration zwölf zufällig ausgewählte Fahrbahnbilder[10] betrachtet und aus diesen, analog zu den Untersuchungen in den vorherigen Abschnitten, insgesamt 1.024 neue Beispiele extrahiert. Der Datensatz wurde, ausgehend von dem für das initiale Training verwendeten ersten Teil des GAPs-Trainingsdatensatzes (siehe Tabelle 6.1, Seite 94), über 40 Iterationen bis zu einer Gesamtanzahl von 51.200 Beispielen erweitert. Die in den Untersuchungen verwendeten Lernparameter wurden analog zu denen in Abschnitt 6.1.1 gewählt.

Da bekannt ist, dass die für die Fahrbahnbilder verfügbaren Labelinformationen nur sehr grob sind, wurden die gleichen Experimente ebenfalls auf den 1.418 Basisbildern des GAPs-Trainingsdatensatzes (siehe Abschnitt 4.2, Seite 40) durchgeführt. Ziel dieser zusätzlichen Untersuchungen ist es, den Einfluss der Labelinformationen auf die Lernprozesse besser untersuchen zu können. Weiterhin kann die Auswirkung von unterschiedlich balancierten Daten zielführender analysiert werden. Da die trainierten Klassifikatoren durch die in Abschnitt 5.4.1.1 (Seite 85) beschriebene Modifikation auf beliebige Bildgrößen angewendet werden können, musste für die Anwendung auf die GAPs-Basisbilder lediglich der Extraktionsprozess für neue Patches modifiziert werden. So musste die Vorgehensweise zur Erstellung der Masken für verbotene Patchpositionen (für Fahrbahnbilder siehe Abbildung 5.7, Seite 89) an die sich unterscheidenden Labelinformationen angepasst werden. Neben gekennzeichneten Straßenmarkierungen und anderen zu ignorierenden Bereichen wurden dabei ebenfalls die bereits für das initiale Training verwendeten Patches in die Masken aufgenommen. Weiterhin musste die Labelvergabe für Patches der Schadklassse (Klasse 1) an die genaueren Labelinformationen angepasst werden. Neue Bildausschnitte wurden als Klasse 1 gelabelt,

[10] Die Auswahl erfolgte durch Ziehen ohne Zurücklegen.

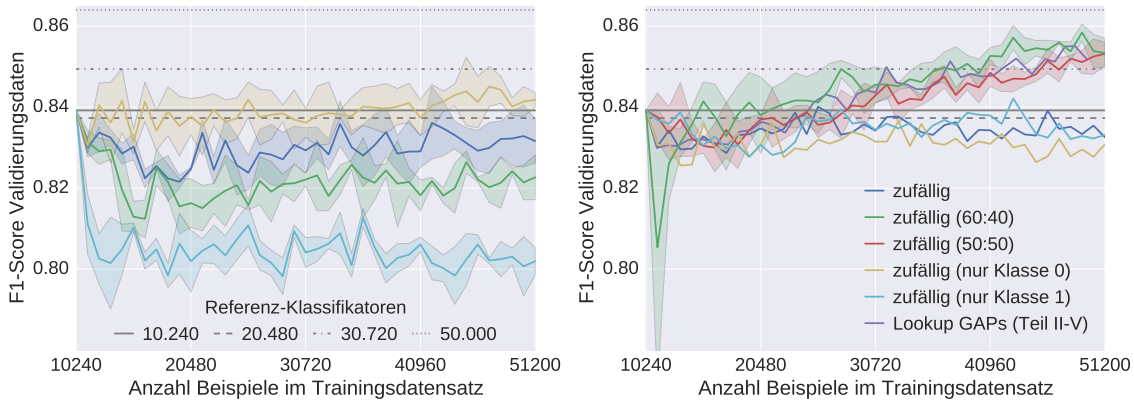
wenn sich die zentralen 32x32 Pixel des Patches in einem markierten Schaden befunden haben. Während auf diese Weise ein guter Kompromiss zwischen Zentrierung der Schadstelle im Patch[11] und zusätzlich als Klasse 1 gelabelten Bereichen gefunden werden konnte, führt diese Vorgehensweise auch dazu, dass 10,82% der Markierungen nicht für die Extraktion von Patches herangezogen werden können. Da diese jedoch nur einen Anteil von 3,5% der Gesamtschadfläche ausmachen, können diese Markierungen vernachlässigt werden. Da die Basisbilder des GAPs-Datensatzes deutlich kleiner sind, wurde die Anzahl der in jeder Iteration betrachteten Bilder auf 35 erhöht. Alle anderen Parameter wurden wie in den Experimenten zu den Fahrbahnbildern gewählt.
In den nachfolgenden Abschnitten werden für beide Experimentgruppen verschiedene Szenarien zur Erweiterung der Trainingsdaten untersucht. Die Leistung der einzelnen Lernprozesse wird dabei zunächst, analog zu den vorherigen Abschnitten, auf den Validierungsdaten evaluiert. Da bekannt ist, dass die Basisbilder des GAPs-Datensatzes möglicherweise ähnlicher zu den Validierungsdaten sind, werden am Ende der Untersuchungen zusätzlich Ergebnisse auf dem Test-Validierungsdatensatz präsentiert.

6.3.1 Zufällige Auswahl neuer Trainingsbeispiele

Analog zu den Experimenten in Abschnitt 6.1 (Seite 93) wurde zunächst das Hinzufügen von zufällig gewählten Trainingsbeispielen untersucht. Die Ergebnisse dieser Untersuchungen bilden die Referenz für die im Anschluss betrachteten inkrementellen Lernprozesse unter Berücksichtigung der extrahierten Unsicherheiten. Neben einer zufälligen Auswahl unabhängig von den zugehörigen Klasseninformationen wurde ebenfalls untersucht, wie sich das ausschließliche Hinzufügen von Patches einer Klasse auf die Lernprozesse auswirkt. Weiterhin wurde analysiert, ob das Lernverhalten durch eine Erweiterung mit Beispielen in gleichen Anteilen wie im initialen Trainingsdatensatz verbessert werden kann. Der für das initiale Training verwendete erste Teil des GAPs-Trainingsdatensatzes besitzt durch die zufällige Zuordnung der Beispiele zu den einzelnen Teildatensätzen eine zum kompletten GAPs-Trainingsdatensatz ähnliche Klassenverteilung von ungefähr 60:40 (siehe Tabelle 6.1, Seite 94). In dem zuletzt

[11] 66,82% der Markierungen für Schäden sind kleiner als 64 Pixel.

genannten Szenario wurden daher zufällige Patches in genau diesem Klassenverhältnis hinzugefügt. Abbildung 6.7 stellt die auf den Validierungsdaten erzielten mittleren F1-Scores dar. Jeder Lernprozess wurde hierbei dreimal wiederholt, wobei für die einzelnen Lernprozesse in den unterschiedlichen Szenarien stets die gleiche zufällige Reihenfolge der Ausgangsbilder verwendet wurde.



(a) Fahrbahnbilder **(b)** GAPs-Basisbilder

Abbildung 6.7: Ergebnisse zufällige Auswahl

Dargestellt sind die Ergebnisse (mittlerer F1-Score und einfache Standardabweichung für ausgewählte Kurven) der experimentellen Untersuchungen zur zufälligen Auswahl von neuen Trainingsbeispielen.

Die in Abbildung 6.7a präsentierten Ergebnisse zeigen, dass die Klassifikationsleistung nur minimal durch zufällig gewählte Patches von intakten Straßenausschnitten (gelbe Kurve) verbessert werden kann. Alle anderen Szenarien wirken sich durchgehend negativ auf die Leistung aus. Hierbei ist zudem eine stufenweise Verschlechterung in Abhängigkeit der Anzahl der für Klasse 1 hinzugefügten Patches zu beobachten. Während die zufällige Auswahl unabhängig von den Klasseninformationen (dunkelblaue Kurve), bei der durchschnittlich lediglich 17,5% der Patches auf Klasse 1 entfallen, noch zu den Ergebnissen führt, welche nur geringfügig unter der Ausgangsleistung liegen, resultieren alle anderen Szenarien, bei denen mehr Beispiele für Klasse 1 hinzugefügt werden, in zunehmend schlechteren Ergebnissen. Diese Beobachtung legt nahe, dass die für Klasse 1 hinzufügten Bildausschnitte infolge der groben Labelinformationen überwiegend widersprüchlich sind. Die dunkelblaue Kurve zeigt jedoch auch, dass

fehlerhafte Daten bis zu einem bestimmten Grad toleriert werden können, ohne dass die Klassifikationsergebnisse deutlich einbrechen.

Die in Abbildung 6.7b dargestellten Ergebnisse für die GAPs-Basisbilder verdeutlichen, dass die Klassifikationsleistung nur dann verbessert werden kann, wenn Beispiele für beide Klassen hinzugefügt werden[12]. Eine Erweiterung in genau dem gleichen Verhältnis wie im initial verwendeten Trainingsdatensatz (grüne Kurve) führt zu den besten Ergebnissen. In den Untersuchungen in Abschnitt 6.1 (Seite 93) ist diese Voraussetzung stets durch die Verwendung des GAPs-Datensatzes gegeben gewesen. Der zusätzlich dargestellte Vergleich mit der Auswahl von Trainingsbeispielen, welche exakt den Patches aus den GAPs-Trainingsdatensatzteilen II-V entsprechen (violette Kurve), bestätigt diese Bobachtungen und zeigt, dass mit dem in dieser Masterarbeit realisierten Gesamtsystem zu Abschnittt 6.1.1 (Seite 95) vergleichbare Ergebnisse erzielt werden können[13]. Eine Erklärung, warum der Lernprozess in den Experimenten in Abschnitt 6.2 (Seite 100) durch die Berücksichtigung der Unsicherheiten verbessert werden konnte, obwohl das spezielle Klassenverhältnis von 60:40 nicht berücksichtigt wurde, wird in den anschließenden Untersuchungen gegeben.

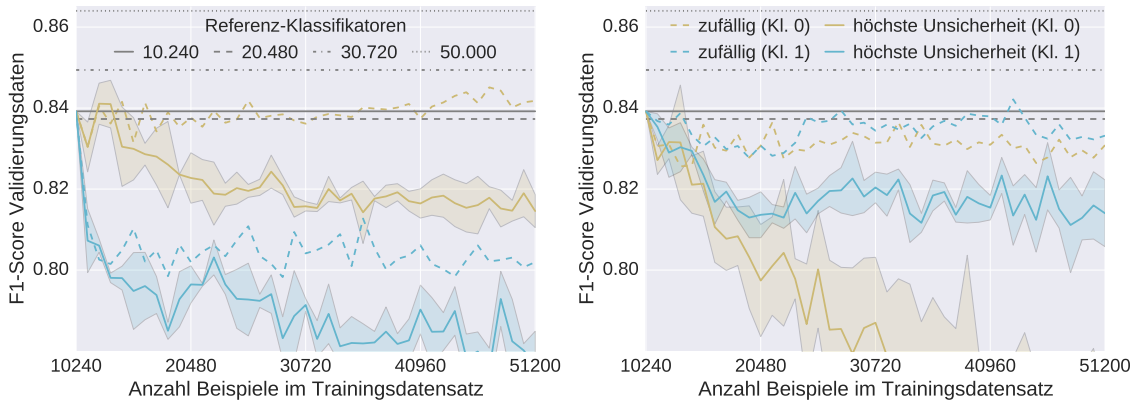
6.3.2 Berücksichtigung der Unsicherheiten

Im Anschluss wurden, ähnlich wie in Abschnitt 6.2 (Seite 100), die extrahierten Unsicherheiten bei der Auswahl neuer Trainingsbeispiele berücksichtigt. Hierbei wurde zunächst untersucht, wie sich das klassenweise Hinzufügen von Beispielen mit höchster Unsicherheit auf die inkrementellen Lernprozesse auswirkt. Diese Untersuchungen ermöglichen zudem die Bewertung der anhand der Unsicherheiten ausgewählten Bildausschnitte und somit das Ableiten einer Aussage zur Qualität der extrahierten Unsi-

[12] Eine zufällige Auswahl unabhängig von den Klasseninformationen (dunkelblaue Kurve in Abbildung 6.7b) entspricht in diesem Fall dem Hinzufügen von ungefähr 2% Patches für Klasse 1.

[13] Die Ergebnisse können nicht direkt verglichen werden, da in den Untersuchungen in Abschnitt 6.1.1 (Seite 95) stets aus allen Patches der GAPs-Trainingsdatenteile II-V ausgewählt werden konnte und die verwendeten Datensatzteile zudem zufällige Ausschnitte aus allen GAPs-Basisbildern beinhalten. Es ist daher davon auszugehen, dass diese Daten in jeder Iteration ein größeres Generalisierungspotenzial bieten als die in diesen Experimenten jeweils gleichzeitig betrachteten 35 GAPs-Basisbilder.

cherheiten. Anschließend wurden verschiedene Szenarien betrachtet, in denen Beispiele für beide Klassen hinzugefügt wurden. Abbildung 6.8 veranschaulicht die Ergebnisse der klassenweisen Untersuchungen und vergleicht die Klassifikationsleistung mit der zufälligen Auswahl aus Abbildung 6.7.



(a) Fahrbahnbilder **(b)** GAPs-Basisbilder

Abbildung 6.8: Ergebnisse klassenweise Erweiterung höchste Unsicherheit
Dargestellt sind die Ergebnisse (mittlerer F1-Score und einfache Standardabweichung für ausgewählte Kurven) der experimentellen Untersuchungen zur klassenweisen Erweiterung der Trainingsdaten unter Berücksichtigung der Unsicherheiten.

