Projektarbeit: Semantische Segmentierung von Unterwasserbilddaten in Echtzeit mit PID-Net

Prüfungsleistung für das Modul „Deep Learning für Computer Vision“

im Wintersemester 2022/23

des Master-Studiengangs

Informatik

an der Technischen Hochschule Mittelhessen

28. Januar 2023

Bearbeitet von:

Ben Riegel 5178979

Fabian Schwickert 5455436

Gutachter der Studienakademie Prof. Dr.-Ing. Seyed Eghbal Ghobadi

Eigenständigkeitserklärung

Wir versichern hiermit, dass wir die Projektarbeit mit dem Thema: „Projektarbeit: Se­man­ti­sche Segmentierung von Unterwasserbilddaten in Echtzeit mit PID-Net“ selbstständig ver­fasst und keine anderen als die angegebenen Quellen und Hilfs­­mit­tel benutzt haben.

Ben Riegel

Gießen, 28. Januar 2023

Ort, Datum Unterschrift

Fabian Schwickert

Gießen, 28. Januar 2023

Ort, Datum Unterschrift

Inhaltsverzeichnis

[Eigenständigkeitserklärung II](#_Toc125815137)

[Inhaltsverzeichnis III](#_Toc125815138)

[Abbildungsverzeichnis IV](#_Toc125815139)

[Tabellenverzeichnis V](#_Toc125815140)

[Abkürzungsverzeichnis VI](#_Toc125815141)

[1 Problemstellung, Ziel und Aufbau 1](#_Toc125815142)

[2 Semantische Segmentierung von Bilddaten 3](#_Toc125815143)

[3 Untersuchungsobjekt 7](#_Toc125815144)

[3.1 PID-Regler 7](#_Toc125815145)

[3.2 PID-Net 9](#_Toc125815146)

[4 Untersuchungsumgebung 19](#_Toc125815147)

[4.1 Datensätze für Semantische Segmentierung 19](#_Toc125815148)

[4.2 Der SUIM-Datensatz 20](#_Toc125815149)

[4.3 Data Augmentation 23](#_Toc125815150)

[5 Untersuchungsergebnisse 25](#_Toc125815151)

[Literaturverzeichnis VII](#_Toc125815152)

Abbildungsverzeichnis

[Abb. 1: Drei Varianten der Segmentierung 6](#_Toc125814679)

[Abb. 2: Die PID-Net Architektur 10](#_Toc125814680)

[Abb. 3: Overshooting bei PI-Reglern (links) und PI-Nets (rechts) 11](#_Toc125814681)

[Abb. 4: Ein residueller Standard-Faltungsblock 12](#_Toc125814682)

[Abb. 5: Der Aufbau der Pag-Komponente 14](#_Toc125814683)

[Abb. 6: Der Aufbau der PAPPM-Komponente 15](#_Toc125814684)

[Abb. 7: Der Aufbau der Bag- (a) bzw. Light-Bag-Komponente (b) 17](#_Toc125814685)

[Abb. 8: Der Aufbau der S- und B-Head-Komponente 17](#_Toc125814686)

[Abb. 9: Bilder (obere Zeile) und Labels (untere Zeile) aus dem SUIM-Datensatz 21](#_Toc125814687)

[Abb. 10: Auftreten der Klassen im SUIM-Trainingsdatensatz 23](#_Toc125814688)

[Abb. 11: Drei von acht verwendeten Data-Augmentierungs-Methoden 24](#_Toc125814689)

[Abb. 12: Bewertung der PID-Net Architektur mit mIoU und Vorhersagezeit 26](#_Toc125814690)

[Abb. 13: Ausgaben des PID-Net-S Modells nach 38 Trainingsepochen 28](#_Toc125814691)

[Abb. 14: Overfitting auf Klassen 29](#_Toc125814692)

Tabellenverzeichnis

[Tab. 1: Abbildung der RGB-Label-Werte auf Klassen-Label-Werte 22](#_Toc125814693)

[Tab. 2: Trainingsergebnisse des PID-Net-S Modells auf dem SUIM-Datensatz 27](#_Toc125814694)

Abkürzungsverzeichnis

Add Addition

Bag Boundary-attention-guided fusion module

BAS Boundary Awareness

BCE Binary Cross Entropy

B-Head Boundary-Head

BN Batch-Normalisierung

CE Cross Entropy

CNN Convolutional Neural Networks

Conv Convolution

DAPPM Deep Aggregation PPM

fps frames per second

IoU Intersection over Union

KI Künstliche Intelligenz

KNN Künstliches Neuronales Netz

mIoU Mean Intersection over Union

ms Millisekunden

Pag Pixel-attention-guided fusion module

PID Proportional-Integral-Differential

PPM Pyramid Pooling Module

RGB Rot-Grün-Blau

SGD Stochastischer Gradienten Abstieg

S-Head Semantic-Head

SUIM Semantic Segmentation of Underwater Imagery

SUM Summe

Up Upsampling

# Problemstellung, Ziel und Aufbau

Computer spielen in der heutigen Gesellschaft eine große Rolle. Im privaten, ge­sell­schaft­li­­­chen und betrieblichen Umfeld werden Computer eingesetzt, um Aufgaben zu au­to­ma­ti­sie­­ren und dem Menschen Arbeit abzunehmen. In den letzten Jahren ist besonders die „Künst­­­­liche Intelligenz“ (KI) in den Vordergrund gerückt. Denn mit KI-Software sind Com­pu­­­ter dazu in der Lage, Aufgaben wie z. B. das Erkennen von Objekten in Bil­dern bes­ser als der Mensch zu erledigen. Ohne KI-Software sind Computer nur schwerlich da­zu fä­hig.[[1]](#footnote-2)

An­fang 2023 kann auch KI-Software noch nicht alle Probleme perfekt lösen, die z. B. beim au­­­­­to­no­­men Fahren anfallen. Für diese Probleme wird noch leistungsstärkere KI-Software be­­­­­nö­­­­tigt. Besonders für die Anwendung im autonomen Fahren ist es zudem ent­schei­dend, dass die KI-Software zuverlässig die richtigen Ergebnisse in Echtzeit liefert. Denn wenn die KI-Soft­­­ware Gefahren nicht schnell genug erkennt, kann es zu Unfällen kommen.[[2]](#footnote-3)

Ty­­pi­scher­wei­se ist KI-Software, die in Echtzeit agieren soll, weniger komplex als KI-Soft­ware, die kei­ner solchen Zeitbeschränkung unterliegt. Denn je komplexer die KI-Software ist, desto mehr Berechnungen müssen für die Verarbeitung von Daten durchgeführt wer­den und desto länger benötigt die KI-Software für den Verarbeitungsprozess. Dadurch ist Echt­­­zeit-KI-Software meist weniger zuverlässig und ungenauer als andere KI-Soft­ware. Denn je komplexer die KI-Software ist, desto mehr variable Parameter besitzt die KI-Soft­ware und desto genauer sind ihre Berechnungen. Für An­wen­dungs­­felder wie z. B. das au­to­nome Fahren muss ein Kompromiss zwischen Zeiteffizienz und Ef­­fektivität ge­fun­­den wer­den.[[3]](#footnote-4)

Ein besonders für das autonome Fahren relevanter Aufgabentyp ist die „Segmentierung“ von Bildern bzw. Videos. Bei einer Segmentierungsaufgabe sollen die in den Bilddaten ent­­­­haltenen Objekte erkannt und klassifiziert werden. Beim autonomen Fahren wird da­durch z. B. der Gegenverkehr, die Straße und auch Fußgänger erkannt.[[4]](#footnote-5)

Um bei derartig komplexen Aufgaben einen Kompromiss zwischen Zeiteffizienz und Ef­­fek­­­­­tivität zu finden, wurde in den letzten Jah­ren verschiedene neuartige KI-Software ent­wi­­­­­ckelt. Eines dieser KI-Softwaresysteme ist „PID-Net“. PID-Net ist ein „Künstliches Neu­ro­­­­nales Netz“ (KNN), das Ende 2022 veröffentlicht wurde und Anfang 2023 zu den zeit­ef­fi­­­zientesten und effektivsten KNN-Modellen für die Semantische Segmentierung von Bild­da­­­ten in Echtzeit zählt. Dabei ist der besagte Kompromiss zwischen Zeiteffizienz und Ef­­fek­­­­tivität mit PID-Net besonders gut gelungen.[[5]](#footnote-6)

Das Ziel der vorliegenden Projektarbeit ist es, die Forschungsergebnisse zu dem Anfang 2023 noch jungen PID-Net-Modell zu bestätigen. Dafür wird das PID-Net-Modell auf ei­nem Datensatz für die Semantische Segmentierung trainiert, der nicht von den Erfindern der PID-Net-Architektur ver­wendet wur­de. Dabei wird die Zeiteffizienz und die Ef­­fek­­ti­vi­tät des PID-Net-Modells erhoben. Die eigens erhobenen Daten werden mit den von den Er­findern der PID-Net-Architektur erhobenen Daten verglichen.

Für das Erreichen des besagten Ziels ist die vorliegenden Projektarbeit wie folgt ge­glie­dert. In Kapitel 2 der vorliegenden Projektarbeit wird zunächst genauer auf die Pro­blem­stel­­­lung eingegangen, die von PID-Net gelöst werden soll. Dabei wird genauer be­schrie­ben, was Segmentierung ist, welche Arten der Segmentierung es gibt und wie dieser Auf­ga­­bentyp sich von anderen unterscheidet.

In Kapitel 3 wird das Untersuchungsobjekt der vorliegenden Projektarbeit beschrieben. Es wird die Idee hinter der PID-Net-Architektur (Unterkapitel 3.1) und ihre Umsetzung (Un­terkapitel 3.2) erläutert.

In Kapitel 4 wird der Untersuchungsbereich festgelegt, in dem das Untersuchungsobjekt be­­trachtet wird. Es wird geklärt, was einen Datensatz für die Semantische Seg­men­tie­rungs-Aufgaben aus­­macht (Unterkapitel 4.1), welcher Datensatz konkret für die vor­lie­gen­de Pro­jektarbeit verwendet (Unterkapitel 4.2) und wie dieser augmentiert wird (Un­ter­kapitel 4.3).

Im Abschließenden Kapitel 5 wird kurz beschrieben, wie die Untersuchung durchgeführt wur­­de. Anschließend werden ausführlich die Ergebnisse der Untersuchung vorgestellt.

# Semantische Segmentierung von Bilddaten

In dem Forschungsfeld der KI gibt es eine Reihe von Aufgabentypen. Die be­­kanntesten die­­­­ser Aufgabentypen sind die Assoziation, die Klassifizierung, die Re­gression und die Seg­mentierung. Nachfolgend werden diese vier Aufgabentypen kurz beschrieben.[[6]](#footnote-7)

1. Assoziation

Als Assoziation wird das Herstellen einer Korrelation zwischen Objekten bezeichnet. Da­durch können Paare oder Gruppen von Objekten gebildet werden, die eine semantische Be­­­­ziehung zueinander haben.[[7]](#footnote-8) Beispielsweise könnten Artikel gruppiert werden, die häu­fig zusammen gekauft werden.

1. Klassifizierung

Bei einer Klassifizierungs-Aufgabe werden z. B. Bilder aus einer Gesamtmenge in klar dif­fe­­­ren­­zierte Teilmengen aufgeteilt. Dabei werden nicht mehrere Objekte in einem Bild se­pa­­rat erkannt, sondern das Bilder als Ganzes klassifiziert. Ein Beispiel für eine Klas­si­fi­zie­rungs­­­aufgabe ist die Einteilung von Kunden in Kre­dit­würdigkeits-Klassen.[[8]](#footnote-9)

1. Regression

Mittels Regression kann aus Daten der Vergangenheit zukünftiges Verhalten prog­nos­ti­ziert werden. Dafür modellieren KI-Systeme einen funktionalen Zusammenhang. Bei­spiel­­haft für solche Prognosen ist das „Predictive Maintenance“, bei dem vorausberechnet wird, wann eine Maschine gewartet werden muss.[[9]](#footnote-10)

1. Segmentierung

Bei einer Segmentierungs-Aufgabe werden z. B. Bilder aus einer Gesamtmenge in klar dif­fe­­­­­ren­­zierte Teilmengen aufgeteilt. Allerdings werden im Gegensatz zur Klassifizierung nicht die Bilder als Ganzes klassifiziert. Stattdessen werden Bilder in Segmente aufgeteilt und jedes dieser Segmente einer Klasse zugeordnet. Segmentierung wird z. B. häufig beim au­­­­tonomen Fahren eingesetzt, um relevante Objekte wie z. B. Verkehrsschilder, die Straße oder andere Verkehrsteilnehmer im Verkehr zu erkennen.[[10]](#footnote-11)

Die vorliegende Projektarbeit befasst sich mit einer Segmentierungsaufgabe. Daher wird nach­­­­­folgend nur noch dieser Aufgabentyp betrachtet. Segmentierungs-Aufgaben treten in ver­­schiedenen Anwendungsfeldern auf. Besonders viel Aufmerksamkeit erhält dabei die Bil­d­­­er­ken­nung. In diesem Anwendungsfeld werden KI-Systeme ein­­­gesetzt, um den Inhalt von Bildern oder Videos zu bestimmen. In der vorliegenden Pro­­jekt­ar­beit wird aus­schließ­­lich das Anwendungsgebiet der Bildverarbeitung be­trach­tet.

Seg­men­tie­rungs-Aufgaben im Anwendungsfeld der Bilderkennung werden in drei Un­ter­ty­­pen aufgeteilt: Die „Semantische Segmentierung“, die „In­­stanz Segmentierung“ und die „Pa­­­noptische Seg­men­tie­rung“.[[11]](#footnote-12) Nachfolgend werden diese drei Varianten be­schrie­ben. In Abbildung 1 werden die Unterschiede zwischen den drei Varianten her­vor­ge­ho­ben. Da­­­bei werden die Klassen, denen Objekte zugeordnet werden durch verschiedene Farben re­­­präsentiert.

