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

Multi-Task Deep Learning auf mobilen Robotern zur Personendetektion und Schätzung der Oberkörperorientierung

Masterarbeit zur Erlangung des akademischen Grades Master of Science

Dominik Höchemer

Betreuer: M.Sc. Benjamin Lewandowski
M.Sc. Daniel Seichter

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

Die Masterarbeit wurde am 14.01.2020 bei der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau eingereicht.

An dieser Stelle möchte ich meinen ganz besonderen Dank all denen aussprechen, die mich während der Anfertigung dieser Masterarbeit tatkräftig unterstützt haben.
Zuerst bedanke ich mich herzlichst bei Prof. Dr. H.-M. Groß sowie bei meinen beiden Betreuern, M. Sc. Benjamin Lewandowski und M. Sc. Daniel Seichter, für die hervorragende Betreuung und Unterstützung. Ein großes Dankeschön geht auch an alle Personen, die sich für die Datenaufnahme im TUI-Zuse Datensatz zur Verfügung gestellt haben und an alle Mitarbeiter des Fachgebiets NIKR, welche einem bei Fragen jederzeit aushelfen konnten.
Zuletzt möchte ich mich bei meiner Familie und meinen Freunden für die moralische Unterstützung bedanken. Besonders zur Seite standen mir dabei meine Eltern Ulrike und Hermann sowie meine Freundin Olivia Treuheit.
Vielen herzlichen Dank dafür!

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

Ilmenau, 14.01.2020 .....

Dominik Höchemer

Inhaltsverzeichnis

Abkürzungsverzeichnis

CNN	Convolutional Neural Network
Conv	Convolution / Convolutional-Layer
Pool	Pooling / Pooling-Layer
SVM	Support-Vector-Maschine
ROI	Region of Interest
MTL	Multi-Task Learning
FC	Fully Connected Layer
NIKR	Fachgebiet Neuroinformatik und Kognitive Robotik der TU Ilmenau
TUI	Technische Universität Ilmenau
IoU	Intersection Over Union
FPPI	False Positives Per Image / Falsch-Positive pro Bild
GAP	Global Average Pooling
LAP	Local Average Pooling
DET	Detection Error Tradeoff
MAAD	Mean Absolute Angle Difference

Kapitel 1

Einleitung

1.1 Motivation

Machine Learning wurde schon immer stark vom menschlichen Vorbild und dessen neuronaler Verarbeitung geprägt. So stammen geläufige Begriffe, wie *Neuronen* oder *Aktivierungen*, aus Forschungen der Neurowissenschaften. Jedoch gibt es einen sehr großen Unterschied zwischen dem Lernen eines künstlichen Systems und dem des Menschen: Wächst zum Beispiel ein Kind auf, so übt es nicht einen Tag lang das Sprechen, einen Tag lang das Laufen und einen Tag lang soziale Interaktionen mit Anderen, sondern es lernt alles gleichzeitig. Gerade durch diese Kombination kann der Mensch zu einem Wesen heranwachsen, welches vielfältigste Aufgaben gleichzeitig lösen kann. Warum sollte man dieses Konzept des sogenannten *Multi-Task Learning* nicht auch auf ein technisches System anwenden?

Auch wenn die ersten Forschungen zum *Multi-Task Learning* schon zwei Jahrzehnte zurückliegen [Caruana, 1997], wurden in den letzten Jahren viele Fortschritte, vor allem im Bereich der Computer Vision, gemacht. [Kokkinos, 2017] erkennt so beispielsweise in einem RGB-Eingangsbild gleichzeitig Objekte, Körperteile, Oberflächennormalen, Kanten und eine semantische Segmentierung (Abbildung 1.1).

Weiterhin sind Systeme, die mehrere Aufgaben gleichzeitig lösen können, meist rechen-effizienter und schneller als mehrere Single-Task-Systeme. Gerade in der mobilen Assistenzrobotik kann somit durch die geschickte Wahl von Multi-Task-Systemen Rechenzeit und Energie eingespart werden. Das führt wiederum zu einer längeren Betriebszeit der Roboter.

Abbildung 1.1: Multi-Task-System schätzt verschiedenste Aufgaben in einem Neuronalen Netzwerk
*UberNet von [*Kokkinos*, 2017] detektiert durch ein neuronales Netzwerk in einem einzigen RGB-Bild Objekte, Kanten, Körperteile sowie Normalen und erzeugt eine semantische Segmentierung. Entnommen aus [*Kokkinos*, 2017].*

1.2 Problemstellung und Einordnung in die Arbeiten des Fachgebiets

Aktuell sind die Roboter des Fachgebiets NIKR[1] bereits in der Lage, Personen zu erkennen als auch deren Oberkörperorientierung robust und präzise zu schätzen [Lewandowski et al., 2019b]. Beides zusammen erlaubt in verschiedensten Projekten eine bessere Interaktion mit dem Menschen und intelligentere Navigationsstrategien. So kann beispielsweise durch die Information, wie genau eine Person gerade im Raum gedreht steht, besser eingeschätzt werden, wo sich der Roboter für eine Interaktion platzieren soll, oder wo am Menschen vorbei gefahren werden soll, wenn keine Interaktion geplant ist. Eine solche Situtation ist beispielhaft in Abbildung 1.2 dargestellt. Durch weitere Informationen, wie die Haltung einer Person (z.B. stehend, sitzend, hockend), könnten weitere Verbesserungen in die Handlungsstrategien integriert werden.

Sollen nun diese weiteren Attribute geschätzt werden, bedarf es zum aktuellen Zeitpunkt seperater Systeme, welche auf die erkannten Personendetektionen angewendet werden müssen. Dies erzeugt natürlich einen, mit der Anzahl an zu schätzenden Attributen, steigenden Ressourcenverbrauch, einen höheren Stromverbrauch und abhängig von der Implementierung auch eine höhere Last beim Übertragen von Daten zwischen Detektor und Attributsschätzer. Deshalb würde sich die Schätzung verschiedener Aufgaben

[1] Fachgebiet Neuroinformatik und Kognitive Robotik der TU Ilmenau

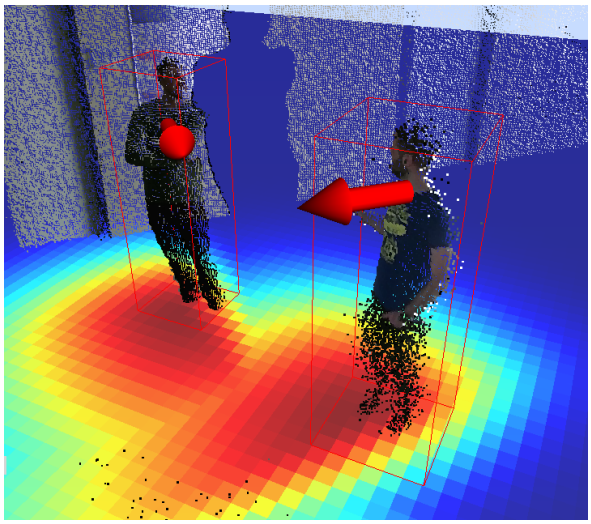


Abbildung 1.2: Darstellung des Social Space einer Personenkonstellation
Erkennbar ist auf der Bodenebene eine Kostenfunktion, die angibt, welche Bereiche der Roboter bei der Pfadplanung meiden sollte (rot) und welche Bereiche befahrbar sind (blau). Durch die Orientierungsschätzung wird der Bereich vor und rechts vor den Personen mit höheren Kosten bewertet. Das führt einerseits zu einem geringeren Eindringen des Roboters in den Bereich vor den Personen und andererseits wird so ein Rechtsfahrverhalten um die Personen herum erreicht.

in Verbindung mit einer Personendetektion anbieten, was mit dieser Masterarbeit umgesetzt werden soll.

1.3 Zielsetzung

Aufgrund der überzeugenden Ergebnisse aus der Literatur soll im Rahmen dieser Masterarbeit ein System für die Kombination aus den beiden Aufgaben *Detektion von Personen* und *Schätzung der Oberkörperorientierung* entworfen werden. In Abbildung 1.3 sind die Ein- und Ausgaben des Komplettsystems dargestellt.
Der Kern der Masterarbeit ist das Training eines tiefen Neuronalen Netzwerks, welches auf verschiedene Aufgaben gleichzeitig trainiert werden kann. Als Vorverarbeitung für das Netzwerk werden aus dem Eingangstiefenbild einer Kinect2-Kamera über einen punktwolkenbasierten Ansatz [Jafari et al., 2014] potenziell interessante Bildausschnitte erzeugt. Somit muss das Netzwerk statt einer Detektion nur eine Klassifikation der

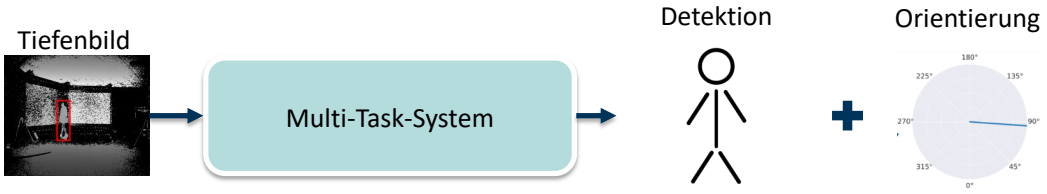


Abbildung 1.3: Geplantes System dieser Masterarbeit
Auf einem Eingangstiefenbild werden sowohl eine Personendetektion, als auch eine Oberkörperorientierungsschätzung für die detektierten Personen durchgeführt.

einzelnen Ausschnitte durchführen.
Für die Umsetzung sollen verschiedene Netzwerkarchitekturen, Loss-Funktionen und weitere, für das Multi-Task Learning notwendige Parameter recherchiert und in das Deep-Learning-Framework PyTorch[2] integriert werden. Die Netzwerke sollen dann auf den Personendatensätzen SuPer [Lewandowski et al., 2019a] und SRL [Linder et al., 2015], als auch auf dem Oberkörperorientierungsdatensatz Deep-Orientation [Lewandowski et al., 2019b] getestet und evaluiert werden. Für eine noch bessere Einordnung der Netzwerkperformance wurde im Rahmen dieser Masterarbeit ein zusätzlicher Datensatz, bestehend aus Personen mit Verdeckungen und Negativdaten, aufgenommen, welcher dann für den Test der Netzwerke verwendet werden kann.
Insgesamt sollen im Rahmen dieser Masterarbeit Erkenntnisse darüber entstehen, inwiefern das kombinierte Lernen von verschiedenartigen Aufgaben im Kontext der Personenwahrnehmung mobiler Roboter möglich ist und welche Vorteile bzw. Probleme ein solches Vorgehen mit sich führt.

1.4 Überblick über die Masterarbeit

Im folgenden Kapitel 2 beginnt diese Ausarbeitung mit einer kurzen Erklärung von wichtigen Fachtermini aus den Bereichen der Personendetektion und Orientierungsschätzung. Diese sind essentiell, um das Multi-Task-System aus Kapitel 4 verstehen zu können. In Kapitel 3 folgt dann eine State-of-the-Art-Recherche mit Fokus auf

[2] `https://pytorch.org/` - Abgerufen am 16.12.2019

grundsätzliche Ansätze zum Multi-Task Learning, sowie der Behandlung von speziellen Problemen, welche das Multi-Task Learning mit sich bringt. Daraufhin folgt in Kapitel 4 eine Beschreibung des umgesetzten Multi-Task Verfahrens. Zusätzlich werden für jede Aufgabe getrennte Single-Task-Baseline-Architekturen entworfen, mit denen dann Vergleiche durchgeführt werden können. Die experimentelle Analyse der Baselines und des Multi-Task-Systems erfolgt dann in Kapitel 5. Abschließend wird diese Masterarbeit in Kapitel 6 kurz zusammengefasst und es wird ein Ausblick für weiterführende Arbeiten gegeben.

Kapitel 2

Theoretische Grundlagen

Um diese Masterarbeit besser verstehen zu können, wird zuerst auf verschiedene Grundlagen aus den Bereichen Personendetektion, Winkeldarstellungen und Deep Learning eingegangen werden.

2.1 Typen Neuronaler Netzwerke

Zur Verarbeitung von Bildern können verschiedenste Netzwerktypen verwendet werden. Im Rahmen dieser Masterarbeit werden hier Netzarchitekturen ähnlich der Deep-Learning-Netzwerkfamilien der *VGG Netzwerke* [Simonyan und Zisserman, 2015] und der *ResNets* [He et al., 2016] verwendet. Diese beiden Klassen werden im Folgenden kurz vorgestellt.

2.1.1 VGG-Architekturen

Um große rezeptive Felder in einem Netzwerk erzeugen zu können, können entweder wenige große Filter oder viele kleine Filter (z.B. 3x3) verwendet werden. [Simonyan und Zisserman, 2015] setzen hierbei bei den VGG Netzwerken auf eine höhere Anzahl an kleinen Filtern. Bei dieser Architektur werden jeweils eine bis vier 3x3 Filterschichten hintereinandergeschaltet, worauf dann eine Pooling-Schicht folgt. Nach einer gewissen Anzahl solcher Blöcke folgt dann die Entscheidungsfindung durch mehrere vollverschaltete Schichten.

Dieser Architektur folgt auch das *BeyerModRelu*-Netzwerk, welches für die Orientierungsschätzung im Verfahren Deep-Orientation [Lewandowski et al., 2019b] eingesetzt wird. Dieses Modell hat eine, im Vergleich zu den von [Simonyan und Zisserman, 2015] verwendeten VGG-Basisnetzwerken, geringe Parameterzahl und Komplexität, da hier weniger Feature Maps eingesetzt werden. Die Architektur besteht aus acht Faltungs- und Batch-Normalization-Schichten [Ioffe und Szegedy, 2015], drei Poolingschichten und zwei vollverschalteten Schichten mit Dropout [Srivastava et al., 2014], wobei die Letzte aus zwei Ausgabeneuronen besteht, um die Orientierung zu schätzen (Sinus- und Cosinusanteil, siehe hierzu auch Kapitel 2.3). Dargestellt ist die Architektur in Abbildung 2.1.

Dieses Netzwerk setzt auf einen relativ schwachen Feature-Extraktor in den Faltungsschichten, nutzt dafür jedoch einen sehr hohen Anteil (83%) der Gewichte in den hinteren, vollverschalteten Schichten zur finalen Entscheidungsfindung. Im Kontext der Robotik lässt sich dieser Netzwerktyp für die Klassifikation von Bildausschnitten einfach einsetzen, da er sowohl performant auf einer CPU als auch auf einer schwachen GPU, wie der NVIDIA Jetson Serie, einsetzbar ist.

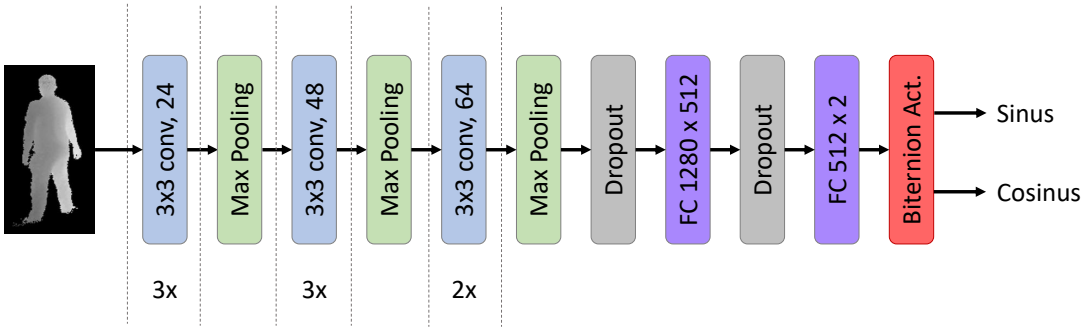


Abbildung 2.1: Aufbau des BeyerModRelu-Netzwerks
Erkennbar ist die blockweise Anordnung von Faltungs- und Poolingschichten und der nachgeschaltete, leistungsfähige Entscheidungsfinder aus vollverschalteten Schichten.

2.1.2 ResNet-Architekturen

Als Vertreter von leistungsfähigen Netzwerken zur Klassifikation von Bildern wird im Rahmen dieser Masterarbeit auch die Familie der *Residual Networks (ResNets)* getestet. Diese Netzwerke zeichnen sich durch einen sehr leistungsstarken Feature-Extraktor in den Faltungsschichten und eine einzelne vollverschaltete Schicht zur Lösung der konkreten Problemstellung (meist Klassifikation) aus. Die Faltungsschichten werden hierbei in verschiedenen Blöcken strukturiert und mit Skip Connections kombiniert [He et al., 2016]. Ein solcher *Basic Block* ist in Abbildung 2.2 dargestellt.
Die Ergebnisse des Feature-Extraktors werden am Ende durch ein *Global Average Pooling* verrechnet und von einer vollverschalteten Schicht auf die Form der Ausgabe gebracht.
ResNet-Strukturen existieren in verschiedenen Netzwerktiefen, wobei mit steigender Tiefe sowohl die Klassifikationsleistung, aber auch die Rechenkomplexität steigt. Aufgrund der begrenzten Ressourcen (Rechenleistung, Stromverbrauch) bietet sich im robotischen Bereich also eher eine niedrigere Tiefe an. Daher sollen in dieser Masterarbeit das ResNet18 als auch das ResneXt50 zum Einsatz kommen, wobei letzteres eine abgewandelte Form des ResNet50 ist, welches jedoch bei ähnlicher Rechenkomplexität leicht bessere Ergebnisse auf der Bildklassifikation zeigt [Xie et al., 2017].

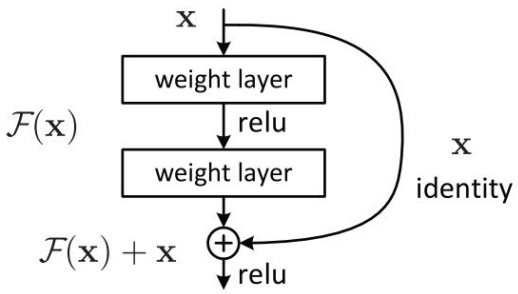


Abbildung 2.2: Aufbau eines ResNet-Blocks
Dargestellt ist ein Block, aus dem die ResNet-Architekturen zusammengesetzt werden. Hierbei wird der Input anhand von zwei aufeinanderfolgenden Faltungsschichten verarbeitet. Auf das Ergebnis der Faltungen wird dann wiederum der Input addiert. Diese Struktur wird als Shortcut im Netzwerk bezeichnet. Entnommen aus [He et al., 2016].

2.2 Echtzeitfähigkeit in der Robotik am Fachgebiet NIKR

Zur Problemlösung stehen sehr viele verschiedene Netzwerkstrukturen zur Verfügung. Jedoch sind viele hiervon auf einer mobilen Plattform mit begrenzten Rechenkapazitäten in der Inferenz nicht *echtzeitfähig* einsetzbar. Echtzeitfähig ist ein Detektor im Rahmen der Robotik des Fachgebiets NIKR dann, wenn dieser seine Detektionen mit einer Datenrate von mindestens fünf Hz bereitstellen kann. Weiterhin muss diese Datenrate durch ausschließliche Verwendung der CPU oder eines externen NVIDIA Jetson-Grafikmoduls erreicht werden, da aufgrund des hohen Stromverbrauchs keine leistungsfähigere Grafikkarte eingesetzt werden kann.

2.3 Umgang mit Winkeln beim Netzwerktraining

Oberkörperorientierungen werden im Rahmen dieser Masterarbeit als Winkel, relativ zur Kamera, angegeben. Deshalb muss sowohl während des Trainings in der Loss-Funktion, als auch bei den Bewertungsmaßen mit Winkeln und deren Differenzen umgegangen werden. Daher soll nun auf Varianten zur Bewertung von Winkelschätzungen eingegangen werden. Hierbei stellt sich die Frage, ob die Verwendung einer absoluten Winkeldifferenz sinnvoll ist, da sich durch die Periodizität von Winkeln Probleme ergeben. So entspricht der Winkel $0°$ genauso $-360°$, $360°$ oder einem beliebigen ganzzahligen Vielfachen von $360°$. Die absolute Winkeldifferenz bei einer Ground-Truth-Orientierung von $359°$ unterscheidet sich für die beiden Schätzungen $358°$ und $0°$, obwohl sie im Einheitskreis gleich weit von der Ground Truth entfernt sind und beide nur einen Fehler von $1°$ machen: Der Winkel $0°$ wird mit einem absoluten Fehler von $359°$ gewichtet, während $358°$ nur mit einem Fehler von $1°$ gewichtet wird. Die Nutzung einer Modulo-Operation in der Loss-Funktion wäre denkbar, führt jedoch zu Unstetigkeiten und behindert damit die robuste Optimierung des Netzes [Beyer et al., 2015]. Daher wird von [Beyer et al., 2015] empfohlen, die *Von-Mises-Loss-Funktion* (Gleichung 2.1) zu verwenden. Diese kann durch die Cosinusfunktion mit der Periodizität der Winkel umgehen und wichtet die beiden Schätzungen ($0°$ und $358°$) dank der Cosinusfunktion gleich. Durch einen Parameter κ kann einerseits die Steilheit, als auch der Maximalwert der Fehlerfunktion eingestellt werden. In Abbildung 2.3 ist die Von-Mises-Loss-Funktion für verschiedene

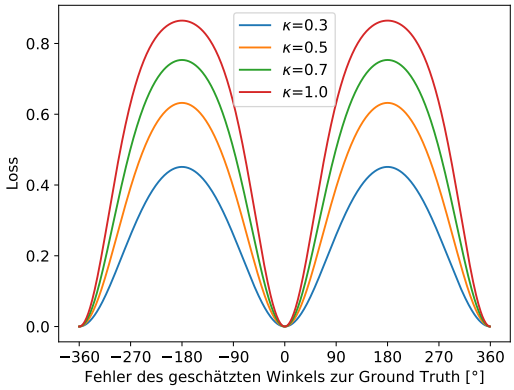


Abbildung 2.3: Loss-Funktion Von Mises
Erkennbar ist die Periodizität der Loss-Funktion: Ein Fehler von 360° entspricht dem von 0°. Weiterhin ist gut verständlich, dass die Funktion bei ungeraden Vielfachen von 180° ihre Maxima besitzt.

Parameter κ und Winkelfehler dargestellt.

$$L_{Von\_Mises} = 1 - e^{\kappa \cdot (cos(\phi_{predicted} - \phi_{groundtruth}) - 1)} \tag{2.1}$$

L_{Von_Mises}	Von-Mises-Loss
κ	Wichtungsfaktor der Loss-Funktion
$\phi_{predicted}$	Schätzung des Winkels
$\phi_{groundtruth}$	Groundtruth des Winkels

Weiterhin empfehlen [Beyer et al., 2015], den Winkel vom Netzwerk nicht durch eine einzelne Ausgabe ϕ schätzen zu lassen, sondern stattdessen die *Biternion-Repräsentation* des Winkels zu schätzen: Hierbei werden am Netzwerkausgang zwei Neuronen verwendet, welche dann den Sinus- und Cosinusanteil des Winkels ($sin(\phi)$, $cos(\phi)$) erzeugen. Wird dabei die Ausgabe des Netzwerks auf Eins normiert, kann statt des Cosinus in der Von-Mises-Loss-Funktion auch einfach das Skalarprodukt verwendet werden. Somit ergibt sich die Von-Mises-Loss-Funktion für Biternion-Repräsentationen wie in Gleichung 2.2

$$L_{Von\_Mises\_Biternion} = 1 - e^{\kappa(y_{predicted} \cdot y_{groundtruth} - 1)} \tag{2.2}$$

$L_{Von\_Mises\_Biternion}$	Von-Mises-Loss
κ	Wichtungsfaktor der Loss-Funktion
$\underline{\mathbf{y_{predicted}}}$	Sinus und Cosinus des geschätzten Winkels
$\underline{\mathbf{y_{groundtruth}}}$	Sinus und Cosinus des Groundtruth-Winkels

Hiermit sind die Probleme der Wichtung von Winkelfehlern gelöst und ein robustes Training von Netzwerken wird möglich.

2.4 Bewertungsmaße

Neben den Loss-Funktionen beim Training sind für die Auswahl und Bewertung von Neuronalen Netzwerken auch interpretierbare Maßzahlen erforderlich.

2.4.1 Winkelfehlermaß für die Orientierungsschätzung

Um die Genauigkeit einer Orientierungsschätzung bewerten zu können, muss hierzu ein Gütemaß verwendet werden. Hierzu wird oft der mittlere, absolute Fehler (MAE) zwischen geschätztem Winkel und Ground-Truth-Winkel verwendet. Um mit der Periodizität der Winkel umgehen zu können, wird der Fehler wie folgt berechnet:

$$\phi_{Differenz} = \phi_{pred} - \phi_{gt} \tag{2.3}$$

$$\phi_{Fehler} = (\phi_{Differenz} + 180°) \mod 360° - 180° \tag{2.4}$$

Formeln 2.3 und 2.4 errechnen den kleinsten Winkel zwischen Groundtruth und Schätzung im Bereich von $-180°$ bis $180°$. Die ϕ_{Fehler} werden dann betragsweise aufsummiert und durch die Anzahl der Elemente geteilt, was auf den mittleren Winkelfehler ϕ_{MAE} führt.

2.4.2 Detektionsmaße

Zur Bewertung von Personendetektoren kann eine Bewertung durch Detection-Error-Tradeoff-Kurven durchgeführt werden. Hierbei werden für jeden Detektor die Falsch-

Positiven Detektionen pro Bild (FPPI) auf der X-Achse und die Miss Rate (MR) auf der Y-Achse aufgetragen. Gibt der Detektor für jede Detektion einen Score an, kann durch Auswahl von unterschiedlichen Schwellwerten eine durchgängige Kurve eingezeichnet werden. Im Allgemeinen erreichen Systeme immer dann eine sehr gute Erkennungsleistung (geringe Miss Rate), wenn der Schwellwert liberal gewählt wird. Dies führt dann aber auch zu einem höheren Anteil an Falsch-Positiven Detektionen. Umgekehrt gilt dies genauso für einen konservativen Schwellwert. Anhand dieser Diagramme lassen sich verschiedene Detektoren gut miteinander vergleichen. In Abbildung 2.4 sind beispielhaft DET-Kurven für verschiedene Detektoren dargestellt.

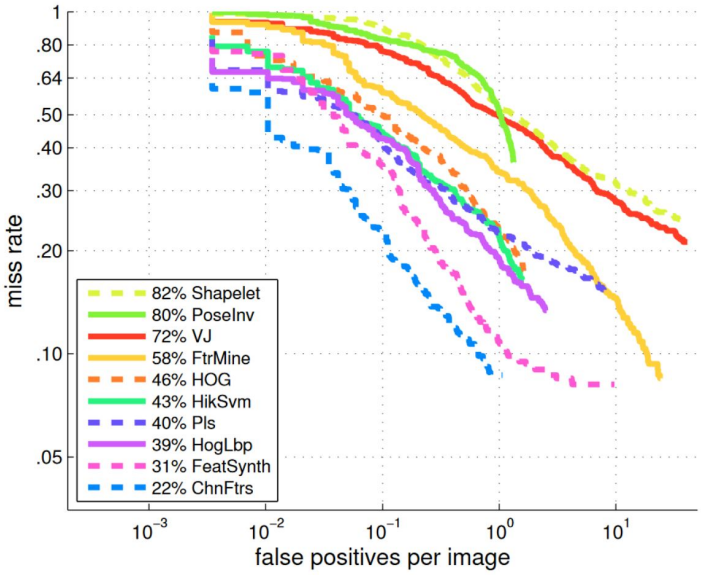


Abbildung 2.4: DET-Kurven für verschiedene Detektoren
*Auf der X-Achse ist logarithmisch der Anteil der Falschdetektionen pro Bild aufgetragen, auf der Y-Achse die Miss Rate. Erkennbar ist, dass der blaue Detektor bei den gleichen Falschdetektionsraten eine geringere Miss Rate als die anderen Detektoren hat, also mehr Personen erkannt werden können. Entnommen aus [*Dollar *et al., 2012].*

2.4.3 Klassifikationsmaße

Zur Bewertung von binären Klassifikationen (z.B. Person / Nicht Person) bieten sich verschiedenste Fehler- und Gütemaße an. Anhand der beiden Klassen können eine

Konfusionsmatrix aufgestellt und unterschiedliche Kennzahlen berechnet werden.
Sind in den Daten Negativbeispiele enthalten, bietet sich ein großes Spektrum an Maßen an. Für diese Masterarbeit sind hierbei vor allem der F1-Score (Harmonisches Mittel aus Recall und Precision) und die Balanced Error Rate wichtig.

$$Recall = \frac{tp}{P} \tag{2.5}$$

$$Precision = \frac{tp}{tp + fp} \tag{2.6}$$

$$F1\_Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.7}$$

$$BER = \frac{1}{2}(\frac{fn}{P} + \frac{fp}{N}) \tag{2.8}$$

tp	Richtig-Positive Detektionen
fp	Falsch-Positive Detektionen
fn	Falsch-Negative Detektionen
P	Elemente der Positivklasse
N	Elemente der Negativklasse

Die *Balanced Error Rate* wichtet die Fehler auf Positiv- und Negativklasse zu je 50%. Somit gehen der Prozentsatz von nicht erkannten Personen und der Prozentsatz von falsch als Personen detektierten Negativdaten gleichermaßen in die Wertung ein.
Der F1_Score mittelt den Anteil an erkannten Personen (Recall) und den Anteil an richtigen Detektionen an allen Detektionen (Precision) über das harmonische Mittel.

Sind in einem Datensatz keine Negativdaten vorhanden (N=0), sind die genannten Maße nicht mehr berechenbar (BER) oder nicht mehr aussagekräftig (F1_Score). Daher kann hier nur der Recall verwendet werden, um die binäre Klassifikations- bzw. Erkennungsleistung zu bewerten.

Kapitel 3

State of the Art

Multi-Task Learning ist ein sehr breites Forschungsfeld des Machine Learning, bei dem es um das gemeinsame Lernen von verschiedenen Aufgaben bzw. Aufgabenfeldern in einem kombinierten System geht. Im Rahmen dieser Masterarbeit soll es eingesetzt werden, um Bildausschnitte aus Tiefenbildern als Personen zu klassifizieren und deren Oberkörperorientierung zu schätzen. In Abbildung 3.1 ist dies nochmals kurz dargestellt.

Deshalb wird in diesem Kapitel der State of the Art zum Thema Multi-Task Learning herausgearbeitet. Zu Beginn wird kurz auf eine Taxonomie der verschiedenen Arten des Multi-Task Learning eingegangen (Kapitel 3.1). Daraufhin werden verschiedene Punkte diskutiert, welche für ein erfolgreiches Training eines Multi-Task-Systems notwendig sind:

- Die richtige Netzwerkarchitektur (Kapitel 3.2 und 3.3)
- Eine passende Loss-Funktion (Kapitel 3.4)
- Möglichkeiten, verschiedene Datensätze zu kombinieren (Kapitel 3.5)

Daraufhin folgt noch ein kurzer Einblick in eine Herangehensweise, Multi-Task Learning einzusetzen, um eine bessere Single-Task Performance zu erreichen (Kapitel 3.6), woraufhin dieses Kapitel mit einem kurzen Fazit, welches die wichtigsten Erkenntnisse für diese Masterarbeit zusammenfasst, abschließt (Kapitel 3.7).

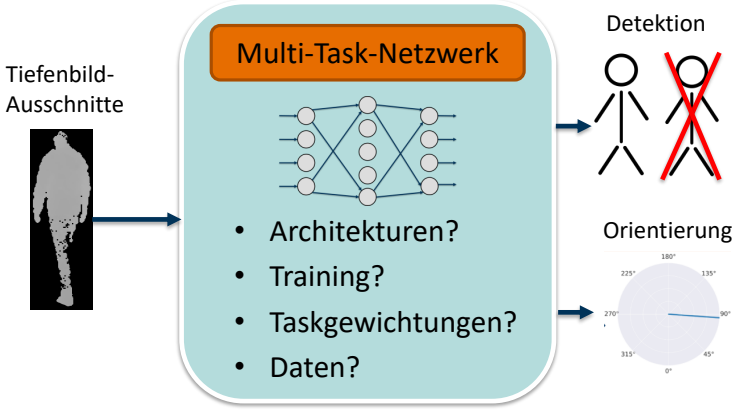


Abbildung 3.1: Machine-Learning-Teil der Systemarchitektur
Die Tiefenbildausschnitte werden anhand eines Multi-Task-Systems als Personen klassifiziert und deren Orientierung geschätzt. Die hierfür wichtigen Einflussfaktoren werden im Rahmen dieser State-of-the-Art-Recherche herausgearbeitet.

3.1 Varianten des Multi-Task Learning

Zu Beginn werden der Begriff und die Varianten des Multi-Task Learning kurz eingeordnet. Hierzu ist in Abbildung 3.2 eine Systematisierung der verschiedenen Themenfelder des Multi-Task Learning dargestellt.

3.1.1 Grundlegende Einteilung von Multi-Task Learning

Sollen mehrere *gleichartige* Aufgaben in einem System gelöst werden, wird das als *homogenes* Multi-Task Learning bezeichnet. Hierbei sind die Ausgaberäume aller Aufgabenfelder gleich. Als Beispiel hierfür wäre die Kombination aus der Klassifikation „Person trägt einen Hut“ mit der Klassifikation „Person trägt eine Brille“ zu nennen. Beide Klassifikationen sind binär.
Im Gegensatz dazu können bei Methoden des heterogenen Multi-Task Learning auch unterschiedliche Ausgaberäume kombiniert werden [Ruder, 2017]. Deshalb gehört beispielsweise ein System, das eine binäre Klassifikation eines Bildausschnitt mit der kontinuierlichen Regression eines Attributes, wie der Körpergröße, kombiniert, zum heterogenen Multi-Task Learning.
Erkennbar ist, dass im heterogenen Fall mehr Probleme kombiniert werden können. Gerade auch im Hinblick auf die Zielstellung dieser Masterarbeit, eine Klassifikation von

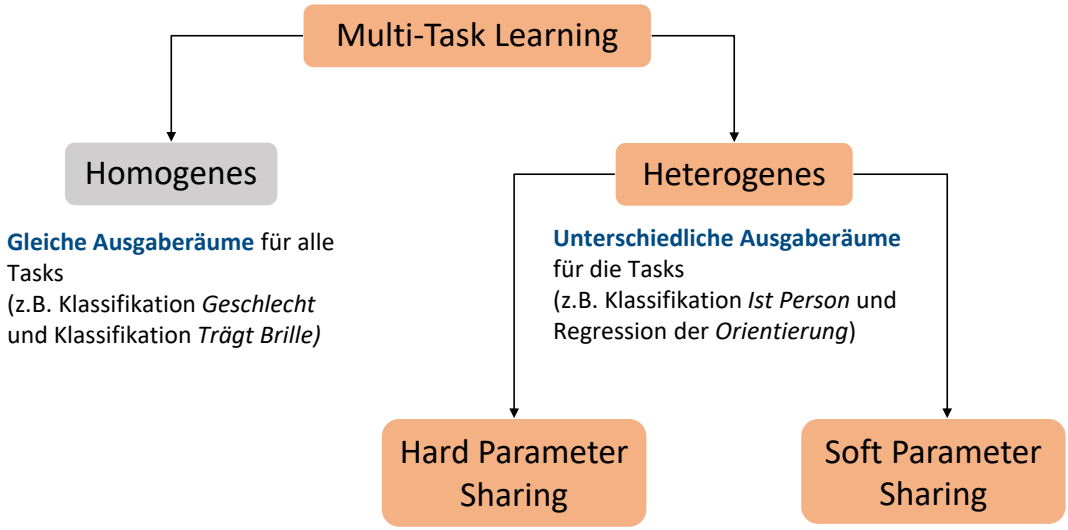


Abbildung 3.2: Taxonomie zum State of the Art des Multi-Task Learning
Multi-Task Learning lässt sich, abhängig von den Ausgaberäumen der einzelnen Aufgaben, in die zwei Kategorien, homogenes *und* heterogenes *Multi-Task Learning einteilen. Das* heterogene *Multi-Task Learning gliedert sich dabei weiter in die Methoden des* Hard Parameter Sharing *und des* Soft Parameter Sharing *auf.*

Personenausschnitten mit der Schätzung der Oberkörperorientierung zu kombinieren, soll der Fokus im Weiteren auf dem heterogenen Multi-Task Learning liegen.

3.1.2 Informationsaustausch im Netzwerk

Multi-Task-Systeme lassen sich, abhängig davon, wie die Netzwerke aufgebaut und die Informationen zwischen den Tasks ausgetauscht werden, in zwei Varianten untergliedern [Ruder, 2017]. Beim *Hard Parameter Sharing*, dargestellt in Abbildung 3.3, wird vom Entwickler eine feste Struktur entworfen, welche sich in geteilte Schichten zur Feature-Extraktion und taskspezifische Schichten zur Entscheidungsfindung aufteilt. Diese sind jeweils klar voneinander getrennt und letztere sind eindeutig den Tasks zuordenbar. Beim *Soft Parameter Sharing*, dargestellt in Abbildung 3.4, wird der Beitrag bzw. die Zugehörigkeit einer Schicht zu einem bestimmten Task entweder adaptiv bestimmt

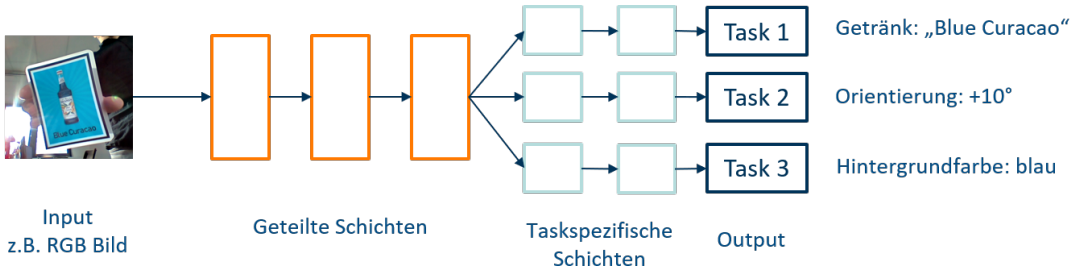


Abbildung 3.3: Schematische Darstellung des Hard Parameter Sharing
Beim Hard Parameter Sharing werden aus dem Input in den geteilten Schichten Features berechnet, welche für alle Aufgaben nützlich sind. Diese Features werden dann von allen taskspezifischen Schichten verrechnet, um die jeweilige Ausgabe zu erzeugen [Zhang et al., 2014].

oder gelernt. Hier ist weder eine direkte Trennung, noch eine direkte Zuordnung von den Schichten zu den einzelnen Tasks erkennbar.

3.1.3 Zielstellung des Netzwerks

Weiterhin unterscheiden sich Multi-Task-Systeme in der Frage, ob das resultierende System alle trainierten Tasks in der Inferenz verarbeiten soll, oder ob sich die Tasks in Haupt- und Nebentasks unterteilen lassen.

Ersteres ist der Standard-Fall des Multi-Task Learning, bei dem ein System trainiert werden soll, welches eine Vielzahl an Aufgaben gleichzeitig löst.

Die im weiteren Verlauf dieser State-of-the-Art-Recherche vorgestellten Methoden des Multi-Task Learning lassen sich jedoch auch für eine Verbesserung der Netzwerkleistung einer einzelnen Aufgabe einsetzen. So kann die Genauigkeit eines Systems auf einem (Haupt-)Task erhöht werden, indem sogenannte *Auxillary Tasks* in das System integriert werden. Auxillary Tasks sind Aufgaben, an deren Ergebnissen der Entwickler nicht interessiert ist, die aber für das Training dennoch von Vorteil sein können [Zhang et al., 2014] [Caruana, 1997] [Ruder, 2017].

In Kapitel 3.6 wird auf dieses Paradigma nochmals genauer eingegangen.