Die in Abbildung 6.8 dargestellten Lernprozesse zeigen, dass sich die Berücksichtigung der Unsicherheiten sowohl für die Fahrbahnbilder als auch für die GAPs-Basisbilder deutlich negativ auswirkt. Da sich dieses Verhalten nicht mit den bisherigen Erkenntnissen aus Abschnitt 6.2.1 (Seite 102) deckt, wurden die jeweiligen zur Erweiterung der Trainingsdaten ausgewählten Patches genauer analysiert. In Anhang A.3 (Seite 134) werden mehrere in den Lernprozessen betrachtete Ausgangsbilder und die aus diesen extrahierten Bildausschnitte veranschaulicht. Für die GAPs-Basisbilder wird dabei deutlich, dass den Trainingsdaten für Klasse 0 überwiegend problematische Patches hinzugefügt werden (siehe Abbildung A.5, Seite 135). Die hohen Unsicherheiten sind dabei sehr oft widersprüchlich gelabelten Ausschnitten geschuldet. So konnte festgestellt werden, dass die GAPs-Basisbilder verschlossene Arbeitsnähte und ausgebesserte Schadstellen enthalten, welche in einigen Fällen als Schaden und in anderen Fällen als

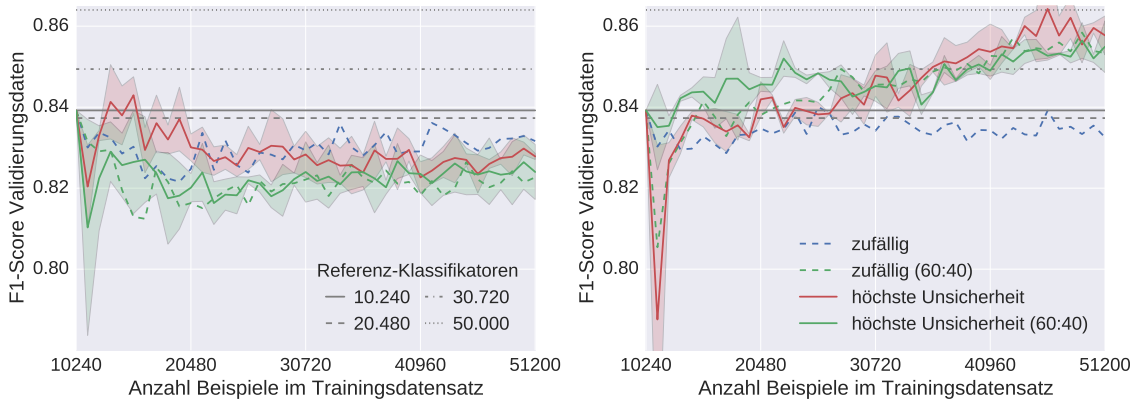
intakte Straßenausschnitte gelabelt sind. Weiterhin werden kleinste, kaum erkennbare Risse oder Strukturen, welche selbst unter Zuhilfenahme der angrenzenden Bildbereiche nur sehr schwer von den für das Training verwendeten Ausschnitten der Schadklasse zu unterscheiden sind, zu den Trainingsdaten für Klasse 0 hinzugefügt. In der Folge enthält der Trainingsdatensatz zunehmend widersprüchlichere Daten und die Klassifikationsergebnisse (True Positives) brechen ein. Da für den GAPs-Trainingsdatensatz bewusst Straßenabschnitte in schlechtem Zustand gewählt wurden, sind die Auswirkungen auf den GAPs-Basisbildern deutlich ausgeprägter als auf den Fahrbahnbildern. In den inkrementellen Lernprozessen auf den Fahrbahnbildern stabilisieren sich die Ergebnisse, weil neben diesen problematischen Bereichen merklich mehr korrekte Patches für Klasse 0 ausgewählt werden (siehe Abbildung A.4, Seite 134).

Bei der ausschließlichen Erweiterung mit Beispielen der Klasse 1 zeigt sich ein ähnliches Bild (siehe Abbildung A.6 und A.7, Seite 136). In diesem Fall werden die problematischen Bereiche jedoch überwiegend als Beispiele für die Schadklasse hinzugefügt. Zudem wurde festgestellt, dass die Unsicherheiten vor allem in den unmittelbar an die Schadstellen angrenzenden Bildbereichen sehr groß sind. In der Folge werden viele Patches hinzugefügt, welche nur sehr kleine Teile oder gar keinen Ausschnitt der schadhaften Stellen enthalten. Daher führen auch diese inkrementellen Lernprozesse zu schlechteren Ergebnissen als bei einer zufälligen Auswahl von neuen Trainingsbeispielen (False Positives steigen). Für die Fahrbahnbilder wirken sich die vielen aufgrund der groben Labelinformationen falsch gelabelten Bildausschnitte zusätzlich negativ aus.

Die Untersuchungen zeigen, dass die mit Hilfe des Dropout-Ansatzes bestimmte Variationsrate wirklich existierende Unsicherheiten beschreibt. Es wird jedoch auch deutlich, dass der Einfluss von eventuellen Labelungenauigkeiten und schwer zu unterscheidenden Bildausschnitten durch eine Auswahl unter Berücksichtigung der Unsicherheiten drastisch verstärkt wird. Im Anschluss wurde daher untersucht, ob das Lernverhalten durch das Hinzufügen von Beispielen für beide Klassen zielführender beeinflusst werden kann.

Ähnlich wie in Abschnitt 6.2.1 (Seite 102) wurde zunächst getestet, wie sich das Hinzufügen von Beispielen mit höchster Unsicherheit unabhängig von der Klassenzugehörigkeit auswirkt. Aufbauend auf den Erkenntnissen aus Abschnitt 6.3.1 (Seite 109)

wurde anschließend ebenfalls ein festes Klassenverhältnis von 60:40 verwendet. Abbildung 6.9 veranschaulicht die Ergebnisse der Untersuchungen und vergleicht diese mit der zu Beginn durchgeführten zufälligen Auswahl neuer Trainingsbeispiele. Ergänzende Visualisierungen zu den ausgewählten Patches sind erneut in Anhang A.3 (Seite 138) dargestellt.



(a) Fahrbahnbilder **(b)** GAPs-Basisbilder

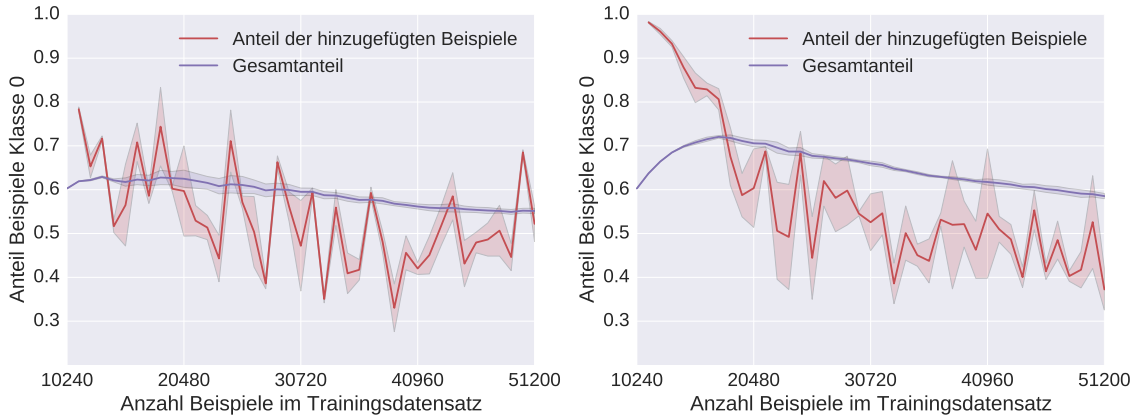
Abbildung 6.9: Ergebnisse Erweiterung höchste Unsicherheit
Dargestellt sind die Ergebnisse (mittlerer F1-Score und einfache Standardabweichung für ausgewählte Kurven) der experimentellen Untersuchungen zur Erweiterung der Trainingsdaten unter Berücksichtigung der Unsicherheiten.

Abbildung 6.9a zeigt, dass sich für die Fahrbahnbilder ein zur zufälligen Auswahl vergleichbares Lernverhalten einstellt und dass die Klassifikationsleistung auf den Validierungsdaten auch unter Berücksichtigung der Unsicherheiten nicht verbessert werden kann. Die im Anhang dargestellten Visualisierungen (siehe Abbildung A.8, Seite 138) zeigen zwar, dass durch das Hinzufügen von Beispielen beider Klassen generell plausiblere Patches ausgewählt werden, es wird jedoch auch deutlich, dass nach wie vor viele Bildausschnitte mit falschen Labelinformationen hinzugefügt werden. Aufgrund der zu ungenauen Schadensmarkierungen betrifft dies vor allem neue Beispiele für die Schadklasse. Bei der Analyse der verarbeiteten Bilder wurde aber auch festgestellt, dass in den verfügbaren Fahrbahnbildern viele nicht markierte Schäden existieren, welche den Trainingsdaten besonders häufig in späteren Iterationen hinzugefügt werden. Zusammen mit der minimal besseren Auswahl für Patches der Schadklasse ergibt sich so

ein Lernverhalten (True Positives nehmen leicht ab, False Positives steigen etwas an), welches sich von einer zufälligen Erweiterung nicht groß unterscheidet.

Für die in Abbildung 6.9b dargestellten Lernprozesse auf den GAPs-Basisbildern stellt sich ein zu Abschnitt 6.2.1 (Seite 102) vergleichbares Lernverhalten ein[14]. Die Lernkurven zeigen, dass sich sowohl die Berücksichtigung der Unsicherheiten als auch das Hinzufügen im Verhältnis von 60:40 positiv auf die Lernprozesse auswirkt. Die in Anhang A.3 (Seite 139) dargestellten Visualisierungen verdeutlichen zudem, dass weitaus weniger widersprüchliche Patches ausgewählt werden. Speziell die zuvor beobachteten Probleme an Arbeitsnähten und ausgebesserten Schadstellen bleiben aus. So werden für Klasse 0 ausschließlich Patches hinzugefügt, welche unmittelbar an die Schadstellen angrenzen. Die eigentlichen Schäden werden deutlich zentrierter den Trainingsdaten für Klasse 1 hinzugefügt. In der Folge kann ein erfolgreicher inkrementeller Lernprozess realisiert und die Klassifikationsleistung schrittweise verbessert werden. Die Verbesserungen werden dabei hauptsächlich durch das Erschließen neuer True Positives erreicht. Die Anzahl der False Positives bleibt währenddessen nahezu unverändert. Die besten Ergebnisse werden jedoch entgegen der Erwartungen nicht unter festgesetztem Klassenverhältnis (grüne Kurve in Abbildung 6.9b) sondern mit einer Auswahl unabhängig von den Klassenzugehörigkeiten (rote Kurve in Abbildung 6.9b) erzielt. Die dargestellten mittleren F1-Scores und die zugehörigen Standardabweichungen zeigen zudem, dass zu Beginn stets der Lernprozess mit festgesetztem Klassenverhältnis und nach Erreichen einer Datensatzgröße von ungefähr 30.720 Beispielen (20 Iterationen) der Lernprozess mit freier Klassenverteilung dominiert. Diese Beobachtung wirft die Frage auf, wie sich die alleinige Berücksichtigung der Unsicherheiten auf das Klassenverhältnis im Datensatz auswirkt. Abbildung 6.10 zeigt sowohl den Anteil der für Klasse 0 in jeder Iteration hinzugefügten Beispiele als auch den sich im Datensatz ergebenden Gesamtanteil.

[14] Die Ergebnisse können nicht direkt verglichen werden, da in den Untersuchungen in Abschnitt 6.2.1 (Seite 102) stets aus allen verbleibenden Patches ausgewählt werden konnte und die verwendeten GAPs-Trainingsdatensatzteile II-V zudem zufällige Ausschnitte aus allen GAPs-Basisbildern enthalten. Es ist daher davon auszugehen, dass diese Daten in jeder Iteration ein größeres Generalisierungspotenzial bieten als die in diesen Experimenten jeweils gleichzeitig betrachteten 35 GAPs-Basisbilder.



(a) Fahrbahnbilder **(b)** GAPs-Basisbilder

Abbildung 6.10: Klassenverteilung Erweiterung höchste Unsicherheit
Dargestellt sind der in jeder Iteration hinzugefügte Anteil an Beispielen für Klasse 0 und der sich ergebende Gesamtanteil (jeweils Mittelwert und einfache Standardabweichung) für die in Abbildung 6.9 rot dargestellten Lernprozesse.

Die dargestellten Ergebnisse zeigen, dass bei einer Berücksichtigung der Unsicherheiten unabhängig von der Klassenverteilung in den verfügbaren Daten[15] automatisch ein in etwa ausgeglichenes Klassenverhältnis angestrebt wird (siehe Tendenz violette Kurven). Für Abbildung 6.10b sind zudem Beziehungen zum Lernprozess in Abbildung 6.9b zu erkennen. So werden zu Beginn überwiegend Beispiele für Klasse 0 hinzugefügt, was zu einem schlechteren Lernverhalten führt. Der weitere Verlauf zeigt jedoch, dass das optimale Klassenverhältnis automatisch gefunden wird und dass für einen möglichst erfolgreichen inkrementellen Lernprozess leicht schwankende Anteile erforderlich sind. Abbildung 6.10a verdeutlicht, dass für die Experimente auf den Fahrbahnbildern ähnliches gilt. Auffällig ist an dieser Stelle jedoch, dass der Anteil der für Klasse 0 hinzugefügten Beispiele, trotz Mittelung über drei Durchläufe, von Iteration zu Iteration stark schwankend ist. Die zugehörigen Standardabweichungen zeigen zudem, dass dies trotz unterschiedlicher Reihenfolge der Ausgangsbilder für alle Durchläufe gilt. Dieses Verhalten deutet darauf hin, dass überwiegend widersprüchliche Daten hinzugefügt werden, welche dazu führen, dass die Unsicherheiten für die jeweils andere Klasse stark zunehmen.

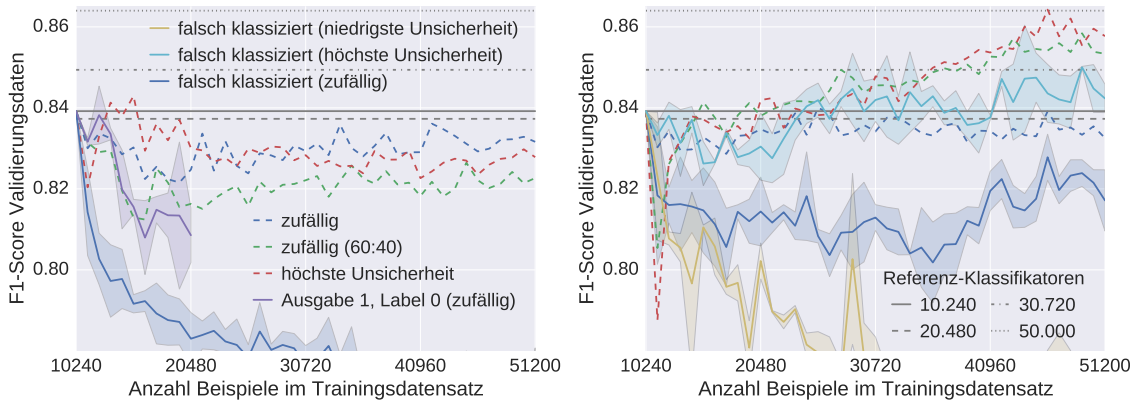
[15] Die Schadklasse ist in beiden Experimentgruppen flächenmäßig unterrepräsentiert.

Die Untersuchungen in diesem Abschnitt haben gezeigt, dass sich die Berücksichtigung der extrahierten Unsicherheiten positiv auf die Lernprozesse auswirkt, sofern den Trainingsdaten neue Beispiele für beide Klassen hinzugefügt werden. Das Lernverhalten kann jedoch nur verbessert werden, wenn, wie im Fall der GAPs-Basisbilder, entsprechend genaue Labelinformationen zur Verfügung stehen und aussagekräftige Beispiele ausgewählt werden können. Für die Fahrbahnbilder hat sich gezeigt, dass auch unter Zuhilfenahme der Unsicherheiten nur in den wenigsten Fällen korrekt gelabelte Bildausschnitte hinzugefügt werden und dass infolge dieser widersprüchlichen Beispiele die Klassifikationsleistung letztlich nicht verbessert werden kann.