1. Semantische Segmentierung

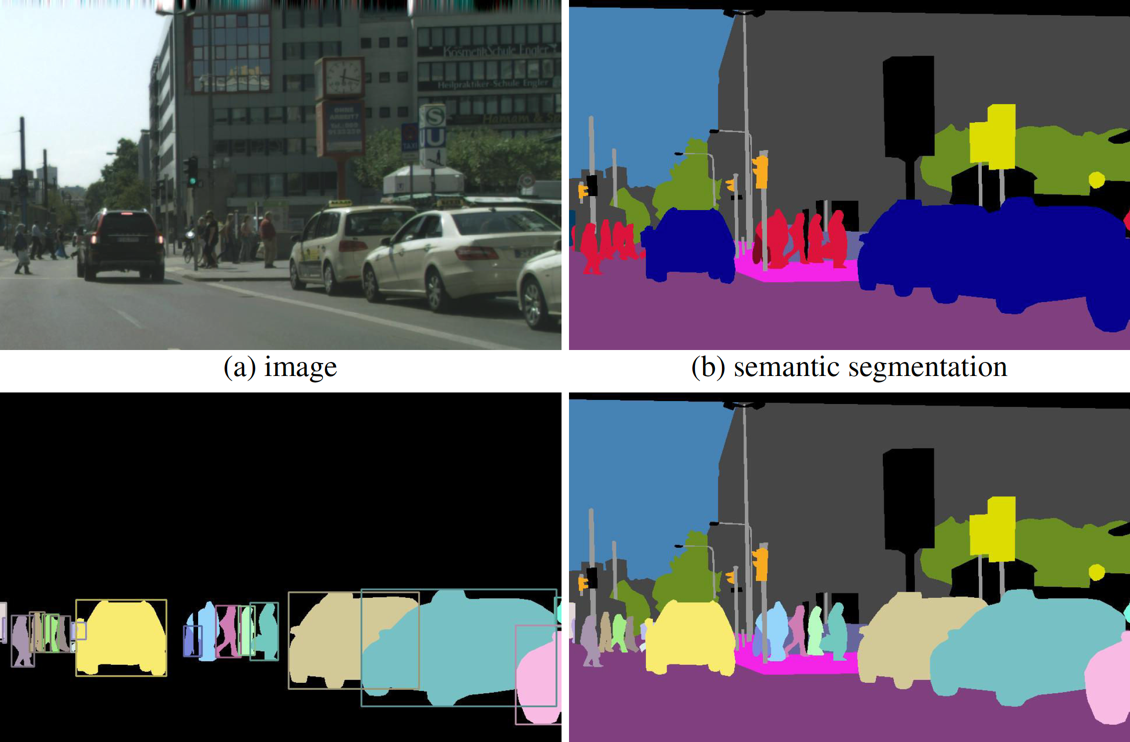
Bei der Semantischen Segmentierung wird das gesamte Bild in Segmente unterteilt. Diese Seg­­­­mente werden ausschließlich nach der semantischen Zugehörigkeit zu einer Klasse klas­­sifiziert. Es wird nicht zwischen zwei Segmenten mit der gleichen Klas­sen­zu­ge­hö­rig­keit unterschieden.[[12]](#footnote-13) So würden z. B. zwei überlappend dargestellte Menschen (z. B. hin­ter­­­ei­nanderstehend) nicht durch Se­man­tische Segmentierung unterschieden werden (sie­­­he Abbildung 1B).

1. Instanz Segmentierung

Im Unterschied zur Semantische Segmentierung, werden bei einer Instanz Segmentierung le­­­diglich zählbare Objekte in einem Bild erkannt. Diese Segmente werden nach der se­man­­­­ti­schen Zugehörigkeit zu einer Klasse und der abgebildeten Instanz der Klasse klas­si­­­fiziert. Es wird im Unterschied zur Semantischen Segmentierung zwischen zwei Seg­men­­­ten mit der gleichen Klassenzugehörigkeit unterschieden.[[13]](#footnote-14) So würden z. B. zwei über­­­­lap­pend dar­ge­stellte Menschen (z. B. hintereinanderstehend) durch Instanz Seg­men­tie­­­­rung un­ter­schie­den werden (siehe Abbildung 1C).

1. Panoptische Segmentierung

Die Panoptische Segmentierung vereint die Semantische und die Instanz Segmentierung. Bei der Panoptischen Segmentierung wird das gesamte Bild in Segmente aufgeteilt. Alle Seg­­­­mente werden nach der se­man­­ti­schen Zugehörigkeit zu einer Klasse klassifiziert. Seg­men­te, die zählbare Objekte repräsentieren, werden zusätzlich nach der abgebildeten In­stanz der Klasse klas­si­fiziert (siehe Abbildung 1D).[[14]](#footnote-15)



(A) Original Bild

(B) Semantische Segmentierung

(C) Instanz Segmentierung

(D) Panoptische Segmentierung

Abb. 1: Drei Varianten der Segmentierung[[15]](#footnote-16)

In der vorliegenden Projektarbeit wird ei­ne Semantische Segmentierungs-Aufgabe be­trach­­tet. Daher wird nachfolgend nur noch die­se Variante betrachtet.

# Untersuchungsobjekt

## PID-Regler

Die PID-Net-Architektur orientiert sich an dem Aufbau eines „Proportional-Integral-Dif­fe­­rential-Reglers“ (PID-Regler). Das Konzept des PID-Reglers stammt aus der Re­ge­lungs­tech­­nik. Die Aufgabe eines Reglers ist es, den Wert eine „Regelgröße“ möglichst auf den Wert einer „Führungsgröße“ zu bringen. Dieser angestrebte Zustand soll von dem PID-Reg­­ler auch bei sich ändernden Umgebungsbedingungen beibehalten werden. Dafür muss die Regelgröße wiederholt ge­messen und mit der Führungsgröße verglichen wer­den. Die Dif­­ferenz zwischen der Re­gel- und der Führungsgröße wird dann durch die Ver­än­derung des Werts einer „Stell­grö­ße“ iterativ reduziert.[[16]](#footnote-17)

Das Verhalten eines PID-Reglers kann anhand einer Drohne, die auf einer bestimmten Hö­he schweben soll, verdeutlicht werden. In diesem Beispiel ist die Regelgröße die Flughöhe der Drohne; die Führungsgröße ist die Ziel-Flughöhe, auf der die Drohne schweben soll und die Stellgröße ist die Rotationsgeschwindigkeit der Rotoren, die die Drohne bewegen. Da­mit die Drohne konstant die Ziel-Flughöhe hält, muss die Rotationsgeschwindigkeit der Ro­toren wiederholt entsprechend der Differenz zwischen aktueller und Ziel-Flughöhe an­ge­­passt werden. Eine solche Anpassung ist besonders bei äußeren Einflüssen, wie z. B. ei­­nem Windstoß notwendig.

Die Besonderheit eines PID-Reglers ist, dass er drei einfacheren Regler-Komponenten ver­­­eint und da­durch die Nachteile der einzelnen Komponenten kompensiert. Ein PID-Reg­­­ler besteht aus einer P-, einer I- und einer D-Komponente. Die P-Komponente bildet den proportionalen Zusammenhang zwischen der Regelgröße und der Stellgröße ab und dient der Reaktion auf aktuelle Änderungen der Regelgröße. An­hand des Drohnen-Bei­spiels würde sich die Propellergeschwindigkeit pro­­­por­­tional zu der Differenz der ak­tu­el­len und der Ziel-Flughöhe verändern.[[17]](#footnote-18)

Das führt al­­lerdings zu dem Problem, dass die Propellergeschwindigkeit mit zu­neh­men­der An­nä­he­rung an die Ziel-Flughöhe immer geringer wird. Auf diese Weise würde die Droh­­­ne ab dem Punkt nicht mehr steigen, an dem die durch die Ro­toren erzeugte Auf­triebs­kraft genau der Anziehungskraft der Erde entspricht. Die dafür not­wen­dige Ro­ta­tions­geschwindigkeit kann allerdings auf Grund des pro­por­tio­na­len Zu­sam­men­hangs mit der Differenz zwischen der aktuellen und der Ziel-Flughöhe von der P-Komponente nie auf der Ziel-Flughöhe erreicht werden. Die Drohne würde unter der Ziel-Flughöhe schwe­ben und konstant eine nicht reduzierbare Abweichung aufweisen. Dies wird als „Steady-State-Fehler“ be­zeich­net.[[18]](#footnote-19)

Der Steady-State-Fehler der P-Komponente wird von der I-Komponente eines PID-Reg­lers durch Einbeziehen von vergangenen Daten behoben. Die I-Komponente bil­det das In­te­­­­gral der vergangenen Differenzen zwi­schen gegenwärtiger und Ziel-Flughöhe. Das In­te­­gral der konstanten Differenz zwi­schen aktueller und Ziel-Flughöhe ist eine lineare Funk­­tion. Die I-Komponente verändert damit die Ro­ta­tions­ge­schwin­digkeit bei einem Steady-State-Fehler immer weiter, bis die Dif­fe­renz zwi­­­schen aktueller und Ziel-Flughöhe den Wert 0 er­reicht.[[19]](#footnote-20)

Das Problem der I-Komponente ist, dass es bei einem Steady-State-Fehler sehr nahe der Ziel-Flughöhe zu einem „Overshooting“ kommen kann. Dabei erhöht die I-Komponente die Rotationsgeschwindigkeit zu stark, sodass die Drohne über die Ziel-Flughöhe fliegt und die Differenz aus aktueller und Ziel-Flughöhe negativ wird. Dadurch wird die Ro­ta­tions­­geschwindigkeit reduziert und die aktuelle Flughöhe fällt wieder unter das Ziel. Auf die­­se Weise kann es zu einem Pendeln um die Ziel-Flughöhe kommen.[[20]](#footnote-21)

Um das Overshooting der I-Komponente zu verhindern, besitzt der PID-Regler eine D-Kom­­ponente. Die D-Komponente berechnet die Ableitung der Differenz aus aktueller und Ziel-Flughöhe, um den zukünftigen Wert dieser Differenz zu erraten. Da diese Differenz zu­­nehmend kleiner wird, hat das Ergebnis der D-Komponente immer eine negative Stei­gung, die die Veränderung der Rotationsgeschwindigkeit reduziert. Auf diese Weise wird bei einer zu schnellen Annäherung an die Ziel-Flughöhe und ein drohendes Overshooting die Änderung der Rotationsgeschwindigkeit gedrosselt.[[21]](#footnote-22)

## PID-Net

Das Untersuchungsobjekt der vorliegenden Projektarbeit ist die PID-Net-Architektur für KNNs. Diese Architektur wurde speziell für die Semantische Segmentierung von Bild­da­ten in Echtzeit entwickelt. Die PID-Net-Architektur verwendet vornehmlich Faltungs-Schich­­­ten, um die semantisch stark verwobenen Bilddaten effizient und effektiv zu ver­ar­bei­­ten. Die PID-Net-Architektur gehört daher zu den „Convolutional Neural Networks“ (CNN, zu Deutsch „Neuronale Faltungsnetze“).[[22]](#footnote-23)

Die PID-Net-Architektur spaltet ihre Datenverarbeitung entsprechend dem Konzept eines PID-Reglers in drei parallel verlaufende „Bran­ches“ (zu Deutsch „Zweige“) ein. Jeder die­ser Bran­ches hat eine eigene Aufgabe, die für eine korrekte Semantische Segmentierung er­­le­digt werden muss. Die Ergebnisse der drei Branches werden am Ende wieder zu­sam­men­­geführt, um ein gemeinsames Ergebnis zu erzeugen.[[23]](#footnote-24) Abbildung 2 zeigt die Struktur der PID-Net Architektur.

Der P-Branch dient der Erkennung von Detailinformationen, die sich aus den Pixeln in ei­­­nem kleinen Bereich zusammensetzen. Der P-Branch versucht möglichst alle Aspekte der Detailinformationen zu erkennen und erzeugt dafür hochauflösende Fea­ture­maps.[[24]](#footnote-25)

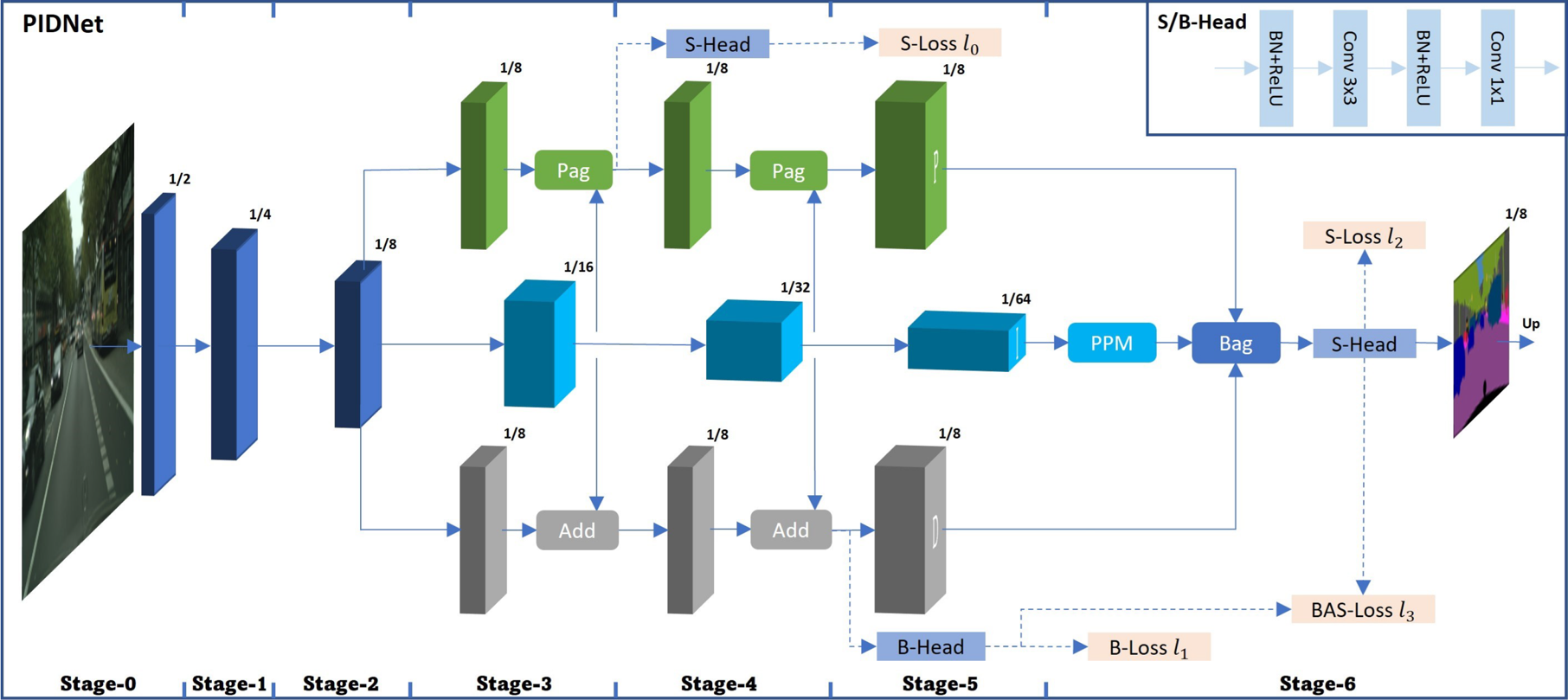


Abb. 2: Die PID-Net Architektur[[25]](#footnote-26)