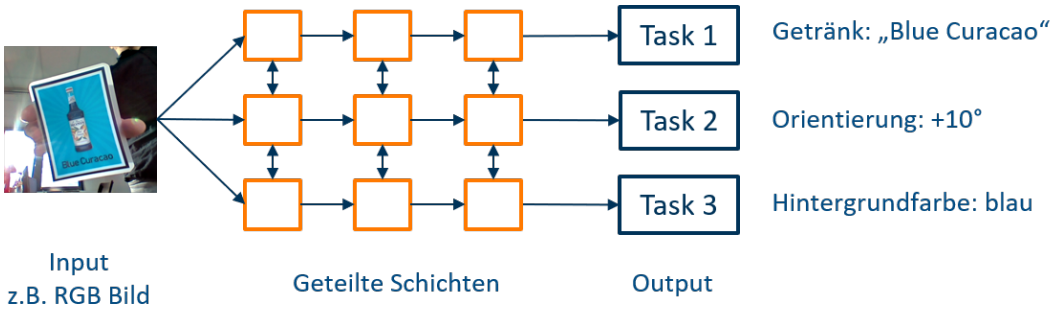


Abbildung 3.4: Schematische Darstellung des Soft Parameter Sharing
*Beim Soft Parameter Sharing ist eine Trennung zwischen Schichten, die von mehreren Tasks verwendet werden und Schichten, die nur für einen Task verwendet werden, nicht direkt sichtbar und auch nicht vom Entwickler festgelegt. Hier sind die Anteile, inwiefern Schichten zu welchem Task beitragen, adaptiv bestimmt oder gelernt [*Misra *et al., 2016]. Einfache Constraints zwischen den Schichten, welche zum Beispiel ähnliche Features erzeugen sollen, sind in der Abbildung als Doppelpfeile dargestellt.*

3.1.4 Vorteile des Multi-Task Learning

Im Folgenden wird auf allgemeine Vorteile des Multi-Task Learning, unabhängig von gewählter Architektur oder dem Netzwerkaufbau, eingegangen. Eine genauere Betrachtung der Vor- und Nachteile der einzelnen Multi-Task-Architekturen folgt in Abschnitt 3.2.

Implizite Augmentierung

Wird ein Multi-Task-System auf der selben Datengrundlage wie ein Single-Task-System trainiert, stehen dem Multi-Task-System im Allgemeinen durch die verschiedenen Tasks mehr Labels für die Input-Daten zur Verfügung. Die Tasks haben nach [Ruder, 2017] oftmals unterschiedliche Rauschmuster in den Labels. Da das Multi-Task-System beim Training eine Feature-Repräsentation sucht, welche alle Tasks möglichst gut löst, können rauschunempfindlichere Features gefunden werden, da diese invariant gegenüber dem Rauschen aller Tasks sein müssen. Man spricht daher auch von einer *impliziten Augmentierung* der Daten [Ruder, 2017].

Besseres Training von Tasks mit knapper Datengrundlage

Neben der impliziten Augmentierung durch mehr verfügbare Labels auf den selben Daten, kann Multi-Task Learning auch helfen, wenn für einen Task die Datengrundlage nur sehr klein ist und sich damit kein komplett eigenes System trainieren lassen würde. Durch die Hinzunahme von ähnlichen Daten, zum Beispiel von öffentlich verfügbaren Datensätzen, welche jedoch für andere Tasks gelabelt sind, kann ein System trainiert werden, was sowohl den Task mit guter Datengrundlage löst, als auch den Task mit geringer Datengrundlage. Durch die Gesamtheit der Daten können damit der geteilte Netzwerkteil und anhand der taskspezifischen Beispiele dann die jeweiligen Köpfe des Netzwerks gut trainiert werden [Long und Wang, 2015]. Laut [Ruder, 2017] reduziert Multi-Task Learning durch die vergrößerte Daten- bzw. Labelgrundlage zusätzlich das Overfitting.

Eavesdropping

In der Literatur wird als Vorteil für das Lernen von kombinierten Aufgaben auch das sogenannte *Eavesdropping* (*Lauschen*) genannt. Wird ein Multi-Task-System (auf einem oder mehreren Datensätzen) trainiert, bilden sich für beide Tasks gewisse nützliche Features aus. Hierbei kann jeder Task auch auf Features zugreifen, die für den anderen Task hilfreich sind (*Lauschen*). Ist nun beispielsweise eine Menge an Features G hilfreich, um den Task A zu lösen, aber lassen sich diese Features leichter durch Trainingslabels von Task B finden, kann ein System den Task A leichter lösen, wenn es zusätzlich auf Task B (als Multi-Task-System) trainiert wird [Ruder, 2017] [Abu-Mostafa, 1990].

Weniger Overhead und Latenz durch Kopiervorgänge

Ein weiterer Vorteil des Multi-Task Learning, welcher sich beispielsweise in einem robotischen Kontext ergibt, ist die geringere Anzahl an notwendigen Datenübertragungen. Klassifiziert ein System einen Bildausschnitt gleichzeitig als Person und schätzt dessen Orientierung, tritt ein geringerer Datenfluss auf, als bei einem sequentiellen System. Hierbei müsste zuerst ein System eins klassifizieren, welche Bildausschnitte Personen enthalten. Anschließend müssen diese Bildausschnitte dann an ein System zwei weitergeleitet werden. Dies führt dazu, dass jeder positiv klassifizierte Bildausschnitt mehrfach übertragen werden muss, was bei einem Multi-Task-System nicht der

Fall wäre. Gerade bei einem Robotersystem, wie es am Fachgebiet NIKR eingesetzt wird, machen Datentransporte dieser Art eine hohe Belastung der Rechner und des Netzwerks aus.

3.2 Multi-Task-Basisarchitekturen

Die in Kapitel 3.1.2 vorgestellten Basisarchitekturen, *Hard Parameter Sharing* und *Soft Parameter Sharing*, stellen die Grundlage vieler Multi-Task-Systeme dar. In den folgenden Abschnitten werden diese genauer anhand ihrer Vor- und Nachteile charakterisiert.

3.2.1 Hard Parameter Sharing

Die exemplarische Hard-Parameter-Sharing-Struktur aus dem vorherigen Abschnitt, Abbildung 3.3, besteht aus drei geteilten Schichten und zwei taskspezifischen Schichten und ist darauf ausgelegt, gleichzeitig drei verschiedene, beispielhaft gewählte Probleme zu lösen. Anhand der Struktur sind die Vor- und Nachteile des Hard Parameter Sharing gut erkennbar.

Vorteile von Hard Parameter Sharing

Hard-Parameter-Sharing-Netzwerke sind einfach zu implementieren, da, wie beim Single-Task Learning, lediglich Schichten miteinander in einer Feed-Forward-Architektur verschaltet werden, ohne, dass weitere, spezielle Architekturmerkmale verwendet werden müssen.
Außerdem sind Hard-Parameter-Sharing-Systeme durch ihren geteilten Feature-Extraktor performanter als die Verwendung von geteilten, Single-Task-Netzwerken, da im Normalfall bei einer Multi-Task-Architektur mit Hard Parameter Sharing die Teile des Netzwerks, welche von allen Tasks verwendet werden, mehr Gewichte und Rechenzeit in Anspruch nehmen, als die Teile, welche nur für einen spezifischen Task verantwortlich sind. Somit nimmt die Rechenzeit mit steigender Anzahl an Tasks nur geringfügig zu [Ruder, 2017]. Ein ResNet18 [He et al., 2016] besitzt beispielsweise ca. 11 Millionen Gewichte. In der Ausgangsschicht stehen dann 512 Features bereit. Für einen binären Klassifikationskopf sind also, im einfachsten Fall, $2 \cdot 512$ Gewichte + 2 Biasgewichte

nötig. Hierbei ist erkennbar, dass die Anzahl der binären Klassifikations-Tasks nur einen marginalen Einfluss auf die Gesamtrechenzeit hat.

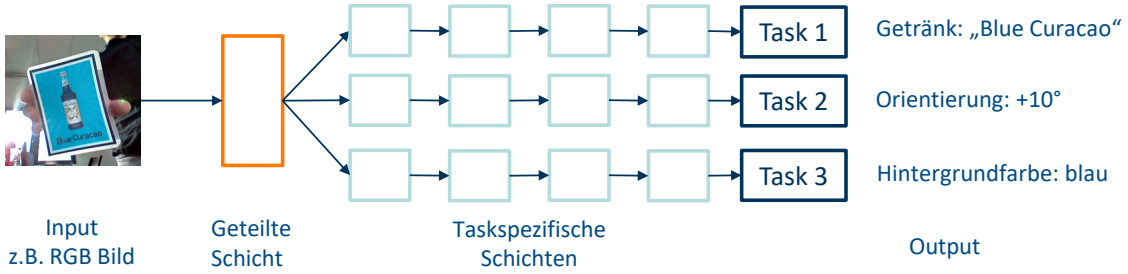
Probleme durch *Under-Transfer* und *Over-Transfer*

Nachteilig ist bei der Verwendung von Hard-Parameter-Sharing-Modellen, dass der Entwickler die genaue Struktur und damit den Übergang von geteilten zu taskspezifischen Schichten, selbst festlegen muss. Die Wahl des Aufteilungspunktes ist keine triviale Entscheidung. Wird dieser zu früh oder zu spät gesetzt, können *Under-Transfer* und *Over-Transfer* auftreten. Exemplarisch sind in Abbildung 3.5 die zwei Extremfälle der bisherigen Hard-Parameter-Sharing-Netzwerkarchitektur dargestellt, anhand derer die zwei Probleme gut nachvollzogen werden können.

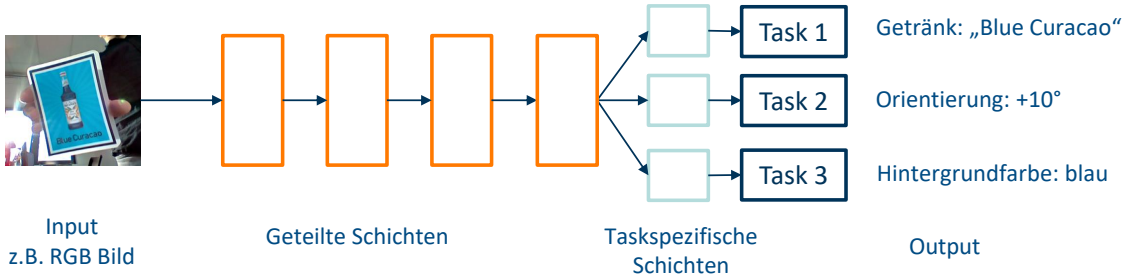
Als Under-Transfer wird bezeichnet, dass ein System nicht genügend Austausch zwischen den Tasks erlaubt und damit das Gesamtsystem nicht die volle Leistung erreichen kann [Long und Wang, 2015]. Hierbei ist der Aufteilungspunkt im Netzwerk zu früh gesetzt, wie in Abbildung 3.5a erkennbar. Der Extremfall wären getrennte Netze, welche unabhängig ihre Tasks lösen. Sie teilen sich somit nur den selben Input, aber keine Gewichte. Hierbei erreichen beide Netzwerke zwar ihre optimale Single-Task-Performance, es gibt jedoch keine positive Beeinflussung des Gesamtsystems durch Ausnutzung des geteilten Wissens und keine Reduktion der Rechenzeit, wie es in Kapitel 3.1.4 vorgestellt wurde.

Im Gegensatz zu Under-Transfer existiert auch Over-Transfer (oder Negative-Transfer), welcher beschreibt, dass zu wenig Netzwerkleistung in den taskspezifischen Schichten zur Verfügung steht. Der Grund hierfür ist, dass im Netzwerk erst zu spät von den geteilten Schichten zu den taskspezifischen Schichten übergegangen wird. Es wird also zu viel gemeinsam und zu wenig getrennt gelernt. Dies führt zwangsläufig zu einer schlechteren Netzwerkperformance als bei den Single-Task-Systemen, falls sich die Tasks stark unterscheiden [Ruder, 2017]. Dargestellt ist eine Architektur, welche dieses Problem aufweist, in Abbildung 3.5b.

Es ist erkennbar, dass aus diesen beiden Gründen eine geeignete Wahl der Netzwerkarchitektur sehr wichtig ist, um die Tasks entsprechend ihrer Komplexität nicht zu früh, aber auch nicht zu spät aufzuteilen.



(a) Netzwerk mit suboptimaler Leistung aufgrund von Under Transfer



(b) Netzwerk mit Problemen durch Over Transfer

Abbildung 3.5: Darstellung zweier beispielhafter Netzwerke mit Problemen durch Under- bzw. Over-Transfer
Wird bei einem Multi-Task-System die Aufteilung von den geteilten hin zu den taskspezifischen Schichten zu spät oder zu früh vorgenommen, bilden sich entweder nicht genügend taskspezifische Filter aus (Abb. 3.5b), oder es wird das geteilte Wissen nicht vollständig ausgenutzt (Abb. 3.5b).

3.2.2 Soft Parameter Sharing

Im Gegensatz zum Hard Parameter Sharing wird beim Soft Parameter Sharing nicht direkt festgelegt, wo das Netzwerk genau aufgeteilt wird.
Oftmals werden hier unabhängige Netzwerke für die einzelnen Tasks als Ausgangspunkt für den Entwurf eines Muti-Task-Systems verwendet. Diese Netzwerke werden dann miteinander verbunden, um einen Wissensaustausch zwischen den Tasks zu ermöglichen. In frühen Varianten des Soft Parameter Sharing werden hierfür Constraints eingesetzt, die zum Beispiel ähnliche Features in den einzelnen Schichten für alle [Evgeniou et al., 2005] oder nur ähnliche [Argyriou et al., 2008] Tasks erzwingen. Diese Constraints sind

in der schematischen Abbildung 3.4 bei den geteilten Schichten erkennbar und sorgen dafür, dass nur jene Features ausgewählt werden, welche deskriptiv für verschiedene Aufgaben des Netzwerks sind. Diese Features sind dann oftmals rauschärmer und sorgen für eine bessere Generalisierbarkeit [Ruder, 2017].

Im Gegensatz zu diesen einfachen Constraints existieren aber auch viele fortschrittliche Methoden für den Wissenstransfer zwischen den Tasks, auf welche in Kapitel 3.3.2 näher eingegangen wird.

Vorteile des Soft Parameter Sharing

Im Gegensatz zum Hard Parameter Sharing muss beim Soft Parameter Sharing nicht festgelegt werden, wo das Netzwerk von geteilten zu taskspezifischen Schichten aufgetrennt wird, da der Wissensaustausch während des Trainings adaptiv eingestellt wird. Dies führt zu weniger Aufwand bei der Suche nach der passenden Netzwerkstruktur [Ruder, 2017].

Nachteile des Soft Parameter Sharing

Das größte Problem des Soft Parameter Sharing ist deutlich in der Grundstruktur aus Abbildung 3.4 sichtbar. Durch das Verbinden von mehreren Netzwerken ergibt sich eben keine Reduktion der Gewichte, wie dies beim Hard Parameter Sharing der Fall ist. Durch zusätzliche Constraints ist die Anzahl der Gewichte sogar höher, als bei der Verwendung von unabhängigen Single-Task-Netzwerken. Daher bietet sich das Soft Parameter Sharing für den Einsatz auf einer mobilen Plattform weniger an, da der Hauptvorteil des Multi-Task Learning verloren geht.

3.3 Multi-Task-Architekturen am Beispiel

Nachdem die grundsätzliche Vorgehensweise des Hard- und Soft Parameter Sharing in den vorangegangenen Kapiteln behandelt wurde, werden nun jeweils zwei exemplarische Vertreter dieser beiden Varianten vorgestellt.

3.3.1 Hard-Parameter-Sharing-Architekturen

Im Folgenden soll nun auf zwei konkrete Vertreter der Hard-Parameter-Sharing-Architekturen eingegangen werden. Die beiden Architekturen unterscheiden sich anhand des Abstraktionsniveaus der verwendeten Features. Das Verfahren von [Zhang et al., 2014] folgt eher der Grundstruktur aus Abbildung 3.3 in Verbindung mit Convolutional Neural Networks und nutzt damit lediglich Features einer Abstraktionsstufe. Hierbei wird von *Single-Scale-Netzwerken* gesprochen. Im Gegensatz dazu nutzt die Architektur des Verfahrens *HyperFace* von [Ranjan et al., 2019] eine Feature-Repräsentation aus verschiedenen Abstraktionsstufen (*Multi-Scale*), um nochmals bessere Detektionsergebnisse zu erreichen.

Single-Scale-Deep-Multi-Task-Learning

[Zhang et al., 2014] verbessern ihre bestehende Landmarkendetektion auf Gesichtsausschnitten durch die Verwendung von Multi-Task Learning. Zusätzlich zu den fünf Landmarken (Nase, Mundwinkel und Augen) schätzen sie als Auxillary Tasks (siehe Kapitel 3.6) neben Geschlecht und Orientierung auch, ob die Person lächelt oder eine Brille trägt. Durch diese zusätzlichen Informationen kann das Netzwerk die Hauptaufgabe der Landmarkendetektion besser lösen und erreicht eine Verbesserung des Anteils an richtig geschätzten Landmarken von 65% (ohne Auxillary Tasks) zu 75% (mit allen vier Auxillary Tasks).
Die Netzstruktur ist in Abbildung 3.6 abgebildet. Es ist die Ähnlichkeit zur Grundstruktur eines Hard-Parameter-Sharing-Systems aus Abbildung 3.3 erkennbar, wobei lediglich die geteilten Schichten als Faltungsschichten und die taskspezifischen Schichten als vollverschaltete Schichten ausgeführt sind. Da diese Architektur lediglich Features aus einem Abstraktionsniveau verwendet, wird es der Klasse der *Single-Scale-Netzwerke* zugeordnet.
Problematisch bei dieser Architektur ist jedoch die Festlegung des Punktes, ab dem die Trennung der Pfade im Netzwerk stattfindet und damit der Wahl des Abstraktionsniveaus der Features für die taskspezifischen Schichten. Wird die Aufteilung zu nah beim Output gewählt, sind die Features nicht spezifisch genug, um die einzelnen Tasks zu lösen. Damit leidet das System unter *Over-Transfer* (siehe Kapitel 3.2.1). Wird der Trennpunkt zu nah beim Input gewählt, wird das geteilte Wissen nicht ausgenutzt und

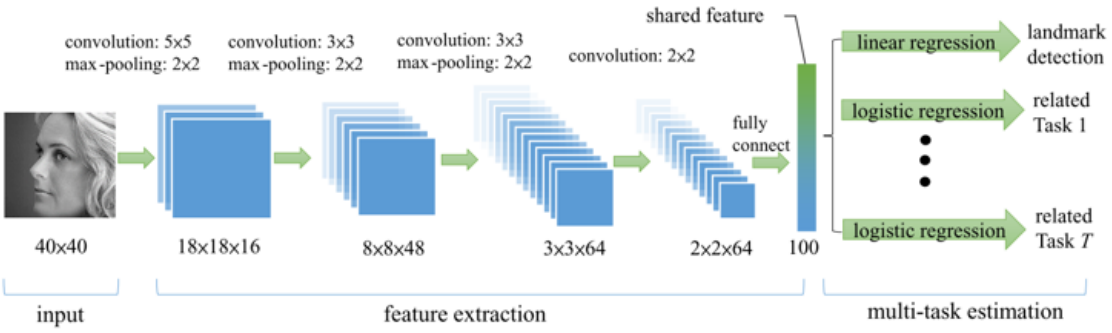


Abbildung 3.6: Netzwerkstruktur zum Single-Scale-Deep-Multi-Task-Verfahren von [Zhang et al., 2014]
Aus dem Input werden durch mehrere Faltungsschichten Features extrahiert, welche für alle Aufgaben genutzt werden. Die Feature Maps der letzten Faltungsschichten (2x2x64 = 256 Features) werden dann als Eingabe für eine geteilte, vollverschaltete Schicht verwendet. Die Ausgabe dieser wird dann von den taskspezifischen Schichten verwendet, um die Netzwerkausgabe zu erzeugen. Das Prinzip des Hard Parameter Sharing ist dabei gut erkennbar. Entnommen aus [Zhang et al., 2014].

es tritt *Under-Transfer* auf (siehe Kapitel 3.2.1). Dennoch kann eine solche Architektur gut für die Problemstellung dieser Masterarbeit entworfen werden.

Multi-Scale-Deep-Multi-Task-Learning

Um das Problem des festen Abstraktionsniveaus der Features der Single-Scale-Netzarchitektur aus dem vorangegangenen Kapitel 3.3.1 zu reduzieren, nutzen [Ranjan et al., 2019] eine verbesserte Netzwerkstruktur, welche in Abbildung 3.7 abgebildet ist. Als Feature-Extraktor wird ein vortrainiertes Neuronales Netzwerk (AlexNet [Krizhevsky et al., 2012] oder ResNet [He et al., 2016], siehe Kapitel 2.1) verwendet. Aus diesem werden in verschiedenen Tiefen Features extrahiert, wodurch die geteilte Feature-Repräsentation sowohl lokale Features aus den frühen Schichten, als auch abstraktere Features aus den späten Schichten enthält. Hier spiegelt sich die *Multi-Scale-Vorgehensweise* dieses Ansatzes wieder.
Der komplette Block zur Feature-Extraktion, inklusive der ersten vollverschalteten Schicht (Fc_full), entspricht somit den geteilten Schichten im Hard-Parameter-Sharing-Paradigma. Die kombinierten Features werden dann auf je eine vollverschaltete Schicht

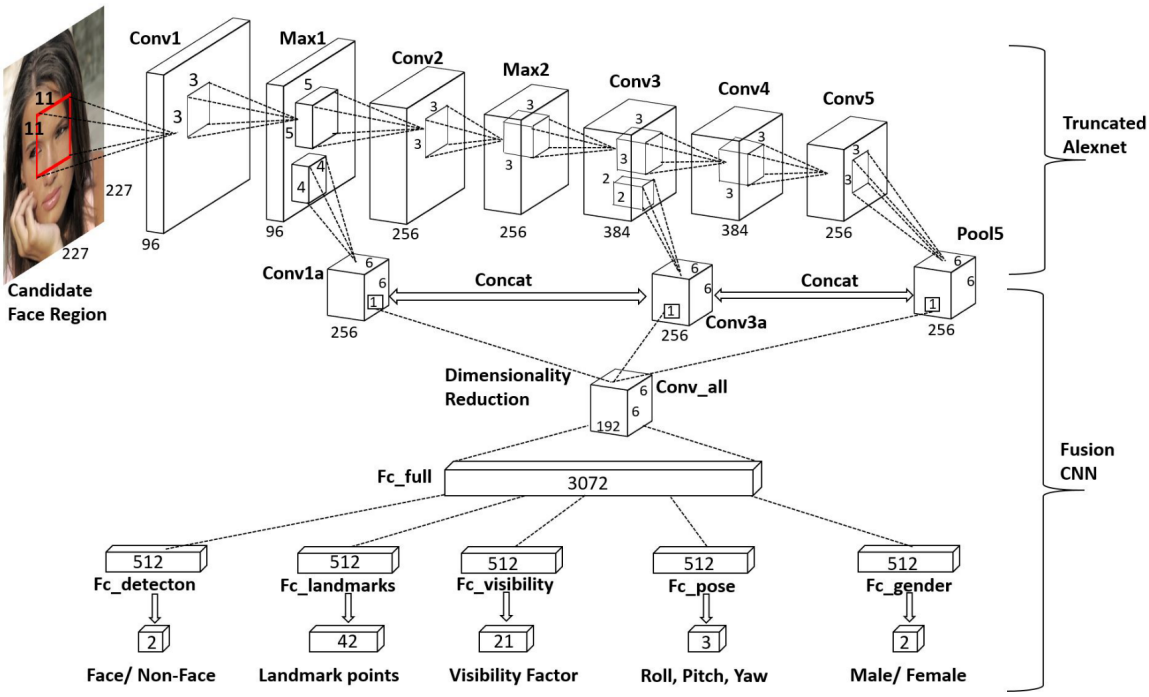


Abbildung 3.7: Netzwerkstruktur zum Verfahren *HyperFace* von [Ranjan et al., 2019]

Aus dem Input werden durch mehrere Faltungsschichten Features extrahiert, welche für alle Aufgaben genutzt werden. Anstatt nur die Features der letzten Schicht zu verwenden, werden diese bei HyperFace mit zusätzlichen Features von früheren Netzwerkschichten konkateniert. Dadurch können die einzelnen Tasks sowohl auf geteilte Features mit einer höheren Abstraktion, als auch auf geteilte Features mit einem höheren Ortsbezug zurückgreifen. Entnommen aus [Ranjan et al., 2019].

für die einzelnen Ausgaben aufgeteilt.

Durch diese Architektur legten [Ranjan et al., 2019] den State of the Art im Bereich der Landmarkendetektion mit einer *mean average precision* von 99,4% auf dem Gesichtsdatensatz AFW und 96,2% auf dem PASCAL-Datensatz fest. Hierbei ist jedoch auch anzumerken, dass der HyperFace-Ansatz aus der Publikation von [Ranjan et al., 2019] sehr rechenaufwendig und somit nicht echtzeitfähig auf einer mobilen Plattform einsetzbar ist. Alleine im Übergang von der letzten Merkmalskarte zum geteilten Feature-Vektor (Fc_full) sind über 21 Millionen Gewichte vorhanden (192x6x6x3072). Daher muss für eine Verwendbarkeit im Rahmen dieser Masterarbeit eine geringere

Anzahl an Feature Maps und deren Größe eingestellt werden.

3.3.2 Soft-Parameter-Sharing-Architekturen

Nach den Hard-Parameter-Sharing-Architekturen werden nun zwei weitere Soft-Parameter-Sharing-Architekturen vorgestellt. Das Verfahren von [Ruder et al., 2017] baut hierbei auf dem von [Misra et al., 2016] auf.

Cross-Stitch Networks

[Misra et al., 2016] verbinden mehrere Tasks, indem durch sogenannte *Cross-Stitch Units* Features zwischen den Tasks ausgetauscht werden können. Bei einer solchen Einheit wird der Vorwärtspfad in den jeweiligen Netzwerken der einzelnen Tasks unterbrochen und eine Verbindung zwischen den Tasks eingeführt, vgl. Abbildung 3.8. Dabei werden an diesen Stellen im Netzwerk für jeden Task die Feature Maps erzeugt, indem die Feature Maps der vorherigen Schicht von allen Tasks linear kombiniert werden. Existieren F Feature Maps und T Tasks, existiert für jeden Task eine α-Matrix der Dimension $T \times F$. Es werden also für jeden Task F Feature Maps erzeugt, indem die entsprechenden Feature Maps von allen Tasks in der vorherigen Netzwerkschicht der T Tasks linear kombiniert werden. Die hierfür verwendeten Cross-Stitch Units sind in Abbildung 3.9 dargestellt. Die α-Wichtungsfaktoren werden dabei vom Netzwerk gelernt, wodurch das Netzwerk entscheiden kann, ob eher ein Feed-Forward oder eher ein Wissensaustausch an der Stelle der Cross-Stitch Unit vorteilhaft wäre [Misra et al., 2016].
Durch dieses Vorgehen muss der Entwickler den Punkt vom Übergang zwischen geteilten und spezifischen Schichten nicht festlegen und es soll automatisch ein optimales Maß an Wissen zwischen den Tasks ausgetauscht werden.

Sluice Networks

Kombiniert man die Idee, den Wissensaustausch durch Cross-Stitch Units lernbar zu gestalten, mit der Idee, für die Entscheidungsfindung Features aus unterschiedlichen Abstraktionsniveaus des Netzwerks zu verwenden, ergibt sich hierbei das Konzept der Sluice Networks von [Ruder et al., 2017].

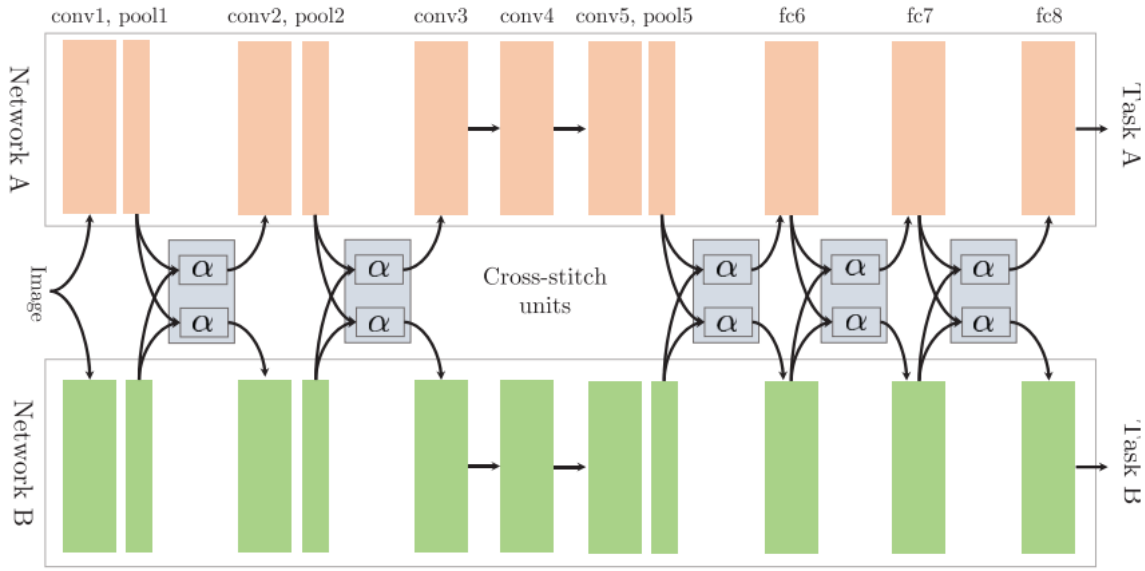


Abbildung 3.8: Netzwerkstruktur eines Cross-Stitch Networks von [Misra et al., 2016]
Zwei Tasks werden miteinander zu einem Multi-Task-System kombiniert, indem Vorwärtspfade in den Netzwerken aufgetrennt und durch Cross-Stitch Units ersetzt werden. Diese lernen im Trainingsprozess, inwiefern an einer bestimmten Stelle Wissen zwischen den Tasks ausgetauscht werden soll oder nicht. Eine Cross-Stich Unit verbindet hierbei die einzelnen Netzwerkausgaben, wichtet diese jeweils mit $\alpha_{i,j}$ *und summiert sie auf. Ein Alpha in der Abbildung steht somit immer für einen Vektor* $\underline{\alpha}$ *mit zwei Elementen. Entnommen aus [*Misra *et al., 2016].*

Die α Parameter sind hier wieder die Linearkombinationsparameter für die Zusammensetzung des Inputs einer Schicht aus den Outputs von allen Tasks aus der Schicht vorher. Die β Parameter sind neu eingeführte Linearkombinationsparameter. Diese geben an, wie die letzte Feature Map, welche jeweils spezifisch für einen Task ist, aus den Ausgaben der vorherigen Schichten zusammengesetzt werden soll. Dargestellt ist eine solche Architektur in Abbildung 3.10.

Sowohl bei den Cross-Stitch Networks als auch bei den Sluice Networks ist erkennbar, dass hierbei viel komplexere Netzwerkstrukturen entstehen, als bei den Hard-Parameter-Sharing-Modellen. Da für jeden Task ein eigenständiges Netzwerk erforderlich ist, ist die Rechenkomplexität vergleichsweise hoch. Weiterhin müssen die oben genannten

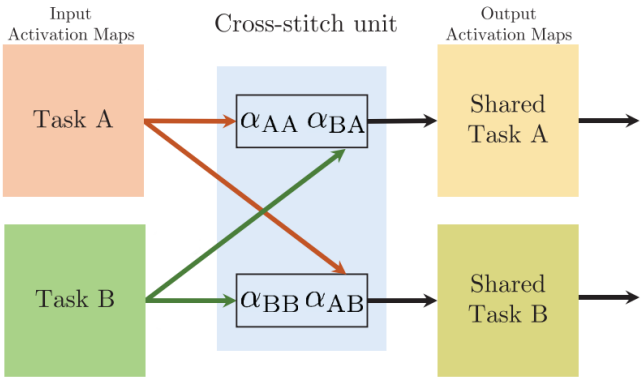


Abbildung 3.9: Aufbau einer Cross-Stitch Unit von [Misra et al., 2016]
Die Feature Maps für die Tasks werden bei einer Cross-Stitch Unit errechnet, indem die Feature Maps aller Tasks aus der vorherigen Schicht miteinander linear kombiniert werden. Die Cross-Stitch Unit behält die Anzahl an Feature Maps konstant. Die Wichtungsfaktoren α werden während des Trainings adaptiert. Sind die Parameter α_{BA} und α_{AB} gleich null und α_{AA} und α_{BB} gleich eins, entspricht die Cross-Stitch Unit einem Standard Vorwärtspfad für die jeweilige Feature Map. Entnommen aus [Misra et al., 2016].

Parameter (α und β) während des Trainings optimiert werden. Daher bieten diese Netzwerkstrukturen einen schwierigeren Einstieg in das Themenfeld des Multi-Task Learning.

3.4 Aufstellung der Loss-Funktion bei Multi-Task Learning

Nachdem nun verschiedene Architekturen sowie deren Vor- und Nachteile beschrieben wurden, wird jetzt auf die für das Training essentielle Loss-Funktion eingegangen. Multi-Task-Systeme lassen sich grundsätzlich genau wie Single-Task-Systeme über den Backpropagation-Algorithmus trainieren. Hierzu wird zuerst für jeden Task eine eigene Loss-Funktion aufgestellt. Das Training erfolgt dann durch geeignete Verrechnung der taskspezifischen Loss-Funktionen zu einer Gesamt-Loss-Funktion. Hierbei wird allgemein folgende Formulierung verwendet [Lu et al., 2017]:

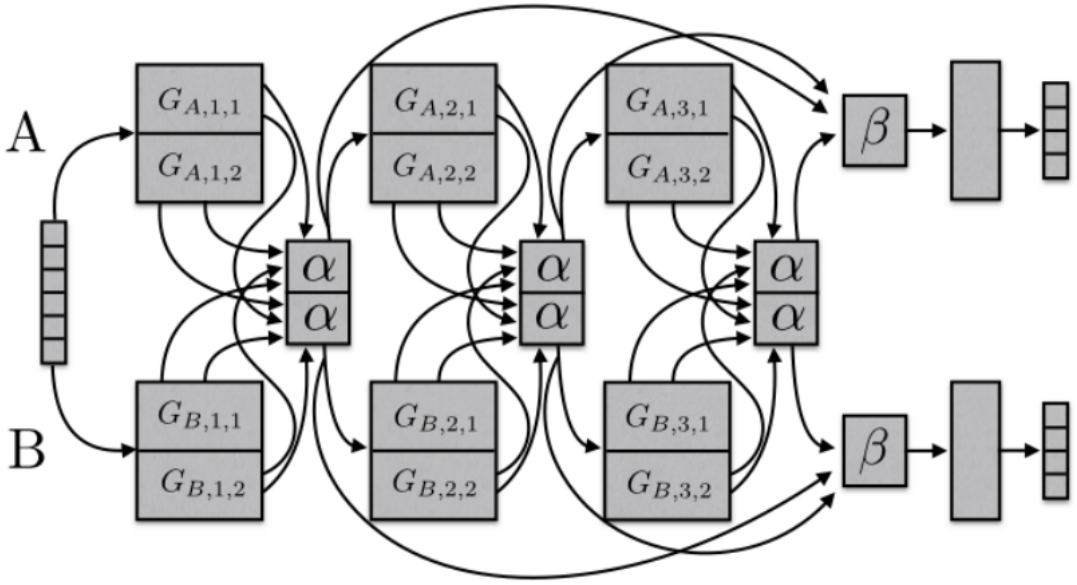


Abbildung 3.10: Netzwerkstruktur eines Sluice Networks von [Ruder et al., 2017] *Bei den Sluice Networks werden zwei Netzwerke miteinander zu einem Multi-Task-System kombiniert, indem Netzwerkschichten jeweils durch Cross-Stitch Units mit Wichtungen α verbunden werden und die finale Entscheidungsfindung durch gewichtete Features aus unterschiedlichen Abstraktionsebenen mit den Wichtungsfaktoren β ausgeführt wird. Entnommen aus [Ruder, 2017].*

$$L_{multi\_task}(t) = \sum_{i=1}^{N} \alpha_i(t) \cdot L_{single\_task\_i} \tag{3.1}$$

$L_{multi\_task}(t)$	Gesamt-Loss des Multi-Task-Systems
$L_{single\_task\_i}(t)$	Loss für Task i
N	Anzahl der Tasks
$\alpha_i(t)$	Zeitabhängiger Wichtungsfaktor für Task i

Es ist erkennbar, dass sich das Problem der Kombination der taskspezifischen Loss-Funktionen zu einem Gesamt-Loss auf eine Suche nach optimalen Wichtungsfaktoren $\alpha_i(t)$ zurückführen lässt. Hierzu gibt es verschiedene Möglichkeiten.

Konstante Wichtungsfaktoren

Werden die Wichtungsfaktoren $\alpha_i(t)$ während des gesamten Trainings konstant gelassen, müssen am Standard-Trainingsablauf keine Veränderungen vorgenommen werden. Nachteilig ist jedoch, dass Experimente für verschiedene Wichtungen durchgeführt werden müssen, um die beste Wichtung zu finden. Zudem ist der einfachste Ansatz, eine gleiche Gewichtung von allen Tasks, zwar möglich, jedoch oft nicht die optimale Gewichtung [Lu et al., 2017]:

$$\alpha_i(t) = \frac{1}{N} \; \forall i \; \forall t \tag{3.2}$$

$\alpha_i(t)$	Wichtungsfaktor für Task i
N	Anzahl der Tasks

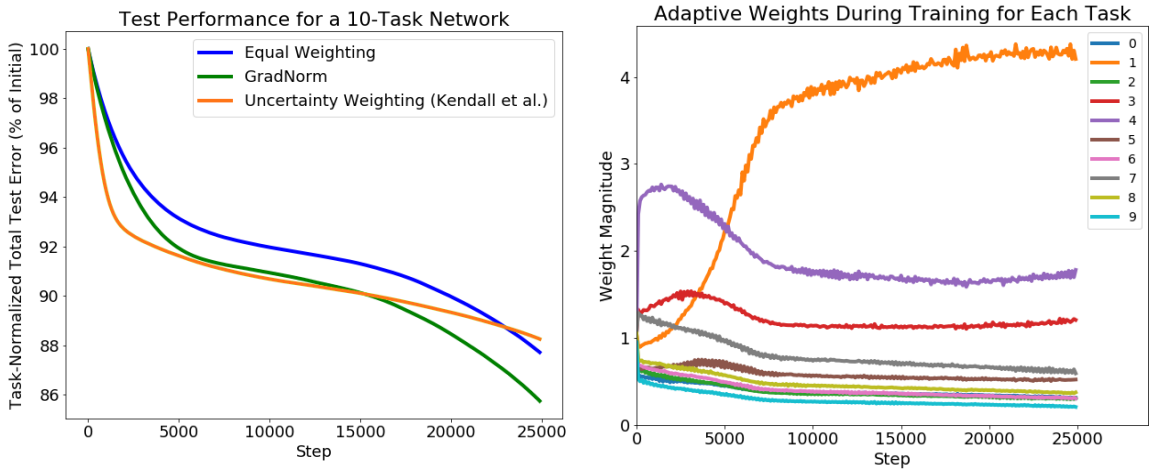
Wichtungsfaktoren abhängig von der Taskunsicherheit

In der Literatur findet sich eine Vielzahl an adaptiven Verfahren zur Bestimmung der Loss-Gewichtung. [Lu et al., 2017] implementierten beispielsweise die Taskgewichtungen als zeitlich veränderliche Paramter, die während des Trainings anhand von Unsicherheiten der Tasks bestimmt werden. Hiermit ergibt sich eine optimale Taskgewichtung ohne aufwendige Versuchsreihen durchführen zu müssen. Weiterhin zeigen die Autoren auch, dass eine während des Trainings variable Taskgewichtung zu besseren Ergebnissen führen kann, als die aufwendige Suche nach der besten konstanten Taskgewichtung. Laut [Lu et al., 2017] erreicht die am Trainingsende bestimmte, optimale Taskgewichtung schlechtere Ergebnisse, wenn diese für ein neues Training verwendet und über den Verlauf des Trainings konstant gelassen wird.