6.3.3 Zusätzliche Berücksichtigung der Klassifikatorausgabe

Abschließend wurde untersucht, ob das Lernverhalten zusätzlich verbessert werden kann, wenn die Trainingsdaten ausschließlich mit Patches erweitert werden, welche vom Netzwerk falsch klassifiziert wurden. Da zuvor festgestellt wurde, dass stets Beispiele für beide Klassen hinzugefügt werden müssen, wurden an dieser Stelle in jeder Iteration mindestens 40% Patches für die Schadklasse ergänzt. Neben einem an klassisches Bootstrapping angelehnten Szenario, bei dem die Trainingsdaten mit zufällig gewählten, falsch klassifizierten Beispielen erweitert wurden, wurden ebenfalls zwei Varianten unter Berücksichtigung der Unsicherheiten getestet. So wurden in zwei zusätzlichen Szenarien jeweils nur Beispiele mit den höchsten und den niedrigsten Unsicherheitswerten hinzugefügt. Während eine Auswahl auf Basis der höchsten Unsicherheit die Experimente in Abschnitt 6.3.2 erweitert, kann ein Hinzufügen von sicheren, aber dennoch falsch klassifizierten Patches als ein Lernen anhand von schwerwiegenden Fehlern interpretiert werden. Da diese Szenarien den Einfluss von zu groben oder fehlerhaften Labelinformationen erheblich verstärken, wurde für die Fahrbahnbilder zuvor zusätzlich untersucht, wie sich das ausschließliche Erweitern mit Beispielen auswirkt, welche als Klasse 1 klassifiziert wurden, obwohl sie laut Labelinformationen keinen Schaden zeigen. Hierbei hat sich herausgestellt, dass bereits bei einer zufälligen Auswahl nach zehn Iterationen keine Bildausschnitte mehr extrahiert werden können, welche diesem

Auswahlkriterium entsprechen[16]. Die in Anhang A.4 (Seite 140) dargestellten Visualisierungen (Abbildung A.10) zeigen, dass neben echten Fehlklassifikationen ebenfalls viele Bildausschnitte gewählt werden, welche nicht markierte Schäden zeigen. Während der Normalzustand in den Fahrbahnbildern durch dieses Auswahlszenario sehr schnell gelernt werden kann, führen die fehlerhaften Patches aber dazu, dass insgesamt auch deutlich weniger Schäden erkannt werden können (True Positives sinken deutlich). In der Folge brechen die Klassifikationsergebnisse bereits bei diesen eingeschränkten Untersuchungen ein. Die zu Beginn beschriebenen Szenarien wurden daher nur in einem eingeschränkten Umfang für die Fahrbahnbilder durchgeführt[17]. Abbildung 6.11 zeigt die Ergebnisse aller Untersuchungen.



(a) Fahrbahnbilder **(b)** GAPs-Basisbilder

Abbildung 6.11: Ergebnisse Erweiterung mit falsch klassifizierten Beispielen
Dargestellt sind die Ergebnisse (mittlerer F1-Score und einfache Standardabweichung für ausgewählte Kurven) der experimentellen Untersuchungen zur Erweiterung der Trainingsdaten mit laut Labelinformationen falsch klassifizierten Beispielen.

Die Ergebnisse für die Fahrbahnbilder in Abbildung 6.11a veranschaulichen sowohl den zuvor beschriebenen Lernprozess, bei dem lediglich falsch klassifizierte Beispiele zu Klasse 0 hinzugefügt wurden (violette Kurve in Abbildung 6.11a), als auch, dass

[16] Konnten in einer Iteration nicht genügend neue Beispiele extrahiert werden, wurden die Ausgangsbilder der jeweils nächsten Iterationen zusätzlich mit in den Auswahlprozess einbezogen.

[17] Es ist davon auszugehen, dass bei einer Berücksichtigung der laut Labelinformationen nicht korrekt als Klasse 1 klassifizierten Patches noch mehr falsch gelabelte Daten hinzugefügt werden.

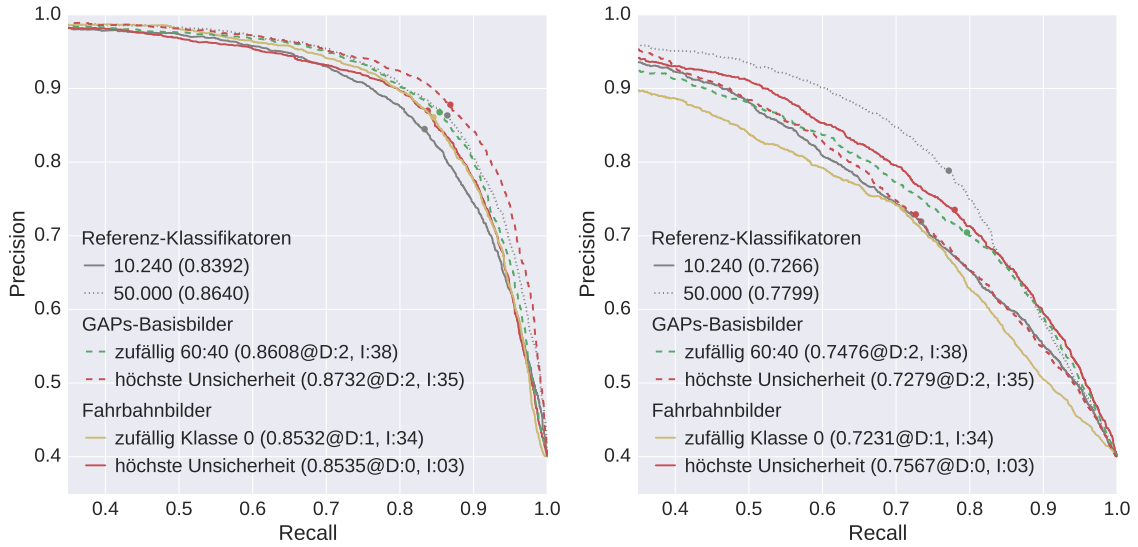
eine Erweiterung auf beide Klassen (blaue Kurve in Abbildung 6.11a), wie erwartet, zu noch einmal schlechteren Ergebnissen führt. In diesem Fall werden den Trainingsdaten zusätzlich fehlerhafte Patches für Klasse 1 hinzugefügt, sodass die Schadklasse zunehmend schlechter abgegrenzt werden kann und die Anzahl der False Positives ebenfalls ansteigt.

Die Lernprozesse in Abbildung 6.11b zeigen, dass sich die Einschränkung auf falsch klassifizierte Bildausschnitte auch bei den GAPs-Basisbildern überwiegend negativ auswirkt. Besonders bei einer Erweiterung mit Patches, welche als schwerwiegend falsch eingestuft werden, brechen die Klassifikationsergebnisse deutlich ein (gelbe Kurve in Abbildung 6.11b). Abbildung A.11 in Anhang A.4 (Seite 141) veranschaulicht die Gründe für diesen Einbruch. So werden der Schadklasse überwiegend wenig aussagekräftige Patches hinzugefügt, die oft nur an den eigentlichen Schaden angrenzen. Zusätzlich werden die Trainingsdaten für den Normalzustand mit nicht markierten Schäden oder nur schwer von Schäden abzugrenzenden Bildausschnitten erweitert. In der Folge können zunehmend weniger korrekte Klassifikationen vorgenommen werden (True Positives sinken, False Positives steigen stark an) und die Klassifikationsergebnisse brechen immer weiter ein. Einzig bei einer Erweiterung mit Patches, welche als unsicher eingestuft werden, zeigt sich ein positives Lernverhalten (hellblaue Kurve in Abbildung 6.11b). Die ausgewählten Bildausschnitte ähneln in diesem Fall sehr stark denen, welche bei einer ausschließlichen Berücksichtigung der Unsicherheit (siehe Abbildung A.8, Seite 138) hinzugefügt werden. Der Vergleich beider Lernprozesse (rote und hellblaue Kurve in Abbildung 6.11b) verdeutlicht jedoch, dass auch dieses Szenario durch die beschriebenen Probleme beeinflusst wird und nicht ganz die Leistung einer Auswahl ausschließlich auf Basis der Unsicherheiten erreicht werden kann.

Die Untersuchungen in diesem Abschnitt haben gezeigt, dass für die verfügbaren Straßendaten in Verbindung mit den eingesetzten Klassifikatoren kein klassisches Bootstrapping mit Hilfe von falsch klassifizierten Beispielen angewendet werden kann. Durch eine zusätzliche Berücksichtigung der Unsicherheiten zur Einschränkung der Beispiele auf unsichere Patches kann das Lernverhalten zwar verbessert werden, vergleichbare Ergebnisse, wie bei einer alleinigen Auswahl auf Basis der Unsicherheiten, können jedoch nicht erreicht werden.

6.3.4 Ergebnisse für Test-Validierungsdatensatz

Im Anschluss an die Untersuchungen zu den einzelnen Szenarien wurden die erfolgversprechendsten Lernprozesse, entsprechend der Empfehlung in [Ng, 2016], für einen Vergleich ebenfalls auf den Test-Validierungsdatensatz evaluiert. Abbildung 6.12 stellt die jeweils beste Iteration der Lernprozesse in einer Precision-Recall-Kurve sowohl für den Validierungs- als auch für den Test-Validierungsdatensatz dar.



(a) Validierungsdatensatz **(b)** Test-Validierungsdatensatz

Abbildung 6.12: Precision-Recall-Kurven für ausgewählte Lernprozesse
Dargestellt sind die nach dem Validierungsdatensatz jeweils besten Iterationen der erfolgversprechendsten Lernprozesse in einer Precision-Recall-Kurve sowohl für den Validierungs- als auch für den Test-Validierungsdatensatz. In Klammern ist der in den Kurven markierte maximale F1-Score angegeben. Für die Lernprozesse ist zusätzlich angezeigt, in welchem Durchlauf (D) und in welcher Iteration (I) dieses Ergebnis erzielt wurde.

Abbildung 6.12a vergleicht die besten Iterationen der einzelnen Lernprozesse zunächst auf dem Validierungsdatensatz. Hierbei wird deutlich, dass mit Hilfe des auf den GAPs-Basisbildern inkrementell und unter Berücksichtigung der Unsicherheiten trainierten Klassifikators (rote, gestrichelte Kurve in Abbildung 6.12a) sogar die Leistung des mit 50.000 Beispielen trainierten Referenz-Klassifikators (graue, gepunktete Linie in Abbil-

dung 6.12a) übertroffen werden kann. Das Ergebnis für den Lernprozess, bei dem die Trainingsdaten mit zufälligen Beispielen erweitert wurden (grüne, gestrichelte Kurve in Abbildung 6.12a), verdeutlicht zudem, dass auch mit einem inkrementellen Lernprozess und zufällig gewählten Beispielen fast bis zur Güte des Referenz-Klassifikators aufgeschlossen werden kann. Weiterhin wird deutlich, dass das bessere Ergebnis des unter Berücksichtigung der Unsicherheiten trainierten Klassifikators durch das Einbringen der Unsicherheiten in den Auswahlprozess bedingt ist[18]. Die Ergebnisse für die Lernprozesse (gelbe und rote Kurve in Abbildung 6.12a) auf den Fahrbahnbildern zeigen dagegen nur geringe Verbesserungen. Die Klassifikatoren kommen zu einem ähnlichen Ergebnis, welches zwischen der Leistung der Referenz-Klassifkatoren liegt.
Der in Abbildung 6.12b dargestellte Vergleich auf dem Test-Validierungsdatensatz zeigt ein deutlich anderes Bild. So ist die Klassifikationsleistung in allen Fällen geringer, was nahelegt, dass sich die Daten merklich voneinander unterscheiden. Zudem schneiden die auf den GAPs-Basisbildern inkrementell trainierten Klassifikatoren schlechter ab, als der auf den Fahrbahnbildern unter Berücksichtigung der Unsicherheiten trainierte Klassifikator (rote Kurve in Abbildung 6.12b). Diese Beobachtung deutet darauf hin, dass die auf Basis der Unsicherheiten aus den Fahrbahnbildern gewählten Bildausschnitte eine höhere Generalisierungsfähigkeit bieten. Unsichere Patches aus den GAPs-Basisbildern scheinen hingegen zu einer Überanpassung auf die Trainings- und Validierungsdaten zu führen. So schneidet dieser Klassifikator (rote, gestrichelte Kurve in Abbildung 6.12b) auf dem Test-Validierungsdatensatz deutlich schlechter ab als der Klassifikator, bei dem lediglich zufällige Patches hinzugefügt wurden (grüne, gestrichelte Kurve in Abbildung 6.12b).
Der Vergleich zeigt, dass die Fahrbahnbilder das Potenzial für Verbesserungen bieten, welches jedoch aufgrund der zu groben Labelinformationen nur sehr eingeschränkt genutzt werden kann. Weiterhin legen die Ergebnisse nahe, dass die Validierungsdaten deutlich ähnlicher zu den Trainingsdaten sind als die Beispiele aus dem Test-Validierungsdatensatz und dass eine Optimierung auf die Validierungsdaten nur bedingt zu einer guten Generalisierungsfähigkeit führt.

[18] Beide Ergebnisse wurden im selben Durchlauf und somit mit der selben zufälligen Reihenfolge der Ausgangsbilder erzielt.

6.4 Zusammenfassung

Im Folgenden werden die wichtigsten Erkenntnisse der experimentellen Untersuchungen noch einmal zusammengefasst. Zunächst wurden verschiedene Experimente auf dem GAPs-Datensatz durchgeführt. Hierbei hat sich gezeigt, dass sich beide Erweiterungsansätze (Dropout- und Ensemble-Ansatz) sowohl zur Realisierung von inkrementellen Lernprozessen als auch zur Extraktion von Unsicherheitsinformationen eignen. Die mit Hilfe des Ensemble-Ansatzes erweiterten Klassifikatoren sind jedoch deutlich schwerer zu trainieren und erreichen nicht die Leistung der mittels Dropout-Ansatz erweiterten Klassifikatoren. Der Einsatz von Adversarial Training konnte die Klassifikationsleistung bei der Verwendung des Dropout-Ansatzes zwar steigern, hat die Realisierung eines erfolgreichen inkrementellen Lernprozesses jedoch unmöglich gemacht. Für den Fall einer Erweiterung mittels Ensemble-Ansatz hat sich gezeigt, dass es aufgrund der geteilten Netzwerkstruktur zu Problemen bei der Erstellung zielführender Adversarial Examples kommt und dass die Klassifikationsleistung infolge dieser nicht gesteigert werden kann. Weiterhin wurden auf dem GAPs-Datensatz verschiedene Maße zur Modellierung der Unsicherheiten getestet. Für den Dropout-Ansatz hat sich hierbei die Variationsrate als besonders geeignet herausgestellt. Die Verwendung der maximalen Entropie konnte hingegen bei einer Erweiterung mittels Ensemble-Ansatz überzeugen. Insgesamt konnte das beste inkrementelle Lernverhalten mit Hilfe des Dropout-Ansatzes in Verbindung mit einer Unsicherheitsmodellierung mittels Variationsrate erzielt werden.

Dieses Gesamtsystem wurde anschließend sowohl auf die Fahrbahnbilder als auch auf die GAPs-Basisbilder angewendet. Hierbei hat sich gezeigt, dass die Klassifikationsleistung nur dann verbessert werden kann, wenn den Trainingsdaten neue Beispiele für beide Klassen hinzugefügt werden. Durch eine Berücksichtigung der extrahierten Unsicherheiten konnte das Lernverhalten zusätzlich positiv beeinflusst werden. In den Untersuchungen wurde außerdem festgestellt, dass eine Auswahl auf Basis der Unsicherheiten eine regulierende Wirkung auf die Klassenverteilung im Trainingsdatensatz hat und dass automatisch das für den Lernprozess optimale Verhältnis angestrebt wird. Das zusätzliche Einbeziehen der Klassifikatorausgabe in den Auswahlprozess

zur Realisierung an Bootstrapping angelehnter Szenarien konnte hingegen nicht überzeugen, da in diesem Fall überwiegend widersprüchlich gelabelte oder nur schwer zu unterscheidende Bildausschnitte ausgewählt werden. Insgesamt hat sich in allen Experimenten gezeigt, dass die Klassifikationsergebnisse auf dem Validierungsdatensatz nur dann deutlich verbessert werden können, wenn neue True Positives (korrekt als Schaden klassifizierte Patches) erschlossen werden. Die für die Fahrbahnbilder verfügbaren Labelinformationen machen dies jedoch unmöglich. In der Folge konnte die mit Hilfe der Fahrbahnbilder angestrebte inkrementelle Verbesserung der Klassifikationsleistung nicht realisiert werden. Der abschließende Vergleich mit Ergebnissen auf dem Test-Validierungsdatensatz hat gezeigt, dass die Validierungsdaten sehr ähnlich zu den Trainingsdaten sind und dass eine Optimierung auf die Validierungsdaten nur bedingt zu einer guten Generalisierungsfähigkeit führt.
Im Rahmen des Ausblicks im anschließenden Kapitel werden viele der genannten Punkte erneut aufgegriffen und mögliche Lösungsvorschläge präsentiert.