Der I-Branch aggregiert lokale Detailinformationen und komplexe, globale Informationen, um semantische Verknüpfungen zwischen diesen Beiden Informationen herzustellen. Der I-Branch ermöglicht es einem PID-Net dadurch Detailinformationen in einem größeren Kon­­text zu betrachten.[[26]](#footnote-27)

Der D-Branch wird verwendet, um die Grenzen zwischen Objekten zu erkennen. Dadurch wird verhindert, dass die Detailinformationen aus dem P-Branch von den Kon­text­in­for­ma­tionen aus dem I-Branch überwältigt werden. Das ist das Äquivalent zu dem in Un­ter­ka­­­pitel 3.1 betrachteten Overshooting eines PID-Reglers (siehe links in Abbildung 3). Bei ei­nem PID-Net äußert sich ein solches Overshooting durch das Verschwimmen der Gren­zen zwischen Objekten oder so­gar das teilweise verschwinden eines Objekts durch Über­de­ckung in den Vorhersagen des Modells (siehe rechts in Abbildung 3).[[27]](#footnote-28)

Die Struktur der PID-Net Architektur basiert unabhängig vom Branch auf „residuellen Fal­tungs­­blöcken“ (siehe große dreidimensional dargestellte Boxen in Abbildung 2). Diese re­si­duellen Faltungsblöcke sind Zusam­men­schlüs­­se von mehreren Faltungsschichten je­weils ge­folgt von Aktivierung und Batch-Nor­ma­li­sie­rung. Dabei ist die Konfiguration der Fal­­tungsschichten nicht festgelegt und variiert je nach Position in dem PID-Net Modell. Die in einen solchen Block ein­ge­henden Da­ten fließen nicht nur durch diese Schicht-Folge, son­­dern auch über ei­ne „Shortcut-verbindung“ direkt zum Ende des Blocks. Dort werden bei­­de Datenflüsse ad­­diert und als Ausgabe des Blocks zurückgegeben (siehe Abbildung 4). Die residuellen Blöcke wurden für die PID-Net Architektur gewählt, da sie effizienter sind als einfache Aneinanderreihungen von Schichten.[[28]](#footnote-29)

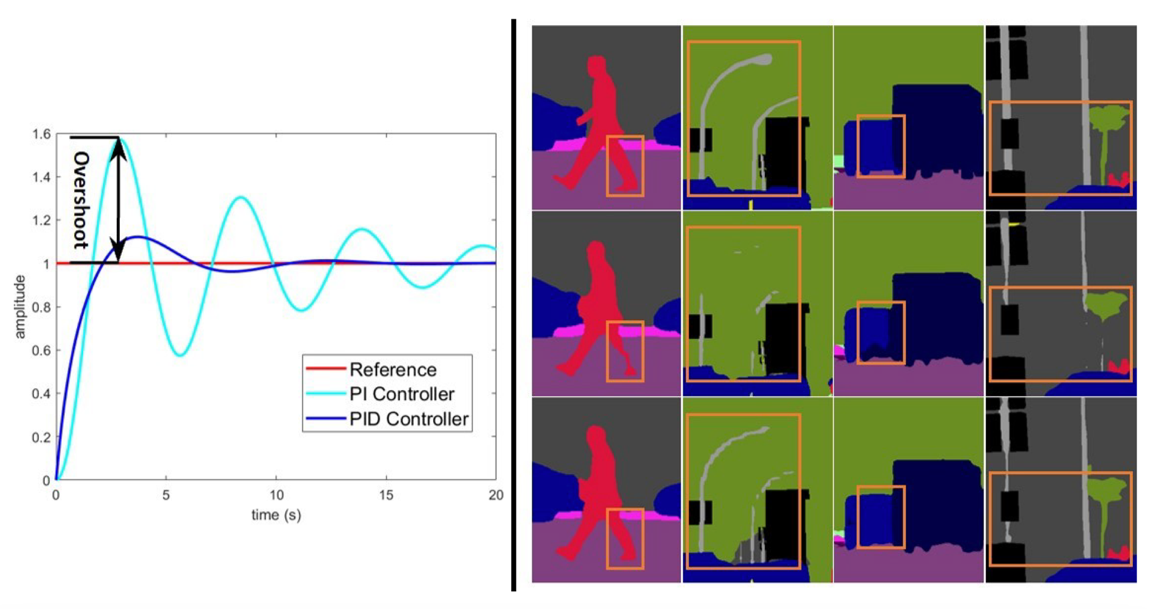


Abb. 3: Overshooting bei PI-Reglern (links) und PI-Nets (rechts)[[29]](#footnote-30)

Die PID-Net Ar­chi­tek­tur verwendet zwei Varianten von residuellen Faltungsblöcken: Eine Stan­­dard-Va­­­riante und eine Bottleneck-Variante. Die Standard-Variante besteht aus zwei Fal­­­tungs­schich­ten inklusive Aktivierung und Batch-Nor­ma­li­sie­rung und dient aus­schließ­lich der Er­­­kennung von Merkmalen in Daten. Die Standard-Variante ist der Kern der Bild­ver­­ar­bei­tung mit einem PID-Net-Modell und wird daher wesentlich häufiger als Die Bottle­­neck-Va­riante verwendet.[[30]](#footnote-31)

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abb. 4: Ein residueller Standard-Faltungsblock

Die Bottleneck-Variante besteht aus drei Fal­tungs­schich­ten inklusive Aktivierung und Batch-Nor­ma­li­sie­rung, wobei lediglich die mittlere Fal­tungsschicht der Er­kennung von Merk­­­malen in Daten dient. Die erste und die letzte Fal­tungsschicht dienen der effizienten Ex­­­­pansion der Anzahl Featuremaps. Die Bottleneck-Variante existiert nur wegen dieser Zu­­­satzfunktion. Diese Funktion wird am Ende der drei Branches von PID-Net-Modellen ver­­­wendet, um die Dimensionen der Ausgaben der drei Branches anzugleichen.[[31]](#footnote-32)

Neben den residuellen Faltungsblöcken werden in der PID-Net Architektur noch weitere Kom­­­ponenten verwendet (siehe Abbildung 2).[[32]](#footnote-33) Diese werden nachfolgend näher be­schrie­­­ben.

1. Add-Komponente:

Die Add-Komponente (Addition) ist keine Komponente in Form einer Schicht, sondern sym­bolisiert le­­diglich die elementweise Addition zweier Featuremaps. Diese Additionen wer­den ver­wen­­det, um die Kontextinformationen aus dem I-Branch in den D-Branch ein­zu­bringen (sie­­­he Abbildung 2).[[33]](#footnote-34)

1. Pag-Komponente:

Die Pag-Komponente (Pixel-attention-guided fusion module) bringt die Kon­text­in­for­ma­tio­­nen aus dem I-Branch in den P-Branch ein. Allerdings ist dieser Prozess wesentlich kom­­­­plizierter als noch bei der Add-Komponente. Der Prozess ist komplexer, da bei einer ein­fachen elementweisen Addition der P-Branch mit Kontextinformationen überlastet wird. Dadurch kommt es zu dem oben angesprochenen Overshooting. Um diese Über­las­tung zu verhindern werden die irrelevanten Kontext-Informationen von der Pag-Kom­po­nen­te herausgefiltert.[[34]](#footnote-35)

Die Struktur der Pag-Komponente ist in Abbildung 5 zu sehen. Dabei steht der Operator für eine elementweise Addition und der Operator für eine elementweise Mul­ti­pli­ka­tion. Die Abkürzung „Conv“ (Convolution) steht für eine Faltungsschicht und „BN“ steht für die Batch-Normalisierung. „Up“ (Upsampling) meint in Abbildung 5 das Vergrößern der Fläche einer Featuremap mittels bilinearer Interpolation. Der mit „Sum“ (Summe) be­schrif­tete Pfeil steht für das Summieren der Kanal-Werte einer Featuremap.

Das steht für das Er­gebnis der Sigmoid-Funktion und gibt an, ob zwei Pixel zu dem glei­chen Objekt gehören. Der I-Branch diese Situation auf Grund seiner umfangreichen Kon­text­­informationen wesentlich besser beurteilen als der P-Branch mit begrenztem Kon­text­­wissen. Daher beeinflusst das Ergebnis des P-Branchs das Endergebnis we­ni­ger stark als das Ergebnis des I-Branchs .[[35]](#footnote-36)

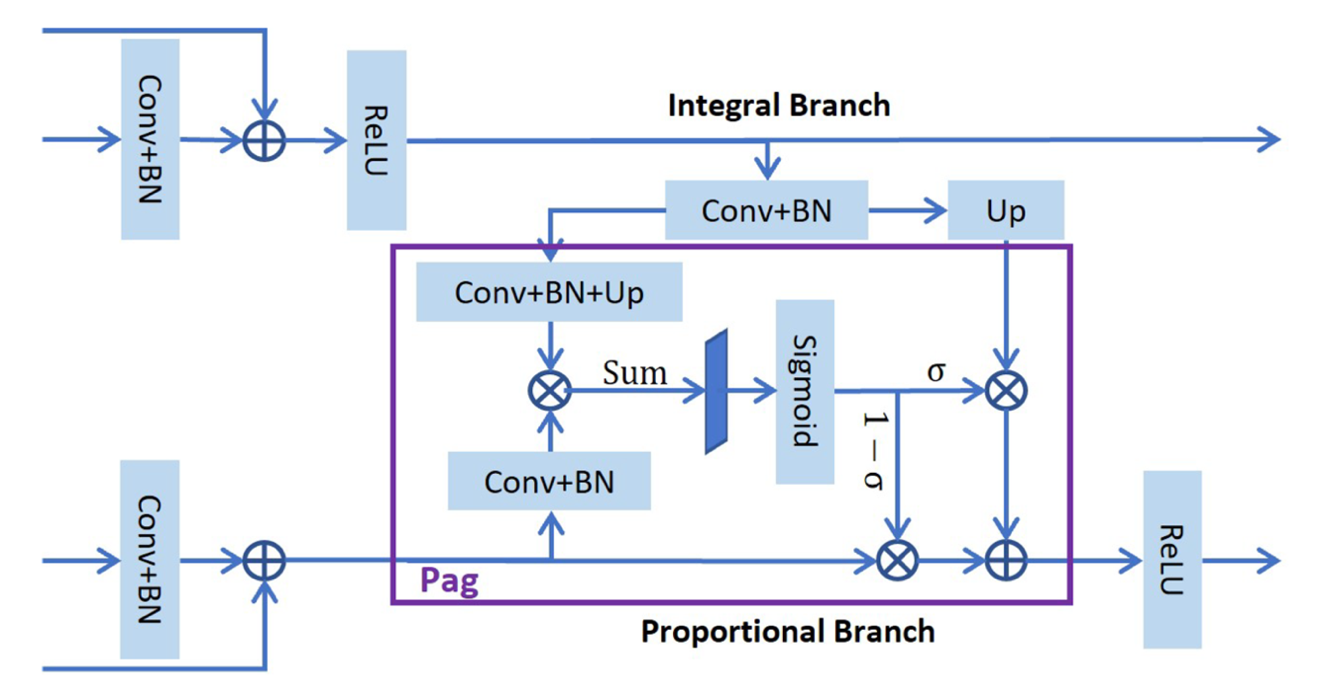


Abb. 5: Der Aufbau der Pag-Komponente[[36]](#footnote-37)

1. PPM-Komponente:

Die PPM-Komponente (Pyramid Pooling Module) sitzt am Ende des I-Branchs (siehe Ab­bildung 2) und dient der Verarbeitung globaler semantischer Abhängigkeiten im Bild. Die­se Komponente stammt ursprünglich aus der PSPNet Architektur, wurde allerdings in meh­­­reren anderen KNN-Modellen wiederverwendet und verbessert. Für die DDRNet Ar­chi­­­tektur wurde die effektivere DA-Variante (Deep Aggregation PPM) ent­wi­ckelt, die auch in sehr großen Varianten der PID-Net Architektur verwendet wer­den.[[37]](#footnote-38)

Al­lerdings kann die DA-Variante nicht parallelisiert ausgeführt werden. Zudem ent­hält die DA-Variante zu viele Parameter, die die Repräsentationsfähigkeit von klei­ne­ren Mo­del­­len übersteigt. Daher kommt die DA-Variante nicht für kleinere PID-Net Mo­delle in Frage.[[38]](#footnote-39) In der vorliegenden Projektarbeit wird ein solches kleineres PID-Net Modell im­ple­­mentiert.