Wichtungsfaktoren durch Gradientennormierung

Andere Verfahren, wie GradNorm [Chen et al., 2017] versuchen, die Taskgewichtung so zu errechnen, dass möglichst alle Tasks *gleich schnell* lernen. Neben der Definition einer *Training Rate*, welche abhängig von initialem und aktuellem Loss-Wert angibt, wie schnell ein Task lernt, werden bei diesem Verfahren die Gradienten von der letzten geteilten Schicht zu den taskspezifischen Schichten normiert. Anhand dieser Normierungsfaktoren und Trainingsraten wird dann eine Taskgewichtung für jede Epoche berechnet.

In Abbildung 3.11 ist ein Vergleich zwischen der festen, gleichen Wichtung für alle Tasks mit den beiden genannten adaptiven Ansätzen dargestellt. Erkennbar ist in Abbildung 3.11, dass das Verfahren von [Chen et al., 2017] vielversprechende Ergebnisse für diese Masterarbeit liefern könnte, da eben keine intensive Suche nach der besten Taskgewichtung durchgeführt werden muss und dennoch gute Ergebnisse erzielt werden können.



(a) Fehlerkurven

(b) Taskgewichtungen bei GradNorm

Abbildung 3.11: Ergebnisse zur adaptiven Bestimmung der Taskgewichte.
In Abbildung 3.11b ist gut erkennbar, wie sich die Taskgewichtungen für 10 beispielhafte Tasks bei GradNorm im Laufe des Trainings adaptieren. Abbildung 3.11a zeigt durch Fehlerkurven für die drei genannten Verfahren, dass Uncertainty Weighting von [Lu et al., 2017] zwar schneller lernt, jedoch durch das Verfahren GradNorm von [Chen et al., 2017] oder eine konstante Wichtung bessere Endergebnisse erzielt werden können. Entnommen aus [Chen et al., 2017]

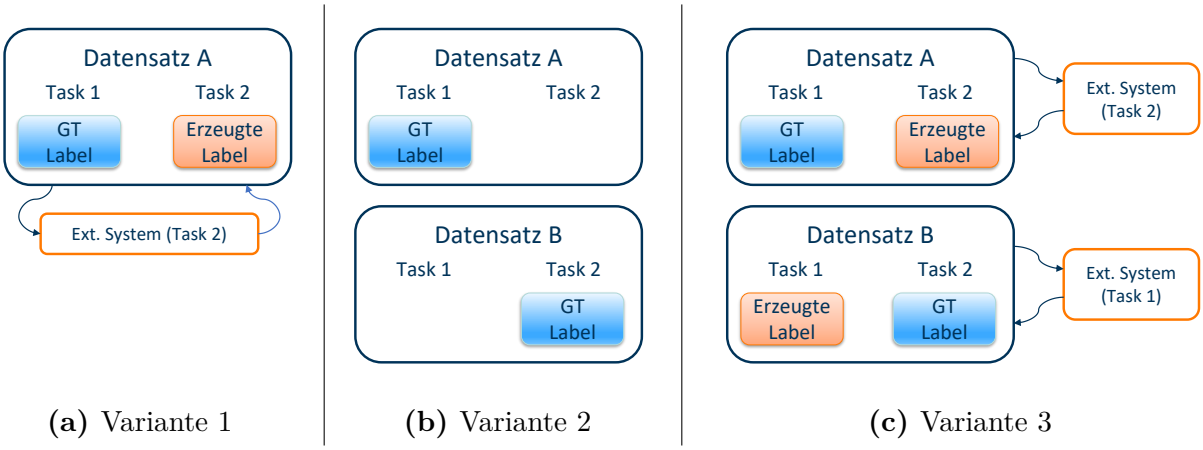
3.5 Umgang mit verschiedenen Datensätzen

Multi-Task Learning erfordert entweder den Einsatz von Datensätzen, in denen alle zu trainierenden Tasks vorhanden und gelabelt sind oder den Einsatz von verschiedenen Datensätzen, die dann in ihrer Gesamtheit alle Tasks abdecken. Dabei gibt es drei grundsätzliche Varianten, wie verschiedene Datensätze eingesetzt werden können

[Bornstein, 2018]. Diese sollen im Folgenden anhand eines Beispiels mit zwei abstrakten Datensätzen (A und B) und zwei Tasks (1 und 2) aus Abbildung 3.12 erläutert werden:

3.5.1 Verwendung von einem Datensatz

Die erste Möglichkeit, dargestellt in Abbildung 3.12a, wird oft in Verbindung mit dem in Kapitel 3.6 vorgestellten Auxillary-Task Learning verwendet. Hier wird ein Datensatz, dem bestimmte Labels fehlen, durch ein fertiges System, welches genau diese Labels erzeugt, ergänzt. So könnte beispielsweise ein Datensatz zur Personendetektion um Orientierungslabel ergänzt werden. Dazu wird ein fertiger Orientierungsschätzer, wie z.B. [Lewandowski et al., 2019b], eingesetzt. Vorteilhaft ist, dass der komplette Datensatz Labels enthält und somit keine Fälle von Trainingsdaten ohne Labels für manche Tasks auftreten können. Diese Variante hat jedoch den Nachteil, dass die prädizierten Labels nicht korrekt sein müssen und das Multi-Task-System daraufhin auf falschen Labels trainiert. Deshalb bietet sich diese Variante für die Aufgabenstellung dieser Masterarbeit weniger an, als die folgenden Methoden.



(a) Variante 1 **(b)** Variante 2 **(c)** Variante 3

Abbildung 3.12: Drei verschiedene Möglichkeiten, mit fehlenden Labels umzugehen
Um ein vollständiges Trainingsset für alle Tasks zu erhalten, können entweder externe Systeme verwendet werden, um eine vollständige Abdeckung eines Datensatzes mit Labels zu erreichen (a), mehrere Datensätze kombiniert werden (b) oder beide Herangehensweisen kombiniert werden (c)

.

3.5.2 Verwendung von mehreren Datensätzen

Die zweite Möglichkeit besteht darin, mehr als einen Datensatz zu verwenden, um für alle Tasks gelabelte Trainingsdaten verwenden zu können. Dies hat den Vorteil, dass das System nur mit Ground-Truth-Daten trainiert wird, welche richtig gelabelt sind. Jedoch sind die einzelnen Samples der Datensätze unvollständig, da nicht alle Labels in allen Datensätzen vorhanden sind. Dies ist auch in Abbildung 3.12b gut erkennbar. Zum erfolgreichen Training mit solch einer Datengrundlage gibt es verschiedene Lösungswege:

Asynchrones batchloses Training

Eine Variante ist der in der Literatur vorgestellte Ansatz des asynchronen Traininigs nach [Kokkinos, 2017]. Dabei wird nicht mehr von einer konstanten Batch Size ausgegangen, sondern es existiert für jeden Task eine gewünschte, virtuelle Batch Size. Die Trainingssamples werden einfach nacheinander präsentiert und für jeden Task wird gezählt, wieviele Eingangsdaten mit passenden Labels bisher präsentiert wurden. Entspricht die Anzahl der gewünschten, virtuellen Batch Size für den entsprechenden Task, wird ein Parameterupdate für diesen Task durchgeführt. Somit laufen nicht mehr alle Tasks synchron, sondern asynchron und abhängig von der Datensatzzusammenstellung. Diese Methode hat den Nachteil, dass eine komplett andere Herangehensweise an die Verarbeitung und die Trainingspipeline nötig ist, da nicht mehr mit fixen Batches gearbeitet wird. Dies kann zu geringerer Auslastung der Grafikkarten und zu höheren Trainingszeiten führen. Die genaue Umsetzung dieses Trainingsablaufes, vor allem der Umgang mit dem Grafikkartenspeicher in Form einer *Low-Memory Backpropagation*, kann bei [Kokkinos, 2017] nachgelesen werden.

Normales Batch-Training

Wird stattdessen klassisch mit fixen Batches gearbeitet, muss, wenn kein Trainingslabel verfügbar ist, der entsprechende Loss aus der Gesamtsumme ausmaskiert werden [Lu et al., 2017]. Abhängig von der gewählten Zusammensetzung der Trainingsdatensätze können so jedoch Probleme auftreten, falls in einem Batch keine oder zu wenige Beispiele für einen gewissen Task enthalten sind. Dadurch kann der Gradient bei der Optimierung instabil werden, weil er durch eine variierende Anzahl Elemente gemittelt wird. Dies kann zu einem schlechteren Endergebnis führen. Außerdem können sich die effektiven

Batchgrößen für die Tasks unterscheiden. Es muss dann Wert darauf gelegt werden, wie die Datensätze in Batches zusammengestellt werden.
Alles in allem bietet sich diese Variante in Verbindung mit der Kombination von mehreren Datensätzen am besten für diese Masterarbeit an, da verschiedene Datensätze vorhanden sind, welche in ihrer Gesamtheit alle gewünschten Tasks abdecken. In Kapitel 4.2.3 wird nochmals genauer auf die Zusammensetzung der Trainingsbatches eingegangen.

3.5.3 Alternative Herangehensweise an mehrere Datensätze

Die dritte Möglichkeit, mehrere Datensätze zu verbinden, ist die Kombination aus den zwei anderen Methoden, dargestellt in Abbildung 3.12c. Es werden mehrere Datensätze kombiniert, um eine vollständige Abdeckung der Tasks mit richtigen Labels zu erhalten und zusätzlich werden die in den einzelnen Datensätzen fehlenden Labels mittels eines externen Systems erzeugt. Somit kann das Training auf allen Datensätzen auf alle Labels zugreifen. Dies kombiniert die Vorteile der beiden Methoden aus Kapitel 3.5.1 (vollständige Abdeckung mit Labels) und 3.5.2 (Trainingsdaten enthalten gelabelte und nicht nur erzeuge Labels), ist jedoch auch am aufwendigsten in der Datenaufbereitung, da sowohl mit mehreren Datensätzen, als auch mit mehreren externen Systemen umgegangen werden muss.

Für diese Masterarbeit bietet sich, aufgrund der bestehenden Datenlage, entweder die reine Verwendung von mehreren Datensätzen (Kapitel 3.5.2) oder die Verwendung von mehreren Datensätzen mit mehreren externen Systemen (Kapitel 3.5.3) an. Aufgrund einer einfacheren Umsetzung soll die Wahl auf den Ansatz ohne externe Systeme fallen, da somit mehr Zeit für wichtige Multi-Task-Trainings und Experimente zur Verfügung steht.

3.6 Auxillary Multi-Task Learning

Wie schon in Kapitel 3.1.3 beschrieben, bezeichnet das *Auxillary Multi-Task Learning* ein Vorgehen, bei dem die Performance eines Systems auf einem oder mehreren Haupttasks verbessert werden soll, indem eine Kombination aus Haupt-Tasks und zusätzlichen

Auxillary Tasks trainiert wird. Die Idee hinter diesen Trainings ist, dass durch das Ausnutzen von domänenübergreifendem Wissen in Form von weiteren Labels ein besser generalisierendes Netzwerk entsteht. Auxillary Tasks können aus verschiedenen Bereichen kommen: Sowohl eine zusätzliche Klassifikation des Bildausschnittes, eine Attributeschätzung oder eine Landmarkendetektion sind hier denkbar [Zhang et al., 2014].
[Zhang et al., 2014] nutzen zur Verbesserung des Haupttasks *Landmarkendetektion in Gesichtern* folgende Auxillary Tasks:

- Klassifikation: Trägt die Person eine Brille?
- Klassifikation: Welches Geschlecht hat die Person?
- Klassifikation: Lacht die Person?
- Klassifikation: Gesichtsorientierung in fünf Klassen

Unter Verwendung der Netzwerkarchitektur aus Kapitel 3.3.1 erreichen die Autoren so eine Reduktion der falsch geschätzten Landmarken durch Verwendung dieser vier Auxillary Tasks von 35,62% auf 25,00%.

Abbruch des Lernens von den Auxillary Tasks

Ein wichtiger Einfluss für die Performance eines Systems aus Haupt- und Auxillary-Tasks ist neben der Auswahl der geeigneten Tasks das sogenannte *Task-Wise Early Stopping* [Zhang et al., 2014]. Diese Vorgehensweise besagt, dass es für das Training vorteilhaft sein kann, wenn die Nebentasks nicht alle bis zum Trainingsende mitlernen. Hierzu wird ein Kriterium eingeführt, welches bewertet, inwiefern ein Auxillary Task noch weiterlernt: Stagniert der Loss über eine gewisse Zeit, wird dieser Task abgebrochen. In Abbildung 3.13 ist gut erkennbar, wie durch dieses Stop-Schema der Fehler auf dem Haupttask, der Landmarkendetektion, viel stärker abfällt, als ein Training aller Tasks bis zum Ende.

3.7 Abschließendes Fazit zum State of the Art

Multi-Task Learning ist ein sehr breites Forschungsfeld. In diesem Kapitel wurde auf eine Vielzahl an Architekturen und Techniken als auch auf Probleme und deren Lösungsmöglichkeiten eingegangen. Oft existiert jedoch keine *beste* Lösung. Daher sollen

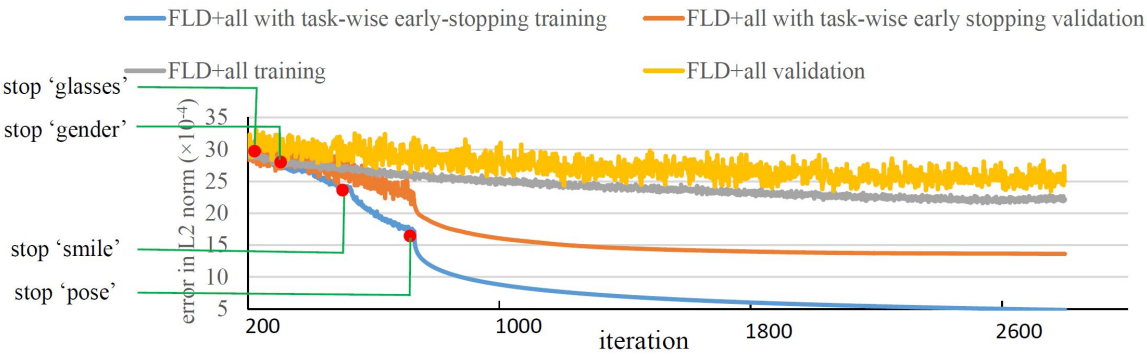


Abbildung 3.13: Trainings- und Validierungsfehler mit und ohne Task-Wise Early Stopping
*Dargestellt ist der euklidische Fehler für eine Landmarkendetektion in Gesichtern (engl.: Facial Landmark Detection, FLD). Erkennbar ist, dass durch ein Stoppen der Auxillary Tasks ein geringerer Fehler in der Landmarkendetektion erreicht wird. Entnommen aus [*Zhang *et al., 2014].*

im Rahmen dieser Masterarbeit die für die Aufgabenstellung (Personendetektion und Schätzung der Oberkörperorientierung) wichtigen Parameter experimentell bestimmt werden.

Somit müssen zusätzlich zu den normalen Trainingsparametern des Single-Task Learning, wie Lernrate, Epochenzahl oder Batchgröße, folgende, speziell für das Multi-Task Learning wichtige, Aspekte beachtet werden:

Architektur Für die Netzwerkarchitektur soll aufgrund der geringeren Rechenzeit und der einfacheren Implementierbarkeit der Fokus auf Hard-Parameter-Sharing-Modelle gelegt werden. Hierbei bieten sich sowohl Single-Scale-, als auch Multi-Scale-Ansätze, wie in Kapitel 3.3.1 vorgestellt, an.

Taskgewichtung Die Wahl der Taskwichtungen wird einen großen Einfluss auf die Leistung des Gesamtsystems zur Personenwahrnehmung haben. Werden konstante Wichtungsfaktoren verwendet, muss auf eine gründliche Analyse von diesen geachtet werden. Außerdem bietet sich ein Versuch der adaptiven, automatischen Herangehensweisen aus Kapitel 3.4 an.

Datensätze Aufgrund der Verfügbarkeit von verschiedenen Datensätzen soll eine Kombination aus mehreren Datensätzen verwendet werden. Diese werden lediglich kombiniert, ohne fehlende Labels durch externe Systeme zu ergänzen, vgl. Kapitel 3.5.2.

Auxillary Tasks Die Verwendung der Methoden aus dem Kapitel zu Auxillary Tasks (3.6) kann sich als sehr sinnvoll erweisen, bietet sich im Rahmen dieser Masterarbeit jedoch weniger an, da zusätzliche Tasks in den verwendeten Daten annotiert werden müssten, um das Training auf weitere Auxillary Tasks auszuweiten.

Mit diesen Ergebnissen aus dem State of the Art wird im folgenden Kapitel der Entwurf und die Umsetzung eines leistungsfähigen Multi-Task-Systems für die kombinierte Personendetektion und Oberkörperorientierungsschätzung vorgestellt.

Kapitel 4

Multi-Task-System zur Personenwahrnehmung

In diesem Kapitel wird der Entwurf und die Umsetzung des, im Rahmen dieser Masterarbeit realisierten, Multi-Task-Systems vorgestellt. Dieses führt auf einem Tiefenbild sowohl eine Personendetektion als auch eine Oberkörperorientierungsschätzung der erkannten Personen durch. Der aktuelle Stand am Fachgebiet NIKR ist hier eine sequentielle Verarbeitung: Personen werden über einen Detektor erkannt (wie z.B. [Lewandowski et al., 2019a]), im Bild ausgeschnitten und an einen Orientierungsschätzer ([Lewandowski et al., 2019b]) weitergereicht. Im Rahmen dieser Masterarbeit sollen beide Aufgaben gleichzeitig und mittels eines tiefen Neuronalen Netzes durchgeführt werden.
Dieses Kapitel thematisiert zu Beginn kurz die Einbindung des Systems in die Anwendung und den grundsätzlichen Systemaufbau. Daraufhin folgt eine Betrachtung des Trainingsablaufs des Multi-Task-Netzwerks. Hierbei wird sowohl auf die Datengrundlage als auch auf Netzwerkarchitekturen und Trainingsparameter eingegangen.

4.1 Das Multi-Task-System aus Anwendungssicht

Zu Beginn dieses Kapitels wird der Fokus auf das Multi-Task-System als Gesamtsystem gelegt. In Abbildung 4.1 ist daher dargestellt, wie das System zur Personenwahrnehmung, von der Sensorausgabe bis hin zur Detektion und Oberkörperorientierungsschätzung, auf einem Roboter eingesetzt werden soll.

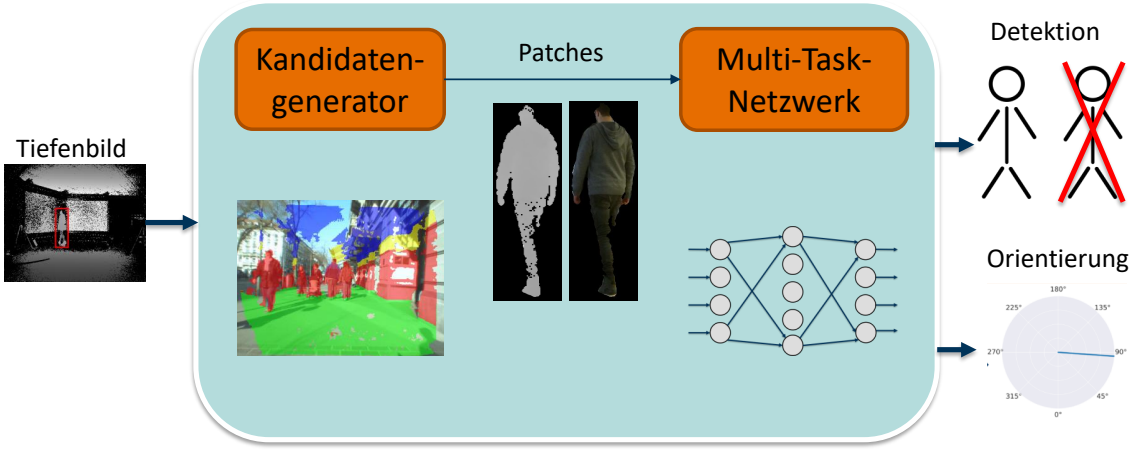


Abbildung 4.1: Übersicht der realisierten Verarbeitungspipeline zur Anwendung auf einem mobilen Roboter
*Das Tiefenbild der Kinect2 wird von einem Kandidatengenerator nach [*Jafari *et al., 2014] in mehrere 3D-Cluster geteilt. Diese werden dann, in Anlehnung an die Verarbeitung von [*Lewandowski *et al., 2019b], mithilfe einer Rückprojektion in hintergrundfreie Tiefenbildausschnitte (Patches) umgewandelt. Mit diesen kann dann vom Multi-Task-Netzwerk eine Klassifikation von Personen und eine Orientierungsschätzung durchgeführt werden.*

Hierbei wird das Eingangstiefenbild zuerst in eine Punktewolke umgewandelt. Ein Ansatz zur Bestimmung von potenziell interessanten Bereichen in der Punktewolke von [Jafari et al., 2014] erzeugt eine Vielzahl von 3D-Punktwolkenausschnitten aus der Eingangspunktewolke. Die erzeugten Punktwolkencluster werden mithilfe der Kameraparameter daraufhin in jeweils leere 2D-Tiefenbilder des Sensors zurückprojiziert und passend zugeschnitten. Somit liegt der Vordergrund, welcher sowohl aus einer Person oder einem Objekt bestehen kann, ohne Rand vor. Diese drei Verarbeitungsschritte sind in Abbildung 4.1 unter dem Namen *Kandidatengenerator* zusammengefasst. Durch diese Vorverarbeitung entsteht eine Vielzahl an hintergrundfreien Tiefenbildausschnitten (Patches) pro Eingangsbild. In Abbildung 4.2 sind sowohl die Ein- als auch die Ausgabe der Vorverarbeitung exemplarisch für eine Szene dargestellt.

Die entstandenen Bildausschnitte werden dann in eine feste Auflösung, abhängig vom gewählten Netzwerk, skaliert und so vom Deep-Learning-Netzwerk verarbeitet. Sie

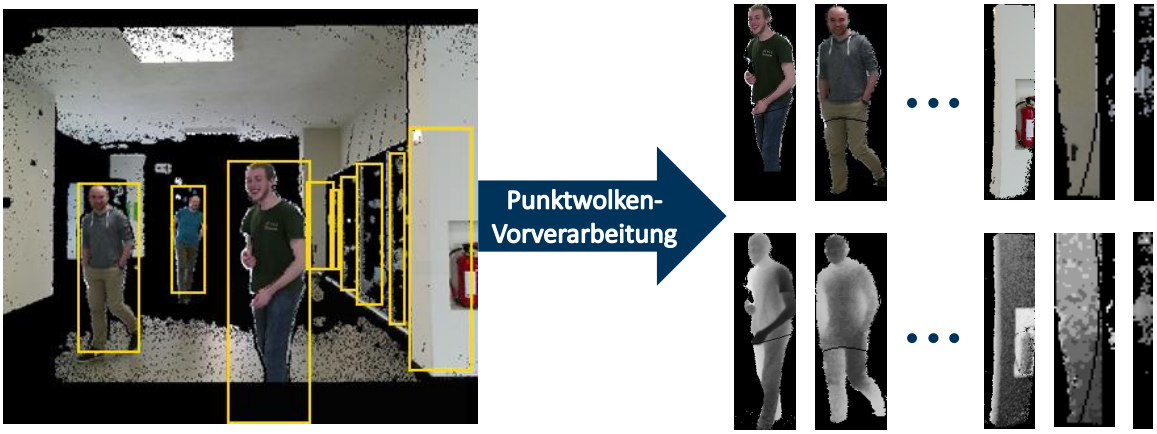


Abbildung 4.2: Ein- und Ausgabedaten der punktwolkenbasierten Generation von Kandidaten
Durch die Vorverarbeitung nach [Jafari et al., 2014] wird das Eingangstiefenbild in eine Punkewolke umgewandelt und in eine Vielzahl interessanter Regionen zerlegt. Die entstandenen 3D-Punkteregionen werden daraufhin in hintergrundfreie 2D-Bilder rückprojiziert und ergeben die Eingabe für das Neuronale Netzwerk. Für eine bessere Erkennbarkeit der Szene ist als Eingabe nicht das Tiefenbild, sondern das dazu passende Farbbild dargestellt.

werden vom Netzwerk jeweils als Person oder Nicht Person klassifiziert und zeitgleich wird eine Orientierung geschätzt. Diese ist jedoch im Sinne einer Anwendung nur gültig, wenn der Bildausschnitt auch eine Person enthält.
Die gesamte Verarbeitungspipeline ist in das am Fachgebiet NIKR verwendete Roboter-Framework MIRA[1] integriert.

4.2 Training des Multi-Task-Systems

Nachdem die Anwendung des Gesamtsystems dargestellt wurde, wird in den folgenden Abschnitten auf den Ablauf des Trainings des Multi-Task-Netzwerks eingegangen. Hierzu beginnt Abschnitt 4.2.1 mit einer Vorstellung und Einordnung der vier verwendeten Datensätze. Daraufhin folgen in Abschnitt 4.2.2 verschiedene Multi-Task-Netzwerkarchitekturen, welche im Rahmen dieser Masterarbeit miteinander verglichen

[1] `http://www.mira-project.org` - Abgerufen am 16.12.2019

werden. Abschließend wird in Abschnitt 4.2.3 der Trainingsablauf und verschiedene, für das Multi-Task Learning notwendige, Parameter vorgestellt.
Für das weitere Vorgehen ist wichtig, dass das Multi-Task-Netzwerk nur auf Bildausschnitten (Patches, vgl. Abbildung 4.1) arbeitet. Statt einer Personendetektion muss das Netzwerk also lediglich eine Klassifikation (Person und Nicht Person) der Patches durchführen und deren Oberkörperorientierung schätzen. Auf den genauen Umgang mit diesen Bildausschnitten, inklusive deren Extraktion und Verarbeitung wird in Abschnitt 4.2.3 näher eingegangen.

4.2.1 Datengrundlage

Um ein System zur Personenwahrnehmung zu trainieren, müssen die entsprechenden Trainingsdaten vorliegen. Im Rahmen dieser Masterarbeit sollen sowohl zwei Datensätze des Fachgebiets Neuroinformatik und Kognitive Robotik der Technischen Universität Ilmenau, ein Datensatz des Social Robotics Lab (SRL) der Universität Freiburg und ein selbst aufgenommener Datensatz verwendet werden:

- SuPer Datensatz [Lewandowski et al., 2019a]: Datensatz zur Personendetektion in einer Supermarktumgebung im Rahmes des Projektes ROTATOR[2]
- TUI-Zuse Datensatz: Im Rahmen dieser Masterarbeit aufgenommener Datensatz zur Personendetektion mit künstlichen Verdeckungen
- SRL Datensatz [Linder et al., 2015]: Datensatz zur Klassifikation von menschlichen Attributen
- Deep-Orientation Datensatz [Lewandowski et al., 2019b]: Datensatz zur Oberkörperorientierungsschätzung von Personen

Insgesamt stehen damit verschiedene Datensätze für unterschiedliche Problemstellungen zur Verfügung. Alle enthalten Tiefenbildaufnahmen der Microsoft Kinect2 Kamera, sind jedoch entweder mit dem Standard-Treiber (SRL, Deep-Orientation) oder mit dem erweiterten Treiber (SuPer, TUI-Zuse) für den von acht auf 18 Meter erweiterten Tiefenbereich [Lawin et al., 2016] aufgenommen. Die Datensätze sind, abhängig von der Problemstellung, unterschiedlich gelabelt. Deshalb muss in den weiterführenden

[2] `https://www.tu-ilmenau.de/neurob/projects/finished-projects/rotator/` - Abgerufen am 16.12.2019

Experimenten eine Vorgehensweise angewendet werden, welche mehrere Datensätze so kombiniert, dass alle benötigten Trainingslabels abgedeckt werden, vgl. hierzu Kapitel 3.5.2. Beispielhafte Aufnahmen aus den vier Datensätzen sind in den Abbildungen 4.3 bis 4.6 dargestellt.
Um die Datensätze miteinander vergleichbar zu machen, bietet sich ein Blick auf die durchschnittliche Entfernung der Objekte zum Sensor an. Diese gibt einen Aufschluss darüber, wie gut beispielsweise Personen in den einzelnen Datensätzen erkennbar sind und in welchen Tiefen überhaupt (Trainings-)Daten vorliegen. In Abbildung 4.7 ist die Verteilung der Tiefenwerte für alle vier Datensätze dargestellt, und in Tabelle 4.1 ist die Aufteilung der Datensätze in Training, Validierung und Test ersichtlich. Nach einer kurzen Vorstellung und Einordnung der Datensätze soll auf diese Verteilungen nochmals kurz eingegangen werden.

SuPer Datensatz (Detektion)

Für das Training eines Personendetektors bietet sich der SuPer (Supermarkt Personen) Datensatz [Lewandowski et al., 2019a], Abbildung 4.3, an. Dieser wurde in einer Supermarktumgebung aufgenommen und vollständig annotiert. Problematisch ist an diesem Datensatz jedoch eine recht starke Überschneidung von Trainings- und Testdaten. Diese sind alle in der selben Umgebung mit den gleichen Testpersonen aufgenommen worden. Da der Datensatz nicht in Training und Validierung aufgeteilt ist und die Anzahl der verschiedenen Personen zu gering ist, kann das Validierungsset nur als zufälliges Subset (25% der Daten) aus dem Trainingsteil des Datensatzes erstellt werden. Somit ist davon auszugehen, dass die Validierungsdaten wenig Aussagekraft über die Leistungsfähigkeit der trainierten Netze geben können.

TUI-Zuse Datensatz (Detektion)

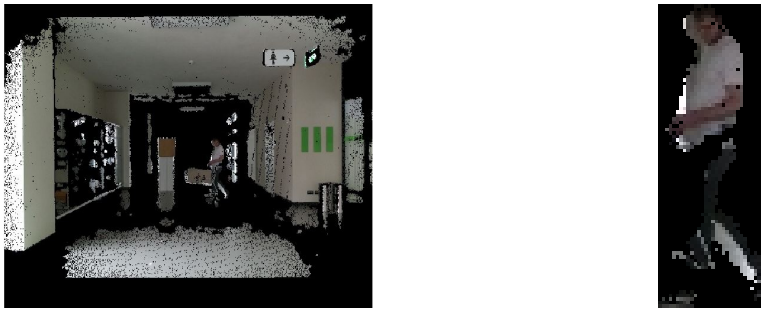
Aufgrund der beschriebenen Probleme des SuPer Datensatzes wurde für eine bessere Vergleichbarkeit im Rahmen dieser Masterarbeit ein weiterer Datensatz aufgenommen, der große konzeptionelle Ähnlichkeiten zum SuPer Datensatz aufweist, jedoch aus mehr Personen und Negativdaten besteht. Somit ist eine Aufteilung des Datensatzes anhand der Personen, im Gegensatz zum SuPer Datensatz, möglich. Hierzu wurden im Zusebau der TU Ilmenau verschiedene Hindernisse (Kisten, Einkaufswagen, Säulen)

(a) Vollbildaufnahme aus der Supermarktumgebung

(b) Bildausschnitt Personenklasse

(c) Bildausschnitt Hintergrundklasse

Abbildung 4.3: Beispielaufnahme des SuPer Datensatzes
In Abb. 4.3a ist das komplette Farbbild der Kinect2-Kamera dargestellt. In Abb. 4.3b ist ein Personen-Patch, in Abb. 4.3c ein Hintergrund-Patch eines Regals zu erkennen.



(a) Vollbildaufnahme aus der TUI-Zuse Umgebung

(b) Bildausschnitt Personenklasse

Abbildung 4.4: Beispielaufnahme des TUI-Zuse Datensatzes
In Abb. 4.4a ist das komplette Farbbild der Kinect2-Kamera dargestellt. In Abb. 4.4b ist ein Personen-Patch zu erkennen. Die Negativdaten des Datensatzes sehen denen des SuPer Datensatzes (Abb. 4.3c) sehr ähnlich, stammen jedoch aus einer anderen Umgebung)

in unterschiedlichen Konstellationen platziert. Eine Beispielszene ist in Abbildung 4.4 dargestellt. Durch ein für jede Aufnahme gelerntes Hintergrundmodell und die Vorgehensweise, jeweils nur eine Person gleichzeitig aufzunehmen, konnten so schnell neue Positivdaten mit typischen Verdeckungsmustern erzeugt werden. Insgesamt wurden 17 Personen (6 weiblich, 11 männlich) aufgenommen. Hierbei wurden Personen, welche

bereits in den Datensätzen SuPer oder Deep-Orientation im Trainingsteil oder im Validierungsteil vorkommen auch jeweils Training (vier Personen) bzw. Validierung (eine Person) zugeordnet. Die restlichen 12 Personen werden für den Test verwendet. Die Negativdaten wurden sowohl aus einer Rundfahrt durch den Zusebau ohne Personen als auch aus personenfreien Aufnahmen eines anderen Supermarktes erzeugt. Der SuPer Datensatz (Abbildung 4.7a) enthält zwar Elemente in den Entfernungen über 10m, jedoch nimmt hier die Datendichte massiv ab. Daher wurde der neue Datensatz nur mit Daten in maximal 10m Entfernung aufgenommen. Der Datensatz wird im Folgenden als TUI-Zuse bezeichnet und dient hauptsächlich der abschließenden Bewertung und dem Vergleich der Netzwerke.

SRL Datensatz (Detektion)

Für weitere Experimente wurde der SRL-Datensatz [Linder et al., 2015] des Instituts für Soziale Robotik der Universität Freiburg verwendet, vgl. Abbildung 4.5. Dieser wurde ursprünglich für die Klassifikation von Attributen, wie z.B. Alter oder Geschlecht von Personen, aufgenommen. Daher enthält der Datensatz keine Negativdaten oder Verdeckungen, dafür jedoch 118 Personen in unterschiedlichen Entfernungen und Ausrichtungen. Auch sehr nahe Aufnahmen in ca. 0,5 Metern Entfernung zum Sensor, in denen die Personen nicht mehr vollständig zu sehen sind, sind enthalten. Die maximale Distanz der Daten beträgt 4,5 Meter, siehe Abbildung 4.7c.

(a) Vollbildaufnahme

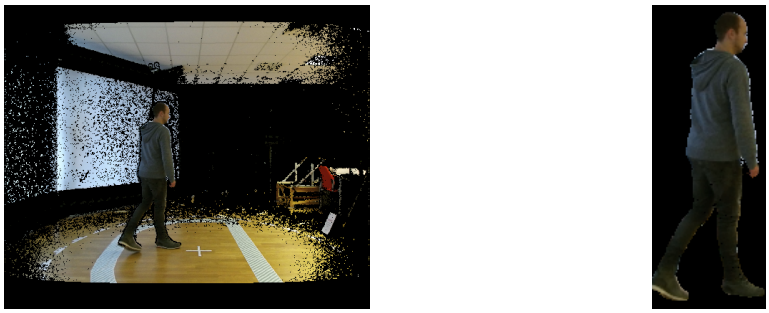
(b) Bildausschnitt

Abbildung 4.5: Beispielaufnahme des SRL Datensatzes
In Abb. 4.5a ist ein komplettes Farbbild der Kinect2-Kamera des Datensatzes dargestellt. In Abb. 4.5b ist ein beispielhafter Personen-Patch zu erkennen.

Dieser Datensatz kann aufgrund der großen Personenanzahl zur Verbesserung der Detektionsleistung beitragen oder als Testdatensatz verwendet werden. Der Datensatz wurde anhand der Personen-ID in Training (Person 1-59), Validierung (Person 50-89) und Test (Person 90-117) aufgeteilt.

Deep-Orientation Datensatz (Orientierungsschätzung)

Für die Orientierungsschätzung soll der Datensatz Deep-Orientation [Lewandowski et al., 2019b] verwendet werden. Dieser wurde in einer Indoor-Umgebung aufgenommen. Eine Beispielszene ist in Abbildung 4.6 dargestellt. Durch ein Kameratrackingsystem mit Infrarot-Markern ist in diesem Datensatz die Oberkörperorientierung kontinuierlich annotiert. Der Datensatz besteht aus 22 männlichen und 15 weiblichen Personen in insgesamt 108.505 Aufnahmen und ist damit der zweitgrößte verwendete Datensatz. Für die Detektion kann Deep-Orientation ebenfalls (mit-)verwendet werden, indem jede Aufnahme als Positivklasse eingeordnet wird.



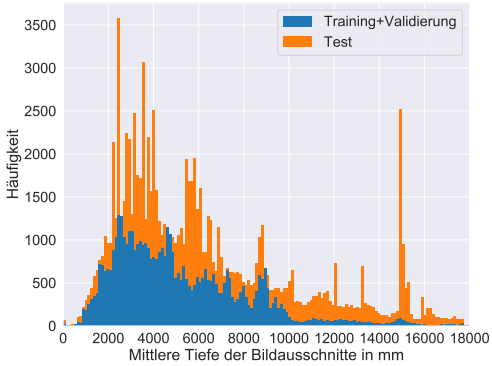
(a) Vollbild **(b)** Bildausschnitt

Abbildung 4.6: Beispielaufnahme des Deep-Orientation Datensatzes
In Abb. 4.6a ist ein komplettes, registriertes Farbbild der Kinect2-Kamera des Datensatzes dargestellt. In Abb. 4.5b ist ein beispielhafter Personen-Patch zu erkennen.

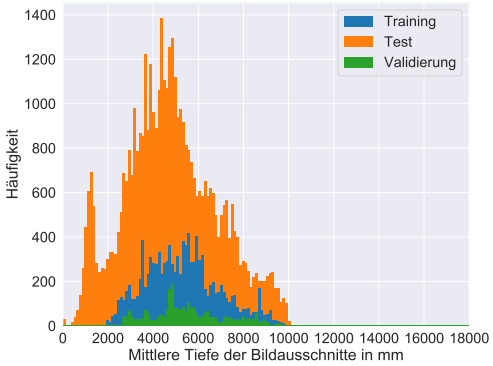
Fazit und Datensatzzusammenstellung für die folgenden experimentellen Untersuchungen

Für das Training der folgenden Experimente wird die Kombination aus den Datensätzen SuPer und Deep-Orientation gewählt. Somit stehen sowohl ausreichend Detektionslabel inklusive Negativdaten zur Verfügung, als auch genügend Orientierungslabels. Insgesamt ist zu erkennen, dass für die Personendetektion bis in zehn Metern Entfernung

Trainingsdaten bereitstehen, vgl. Abbildung 4.7a. Für die Orientierungsschätzung liegen Daten bis zu einer Entfernung von sechs Metern vor, vgl. Abbildung 4.7d. Zum Testen werden die Testdatensätze von allen vier verwendeten Datensätzen herangezogen. Die Detektion kann anhand des SuPer- SRL- und TUI-Zuse-Datensatzes bewertet werden, für die Orientierungsschätzung wird der Deep-Orientation Datensatz verwendet.

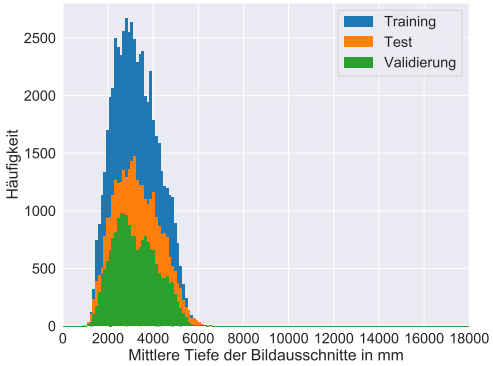


(a) Verteilung der SuPer Daten



(b) Verteilung der TUI-Zuse Daten

(c) Verteilung der SRL Daten



(d) Verteilung der Deep-Orientation Daten

Abbildung 4.7: Entfernungsverteilung der Samples über den Datensätzen SuPer, Deep-Orientation, TUI-Zuse und SRL
Die Entfernung eines Samples (Personen und Negativbeispiele) entspricht dem mittleren Tiefenwert des zugehörigen Bildausschnittes.