Kapitel 7

Zusammenfassung und Ausblick

Nachdem die umgesetzte Vorgehensweise zur Realisierung von intelligenten inkrementellen Lernprozessen in Kapitel 5 vorgestellt und wesentliche Aspekte anschließend in Kapitel 6 evaluiert wurden, werden in diesem Kapitel die wichtigsten Punkte und Ergebnisse dieser Masterarbeit zusammengefasst. Abschließend wird ein Überblick über mögliche Ansatzpunkte für weiterführende Arbeiten gegeben.

7.1 Zusammenfassung

Grundvoraussetzung für die Realisierung von intelligenten inkrementellen Lernprozessen ist die Modellierung der Unsicherheiten in den Klassifikatorausgaben. Nur mit deren Hilfe können möglichst informative, neue Trainingsbeispiele ausgewählt und redundante Daten vermieden werden. Zu Beginn dieser Masterarbeit wurden daher im Rahmen einer ausführlichen State-of-the-Art-Analyse Möglichkeiten zur Extraktion von Unsicherheitsinformationen recherchiert. Die Analyse hat ergeben, dass Deep-Learning-basierte Klassifikatoren zum Ableiten von Unsicherheitsinformationen strukturell erweitert werden müssen und dass es dafür zwei Optionen in Form von approximierenden bayes'schen und nicht-bayes'schen Ansätzen gibt. Mit dem Dropout- und dem Ensemble-Ansatz wurde für jede der beiden möglichen Vorgehensweisen ein vielversprechender Erweiterungsansatz ausgewählt und im weiteren Verlauf der Masterarbeit implementiert. Weiterhin wurden im Rahmen der State-of-the-Art-Analyse

Maße zur Modellierung der Unsicherheiten recherchiert, auf deren Basis möglichst informative, neue Trainingsbeispiele ausgewählt werden können. Die zugehörigen experimentellen Untersuchungen auf dem GAPs-Datensatz haben gezeigt, dass sich beide Erweiterungsansätze zur Realisierung von intelligenten inkrementellen Lernprozessen eignen. Die mit Hilfe des Ensemble-Ansatzes erweiterten Klassifikatoren sind jedoch deutlich schwerer zu trainieren und führen im Vergleich zum Dropout-Ansatz zu geringfügig schlechteren Klassifikationsergebnissen[1]. Das beste inkrementelle Lernverhalten konnte mit Hilfe des Dropout-Ansatzes in Verbindung mit einer Unsicherheitsmodellierung mittels Variationsrate erzielt werden.

Anschließend wurde das anhand der Untersuchungen auf dem GAPs-Datensatz abgeleitete Gesamtsystem zur Realisierung von intelligentem inkrementellem Lernen auf die verfügbaren Fahrbahnbilder angewendet. Bevor diese jedoch verwendet werden konnten, mussten zunächst mehrere Vorverarbeitungsschritte umgesetzt werden. Damit die verfügbaren Labelinformationen den korrekten Bildbereichen in den Fahrbahnbildern zugeordnet werden können, musste ein Verfahren zur Bestimmung der genauen Fahrbahnposition in einem Bild realisiert werden. Weiterhin musste ein Algorithmus zur Erkennung und Maskierung der in den Bildern enthaltenen Stitching-Kanten umgesetzt werden. Dieser Schritt ist notwendig, um Fehlklassifikationen infolge der starken Ähnlichkeit der Kanten zu Rissen vorbeugen zu können. Beide Verfahren wurden auf den verfügbaren Fahrbahnbildern evaluiert und können durch ihren allgemeingültigen Ansatz über diese Masterarbeit hinaus auch auf andere Fahrbahnbilder angewendet werden. Die durchgeführten experimentellen Untersuchungen auf den Fahrbahnbildern haben jedoch gezeigt, dass die verfügbaren Labelinformationen zu großflächig sind und die Realisierung des angestrebten inkrementellen Lernprozesses massiv erschweren. Ausschlaggebend für das Ausbleiben von Verbesserungen ist dabei, dass dem Trainingsdatensatz aufgrund der zu ungenauen Labelinformationen nur wenige korrekt gelabelte Beispiele für die Schadklasse hinzugefügt werden können. Das zu-

[1] Die für den Ensemble-Ansatz verwendeten Lernparameter wurden größtenteils anhand des in [Eisenbach et al., 2017] vorgestellten Klassifikators abgeleitet. Es kann daher nicht ausgeschlossen werden, dass eine andere Parameterwahl zu einem besseren Lernverhalten geführt hätte. Durch den begrenzten zeitlichen Rahmen der Masterarbeit konnten jedoch keine ausgiebigen Parametertests durchgeführt werden.

sätzliche Berücksichtigen der Unsicherheiten konnte dieses Verhalten ebenfalls nicht verbessern. Aus diesem Grund wurden ergänzende Untersuchungen auf den Basisbildern des GAPs-Datensatzes durchgeführt. Diese haben gezeigt, dass eine inkrementelle Verbesserung mit genauen Schadensmarkierungen möglich ist und dass das Einbeziehen der Unsicherheiten das Lernverhalten verbessern kann. Die Experimente haben aber auch gezeigt, dass die Auswirkungen von widersprüchlich gelabelten und nur schwer von Schäden zu unterscheidenden Bildausschnitten durch die Berücksichtigung der Unsicherheiten verstärkt werden.

7.2 Weiterführende Arbeiten

Aufbauend auf den Ergebnissen und Erkenntnissen dieser Masterarbeit ergeben sich zahlreiche Ansatzpunkte für anschließende Arbeiten. So könnten weiterführende Untersuchungen zu den zur Extraktion von Unsicherheitsinformationen verwendeten Erweiterungsansätzen durchgeführt werden. Speziell für den Ensemble-Ansatz würde sich eine systematische Evaluierung der Lernparameter anbieten. Die Verwendung des Ensemble-Ansatzes zur Extraktion von Unsicherheitsinformationen hätte den Vorteil, dass sich die in der Anwendungsphase erforderliche Zeit zur Verarbeitung eines Bildes im Gegensatz zum Dropout-Ansatz kaum verlängert[2]. Für den Dropout-Ansatz könnte die kürzlich in [Li und Gal, 2017] publizierte Erweiterung zur Verbesserung der Qualität der extrahierten Unsicherheitsinformationen betrachtet werden. In dieser Publikation wird gezeigt, dass das für Variational Inference typische Unterschätzen der Unsicherheiten durch die Verwendung einer erweiterten, aber dennoch auf Variational Inference aufbauenden Inferenzmethode vermieden werden kann. Die Anwendung der vorgeschlagenen Vorgehensweise impliziert jedoch auch, dass in der Trainingsphase für die Verarbeitung eines Mini-Batches bis zu zehn Vorwärtsprozesse durch das Netzwerk betrachtet werden müssen. Daher sollte an dieser Stelle eine zusätzliche Abwägung zwischen Aufwand und Nutzen erfolgen. Nicht zuletzt wäre eine Kombination beider Ansätze zur Extraktion von Unsicherheitsinformationen ebenfalls denkbar.

[2] Die Zeitersparnis ergibt sich durch die verwendete geteilte Netzwerkstruktur. Gleichzeitig muss jedoch mit einem erhöhten Zeitbedarf für das Training gerechnet werden.

Unabhängig vom genutzten Erweiterungsansatz sollte ebenso die Verwendung größerer Bildausschnitte untersucht werden. Im Rahmen der Masterarbeit konnte immer wieder beobachtet werden, dass bei einer Patchgröße von 64x64 Pixel Bildausschnitte existieren, welche durch Menschen kaum unterschieden werden können und für dessen korrekte Labelbestimmung das Einbeziehen angrenzender Bildbereiche notwendig ist. Die Verwendung größerer Patches ergibt sich als mögliche Konsequenz dieser Beobachtung. Alternativ könnten auch Netzwerkstrukturen genutzt werden, welche zusätzlich räumliche Zusammenhänge zwischen einzelnen Patches berücksichtigen können.
Obwohl sich in dieser Masterarbeit herausgestellt hat, dass die Fahrbahnbilder nicht direkt für die Realisierung von inkrementellen Lernprozessen genutzt werden können, ergeben sich auf Basis der umgesetzten Verfahren mehrere Möglichkeiten, wie die Bilder dennoch zur Verbesserung der Klassifikationsleistung eingesetzt werden können. Zum einen könnten menschliche Operatoren in den inkrementellen Lernprozess eingebunden werden, mit deren Hilfe die dem Datensatz hinzugefügten Patches vor dem Nachtraining des Klassifikators überprüft werden. Die Klassifikatorausgaben und die extrahierten Unsicherheiten könnten diesen Validierungsprozess zusätzlich unterstützen. Zum anderen könnte das im Rahmen der Vorverarbeitung realisierte Verfahren zur Bestimmung der Fahrbahnposition in den Fahrbahnbildern genutzt werden, um die Labelinformationen konkreten Bildteilen zuzuordnen. Durch eine gezielte Extraktion der als Schaden markierten Bereiche aus dem Gesamtbestand der Fahrbahnbilder und einem anschließenden händischen Labeln mit genauen Schadensmarkierungen, könnte ein Datensatz mit deutlich größerer Diversität und möglicherweise besseren Generalisierungseigenschaften erstellt werden.

Anhang A

Ergänzungen zu den Experimenten

Im Folgenden werden Ergänzungen zu den durchgeführten experimentellen Untersuchungen beschrieben.

A.1 Ensemble aus mehreren Netzwerken

In diesem Abschnitt wird die Klassifikationsleistung von Ensembles bestehend aus einem geteilten Netzwerk und verschiedenen Heads mit der von herkömmlichen Ensembles aus mehreren eigenständigen Netzwerken verglichen. Ziel ist es, den Einfluss der geteilten Netzwerkarchitektur auf die Klassifikationsleistung zu untersuchen.
Für den Vergleich mit den in Abschnitt 6.1.2 (Seite 97) beschriebenen Referenz-Klassifikatoren wurden zehn Einzelnetzwerke ohne Adversarial Training und sieben eigenständige Netzwerke mit Adversarial Training trainiert. Zur Gewährleistung von Diversität wurden die Gewichte der Netzwerke unterschiedlich initialisiert und die beschriebenen binären Sample Weights ebenfalls auf das Einzeltraining übertragen. Alle weiteren Lernparameter wurden unverändert aus Abschnitt 6.1.2 (Seite 97) übernommen. Da das Training einzelner Klassifikatoren deutlich zeitaufwendiger ist, wurden lediglich die Klassifikatoren für 10.240 und 50.000 Trainingsbeispiele trainiert und für den Vergleich herangezogen. Die zusätzlich trainierten Ensembles wurden zunächst in Anlehnung an die Vorgehensweise bei einem Ensemble mit einer geteilten Netzwerkarchitektur evaluiert. Dafür wurde für jede Epoche die mittlere Klassifikatorausgabe

bestimmt und die Epoche mit der besten Gesamtleistung gewählt. Zusätzlich wurden die Einzelnetzwerke, wie bei einem herkömmlichen Ensemble-Ansatz üblich, getrennt evaluiert und ausschließlich die Klassifikatoren in ihrer jeweils besten Konfiguration zur Bestimmung der mittleren Ausgabe genutzt. Tabelle A.1 zeigt die Ergebnisse der Untersuchungen.

		Ensemble-Realisierung		
		geteilt (Abschnitt 6.1.2)	eigenständig	eigenständig*
10.240	4 Heads/Netzwerke	0,8318 (220)	0,8328 (464)	0,8417
	7 Heads/Netzwerke	0,8333 (227)	0,8377 (692)	0,8486
	10 Heads/Netzwerke	0,8337 (234)	0,8363 (690)	0,8483
50.000	4 Heads/Netzwerke	0,8580 (091)	0,8657 (697)	0,8773
	7 Heads/Netzwerke	0,8583 (183)	0,8653 (662)	0,8820
	10 Heads/Netzwerke	0,8607 (095)	0,8678 (794)	0,8840
10.240	4 Heads/Netzwerke (AT)	0,8415 (651)	0,8573 (352)	0,8680
	7 Heads/Netzwerke (AT)	0,8323 (606)	0,8574 (352)	0,8694
50.000	4 Heads/Netzwerke (AT)	0,8638 (463)	0,8789 (453)	0,8873
	7 Heads/Netzwerke (AT)	0,8567 (788)	0,8799 (432)	0,8918

Tabelle A.1: Erzielte F1-Scores der Referenz-Klassifikatoren

Dargestellt sind die von den Referenz-Klassifikatoren erzielten F1-Scores in Abhängigkeit der Headanzahl und der Größe des Trainingsdatensatzes. Die jeweils beste Epoche ist in Klammern angegeben. „AT“ zeigt die Verwendung von Adversarial Training an. Zur Generierung der mit „“ gekennzeichneten Ergebnisse wurden ausschließlich die Klassifikatoren in ihrer jeweils besten Konfiguration verwendet.*

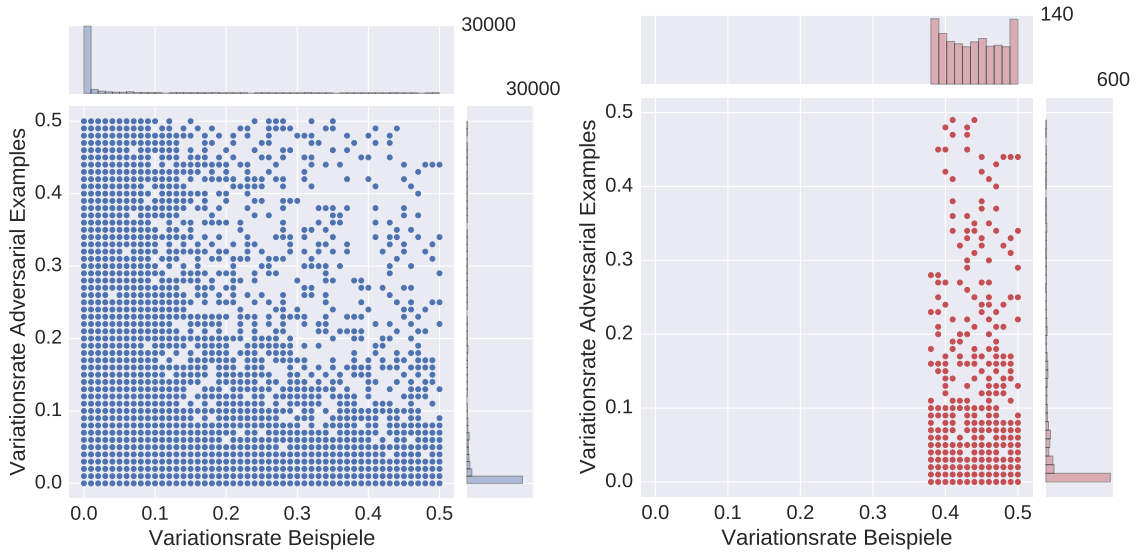
Die in Tabelle A.1 dargestellten Ergebnisse zeigen, dass sich die geteilte Netzwerkarchitektur in allen Fällen negativ auf die Klassifikationsleistung auswirkt. So können mit einem Ensemble aus mehreren eigenständigen Netzwerken trotz gleicher Vorgehensweise bei der Evaluation bessere Ergebnisse erzielt werden (erste und zweite Spalte in Tabelle A.1). Die Verwendung der Einzelnetzwerke in ihrer jeweils besten Konfiguration (dritte Spalte in Tabelle A.1) führt des Weiteren zu dem von einem Ensemble erwarteten Zugewinn in der Klassifikationsleistung. Die Ergebnisse für die mit Adversarial Training trainierten Klassifikatoren zeigen zudem, dass die in Abschnitt 6.1.2 (Sei-

te 97) beschriebenen Probleme bei der Nutzung von Adversarial Training durch die geteilte Netzwerkarchitektur bedingt sind und durch Verwendung von Einzelnetzwerken vermieden werden können. Das Ensemble aus sieben eigenständigen Netzwerken in ihrer jeweils besten Konfiguration erzielt mit einem F1-Score von 0,8918 das bislang beste Ergebnis für einen Klassifikator auf Basis von 64x64 Pixel großen Patches. Ein permanentes Training mehrerer Netzwerke ist jedoch sehr zeitintensiv und daher nicht realisierbar.