Statt der DA-Variante wird die PA-Variante (Parallel Aggregation PPM) ver­wen­det. Dies ist eine in der Größe reduzierte und parallelisierte Form der DA-Va­riante und wurde für die PID-Net Architektur entwickelt. Abbildung 6 zeigt den Auf­bau der PA-Variante. Da­bei steht auch hier die Abkürzung „Conv“ wieder für eine Fal­tungs­schicht und der Ope­ra­tor für eine elementweise Addition. Der Begriff „Up­samp­ling“ steht für das Vergrößern der Fläche einer Featuremap mittels bilinearer In­ter­po­la­tion. Mittels „Concatenate“ wer­den die fünf auf verschiedene Weisen verarbeiteten Ein­ga­bedaten zu einer einzelnen Aus­ga­be ver­eint (siehe Abbildung 6). Dabei werden die Fea­ture­maps lediglich über­ei­nan­der­ge­legt. Das Ergebnis der PAPPM-Komponente hat damit so viele Kanäle wie die fünf Ver­ar­bei­tungs­­stränge zusammen.[[39]](#footnote-40)

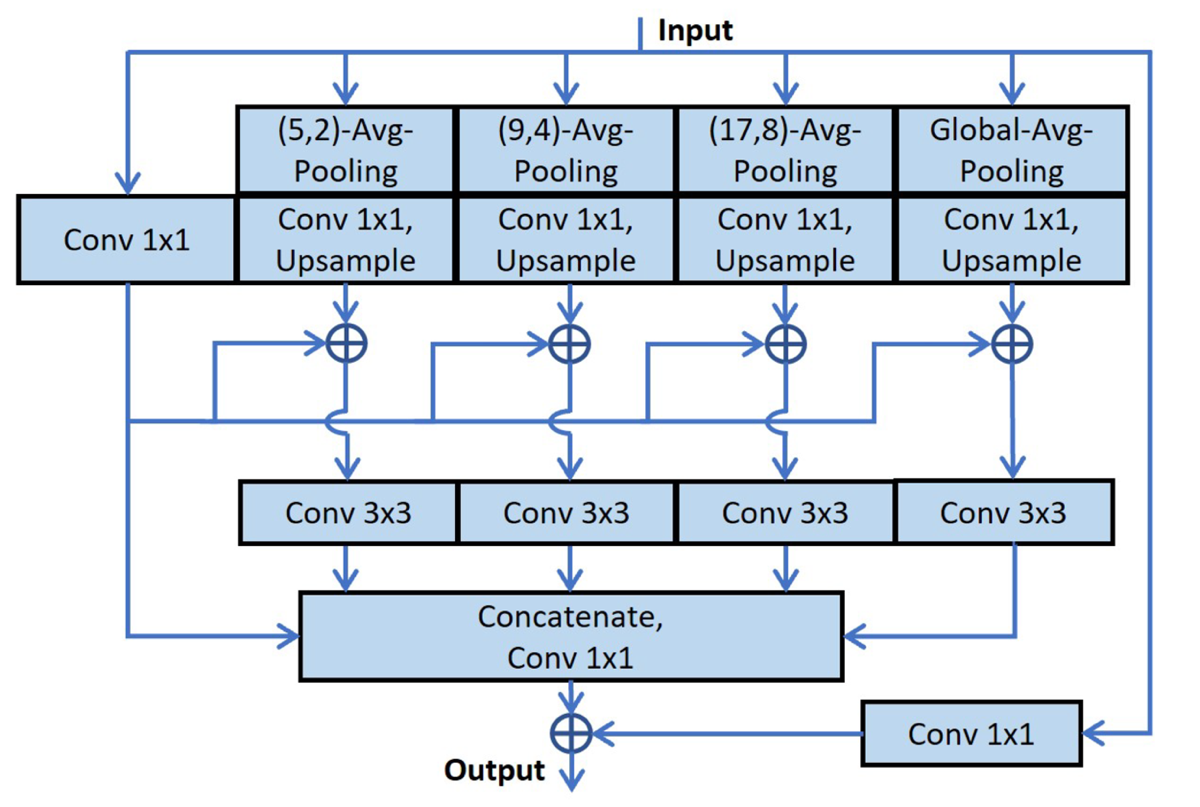


Abb. 6: Der Aufbau der PAPPM-Komponente[[40]](#footnote-41)

1. Bag-Komponente:

Die Bag-Komponente (Boundary-attention-guided fusion module) führt die Ergebnisse der drei Branches zusammen. Entsprechend sitzt die Bag-Komponente am Ende der PID-Net Architektur. Die Bag-Komponente hat die Aufgabe die Fusion der Ergebnisse aus dem P- und dem I-Branch mit den Ergebnissen aus dem D-Branch anzuleiten, sodass es nicht zu einem Overshooting kommt (sieh Un­terkapitel 3.1). Für die Grenzregionen zwischen Ob­­jekten sowie für kleine und stark detaillierte Objekte werden die Ergebnisse des P-Bran­­ches bevorzugt, der auf solche Aufgaben spezialisiert ist. Die anderen Regionen sol­len mit den Ergebnissen des I-Branches ab­gedeckt werden, der den semantischen Kon­text des Bildes im Blick hat.[[41]](#footnote-42)

Abbildung 7 zeigt die Struktur von zwei Implementierungen der Bag-Komponente. Dabei ent­­­­spricht die Symbolik der Symbolik bei der Pag-Komponente in Abbildung 5. Die links in Abbildung 7 dargestellte Variante der Bag-Komponente (a) ist die Standardausführung mit einer 3x3 Faltungsschicht. Rechts (b) ist die Light-Bag Variante abgebildet, die diese 3x3 Faltungsschicht durch zwei 1x1 Faltungsschichten ersetzt. Das verändert die Funk­tion der Komponente nicht, verbessert allerdings ihre Performanz.[[42]](#footnote-43) Daher wird in der vor­­liegenden Projektarbeit die Light-Variante der Bag-Komponente ver­wen­det.

In beiden Varianten der Bag-Komponente werden die Ergebnisse des P-Branches für das End­­ergebnis in Grenzregionen oder hochdetaillierten Bereichen verwendet; in allen an­de­ren Regionen wird das Ergebnis des I-Branchs verwendet. Das wird durch das Re­sul­­tat der Sigmoid-Funktion von dem Ergebnis des D-Branches umgesetzt. Wenn der Wert von ist, werden die Ergebnisse des P-Branches verwendet; anderenfalls werden die Ergebnisse des I-Branches verwendet.[[43]](#footnote-44)

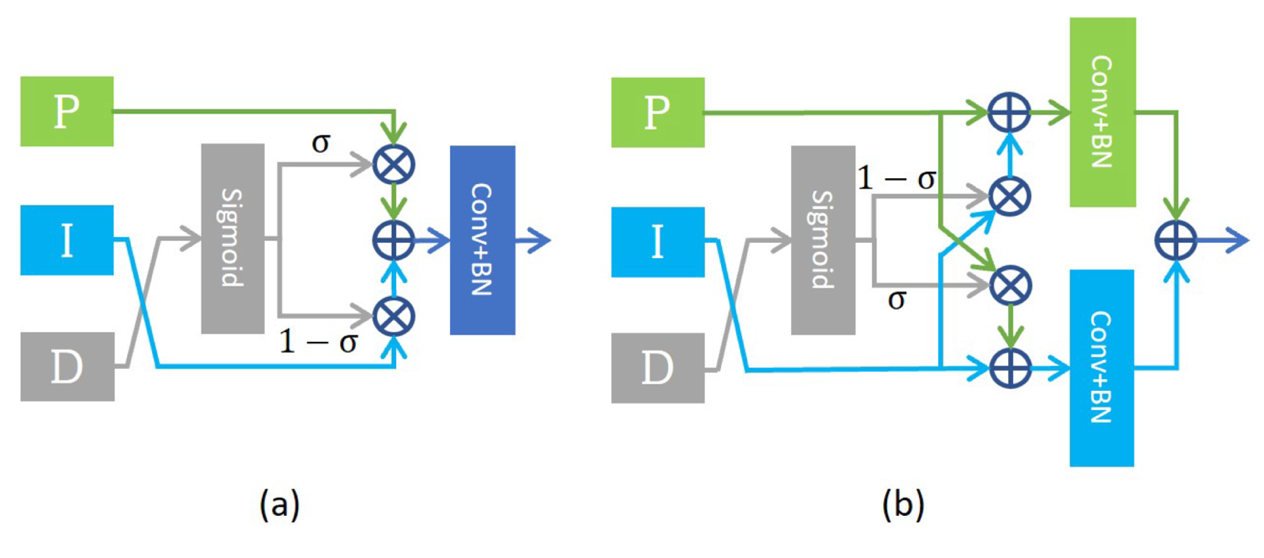


Abb. 7: Der Aufbau der Bag- (a) bzw. Light-Bag-Komponente (b)[[44]](#footnote-45)

1. S- und B-Head-Komponenten

Die S-Head- (Semantic-Head) und B-Head-Komponenten (Boundary-Head) sind alle nach dem gleichen in Abbildung 8 gezeigten Schema aufgebaut. Sie dienen der Aufbereitung der Ausgaben einer Komponente für die Berechnung eines Loss-Werts oder die Anzeige des Endergebnisses. Entsprechend befindet sich ganz am Ende der PID-Net Architektur ei­ne S-Head-Komponente (siehe Abbildung 2).[[45]](#footnote-46)

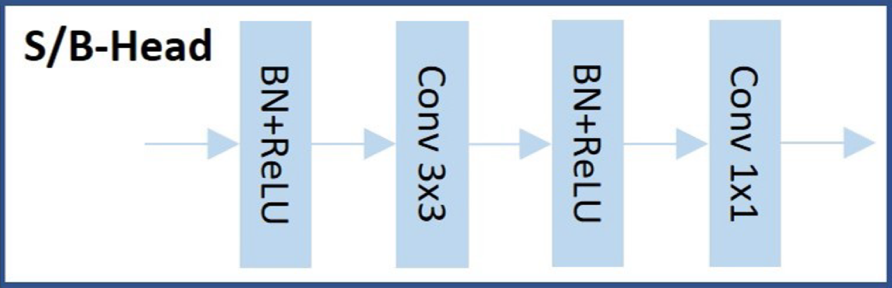


Abb. 8: Der Aufbau der S- und B-Head-Komponente[[46]](#footnote-47)

Es befindet sich allerdings noch eine S-Head-Komponente in Stage 4, die von der Ausgabe der ersten Pag-Komponente gespeist wird. Diese S-Head-Komponente dient der Op­ti­mie­rung des gesamten Modells. Die einzige B-Head-Komponente der PID-Net Architektur be­fin­­det sich in Stage 5 und wird durch die Ausgabe der zweiten Add-Komponente gespeist (sie­he Abbildung 2). Die­se B-Head-Komponente soll speziell bei der Optimierung der Er­ken­nung der Grenz­re­gio­nen helfen.[[47]](#footnote-48)

Die Ausgaben der beschriebenen S- und B-Head-Komponenten werden für die Er­rech­nung der Loss-Werte verwendet, mit denen PID-Net Modelle optimiert werden. Die PID-Net Architektur sieht für die verschiedenen S- bzw. B-Head-Komponenten un­ter­schied­li­che Loss-Funktionen vor. Für die Berechnung der Loss-Komponente wird die ge­wich­te­­te Kreuzentropie (CE, Cross Entropy) als Loss-Funktion auf die Ausgabe der vorderen S-Head-Komponente angewendet (siehe Abbildung 2). Für die Berechnung der Loss-Kom­­ponente wird die ge­wich­te­te binäre Kreuzentropie (BCE, Binary Cross En­tro­py) als Loss-Funktion auf die Ausgabe der B-Head-Komponente angewendet (siehe Ab­bil­dung 2). Für die Berechnung der Loss-Komponente wird die gewichtete CE-Loss-Funktion auf die Ausgabe der hinteren S-Head-Komponente angewendet (siehe Abbildung 2).[[48]](#footnote-49)

Für die Be­­­rech­nung der Loss-Komponente wird ein Boundary Awareness CE-Loss-Funk­tion (BAS-Loss in Abbildung 2) auf die Ausgabe der hinteren S-Head-Komponente an­­ge­wen­det. Diese BAS-Loss-Funktion berechnet die gewichtete Kreuzentropie genau wie die bei den Loss-Komponenten und . Allerdings wird durch die Ausgabe der   
B-Head-Kom­po­nen­te bestimmt, dass dies nur im näheren Umfeld von Grenzregionen und de­taillierten Be­reichen stattfinden soll

In der Formel für die Berechnung von steht für den Index eines Pixels und für eine von Klassen, denen der Pixel zugewiesen werden kann. ist die Ausgabe der B-Head-Kom­­ponente, die Ausgabe der S-Head-Komponente und die Klasse, der der Pi­xel am Index tatsächlich zugeordnet werden sollte (ground truth). Der Parameter ist ein Grenz­­wert, der angibt, wann die Kreuzentropie berechnet werden soll. Dieser Parameter wur­­de von den Erfindern der PID-Net Architektur auf festgelegt.[[49]](#footnote-50)

Alle benannten Loss-Komponenten werden als gewichtete Summe zu dem Gesamt-Loss ag­­gregiert, der zurückpropagiert wird

.

Dabei wurden die Gewichte durch die Erfinder der PID-Net Architektur fest­ge­legt:

[[50]](#footnote-51)

Für die Optimierung von PID-Net Modellen wurde von den Erfindern der Architektur der Sto­­­­chastische Gradienten Abstieg (SGD, Stochastic Gradient Descent) in einer stark pa­ra­me­­­­trisierten Form verwendet. Es wurde ein Momentum von 0,9 und ein Weight-Decay von 0,0001 verwendet. Die Lernrate für die Optimierung eines PID-Net Modells wurde von den Entwicklern so fest­­gelegt, dass sie während des Trainingsprozesses von 0,1 über 0,01 zu 0,001 reduziert wird.[[51]](#footnote-52)

# Untersuchungsumgebung

## Datensätze für Semantische Segmentierung

Bilddatensätze für Semantische Segmentierung bestehen aus den Bilddaten, die seg­men­tiert werden sollen und aus „Labels“, die die korrekte Segmentierung eines Bilds an­ge­ben. Die­­se Labels werden ge­mäß einer Loss-Funktion pixelweise mit der Vor­hersage ei­nes KNNs verglichen, um einen Loss-Wert zu errechnen.[[52]](#footnote-53)

Da­mit dieser pixelweise Vergleich stattfinden kann, müssen die Labels und die Ausgaben ei­­­nes KNNs das gleiche Format haben. Die Ausgaben eines PID-Net Modells haben nicht die Flächen-Größe des eingegebenen Bildes und des zugehörigen Labels. Um diese Flä­chen-Größe herzustellen, werden die Ausgaben eines PID-Net Modells mit bilinearer In­ter­­­­polation auf diese Größe skaliert (siehe „Up“ am rechten Rand von Abbildung 2).[[53]](#footnote-54)

Die Anzahl Kanäle – die Tiefe – der Labels und der Vorhersagen eines KNNs müssen eben­so über­einstimmen. Die Ausgaben eines KNNs für Semantische Segmentierung ha­ben im­mer genau so viele Kanäle, wie Klassen segmentiert werden sollen. Labels liegen häufig nicht in einer solchen Form vor. Stattdessen werden Labels oft als RGB-Bild (Rot-Grün-Blau) bereitgestellt, wobei eine Far­be ei­­ne Klas­se repräsentiert (sieh Abbildung 1B). Um die Tiefe von Labels und KNN-Ausgaben anzugleichen, werden i. d. R. die RGB-Labels in Klas­­­sen-Labels mit einer Tiefe der Anzahl zu segmentierender Klassen überführt.[[54]](#footnote-55)