Datensatz	**Training**	**Validierung**	**Test**	**Gesamt**
SuPer*	36.224	12.032	101.188	149.444
TUI-Zuse	11.226	2.684	43.821	57.731
SRL*	55.731	28.244	22.557	106.532
Deep-Orientation	57.717	19.368	31.420	108.505

Tabelle 4.1: Übersicht über die Anzahl der Tiefenbildausschnitte für die Datensätze
** Für den Datensatz SuPer existiert keine definierte Aufteilung in Training und Validierung. Da die Personen anhand der Labels nicht eindeutig identifizierbar sind, wurde der Validierungsteil als zufälliges 25% Subset der Trainingsdaten abgespaltet. Somit besteht Training und Validierung aus den gleichen Personen, jedoch in unterschiedlichen Aufnahmen. Der SRL Datensatz konnte im Gegensatz zum SuPer Datensatz gut anhand der Personeninformationen in Training, Validierung und Test aufgeteilt werden.*

4.2.2 Netzwerkarchitekturen

Mit der festgelegten Datengrundlage kann das Multi-Task-Netzwerk trainiert und evaluiert werden. Deshalb wird in diesem Abschnitt auf verschiedene, einsetzbare Netzwerkarchitekturen eingegangen. Wie bereits im Fazit zum State of the Art (Abschnitt 3.7) festgelegt wurde, kommen hier Netzwerkarchitekturen des Hard Parameter Sharing zum Einsatz (vgl. Abschnitt 3.3.1). Dargestellt ist diese Herangehensweise in Abbildung 4.8. Notwendig ist hier ein Basisnetzwerk als Feature-Extraktor und taskspezifische Netzwerkköpfe für die einzelnen Tasks. Als Basisnetzwerke werden in dieser Masterarbeit zwei Ansätze verfolgt.
Die performanteren Ansätze verwenden als Basisnetzwerk die BeyerModRelu-Architektur (Abschnitt 2.1.1), welche auch von [Lewandowski et al., 2019b] als Orientierungsschätzer eingesetzt wird. Diese Architektur ist leichtgewichtig, erreicht eine sehr gute Leistung bei der Regression der Oberkörperorientierung und ist dabei sehr schnell in der Inferenz.
Die leistungsfähigeren Ansätze setzen auf ein ResNet (Abschnitt 2.1.2) als Basisnetzwerk. ResNets besitzen eine größere Tiefe, Komplexität und Parameterzahl als das BeyerModRelu-Netzwerk. Zudem ergibt sich bei diesem Netzwerk die Ausgabe durch ein Global Average Pooling, wodurch der räumliche Bezug der Features zwar verloren geht, die Anzahl der Features jedoch stark reduziert werden kann. Aufgrund guter

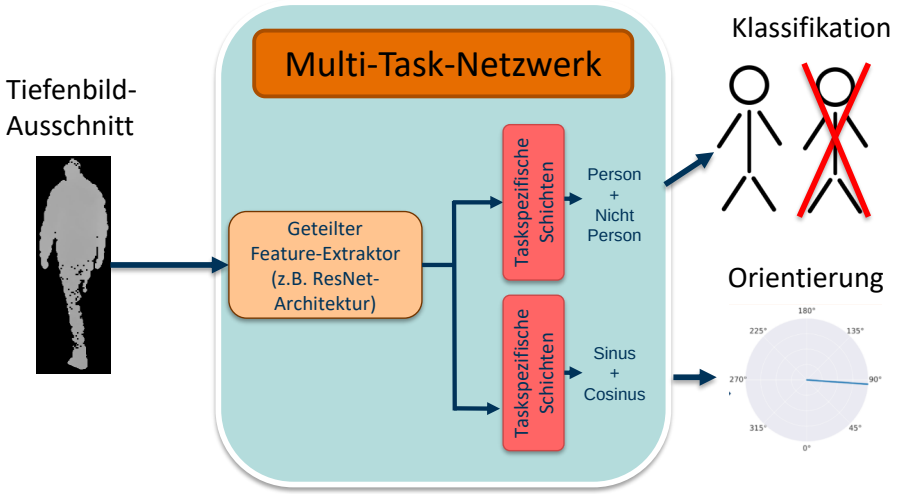


Abbildung 4.8: Systemarchitektur zum Training eines Multi-Task-Netzwerkes
Das Netzwerk wird auf die Klassifikation und Orientierungsschätzung auf Patch-Ebene trainiert. Erkennbar ist die aus der State-of-the-Art-Recherche ausgewählte Herangehensweise des Hard Parameter Sharing (Kapitel 3.2.1), aufgeteilt in Feature-Extraktor und taskspezifische Schichten.

Ergebnisse auf Benchmarkdatensätzen [He et al., 2016] könnten diese Netzwerke auch im beschriebenen Einsatzszenario dieser Masterarbeit überzeugende Ergebnisse liefern. Aufgrund des hohen Ressourcenbedarfs wurden tiefere ResNets (z.B. ResNeXt50) zwar in den Vorexperimenten (Kapitel 5.2.1) auf ihre Eignung als Basisnetzwerk getestet, jedoch wegen der hohen Rechenzeit (Kapitel 5.4.1) in der Anwendung nicht weiter analysiert. Deshalb soll hier die Wahl auf die kleinere ResNet18-Architektur fallen.

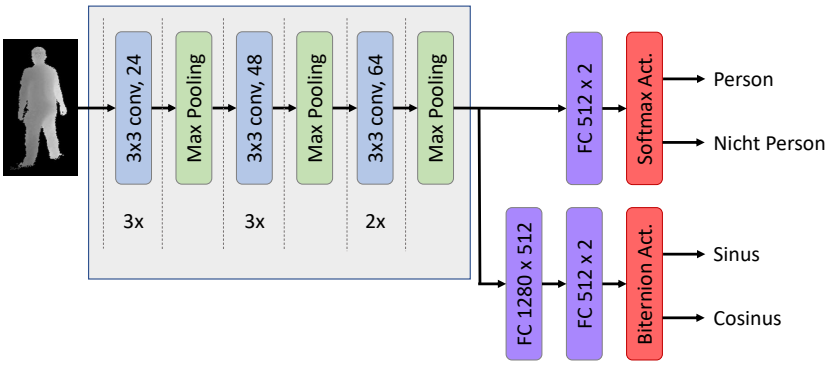
Um die Eignung der beiden genannten Netzwerkarten (ResNet18 und BeyerModRelu) als Multi-Task-Basisnetzwerke zu evaluieren, werden im Folgenden Multi-Task-Architekturen aus diesen Netzwerken vorgestellt. Zuerst wird dabei auf Multi-Task-Architekturen eingegangen, welche nur Features einer Abstraktionsebene verwenden (Single-Scale-Netzwerke, siehe auch Abschnitt 3.3.1). Daraufhin folgt eine Erweiterung der Architekturen um die Nutzung von Features verschiedener Abstraktionsebenen (Multi-Scale-Netzwerke, siehe auch Abschnitt 3.3.1).

Single-Scale-Multi-Task-Architekturen

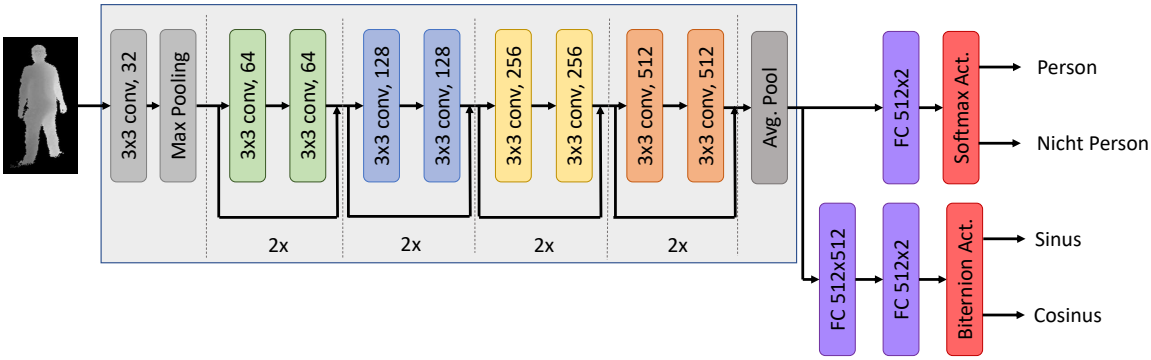
Für die ersten Tests zur grundsätzlichen Funktionsweise von Multi-Task Learning soll die einfache Struktur des Single-Scale-Multi-Task-Learning aus Kapitel 3.3.1 zum Einsatz kommen. Hierzu werden als Feature-Extraktoren die Faltungsschichten der BeyerModRelu- und ResNet18-Architektur verwendet. Am Ende dieser Schichten wird ein zusätzlicher vollverschalteter Teil für jeden Task angehängt.
Für die Personenklassifikation wird hierzu eine vollverschaltete Schicht mit Softmax-Ausgabefunktion und zwei Ausgabeneuronen (*Person* und *Nicht Person*) eingesetzt.
Für die Orientierungsschätzung erfolgt die Schätzung des Winkels, in Anlehnung an [Beyer et al., 2015] und [Lewandowski et al., 2019b], über eine Schätzung von Sinus- und Cosinusanteil, Normalisierung auf die quadratische Summe von eins und anschließende Berechnung des Winkels über den Arkustangens. Dies wird als Biternion-Aktivierungsfunktion bezeichnet. Im Rahmen der experimentellen Untersuchungen im Single-Task-Szenario (Kapitel 5.2.2) stellte sich für diesen Task heraus, dass die Nutzung von zwei aufeinanderfolgenden vollverschalteten Schichten vorteilhaft für die Regressionsleistung ist.
Die resultierenden Architekturen sind in Abbildung 4.9 schematisch dargestellt. In Kapitel 5.3.3 folgen die Analysen und Ergebnisse zu diesen Single-Scale-Multi-Task-Architekturen.

Multi-Scale-Multi-Task-Architektur

Neben den Single-Scale-Multi-Task-Architekturen werden die Experimente auch für erweiterte Netzwerkstrukturen durchgeführt. Der HyperFace-Ansatz von [Ranjan et al., 2019] wurde bereits in der State-of-the-Art-Recherche in Kapitel 3.3.1 kurz vorgestellt. Im Rahmen dieser Masterarbeit soll die Eignung von HyperFace-ähnlichen Multi-Scale-Netzwerkstrukturen für das gegebene Problem untersucht werden. Da sowohl die ResNet- als auch die BeyerModRelu-Architektur blockweise aufgebaut sind, bietet es sich an, die Features der einzelnen Blöcke zu kombinieren, um eine neue Feature-Repräsentation mit unterschiedlichen Abstraktionsniveaus zu erzeugen. Diese soll dann zum Lernen aller Tasks eingesetzt werden. Da sich die BeyerModRelu-Architektur jedoch als grundsätzlich weniger leistungsfähig als die ResNet18-Architektur zeigt, wird im Folgenden nur weiter auf die ResNet18-Architekturen eingegangen.



(a) BeyerModRelu-Single-Scale-Multi-Task-Architektur



(b) ResNet18-Single-Scale-Multi-Task-Architektur

Abbildung 4.9: Single-Scale-Netzwerkarchitekturen zum Multi-Task Learning von Personenklassifikation und Regression der Oberkörperorientierung
Durch Kombination eines Basisnetzwerks (ResNet18 oder BeyerModRelu, grau hinterlegt) mit beiden Netzwerkköpfen (Klassifikation und Orientierungsschätzung) entsteht eine Multi-Task-Architektur nach dem Prinzip des Hard Parameter Sharing. Die Schichten für die Batch-Normalisierung und Dropout im Orientierungskopf sowie die ReLu-Aktivierungsfunktionen sind der Einfachheit halber nicht dargestellt.

Weitere Details zu den BeyerModRelu-Multi-Scale-Experimenten befinden sich im Anhang in Kapitel A.3.

Um das Multi-Scale-Vorgehen mit der Netzwerkstruktur der ResNets zu kombinieren, wird nach jedem der vier Blöcke eine Global-Average-Pooling-Schicht eingesetzt. Die Ausgaben der Global-Average-Pooling-Schichten werden daraufhin konkateniert und es entsteht ein längerer Feature-Vektor als bei der Single-Scale-Architektur. Daher müssen die Dimensionen der taskspezifischen Schichten entsprechend angepasst werden. Hierdurch entsteht eine Struktur, welche im Folgenden *ResNet18-Multi-Scale-GAP* genannt wird. Sie ist in Abbildung 4.10 dargestellt.

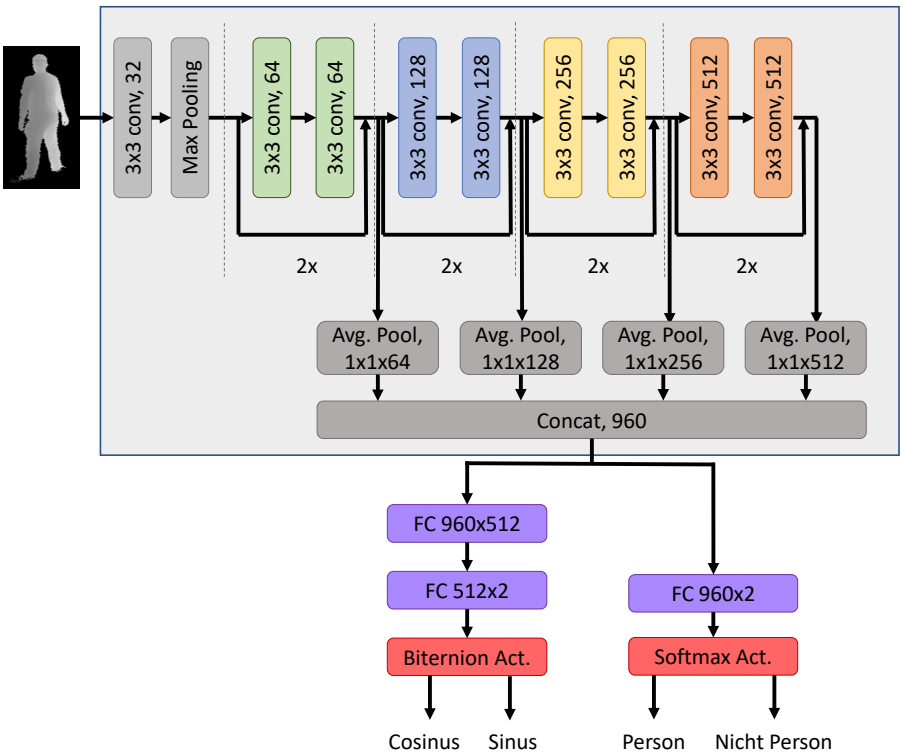


Abbildung 4.10: Netzwerkarchitektur ResNet18-Multi-Scale-GAP

Aus dem ResNet18-Feature-Extraktor werden auf unterschiedlichen Abstraktionsniveaus Features durch Global Average Pooling extrahiert. Hierbei entsteht ein Multi-Scale-Feature-Extraktor, welcher grau hinterlegt dargestellt ist. Die Features werden dann von beiden Netzwerkköpfen verarbeitet. Die Schichten für die Batch-Normalisierung und Dropout im Orientierungskopf sowie die ReLu-Aktivierungsfunktionen sind der Einfachheit halber nicht dargestellt.

Da durch das Global Average Pooling die Positionen der Features in den Feature Maps verloren geht, soll zusätzlich eine Abwandlung des Netzwerks untersucht werden. Es werden mithilfe eines Local Average Poolings aus den Ausgaben der Layer eins, zwei und drei Feature Maps mit den Dimensionen 8×8, 4×4 und 2×2 erzeugt. Somit wird ein räumlicher Bezug beibehalten. Da gerade in den frühen Layern eine sehr hohe Anzahl an Merkmalen entsteht (Feature-Map-Dimension 8×8 bei 64 Feature Maps: $8 \cdot 8 \cdot 64 = 4096$ Features), werden diese durch die Verwendung von 1x1-Faltungen, wie bei [Ranjan et al., 2019], reduziert. Da die Anzahl der Features aus dem letzten Layer fest ist (512), bietet es sich an, die selbe Anzahl an Features aus den anderen Layern zu extrahieren. Durch geeignete Wahl der Parameter der 1x1-Faltungen kann die resultierende Feature-Anzahl nach der Reduktion auf einen Feature-Vektor je Layer auf 512 gebracht werden. Durch Konkatenation entsteht dann ein $4 \cdot 512 = 2048$ Elemente langer Feature-Vektor, welcher dann von allen taskspezifischen Schichten verwendet wird. Diese Architektur soll im Folgenden *ResNet18-Multi-Scale-LAP* genannt werden. Sie ist in Abbildung 4.11 dargestellt.
Die Ergebnisse zu den beiden ResNet18-Multi-Scale-Architekturen sind, zusammen mit einem Vergleich zu den ResNet18-Single-Scale-Architekturen, in Kapitel 5.3.4 dargestellt.

4.2.3 Trainingspipeline und Ablauf

Nach der Beschreibung der Netzwerkarchitekturen kann nun auf den Ablauf des Trainings eingegangen werden. Das gesamte Deep-Learning-System wurde in Pytorch[3] mit Python[4] implementiert, als auch trainiert. Dadurch ist es leicht in das Roboter-Framework oder andere Anwendungen einzubinden. Das komplette Training läuft anhand der bereits im vorangegangenen Abschnitt beschriebenen Architektur auf Patch-Ebene (Abbildung 4.8) ab. Es wird somit nur die rechte Hälfte der kompletten Architekturübersicht aus Kapitel 4.1 zum Training benötigt.
Folglich müssen zuerst aus allen Datensätzen die Patches extrahiert werden. Für die Datensätze SuPer und TUI-Zuse aus Kapitel 4.2.1 werden die Tiefenbildausschnitte anhand der in Kapitel 4.1 beschriebenen Vorgehensweise extrahiert. Für den Datensatz TUI-Zuse wird hierbei zusätzlich noch eine Hintergrundsubtraktion durchgeführt.

[3] `https://pytorch.org/` - Abgerufen am 16.12.2019
[4] `https://www.python.org/` - Abgerufen am 16.12.2019

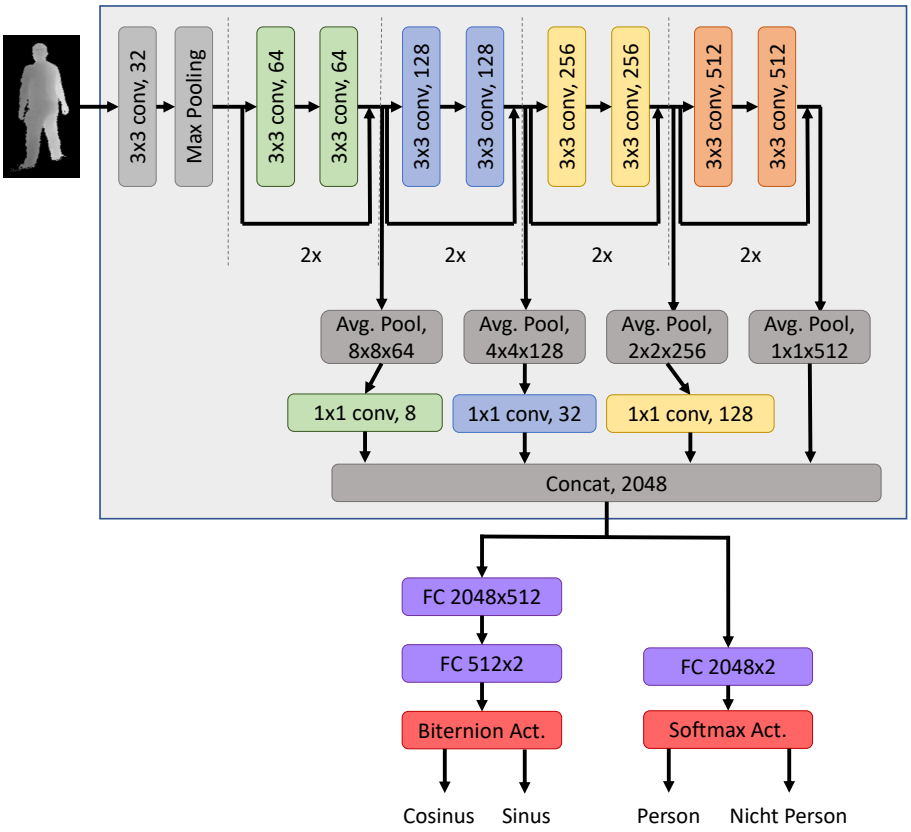


Abbildung 4.11: Netzwerkarchitektur ResNet18-Multi-Scale-LAP

Aus dem ResNet18-Feature-Extraktor werden auf unterschiedlichen Abstraktionsniveaus Features durch Local Average Pooling extrahiert. Zur Reduktion der Feature-Anzahl werden 1x1-Faltungen nachgeschaltet. Hierbei entsteht ein Multi-Scale-Feature-Extraktor, welcher grau hinterlegt dargestellt ist. Die Features werden dann von beiden Netzwerkköpfen verarbeitet. Die Schichten für die Batch-Normalisierung und Dropout im Orientierungskopf sowie die ReLu- Aktivierungsfunktionen sind der Einfachheit halber nicht dargestellt.

Zu jedem Bildausschnitt ist anhand der Ground-Truth-Labelinformationen der vollständigen Bilder bekannt, ob es sich bei dem Patch um eine Person handelt oder nicht. Für die Datensätze SRL und Deep-Orientation stehen die Bildausschnitte bereits in gelabelter Form zur Verfügung.
Neben dem Training wird auch die Evaluation und Auswahl der Netzwerke im weiteren Verlauf dieser Masterarbeit auf Patch-Ebene erfolgen (Kapitel 5.2 und 5.3). Dabei sind die Ergebnisse zwar gut vergleichbar, dennoch ist auch eine Interpretation auf Bild-Ebene (komplette Szenen) notwendig. Deshalb wird die beste Netzwerkarchitektur schlussendlich auch anhand der Anwendungspipeline ausgewertet (Kapitel 5.4).
Im Folgenden wird auf verschiedene Details des Trainingsprozesses eingegangen. Die Vorverarbeitung der Daten, Auswahl der Lossfunktionen und der Einsatz einer Augmentierung sind dabei klassische Fragestellungen beim Machine Learning. Dahingegen sind die Zusammenstellung der Datensätze, die Suche nach der besten Taskwichtung und der besten Epoche eher den Problemstellungen des Multi-Task Learning zuzuordnen.

Vorverarbeitung der Daten

Die Eingangstiefendaten können in unterschiedlicher Art und Weise dem Netzwerk präsentiert werden. Im Rahmen dieser Masterarbeit werden zwei verschiedene Ansätze miteinander verglichen.

- Skalierung aller Tiefenwerte auf den Bereich [0...1] durch Teilen der Pixelwerte durch den maximalen Tiefenwert des Sensors (18m); Bezeichnung in den Experimenten: *Skalierung*
- Normalisierung aller Tiefenwerte über den gesamten Trainingsdatensatz auf Mittelwert null und Standardabweichung eins; Bezeichnung in den Experimenten: *Normalisierung*

Die Ergebnisse zur Vorverarbeitung befinden sich in Kapitel 5.2.1.

Lossfunktionen

Für die Personenklassifikation wird die Kreuzentropie [Gross, 2019] als Loss-Funktion verwendet, die Orientierungsschätzung wird mit dem Von-Mises-Loss aus Kapitel 2.3 bewertet.

Datenaugmentierung

Für eine bessere Generalisierungsfähigkeit des trainierten Netzwerks können die Eingabedaten im Trainingsablauf augmentiert werden. Hierbei wird durch Einsatz von Bildtransformationen die Anzahl an Trainingssamples künstlich erhöht. Aufgrund der Form von Personen sind gewisse Transformationen realitätsfern (vertikale Spiegelungen oder starke Rotationen). Eine sinnvolle Art der Augmentierung, die im Rahmen dieser Masterarbeit angewendet wird, ist jedoch die zufällige, horizontale Spiegelung der Patches mit Wahrscheinlichkeit 50%. Hierbei bleiben die Personenattribute gleich, die Orientierung wird jedoch, nach Formel 4.1, gespiegelt.

$$\phi_{horizontal\_gespiegelt} = (360° - \phi) \mod 360° \tag{4.1}$$

$\phi_{horizontal\_gespiegelt}$	Augmentierter Winkel
ϕ	Ground-Truth-Winkel

Erzeugung des kombinierten Trainingsdatensatzes

In der State-of-the-Art-Recherche (Abschnitt 3.5) und der Behandlung der Datensätze (Kapitel 4.2.1) stellte sich heraus, dass die Verwendung von mehreren Datensätzen auch ohne zusätzliche Ergänzung von Labels zu einem ausreichenden Trainingsdatensatz führt. Mehrere Datensätze können anhand der Vorgehensweisen aus Kapitel 3.5 unterschiedlich kombiniert werden. Da die Datensätze nicht die selbe Anzahl an Elementen haben, kann es einen Unterschied machen, ob die Datensätze einfach konkateniert, oder auf eine einheitliche Elementezahl gebracht werden. Doch selbst dann ist zu beachten, dass manche Datensätze mehr Labels mitbringen, als andere. So bringt ein Orientierungs-Sample neben dem Winkel für die Orientierungsschätzung auch für die Klassifikation die Information, dass es sich um eine Person handelt. Ein Sample aus einem Detektionsdatensatz bringt ausschließlich eine Information für den Klassifikations-Task. Würden beide Datensätze in gleichem Mengenverhältnis kombiniert, beinhaltet nur die Hälfte der Elemente des Trainingsdatensatzes Orientierungsinformationen. Damit wird die Orientierungsschätzung, wenn zur Berechnung der Loss-Funktion alle Samples verwendet werden, effektiv nur mit der halben Batch Size der Klassifikation trainiert.

Deshalb sollen die folgenden drei Varianten der Zusammenstellung von verschiedenen

Datensätzen zu Batches im Rahmen dieser Masterarbeit miteinander verglichen werden:

- Konkatenieren der Datensätze und zufällige Permutation der Elemente, unabhängig von deren Elementeanzahl; Bezeichnung in den Experimenten: *konkateniert*
- Zusammenführen der Datensätze zu Batches mit einem bestimmten Mengenverhältnis (z.B. 50-50). Bezeichnung in den Experimenten: *50-50*
- Zusammenführen der Datensätze zu Batches mit einem bestimmten Mengenverhältnis (z.B. 50-50). Zusätzlich soll durch Auswahl von Elementen, die in die Loss-Funktion einfließen (Sampling), sichergestellt werden, dass jeder Task exakt die selbe Anzahl an Samples erhält; Bezeichnung in den Experimenten: *exakt 50-50*

Für ein besseres Verständnis werden die drei Varianten an einem Beispiel genauer erläutert. Hier soll von der Batch Size 10 ausgegangen werden. Da der Trainings-Teil des Deep-Orientation Datensatzes 60% größer ist als der des SuPer Datensatzes, ergeben sich bei der Verwendung der Variante *konkateniert* im Mittel 3,8 Patches aus dem SuPer Datensatz und 6,2 Patches aus dem Deep-Orientation Datensatz pro Batch. Dies führt zu 10 Klassifikations-Labels, da beide Datensätze Klassifikations-Labels mitbringen, jedoch im Mittel nur zu 6,2 Orientierungs-Labels. Weiterhin sind die genauen Mengenverhältnisse nicht fest, sondern hängen von der zufälligen Zuordnung der Elemente zu den Batches ab.
Bei Verwendung der Variante *50-50* würde der Deep-Orientation Datensatz um die Anzahl an Elementen, die er größer ist, als der SuPer Datensatz, reduziert. Die genaue Auswahl geschieht jede Epoche zufällig. Daraufhin werden die Batches zusammengestellt, indem jeweils 5 Patches zufällig aus beiden Datensätzen gezogen werden. Somit ergibt sich eine Batch-Zusammenstellung von genau 5 Patches aus dem SuPer Datensatz und 5 Patches aus dem Deep-Orientation Datensatz. Dennoch wird, aus dem selben Grund wie bei der vorherigen Variante, hier die Orientierungsschätzung nur mit 5 Informationen, die Klassifikation jedoch mit 10 Informationen trainiert.
Bei Verwendung der Variante *exakt 50-50* ändert sich an der Batch-Zusammenstellung zum vorherigen Ansatz nichts. Jedoch werden von den 10 Labelinformationen für die Klassifikation nur 5 zufällig ausgewählt und in die Loss-Berechnung miteinbezogen. Somit trainieren beide Tasks mit der selben Batch Size (5).
Die Ergebnisse zu den drei Vorgehensweisen sind in Kapitel 5.3.2 nachzuschlagen.

Wichtung der Tasks beim Training

Wie bereits in der State-of-the-Art-Analyse in Kapitel 3.4 vorgestellt, können im Rahmen dieser Masterarbeit verschiedene Varianten der Kombination von Loss-Funktionen zu einem Gesamt-Loss untersucht werden:

- Konstante Wichtungsfaktoren während des Trainings; Bezeichnung in den Experimenten: *Konstante Wichtungen*
- Adaptive Wichtungsfaktoren durch Gradientennormierung [Chen et al., 2017]; Bezeichnung in den Experimenten: *GradNorm*
- Adaptive Wichtungsfaktoren durch Task-Unsicherheiten [Lu et al., 2017]; Bezeichnung in den Experimenten: *Uncertainty Weighting*

Die Experimente zu den Multi-Task-Architekturen in Kapitel 5 werden mit konstanten Wichtungsfaktoren durchgeführt. Die beiden adaptiven Verfahren werden am Beispiel der ResNet18-Multi-Task-Architektur untersucht. Die Ergebnisse hierzu können im Anhang in Kapitel A.2 nachgelesen werden.

Auswahl der besten Epoche

Beim Single-Task Learning wird die beste Epoche eines Trainings anhand eines gewissen Güte- oder Fehlermaßes auf den Validierungsdaten ausgewählt. Beim Multi-Task Learning ist dies jedoch nicht ohne weiteres möglich, da für jeden Task ein anderes Gütemaß geeignet ist. Die Auswahl nach nur einem Task ist nicht zuverlässig, da der jeweils andere Task nicht berücksichtigt wird. Daher bieten sich Techniken der Informationsfusion [Gross, 2018] an.
Die Gütemaße auf *Score-Level* zu fusionieren ist schwierig, da sehr unterschiedliche Wertebereiche vorliegen (siehe hierzu auch Kapitel 5.1). Zudem ist hier zusätzlich eine Wichtung der Tasks nötig, um anzugeben, welches Gütemaß wie stark in die Berechnung des kombinierten Gütemaßes eingehen soll. Deshalb soll die Fusion in dieser Masterarbeit auf *Rank-Level* angegangen werden: Anhand der Gütemaße wird eine Rangfolge der besten Epochen je Task aufgestellt. Für jede Epoche werden dann die Ränge der verschiedenen Tasks addiert. Daraufhin wird die Epoche ausgewählt, welche den geringsten Gesamt-Rang hat. Sie entspricht somit dem besten Kompromiss aller Tasks. Diese Methodik löst somit direkt die Probleme der unterschiedlichen Wertebereiche und der gegenseitigen Wichtungen.

In Kapitel 5.3.1 befinden sich experimentelle Untersuchungen, in welchen die Auswahl nach nur einem Task (Klassifikation oder Orientierungsschätzung) mit der beschriebenen Fusionsmethode verglichen wird.

Lernraten und konstante Parameter

Weiterhin werden im Rahmen der Experimente unterschiedliche Lernraten evaluiert. Darüberhinaus gibt es eine Vielzahl weiterer Parameter. Aufgrund des stark steigenden Zeitbedarfes, alle diese systematisch auszuwerten, wird bei den folgenden Parametern an den Werten aus [Lewandowski et al., 2019b] festgehalten:

- Netzoptimierer: SGD
- Momentum-Term: 0,9
- Polynomieller Decay der Lernrate mit Exponent 0,9
- Kappa-Term des Von-Mises-Loss: 1,0
- Batch-Größe: 256
- Anzahl Trainingsepochen: 200

Die Batch-Größe wurde so groß wie möglich gewählt. 256 zeigte sich als gute Wahl, da hiermit alle folgenden Netzwerkarchitekturen auf den bestehenden Grafikkarten (NVIDIA GTX1080 Ti und NVIDIA RTX2080 Ti) trainiert werden können.

Kapitel 5

Experimentelle Untersuchungen

Um verlässliche Aussagen über die Genauigkeit des entworfenen Multi-Task-Systems zu erhalten, wurden systematische Experimente durchgeführt. Hierzu wird in diesem Kapitel zuerst auf verwendete Maße zur Bewertung von Klassifikationen, Detektionen und Orientierungsschätzungen eingegangen (Kapitel 5.1). Daraufhin werden, zur besseren Einordnung der Leistung der Multi-Task-Systeme, in Kapitel 5.2 Single-Task-Baselines für die patchbasierte Personenklassifikation und Oberkörperorientierungsschätzung trainiert und diskuiert. Im Anschluss werden die Experimente zu dem Multi-Task-System beschrieben. Hierbei wird zuerst auf das Training des Neuronalen Netzwerks (Kapitel 5.3) und danach auf das resultierende Gesamtsystem (Kapitel 5.4) eingegangen.

5.1 Bewertungsmaße

Um die Systemleistung bewerten zu können, kommen die Bewertungmaße aus Kapitel 2.4 zur Anwendung.
Zur Bewertung der Klassifikationsergebnisse der Netzwerke auf Patch-Ebene wird während des Trainings auf die Balanced Error Rate (BER) zurückgegriffen, da sie sich in den Vorexperimenten (Anhang A.1) als das bessere Maß zur Auswahl von Netzwerken als der F1-Score herausgestellt hat. Anhand dieser Fehlerrate kann auf den Validierungsdaten des Trainings die beste Epoche und deren Schwellwert zur Zuordnung der Personenklasse ausgewählt werden. Mit dieser Epoche und dem dazugehörigen Schwellwert wird anschließend auf den Testdaten evaluiert.
Auf den Testdaten kommen, abhängig vom Datensatz, unterschiedliche Fehlermaße

zum Einsatz:

- Datensätze mit Negativdaten (SuPer, TUI-Zuse): F1-Score
- Datensätze ohne Negativdaten (SRL, Deep-Orientation): Recall

Zur Bewertung der Detektion auf Bild-Ebene kommen Detection-Error-Tradeoff-Kurven (DET-Kurven) zur Anwendung. Mit diesen kann das Gesamtsystem aus Kandidatengenerator und Neuronalem Netzwerk mit anderen Detektoren verglichen werden.
Für die Orientierungsschätzung wird als Fehlermaß die mittlere, absolute Winkeldifferenz (MAAD, Mean Absolute Angle Difference) eingesetzt. Anhand dieser Maßzahl wird, wie bei der Klassifikation auch, die beste Epoche während des Trainings aus Sicht der Orientierungsschätzung ausgewählt und die Güte der Orientierungsregression auf den Testdaten bestimmt.

5.2 Ergebnisse der Vorexperimente und Baselines

Für die Orientierungsschätzung werden in [Lewandowski et al., 2019b] bereits Ergebnisse und Trainingsparameter vorgestellt. Um jedoch eine grundsätzliche Eignung von Neuronalen Netzwerken zur patchbasierten Personenklassifikation auf Tiefenbildern festzustellen, wurde hierzu eine Reihe von Vorexperimenten durchgeführt. Anhand dieser konnten erste Erkenntnisse über den Trainingsablauf und die dazugehörigen Hyperparameter gewonnen werden. Diese Ergebnisse werden nun kurz zusammengefasst und daraufhin in den Experimenten zu den Single-Task-Baselines verwendet. Der genaue Ablauf der Vorexperimente befindet sich im Anhang in Abschnitt A.1.

5.2.1 Vorexperimente zur Personenklassifikation

Anhand einer Reihe von Vorexperimenten wurden verschiedene Netzwerkarchitekturen (BeyerModRelu, ResNet18, ResNeXt50), Vorverarbeitungen (Skalierung, Normalisierung), Lernraten und Trainingsdatensätze analysiert.

Architektur

Die Experimente zeigen, dass sich sowohl das BeyerModRelu-Netzwerk als auch die ResNet-Typen grundsätzlich für eine Klassifikation von Bildausschnitten aus Tiefenbildern eignen. Zudem bringt die Nutzung von auf ImageNet vortrainiertn Gewichten

bei den ResNet-Architekturen leichte Verbesserungen der Erkennungsleistung. Deshalb werden für alle ResNet-Architekturen zur Personenklassifikation im Folgenden die Gewichte des *Torchvision Model Zoo*[1] verwendet.

Es ist zudem erkennbar, dass die komplexeren, rechenaufwendigeren Netzwerke (ResNet18, ResNeXt50) eine höhere Klassifikationsleistung auf allen Datensätzen erreichen, als das einfachere BeyerModRelu-Netzwerk. Zwischen dem ResNet18 und dem ResNeXt50 wird jedoch der Unterschied in der erreichten Erkennungsleistung viel geringer. So ist der F1-Score des ResNeXt50 im Vergleich zum ResNet18 auf dem SuPer Datensatz um 1,75 Prozentpunkte (86,9% gegen 85,2%) sowie der Recall auf dem SRL-Datensatz nur um 0,5 Prozentpunkte (95,0% gegen 94,5%) besser, vgl. Kapitel A.1.4 im Anhang. Um jedoch auf robotischen Plattformen einsetzbar zu sein, wird sich im Folgenden auf die performanteren Architekturen BeyerModRelu und ResNet18 beschränkt.

Lernrate

Für die initiale Lernrate zeigte sich bei den Vorexperimenten der Personenklassifikation, dass der Bereich $1 \cdot 10^{-2}$ bis $1 \cdot 10^{-4}$ ein guter Ausgangspunkt ist, vgl. Abschnitt A.1.4. Hier erreichen die Netzwerke alle eine vergleichbare Klassifikationsleistung.

Vorverarbeitung

Als Vorverarbeitung empfiehlt sich aus Sicht der Personenklassifikation die Skalierung der Tiefenwerte mit dem maximalen Sensorwert (18m) auf das Intervall [0...1] statt einer Normalisierung der Daten. Dies führt zu leicht besser generalisierenden Netzwerken und es muss nicht für jede Datensatzkombination ein anderer Mittelwert verwendet werden. Zusätzlich werden bei der Skalierung ungültige Tiefenwerte mit Tiefe *null* ohne zusätzliche Maskierung weiterhin auf das neutrale Element *null* abgebildet. Außerdem zeigen die vortrainierten ResNets keine Probleme beim Umlernen von den normalisierten RGB-Daten des Vortrainings zu den hier verwendeten skalierten Tiefendaten. Dennoch muss vor einer endgültigen Entscheidung noch evaluiert werden, ob die Skalierung auch für die Orientierungsschätzung einsetzbar ist, vgl. Abschnitt 5.2.2.

[1] `https://pytorch.org/docs/stable/torchvision/models.html` - Abgerufen am 16.12.2019

Kombination von Datensätzen

Abschließend zeigten die Vorexperimente, dass bei Nutzung von kombinierten Datensätzen (z.B. SuPer und Deep-Orientation) bessere Ergebnisse erzielt werden können, jedoch müssen Ergebnisse auf Datensätzen ohne Negativdaten kritisch bewertet werden. So lernt ein System, welches unter anderem den Deep-Orientation Datensatz gesehen hat, sehr schnell, dass alle Beispiele aus diesem Datensatz Personen sind. Anhand von leicht unterschiedlichen Aufnahmesituationen oder Treibern kann das Netzwerk den Datensatz leicht identifizieren. Die Performance auf den Testdaten erreicht dann nahezu 100%, jedoch hat diese Aussage keinen Wert, da das Netzwerk nur gelernt hat, den Datensatz zu erkennen, nicht aber, Personen zu erkennen. Aus diesem Grund wird in den folgenden Multi-Task-Experimenten die Detektions- bzw. Klassifikationsleistung auf dem Deep-Orientation Testdatensatz nicht berücksichtigt. Genutzt wird der Testdatensatz jedoch für die Bewertung der Qualität der Orientierungsschätzung.

5.2.2 Baseline-Experimente

Wie schon in Kapitel 3.1.4 beschrieben wurde, kann Multi-Task Learning durch das Ausnutzen von geteilten Informationen zu einer verbesserten Leistung auf den einzelnen Tasks führen. Um dies im Kontext der Kombination von Personenklassifikation und Orientierungsschätzung zu untersuchen, sind entsprechende Single-Task-Systeme zum Vergleich notwendig. Neben einem bestehenden SVM-basierten Punktwolkenklassifikator von [Lewandowski et al., 2019a] und dem bestehenden Orientierungsschätzer *Deep-Orientation* [Lewandowski et al., 2019b] sollen selbst trainierte Netzwerke als Baselines für die Tasks dienen. Hierzu werden die in Kapitel 4.2.2 vorgestellten Architekturen BeyerModRelu und ResNet18 jeweils für die Tasks Personenklassifikation und Schätzung der Oberkörperorientierung trainiert. Im Folgenden wird zuerst auf die Baseline für die Personenklassifikation eingegangen. Daraufhin folgt die Baseline der Orientierungsschätzung.