A.2 Berücksichtigung der Adversarial Examples

Die in Abschnitt 6.2.1 (Seite 102) dargestellten Experimente haben gezeigt, dass bei Verwendung des Dropout-Ansatzes in Verbindung mit Adversarial Training auch unter Zuhilfenahme der Unsicherheiten kein inkrementeller Lernprozess realisiert werden kann. Daher wurde für eine Modellierung der Unsicherheiten mittels Variationsrate ergänzend untersucht, ob das Lernverhalten durch eine zusätzliche Berücksichtigung der Adversarial Examples bei der Auswahl neuer Beispiele verbessert werden kann.
Sowohl für den Erstellungsprozess als auch für die Berücksichtigung ergeben sich mehrere Möglichkeiten. In jeder Iteration können entweder deterministische (ohne Dropout) oder aber auch probabilistische (mit Dropout) Adversarial Examples erzeugt und zusätzlich mit in den Entscheidungsprozess eingebracht werden. Probabilistische Adversarial Examples können zudem einmalig oder für jeden Vorwärtsprozess separat erstellt werden. Während eine einmalige Erzeugung die für die Bestimmung neuer Trainingsbeispiele benötigte Zeit ungefähr verdoppelt, führt die für jeden Vorwärtsprozess getrennte Erstellung zu deutlich längeren Verarbeitungszeiten. Im Rahmen dieser Masterarbeit wurden alle drei Erstellungsmethoden getestet. Hierbei hat sich herausgestellt, dass deterministisch erzeugte Adversarial Examples am aussagekräftigsten sind[1]. Abbildung A.1 zeigt den von den in der ersten Iteration verfügbaren Beispielen aufgespannten Raum der Unsicherheiten und veranschaulicht die ohne Berücksichtigung der Adversarial Examples gewählten neuen Trainingsbeispiele.

[1] Diese Erkenntnis legt nahe, dass Adversarial Examples auch in der Trainingsphase ohne Dropout erzeugt werden sollten.



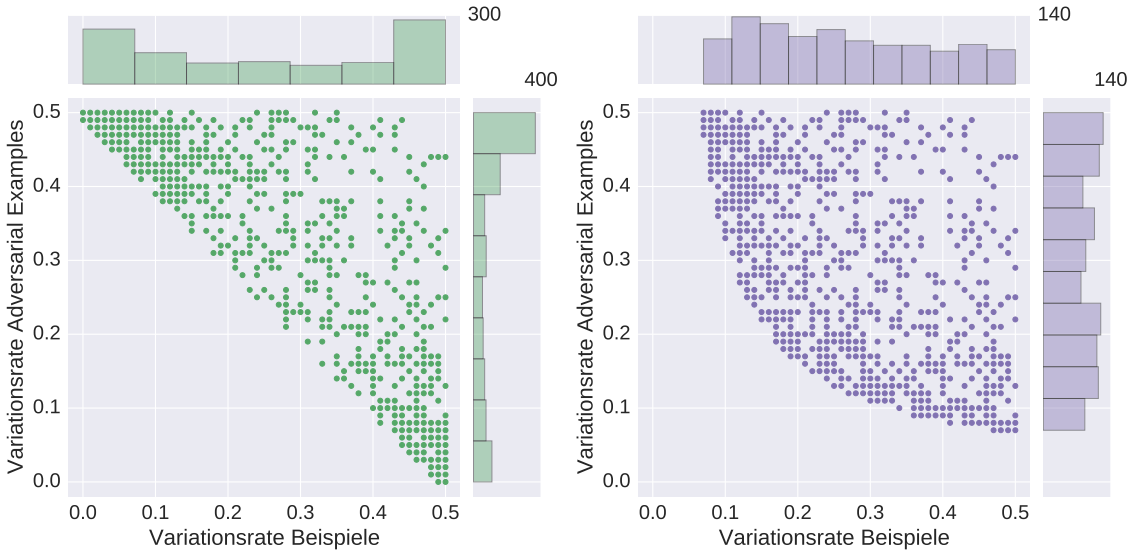
(a) Raum der Unsicherheiten **(b)** Auswahl ohne Berücksichtigung

Abbildung A.1: Raum der Unsicherheiten

Dargestellt ist der von den in der ersten Iteration verfügbaren Beispielen aufgespannte Raum der Unsicherheiten und die ohne Berücksichtigung der Adversarial Examples gewählten neuen Trainingsbeispiele.

Der in Abbildung A.1a dargestellte Raum der Unsicherheiten zeigt, dass nur sehr wenige Bildausschnitte existieren, bei denen der Klassifikator sowohl bei den normalen Beispielen als auch bei den Adversarial Examples unsicher ist (Bereich oben rechts). Die Visualisierung der ohne Berücksichtigung der Adversarial Examples ausgewählten Beispiele in Abbildung A.1b zeigt, dass nur ein kleiner Teil dieser möglichst informativen Bildausschnitte gewählt wird. Der Großteil der für das Training ausgewählten Beispiele besitzt eine Variationsrate von Null bei den Adversarial Examples. Aus diesem Grund wurde in verschiedenen Auswahlstrategien getestet, wie die Unsicherheiten für die Adversarial Examples besser berücksichtigt werden können. Bei den Experimenten hat sich herausgestellt, dass das arithmetische und das geometrische Mittel aus beiden für ein Beispiel bestimmten Unsicherheiten zu den besten Ergebnissen führt[2]. Abbildung A.2 veranschaulicht die anhand dieser beiden Auswahlstrategien ausgewählten neuen Trainingsbeispiele. Die Ergebnisse werden in Abbildung A.3 dargestellt.

[2] Es wurden weiterhin verschiedene Wichtungsfaktoren getestet.



(a) Arithmetisches Mittel **(b)** Geometrisches Mittel

Abbildung A.2: Möglichkeiten zur Berücksichtigung der Adversarial Examples
Dargestellt sind die getesteten Möglichkeiten zur Berücksichtigung der Adversarial Examples bei der Auswahl neuer Trainingsbeispiele.

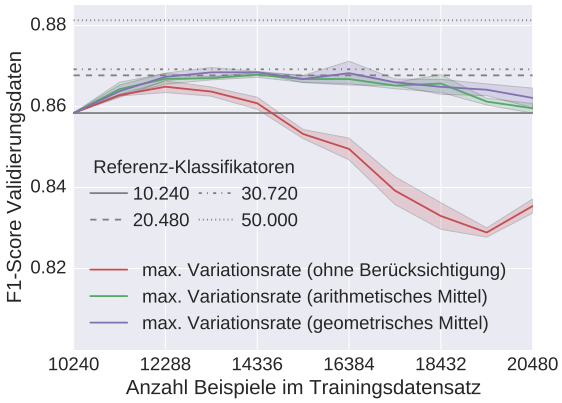


Abbildung A.3: Ergebnisse Berücksichtigung Adversarial Examples
Dargestellt sind die Ergebnisse (mittlerer F1-Score und einfache Standardabweichung) der experimentellen Untersuchungen zur Berücksichtigung der Adversarial Examples bei der Auswahl neuer Trainingsbeispiele.

Die dargestellten Ergebnisse zeigen, dass das Lernverhalten verbessert werden kann. Die Probleme können jedoch nicht vollständig überwunden werden. Zudem erfordert diese Vorgehensweise deutlich mehr Aufwand und benötigt exakte Labelinformationen.

A.3 Ergänzungen zu Abschnitt 6.3.2

In diesem Abschnitt wird ergänzendes Material zu den in Abschnitt 6.3.2 (Seite 111) vorgestellten experimentellen Untersuchungen dargestellt. Die nachfolgenden Abbildungen A.4-A.9 visualisieren die den Trainingsdaten hinzugefügten Beispiele. Patches für Klasse 0 werden dabei gelb und Ausschnitte für Klasse 1 blau überlagert dargestellt. Verbotene Bereiche sind weiterhin rot und gelabelte Schäden grün gekennzeichnet.

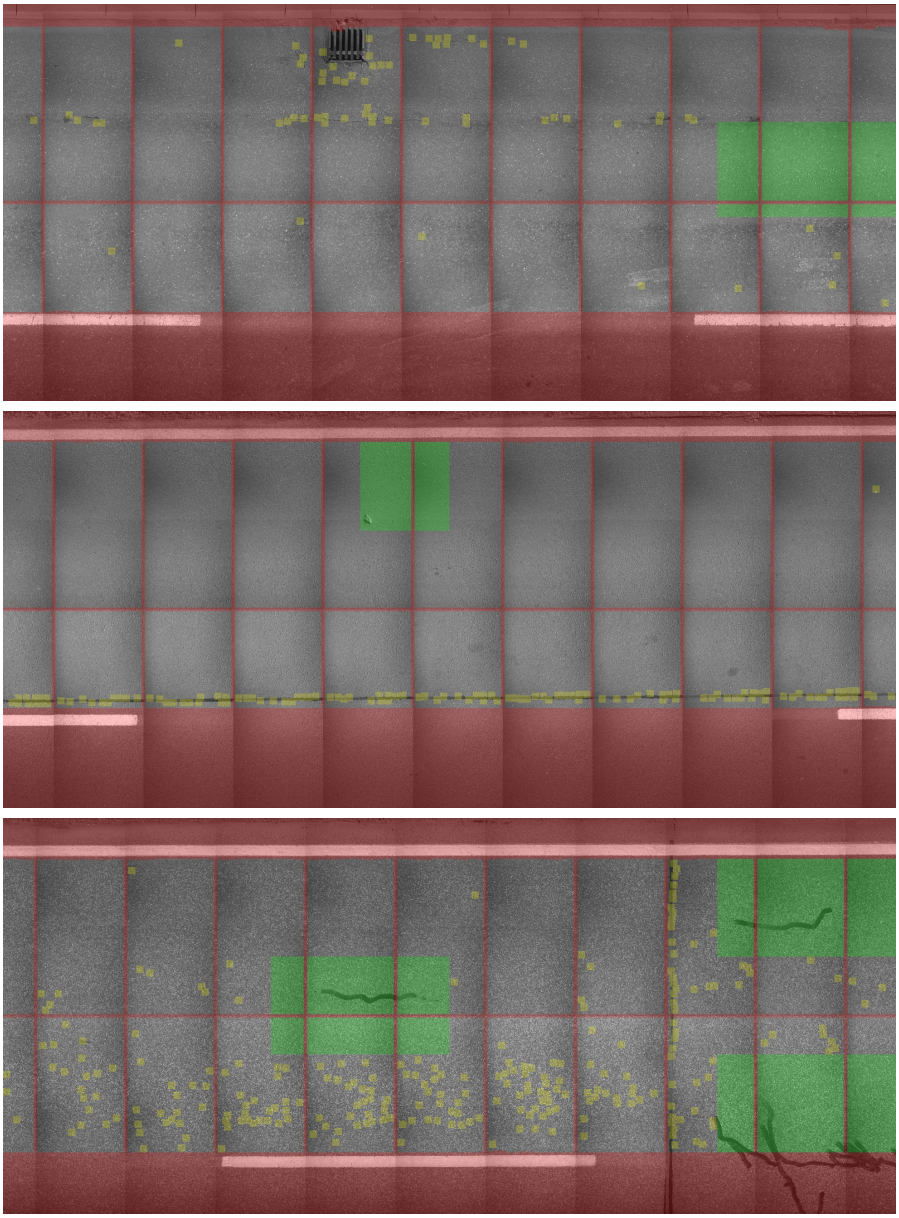


Abbildung A.4: Fahrbahnbilder Patches höchste Unsicherheit Klasse 0

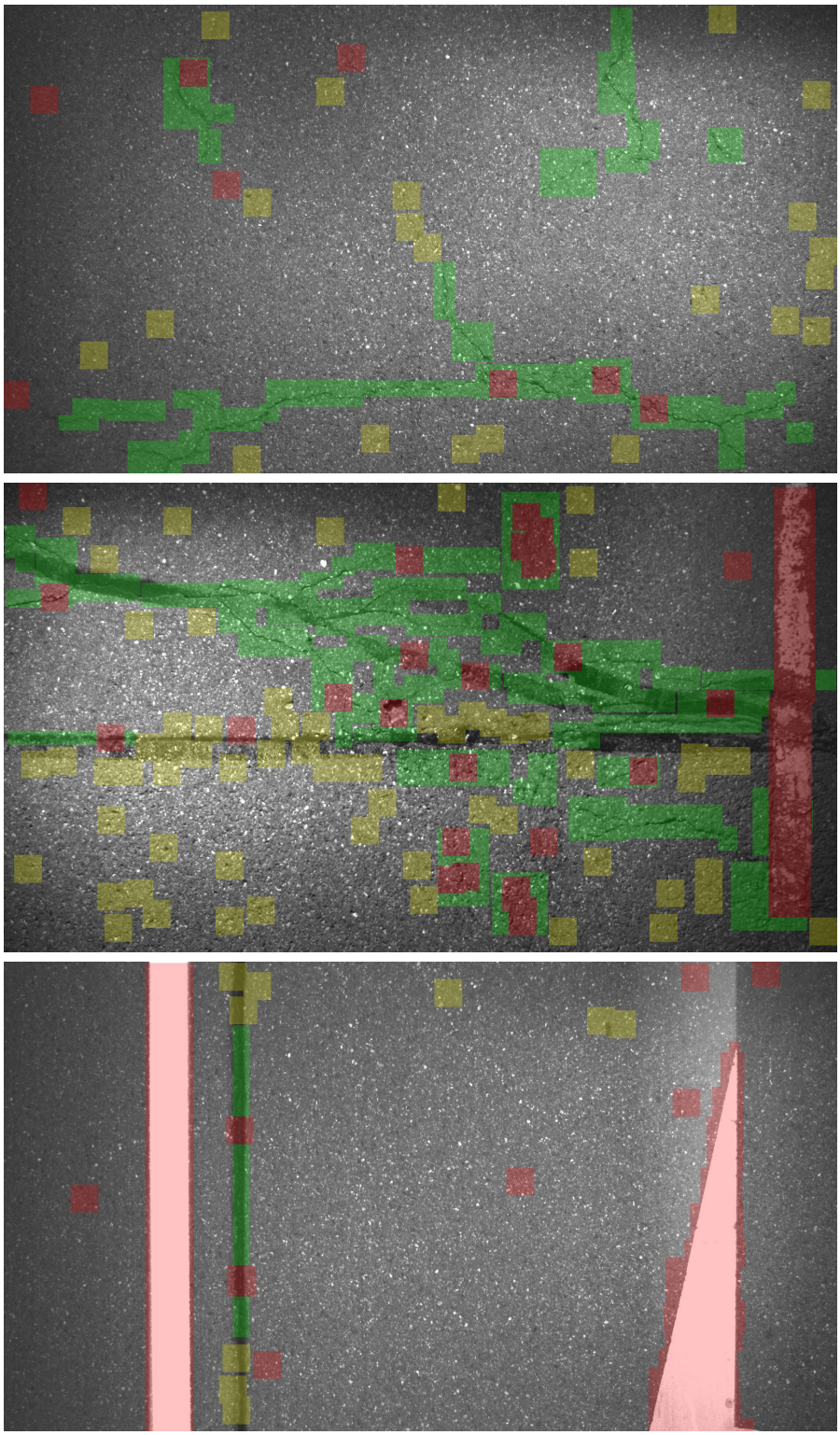


Abbildung A.5: GAPs-Basisbilder Patches höchste Unsicherheit Klasse 0

Für eine Erläuterung der farbigen Überlagerungen sei auf Seite 134 verwiesen.