## Der SUIM-Datensatz

Die vorliegenden Projektarbeit verwendet den „Semantic Segmentation of Underwater Ima­­gery“ (SUIM) Datensatz.[[55]](#footnote-56) Der Datensatz enthält Bilder von Un­ter­was­ser­land­­­­schaf­ten (siehe Abbildung 9) und unterscheidet zwischen den folgenden acht Klas­sen:

1. Hintergrund (Wasserkörper)
2. Taucher
3. Unterwasser-Flora
4. Wracks und Ruinen
5. Roboter und Maschinen
6. Riffe und Korallen
7. Unterwasser-Flora
8. Meeresboden und Steine

Der SUIM-Datensatz enthält ca. 1500 für eine Se­man­ti­sche Segmentierung gelabelte Bil­der für das Training und ca. 110 für eine Semantische Segmentierung gelabelte Bilder für das Testen.[[56]](#footnote-57)

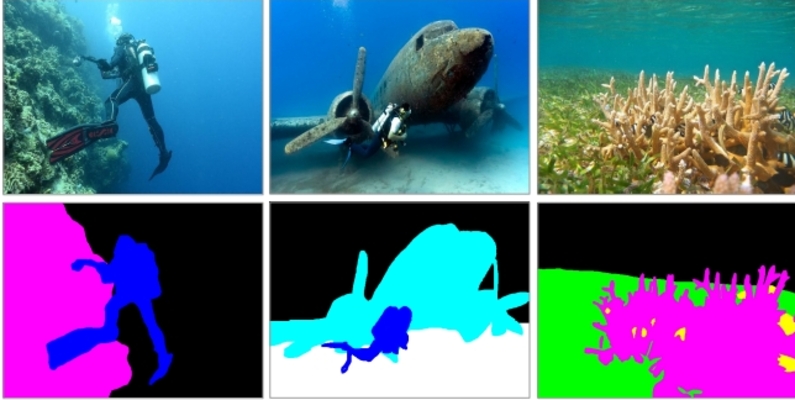


Abb. 9: Bilder (obere Zeile) und Labels (untere Zeile) aus dem SUIM-Datensatz[[57]](#footnote-58)

Die Bilder haben unterschiedliche Flächen-Größen. Für eine Ver­ar­­­­­beitung mit einem KNN werden die Bilder alle auf die Größe 1024x1024 skaliert. Dabei wird eine ge­ring­fü­gi­ge Ver­zer­rung in Kauf genommen. Alle Bilder und auch die da­zu­ge­hö­­­rigen Labels liegen mit drei Farbkanälen für den RGB-Farbraum vor. Die Bilder müssen für eine Ver­ar­bei­tung mit einem KNN in der Tiefe nicht weiter verändert werden. Die La­­­bels müssen al­ler­dings von den drei Farbkanälen auf acht Klassen-Kanäle abgebildet wer­­­den. Dafür wird die in Tabelle 1 dargestellte Menge an Abbildungen von RGB-Farb­wer­­­ten auf die oben angeführten Klasen angewendet.[[58]](#footnote-59)

|  |  |  |  |
| --- | --- | --- | --- |
| Farbe | RGB-Code | Klassenbezeichnung | Klassen-Nr. |
| Schwarz | (0, 0, 0 ) | Hintergrund (Wasserkörper) | 1 |
| Dunkelblau | (0, 0, 255) | Taucher | 2 |
| Grün | (0, 255, 0 ) | Unterwasser-Flora | 3 |
| Himmelblau | (0, 255, 255) | Wracks und Ruinen | 4 |
| Rot | (255, 0, 0 ) | Roboter und Maschinen | 5 |
| Rosa | (255, 0, 255) | Riffe und Korallen | 6 |
| Gelb | (255, 255, 0 ) | Unterwasser-Flora | 7 |
| Weiß | (255, 255, 255) | Meeresboden und Steine | 8 |

Tab. 1: Abbildung der RGB-Label-Werte auf Klassen-Label-Werte[[59]](#footnote-60)

Für ein erfolgreiches Training eines KNNs ist es wich­­tig, dass die verwendeten Daten eine ho­­­he Qualität und Varianz haben. Bei Klas­si­fi­zie­rungs- und Segmentierungsaufgaben ist es besonders wichtig, dass alle relevanten Klas­­sen in ausreichender Zahl gleichmäßig in dem Datensatz enthalten sind. Anderenfalls kann es passieren, dass unterrepräsentierte Klas­­­sen gar nicht erkannt werden oder ganze Bil­der pauschal mit überrepräsentierten Klas­­­sen klassifiziert werden.[[60]](#footnote-61)

Ab­bildung 10 zeigt die Verteilung der Klassen in dem SUIM-Trainingsdatensatz. Dabei sind auf der X-Achse die einzelnen Klassen abgetragen. Die Y-Achse gibt die Anzahl der Bil­­­­der in dem Trainingsdatensatz an, in denen die jeweilige Klasse mindestens einmal zu se­­­­­hen ist. Es fällt auf, dass der SUIM-Trainingsdatensatz stark von den Klassen „Hin­ter­grund“, „Riffe und Korallen“ sowie „Unterwasser-Flora“ dominiert wird. Die Klasse „Mee­res­­­­boden und Steine“ ist in einer einigermaßen großen Menge an Bildern vertreten. Die Klas­­sen „Taucher“, „Unterwasser-Flora“, „Wracks und Ruinen“ sowie „Roboter und Ma­­schi­­nen“ sind stark unterrepräsentiert.[[61]](#footnote-62)

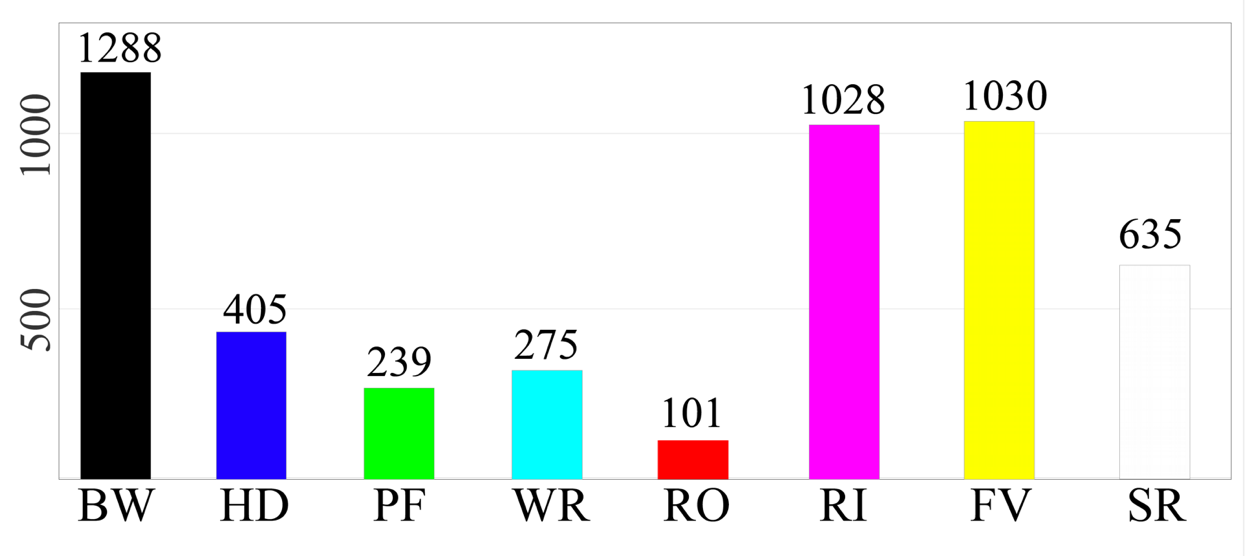


Abb. 10: Auftreten der Klassen im SUIM-Trainingsdatensatz[[62]](#footnote-63)

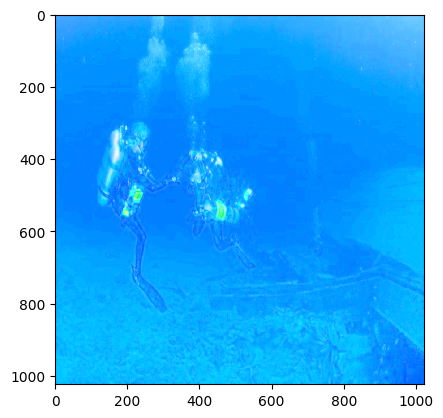
Zudem ist dem Projektteam während des Trainings aufgefallen, dass die Klassen „Hin­ter­grund“, „Riffe und Korallen“ sowie „Mee­res­­boden und Steine“, falls sie in einem Bild ent­hal­­­ten sind, einen sehr großen prozentualen Anteil des Bildes Einnehmen.

## Data Augmentation

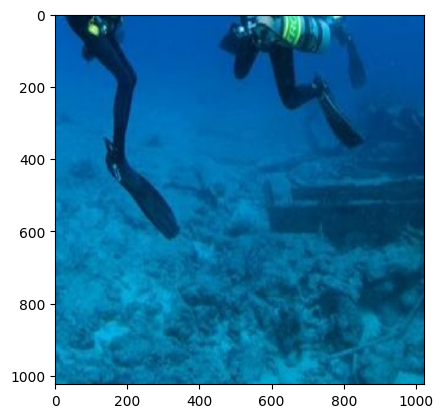
Datensätze für das Training von KNNs sollten für Komplexe Problemstellung so groß wie mög­­­lich sein und eine hohe Varianz aufweisen. Bekannte Datensätze zur Semantischen Seg­­mentierung sind „Cityscapes“ mit 5.000 hochwertig annotierten Bildern und „COCO-Stuff“ mit 10.000 hochwertig annotierten Bildern. Viele KNN Modelle und auch PID-Net Mo­delle ha­­ben in der Vergangenheit gezeigt, dass auf diesen Datensätzen gute Se­man­ti­sche Seg­men­­tierungs-Ergebnisse erzielt werden können.[[63]](#footnote-64)

Der in der vorliegenden Projektarbeit verwendete SUIM-Trainingsdatensatz besteht nur aus ca. 1500 Bildern. Hinzu kommt, dass die Daten eine Klassen-Imbalance haben. Um die­­­se beiden Probleme auszugleichen, wurde in der vorliegenden Projektarbeit Daten-Aug­­­­men­tierung vorgenommen. Dabei wurden die folgenden acht Augmentierungs-Me­tho­­­den separat ver­wendet. Abbildung 11 zeigt beispielhaft drei dieser Augmentierungen.

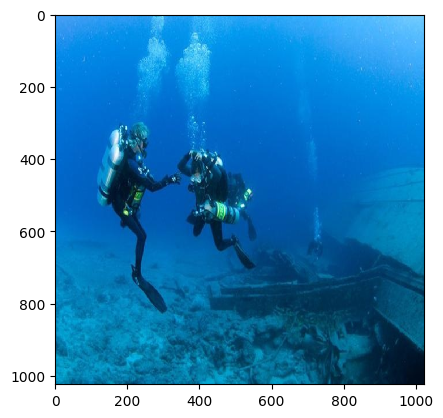
1. Zufällige Änderung der Bildhelligkeit
2. Zufällige Änderung der Bildkontrastierung
3. Zufällige Änderung der Bildunschärfe
4. Invertierung der Bildfarbe (siehe Abbildung 11b)
5. Horizontale, vertikale und kombinierte Spiegelung
6. Erzeugen von Bildausschnitten zufälliger Größe und Position (siehe Abbildung 11d)
7. Verrauschen des Bilds
8. Zufällige Änderung der Bildfarbsättigung (siehe Abbildung 11c)



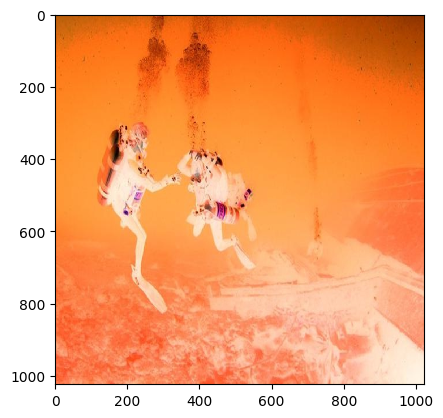
Geänderte Sättigung (c)



Bildausschnitt (d)



Original (a)



Farbinvertiert (b)

Abb. 11: Drei von acht verwendeten Data-Augmentierungs-Methoden

Alle diese Augmentierungs-Methoden wurden auf alle Bilder angewendet. Zusätzlich wur­den Bil­der vervielfältigt, die unterrepräsentierte Klassen beinhalten. Be­son­ders das Er­zeugen von Bildausschnitten von zufälliger Größe und Position wurde dafür he­­ran­ge­zo­gen, mehr Bilder von unterrepräsentierten Klassen mit neu­em Inhalt zu schaffen.

# Untersuchungsergebnisse

Die Untersuchungsergebnisse der vorliegenden Projektarbeit werden anhand der Me­tri­ken „Mean Intersection over Union“ und der „Vorhersagezeit“ bewertet. Diese beiden Me­tri­­ken werden nachfolgend kurz beschrieben. Abbildung 12 zeigt als Vergleichswert die Er­­­geb­nisse dieser Metriken der PID-Net Modelle auf den Cityscapes Testdaten.