Baseline Personenklassifikation

Die genauen Trainingsparameter für die Personenklassifikations-Baseline sind in Tabelle 5.1 dargestellt. Die dort genannten Netzwerkarchitekturen (BeyerModRelu und ResNet18) für die Personenklassifikation sind in Abbildung 5.1 dargestellt. Die BeyerModRelu-

Parameter	**Wert**
Datensatzzusammenstellung (Kap. 4.2.3)	SuPer + Deep-Orientation, konkateniert
Vorverarbeitung (Kap. 4.2.3)	Skalierung auf 0...1
Initiale Lernrate	$1 \cdot 10^{-2}$, $1 \cdot 10^{-3}$ und $1 \cdot 10^{-4}$
Augmentierung (Kap. 4.2.3)	Keine, Horizontale Spiegelung
Architekturen (Kap. 4.2.2)	ResNet18, BeyerModRelu

Tabelle 5.1: Trainingsparameter für die Personenklassifikations-Baseline
*In **fett** sind variable Parameter dargestellt, welche während der Experimente verändert werden. Zusätzlich gelten die in Kapitel 4.2.3, Abschnitt Konstante Parameter, festgelegten Parameter.*

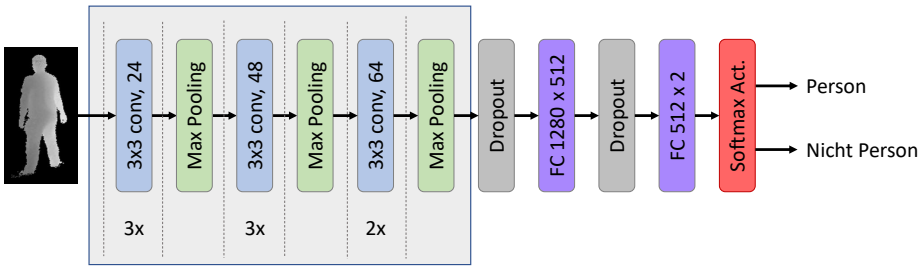
Architektur aus Abbildung 5.1a entspricht der Architektur vom Deep-Orientation-Netzwerk [Lewandowski et al., 2019b], jedoch wird statt der Biternion-Ausgabefunktion eine Softmax-Ausgabefunktion eingesetzt. Die ResNet18-Architektur aus Abbildung 5.1b stammt von der Standard-ResNet18-Architektur von [He et al., 2016] ab, verwendet jedoch nur zwei statt 1000 Ausgabeneuronen. Es ist leicht erkennbar, dass die Anzahl der Feature Maps bei der BeyerModRelu-Archtitektur kleiner ist als bei der ResNet18-Architektur. Das Netzwerk hat insgesamt eine viel geringere Rechenkomplexität und Parameterzahl. Genaue Werte hierzu sind im Kapitel 5.4.1 in Tabelle 5.14 angegeben.

Neben der Analyse der beiden Netzwerkarchitekturen soll hier auch der Einfluss der zufälligen Spiegelung als Augmentierung untersucht werden.

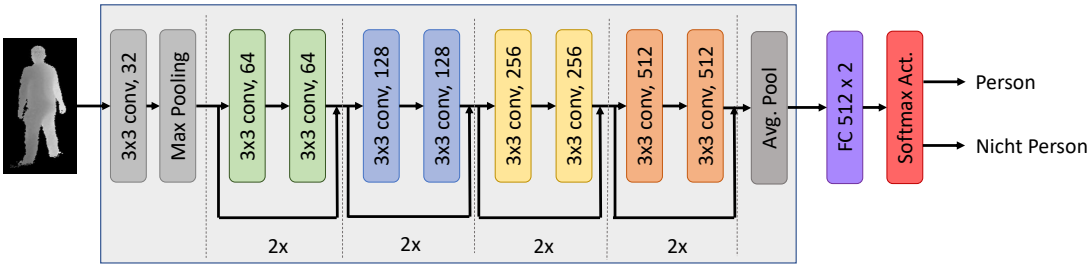
Für jede der vier Kombinationen (beide Netzwerkarchitekturen mit und ohne Augmentierung) wurde mit den Lernraten $1 \cdot 10^{-2}$, $1 \cdot 10^{-3}$ und $1 \cdot 10^{-4}$ trainiert, wobei jedes Training zweimal wiederholt wurde. Die Ergebnisse wurden gemittelt und dann die beste Lernrate ausgewählt. Tabelle 5.2 zeigt für jede Parameterkombination das beste Ergebnis.

Die Ergebnisse zeigen das in den Vorexperimenten aus Kapitel 5.2.1 beschriebene Problem, dass das Netzwerk den Deep-Orientation Datensatz, wegen des Fehlens von Negativbeispielen, nahezu vollständig abtrennen kann. Der Recall nahe 100% ist somit nicht aussagekräftig. Aus diesem Grund wird in den weiteren Auswertungen keine Klassifikationsleistung auf dem Deep-Orientation Datensatz mehr angegeben.

Auf dem schwierigsten der Datensätze, dem SuPer Datensatz, zeigt die Augmentierung



(a) BeyerModRelu-Architektur zur Personenklassifikation



(b) ResNet18-Architektur zur Personenklassifikation

Abbildung 5.1: Netzwerkarchitekturen der Klassifikations-Baselines
Erkennbar ist die einfachere, VGG-ähnliche BeyerModRelu-Architektur sowie die komplexere ResNet18-Architektur mit Blockstruktur, Shortcuts und Average Pooling. Der Netzwerkteil zur Feature-Extraktion ist jeweils grau hinterlegt. Die Schichten für die Batch-Normalisierung und die ReLU-Aktivierungsfunktionen sind der Einfachheit halber nicht dargestellt.

bei beiden Ansätzen einen Performanceanstieg von ca. zwei Prozentpunkten. Auf den anderen beiden Datensätzen macht die Augmentierung nahezu keinen Unterschied in der Erkennungsleistung. Somit bietet sich für die folgenden Multi-Task-Experimente aus Sicht der Personenklassifikation eine Augmentierung der Daten an.

Weiterhin ist in Tabelle 5.2 erkennbar, dass das ResNet18 im Vergleich zum BeyerModRelu-Netzwerk auf dem SuPer Datensatz nur leicht besser abschneidet (drei Prozentpunkte höherer F1-Score), auf dem völlig unbekannten Datensatz SRL jedoch einen um ca. neun Prozentpunkte höhreren Recall besitzt. Das ResNet18 ist somit eindeutig die leistungsfähigere und vor allem die besser generalisierende Architektur. Da sich dies aber auch in der Rechenzeit widerspiegelt (siehe Tabelle 5.14 in Kapitel 5.4.1) und das BeyerModRelu-Netzwerk mehr Verbesserungspotenzial durch Multi-Task Learning hat,

Netzwerk	Augmentierung	F1-Score SuPer	F1-Score TUI-Zuse	Recall Deep-Orientation	Recall SRL
ResNet18	nein	83,22%	96,06%	100,00%	94,75%
ResNet18	ja	**85,24%**	**95,73%**	**99,99%**	**94,51%**
BeyerModRelu	nein	80,12%	94,27%	99,97%	86,07%
BeyerModRelu	ja	**82,19%**	**94,45%**	**99,97%**	**85,36%**

Tabelle 5.2: Ergebnisse der Vortests für die Personenklassifikation
*Dargestellt ist die über zwei Trainings gemittelte Leistung (F1-Score oder Recall) auf den Testdaten für die vier Netzwerkkonfigurationen. Es ist anhand der Ergebnisse erkennbar, dass das ResNet18 besser generalisiert als das BeyerModRelu-Netzwerk. Weiterhin bringt die Augmentierung für beide Netzwerke Vorteile auf dem SuPer Testdatensatz. In **fett** sind die Ergebnisse der besten Konfigurationen dargestellt. Diese werden im Verlauf der Multi-Task-Experimente als Vergleich herangezogen. Im Anhang in Abschnitt A.5.1 sind die Ergebnisse und Varianzen zu allen Lernraten dargestellt.*

werden in den folgenden Experimenten weiterhin beide Netzwerke behandelt.

Baseline Orientierungsschätzung

Für die Orientierungsschätzung kann als Baseline das in Tensorflow[2] auf dem Deep-Orientation Datensatz trainierte BeyerModRelu-Netzwerk aus [Lewandowski et al., 2019b] verwendet werden. Hier wird ein mittlerer absoluter Fehler von 5,44° auf den Testdaten des Deep-Orientation Datensatzes angegeben. Im Rahmen dieser Masterarbeit wurde der Ansatz in Pytorch nachimplementiert und mit den selben Daten trainiert. Weiterhin wird neben der BeyerModRelu-Architektur auch die ResNet18-Architektur sowie eine erweiterte ResNet18-Architektur (ResNet18-Additional-Dense) getestet. Letztere besteht aus dem Feature-Extraktor des ResNet18 und den vollverschalteten Schichten des BeyerModRelu-Netzwerks, wodurch diese Architektur mehr vollverschaltete Gewichte zur Entscheidungsfindung besitzt als das normale ResNet18. Die drei Ansätze sind in Abbildung 5.2 visualisiert.
In Tabelle 5.3 sind die Trainingsparameter dargestellt. Zunächst werden die Netzwerke mit der Skalierung der Tiefenwerte als Vorverarbeitung miteinander verglichen. In Ta-

[2] `https://www.tensorflow.org/` - Abgerufen am 16.12.2019

Parameter	**Wert**
Datensatzzusammenstellung (Kap. 4.2.3)	SuPer + Deep-Orientation, konkateniert
Vorverarbeitung (Kap. 4.2.3	Skalierung auf 0...1, Normalisierung
Initiale Lernrate	$1 \cdot 10^{-1}$, $1 \cdot 10^{-2}$, $1 \cdot 10^{-3}$ und $1 \cdot 10^{-4}$
Augmentierung (Kap. 4.2.3)	Keine, Horizontale Spiegelung
Architekturen (Kap. 4.2.2	ResNet18, ResNet18-Additional-Dense, BeyerModRelu

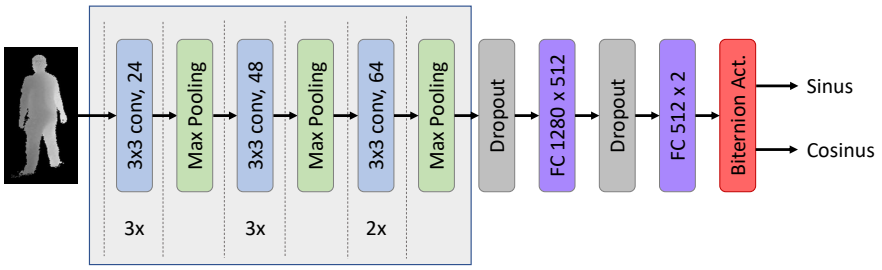
Tabelle 5.3: Trainingsparameter für die Orientierungsschätzer-Baseline
*In **fett** sind variable Parameter dargestellt, welche während den Baseline-Experimenten verändert werden. Zusätzlich gelten die in Kapitel 4.2.3,Abschnitt Konstante Parameter, festgelegten Parameter.*

belle 5.4 sind hierzu die Ergebnisse der Orientierungsschätzung auf dem Testdatensatz Deep-Orientation dargestellt. Hierbei wurden sowohl die Architekturen (BeyerModRelu, ResNet18, ResNet18-Additional-Dense), der Einfluss eines Vortrainings beim ResNet18, als auch der Einfluss einer Augmentierung durch horizontales Spiegeln der Trainingsdaten untersucht. Die vortrainierten Netzwerke nutzen im Feature-Extraktor die auf ImageNet erzeugten Gewichte aus dem *Torchvision Model Zoo*[3].

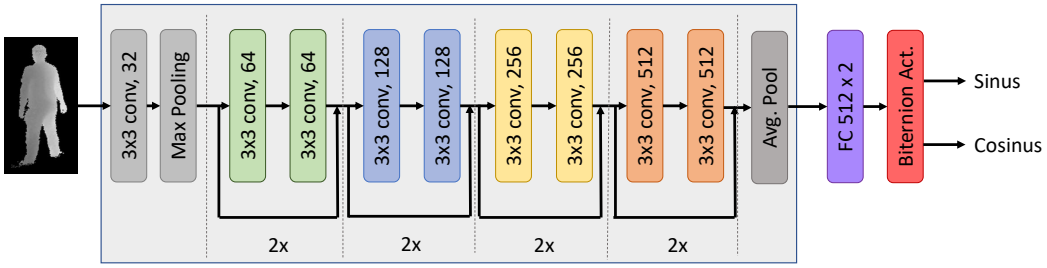
Erkennbar ist, dass die Netzwerke mit Augmentierung durchgehend bessere Ergebnisse liefern. Das BeyerModRelu-Netzwerk erreicht die in [Lewandowski et al., 2019b] angegebene Genauigkeit (vgl. letzte Zeile aus Tabelle 5.4). Außerdem ist weiterhin zu sehen, dass das BeyerModRelu-Netzwerk auf dieser Datengrundlage eine bessere Performance als alle ResNet18-Architekturen zeigt. Dies kann auf die Architektur der ResNets zurückgeführt werden. Dieser Netzwerktyp wurde für die Bildklassifikation entwickelt [He et al., 2016]. Durch die letzte Global-Average-Pooling-Schicht geht der räumliche Kontext der Features verloren, welcher für eine Orientierungsregression hilfreich ist.

Das Vortraining des ResNet ist nach den Ergebnissen in Tabelle 5.4 sehr wichtig, da die Datengrundlage nicht groß genug ist, um ohne Vortraining eine Orientierungsschätzung auf dieser Architektur mit akzeptabler Genauigkeit zu trainieren. Für die BeyerModRelu-Architektur existieren keine vortrainierten Gewichte (z.B. auf ImageNet). Da im Rahmen dieser Masterarbeit die für ein Vortraining benötigte Zeit nicht zur Verfügung stand, kann somit kein Vergleich zu einer vortrainierten BeyerModRelu-Architektur stattfinden.

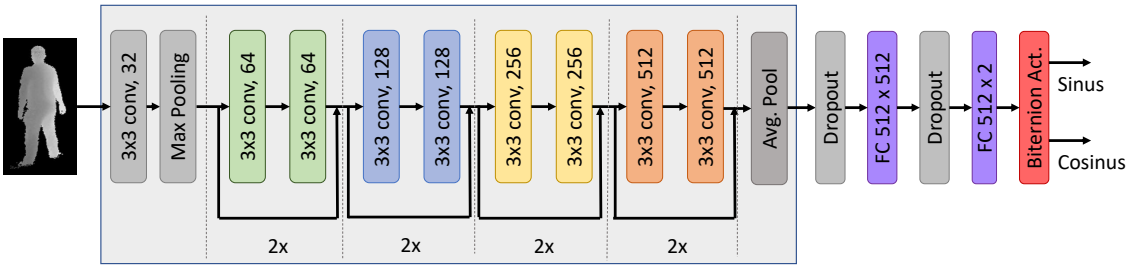
[3] `https://pytorch.org/docs/stable/torchvision/models.html` - Abgerufen am 16.12.2019



(a) BeyerModRelu-Architektur zur Orientierungsschätzung



(b) ResNet18-Architektur zur Orientierungsschätzung



(c) ResNet18-Additional-Dense-Architektur zur Orientierungsschätzung

Abbildung 5.2: Netzwerkarchitekturen der Orientierungsschätzer-Baselines
Der Netzwerkteil zur Feature-Extraktion ist jeweils grau hinterlegt. Die Additional-Dense-Architektur aus Abb. 5.2c kombiniert Feature-Extraktor des ResNet18 (Abb. 5.2b) mit den vollverschalteten Schichten des BeyerModRelu (Abb. 5.2a). Die Schichten für die Batch-Normalisierung und die ReLU-Aktivierungsfunktionen sind der Einfachheit halber nicht dargestellt.

Abschließend lässt sich dennoch aus dieser Versuchsreihe erkennen, dass die vortrainierte ResNet18-Additional-Dense-Architektur bei den ResNet18-Netzwerken die höchste Genauigkeit bei der Orientierungsschätzung liefert. Deshalb werden in den Multi-Task-Experimenten die BeyerModRelu-Architektur (Abbildung 5.2a) und die vortrainierte

ResNet18-Additional-Dense-Architektur (Abbildung 5.2c) auf deren Eigung als Multi-Task-System untersucht.

Netzwerk	**Vortraining**	**mit Augmentierung**	**ohne Augmentierung**
BeyerModRelu	nein	**5,42°**	5,82°
ResNet18	nein	11,29°	12,29°
ResNet18	ja	6,67°	8,42°
ResNet18-Additional-Dense	ja	**6,07°**	7,02°
DeepOrientation	nein	5,44°	-

Tabelle 5.4: Mittlerer absoluter Winkelfehler über den Testdaten Deep-Orientation für die Orientierungsbaseline mit Skalierung als Vorverarbeitung
*Dargestellt sind die über zwei Trainings gemittelten Ergebnisse für die BeyerModRelu- und ResNet18-Architekturen. Hierbei wurde als Vorverarbeitung die Skalierung der Tiefenwerte verwendet. Die ResNet18-Additional-Dense-Architektur kombiniert den Feature-Extraktor vom ResNet18 mit den vollverschalteten Schichten des BeyerModRelu-Netzwerkes. In der letzten Zeile ist das Ergebnis von [*Lewandowski *et al., 2019b] angegeben. In* ***fett*** *sind die Ergebnisse der besten Konfigurationen dargestellt. Diese werden im Verlauf der Multi-Task-Experimente als Vergleich herangezogen. Im Anhang in Abschnitt A.5.2 sind die Ergebnisse und Varianzen zu allen Lernraten und Vorverarbeitungen dargestellt.*

Da beim Verfahren Deep-Orientation [Lewandowski et al., 2019b] als Vorverarbeitung die Normalisierung der Tiefenwerte verwendet wurde, soll hier nochmals kurz evaluiert werden, inwiefern diese Art der Vorverarbeitung für die Orientierungsschätzung vorteilhaft sein könnte. In Tabelle 5.5 sind die Ergebnisse für das ResNet18, jeweils in normaler und erweiterter (Additional-Dense-)Architektur, mit und ohne Augmentierung für die beiden Vorverarbeitungen dargestellt. Aus diesen Zahlen wird kein Vorteil der Normalisierung erkennbar. Da sich die Skalierung der Tiefenwerte für die Personenklassifikation positiv zeigte und hier bei der Orientierungsschätzung keine nennenswerten Unterschiede feststellbar sind, wird sich in den weiteren Multi-Task-Experimenten auf die Skalierung als Vorverarbeitung beschränkt.

Netzwerk	Vortraining	Augmentierung	Skalierung	Normalisierung
ResNet18	ja	nein	8,42°	7,94°
ResNet18	ja	ja	6,67°	7,03°
ResNet18-Additional-Dense	ja	nein	7,02°	6,88°
ResNet18-Additional-Dense	ja	ja	6,07°	6,35°

Tabelle 5.5: Mittlerer Winkelfehler über den Testdaten Deep-Orientation für die Orientierungsbaseline
Dargestellt ist die Abhängigkeit des Winkelfehlers (gemittelt über zwei Trainings) von der Netzwerkstruktur (ResNet18 und ResNet18-Additional-Dense), der Augmentierung und der Vorverarbeitung. Erkennbar ist, dass die Normalisierung keinen nennenswerten Vorteil gegenüber der Skalierung bringt. Im Anhang in Abschnitt A.5.2 sind die Ergebnisse und Varianzen zu allen Lernraten und Vorverarbeitungen dargestellt.

Fazit zu Vorexperimenten und Baselines

In den Vor- und Baseline-Experimenten zeigt sich, dass sowohl die patchbasierte Personenklassifikation, als auch die Oberkörperorientierungsschätzung auf dem gegebenen Datensatzpaar (SuPer und Deep-Orientation) erfolgreich trainiert werden kann. Sowohl die *ResNet18-Architektur*, als auch die *BeyerModRelu-Architektur* lassen sich für die Problemstellung einsetzen. Aufgrund der besseren Ergebnisse der ResNet18-Additional-Dense-Architektur auf der Orientierungsschätzung soll die zusätzliche, vollverschaltete Schicht auch in den Multi-Task-Experimenten bei der Orientierungsschätzung zum Einsatz kommen. Der Einfachheit halber wird jedoch im Folgenden nur noch von der ResNet18-Architektur gesprochen, wobei dies im Rahmen der Orientierungsschätzung die zusätzliche Schicht einschließt. Als Vorverarbeitung stellte sich die *Skalierung der Tiefenwerte* mit dem maximalen Sensorwert als beste Variante heraus. Zusätzlich soll die *Augmentierung* der Trainingsbeispiele in den weiteren Experimenten durchgehend eingesetzt werden. Anhand der Ergebnisse der Baseline-Experimente aus den Tabellen 5.2 (Personenklassifikation) und 5.4 (Orientierungsschätzung) lässt sich die Performance der folgenden Multi-Task-Experimente gut einordnen.

5.3 Multi-Task-Experimente auf Patch-Ebene

Das Multi-Task-System lässt sich sowohl anhand von kompletten Bildern inklusive des Kandidatengenerators (Kapitel 4.1) oder anhand der Bildausschnitte, die das Neuronale Netzwerk verarbeitet (Kapitel 4.2) analysieren. Im Folgenden wird zuerst ausschließlich das Neuronale Netzwerk beschrieben. Die Analyse des Gesamtsystems folgt dann in Kapitel 5.4. Bevor die verschiedenen Architekturen (Single-Scale und Multi-Scale, ResNet18 und BeyerModRelu) miteinander verglichen werden können, muss zu Beginn erst anhand von zwei Experimentalreihen festgelegt werden, wie die beste Epoche im Multi-Task-Szenario bestimmt wird (vgl. Kapitel 4.2.3, Abschnitt Auswahlmethodik) und wie die Datensätze am besten zu Batches zusammengestellt werden (vgl. Kapitel 4.2.3, Abschnitt Erzeugung des kombinierten Trainingsdatensatzes). Da sich die ResNet18-Architektur in den vorangegangenen Single-Task-Experimenten als vielversprechender als die BeyerModRelu-Architektur gezeigt hat, wird diese für die folgenden Experimente jeweils zum Vergleich herangezogen. Hierbei wird die Lernrate zu 0,01 festgesetzt, da dies in den Baseline-Experimenten zu den besten Ergebnissen führte.
Für eine bessere Darstellbarkeit der Ergebnisse wurden zudem die Taskwichtungen zu eins summiert, also gilt:

$$\omega_{klassifikation} + \omega_{orientierung} = 1{,}0 \tag{5.1}$$

$$\omega_{orientierung} = 1{,}0 - \omega_{klassifikation} \tag{5.2}$$

$\omega_{klassifikation}$	Taskwichtungsfaktor Personenklassifikation
$\omega_{orientierung}$	Taskwichtungsfaktor Orientierungsschätzung

Somit lassen sich in den folgenden Analysen die Ergebnisse in Diagrammen antragen, wobei lediglich der Einfluss einer Taskwichtung (Personenklassifikation) auf der X-Achse angegeben wird. Die Taskwichtung der Orientierungsschätzung ergibt sich dann direkt nach Formel 5.2.

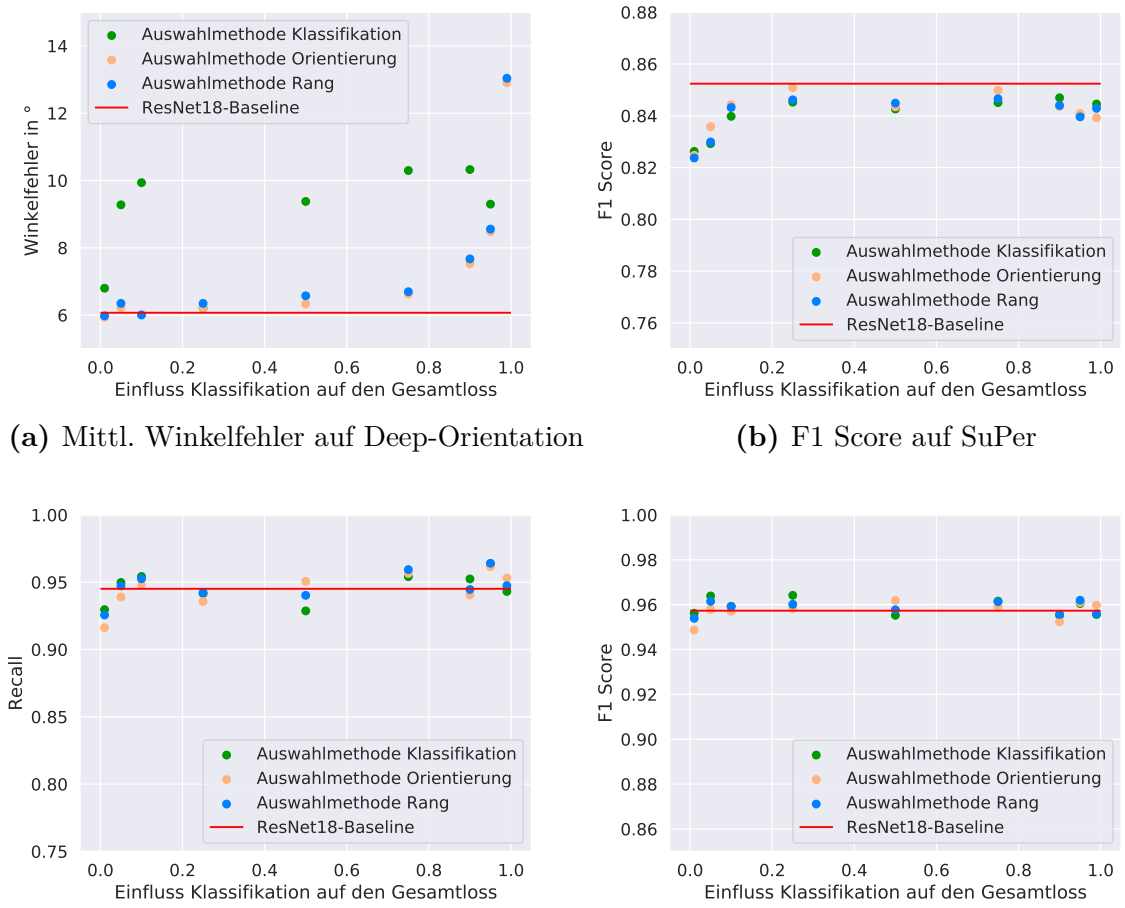
5.3.1 Auswahlmethodik

Um die beste Epoche eines Trainings anhand der Validierungsdaten auszusuchen, muss hierzu ein Maß festgelegt werden. Die drei in Kapitel 4.2.3 vorgestellen Varianten

(Auswahl ohne Fusion anhand eines Tasks, Orientierung oder Klassifikation, und Auswahl des besten Netzes durch Rank-Level-Fusion) sollen nun miteinander verglichen werden. Hierzu wurde die ResNet18-Single-Scale-Multi-Task-Architektur anhand der Parameter aus Tabelle 5.6 trainiert. Bei dieser und den folgenden Versuchsreihen werden die Taskwichtungsfaktoren zu 0,01; 0,05; 0,10; 0,25; 0,50; 0,75; 0,90; 0,95 und 0,99 gewählt, um Ergebnisse für eine möglichst große Abdeckung des Multi-Task-Bereiches zu erhalten, vgl. Kapitel 3.4. Jedes Training wurde dreimal wiederholt und für jeden Durchlauf wurden jeweils die besten Epochen anhand der drei genannten Auswahlmethoden auf den Validierungsdaten ausgewertet. In Abbildung 5.3 sind die gemittelten Ergebnisse der ausgewählten Netzwerke auf den Testdaten dargestellt.
Hierbei wird deutlich, dass eine Auswahl der Netzwerke anhand des Klassifikations-Tasks absolut unzureichende Ergebnisse für die Orientierungsschätzung bringt, vgl. Abbildung 5.3a. Die über die Klassifikation ausgewählten Netzwerke sind, sobald der Klassifikationseinfluss den Wert 0,01 übersteigt, um ca. 4° schlechter, als die per Orientierung ausgewählten Netzwerke. Interessanterweise ist jedoch auch auf den Detektionsdatenätzen (Abbildungen 5.3b bis 5.3d) teilweise ein über die Orientierungsmaße ausgewähltes Netzwerk besser. Dies lässt darauf schließen, dass durch die unzureichende Trennung von Trainings- und Validierungsdaten des SuPer Datensatzes, welche bereits in Kapitel 4.2.1 andiskutiert wurde, keine gute Netzwerkauswahl erfolgen kann.

Parameter	**Wert**
Vorverarbeitung (Kap. 5.2.2 und 5.2.2)	Skalierung auf 0...1
Initiale Lernrate (Kap. 5.2.2 und 5.2.2)	$1 \cdot 10^{-2}$
Augmentierung (Kap. 5.2.2 und 5.2.2)	Horizontale Spiegelung
Architekturen (Kap. 4.2.2)	ResNet18-Single-Scale
Datensatzzusammenstellung (Kap. 5.3.2)	SuPer + Deep-Orientation 50-50
Taskwichtungen (Kap. 3.4)	0,01; 0,05; 0,10; 0,25; 0,50; 0,75; 0,90; 0,95; 0,99
Auswahl der besten Epoche	Per Klassifikation, Per Orientierung, Per Rang-Fusion

Tabelle 5.6: Trainingsparameter für die Multi-Task-Vorexperimente zur Auswahlmethodik
*In **fett** sind variable Parameter dargestellt, welche während der Experimente verändert werden. Zusätzlich gelten die in Kapitel 4.2.3 festgelegten Parameter.*



(a) Mittl. Winkelfehler auf Deep-Orientation

(b) F1 Score auf SuPer

(c) Recall auf SRL

(d) F1 Score auf TUI-Zuse

Abbildung 5.3: Einfluss der Multi-Task-Auswahlmethode

Für die ResNet18-Multi-Task-Systeme ist jeweils dargestellt, wie gut sie auf den Testdatensätzen (gemittelt über drei Trainings) abschneiden. Hierbei ist auf der X-Achse angetragen, wie sich der Trainings-Loss aus Klassifikation und Orientierungsschätzung zusammensetzt. Ist der Klassifikationseinfluss 0,99 (ganz rechts) wird mehr Wert auf die Klassifikation gelegt, während ganz links hauptsächlich nach Orientierung trainiert wird. Die zugehörige ResNet18-Single-Task-Baseline ist jeweils als rote Linie dargestellt. Durch die Farben wird gekennzeichnet, welches Maß zur Auswahl des besten Netzwerks auf den Validierungsdaten verwendet wurde. Die Werte in den Grafiken entsprechen Mittelwerten über drei Trainingsdurchläufe. Die genauen Werte und Standardabweichungen sind im Anhang in Abschnitt A.5.3 angegeben.

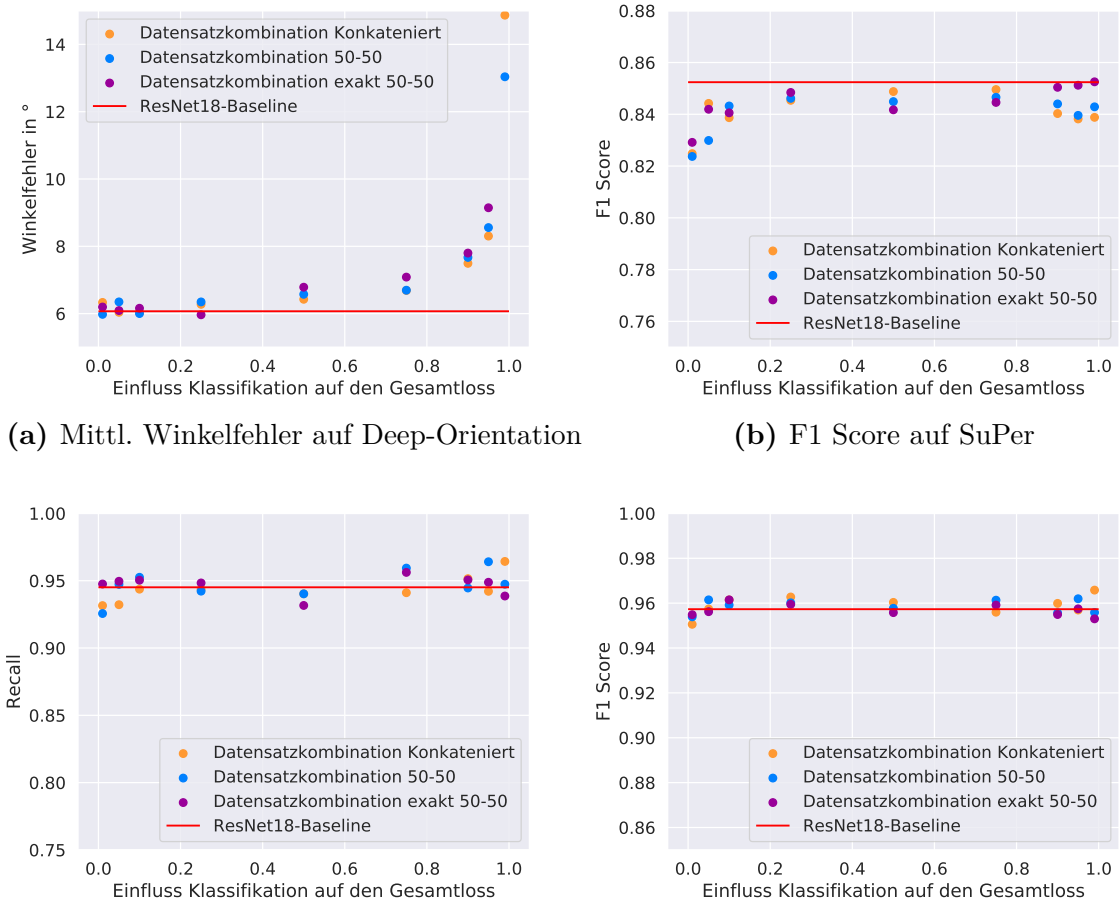
Damit die Klassifikation jedoch nicht gänzlich unbeachtet bleibt, soll für die folgenden Multi-Task-Experimente stets die Auswahl des Netzwerkes mit dem besten Rang (errechnet aus Rang der besten Orientierungsnetze und Rang der besten Klassifikationsnetze) verwendet werden. Diese Methode erreicht ähnlich gute und teils leicht bessere Klassifikationsergebnisse als die reine Orientierungsauswahl und auch auf den Orientierungsdaten werden Netzwerke ausgewählt, welche eine ähnliche Performance zeigen, wie die rein über die Orientierung ausgewählten. Weiterhin bietet diese Variante den Vorteil, dass weitere Tasks leicht in die Auswahl integriert werden können.

5.3.2 Batch-Zusammenstellung der Datensätze

Um für die folgenden Multi-Task-Experimente eine konsistente Datengrundlage zu schaffen, wird evaluiert, wie die Datenbasis des Trainings zu Batches zusammengestellt werden soll. Mit den drei Varianten aus Kapitel 4.2.3 wurden daher jeweils ResNet18-Multi-Task-Netzwerke (Parameterwahl: siehe Tabelle 5.7) trainiert. In Abbildung 5.4 sind die Ergebnisse der Trainings dargestellt. Es ist dabei erkennbar, dass auf den Detektionsdatensätzen (Abbildung 5.4b bis 5.4d) keine der drei Varianten als durchgehend besser oder schlechter zu bewerten ist. Auf dem Deep-Orientation Datensatz (Abbildung 5.4a) ist erkennbar, dass die Variante, die exakt gleichviele Elemente pro Task verwendet, geringfügig schlechter abschneidet als die anderen beiden Varianten.

Parameter	**Wert**
Vorverarbeitung (Kap. 5.2.2 und 5.2.2)	Skalierung auf 0...1
Initiale Lernrate (Kap. 5.2.2 und 5.2.2)	$1 \cdot 10^{-2}$
Augmentierung (Kap. 5.2.2 und 5.2.2)	Horizontale Spiegelung
Architekturen (Kap. 4.2.2)	ResNet18-Single-Scale
Auswahl der besten Epoche (Kap. 5.3.1)	Per Rang-Fusion
Taskwichtungen (Kap. 3.4)	0,01; 0,05; 0,10; 0,25; 0,50; 0,75; 0,90; 0,95; 0,99
Datensatzzusammenstellung	SuPer + Deep-Orientation, konkateniert, 50-50, exakt 50-50

Tabelle 5.7: Trainingsparameter für die Multi-Task-Vorexperimente zur Datensatzzusammenstellung
*In **fett** sind variable Parameter dargestellt, welche während der Experimente verändert werden. Weiterhin ist angegeben, in welchem Kapitel die Parameterwahl behandelt worden ist. Zusätzlich gelten die in Kapitel 4.2.3 festgelegten Parameter.*



(a) Mittl. Winkelfehler auf Deep-Orientation

(b) F1 Score auf SuPer

(c) Recall auf SRL

(d) F1 Score auf TUI-Zuse

Abbildung 5.4: Einfluss der Batch-Zusammenstellung auf die Multi-Task-Performance

Für die ResNet18-Multi-Task-Systeme ist jeweils dargestellt, wie gut sie auf den Testdatensätzen abschneiden. Hierbei ist auf der X-Achse angetragen, wie sich der Trainings-Loss aus Klassifikation und Orientierung zusammensetzt. Ist der Klassifikationseinfluss 0,99 (ganz rechts) wird mehr Wert auf die Klassifikation gelegt, während ganz links hauptsächlich nach Orientierung trainiert wird. Die zugehörige ResNet18-Single-Task-Baseline (konkatenierte Datensätze) ist jeweils als rote Linie dargestellt. Durch die Farben wird gekennzeichnet, wie die Datensätze zu Batches kombiniert wurden. Die Werte in den Grafiken entsprechen Mittelwerten über drei Trainingsdurchläufe. Die genauen Werte und Standardabweichungen sind im Anhang in Abschnitt A.5.4 angegeben.

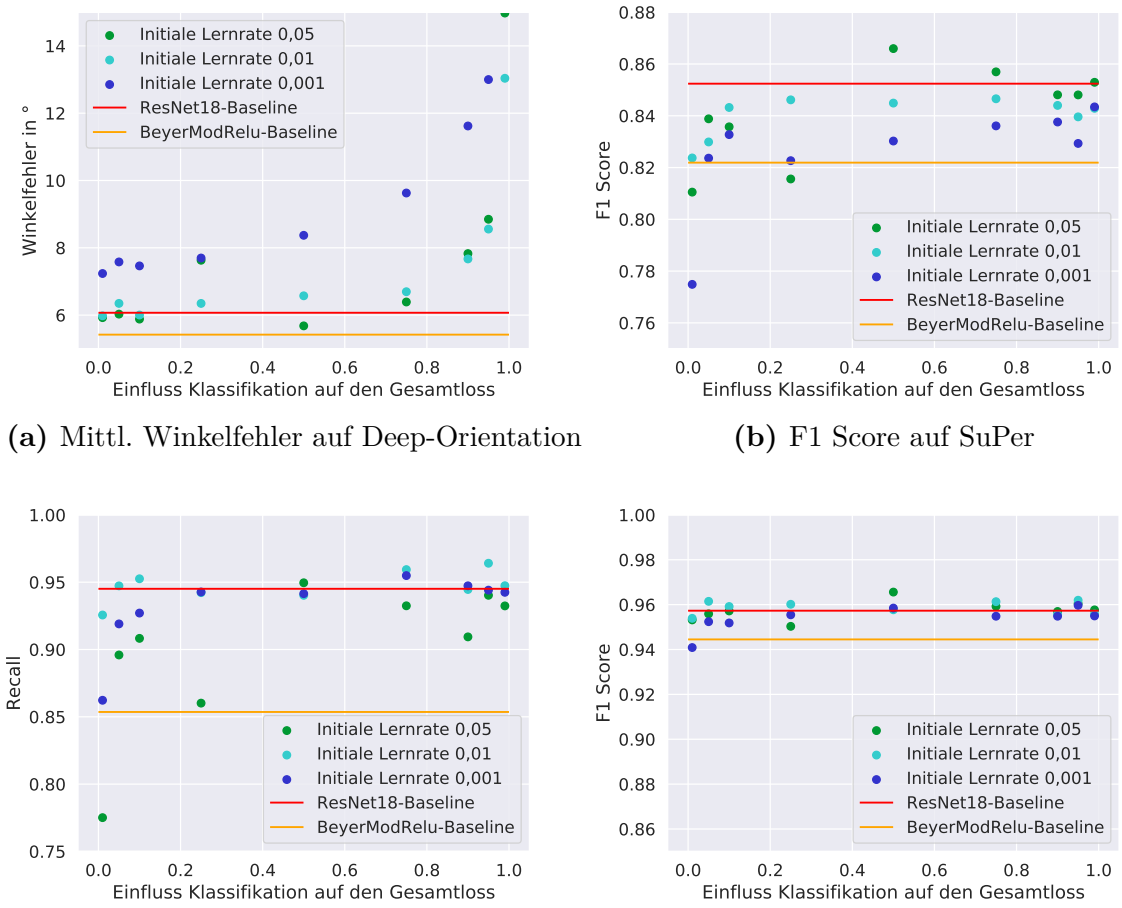
Aus diesem Grund muss sich für die weiteren Experimente nur zwischen dem einfachen Konkatenieren der Datensätze und der normalen 50-50 Strategie entschieden werden. Jedoch ist hierbei sowohl aus Sicht der Recheneffizienz, als auch aus Sicht der Netzwerkperformance keine Entscheidung möglich. Da die reine Konkatenation der Datensätze aber je nach Permutation der Elemente zu Batches führen kann, die nur sehr wenige Labels für die Orientierungsschätzung enthalten, soll in den weiteren Experimenten die normale 50-50 Strategie zum Einsatz kommen. Diese führt zu einer über das Training konstanten Batch Size für die Tasks.