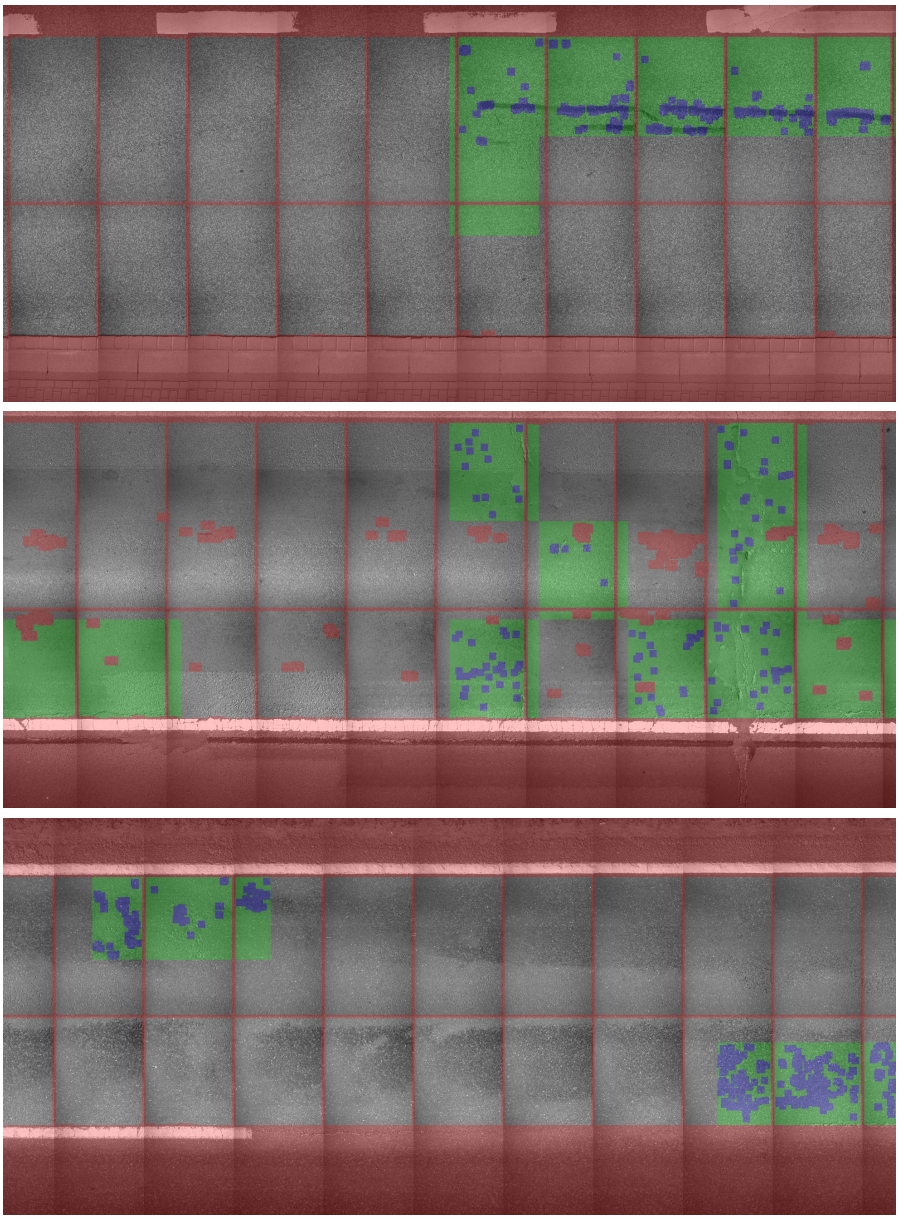


Abbildung A.6: Fahrbahnbilder Patches höchste Unsicherheit Klasse 1

Für eine Erläuterung der farbigen Überlagerungen sei auf Seite 134 verwiesen.

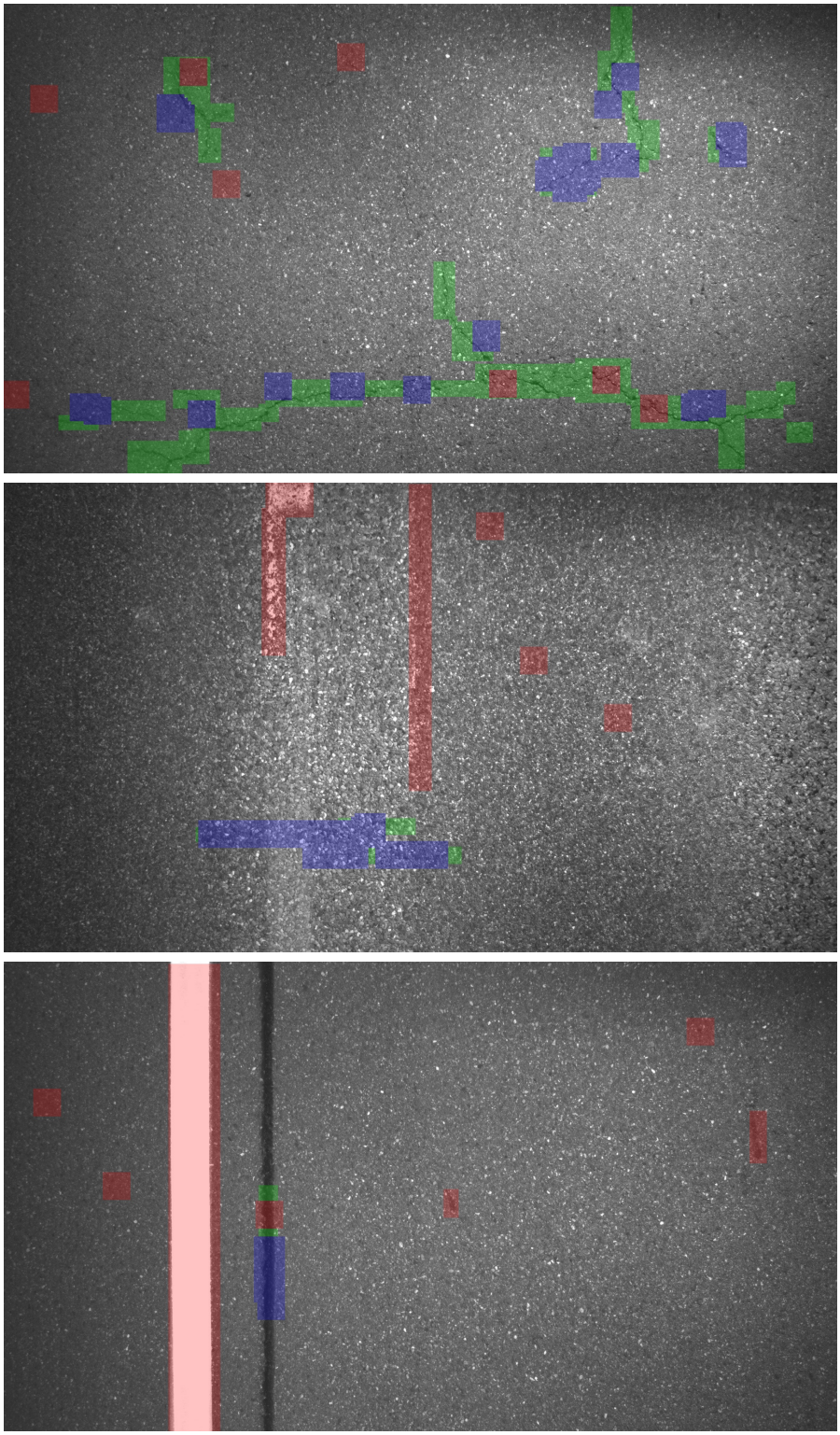


Abbildung A.7: GAPs-Basisbilder Patches höchste Unsicherheit Klasse 1

Für eine Erläuterung der farbigen Überlagerungen sei auf Seite 134 verwiesen.

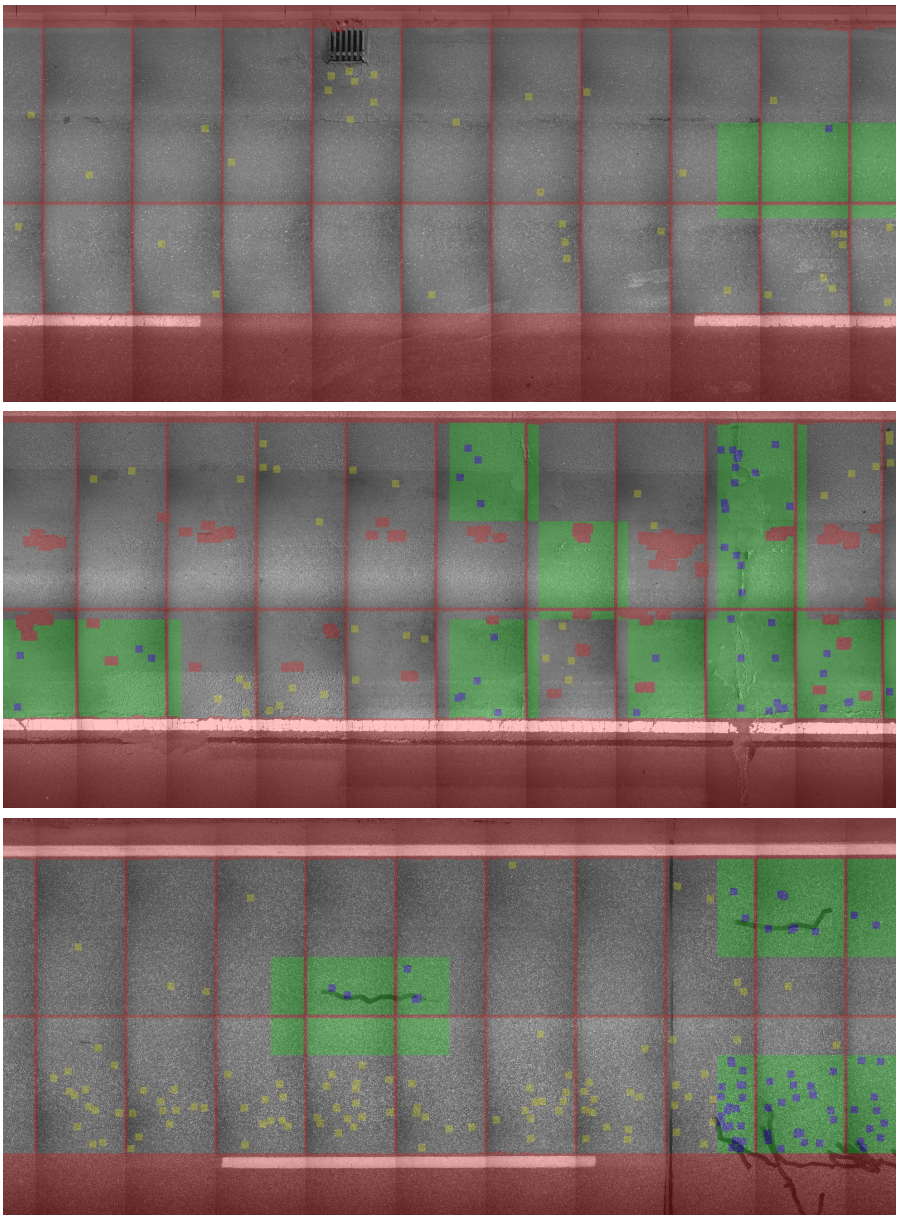


Abbildung A.8: Fahrbahnbilder Patches höchste Unsicherheit

Für eine Erläuterung der farbigen Überlagerungen sei auf Seite 134 verwiesen.

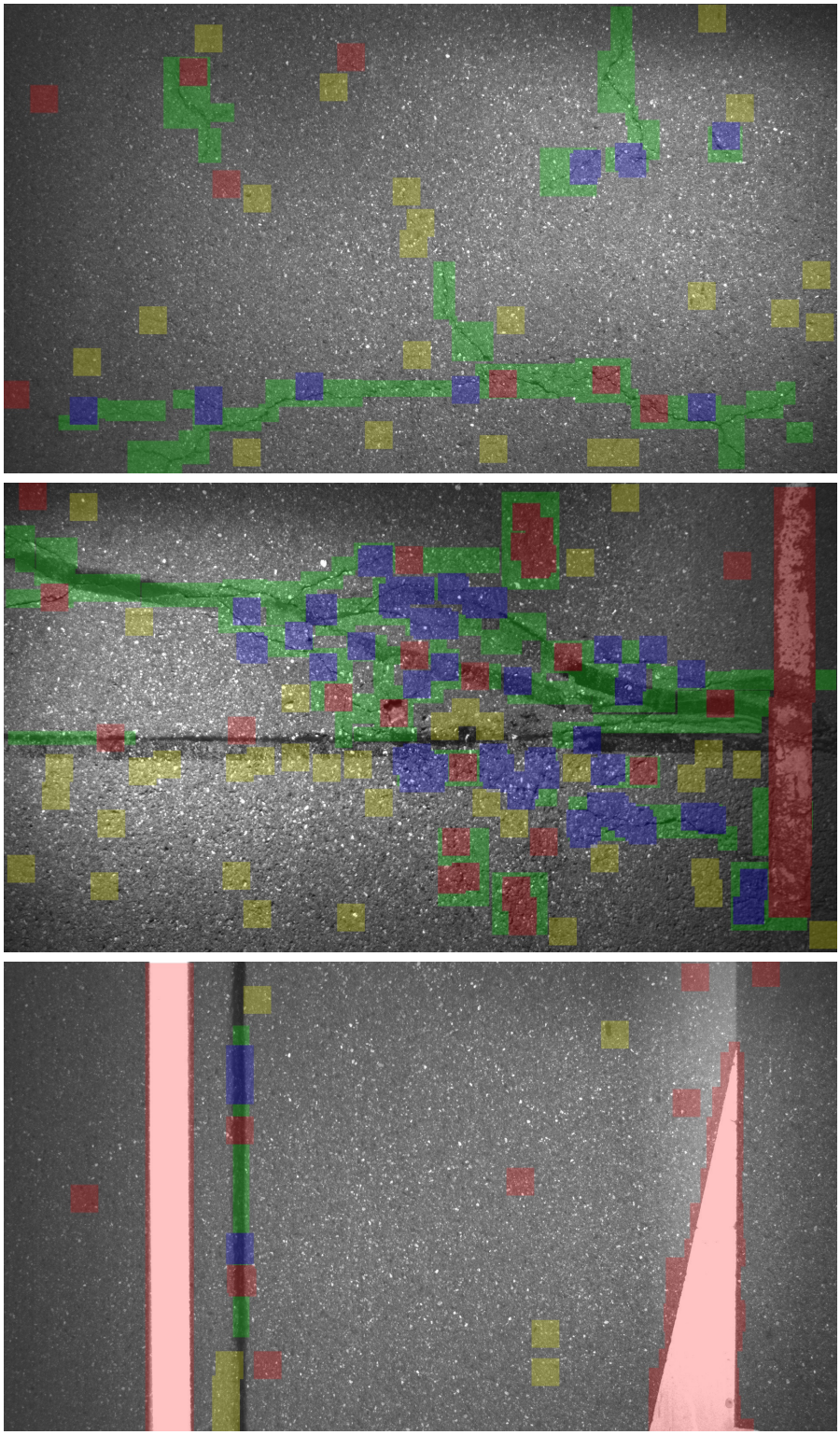


Abbildung A.9: GAPs-Basisbilder Patches höchste Unsicherheit

Für eine Erläuterung der farbigen Überlagerungen sei auf Seite 134 verwiesen.