1. Mean Intersection over Union in %

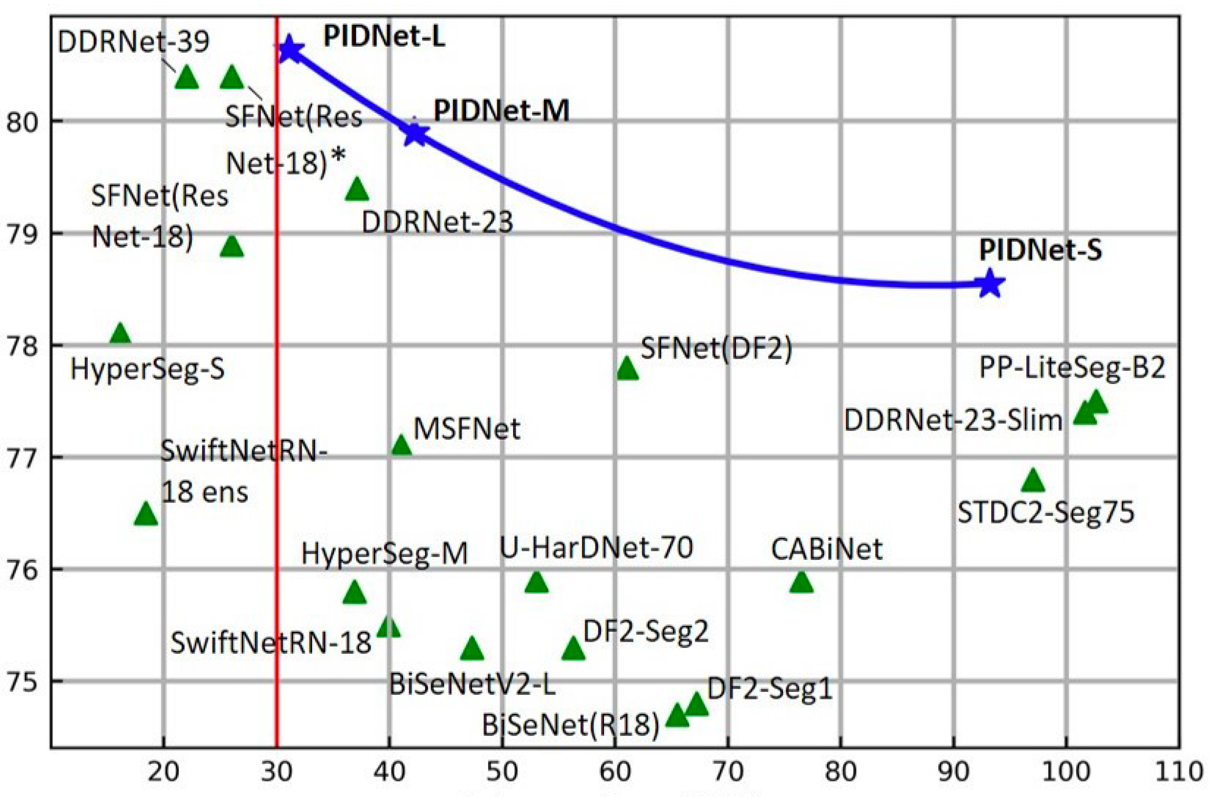
„Mean Intersection over Union“ (mIoU) ist die zentrale Bewertungsmetrik für Segmen­tie­rungs­­­­aufgaben. Sie berechnet sich als Mittelwert aus der „Intersection over Union“ (IoU) für alle Seg­mentierungs-Klassen. Die IoU für eine Klasse wird als das Verhältnis von kor­rekt klassifizierten Pixeln (Intersection) zu der Anzahl aller Pixel (Union) berechnet.[[64]](#footnote-65)

Das PID-Net-S Modell wurde bereits auf dem Cityscapes Datensatz trainiert und hat dabei ei­ne mIoU von 78,6% erreicht (siehe Abbildung 12).[[65]](#footnote-66) Die vorliegende Projektarbeit ver­sucht dieses Ergebnis auf dem SUIM-Datensatz zu validieren. Allerdings lässt sich schon im Voraus sagen, dass dies ver­mutlich nicht möglich sein wird. Denn der SUIM-Datensatz ist wesentlich kleiner als der Cityscapes-Datensatz und hat eine große Imbalance der Klas­­­sen. Das Ziel der vor­lie­gen­den Projektarbeit wurde daher vorerst optimistisch auf ei­­ne mIoU von 50% fest­ge­legt.

1. Vorhersagezeit in Millisekunden

Die Vorliegende Projektarbeit befasst sich mit Semantischer Segmentierung von Bild­da­ten in Echtzeit. Bei dieser Problemstellung muss die Vorhersagezeit des KNNs ge­ring sein. Da­mit ein Standard-Video mit 30 frames per second (fps) von einem KNN in Echtzeit seg­men­­tiert wer­den kann, muss jedes Bild in ca. 34 Millisekunden (ms) verarbeitet wer­den.

Bei dem Training des PID-Net-S Modells auf dem Cityscapes Datensatz konnte eine Vor­her­­­sagegeschwindigkeit von 93,2 fps erzielt werden (siehe Abbildung 12).[[66]](#footnote-67) Das ent­spricht ei­ner Vorhersagezeit von ca. 11 ms pro Bild. Die vorliegende Projektarbeit ver­sucht dieses Er­gebnis auf dem SUIM-Datensatz zu validieren. Im Gegensatz zur mIoU, kann dieses Er­geb­­nis in der vorliegenden Projektarbeit bestätigt werden. Denn die PID-Net Architektur wird nicht verändert. Das Ziel der vor­lie­gen­den Projektarbeit ist daher, ei­­­ne Vor­her­sa­ge­zeit von 11 ms pro Bild zu erreichen.



mIoU in %

Vorhersagegeschwindigkeit in fps

Abb. 12: Bewertung der PID-Net Architektur mit mIoU und Vorhersagezeit[[67]](#footnote-68)

Für die vorliegende Projektarbeit wurde ein PID-Net-S Modell auf dem SUIM-Datensatz für 38 Epochen trainiert. Dabei konnte durchweg bestätigt werden, dass dieses Modell ei­­ne Vorhersagezeit von ca. 11 ms pro Bild hat. Die Vorhersagegenauigkeit dieses Modells konnte allerdings wie erwartet nicht bestätigt werden. Nach zehn Epochen konnte eine mIoU von ca. 9%, nach 20 Epochen eine mIoU von ca. 12%, nach 30 Epochen eine mIoU von ca. 21% und nach 38 Epochen eine mIoU von ca. 28% festgestellt werden (siehe Ta­bel­­le 2).

Dabei zeigte sich, dass die mIoU auf den Trainingsdaten immer zwischen 6% - 8% höher lag als bei den besagten Ergebnissen auf den Testdaten. Wenn überhaupt kann hier von ei­­nem minimalen Overfitting auf den Trainingsdaten gesprochen werden.

|  |  |  |
| --- | --- | --- |
| Epoche | mIoU in % | Vorhersagezeit in ms |
| 10 | 9 | 11 |
| 20 | 12 | 11 |
| 30 | 21 | 11 |
| 38 | 28 | 11 |

Tab. 2: Trainingsergebnisse des PID-Net-S Modells auf dem SUIM-Datensatz

Abbildung 13 zeigt drei Bilder aus dem SUIM-Testdatensatz (obere Zeile) und die von dem 38 Epochen trainierten PID-Net-S Modell produzierten Ausgaben zu dem jeweiligen Bild als Maske über dem Eingabe-Bild (untere Zeile). Bei der Betrachtung dieser Ausgaben fällt auf, dass es große Unterschiede zu optimalen Labels aus Abbildung 9 gibt.

Es ist oft eine Men­ge Rauschen zu sehen, was sich in dem linken und dem rechten Bild in Ab­­­bildung 13 vor allem an den Grenzregionen abspielt. Häufig werden Objekte falsch klas­­­sifiziert (siehe das Schiffswrack im linken Bild in Abbildung 13) und die Objekte wer­den auch nicht scharf in den Grenzregionen abgegrenzt (siehe die Fische im mittleren Bild in Abbildung 13). Und in einigen Fällen werden Bestandteile eines Objekts, die weit vom Zen­­trum des Objekts entfernt sind, nicht mehr zu diesem Objekt hinzugezählt (siehe die Flü­­gel des Flugzeugwracks im rechten Bild in Abbildung 13).

Auf der anderen Seite kann die rudimentäre Erkennung von Objekten (siehe das Schiffs­wrack, den Tauscher und den Hintergrund im linken Bild in Abbildung 13; siehe das Riff und den Mee­res­bo­den im mittleren Bild in Abbildung 13; siehe das Flugzeugwrack und den Meeresboden im rechten Bild in Ab­bil­­dung 13) als Erfolg vermerkt werden. Diese Fä­­higkeit des eigens trainierten PID-Net-S Mo­dells zieht durch die Ausgaben für die SUIM-Test­­daten durch.

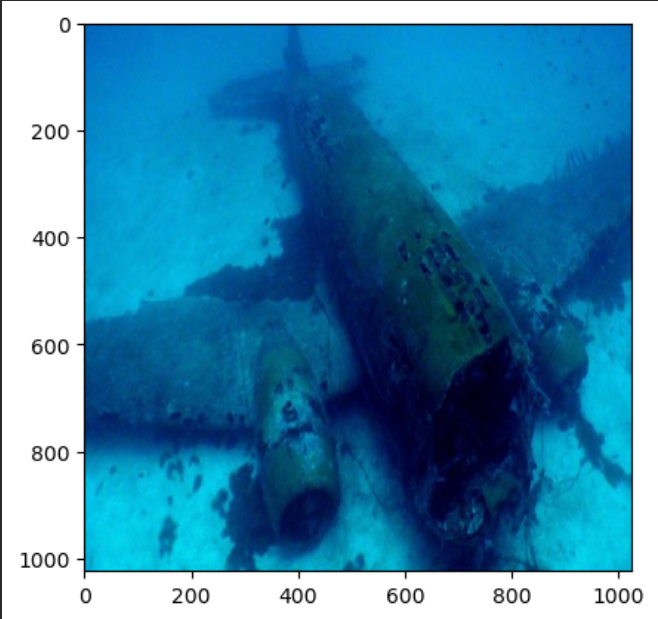
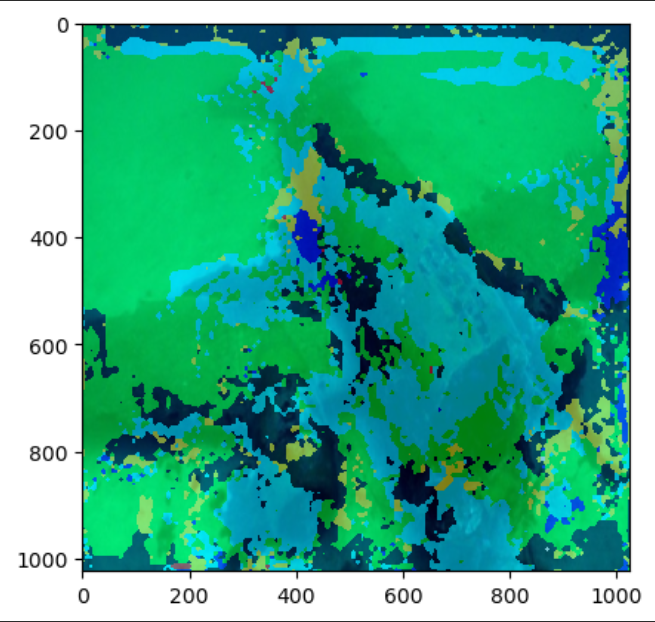
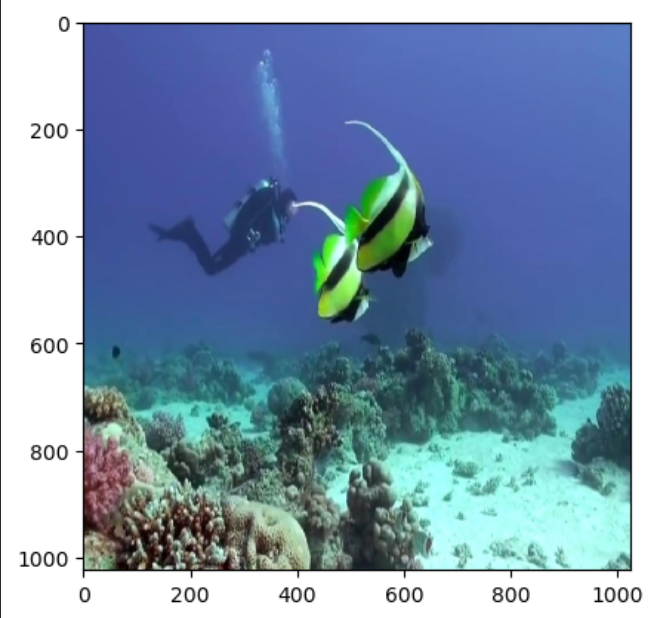
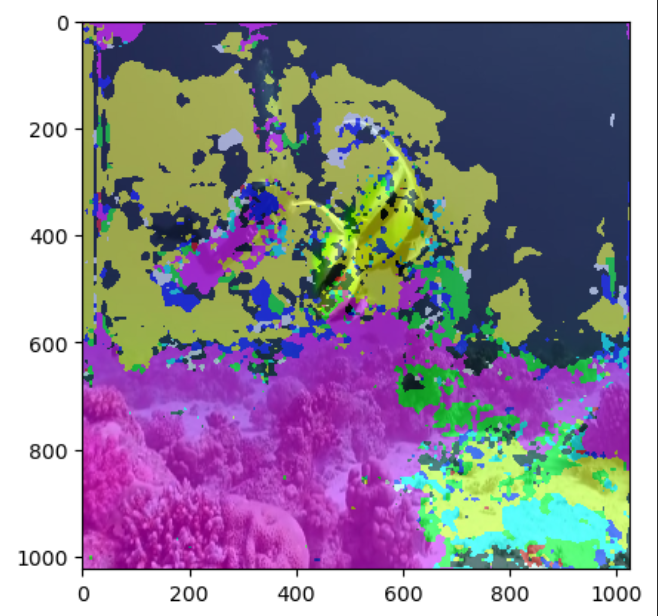
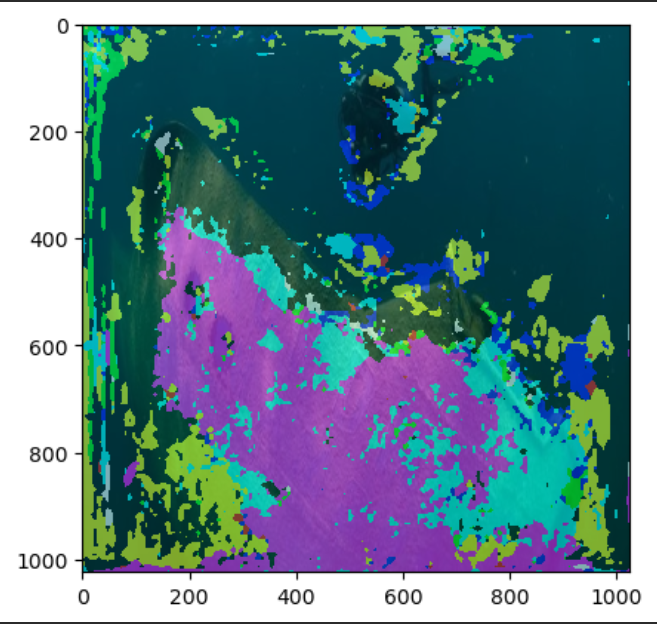
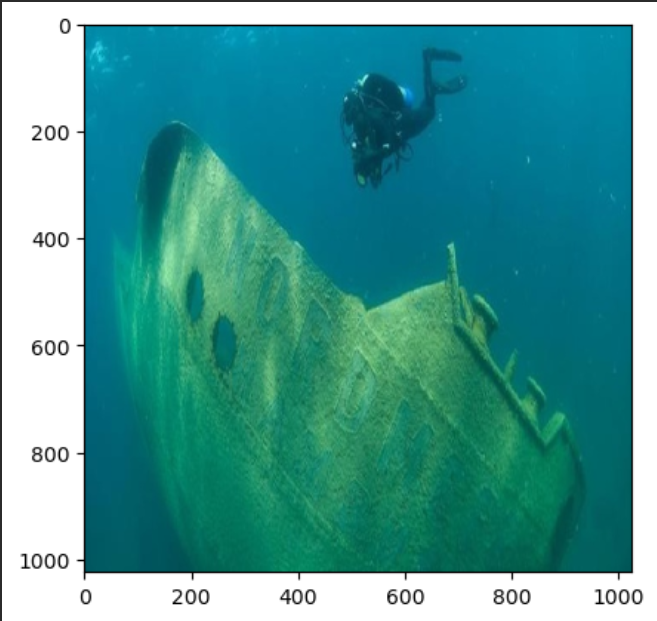


Abb. 13: Ausgaben des PID-Net-S Modells nach 38 Trainingsepochen

Ein weiterer Erfolg ist die korrekte Klassifizierung von einigen Objekten (siehe der Hin­ter­­­grund im linken Bild in Abbildung 13; siehe das Riff, die Fische und den Hintergrund im mittleren Bild in Abbildung 13; siehe das Flugzeugwrack im rechten Bild in Ab­bil­­dung 13). Diese Fähigkeit scheint allerdings bei dem 38 Epochen trainierten PID-Net-S Modell nicht so ausgeprägt zu sein, wie das Erkennen von Objekten. Für einige SUIM-Testdaten wer­­den schlechte Klassifizierungen vorgenommen.