5.3.3 Single-Scale-Multi-Task-Architekturen

Nachdem neben den grundsätzlichen Hyperparametern (Kapitel 4.2.3) und Architekturen (Kapitel 4.2.2) in den vorangegangenen zwei Abschnitten auch die für das Multi-Task Learning spezifischen Problemstellungen der Kombination von Datensätzen und der Auswahl der besten Epochen behandelt wurden, soll im Folgenden auf die Ergebnisse der einfachen, Single-Scale-Multi-Task-Architekturen eingegangen werden. Hierzu werden Experimente für zehn verschiedene Taskwichtungen und drei unterschiedliche Lernraten durchgeführt, vgl. auch Tabelle 5.8. Die Lernrate 0,01 dient hierbei wieder als Ausgangspunkt, da diese in den Baseline-Experimenten zu den besten Ergebnissen führte. Jedes Training wurde dreimal wiederholt.
Zu Beginn wird auf die Ergebnisse der ResNet18-Single-Scale-Multi-Task-Architektur eingegangen. Daraufhin folgt die Betrachtung der BeyerModRelu-Single-Scale-Multi-Task-Architektur und ein abschließender Vergleich zwischen den beiden Ansätzen.

Architektur ResNet18-Single-Scale

In Grafik 5.5 sind die jeweils über drei Trainings gemittelten Ergebnisse der ResNet18-Single-Scale-Architektur dargestellt. In Grafik 5.6 ist exemplarisch für die Lernrate 0,01 dargestellt, wie die einzelnen, ungemittelten Trainings abschneiden bzw. streuen. Die Experimente zeigen deutlich, dass die Taskwichtungen einen großen Einfluss auf die erreichte Performance und die Streuung der einzelnen Netzwerke haben. Wird der Einfluss der Orientierung zu gering gewählt, nimmt deren Performance ab (Abbildung 5.5a) und die Streuung der Netzwerke nimmt zu (Abbildung 5.6a). Wird der Einfluss der Klassifikation zu gering gewählt, nimmt deren Performance auch ab (Abbildung 5.5b).



(a) Mittl. Winkelfehler auf Deep-Orientation

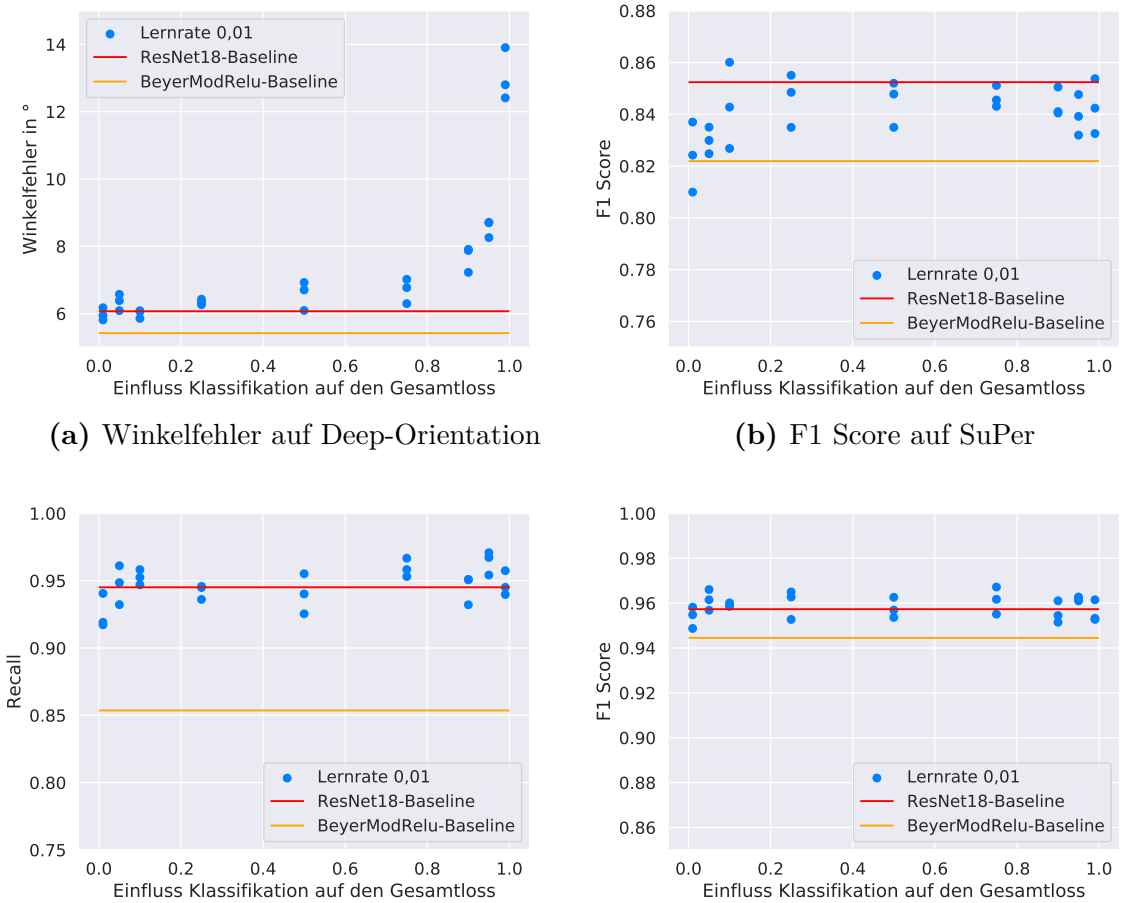
(b) F1 Score auf SuPer

(c) Recall auf SRL

(d) F1 Score auf TUI-Zuse

Abbildung 5.5: Netzwerkperformance der ResNet18-Single-Scale-Multi-Task-Architektur

Dargestellt ist die über drei Durchläufe gemittelte Netzwerkperformance der Single-Scale-Multi-Task-ResNet18-Architektur für verschiedene Lernraten und Taskwichtungen. Erkennbar ist, dass die Netzwerkperformance bei zu kleinen Lernraten (0,001), vor allem auf dem SuPer und Deep-Orientation Datensatz, stark abnimmt. Weiterhin ist erkennbar, dass die Lernrate 0,05 in Verbindung mit einer Taskwichtung von 0,5 für beide Tasks in diesen Experimenten die besten Ergebnisse liefert. Die Baseline-Ergebnisse sind als rote (ResNet18) und orange (BeyerModRelu) Linie eingezeichnet. Die genauen Werte und Standardabweichungen sind im Anhang in Abschnitt A.5.5 angegeben.



(a) Winkelfehler auf Deep-Orientation

(b) F1 Score auf SuPer

(c) Recall auf SRL

(d) F1 Score auf TUI-Zuse

Abbildung 5.6: Ungemittelte Netzwerkperformance der ResNet18-Single-Scale-Multi-Task-Architektur

Dargestellt sind die Ergebnisse auf den Testdaten für die ResNet18-Single-Scale-Architektur mit initialer Lernrate 0,01. Dabei wurden je Taskwichtung drei Trainings durchgeführt, welche hier ungemittelt angetragen sind. Es fällt vor allem in Abbildung 5.6a die mit fallendem Orientierungseinfluss steigende Streuung des Orientierungsschätzers auf. Die genauen Werte, Standardabweichungen und die Grafiken für die Lernraten 0,05 und 0,001 sind im Anhang in Abschnitt A.5.5 nachzuschlagen.

Parameter	**Wert**
Vorverarbeitung (Kap. 5.2.2 und 5.2.2)	Skalierung auf 0...1
Augmentierung (Kap. 5.2.2 und 5.2.2)	Horizontale Spiegelung
Auswahl der besten Epoche (Kap. 5.3.1)	Per Rang-Fusion
Datensatzzusammenstellung (Kap. 5.3.2)	SuPer + Deep-Orientation, 50-50
Taskwichtungen (Kap. 3.4)	0,01; 0,05; 0,10; 0,25; 0,50; 0,75; 0,90; 0,95; 0,99
Initiale Lernrate	$5 \cdot 10^{-2}$, $1 \cdot 10^{-2}$, $1 \cdot 10^{-3}$
Architekturen (Kap. 4.2.2)	ResNet18-Single-Scale, BeyerModRelu-Single-Scale

Tabelle 5.8: Trainingsparameter für die Single-Scale-Multi-Task-Experimente
*In **fett** sind variable Parameter dargestellt, welche während der Experimente verändert werden. Weiterhin ist angegeben, in welchem Kapitel die Parameterwahl behandelt worden ist. Zusätzlich gelten die in Kapitel 4.2.3 festgelegten Parameter.*

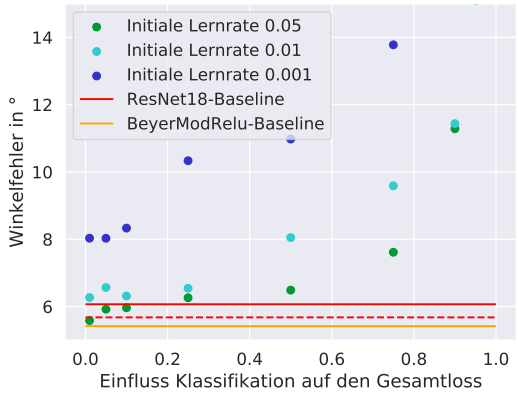
Dies ist jedoch viel geringer ausgeprägt, als bei der Orientierungsschätzung. Das lässt darauf schließen, dass die tiefenbildbasierte Personenklassifikation auf Patchebene ein, im Vergleich zur Orientierungsschätzung, einfacher Task ist, der auch mit sehr geringem Gewicht im Multi-Task-Szenario (z.B. 0,01) trotzdem noch eine gute Leistung erreicht. Insgesamt bietet sich für dieses Szenario eine Taskwichtung der Orientierungsschätzung im Bereich von 0,5 bis 0,9 (Klassifikationsgewicht entsprechend 0,5 bis 0,1) an. Der gleichgewichtete Ansatz (Taskwichtung 0,5 für beide Tasks) zeigt hier eine gute Leistung. Jedoch ist z.B. bei Lernrate 0,01 die Streuung der Netzwerke bei der Taskwichtung 0,75 Orientierung, 0,25 Klassifikation deutlich geringer, vgl. hierzu auch Abbildung 5.6.

Allgemein wird bei diesen Experimentalreihen deutlich, dass einerseits eine in der Baseline optimale Lernrate zwar einen guten Startpunkt für die Lernrate der Multi-Task-Netzwerke liefert, jedoch trotzdem weitere Lernraten miteinander verglichen werden müssen. So zeigt die Lernrate 0,01 für die Multi-Task-Experimente bei der genannten Taskwichtung (0,75 Orientierung, 0,25 Klassifikation) zwar eine vergleichbare Leistung zu den Baselines, es wird aber keine Verbesserung der Genauigkeit erreicht. Dennoch kann der Rechenaufwand durch den eingesetzten Multi-Task-Ansatz im Gegensatz zu zwei getrennten Systemen fast halbiert werden. Die drei vollverschalteten Schichten (eine Schicht für die Klassifikation, zwei für die Orientierungsschätzung, vgl. Abbildung 4.9) enthalten lediglich 264.708 Gewichte (263.682 für die Orientierungsschätzung, 1026 für die Klassifikation). Am Gesamtnetzwerk (11.441.220 Gewichte) macht deshalb der Feature-Extraktor (11.176.512 Gewichte) 97,7% der Gewichte des

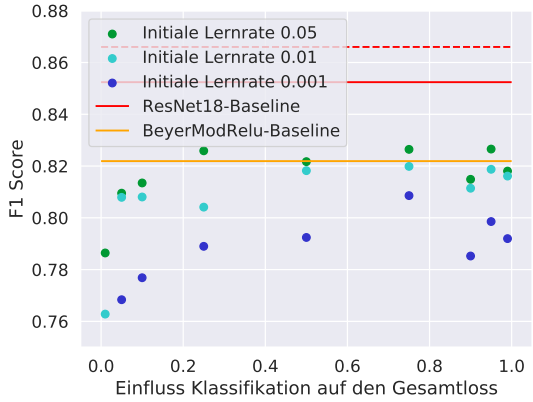
Netzwerkes aus. Im Vergleich zu einem System aus zwei Feature-Extraktoren und den drei vollverschalteten Schichten (2*11.176.512 + 264.708 Gewichte) besitzt das vorgestellte System (11.441.220) somit lediglich 50,6% der Gewichte.
Wird darüber hinaus noch eine andere Lernrate gewählt (z.B. 0,05), können Netzwerke trainiert werden, die sowohl bei der Orientierungsschätzung, als auch bei der Personenklassifikation auf allen vier Datensätzen bessere Ergebnisse zeigen und dennoch die selbe Rechenzeit benötigen. Hierbei zeigte sich aber eine andere Taskwichtung (0,5 für beide Tasks) als optimale Wahl, vgl. hierzu Abbildung 5.5.
Aufgrund begrenzter Zeit für Experimente konnte nur diese begrenzte Anzahl an Lernraten getestet werden. Die Untersuchung weiterer Lernraten (z.B. größer als 0,05 oder zwischen 0,01 und 0,05) kann somit nochmals zu einer weiteren Verbesserung der Systemleistung führen.

Architektur BeyerModRelu-Single-Scale

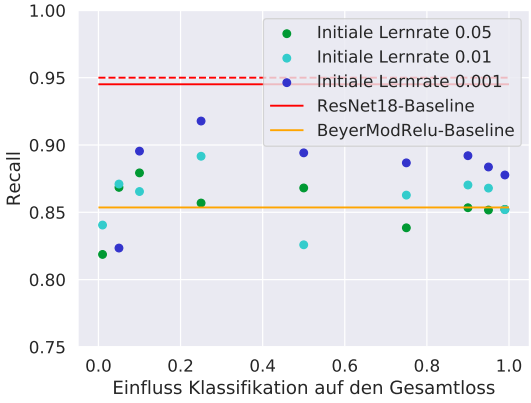
Die selben Experimente wurden auch für die BeyerModRelu-Architektur durchgeführt. In den Baseline-Experimenten zeigte sich diese Architektur als, im Vergleich zum ResNet18, leicht überlegen in der Orientierungsschätzung, aber auch deutlich unterlegen in der Personenerkennung. Beim Einsatz als Multi-Task-System unterscheiden sich die Ergebnisse stark zwischen den Tasks und den Datensätzen. Die Orientierungsschätzung (Abbildung 5.7a) ist im Multi-Task-Fall (Taskwichtung Orientierung > 0.01) schlechter als die BeyerModRelu-Baseline. Das Netzwerk erreicht ungefähr die Netzwerkleistung der ResNet18-Baseline. Die Klassifikationstasks (Abbildungen 5.7b bis 5.7d) zeigen einige interessante Erkenntnisse. Auf dem SuPer Datensatz erreicht die Architektur ungefähr ihre Baseline-Performance. Auf den Datensätzen SRL und TUI-Zuse kann diese jedoch stark übertroffen werden. Bei Letzterem wird sogar das Niveau der ResNet18-Baseline erreicht. Es fällt dabei dennoch auf, dass das BeyerModRelu-Netzwerk bei der Klassifikation auf dem SuPer und TUI-Zuse Datensatz, wie auch bei der Orientierungsschätzung, von einer hohen Lernrate profitiert. Jedoch gerade bei dem komplett unabhängigen Datensatz SRL ist eine viel niedrigere Lernrate vorteilhaft, vgl. Abbildung 5.7c). Insgesamt ist das BeyerModRelu-Netzwerk mit einer Lernrate von 0,05 bei der Orientierungsschätzung auf dem Niveau der ResNet18-Baseline. Bei der Klassifikation wird das Niveau der BeyerModRelu-Baseline erreicht und teilweise leicht übertroffen. Es kann zwar insgesamt nicht die Klassifikationsleistung des ResNet18-Multi-Task-



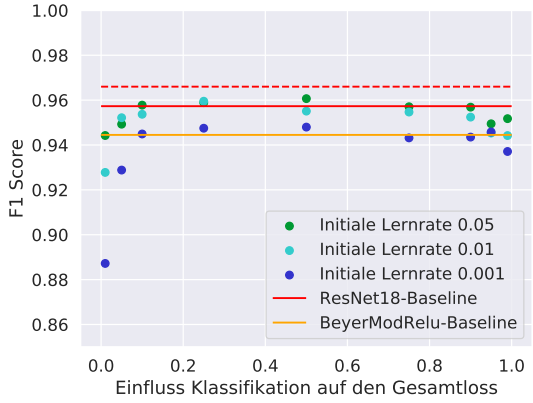
(a) Winkelfehler auf Deep-Orientation



(b) F1 Score auf SuPer



(c) Recall auf SRL



(d) F1 Score auf TUI-Zuse

Abbildung 5.7: Netzwerkperformance der BeyerModRelu-Multi-Task-Architektur *Dargestellt ist die über drei Durchläufe gemittelte Netzwerkperformance der BeyerModRelu-Single-Scale-Multi-Task-Architektur für verschiedene Lernraten und Taskwichtungen. Erkennbar ist, dass dieser Ansatz eine gleichbleibende oder schlechtere Leistung als die Baseline bietet. Die Baseline-Ergebnisse sind als rote (ResNet18-Baseline), rot gestrichelte (beste ResNet18-Single-Scale-Multi-Task-Architektur) und orange (BeyerModRelu-Baseline) Linie eingezeichnet. Die genauen Werte und Standardabweichungen sind im Anhang in Abschnitt A.5.6 angegeben.*

Ansatzes erreicht werden, dafür ist dieser Netzwerktyp jedoch auch viel recheneffizienter als der ResNet18-Ansatz. Daraus lässt sich schlussfolgern, dass das höhere Abstraktionsniveau der Features beim tieferen ResNet zu einer verbesserten Leistung beim Multi-Task Learning führen kann. Dennoch ist auch erkennbar, dass Multi-Task Learning mit der BeyerModRelu-Netzwerkarchitektur möglich ist und abhängig von den Aufgaben auch Verbesserungen im Vergleich zu den Single-Task-Leistungen erreicht werden können.

Fazit zu den Single-Scale-Architekturen

Abschließend lässt sich zusammenfassen, dass sich beide vorgestellten Architekturen zum gleichzeitigen Lernen verschiedener Aufgaben eignen. Bereits mit wenig Aufwand finden sich Parameter (Initiale Lernrate im Bereich der Baselines und gleiche Taskwichtungen für die Tasks), welche zu einem Multi-Task-Netzwerk führen, welches für die Tasks ein Niveau im Bereich der Baselines erreicht. Durch weitere Experimente finden sich auch bessere Parametersätze, welche zu Netzwerken führen, welche die Baseline-Performance sogar übertreffen.
Es ist aber auch festzustellen, dass die BeyerModRelu-Multi-Task-Architektur im Vergleich zum ResNet18 eine geringere Gesamtleistung im beschriebenen Szenario erreicht. Aufgrund dessen wird in den folgenden Multi-Scale-Experimenten hauptsächlich auf die ResNet18-Architektur eingegangen.

5.3.4 Multi-Scale-Multi-Task-Architekturen

Nachdem nun die Netzwerkleistung der verschiedenen Single-Scale-Ansätze des Multi-Task Learning diskutiert wurden, werden nun die beiden Multi-Scale-Ansätze *ResNet18-MultiScale-LAP* und *ResNet18-MultiScale-GAP* aus Kapitel 4.2.2 evaluiert. Zuerst ist ein Blick auf die Rechenkomplexität hilfreich. Hierfür sind in Tabelle 5.9 die zusätzlichen Gewichte der beiden Netzwerke dargestellt. Dabei ist erkennbar, dass die zusätzlichen Gewichte für den Multi-Scale-Ansatz, vor allem bei der GAP-Architektur, lediglich einen einstelligen Prozentsatz an den Netzwerken ausmachen. Deshalb ist keine starke Erhöhung der Inferenzlaufzeiten zu erwarten. Es ist dabei anzumerken, dass die zusätzlichen Gewichte der GAP-Architektur ausschließlich im vollverschalteten Netzwerkteil hinzukommen, während bei der LAP-Architektur durch die 1x1-Faltungen

Netzwerk	**Gewichte**	**Zus. Gewichte zum ResNet18**	**Zus. Gewichte zum ResNet18 in %**
ResNet18-Single-Scale	11.441.220	-	-
ResNet18-Multi-Scale-GAP	11.671.492	230.272	2,0%
ResNet18-Multi-Scale-LAP	12.268.604	827.384	7,2%

Tabelle 5.9: Zusätzliche Netzwerkgewichte der Multi-Scale-Architekturen
Dargestellt ist, wieviele Gewichte die ResNet18-Multi-Scale-GAP- und ResNet18-Multi-Scale-LAP-Architekturen im Vergleich zur ResNet18-Single-Scale-Architektur haben.

auch zusätzliche Faltungsgewichte hinzukommen.

Wie bei den vorherigen Experimenten sind auch für die Multi-Scale-Netzwerke enstprechende Single-Task-Baselines notwendig, um bewerten zu können, inwiefern sich die Multi-Scale-Netzwerke zum gleichzeitigen Lernen mehrerer Tasks eignen. Daher wurden, analog zu Abschnitt 5.2.2, jeweils Single-Task-Netzwerke für die Orientierungsschätzung und für die Personenklassifikation auf Basis der Multi-Scale-Netzwerke (ResNet18-Multi-Scale-GAP und ResNet18-Multi-Scale-LAP) trainiert. Deren Ergebnisse auf den Testdaten sind in Tabelle 5.10 dargestellt. Es wird dabei deutlich, dass die GAP-Architektur im Vergleich zu der Single-Scale-Herangehensweise auf allen Datensätzen minimal besser abschneidet. Vor allem die Verbesserung auf dem Deep-Orientation Datensatz (Mittlerer Winkelfehler 5,72° bei der Multi-Scale-GAP-Architektur gegen 6,07° bei der Single-Scale-Architektur) zeigt, dass die größere Anzahl an Gewichten in den vollverschalteten Schichten in Verbindung mit einem höheren räumlichen Bezug der Features für eine Orientierungsregression vorteilhaft ist. Dies zeigte sich bereits bei den Baseline-Experimenten zur Orientierungsschätzung in Abschnitt 5.2.2.

Die LAP-Architektur-Baseline zeigt jedoch keine Vorteile im Vergleich zur Single-Scale-Baseline: Sie hat 7,2% mehr Gewichte im Netzwerk und erreicht dennoch keine deutliche Verbesserung. Stattdessen nehmen die Baseline-Performancemaße sogar im Vergleich zur Single-Scale-Architektur ab (Datensätze Deep-Orientation und SuPer).

Nach den Baselines können die Architekturen im Multi-Task-Szenario getestet werden. Hierfür wird wieder die Netzwerkperformance der beiden Architekturen auf den vier Datensätzen herangezogen. Die Trainingsparameter sind in Tabelle 5.11 angetragen. Aufgrund der begrenzten Zeit für die experimentellen Untersuchungen wurden die Trai-

Netzwerk	**Mittlerer Winkelfehler Deep-Orientation**	**F1-Score SuPer**	**F1-Score TUI-Zuse**	**Recall SRL**
ResNet18-Single-Scale	6,07°	85,20%	95,73%	94,50%
ResNet18-Multi-Scale-GAP	5,72°	85,40%	95,90%	93,70%
ResNet18-Multi-Scale-LAP	6,34°	84,40%	95,60%	94,70%

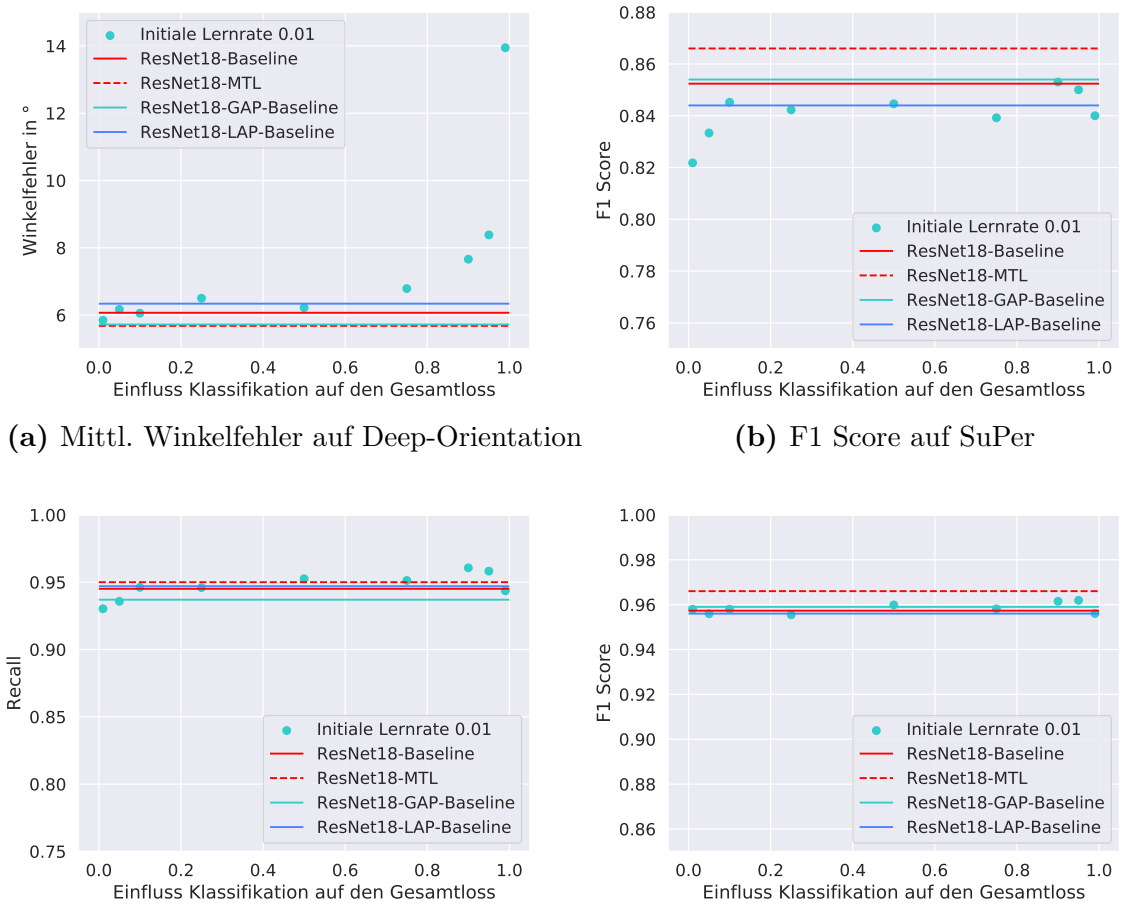
Tabelle 5.10: Baseline-Ergebnisse der Multi-Scale-Ansätze.
Angetragen sind die Ergebnisse der Single-Task-Baselines für die GAP- und LAP-Multi-Scale-Ansätze. Die Maße sind auf den Testdaten anhand von jeweils drei Trainings für die Lernraten 0,05, 0,01 und 0,001 bestimmt. Das Ergebnis für die beste Lernrate ist in der Tabelle eingetragen. Die optimalen Ergebnisse zeigten sich für Lernrate 0,05 (Orientierung und Klassifikation von GAP, sowie Orientierung von LAP) bzw. 0,01 (Klassifikation LAP). Zum Vergleich zeigt die erste Zeile die Baseline-Ergebnisse der ResNet18-Single-Scale-Architektur. Die genauen Ergebnisse und Standardabweichungen für alle Lernraten sind im Anhang in Kapitel A.5.7 angetragen.

nings für die Multi-Scale-Experimente lediglich für eine Lernrate (0,01) durchgeführt. In Abbildung 5.8 sind die Ergebnisse für die ResNet18-Multi-Scale-GAP-Architektur, in Abbildung 5.9 die Ergebnisse für die ResNet18-Multi-Scale-LAP-Architektur dargestellt. Die GAP-Architektur zeigt bei der Orientierungsschätzung (Abbildung 5.8a) und bei der Personenklassifikation auf dem SuPer-Datensatz (Abbildung 5.8b) einen Rückgang der Netzwerkleistung im Vergleich zur Baseline. Auf diesen Datensätzen fällt die Architektur sogar hinter die ResNet18-Single-Scale-Baseline zurück (Rote Linie in Abbildungen 5.8 und 5.9). Auf dem Datensatz SRL (Abbildung 5.8c) ist eine Verbesserung der Erkennungrate durch das Multi-Task Learning ersichtlich. Insgesamt ist die Architektur jedoch, vor allem aufgrund der höheren Rechenkomplexität und der geringeren Leistung auf den Datensätzen Deep-Orientation und SuPer, nicht der ResNet18-Single-Scale-Architektur (Abschnitt 5.3.3) vorzuziehen.
Die Architektur ResNet18-LAP (Abbildung 5.9) zeigt auf allen Datensätzen im Multi-Task-Training ähnliche Ergebnisse wie die ResNet18-GAP Architektur. Die LAP-Variante behält ihre Leistung aus den Baselines, im Gegensatz zu der GAP-Variante, bei, bringt jedoch keine weitere Verbesserung mit sich. Somit ist auch diese Architektur aufgrund der Klassifikations- und Orientierungsleistung und der leicht höheren Anzahl an Gewichten für den in dieser Masterarbeit beschriebenen Multi-Task-Ansatz weniger

Parameter	**Wert**
Vorverarbeitung (Kap. 5.2.2 und 5.2.2)	Skalierung auf 0...1
Augmentierung (Kap. 5.2.2 und 5.2.2)	Horizontale Spiegelung
Auswahl der besten Epoche (Kap. 5.3.1)	Per Rang-Fusion
Datensatzzusammenstellung (Kap. 5.3.2)	SuPer + Deep-Orientation, 50-50
Initiale Lernrate (Kap. 5.2.2 und 5.2.2)	$1 \cdot 10^{-2}$
Taskwichtungen (Kap. 3.4)	0,01; 0,05; 0,10; 0,25; 0,50; 0,75; 0,90; 0,95; 0,99
Architekturen (Kap. 4.2.2)	ResNet18-Multi-Scale-GAP, ResNet18-Multi-Scale-LAP

Tabelle 5.11: Trainingsparameter für die Multi-Scale-Multi-Task-Experimente
*In **fett** sind variable Parameter dargestellt, welche während der Experimente verändert werden. Weiterhin ist angegeben, in welchem Kapitel die Parameterwahl behandelt worden ist. Zusätzlich gelten die in Kapitel 4.2.3 festgelegten Parameter.*

geeignet, als die ResNet18-Single-Scale-Variante.
Auf die BeyerModRelu-Multi-Scale-Architektur wird im Anhang in Kapitel A.3 genauer eingegangen, da sich die BeyerModRelu-Single-Scale-Architektur in den vorangegangenen Experimenten als grundsätzlich weniger leistungsfähig als die ResNet18-Architektur gezeigt hat. Dort ist erkennbar, dass die Multi-Scale-Architektur bei der Orientierungsschätzung viel schlechter als alle bisherigen Netzwerke abschneidet. Bei der Personenklassifikation können im Vergleich zur Single-Scale-Baseline auch keine Verbesserungen auf dem SuPer Datensatz erreicht werden. Auf den Datensätzen SRL und TUI-Zuse können dagegen Verbesserungen erzielt werden, dennoch liegt die Leistung auf diesen Datensätzen immer noch hinter der ResNet18-Baseline.



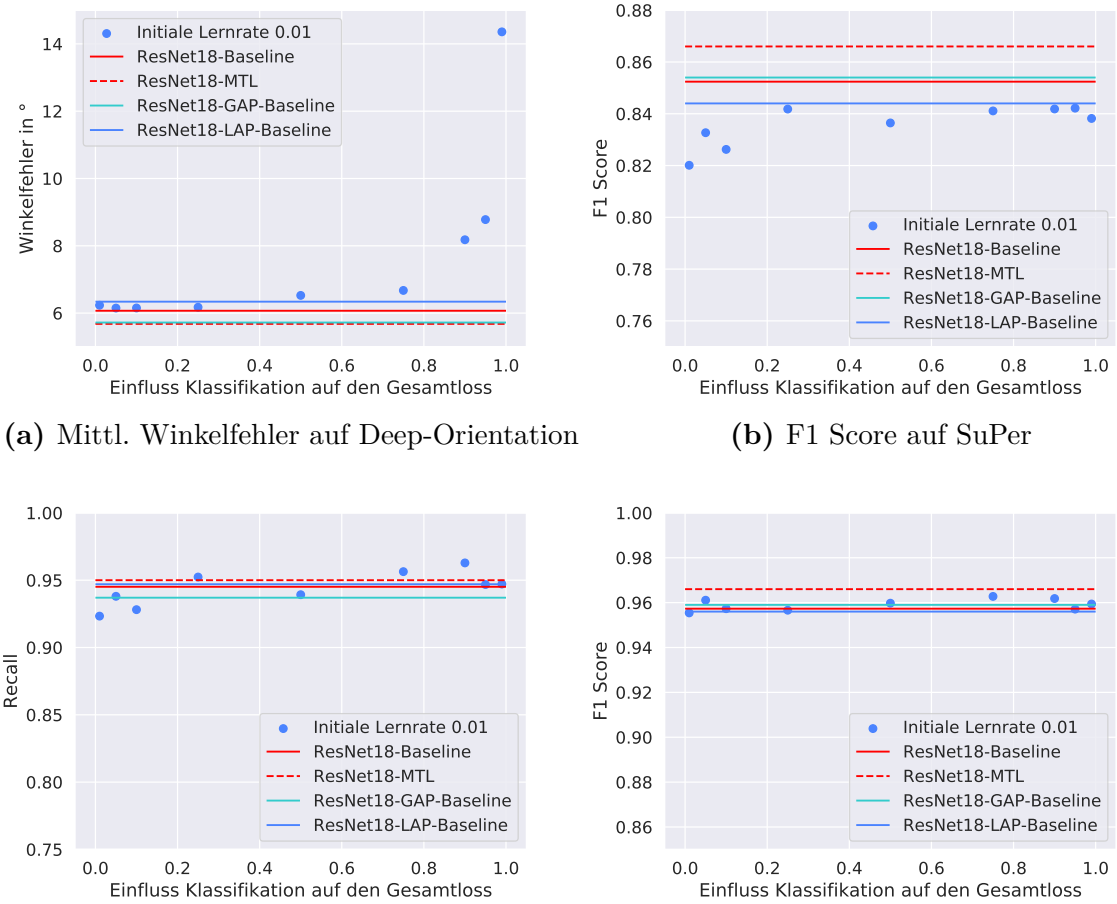
(a) Mittl. Winkelfehler auf Deep-Orientation

(b) F1 Score auf SuPer

(c) Recall auf SRL

(d) F1 Score auf TUI-Zuse

Abbildung 5.8: Netzwerkperformance der ResNet18-GAP-Multi-Task-Architektur *Dargestellt ist die über drei Durchläufe gemittelte Netzwerkperformance der Multi-Task-Architektur ResNet18-Multi-Scale Global Average Pooling für die Lernrate 0,01 und verschiedene Taskwichtungen. Die Multi-Scale-Baseline-Ergebnisse sind eingezeichnet als hellblaue (ResNet18-GAP) und dunkelblaue (ResNet18-LAP) Linie. Zum Vergleich ist in rot zusätzlich die ResNet18-Single-Scale-Baseline und in rot gestrichelt die beste ResNet18-Single-Scale-Multi-Task-Architektur aus Abschnitt 5.3.3 dargestellt. Die genauen Werte und Standardabweichungen sind im Anhang in Abschnitt A.5.8 angegeben.*



(a) Mittl. Winkelfehler auf Deep-Orientation

(b) F1 Score auf SuPer

(c) Recall auf SRL

(d) F1 Score auf TUI-Zuse

Abbildung 5.9: Netzwerkperformance der ResNet18-LAP-Multi-Task-Architektur *Dargestellt ist die über drei Durchläufe gemittelte Netzwerkperformance der Multi-Task-Architektur ResNet18-Multi-Scale Local Average Pooling für die Lernrate 0,01 und verschiedene Taskwichtungen. Die Multi-Scale-Baseline-Ergebnisse sind eingezeichnet als hellblaue (ResNet18-GAP) und dunkelblaue (ResNet18-LAP) Linie. Zum Vergleich ist in rot zusätzlich die ResNet18-Single-Scale-Baseline und in rot gestrichelt die beste ResNet18-Single-Scale-Multi-Task-Architektur aus Abschnitt 5.3.3 dargestellt. Die genauen Werte und Standardabweichungen sind im Anhang in Abschnitt A.5.9 angegeben.*

5.3.5 Fazit zur besten Netzwerkstruktur

Zum abschließenden Vergleich sind in Tabelle 5.12 die Netzwerkleistungen der Baselines (ResNet18, BeyerModRelu) und der Multi-Task-Architekturen (ResNet18-Single-Scale, ResNet18-Multi-Scale-GAP, BeyerModRelu-Single-Scale) dieser Arbeit eingetragen. Anhand der Werte ist deutlich erkennbar, dass die ResNet18-Single-Scale-Multi-Task-Architektur (Abschnitt 5.3.3) die für diese Masterarbeit am besten geeignete Netzwerkarchitektur ist. Sie erreicht sehr gute Performancemaße auf allen vier Datensätzen, teilweise deutlich über denen der BeyerModRelu-Architektur (Abschnitt 5.3.3) und leicht besser als die ResNet18-Baseline-Ergebnisse. Zudem benötigt sie weniger Ressourcen als die Multi-Scale-Ansätze aus Abschnitt 5.3.4 und erreicht dennoch bessere Ergebnisse. Die genauen Trainingsparameter sind in Tabelle 5.13 dargestellt.

Abschließend bietet sich ein Vergleich zu anderen Verfahren an. Anhand der Werte in Tabelle 5.12 ist erkennbar, dass das System nahezu die gleiche Leistung bei der Regression der Oberkörperorientierung wie die Deep-Orientation-Baseline (5,44°) von [Lewandowski et al., 2019b] erreicht. Um zusätzlich auch die Detektionsleistung vergleichen zu können, wird im folgenden Abschnitt das Gesamtsystem aus Kandidatengenerator und Neuronalem Netzwerk anhand des SuPer Datensatzes analysiert.