A.4 Ergänzungen zu Abschnitt 6.3.3

In diesem Abschnitt wird ergänzendes Material zu den in Abschnitt 6.3.3 (Seite 117) vorgestellten Untersuchungen dargestellt. Die nachfolgenden Abbildungen A.10-A.11 visualisieren die den Trainingsdaten hinzugefügten Beispiele. Patches für Klasse 0 werden dabei gelb und Ausschnitte für Klasse 1 blau überlagert dargestellt. Verbotene Bereiche sind weiterhin rot und gelabelte Schäden grün gekennzeichnet.

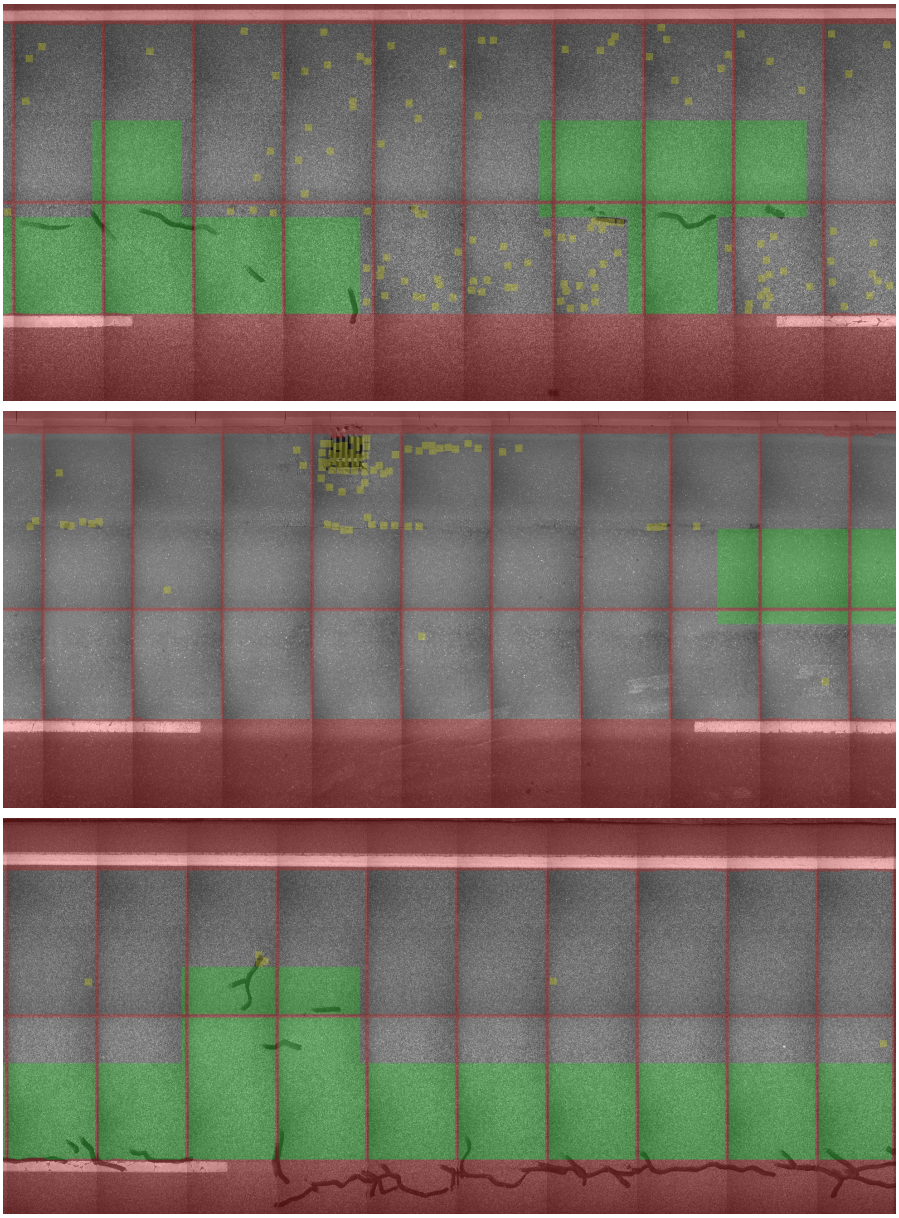


Abbildung A.10: Fahrbahnbilder Patches als Klasse 1 klassifiziert obwohl Klasse 0

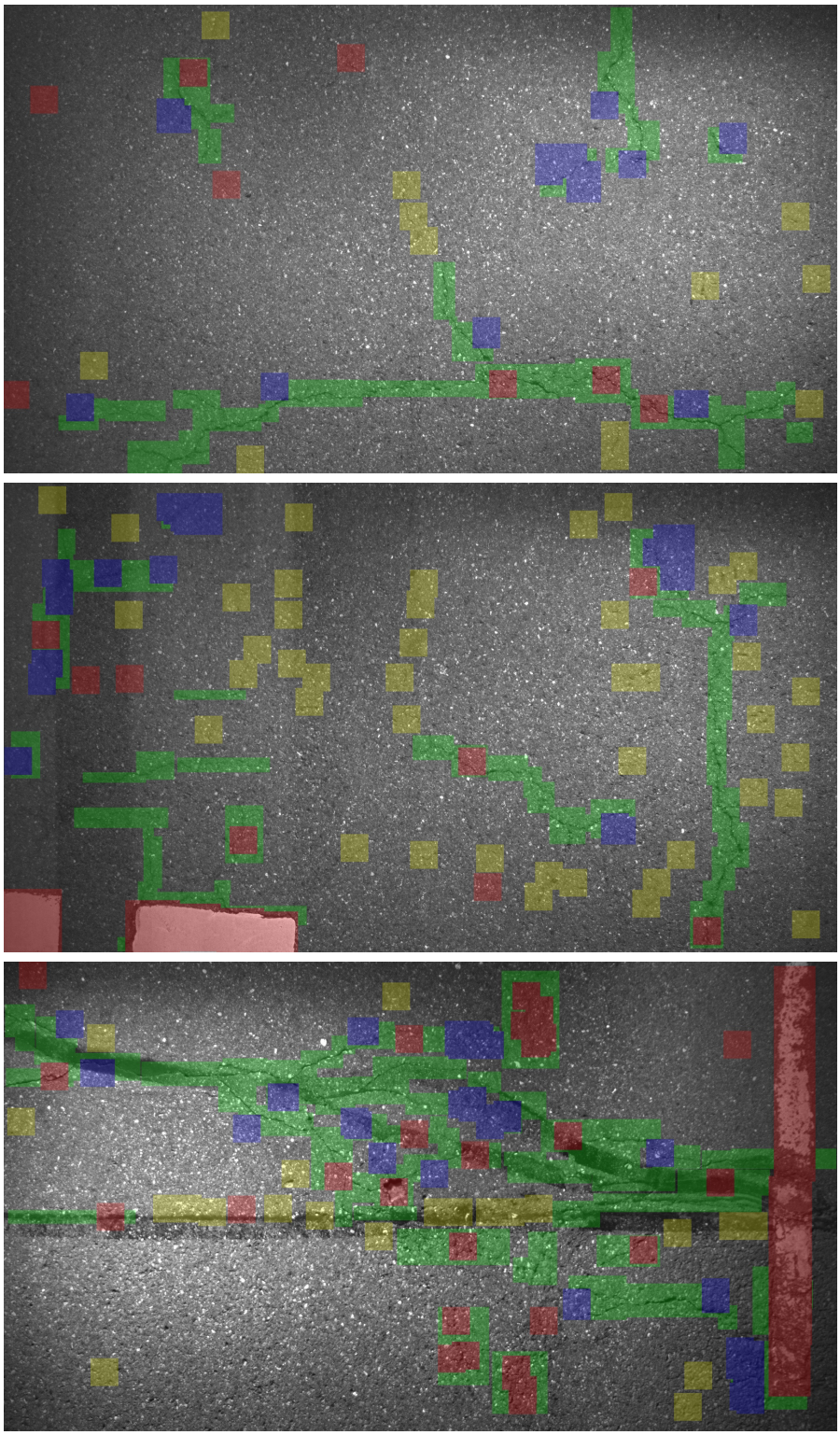


Abbildung A.11: GAPs-Basisbilder Patches falsch klassifiziert (niedrigste Unsicherheit)

Für eine Erläuterung der farbigen Überlagerungen sei auf Seite 140 verwiesen.

Abbildungsverzeichnis

1.1 Datenaufnahme im ASINVOS-Projekt 2
1.2 Datengrundlage im ASINVOS-Projekt 3

2.1 Grafische Darstellung der drei Hauptlernparadigmen 6
2.2 Veränderung der Netzwerkstruktur durch Dropout 10
2.3 Beispiel für ein Adversarial Example 13

3.1 Überblick über State of the Art 19
3.2 Definition der Gewichte 23
3.3 Bayesian SegNet 27
3.4 Netzwerkstruktur nach [Osband et al., 2016] 29
3.5 Vergleich verschiedener Akquirierungsfunktionen 32
3.6 Struktur eines Ladder Networks nach [Rasmus et al., 2015a] 35

4.1 Schadklassen 40
4.2 Gelabelte Full-HD-Bilder für den GAPs-Datensatz 41
4.3 Beispielfahrbahnbilder 44
4.4 Beispielfahrbahnbilder mit Labelinformationen 45
4.5 Beispielfahrbahnbilder mit Labelinformationen 46
4.6 Fehldetektionen infolge von Stitching-Kanten 47
4.7 Beispielfahrbahnbilder 48
4.8 Visualisierung der Peak-Erkennung 50
4.9 Detektionsergebnisse 52
4.10 Pseudocode: Algorithmus zur Lokalisierung der Stitching-Kanten 54

4.11 Detektion fahrbahnbegrenzender Strukturen 60
4.12 Verknüpfung zu einer Gesamtmaske für unzulässige Bereiche 61
4.13 Bestimmung der Lage der Fahrbahn 63
4.14 Angabe der Sicherheit bei der Detektion 65
4.15 Pseudocode: Algorithmus zur Bestimmung des Labelrasters 69
4.16 Beispielbilder zur Erläuterung ausstehender Probleme 72

5.1 Übersicht intelligentes inkrementelles Lernen 76
5.2 Aufbau des ASINVOS-Netzwerk 78
5.3 Strukturelle Modifikation für Ensemble-Ansatz 81
5.4 Veranschaulichung der implementierten Trainingsprozedur 82
5.5 Modifikationen am Basismodell für Ensemble-Ansatz 84
5.6 Umwandlung vollverschaltete Schicht in Faltungs-Schicht 86
5.7 Extraktion von Patches am Beispiel 89

6.1 Ergebnisse Eignung Dropout-Ansatz 96
6.2 Ergebnisse Eignung Ensemble-Ansatz 100
6.3 Ergebnisse Bestimmung der Anzahl der Vorwärtsprozesse 103
6.4 Ergebnisse Dropout-Ansatz 104
6.5 Ergebnisse Bestimmung der Headanzahl 105
6.6 Ergebnisse Ensemble-Ansatz 106
6.7 Ergebnisse zufällige Auswahl 110
6.8 Ergebnisse klassenweise Erweiterung höchste Unsicherheit 112
6.9 Ergebnisse Erweiterung höchste Unsicherheit 114
6.10 Klassenverteilung Erweiterung höchste Unsicherheit 116
6.11 Ergebnisse Erweiterung mit falsch klassifizierten Beispielen 118
6.12 Precision-Recall-Kurven für ausgewählte Lernprozesse 120

A.1 Raum der Unsicherheiten 132
A.2 Möglichkeiten zur Berücksichtigung der Adversarial Examples 133
A.3 Ergebnisse Berücksichtigung Adversarial Examples 133
A.4 Fahrbahnbilder Patches höchste Unsicherheit Klasse 0 134

A.5 GAPs-Basisbilder Patches höchste Unsicherheit Klasse 0 135
A.6 Fahrbahnbilder Patches höchste Unsicherheit Klasse 1 136
A.7 GAPs-Basisbilder Patches höchste Unsicherheit Klasse 1 137
A.8 Fahrbahnbilder Patches höchste Unsicherheit 138
A.9 GAPs-Basisbilder Patches höchste Unsicherheit 139
A.10 Fahrbahnbilder Patches als Klasse 1 klassifiziert obwohl Klasse 0 140
A.11 GAPs-Basisbilder Patches falsch klassifiziert (niedrigste Unsicherheit) . 141

Literaturverzeichnis

[Badrinarayanan et al., 2015] Badrinarayanan, Vijay, A. Kendall und R. Cipolla (2015). *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. arXiv preprint arXiv:1511.00561. *(Seite: 27)*

[Bengio et al., 2007] Bengio, Yoshua, P. Lamblin, D. Popovici und H. Larochelle (2007). *Greedy Layer-Wise Training of Deep Networks*. In: *Advances in Neural Information Processing Systems (NIPS)*, S. 153–160. MIT Press. *(Seite: 7)*

[Blei et al., 2016] Blei, David M, A. Kucukelbir und J. D. McAuliffe (2016). *Variational Inference: A Review for Statisticians*. arXiv preprint arXiv:1601.00670. *(Seite: 16, 18)*

[Blundell et al., 2015] Blundell, Charles, J. Cornebise, K. Kavukcuoglu und D. Wierstra (2015). *Weight Uncertainty in Neural Networks*. In: *Proc. of the 32nd International Conference on Machine Learning (ICML)*, S. 1613–1622, 3045290. JMLR.org. *(Seite: 22, 23, 24)*

[Breiman, 1996] Breiman, Leo (1996). *Bagging Predictors*. Machine learning, 24(2):123–140. *(Seite: 29)*

[Chapelle et al., 2006] Chapelle, Olivier, B. Schölkopf und A. Zien (2006). *Semi-Supervised Learning*. MIT Press. *(Seite: 7, 34)*