Es fällt zudem auf, dass das PID-Net-S Modell hauptsächlich die in dem SUIM-Datensatz über­­repräsentierten Klassen „Hin­tergrund“, „Riffe und Korallen“ sowie „Meeres-Fauna“ rich­­tig klassifizieren kann. Es lässt sich vermuten, dass die in Unterkapitel 4.2 an­ge­spro­che­­ne Imbalance des SUIM-Datensatzes nicht ausreichend durch die in Unterkapitel 4.3 be­­schriebene Data-Augmentierung sowie andere Methoden ausgeglichen werden konnte.

Es konnte allerdings gezeigt werden, dass die Data-Augmentierung positive Aus­wir­kun­gen auf das Training des PID-Net-S Modells hatte. Bei vorherigen Trainingsversuchen be­stan­den die Vorhersagen des KNNs meistens aus zwei der überrepräsentierten Klassen des SUIM-Datensatzes (siehe Abbildung 14). Es konnte zwar auch in diesen Fällen schon ei­­ne Objekterkennungsfähigkeit ausgemacht werden (siehe das linke und das rechte Bild in Abbildung 14). Allerdings wurden Bilder von dem PID-Net-S Modell hauptsächlich fast voll­­ständig als Hintergrund klassifiziert (siehe das mittlere Bild in Abbildung 14).

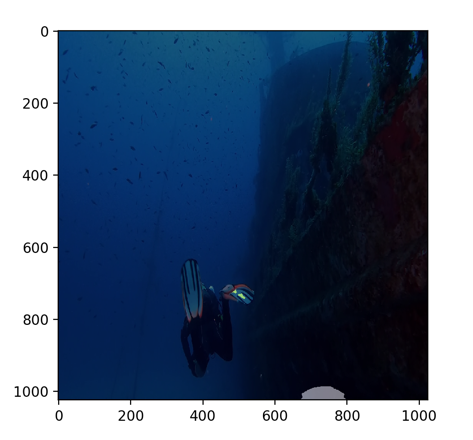
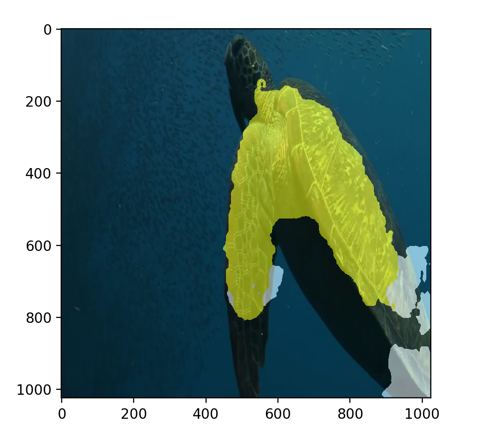
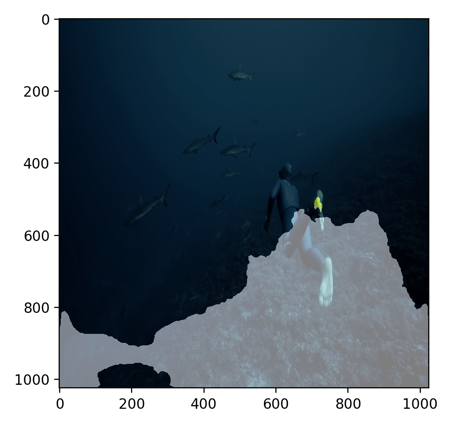


Abb. 14: Overfitting auf Klassen

Dieses Verhalten hatte das Modell vermutlich beim Training erlernt, da die Klasse „Hin­ter­­grund“ mit großem Abstand die Stärkste im Datensatz ist. Daher konnte das PID-Net-S Mo­­dell während des Trainings durch solche vollständig „schwarzen“ Vorhersagen kon­stant gute Ergebnisse erzielen. Die mIoU für diese Trainingsergebnisse lag sogar bei ca. 40%. Allerdings können diese 40% mIoU nicht als Erfolg gewertet werden, da das PID-Net-S Modell seine Funktionalität nicht generalisiert hat.

Zum Abschluss der vorliegenden Projektarbeit kann gesagt werden, dass das angestrebte Ziel von 50% mIoU und einer Vorhersagezeit von 11 ms pro Bild auf dem SUIM-Datensatz nur teilweise erreicht werden konnte. Es konnte die Echtzeitfähigkeit der PID-Net Ar­chi­tek­­tur bestätigt werden. Die Vorhersagegenauigkeit von 50% mIoU wurde nach 38 Trai­nings­epochen mit ca. 28% mIoU allerdings nicht erreicht.

Dabei muss an­gemerkt wer­den, dass sich dieser Wert mit weiterem Training und zu­sätz­li­­chen Anpassungen zum Aus­gleich der Daten-Imbalance noch deutlich verbessern könn­te. Die nach 38 Trainingsepochen erzeugten Vorhersagen sehen vielversprechend aus und weisen auf die Fähigkeit des Modells zur Objekterkennung und Klas­si­fi­zie­rung hin.

Literaturverzeichnis

1. Aggerwal, Charu: Neural Networks and Deep Learning, Springer International Pub­lishing AG (Hrsg.), 2018, https://link.springer.com/book/10.1007/978-3-319-94463-0, abgerufen am 14.01.2023.
2. Bischke, Benjamin et al.: Segmentation of Imbalanced Classes in Satellite Imagery using adaptive uncertainty weighted class loss, Technische Universität Kai­sers­lau­tern und Deutsches Forschungszentrum für Künstliche Intelligenz (Hrsg.), Deut­schland, 2018, https://www.dfki.de/fileadmin/user\_upload/im­port/9877\_ IGARSS\_2018\_bisch­ke\_\_uncertainty\_loss.pdf, abgerufen am 23.01.2023.
3. Brandenbusch, Kai: Semantische Segmentierung mit Deep Convolutional Neural Net­works, Technische Universität Dortmund – Fakultät für Informatik (Hrsg.), 12. November 2018, https://patrec.cs.tu-dort­mund.de/pubs/the­ses/ma\_kbran­den.pdf, abgerufen am 14.01.2023.
4. Cakir, Senay et al.: Semantic Segmentation for Autonomous Driving: Model Eva­lu­ation, Dataset Ge­neration, Perspective Comparison, and Real-Time Capability, Hoch­schule Esslingen (Hrsg.), 26. Juli 2022, https://arxiv.org/pdf/2207.12939. pdf, abgerufen am 14.01.2023.
5. Gholami, Amir et al.: A Survey of Quantization Methods for Efficient Neural Net­work Inference, University of California, Berkeley (Hrsg.), 21. Juni 2021, https:// ar­xiv.org/pdf/2103.13630.pdf, abgerufen am 14.01.2023.
6. Hong, Yuanduo: Deep Dual-resolution Networks for Real-time and Accurate Se­man­tic Segmentation of Road Scenes, arxiv.org (Hrsg.), 01. September 2021, https://arxiv.org/pdf/2101.06085.pdf, abgerufen am 20.01.2023.
7. Islam, Jahidul et al.: Semantic Segmentation of Underwater Imagery: Dataset and Bench­mark, arxiv.org (Hrsg.), 13. September 2020, https://arxiv.org/abs/2004. 01241, ab­gerufen am 23.01.2023.
8. Kalliomäki, Roger: Real-time object detection for autonomous vehicles using deep learning, Uppsala Universität – Department of Information Technology (Hrsg.), Juni 2019, https://uu.diva-portal.org/smash/get/di­va2:1356309/FULL­TEXT01.pdf, abgerufen am 14.01.2023.
9. Kirillov, Alexander: Exploring Aspects of Image Segmentation: Diversity, Global Rea­soning, and Panoptic Formulation, Combined Faculty for the Natural Sciences and Mathematics – Universität Heidelberg, https://archiv.ub.uni-hei­del­berg.de/ voll­textserver/25750/1/thesis\_kirillov.pdf, abgerufen am 13.01.2023.
10. Li, Zeju; Kamnitsas, Konstantinos; Glocker, Ben: Analyzing Overfitting under Class Imbalance in Neural Networks for Image Segmentation, arxiv.org (Hrsg.), 20. Februar 2021, https://arxiv.org/pdf/2102.10365.pdf,

abgerufen am 23.01.2023.

1. O. V.: Hoehenregelung einer Kugel im Luftstrom, Hochschule Bochum – Fach­be­reich Elektrotechnik und Informatik – Institut für Systemtechnik – Labor für An­triebs­technik (Hrsg.), https://www.hochschule-bochum.de/fbe/fach­ge­bie­te/in­sti­tut-fuer-systemtechnik/labor-fuer-antriebstechnik/,   
   abgerufen am 19.01.2023.
2. O. V.: Real-Time Semantic Seg­men­tation on CamVid, paperswithcode.com (Hrsg.), https://paperswithcode.com/sota/real-time-semantic-segmentation-on-camvid, abgerufen am 14.01.2023.
3. O. V.: Real-Time Semantic Seg­men­tation on Cityscapes test, paperswithcode.com (Hrsg.), https://paperswithcode.com/sota/real-time-semantic-segmentation-on-city­scapes, abgerufen am 14.01.2023.
4. Schumacher, W.: Grundlagen der Regelungstechnik, Technische Universität Braun­schweig – Institut für Regelungstechnik (Hrsg.), 03. Januar 2022, https:// srv.ifr.ing.tu-bs.de/static/files/lehre/vorlesungen/gdr/Skript\_GdR.pdf, abge­ru­fen am 19.01.2023.
5. Selle, Stefan: Künstliche Neuronale Netzwerke und Deep Learning, Hochschule für Technik und Wirtschaft des Saarlandes – Fakultät für Wirt­schafts­wis­sen­schaf­ten – Professor für Wirtschaftsinformatik (Hrsg.), Saarbrücken, 12. Mai 2018, https://www.htwsaar.de/wiwi/fakultaet-und-personen/profile/selle-ste­fan/da­tei­en/kuenstliche-neuronale-netz­werke/selle2018e\_kuenstliche\_neu­ro­na­le\_netzwerke.pdf, abgerufen am 14.01.2023.
6. Strecker, Stefan: Künstliche Neuronale Netze – Aufbau und Funktionsweise, in: Ar­beitspapiere WI, Nr. 10/1997, Hrsg.: Lehrstuhl für Allg. BWL und Wirt­schafts­in­formatik, Johannes Gutenberg-Universität: Mainz 1997, abgerufen am 14.01.2023.
7. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Se­man­tic SegmentationNetwork Inspired from PID Controller, Texas A&M University (Hrsg.), 10. Juni 2022, https://arxiv.org/pdf/2206.02066v2.pdf, abgerufen am 14.01.2023.
8. Xu, Jiacong: PIDNet: XuJiacong/PIDNet – models/model\_utils.py, github.com (Hrsg.), 27. Mai 2022, https://github.com/XuJiacong/PID­Net/blob/main/mo­dels/mo­del\_utils.py, abgerufen am 14.01.2023.
9. Xu, Jiacong: PIDNet: XuJiacong/PIDNet – models/pidnet.py, github.com (Hrsg.), 10. Juni 2022, https://github.com/Xu­Jiacong/PIDNet/blob/main/mo­dels/pid­net.py, ab­gerufen am 23.01.2023.
10. Xu, Jiacong: PID­Net: XuJiacong/PIDNet – utils/criteriom.py, github.com (Hrsg.), 28. Mai 2022, https://github.com/XuJiacong/PID­Net/blob/main/utils/cri­te­rion. py, ab­gerufen am 14.01.2023.