Netzwerk	**MAAE Deep-Orientation**	**F1-Score SuPer**	**F1-Score TUI-Zuse**	**Recall SRL**
ResNet18-Single-Scale-Baseline	6,07°	85,24%	95,73%	94,51%
BeyerModRelu-Single-Scale-Baseline	5,42°	82,19%	94,45%	85,36%
ResNet18-Single-Scale	5,68°	86,60%	96,60%	95,00%
ResNet18-Multi-Scale-GAP	6,22°	84,50%	96,00%	95,30%
BeyerModRelu-Single-Scale	6,26°	82,20%	96,10%	86,80%

Tabelle 5.12: Gegenüberstellung von Baselines und Multi-Task-Netzwerkarchitekturen
In der Tabelle können die beiden Baselines (ResNet18 und BeyerModRelu), sowie die drei Multi-Taks-Architekturen (ResNet18-Single-Scale, ResNet18-Multi-Scale-GAP und BeyerModRelu-Single-Scale) anhand der Performancemaße auf den Testdaten miteinander verglichen werden. Erkennbar ist, dass die ResNet18-Single-Scale-Multi-Task-Architektur die beste Leistung für das im Rahmen dieser Masterarbeit beschriebene Anwendungsszenario bietet.

Parameter	**Wert**
Vorverarbeitung (Kap. 5.2.2 und 5.2.2)	Skalierung auf 0...1
Augmentierung (Kap. 5.2.2 und 5.2.2)	Horizontale Spiegelung
Auswahl der besten Epoche (Kap. 5.3.1)	Per Rang-Fusion
Datensatzzusammenstellung (Kap. 5.3.2)	SuPer + Deep-Orientation, 50-50
Architektur (Kap. 4.2.2)	ResNet18-Single-Scale
Taskwichtungen (Kap. 5.3.3)	0,5 für beide Tasks
Initiale Lernrate (Kap. 5.3.3)	$5 \cdot 10^{-2}$
Schwellwert Personenklassifikation (Kap. 5.3.3)	0.68574

Tabelle 5.13: Parameter der besten Multi-Task-Architektur
Zusätzlich gelten die in Kapitel 4.2.3 festgelegten Parameter.

5.4 Multi-Task-Experimente auf Bild-Ebene

Bisher konnten in den Experimenten viele Erkenntnisse zu den verschiedenen Typen Neuronaler Multi-Task-Netzwerke gewonnen werden. Hierzu wurde die Performance der Netzwerke auf den Bildausschnitten der Datensätze analysiert. Dabei stellte sich die ResNet18-Single-Scale-Multi-Task-Architektur mit initialer Lernrate 0,05 und Taskwichtungen 0,5 für beide Tasks als das beste Netzwerk heraus.
Um ein besseres Verständnis für die Gesamtleistung des kompletten Ansatzes zu erhalten, soll nun von der Klassifikation und Regression der Bildausschnitte wieder zurück zum ursprünglichen Problem, der Detektion und Orientierungsschätzung von Personen in Tiefenbildern, übergegangen werden. Hierzu wird zuerst die Detektionsleistung anhand von Detection-Error-Tradeoff-Kurven (Kapitel 2.4) mit anderen Verfahren verglichen. Daraufhin folgt ein Test des Gesamtsystems in einer realen Büroumgebung. Hierbei werden typische Fehlerfälle von Detektion und Orientierungsschätzung diskutiert.

5.4.1 Laufzeiten und Komplexität

Das im Rahmen dieser Masterarbeit entworfene System zur Personendetektion und Schätzung der Oberkörperorientierung besteht grundsätzlich aus zwei Verarbeitungsschritten, vgl. hierzu auch Kapitel 4.1. Zuerst werden durch einen Kandidatengenerator Bildausschnitte (Patches) im Eingabebild gefunden, die potenziell Personen enthalten könnten. Durch diese Vorverarbeitung muss der zweite Verarbeitungsschritt diese Patches nur klassifizieren. Das Neuronale Netz verwendet also das Eingabebild nie im

ganzen, sondern nur das in einzelne interessante Regionen (Regions of Interest, ROIs) aufgeteilte Bild.
Für die Inferenzzeit spielt die Vorverarbeitung daher eine wichtige Rolle, da sich die Gesamtrechenzeit des Detektors aus der Rechenzeit des Kandidatengenerators, der Anzahl der erzeugten Kandidaten und der Rechenzeit des Klassifikators nach Formel 5.3 ergibt. Durch die Optimierungen bei der Netzwerkinferenz ist die Angabe einer Funktion für die Rechenzeit des Netzwerkes in Abhängigkeit der Anzahl Kandidaten ($T_{klassifikator}(N_{kandidaten})$) schwierig möglich. Dennoch ist für eine hohe Geschwindigkeit des Gesamtsystems sowohl eine möglichst niedrige Kandidatenzahl, als auch ein möglichst schnelles Netzwerk zur Verarbeitung der Patches wünschenswert.

$$T_{detektion} = T_{kandidatengenerator} + T_{klassifikator}(N_{kandidaten}) \tag{5.3}$$

$T_{detektion}$	Gesamtverarbeitungszeit für ein Bild
$T_{kandidatengenerator}$	Verarbeitungszeit des Kandidatengenerators
$T_{klassifikator}$	Verarbeitungszeit des Netzwerkes, abhängig von der Anzahl Eingabedaten (Batch Size)
$N_{kandidaten}$	Anzahl der Kandidaten pro Bild

Kandidatengenerator

Die konstante Rechenzeit des Kandidatengenerators wurde auf einem Kern eines AMD Threadripper 2920x Prozessors (3,5Ghz) mit vier Millisekunden gemessen. Eine Analyse der Parameter des Kandidatengenerators und die damit verbundene Anzahl an Kandidaten pro Eingangsbild befinden sich im Anhang in Kapitel A.4. Dort zeigt sich, dass im Mittel von 20 Kandidaten pro Eingabebild auszugehen ist ($N_{kandidaten} = 20$).

Neuronales Multi-Task-Netwzerk

Die Geschwindigkeit des Neuronalen Netzwerks hängt von der gewählten Architektur ab. In Tabelle 5.14 sind die Anzahl der Netzwerkparameter, als auch die Inferenz-Laufzeiten verschiedener Multi-Task-Netzwerke aus dieser Masterarbeit dargestellt. Es wird schnell ersichtlich, dass beide ResNet-Architekturen eine massiv höhere Parameterzahl haben als die BeyerModRelu-Architektur, jedoch vor allem die ResNet18-Architektur dennoch

Netzwerkarchitektur	**Laufzeit [ms/Batch]**	**Anzahl der Netzparameter**
BeyerModRelu [Lewandowski et al., 2019b]	19,7	785.548
ResNet18 [He et al., 2016]	36,0	11.441.220
ResNeXt50 [Xie et al., 2017]	111,6	24.034.116

Tabelle 5.14: Vergleich der drei wichtigsten Multi-Task-Netzwerkarchitekturen auf einer NVIDIA GTX1060
Angegeben sind für die drei Architekturen die Gewichtezahl im beschriebenen Multi-Task-Szenario. Die Inferenzzeit wurde pro Batch mit 20 Elementen gemessen. Dies entspricht der mittleren Kandidatenzahl des Kandidatengenerators. Die Versuche wurden auf einem Rechner mit NVIDIA GTX1060 Grafikkarte, Intel Core i5-6600 Prozessor und 32GB Arbeitsspeicher im Deep-Learning-Framework PyTorch in Version 1.1 mit NVIDIA cuDNN 7.6.2 durchgeführt.

performant auf einer NVIDIA GTX1060 einsetzbar ist. Die ResNeXt50-Architektur benötigt im Vergleich zur ResNet18-Architektur mehr als die dreifache Rechenzeit und bietet sich somit für die Anwendung auf einer Roboterplattform nicht an.
Bei einer mittleren Anzahl von Clustern pro Bild von 20 (vgl. hierzu auch Kapitel A.4) arbeitet ein auf dem ResNet18 basierendes System somit immer noch mit 25Hz, ($T_{detektion} = 4ms + 36ms = 40ms$). Ein ResNeXt50 erreicht lediglich 8,7Hz. Da die NVIDIA-Grafikeinheit auf den mobilen Plattformen des Fachgebiets NIKR eine geringere Rechenleistung als das hier verwendete Testsystem hat, ist leicht erkennbar, dass sich aufgrund der Rechenzeit nur die Netzwerke BeyerModRelu oder ResNet18 auf einer mobilen Plattform für den Praxiseinsatz eignen.

5.4.2 Vergleich der Systeme mit anderen Detektoren

Um die Qualität der Personendetektion des Gesamtsystems vergleichen zu können, wird der Testdatensatz SuPer [Lewandowski et al., 2019a] verwendet. Zur Bewertung wird nach dem Auswerteverfahren von [Dollar et al., 2012] vorgegangen. Anhand der Intersection Over Union (IoU) der 2D-Bounding-Boxen von Detektion und Ground Truth wird festgestellt, wie die Detektionen zu den Ground-Truth-Daten passen. Eine Detektion muss eine IoU von mindestens 50% zur Ground Truth besitzen, um als True Positive gezählt zu werden. Existieren mehrere Detektionen zu einer Ground Truth,

wird nur die mit der höchsten IoU als True Positive gezählt, die restlichen zählen als False Positives. In Abbildung 5.10 sind die Ergebnisse für alle Schwellwerte der Detektoren als DET-Kurven dargestellt. Hierbei werden die vollen 18 Meter, die die Kinect2-Tiefenkamera mit dem Libfreenect-Treiber[4] aufnehmen kann, für die Bewertung herangezogen.

Um einordnen zu können, wie gut die beste Single-Task-Baseline (ResNet18, Kapitel 5.2.2) und das beste Multi-Task-System (ResNet18, Kapitel 5.3.5) Personen detektieren können, sollen die folgenden Vergleichsverfahren verwendet werden:

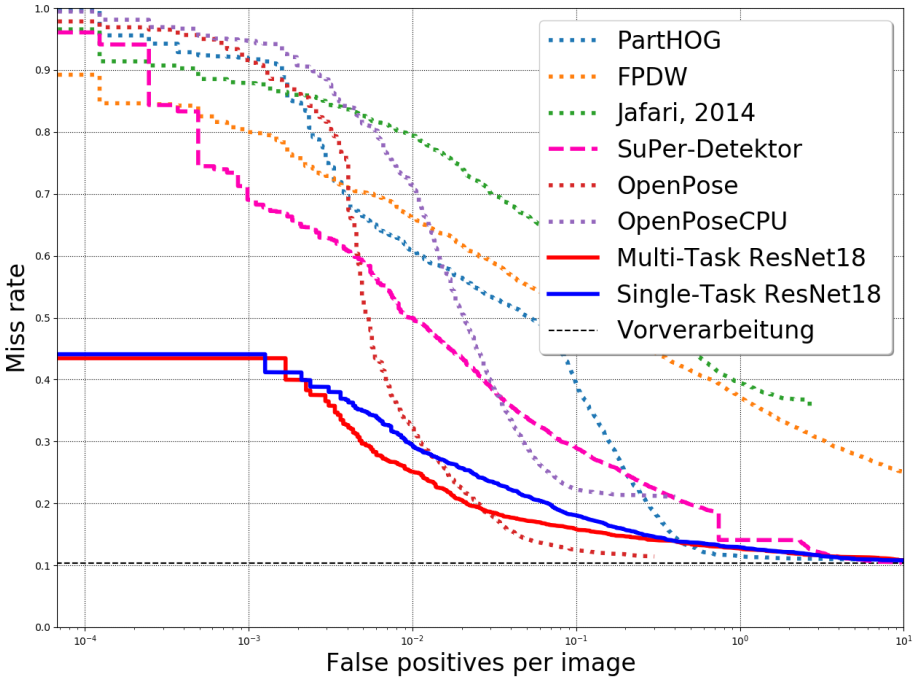


Abbildung 5.10: Detection-Error-Tradeoff-Kurven für den Testdatensatz SuPer, Auswertung bis 18m

Dargestellt sind die Detektionsergebnisse verschiedener klassischer, als auch Deep Learning basierter Systeme zur Personendetektion. Als gestrichtelte, horizontale Linie ist die minimale Miss Rate eingezeichnet, die sich aus der punktwolkenbasierten Vorverarbeitung, welche von den Detektoren SuPer, ResNet18-Single-Task und ResNet18-Multi-Task verwendet wird, ergibt. Eine geringere Miss Rate als diese kann ein nachfolgender Klassifikator nicht erreichen.

[4] `https://github.com/OpenKinect/libfreenect` - Abgerufen am 16.12.2019

- Farbbildbasiert, Klassisch: FPDW [Benenson et al., 2012]
- Farbbildbasiert, Klassisch: PartHOG [Felzenszwalb et al., 2010]
- Farbbildbasiert, Deep Learning: OpenPose(CPU) [Cao et al., 2017]
- Tiefenbildbasiert, Klassisch: Depth-Template [Jafari et al., 2014]
- Tiefenbildbasiert, Klassisch: SuPer-Detektor [Lewandowski et al., 2019a]

Hierbei wurde das Verfahren PartHOG auf den SuPer-Trainingsdaten nachtrainiert. Beim Verfahren FPDW zeigte sich keine Verbesserung, weshalb hier die unvortrainierten Ergebnisse angegeben sind. In beiden Auswertungen ist eindeutig erkennbar, dass sowohl die ResNet18-Single-Task-, als auch die ResNet18-Multi-Task-Architektur eine bessere Erkennungsleistung besitzen als die klassischen Verfahren, sowohl tiefenbildbasiert, als auch farbbildbasiert. Vor allem der Vergleich zum SuPer-Detektor zeigt, dass selbst ohne Multi-Task Learning aus den selben Trainingsdaten viel bessere Ergebnisse erreichbar sind als mit einer Support-Vektor-Maschine auf 3D-Punktwolken-Features. Außerdem wäre eine weitere Verbesserung der Detektionsleistung möglich, wenn der Kandidatengenerator ausgetauscht werden würde. Da dieser nicht alle Personen erkennt, kann selbst ein System, welches zu jedem Bildausschnitt das Label *Person* zuordnet, nur eine Miss-Rate von knapp 11% erreichen, vgl. horizontale, schwarze, gestrichelte Linie in Abbildung 5.10.

Im Vergleich zu aktuellen, farbbildbasierten Deep-Learning-Verfahren ist erkennbar, dass diese, vor allem durch die viel größere Anzahl an Trainingsbeispielen, bessere Erkennungsraten erreichen können, als der in dieser Masterarbeit entwickelte Ansatz. OpenPose nutzt zum Training unter anderem das Panoptic Dataset [Joo et al., 2017], welches aus 154 Millionen Bildern besteht. Dahingegen besteht der Trainingsdatensatz für das Multi-Task-System dieser Masterarbeit aus lediglich 15.309 Tiefenbildern (SuPer Datensatz) und 77.085 Tiefenbildausschnitten (Deep-Orientation Datensatz). Da große Teile dieser Daten am gleichen Tag aufgenommen wurden, ist außerdem die Varianz in den Daten (z.B. getragene Kleidung) geringer.

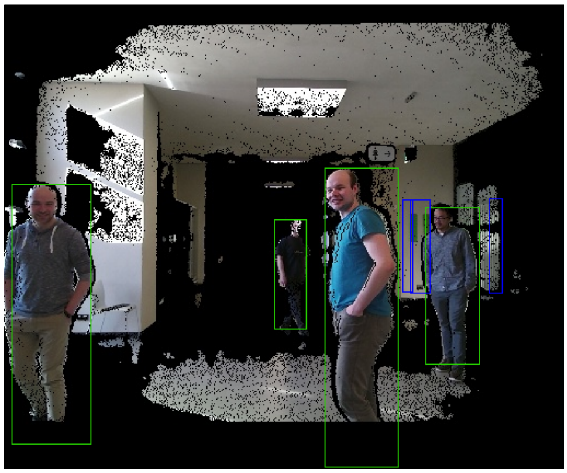
Dennoch ist in beiden Abbildungen gut erkennbar, welche Vorteile das Multi-Task Learning für die Personendetektion bringt. Die rote Multi-Task-Kurve zeigt im Bereich zwischen $2 \cdot 10^{-3}$ und $1 \cdot 10^{-1}$ Falsch-Positiven pro Bild eine deutliche Reduktion der Miss Rate im Vergleich zum Single-Task-System. Durch die Hinzunahme der Orientierungsschätzung als Task kann das Netzwerk die Klassen *Person* und *Nicht Person* somit besser trennen.

5.4.3 Einsatz des Systems auf den Robotern des Fachgebiets NIKR

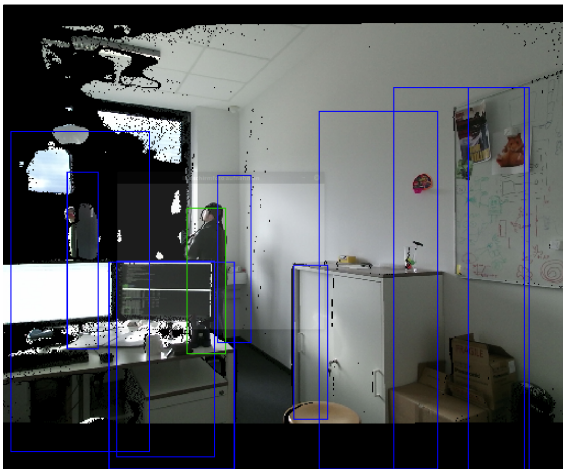
Nachdem in den vorangegangenen Abschnitten viel auf die Performance des Multi-Task-Systems in Form von quantitativen Maßzahlen eingegangen wurde, soll in diesem Kapitel kurz auf ein qualitatives Experiment am Fachgebiet NIKR eingegangen werden. Hierzu wurde das beste ResNet18-Single-Scale-Multi-Task-Netzwerk der Experimente aus Kapitel 5.3.5 ausgewählt und in die Roboterapplikation eingebunden. Hiermit kann das Gesamtsystem aus Kandidatengenerator und Multi-Task-Netzwerk in einer Realweltumgebung getestet werden. Im Folgenden werden zur Beurteilung der Detektionsergebnisse charakteristische, registrierte (über das Tiefenbild gelegte) Farbbilder dargestellt, in denen die Kandidaten der Vorverarbeitung als Boxen eingezeichnet sind. In den Abbildungen 5.11 und 5.14 sind in blau diejenigen Boxen eingefärbt, welche vom Multi-Task-System als *Nicht Person* klassifiziert wurden und wirklich keine Personen enthalten (True Negatives). Die Ausschnitte, welche als *Person* klassifiziert wurden, sind entweder grün (True Positives) oder rot (False Positives) dargestellt. Zur Beurteilung der Oberkörperorientierungsschätzung sind 3D-Punktewolken dargestellt, in denen die Orientierungsschätzung aller detektierten Personen durch einen Pfeil gekennzeichnet ist.

In Abbildung 5.11 sind beispielhaft Detektionsergebnisse aus einem offenen Gang (5.11a) und einer Büroumgebung (5.11b und 5.11c) dargestellt. Hier ist gut erkennbar, dass die Anzahl an Kandidaten pro Bild stark vom Inhalt der Szene abhängt. Während in offenen Gängen lediglich wenige Hintergrundkandidaten generiert werden, ist deren Anzahl in einer vollen Büroszene um ein Vielfaches höher. Dies führt wiederum zu einer längeren Rechenzeit des Gesamtverfahrens (siehe hierzu auch Formel 5.3 aus Abschnitt 5.4.1). Weiterhin ist anzumerken, dass das System sehr gut generalisiert. Die Negativbeispiele des Trainingsdatensatzes (Supermarktumgebung) unterscheiden sich stark von den Negativdaten einer Büroumgebung und außerdem enthalten die Trainingsdaten lediglich hockende, jedoch keine sitzenden Personen. Dennoch werden eben diese Objekte und Personen vom System richtig klassifiziert, vgl. Abbildung 5.11c. Sind sitzende Personen weiter entfernt, werden sie robust erkannt, befinden sie sich direkt vor dem Roboter, werden sie oftmals trotz guter Sichtbarkeit nicht erkannt.

In Abbildung 5.12 und 5.13 sind die Ergebnisse der Orientierungsschätzung dargestellt.



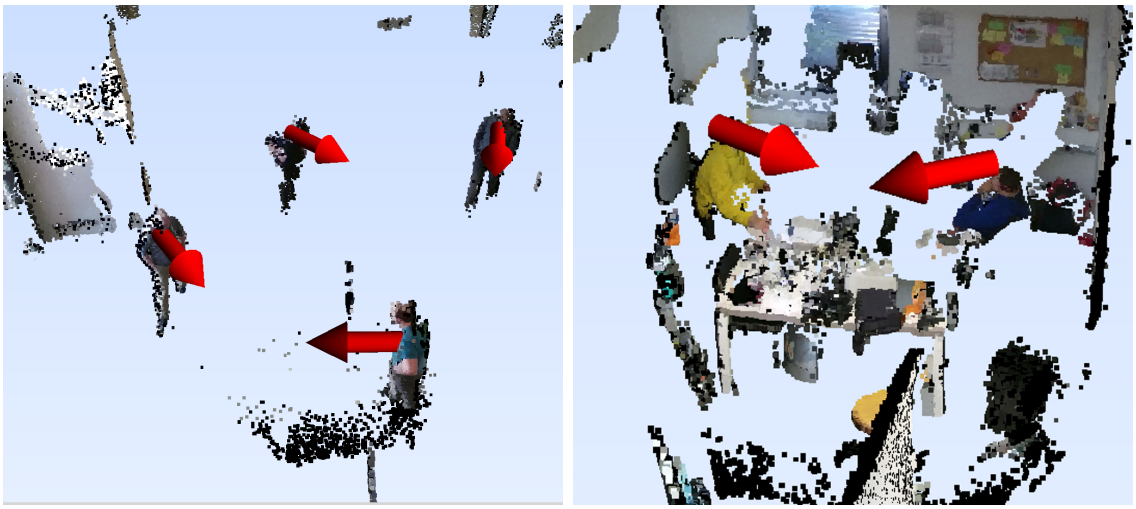
(a) Erkennung von mehreren Personen in einem offenen Gang



(b) Erkennung einer Person mit starken Verdeckungen in einer Büroszene

(c) Erkennung von sitzenden Personen in einer Büroszene

Abbildung 5.11: Detektionsergebnisse des Multi-Task-Systems auf Realweltdaten *In blau sind die generierten Kandidatencluster dargestellt, welche als Nicht Person klassifiziert wurden (True Negatives), in grün sind die Cluster dargestellt, die richtigerweise als Person klassifiziert wurden (True Positives). Erkennbar ist eine gute Erkennungsleistung. Das System zeigt zudem eine gute Generalisierungsfähigkeit, da auch sitzende Personen, welche nicht in den Trainingsdaten vorhanden sind, erkannt werden können.*



(a) Orientierungsschätzung von mehreren Personen in einem offenen Gang. Szene aus Abb. 5.11a.

(b) Orientierungsschätzung von sitzenden Personen in einer Büroszene. Szene aus Abb. 5.11c.

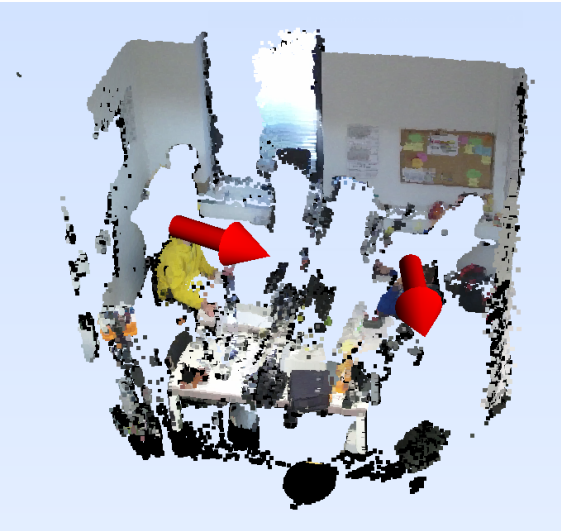
Abbildung 5.12: Ergebnisse der Orientierungsschätzung des Multi-Task-Systems auf Realweltdaten
Dargestellt sind gefärbte Punktwolken zu den Szenen aus Abbildung 5.11a und 5.11c. Jede Detektion des Systems wird anhand eines Pfeiles dargestellt, der in die Richtung der geschätzten Oberkörperorientierung zeigt.

Auch hier kann eine Generalisierungsfähigkeit auf sitzende Personen, wie sie in Abbildung 5.12b dargestellt sind, erkannt werden. Jedoch ist die Orientierungsschätzung bei sitzenden Personen deutlich weniger stabil, als dies bei der Detektion der Fall ist (vgl. Abbildung 5.13a). Weiterhin hängt die Orientierungsschätzung stark von der Sichtbarkeit des Kopfes der Personen ab. Ist dieser außerhalb des Bildes, da die Person sich zu nahe am Roboter befindet, ist das Netzwerk nicht mehr in der Lage, die Oberkörperorientierung zu schätzen. Dies ist in Abbildung 5.13b erkennbar. Dabei ist anzumerken, dass solche Fälle in den Trainingsdaten zur Orientierungsschätzung nicht vorkommen, da dort alle Personen vollständig im Bild zu sehen sind.
Abschließend sind noch einige Problemfälle im Bereich der Detektion in Abbildung 5.14 dargestellt.
Folgende Objekte werden teilweise als Personen erkannt (False Positives):

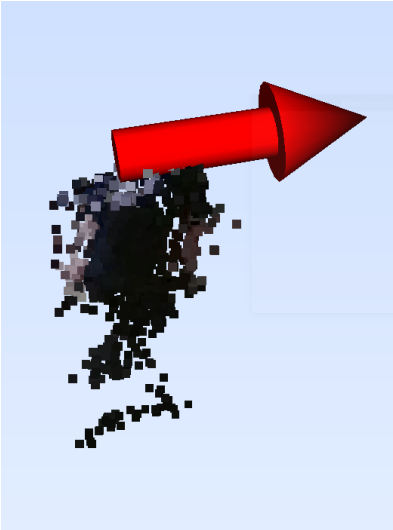
- Jacken, sowohl über Kleidungsständern oder Stühlen (5.14a)

- Mobile Assistenzroboter (5.14b)
- Unstrukturierte Objekte (5.14c)

Dies ist auf die fehlenden Trainingsdaten zurückzuführen. Gerade in einer Supermarktumgebung befinden sich wenige unstrukturierte Objekte oder Jackenständer, wie sie in Büroumgebungen oftmals vorhanden sind. Auch mobile Roboterplattformen wurden dem Netzwerk nie als Trainingsdaten präsentiert. Durch Integration von weiteren Trainingsdaten könnte die Netzwerkleistung daher nochmals erhöht werden.

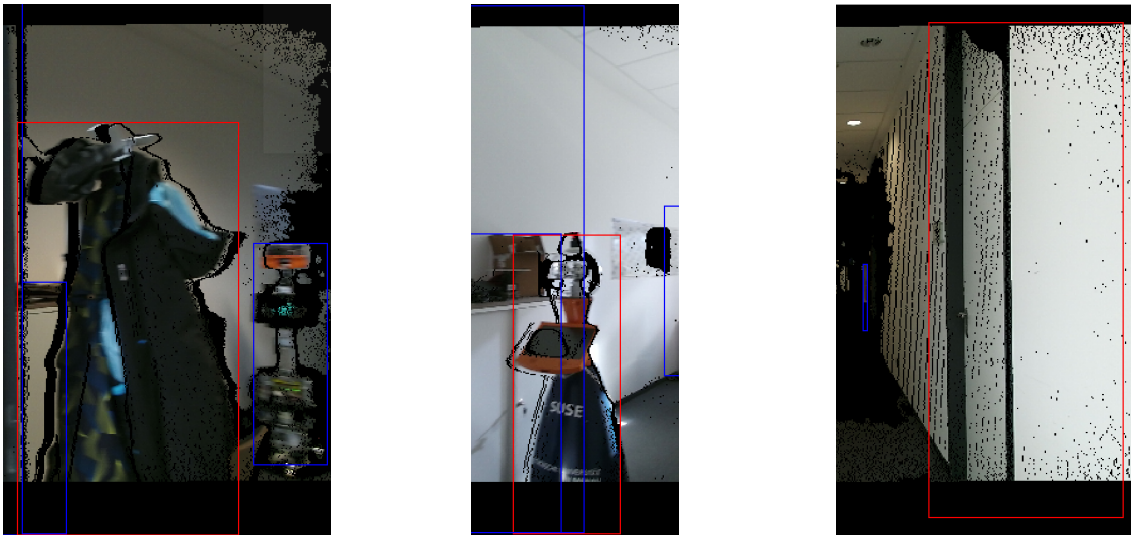


(a) Teilweise wird die Orientierung von sitzenden Personen falsch geschätzt. Szene aus Abb. 5.11c.



(b) Fehlerhafte Orientierungsschätzung, vermutlich, da in der Aufnahme kein Kopf sichtbar ist. Person aus Abb. 5.11a (mitte, im Hintergrund).

Abbildung 5.13: Ergebnisse der Orientierungsschätzung des Multi-Task-Systems auf Realweltdaten
Dargestellt sind gefärbte Punktwolken zu den Szenen aus Abbildung 5.11a und 5.11c. Jede Detektion des Systems wird anhand eines Pfeiles dargestellt, der in die Richtung der geschätzten Oberkörperorientierung zeigt.



(a) Jacken an Ständern **(b)** Roboter **(c)** Untexturierte Objekte

Abbildung 5.14: Typische Problemfälle der Personendetektion des Multi-Task-Systems
Gewisse Objekte werden vom Multi-Task-System stabil in die Personenklasse eingeordnet, obwohl es sich hierbei eindeutig um Negativdaten handelt. In blau sind die generierten Kandidatencluster dargestellt, welche als Nicht Person klassifiziert wurden (True Negatives), in rot sind die Cluster dargestellt, die vom Multi-Task-Netzwerk fälschlicherweise als Person klassifiziert wurden (False Positives).

Kapitel 6

Zusammenfassung und Ausblick

Nach den experimentellen Auswertungen des vorgestellten Multi-Task-Systems zur Personenwahrnehmung schließt diese Masterarbeit mit einem kurzen Überblick über die Vorgehensweisen und die gewonnenen Ergebnisse. Darauf folgt ein kurzer Ausblick über mögliche Erweiterungen, Veränderungen und Verbesserungen des behandelten Systems.

6.1 Zusammenfassung

Das Ziel dieser Masterarbeit war der Entwurf und die Umsetzung eines Systems, welches durch den Einsatz von Methoden des Deep-Multi-Task-Learning in Tiefenbildern Personen erkennt und gleichzeitig deren Oberkörperorientierung schätzt. In der breit gefächerten State-of-the-Art-Recherche wurden wichtige Trainingsparameter und Architekturen des Multi-Task Learning herausgearbeitet. Auf dieser Grundlage wurde eine Verarbeitungspipeline entworfen, welche auf ein Deep-Learning-Netzwerk zurückgreift und unter Verwendung des Hard-Parameter-Sharing-Paradigmas beide Aufgaben zusammen behandelt. Verschiedene Netzwerkarchitekturen wurden anhand von drei bestehenden und einem, im Rahmen dieser Masterarbeit neu aufgenommenem, Datensatz trainiert und quantitativ evaluiert. Dabei konnte eine Architektur gefunden werden, welche eine höhere Erkennungsleistung als entsprechende, einzelne Single-Task-Systeme aufweist. Auf dem SuPer Datensatz kann die Erkennungsrate im Bereich zwischen $1 \cdot 10^{-2}$ und $1 \cdot 10^{-1}$ Falsch-Positiven pro Bild um drei bis sechs Prozentpunkte im Vergleich zu einer Single-Task-Herangehensweise verbessert werden.

Zudem kann die Orientierungsschätzung mit ungefähr der selben Genauigkeit wie beim Vergleichsverfahren Deep-Orientation durchgeführt werden. Die Rechenzeit liegt dabei für beide Tasks zusammen in der Größenordnung der Single-Task-Netzwerke und nimmt mit jedem zusätzlichen Task nur um wenige Prozentpunkte zu. Durch die Wahl von leichtgewichtigen Netzwerkarchitekturen und der Integration in das Roboter-Framework MIRA konnte das Gesamtsystem mit Erfolg qualitativ in einer Realweltanwendung getestet werden. Wird das Netzwerk auf einer NVIDIA Jetson-Plattform eingesetzt, läuft das System echtzeitfähig und erreicht damit eine deutliche Verbesserung der Personenwahrnehmung der Roboter des Fachgebiets NIKR.

6.2 Ausblick

Multi-Task Learning ist ein sehr großer Themenkomplex im Bereich des Machine Learning. Es gibt unzählige verschiedene Varianten, wie ein solches System aufgebaut und trainiert werden kann. Diese sind meistens relativ einfach implementierbar, jedoch benötigen die Trainings durch die höhere Anzahl an Hyperparametern, welche oftmals nicht empirisch festgelegt werden können und somit systematisch analysiert werden müssen, eine viel größere Rechenzeit, als ein vergleichbares Single-Task System. Im Rahmen einer Masterarbeit ist es aus Zeitgründen nicht möglich, auf alle Varianten einzugehen. Deshalb wird in diesem Kapitel eine kurze Übersicht über vielversprechende Verbesserungsmöglichkeiten des vorgestellten Systems gegeben.

6.2.1 Hinzufügen weiterer Tasks

Um einen Einblick zu erhalten, inwiefern sich das Multi-Task-System um weitere Tasks erweitern lässt, bietet es sich an, das auf dem ResNet18-Single-Scale basierende Multi-Task-Netzwerk um einen Klassifikationskopf für die Haltung der Person zu erweitern. Mithilfe der vorhandenen Datensätze wäre eine Unterscheidung der Klassen *stehend* und *hockend* möglich. Werden weitere Daten hinzugezogen, wären auch weitere Haltungen, wie *sitzende Personen* oder andere Personenattribute, wie bei [Zhang et al., 2014], denkbar. So könnte durch eine Erweiterung des Gesamtsystems um *Neben- bzw. Auxillary-Tasks* (vgl. Kapitel 3.6) zudem eine verbesserte Netzwerkleistung für die bereits bestehenden Tasks erreicht werden.

6.2.2 Adaptive Kombination von Loss-Funktionen

Während in dieser Masterarbeit der für das Training verwendete Gesamt-Loss in den meisten Experimenten als gewichtete Summe der taskspezifischen Loss-Funktionen oder versuchsweise durch die Verfahren *Uncertainty Weighting* und *GradNorm* gebildet wurde (vgl. Anhang A.2) existieren in der Literatur natürlich noch weitere adaptive Verfahren zur Bestimmung der Loss-Gewichtung. Das Verfahren der dynamischen Taskpriorisierung von [Guo et al., 2018] könnte zum Beispiel bessere Ergebnisse liefern, als sie in dieser Masterarbeit mit den beschrieben Ansätzen erreicht wurden.

6.2.3 Zusammenstellung der Samples zu Batches

Ein Problem, welches im Rahmen dieser Masterarbeit aufgefallen ist, ist die Zusammenstellung der Trainingsdaten in Batches. Während hier zwar getestet wurde, wie die Datensätze am besten kombiniert werden können, bleibt das Problem bestehen, dass trotz einer Aufteilung nach gleichen Teilen die Anzahl an Trainingslabels pro Batch nicht gleich ist, da z.B. jedes Sample aus dem Deep-Orientation Datensatz sowohl Informationen über Orientierung, als auch ein Klassenlabel mitbringt, die anderen Datensätze jedoch keine Orientierung enthalten. Somit sind die Batches zwar von den Datensätzen her ausbalanciert aufgeteilt, pro Epoche sehen die einzelnen Tasks jedoch trotzdem eine unterschiedliche Anzahl an Labels.
Abhilfe könnte hier ein sehr aktuelles Verfahren von [Jiang et al., 2019] bringen: Anstatt dem Netzwerk alle Daten pro Epoche zu präsentieren, wird hier zuerst ein Vorwärtsdurchlauf durch das Netzwerk berechnet. Für das Training werden dann Batches aus Elementen zusammengestellt, die dem Netzwerk sehr schwer fallen. So ließe sich eine Trainingspipeline implementieren, die einerseits schnellere Trainingszeiten mit sich bringt und andererseits Batches so zusammenstellt, dass ein ausgewogenes Training ermöglicht wird.

6.2.4 Austausch der Vorverarbeitung

Während die anderen Punkte eher bei einer Verbesserung des Multi-Task-Systems angreifen, ist auch eine Verbesserung der Vorverarbeitung denkbar, um die Performance des Gesamtsystems zu erhöhen. Mit den Ergebnissen aus Kapitel A.4 ist klar, dass

selbst ein perfektes Multi-Task-System lediglich 89% der Personen erkennen kann, da die Vorverarbeitung die restlichen 11% nicht als Kandidaten bereitstellt und diese somit direkt verworfen werden. Dies äußert sich dann in den DET-Kurven aus Kapitel 5.4.2 in einer unteren Beschränkung der Miss-Rate auf 11%. Um die Detektionsleistung von aktuellen Deep-Learning-Verfahren, wie z.B. OpenPose [Cao et al., 2017], zu erreichen, muss daher eine bessere Vorverarbeitung eingesetzt werden.
Neue, CNN-basierte Ansätze zur Erzeugung von Kandidaten (Region Proposals), wie das Faster-RCNN von [Ren et al., 2015], könnten hierbei die gewünschte Verbesserung der Gesamtperformance ermöglichen. Jedoch haben diese Verfahren oftmals eine höhere Rechenzeit, welche den echtzeitfähigen Einsatz auf einer mobilen Plattform behindert. Mit einem schnellen, echtzeitfähigen Verfahren könnte jedoch die Gesamtleistung nochmals deutlich verbessert werden.

6.2.5 Änderungen an der Netzwerkstruktur

Abschließend stellt sich die Frage, ob die selben Ergebnisse erreicht werden könnten, wenn eine weniger rechenaufwendige Netzwerkstruktur eingesetzt werden würde. Diese könnte sich von der Komplexität zwischen der BeyerModRelu- und der ResNet18-Architektur einordnen. Um den Durchsatz des Systems zu erhöhen, könnte beispielsweise die Anzahl an Feature Maps bzw. Filtern des ResNet18 verringert werden, oder auf ein ResNet10 gesetzt werden. Außerdem könnte auch eine der *EfficientNet-Architekturen* von [Tan und Le, 2019] eingesetzt werden. Hier erreicht die schnellste vorgestellte Architektur (EfficientNet-B0) im Vergleich zum ResNet18 trotz einer Halbierung der Parameterzahl eine Steigerung der ImageNet-Top-1-Klassifikationsleistung von 69,8% auf 77,3% [He et al., 2016] [Tan und Le, 2019]. Inwiefern dies jedoch die Netzwerkperformance in dem gegebenen Anwendungsfall der Personenwahrnehmung in Tiefenbildern beeinflusst, muss zusätzlich evaluiert werden.

Anhang A

Ergänzende Unterlagen

A.1 Vorversuche

Im Rahmen einer Reihe von Vorversuchen wurde evaluiert, inwiefern sich verschiedene Netzwerkstrukturen für die echtzeitfähige Klassifikation von Personenbildausschnitten eignen. Hierbei wurden die Architekturen ResNet18 (vortrainiert auf ImageNet), ResNeXt50 (vortrainiert auf ImageNet) und BeyerModRelu verglichen. Alle Netzwerke wurden ohne Augmentierung trainiert. Zur Auswertung wurden für jede Konfiguration drei Trainings durchgeführt. Anhand der Balanced Error Rate auf den Validierungsdaten wurde dann die jeweils beste Epoche ausgewählt, welche auf allen vier Testdatensätzen ausgewertet wird. Die Balanced Error Rate stellte sich hier als gutes Maß heraus, da hierbei konstant bessere Netzwerke ausgewählt wurden, als bei Verwendung des Netzwerks mit dem größten **F1-Score auf** den Validierungsdaten.
Es wurden folgende Parameterkombinationen gegeneinander verglichen:

- Vorverarbeitung der Daten: Normalisierung der Tiefenwerte (Anhand des Trainingsdatensatzes), Skalierung der Tiefenwerte (Anhand des maximalen Sensorwertes)
- Training nur mit Datensatz SuPer, Training mit Datensatz SuPer und SRL
- Vergleich von austrainierten Netzwerken nach 400 Epochen mit Netzwerken nach 15 Epochen
- Vergleich von Lernraten zwischen $1 \cdot 10^{-5}$ und $1 \cdot 10^{-1}$

Alle numerischen Ergebnisse sind in Kapitel A.5.11 angetragen.