[Chen et al., 2016] Chen, Changyou, D. Carlson, Z. Gan, C. Li und L. Carin (2016). *Bridging the Gap between Stochastic Gradient MCMC and Stochastic Optimization*. In: *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. *(Seite: 22)*

[Chetlur et al., 2014] Chetlur, Sharan, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro und E. Shelhamer (2014). *cuDNN: Efficient Primitives for Deep Learning*. arXiv preprint arXiv:1410.0759. *(Seite: 24)*

[Deng et al., 2009] Deng, Jia, W. Dong, R. Socher, L.-J. Li, K. Li und L. Fei-Fei (2009). *ImageNet: A Large-Ccale Hierarchical Image Database*. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 248–255. IEEE. *(Seite: 13)*

[Eisenbach et al., 2016] Eisenbach, Markus, D. Seichter, T. Wengefeld und H.-M. Gross (2016). *Cooperative Multi-Scale Convolutional Neural Networks for Person Detection*. In: *Proc. of IEEE World Congress on Computational Intelligence (WCCI)*, S. 267–276. IEEE. *(Seite: 42, 59, 85)*

[Eisenbach et al., 2017] Eisenbach, Markus, R. Stricker, D. Seichter, K. Amende, K. Debes, M. Sesselmann, D. Ebersbach, U. Stoeckert und H.-M. Gross (2017). *How to Get Pavement Distress Detection Ready for Deep Learning? A Systematic Approach*. In: *Proc. of Int. Joint Conference on Neural Networks (IJCNN)*, S. 2039–2047. IEEE. *(Seite: 3, 4, 37, 40, 41, 59, 76, 78, 79, 94, 95, 126)*

[Fukushima, 1980] Fukushima, Kunihiko (1980). *Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position*. Biological cybernetics (BC), 36(4):193–202. *(Seite: 9)*

[Gal, 2016] Gal, Yarin (2016). *Uncertainty in Deep Learning*. Doktorarbeit. University of Cambridge. *(Seite: 15, 16, 18, 25, 26, 79, 80, 103)*

[Gal und Ghahramani, 2015a] Gal, Yarin und Z. Ghahramani (2015a). *Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference*. arXiv preprint arXiv:1506.02158. *(Seite: 27)*

[Gal und Ghahramani, 2015b] Gal, Yarin und Z. Ghahramani (2015b). *Dropout as a Bayesian Approximation: Appendix*. arXiv preprint arXiv:1506.02157. *(Seite: 25, 26)*

[Gal und Ghahramani, 2015c] Gal, Yarin und Z. Ghahramani (2015c). *Dropout as a Bayesian Approximation: Insights and Applications*. Deep Learning Workshop (ICML). *(Seite: 25, 26)*

[Gal und Ghahramani, 2015d] Gal, Yarin und Z. Ghahramani (2015d). *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. arXiv preprint arXiv:1506.02142, 2. *(Seite: 25)*

[Gal et al., 2016] Gal, Yarin, R. Islam und Z. Ghahramani (2016). *Deep Bayesian Active Learning with Image Data*. Workshop on Bayesian Deep Learning, NIPS 2016. *(Seite: 30, 31, 32, 33, 101, 104)*

[Gelfand und Smith, 1990] Gelfand, Alan E und A. F. Smith (1990). *Sampling-Based Approaches to Calculating Marginal Densities*. Journal of the American Statistical Association (JASA), 85(410):398–409. *(Seite: 16)*

[Geng und Smith-Miles, 2009] Geng, Xin und K. Smith-Miles (2009). *Incremental Learning*, S. 731–735. Springer US, Boston, MA. *(Seite: 7)*

[Goodfellow et al., 2013] Goodfellow, Ian J, M. Mirza, D. Xiao, A. Courville und Y. Bengio (2013). *An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks*. arXiv preprint arXiv:1312.6211. *(Seite: 8)*

[Goodfellow et al., 2015] Goodfellow, Ian J, J. Shlens und C. Szegedy (2015). *Explaining and Harnessing Adversarial Examples*. In: *Proc. of International Conference on Learning Representations (ICLR)*. *(Seite: 13, 14)*

[Graves, 2011] Graves, Alex (2011). *Practical Variational Inference for Neural Networks*. In: *Advances in Neural Information Processing Systems (NIPS)*, S. 2348–2356. *(Seite: 22)*

[Gross, 2016] Gross, H.-M. (2016). *Neuroinformatik*. Vorlesungsskript Sommersemester 2016. Fachgebiet Neuroinformatik & Kognitive Robotik, TU Ilmenau. *(Seite: 6)*

[Hasan und Roy-Chowdhury, 2014] Hasan, Mahmudul und A. K. Roy-Chowdhury (2014). *Continuous Learning of Human Activity Models using Deep Nets*. In: *Proc. of European Conference on Computer Vision (ECCV)*, S. 705–720. Springer. *(Seite: 8, 33)*

[Hastings, 1970] Hastings, W Keith (1970). *Monte Carlo Sampling Methods using Markov Chains and Their Applications*. Biometrika, 57(1):97–109. *(Seite: 16)*

[Hinton et al., 2012] Hinton, Geoffrey E, N. Srivastava, A. Krizhevsky, I. Sutskever und R. R. Salakhutdinov (2012). *Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors*. arXiv preprint arXiv:1207.0580. *(Seite: 9)*

[Holub et al., 2008] Holub, Alex, P. Perona und M. C. Burl (2008). *Entropy-Based Active Learning for Object Recognition*. In: *Proc. of Computer Vision and Pattern Recognition Workshop (CVPRW)*, S. 1–8. IEEE. *(Seite: 33)*

[Ioffe und Szegedy, 2015] Ioffe, Sergey und C. Szegedy (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. In: *Proc. of International Conference on Machine Learning (ICML)*, S. 448–456. *(Seite: 37)*

[Islam, 2016] Islam, Riashat (2016). *Active Learning for High Dimensional Inputs using Bayesian Convolutional Neural Networks*. Masterarbeit. University of Cambridge. *(Seite: 30, 33)*

[Jordan et al., 1999] Jordan, Michael I, Z. Ghahramani, T. S. Jaakkola und L. K. Saul (1999). *An Introduction to Variational Methods for Graphical Models*. Machine learning (ML), 37(2):183–233. *(Seite: 17)*

[Joshi et al., 2009] Joshi, Ajay J, F. Porikli und N. Papanikolopoulos (2009). *Multi-Class Active Learning for Image Classification*. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 2372–2379. IEEE. *(Seite: 33)*

[Kendall et al., 2015] Kendall, Alex, V. Badrinarayanan und R. Cipolla (2015). *Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding*. arXiv preprint arXiv:1511.02680. *(Seite: 27)*

[Kendall und Cipolla, 2016] Kendall, Alex und R. Cipolla (2016). *Modelling Uncertainty in Deep Learning for Camera Relocalization*. In: *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, S. 4762–4769. *(Seite: 27)*

[Kingma et al., 2015] Kingma, Diederik P, T. Salimans und M. Welling (2015). *Variational Dropout and the Local Reparameterization Trick*. In: *Advances in Neural Information Processing Systems (NIPS)*, S. 2575–2583. *(Seite: 24, 25)*

[Kirkpatrick et al., 2016] Kirkpatrick, James, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho und A. Grabska-Barwinska (2016). *Overcoming Catastrophic Forgetting in Neural Networks*. arXiv preprint arXiv:1612.00796. *(Seite: 8)*

[Krogh und Hertz, 1992] Krogh, Anders und J. A. Hertz (1992). *A Simple Weight Decay Can Improve Generalization*. In: *Advances in neural information processing systems (NIPS)*, S. 950–957. Morgan-Kaufmann. *(Seite: 79)*

[Käding et al., 2016] Käding, Christoph, E. Rodner, A. Freytag und J. Denzler (2016). *Fine-tuning Deep Neural Networks in Continuous Learning Scenarios*. In: *ACCV Workshop on Interpretation and Visualization of Deep Neural Nets (ACCV-WS)*. *(Seite: 33, 37)*

[Lakshminarayanan et al., 2016] Lakshminarayanan, Balaji, A. Pritzel und C. Blundell (2016). *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*. arXiv preprint arXiv:1612.01474. *(Seite: 27, 28)*

[LeCun et al., 1989] LeCun, Yann, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard und L. D. Jackel (1989). *Backpropagation Applied*

to Handwritten Zip Code Recognition. Neural Computation (NC), 1(4):541–551. *(Seite: 9)*

[Leibig et al., 2016] Leibig, Christian, V. Allken, P. Berens und S. Wahl (2016). *Leveraging Uncertainty Information from Deep Neural Networks for Disease Detection*. bioRxiv, S. 084210. *(Seite: 27)*

[Li et al., 2016] Li, Chunyuan, C. Chen, D. Carlson und L. Carin (2016). *Preconditioned Stochastic Gradient Langevin dynamics for Deep Neural Networks*. In: *Conference on Artificial Intelligence (AAAI)*. *(Seite: 22)*

[Li und Guo, 2013] Li, Xin und Y. Guo (2013). *Adaptive Active Learning for Image Classification*. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 859–866. *(Seite: 33)*

[Li und Gal, 2017] Li, Yingzhen und Y. Gal (2017). *Dropout Inference in Bayesian Neural Networks with Alpha-divergences*. arXiv preprint arXiv:1703.02914. *(Seite: 127)*

[Lin et al., 2013] Lin, Min, Q. Chen und S. Yan (2013). *Network In Network*. arXiv preprint arXiv:1312.4400. *(Seite: 86)*

[Lin et al., 2014] Lin, Tsung-Yi, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick und P. Dollár (2014). *Microsoft COCO: Common Objects in Context*. arXiv preprint arXiv:1405.0312. *(Seite: 33)*

[MacKay, 1992] MacKay, David JC (1992). *A Practical Bayesian Framework for Backpropagation Networks*. Neural Computation (NC), 4(3):448–472. *(Seite: 15)*

[McClure und Kriegeskorte, 2017] McClure, Patrick und N. Kriegeskorte (2017). *Representing Inferential Uncertainty in Deep Neural Networks through Sampling*. arXiv preprint arXiv:1611.01639. *(Seite: 11)*

[Neal, 2011] Neal, Radford M (2011). *MCMC using Hamiltonian dynamics*. Handbook of Markov Chain Monte Carlo, 2:113–162. *(Seite: 21)*

[Neal, 2012] Neal, Radford M (2012). *Bayesian Learning for Neural Networks*, Bd. 118. Springer Science & Business Media. *(Seite: 15)*

[Ng, 2016] Ng, Andrew (2016). *Nuts and Bolts of Applying Deep Learning*. Vortrag Deep Learning School 2016: `https://www.bayareadlschool.org/`. *(Seite: 42, 120)*

[Oliveira et al., 2016] Oliveira, Ramon, P. Tabacof und E. Valle (2016). *Known Unknowns: Uncertainty Quality in Bayesian Neural Networks*. Workshop on Bayesian Deep Learning, NIPS 2016. *(Seite: 24)*

[Osband, 2016] Osband, Ian (2016). *Risk versus Uncertainty in Deep Learning: Bayes, Bootstrap and the Dangers of Dropout*. Workshop on Bayesian Deep Learning, NIPS 2016. *(Seite: 27)*

[Osband et al., 2016] Osband, Ian, C. Blundell, A. Pritzel und B. Van Roy (2016). *Deep Exploration via Bootstrapped DQN*. In: *Advances In Neural Information Processing Systems (NIPS)*, S. 4026–4034. *(Seite: 28, 29, 80, 81, 82, 98)*

[Pezeshki et al., 2016] Pezeshki, Mohammad, L. Fan, P. Brakel, A. Courville und Y. Bengio (2016). *Deconstructing the Ladder Network Architecture*. In: *Proc. of International Conference on Machine Learning (ICML)*, S. 2368–2376. *(Seite: 36)*

[Rasmus et al., 2015a] Rasmus, Antti, M. Berglund, M. Honkala, H. Valpola und T. Raiko (2015a). *Semi-Supervised Learning with Ladder Networks*. In: *Advances in Neural Information Processing Systems (NIPS)*, S. 3546–3554. *(Seite: 34, 35)*

[Rasmus et al., 2015b] Rasmus, Antti, H. Valpola, M. Honkala, M. Berglund und T. Raiko (2015b). *Semi-Supervised Learning with Ladder Networks*. arXiv preprint arXiv:1507.02672. *(Seite: 34)*

[Seichter, 2015a] Seichter, Daniel (2015a). *Deep Neural Network for Detecting Unintended Single or Double AAC Encoding*. Bachelorarbeit. Fraunhofer IDMT und Fachgebiet Neuroinformatik & Kognitive Robotik, TU Ilmenau. *(Seite: 9)*

[Seichter, 2015b] Seichter, Daniel (2015b). *Implementation of a (Deep) Neural Network Architecture for Audio Double-Encoding and Transcoding Detection*. Praktikumsbericht. Fraunhofer IDMT und Fachgebiet Neuroinformatik & Kognitive Robotik, TU Ilmenau. *(Seite: 6, 34)*

[Seichter, 2016] Seichter, Daniel (2016). *Warum Deep Learning funktioniert - Aufarbeitung aktueller Deep-Learning-Techniken*. Hauptseminar. Fachgebiet Neuroinformatik & Kognitive Robotik, TU Ilmenau. *(Seite: 9)*

[Settles, 2010] Settles, Burr (2010). *Active learning Literature Survey*. University of Wisconsin, Madison, 52(55-66):11. *(Seite: 8)*

[Srivastava et al., 2014] Srivastava, Nitish, G. Hinton, A. Krizhevsky, I. Sutskever und R. Salakhutdinov (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. Journal of Machine Learning Research (JMLR), 15:1929–1958. *(Seite: 9, 10, 11, 23, 24, 80, 103)*

[Szegedy et al., 2015] Szegedy, Christian, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke und A. Rabinovich (2015). *Going Deeper with Convolutions*. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 1–9. *(Seite: 13)*

[Szegedy et al., 2014] Szegedy, Christian, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow und R. Fergus (2014). *Intriguing Properties of Neural Networks*. In: *International Conference on Learning Representations (ICLR)*. *(Seite: 13, 21)*

[Tieleman und Hinton, 2012] Tieleman, Tijmen und G. Hinton (2012). *Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of its Recent Magnitude*. COURSERA: Neural networks for machine learning, 4(2). *(Seite: 22)*

[Valpola, 2015] Valpola, Harri (2015). *From Neural PCA to Deep Unsupervised Learning*. Advances in Independent Component Analysis and Learning Machines, S. 143–171. *(Seite: 34)*

[Vincent et al., 2010] Vincent, Pascal, H. Larochelle, I. Lajoie, Y. Bengio und P.-A. Manzagol (2010). *Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion*. Journal of Machine Learning Research (JMLR), 11:3371–3408. *(Seite: 34)*

[Wan et al., 2013] Wan, Li, M. Zeiler, S. Zhang, Y. L. Cun und R. Fergus (2013). *Regularization of Neural Networks using DropConnect*. In: *Proc. of the 30th International Conference on Machine Learning (ICML-13)*, S. 1058–1066. *(Seite: 11)*

[Welling und Teh, 2011] Welling, Max und Y. W. Teh (2011). *Bayesian learning via Stochastic Gradient Langevin Dynamics*. In: *Proc. of International Conference on Machine Learning (ICML)*, S. 681–688. *(Seite: 21, 22)*