1. Vgl. Aggerwal, Charu: Neural Networks and Deep Learning, Springer International Publishing AG (Hrsg.), 2018, https://link.springer.com/book/10.1007/978-3-319-94463-0,

   abgerufen am 14.01.2023, S. 3. [↑](#footnote-ref-2)
2. Vgl. Kalliomäki, Roger: Real-time object detection for autonomous vehicles using deep learning, Upp­sala Universität – Department of Information Technology (Hrsg.), Juni 2019, https://uu.diva-por­tal.org/smash/get/diva2:1356309/FULLTEXT01.pdf, abgerufen am 14.01.2023, S. 1f [↑](#footnote-ref-3)
3. Vgl. Kalliomäki, Roger: Real-time object detection for autonomous vehicles using deep learning,   
   a. a. O., abgerufen am 14.01.2023, S. 65. Vgl. auch Gholami, Amir et al.: A Survey of Quantization Methods for Efficient Neural Network Inference, University of California, Berkeley (Hrsg.), 21. Juni 2021, https://arxiv.org/pdf/2103.13630.pdf, abgerufen am 14.01.2023, S. 1. [↑](#footnote-ref-4)
4. Vgl. Cakir, Senay et al.: Semantic Segmentation for Autonomous Driving: Model Evaluation, Dataset Ge­neration, Perspective Comparison, and Real-Time Capability, Hochschule Esslingen (Hrsg.), 26. Juli 2022, https://arxiv.org/pdf/2207.12939.pdf, abgerufen am 14.01.2023, S. 1. [↑](#footnote-ref-5)
5. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, Texas A&M University (Hrsg.), 10. Juni 2022, https://arx­iv.org/pdf/2206.02066v2.pdf, abgerufen am 14.01.2023. Vgl. auch o. V.: Real-Time Semantic Seg­men­tation on Cityscapes test, paperswithcode.com (Hrsg.), https://paperswithcode.com/sota/real-ti­me-semantic-segmentation-on-cityscapes, abgerufen am 14.01.2023. Vgl. auch o. V.: Real-Time Se­man­tic Seg­men­tation on CamVid, paperswithcode.com (Hrsg.), https://paperswithcode.com/sota/ real-time-semantic-seg­mentation-on-camvid, abgerufen am 14.01.2023. [↑](#footnote-ref-6)
6. Vgl. Selle, Stefan: Künstliche Neuronale Netzwerke und Deep Learning, Hochschule für Technik und Wirt­schaft des Saarlandes – Fakultät für Wirtschaftswissenschaften – Professor für Wirt­schafts­in­for­ma­tik (Hrsg.), Saarbrücken, 12. Mai 2018, https://www.htwsaar.de/wiwi/fakultaet-und-per­so­nen/ pro­file/selle-stefan/dateien/kuenstliche-neuronale-netz­wer­ke/sel­le­2018e\_kuenst­li­che\_neu­ro­na­le\_netzwerke.pdf, abgerufen am 14.01.2023, S. 13ff. Vgl. auch Strecker, Stefan: Künstliche Neuronale Net­ze – Aufbau und Funktionsweise, in: Arbeitspapiere WI, Nr. 10/1997, Hrsg.: Lehrstuhl für Allg. BWL und Wirtschaftsinformatik, Johannes Gutenberg-Universität: Mainz 1997, abgerufen am 14.01.2023, S. 4. [↑](#footnote-ref-7)
7. Vgl. Selle, Stefan: Künstliche Neuronale Netzwerke und Deep Learning, a. a. O., abgerufen am 14.01.2023, S. 15. [↑](#footnote-ref-8)
8. Vgl. Selle, Stefan: Künstliche Neuronale Netzwerke und Deep Learning, a. a. O., abgerufen am 14.01.2023, S. 14. Vgl. auch Brandenbusch, Kai: Semantische Segmentierung mit Deep Convolutional Neural Networks, Technische Universität Dortmund – Fakultät für Informatik (Hrsg.), 12. November 2018, https://patrec.cs.tu-dortmund.de/pubs/theses/ma\_kbranden.pdf, abgerufen am 14.01.2023, S. 5. [↑](#footnote-ref-9)
9. Vgl. Selle, Stefan: Künstliche Neuronale Netzwerke und Deep Learning, a. a. O., abgerufen am 14.01.2023, S. 13f. [↑](#footnote-ref-10)
10. Vgl. Selle, Stefan: Künstliche Neuronale Netzwerke und Deep Learning, a. a. O., abgerufen am 14.01.2023, S. 15. Vgl. auch Brandenbusch, Kai: Semantische Segmentierung mit Deep Convolutional Neural Networks, a. a. O., abgerufen am 14.01.2023, S. 6. [↑](#footnote-ref-11)
11. Vgl. Kirillov, Alexander: Exploring Aspects of Image Segmentation: Diversity, Global Reasoning, and Pa­noptic Formulation, Combined Faculty for the Natural Sciences and Mathematics – Universität Hei­del­berg, https://archiv.ub.uni-heidelberg.de/volltextserver/25750/1/thesis\_kirillov.pdf, abgerufen am 13.01.2023, S. 73f. [↑](#footnote-ref-12)
12. Vgl. Kirillov, Alexander: Exploring Aspects of Image Segmentation: Diversity, Global Reasoning, and Pa­noptic Formulation, abgerufen am 13.01.2023, S. 73f. [↑](#footnote-ref-13)
13. Vgl. Kirillov, Alexander: Exploring Aspects of Image Segmentation: Diversity, Global Reasoning, and Pa­noptic Formulation, a. a. O., abgerufen am 13.01.2023, S. 73f. [↑](#footnote-ref-14)
14. Vgl. Kirillov, Alexander: Exploring Aspects of Image Segmentation: Diversity, Global Reasoning, and Pa­noptic Formulation, a. a. O., abgerufen am 13.01.2023, S. 74. [↑](#footnote-ref-15)
15. Kirillov, Alexander: Exploring Aspects of Image Segmentation: Diversity, Global Reasoning, and Pa­noptic Formulation, a. a. O., abgerufen am 13.01.2023, S. 74. [↑](#footnote-ref-16)
16. Vgl. O. V.: Hoehenregelung einer Kugel im Luftstrom, Hochschule Bochum – Fachbereich Elek­tro­tech­nik und Informatik – Institut für Systemtechnik – Labor für Antriebstechnik (Hrsg.), https://www. hoch­schule-bochum.de/fbe/fachgebiete/institut-fuer-systemtechnik/labor-fuer-antriebstechnik/, ab­gerufen am 19.01.2023, S.22. [↑](#footnote-ref-17)
17. Vgl. O. V.: Hoehenregelung einer Kugel im Luftstrom, a. a. O., ab­gerufen am 19.01.2023, S.22. [↑](#footnote-ref-18)
18. Vgl. Schumacher, W.: Grundlagen der Regelungstechnik, Technische Universität Braunschweig – In­sti­tut für Regelungstechnik (Hrsg.), 03. Januar 2022, https://srv.ifr.ing.tu-bs.de/sta­tic/files/leh­re/ vor­­lesungen/gdr/Skript\_GdR.pdf, abgerufen am 19.01.2023, S. 150ff. [↑](#footnote-ref-19)
19. Vgl. O. V.: Hoehenregelung einer Kugel im Luftstrom, a. a. O., ab­gerufen am 19.01.2023, S.22. Vgl. auch Schumacher, W.: Grundlagen der Regelungstechnik, a. a. O., abgerufen am 19.01.2023, S. 152. [↑](#footnote-ref-20)
20. Vgl. Schumacher, W.: Grundlagen der Regelungstechnik, a. a. O., abgerufen am 19.01.2023, S. 168f. [↑](#footnote-ref-21)
21. Vgl. O. V.: Hoehenregelung einer Kugel im Luftstrom, a. a. O., ab­gerufen am 19.01.2023, S.22. Vgl. auch Schumacher, W.: Grundlagen der Regelungstechnik, a. a. O., abgerufen am 19.01.2023, S. 168f, 172. [↑](#footnote-ref-22)
22. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023. [↑](#footnote-ref-23)
23. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 3. [↑](#footnote-ref-24)
24. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 3. [↑](#footnote-ref-25)
25. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 4. [↑](#footnote-ref-26)
26. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 3. [↑](#footnote-ref-27)
27. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 1ff. [↑](#footnote-ref-28)
28. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 3f. Vgl. auch Hong, Yuan­duo: Deep Dual-resolution Networks for Real-time and Accurate Semantic Segmentation of Road Scenes, arxiv.org (Hrsg.), 01. September 2021, https://arxiv.org/pdf/2101.06085.pdf, abgerufen am 20.01.2023, S. 4. [↑](#footnote-ref-29)
29. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 2. [↑](#footnote-ref-30)
30. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 3f. Vgl. auch Hong, Yuan­duo: Deep Dual-resolution Networks for Real-time and Accurate Semantic Segmentation of Road Scenes, a. a. O., abgerufen am 20.01.2023, S. 4. Vgl. auch Xu, Jiacong: PIDNet: XuJiacong/PIDNet – mo­­dels/model\_utils.py, github.com (Hrsg.), 27. Mai 2022, https://github.com/XuJiacong/PID­Net/ blob/main/mo­dels/model\_utils.py, abgerufen am 14.01.2023. [↑](#footnote-ref-31)
31. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 3f. Vgl. auch Hong, Yuan­duo: Deep Dual-resolution Networks for Real-time and Accurate Semantic Segmentation of Road Scenes, a. a. O., abgerufen am 20.01.2023, S. 4. Vgl. auch Xu, Jiacong: PIDNet: XuJiacong/PIDNet – mo­dels/model\_utils.py, a. a. O., abgerufen am 14.01.2023. [↑](#footnote-ref-32)
32. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 4f. [↑](#footnote-ref-33)
33. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 4. [↑](#footnote-ref-34)
34. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 4f. [↑](#footnote-ref-35)
35. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 4f. [↑](#footnote-ref-36)
36. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 4. [↑](#footnote-ref-37)
37. Vgl. Hong, Yuan­duo: Deep Dual-resolution Networks for Real-time and Accurate Semantic Seg­men­ta­tion of Road Scenes, a. a. O., abgerufen am 20.01.2023, S. 4f. Vgl. auch Xu, Jiacong; Xiong, Zixiang; Bhat­tacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 5. [↑](#footnote-ref-38)
38. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 5. [↑](#footnote-ref-39)
39. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 5. Vgl. auch Xu, Jiacong: PID­Net: XuJiacong/PIDNet – mo­dels/model\_utils.py, a. a. O., abgerufen am 14.01.2023. [↑](#footnote-ref-40)
40. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 5. [↑](#footnote-ref-41)
41. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 5. [↑](#footnote-ref-42)
42. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 5. [↑](#footnote-ref-43)
43. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 5. [↑](#footnote-ref-44)
44. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 5. [↑](#footnote-ref-45)
45. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 4. [↑](#footnote-ref-46)
46. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 4. [↑](#footnote-ref-47)
47. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 4. [↑](#footnote-ref-48)
48. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 4. Vgl. auch Xu, Jiacong: PID­­Net: XuJiacong/PIDNet – utils/criteriom.py, github.com (Hrsg.), 28. Mai 2022, https://git­hub. com/Xu­Jiacong/PIDNet/blob/main/utils/criterion.py, abgerufen am 14.01.2023. [↑](#footnote-ref-49)
49. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 4. [↑](#footnote-ref-50)
50. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 4. [↑](#footnote-ref-51)
51. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 6. [↑](#footnote-ref-52)
52. Vgl. Brandenbusch, Kai: Semantische Segmentierung mit Deep Convolutional Neural Networks,   
    a. a. O., abgerufen am 14.01.2023, S. 3f, 41f. [↑](#footnote-ref-53)
53. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­­­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 4. Vgl. auch Xu, Jiacong: PID­Net: XuJiacong/PIDNet – models/pidnet.py, github.com (Hrsg.), 10. Juni 2022, https://git­hub. com/Xu­Jiacong/PIDNet/blob/main/models/pidnet.py, abgerufen am 23.01.2023. [↑](#footnote-ref-54)
54. Vgl. Brandenbusch, Kai: Semantische Segmentierung mit Deep Convolutional Neural Networks,   
    a. a. O., abgerufen am 14.01.2023, S. 65. [↑](#footnote-ref-55)
55. Der SUIM-Datensatz ist unter dem folgenden Link auf Kaggle.com abrufbar: https://www.kag­gle. com/datasets/ashish2001/semantic-segmentation-of-underwater-imagery-suim. [↑](#footnote-ref-56)
56. Vgl. Islam, Jahidul et al.: Semantic Segmentation of Underwater Imagery: Dataset and Benchmark, ar­xiv.org (Hrsg.), 13. September 2020, https://arxiv.org/abs/2004.01241, abgerufen am 23.01.2023, S. 1f. [↑](#footnote-ref-57)
57. Islam, Jahidul et al.: Semantic Segmentation of Underwater Imagery: Dataset and Benchmark,   
    a. a. =., abgerufen am 23.01.2023, S. 3. [↑](#footnote-ref-58)
58. Vgl. Islam, Jahidul et al.: Semantic Segmentation of Underwater Imagery: Dataset and Benchmark,   
    a. a. O., abgerufen am 23.01.2023, S. 2. [↑](#footnote-ref-59)
59. Vgl. Islam, Jahidul et al.: Semantic Segmentation of Underwater Imagery: Dataset and Benchmark,   
    a. a. O., abgerufen am 23.01.2023, S. 2. [↑](#footnote-ref-60)
60. Vgl. Bischke, Benjamin et al.: Segmentation of Imbalanced Classes in Satellite Imagery using adaptive un­certainty weighted class loss, Technische Universität Kaiserslautern und Deutsches For­schungs­zen­trum für Künstliche Intelligenz (Hrsg.), Deutschland, 2018, https://www.dfki.de/file­admin/user\_ up­load/import/9877\_IGARSS\_2018\_bischke\_\_uncertainty\_loss.pdf, abgerufen am 23.01.2023, S. 1. Vgl. auch Li, Zeju; Kamnitsas, Konstantinos; Glocker, Ben: Analyzing Overfitting under Class Im­ba­lan­ce in Neural Networks for Image Segmentation, arxiv.org (Hrsg.), 20. Februar 2021, https://ar­xiv. org/pdf/2102.10365.pdf, abgerufen am 23.01.2023. [↑](#footnote-ref-61)
61. Vgl. Islam, Jahidul et al.: Semantic Segmentation of Underwater Imagery: Dataset and Benchmark,   
    a. a. O., abgerufen am 23.01.2023, S. 2. [↑](#footnote-ref-62)
62. Islam, Jahidul et al.: Semantic Segmentation of Underwater Imagery: Dataset and Benchmark,   
    a. a. O., abgerufen am 23.01.2023, S. 2. [↑](#footnote-ref-63)
63. Vgl. Brandenbusch, Kai: Semantische Segmentierung mit Deep Convolutional Neural Networks, a. a. O., abgerufen am 14.01.2023, S. 3. Vgl. auch Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 5f. [↑](#footnote-ref-64)
64. Vgl. Brandenbusch, Kai: Semantische Segmentierung mit Deep Convolutional Neural Networks,   
    a. a. O., abgerufen am 14.01.2023, S. 62. [↑](#footnote-ref-65)
65. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­­­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 8. [↑](#footnote-ref-66)
66. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­­­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 8. [↑](#footnote-ref-67)
67. Vgl. Xu, Jiacong; Xiong, Zixiang; Bhattacharyya, Shankar: PIDNet: A Real-time Semantic Segmentation Net­­­work Inspired from PID Controller, a. a. O., abgerufen am 14.01.2023, S. 1. [↑](#footnote-ref-68)