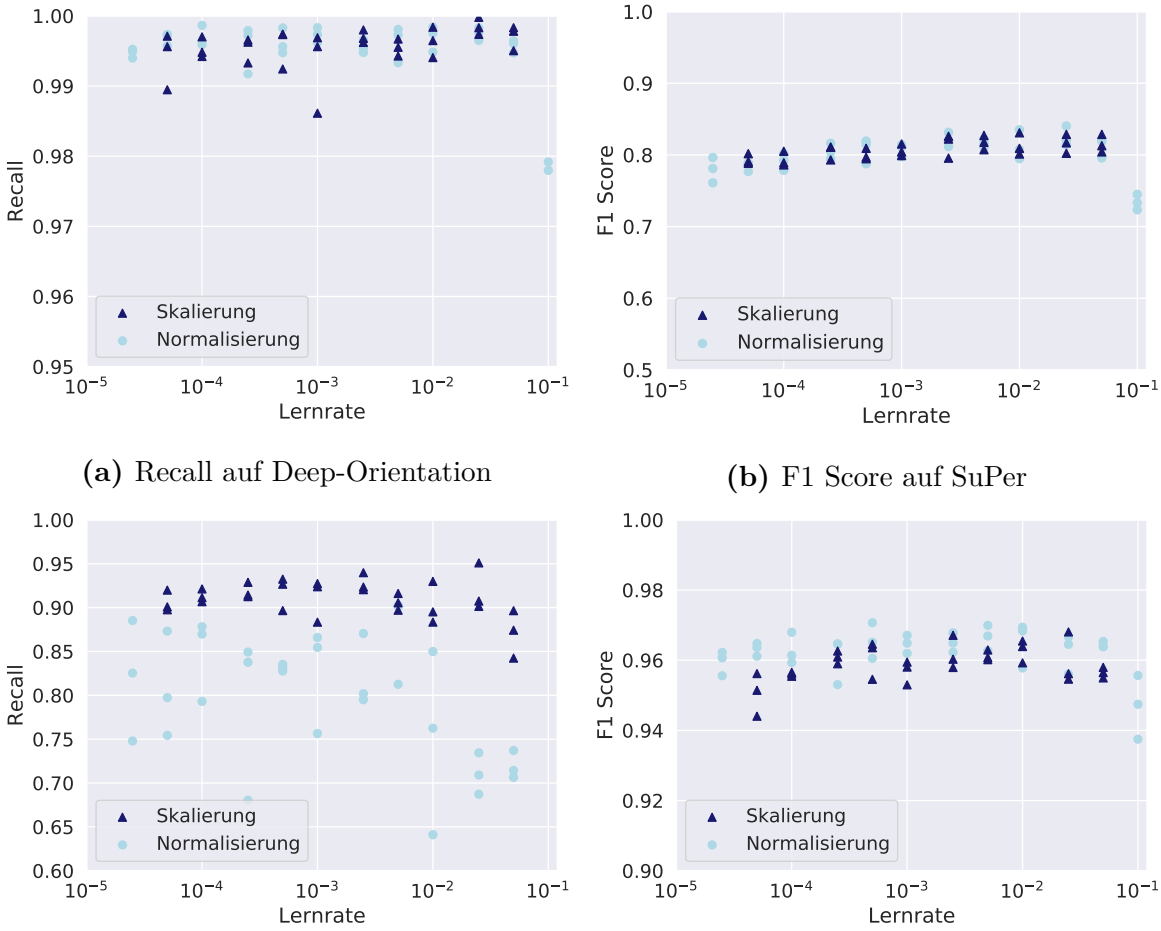
A.1.1 Vorverarbeitung der Daten

Tiefendaten haben einen anderen Wertebereich als RGB-Daten. Während 8-Bit-RGB-Daten jeweils drei Werte pro Pixel mit einem Wertebereich zwischen 0 und 255 speichern, enthalten Tiefenbilder pro Pixel einen Tiefenwert zwischen minimaler Distanz und maximaler Distanz, wobei ein Wert (oftmals null) als ungültiger Messwert interpretiert wird. Somit bestehen grundsätzlich zwei Möglichkeiten, die Daten für ein Netzwerk zu verarbeiten:
Es können einfach alle Tiefenwerte durch den maximal möglichen Tiefenwert geteilt werden, sodass der neue Wertebereich zwischen 0,0 und 1,0 liegt. Hierbei bleibt die Null ohne weitere Behandlung der Wert für die ungültigen Tiefenwerte.
Die zweite Möglichkeit besteht in der Normalisierung der Tiefenwerte. Hierbei wird Mittelwert und Standardabweichung aus allen gültigen Pixeln (Tiefenwert größer null) des Trainingsdatensatzes bestimmt und für die Normalisierung aller weiteren Daten verwendet. Wichtig ist hierbei, dass ungültige Tiefenwerte durch das Abziehen des Mittelwertes nicht mehr bei Null liegen. Diese müssen daher anhand von einer Maske nach der Normalisierung erneut auf Null gezogen werden.

Die Ergebnisse für das ResNet18 sind in Abbildung A.1 dargestellt. Beim Orientierungs- und SuPer-Datensatz (Abbildung A.1a und A.1b) zeigte sich im Rahmen der Vorexperimente kein Unterschied zwischen Skalierung und Normalisierung. Auf dem TUI-Zuse Datensatz (Abbildung A.1d) ist die Performance mit Normalisierung leicht besser. Auf dem SRL-Datensatz (Abbildung A.1c) ist die Performance des Skalierungsnetzwerkes jedoch bedeutend besser. Es ist daher davon auszugehen, dass das Netzwerk mit der skalierenden Vorverarbeitung besser auf unbekannten Daten bzw. Datensätzen generalisiert. Daher soll in den Experimenten im Hauptteil dieser Masterarbeit die Skalierung als Vorverarbeitung verwendet werden.

A.1.2 Hinzunahme von weiteren Datensätzen zum Training

Durch die Hinzunahme des SRL-Datensatzes zum Training wird die Anzahl an Personenbeispielen erhöht. Hierdurch ist eine Verbesserung des Recalls des Systems zu erwarten. In Abbildung A.2 ist jedoch keine eindeutige Tendenz zu erkennen: Auf dem TUI-Zuse Datensatz sinkt der F1-Score leicht ab, auf dem SuPer Datensatz steigt der



(a) Recall auf Deep-Orientation

(b) F1 Score auf SuPer

(c) Recall auf SRL

(d) F1 Score auf TUI-Zuse

Abbildung A.1: Einfluss von Lernrate und Vorverarbeitung auf die Netzwerkleistung des Single-Task ResNet18

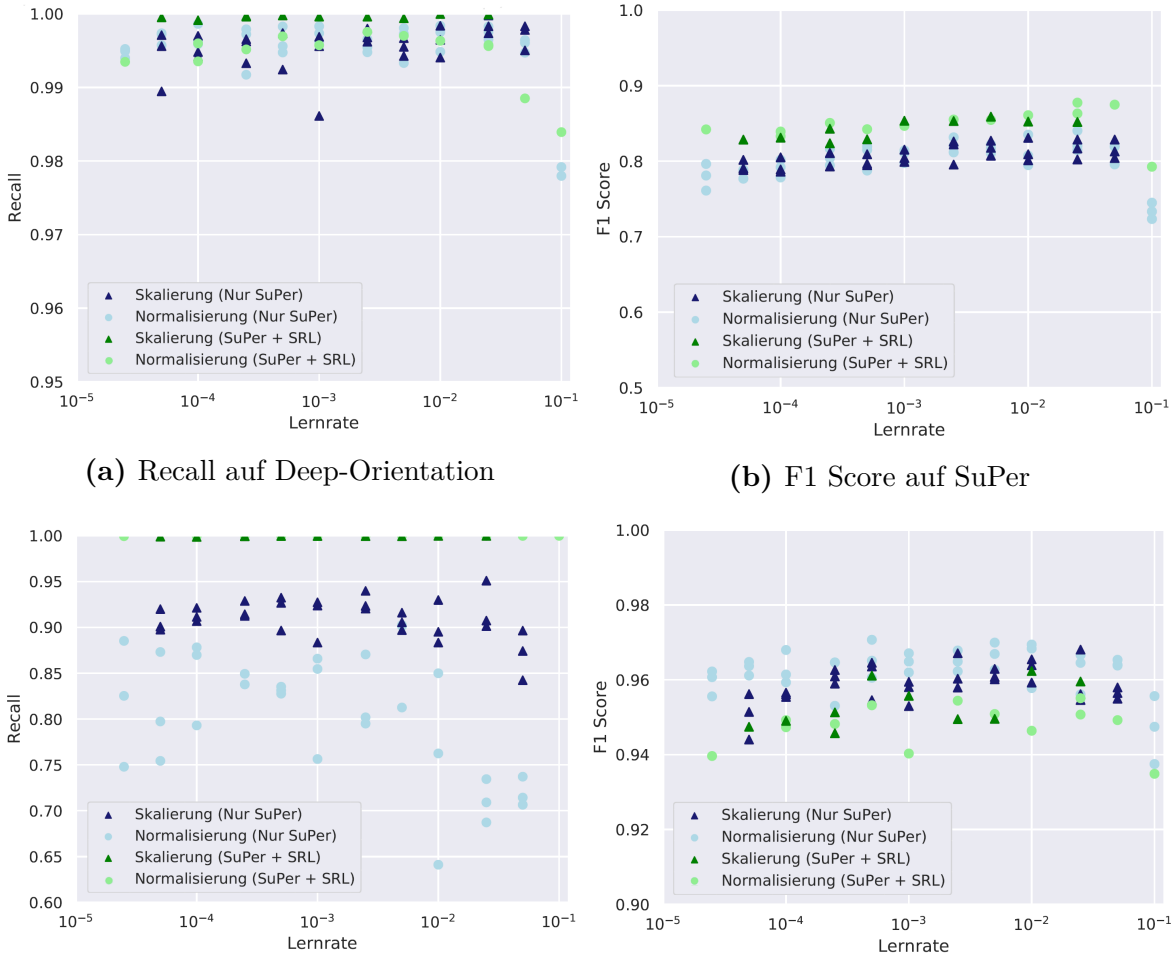
Das auf dem SuPer-Datensatz trainierte ResNet18 zeigt für die beiden Vorverarbeitungen unterschiedliche Ergebnisse, abhängig vom Datensatz. Vor allem in Abbildung A.1c ist erkennbar, dass die Skalierung besser generalisierende Netzwerke erzeugt und zudem die Streuung der Netzwerke geringer ausfällt. Die Dreiecke entsprechen Netzwerken mit Skalierung als Vorverarbeitung, die Kreise entsprechen Netzwerken mit Normalisierung als Vorverarbeitung.

F1-Score merklich an und auf dem Deep-Orientation Datensatz ist die Klassifikationsrate konstant. Interessant ist hierbei, dass auf dem SuPer Datensatz der Recall gleich bleibt, der steigende F1-Score also allein auf eine verbesserte Precision und damit eine geringere Zahl von Falsch-Positiven zurückzuführen ist. Weiterhin ist erkennbar, dass auf dem SRL-Datensatz keine Aussagen mehr über die Netzwerkqualität getroffen werden können. Durch die fehlenden Negativdaten lernt das Netzwerk lediglich, anhand welcher Merkmale der SRL-Daten vom SuPer Datensatz abgetrennt werden kann (z.B. unterschiedliche Treiber) und, dass ein Bildausschnitt aus dem SRL-Datensatz immer eine Person ist. Dies führt dann zu einer fehlerlosen Klassifikation von allen Patches aus dem SRL-Datensatz, vgl. Abbildung A.2c.

Hierdurch ergeben sich zwei wichtige Schlussfolgerungen: Einerseits führt eine Hinzunahme von Trainingsdaten nicht unbedingt zu einer verbesserten Performance auf allen Datensätzen und andererseits kann durch Hinzunahme von einem Datensatz ohne Negativdaten danach keine Aussage mehr auf dem jeweiligen Datensatz getroffen werden, da dem Netzwerk die Unterscheidung der Datensätze zu leicht fällt. Deshalb bietet es sich an, den SRL-Datensatz nicht zum Training zu verwenden, um diesen als großen, unabhängigen Testdatensatz verwenden zu können. Im Hauptteil dieser Arbeit wird deshalb die Kombination aus Deep-Orientation Datensatz (aufgrund der Orientierungsschätzung) und SuPer Datensatz (aufgrund der Personendetektion) zum Training eingesetzt.

A.1.3 Ergebnisse nach geringerer Epochenzahl

Da die ResNets mit normalisierten Daten vortrainiert sind, stellt sich die Frage, ob die Netzwerke Probleme mit dem Umlernen von normalisierten zu skalierten Daten haben könnten. Daher soll untersucht werden, ob der Unterschied zwischen Normalisierung und Skalierung nach einer kürzeren Trainingszeit (15 Epochen) größer ist, als nach einer längeren Trainingszeit (400 Epochen). In Abbildung A.3 ist erkennbar, dass bei Lernraten unterhalb $1 \cdot 10^{-3}$ die Netzwerkperformance im Vergleich zu den Tests mit 400 Epochen (Abbildung A.1) stark abfällt, jedoch ohne Unterschied zwischen Normalisierung und Skalierung. Darüber ist jedoch fast kein Unterschied zwischen den austrainierten und den früher gestoppten Trainings erkennbar. Einzig im Orientierungsdatensatz besteht hier ein Unterschied: Die Skalierungsnetzwerke sind hier bei niedrigen



(a) Recall auf Deep-Orientation

(b) F1 Score auf SuPer

(c) Recall auf SRL

(d) F1 Score auf TUI-Zuse

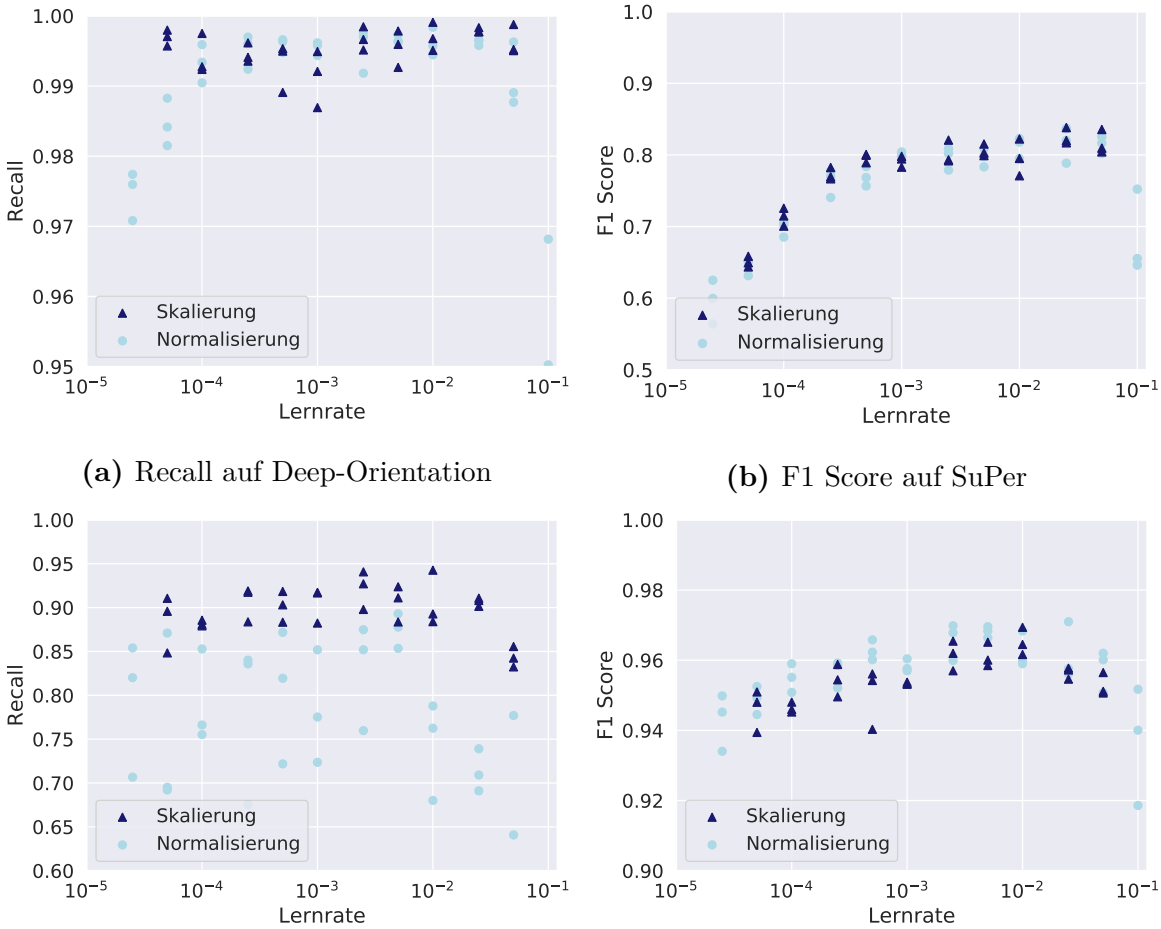
Abbildung A.2: Einfluss von Lernrate, Vorverarbeitung und Datensätzen auf die Netzwerkleistung des Single-Task-ResNet18

In blau sind Netzwerke dargestellt, welche ausschließlich mit dem SuPer-Datensatz trainiert wurden, in grün Netzwerke, welche mit SuPer- und SRL-Datensatz trainiert wurden. Die Dreiecke entsprechen Netzwerken mit Skalierung als Vorverarbeitung, die Kreise entsprechen Netzwerken mit Normalisierung als Vorverarbeitung.

Lernraten besser, obwohl diese auf normalisierten Daten vortrainiert wurden. Somit stellt es kein Problem dar, die vortrainierten Netzwerke mit skalierten Tiefendaten zu verwenden.

A.1.4 Vergleich der Verfahren

Sowohl für das BeyerModRelu als auch für ResNet18 und ResNeXt50 wurden verschiedene Lernraten getestet. Die Ergebnisse sind in Abbildung A.4 dargestellt. Hier ist erkennbar, dass bei beiden ResNet-Architekturen ein breiter Bereich der Lernrate eingesetzt werden kann, ohne die Performance des Netzwerks zu beeinflussen. Beim BeyerModRelu Netzwerk ist der Bereich etwas kleiner und er unterscheidet sich von Datensatz zu Datensatz. Insgesamt ist erkennbar, dass das ResNeXt50 die leistungsfähigste Architektur ist. Darauf folgt das ResNet18. Das BeyerModRelu-Netzwerk ist die leistungsschwächste Architektur aus diesem Vergleich.



(a) Recall auf Deep-Orientation

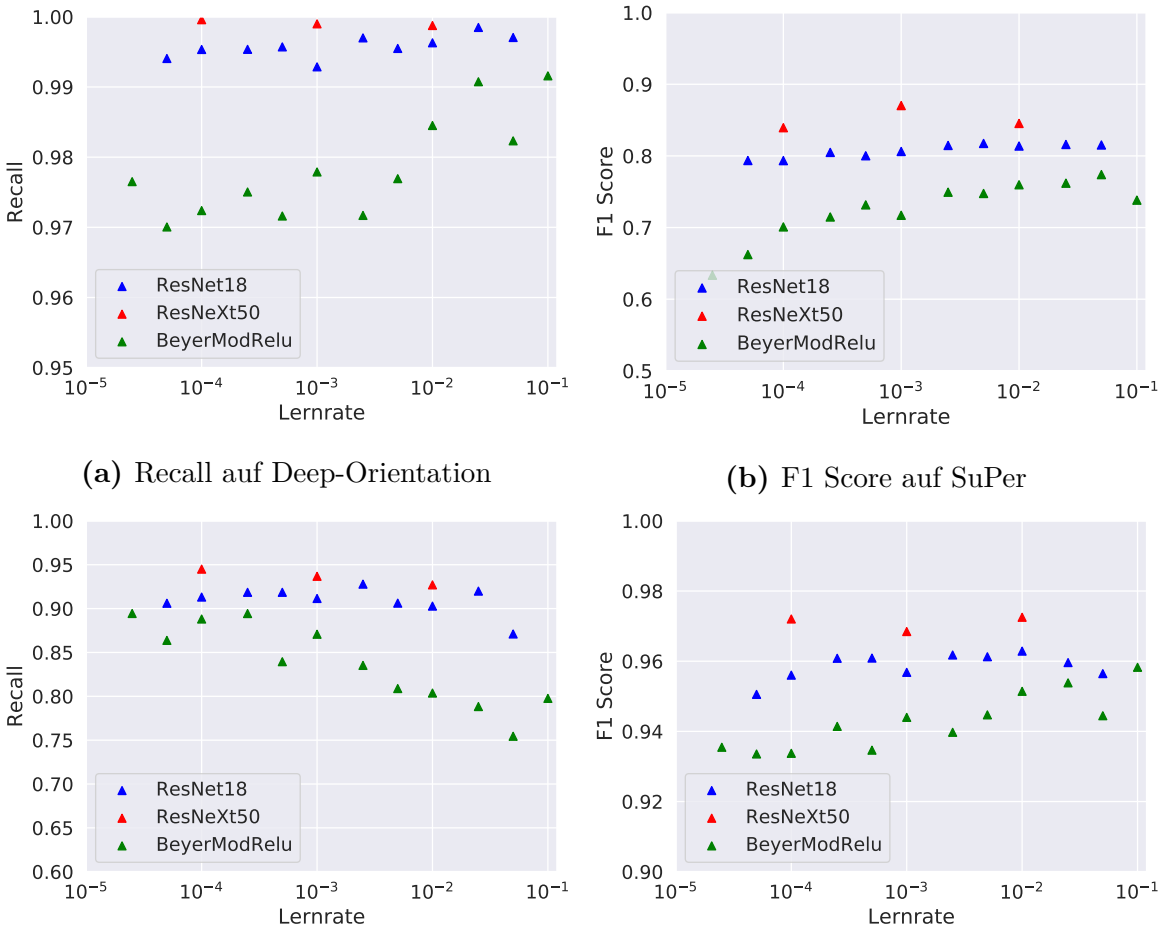
(b) F1 Score auf SuPer

(c) Recall auf SRL

(d) F1 Score auf TUI-Zuse

Abbildung A.3: Einfluss von Lernrate und Vorverarbeitung auf die Performance des ResNet18

Im Gegenzug zur vorherigen Experimentalreihe wurden die Netzwerke hier lediglich 15 Epochen lang trainiert. Die Dreiecke entsprechen Netzwerken mit Skalierung als Vorverarbeitung, die Kreise entsprechen Netzwerken mit Normalisierung als Vorverarbeitung.



(a) Recall auf Deep-Orientation

(b) F1 Score auf SuPer

(c) Recall auf SRL

(d) F1 Score auf TUI-Zuse

Abbildung A.4: Einfluss von Lernrate und Verfahren auf die mittlere Netzwerkperformance auf den Testdaten

Alle Verfahren wurden 400 Epochen auf dem SuPer Datensatz trainiert, als Vorverarbeitung wurde die Skalierung der Tiefenwerte eingesetzt.

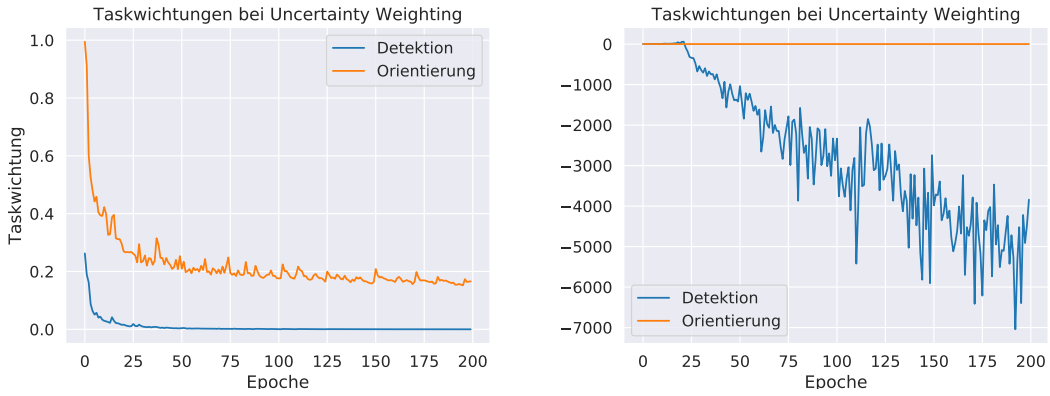
A.2 Einsatz adaptiver Multi-Task Lernverfahren

Neben der Evaluierung verschiedener Architekturen zur kombinierten Personenwahrnehmung im Rahmen dieser Masterarbeit, stellt sich die Frage, ob die rechenaufwendige Suche nach der optimalen Taskgewichtung mit den Methoden des *Self Paced Multi-Task Learning* anders angegangen werden kann. Die Verfahren Uncertainty Weighting (Kapitel 3.4) und GradNorm (Kapitel 3.4) sollen zu einer automatischen und optimalen Wahl der Taskwichtungsparameter während des Trainings führen. Eine solche Vorgehensweise könnte den Trainingsablauf stark vereinfachen. Jedoch zeigen beide Verfahren in den folgenden Experimenten in Verbindung mit der besten Multi-Task-Architektur, dem ResNet18-Single-Scale, keine nutzbaren Ergebnisse.

A.2.1 Uncertainty Weighting

[Lu et al., 2017] versuchen anhand der Unsicherheiten der einzelnen Tasks eine automatische Gewichtung zu schätzen. Hierzu werden die Wichtungsparameter der Loss-Funktionen während des Trainings optimiert. Eine PyTorch-Implementierung wurde aus Github entnommen [1]. Dieses Verfahren zeigte einen stabilen Verlauf der Taskwichtungen, vgl. Abbildung A.5a. Es ist zu erkennen, dass dabei die Orientierungsschätzung höher gewichtet wird, als die Personenklassifikation. Ein solches Verhalten zeigte sich auch bei den vorangegangenen Experimenten mit konstanten Wichtungen (Abschnitt 5.3.3). Dennoch sind die Ergebnisse der Netzwerke viel schlechter als bei den bisherigen Experimenten. Der Winkelfehler aus Abbildung A.6a beträgt auf den Deep-Orientation-Validierungsdaten zum Ende des Trainings 24°, wohingegen die besten Netzwerke mit konstanten Wichtungen hier Werte zwischen 7° und 8° erreichen. Auch der Klassifikationsfehler (Balanced Error Rate, Abbildung A.6b) auf den Validierungsdaten SuPer und Deep-Orientation ist doppelt so groß (0,0015 bei Uncertainty Weighting, 0,0007 bei konstanter Wichtung von 0,5). Damit ermöglicht das Verfahren zwar eine Anpassung der Wichtungsfaktoren, jedoch erreichen die Netzwerke trotzdem eine schlechtere Gesamtleistung. Dies könnte auf den kleineren Betrag der Taskwichtung (z.B. 0,2 für die Orientierungsschätzung, Personenklassifikation nahe 0,0) und der damit effektiv geringeren Lernrate zurückzuführen sein.

[1] `https://github.com/ranandalon/mtl` - Abgerufen am 16.12.2019



(a) Uncertainty Weighting **(b)** GradNorm

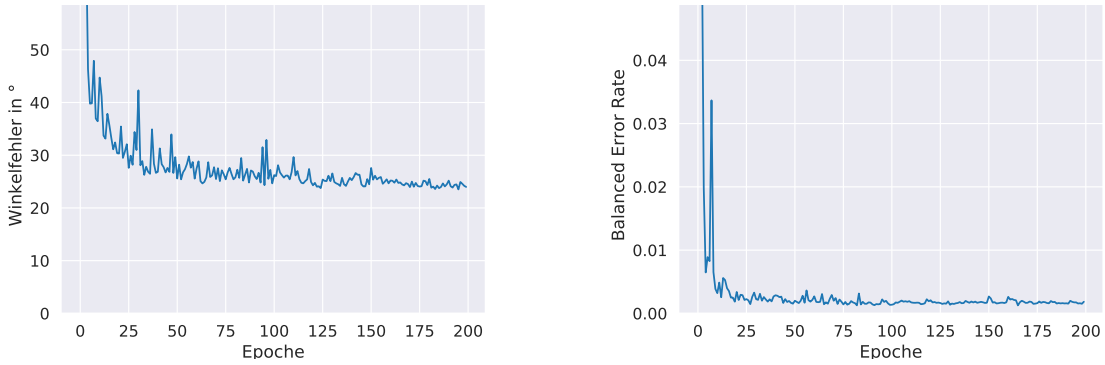
Abbildung A.5: Wichtungsfaktoren der Tasks für Uncertainty Weighting und GradNorm

Erkennbar ist bei Uncertainty Weighting [Lu et al., 2017] (A.5a) die erfolgreiche Anpassung der Gewichte anhand der Taskunsicherheiten. Wie in den vorangegangenen Experimenten erhält hier die Orientierungsschätzung ein höheres Gewicht als die Personenklassifikation. In Abbildung A.5b ist der Wichtungsverlauf bei Einsatz von GradNorm [Chen et al., 2017] zu sehen. Zuerst steigt die Wichtung von der Klassifikation über die Wichtung der Orientierungsschätzung. Daraufhin divergiert das Training.

A.2.2 GradNorm

Die Implementierung des Verfahrens GradNorm wurde ebenfalls aus Github entnommen[2]. Leider zeigt das Verfahren im beschriebenen Anwendungsszenario kein stabiles Training. In Abbildung A.5b ist dargestellt, wie sich die Gewichtungen während des Trainings adaptieren. Zuerst steigt die Gewichtung des Klassifikationstasks im Vergleich zur Orientierungsschätzung an, daraufhin divergiert die Klassifikationswichtung in den negativen Bereich. Damit ist das Verfahren so nicht einsetzbar. Dieser Effekt wurde für verschiedene GradNorm-Hyperparameter festgestellt. Somit ist GradNorm in diesem Anwendungsfall nicht einsetzbar.

[2] `https://github.com/hosseinshn/GradNorm` - Abgerufen am 16.12.2019



(a) Winkelfehler

(b) Klassifikationsfehler

Abbildung A.6: Netzwerkleistung auf den Validierungsdaten bei Verwendung von Uncertainty Weighting
Dargestellt ist über die 200 Trainingsepochen der Verlauf von Winkel- und Klassifikationsfehler auf den Validierungsdaten. Erkennbar ist, dass beide Fehler auf einen Wert konvergieren, der jedoch deutlich höher liegt, als bei der Verwendung von konstanten Wichtungen.

A.3 Multi-Scale-Architektur zum BeyerModRelu-Netzwerk

Im Rahmen der Experimente wurde auch ein Multi-Scale-Ansatz zur BeyerModRelu-Architektur entworfen und evaluiert. Gerade aufgrund der geringeren Klassifikationsleistung des BeyerModRelu-Netzwerkes bzw. der BeyerModRelu-Multi-Task-Architektur, ist zu erwarten, dass eine Multi-Scale-Herangehensweise bessere Ergebnisse liefern kann.

A.3.1 Architektur

Hierbei bietet sich die Netzwerkstruktur wie beim ResNet18-Multi-Scale-LAP an, da im BeyerModRelu-Netzwerk auch lokales Pooling verwendet wird. Es werden nach den drei Gruppen jeweils an der Ausgabe der Max-Pooling-Schichten die Feature Maps abgegriffen und mittels einer Faltung auf eine nutzbare Größe für die taskspezifischen Schichten gebracht. Durch diese Filterwahl kommen aus der letzten Gruppe 960 Features, aus den Gruppen eins und zwei 448 bzw. 480. Dies führt zu einem 1888 Elemente langen Feature-Vektor für die Tasks. Die BeyerModRelu-Multi-Scale-Architektur ist in

Abbildung A.7 visualisiert.

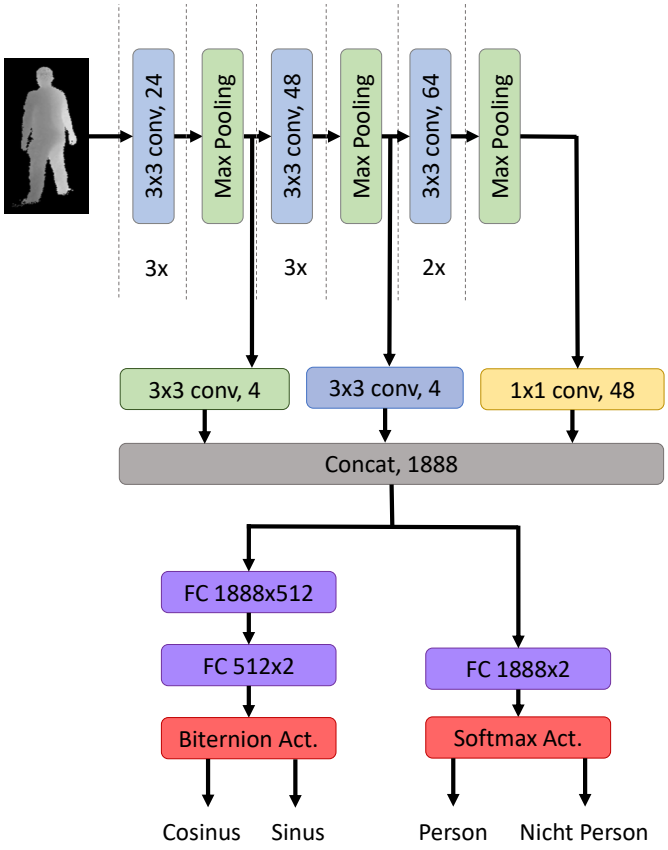


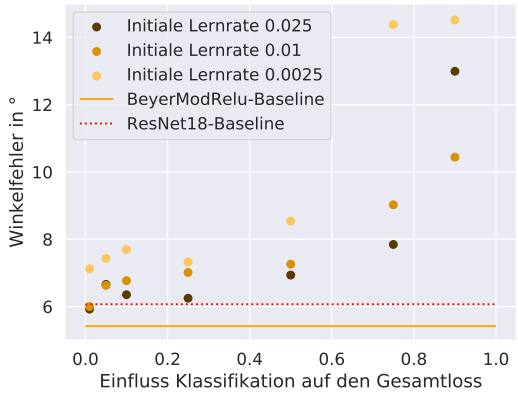
Abbildung A.7: Netzwerkarchitektur BeyerModRelu-Multi-Scale für das Multi-Task Learning von Personenklassifikation und Regression der Oberkörperorientierung *Aus den drei Blöcken des BeyerModRelu-Netzwerk werden durch Faltungsschichten Features extrahiert. Diese werden konkateniert und von den zwei vollverschalteten Netzwerkköpfen verarbeitet.*

A.3.2 Experimente und Ergebnisse

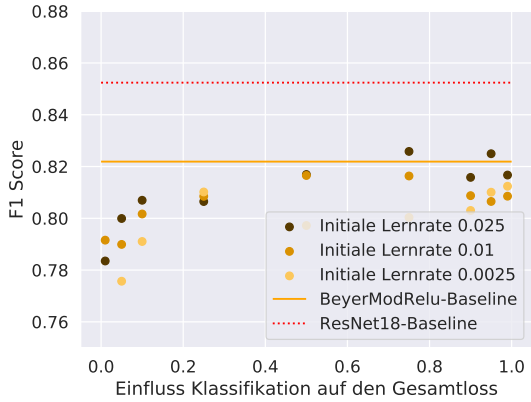
Die BeyerModRelu-Multi-Scale-Architektur zeigt im Rahmen der Experimente keine überzeugenden Ergebnisse. In Abbildung A.8a ist erkennbar, dass die Orientierungsleistung schon bei einer Task-Wichtung von 0,95 für die Orientierungsschätzung sehr stark einbricht und deutlich unter die der Baselines fällt.

Auch bei der Detektion ist auf dem SuPer Datensatz (Abbildung A.8b) durchgehend eine

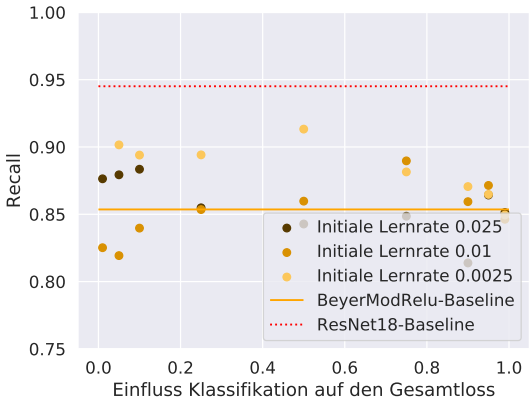
schlechtere Performance im Vergleich zur BeyerModRelu-Baseline festzustellen. Auf dem SRL-Datensatz zeigt sich eine extreme Streuung der Trainings (vgl. Abbildung A.8c). Auf dem TUI-Zuse Datensatz (Abbildung A.8d) ist eine leichte Verbesserung der Erkennungsrate erkennbar. Aufgrund der schlechteren Ergebnisse und der instabilen Trainings bietet sich für einen Einsatz auf dem Roboter also, wenn die BeyerModRelu-Architektur eingesetzt werden soll, eher die Single-Scale-Herangehensweise aus Kapitel 5.3.3 an.



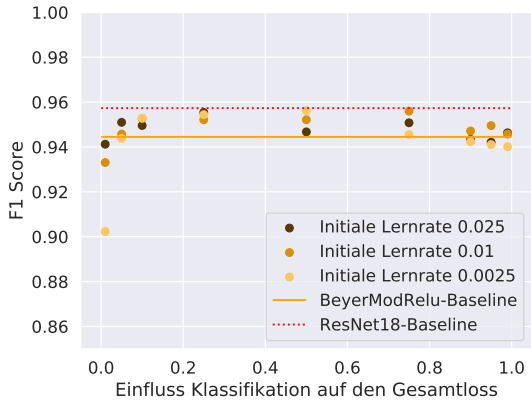
(a) Mittlerer Winkelfehler auf Deep-Orientation



(b) F1 Score auf SuPer



(c) Recall auf SRL



(d) F1 Score auf TUI-Zuse

Abbildung A.8: Netzwerkleistung der BeyerModRelu-Multi-Scale-Architektur. *Dargestellt ist die Netzwerkleistung der BeyerModRelu-Multi-Scale-Architektur auf den vier Testdatensätzen. Die Single-Task-Baselines sind als horizontale Linien eingezeichnet. Die genauen Zahlenwerte und Varianzen sind in Kapitel A.5.10 angegeben.*

A.4 Evaluation der Vorverarbeitung

In der Systempipeline aus Kapitel 4.1 ist erkennbar, dass die Detektionsleistung des Gesamtsystems von zwei Faktoren beeinflusst wird:

1. Wieviele Personen erkennt die Vorverarbeitung als potenziell interessante Region?
2. Wie gut ist die Klassifikation des Multi-Task-Systems?

Nur eine Person, die sowohl von der Vorverarbeitung erkannt als auch vom Multi-Task-System als Person klassifiziert wurde, wird als Person ausgegeben. Daher ist es wichtig, dass die Vorverarbeitung eine möglichst hohe *True Positive Rate (TPR)* aufweist. Dennoch ist aus Sicht der Rechenzeit eine möglichst geringe Anzahl an Falsch Positiven Kandidaten wünschenswert, vgl. hierzu auch Kapitel 5.4.1.
Aus diesen Gründen bietet es sich an, den Kandidatengenerator auf dessen Erkennungsrate und Falschdetektionsrate hin zu untersuchen. Der Kandidatengenerator besitzt insgesamt 12 einstellbare Parameter mit unterschiedlichen Wertebereichen und Bedeutungen (z.B. Höhenangaben für die Schichteneinteilung der Punktewolke oder Filtergrößen zur Rauschminderung). Diese wurden im Rahmen der Untersuchungen unter der Annahme einer maximalen Personengröße und quadratischen Zellen auf fünf relevante, voneinander abhängige Parameter reduziert. Um den Einfluss dieser Parameter auf die False-Positive-Rate und die True-Positive-Rate zu untersuchen, wurde ausgehend von der bisherigen, experimentell bestimmten Parameterkombination eine Grid-Search durchgeführt. Für 1920 verschiedene Parameterkombinationen wurde jeweils die Miss-Rate (MR) und die Falsch-Positiven pro Bild (FPPI) über dem SuPer Testdatensatz ausgewertet. Die Ergebnisse hierzu sind in Abbildung A.9 dargestellt. Um die oben genannten Anforderungen zu erfüllen, bietet sich somit eine Parameterkombination an, die möglichst weit unten links im Miss-Rate- und FPPI-Raum liegt.

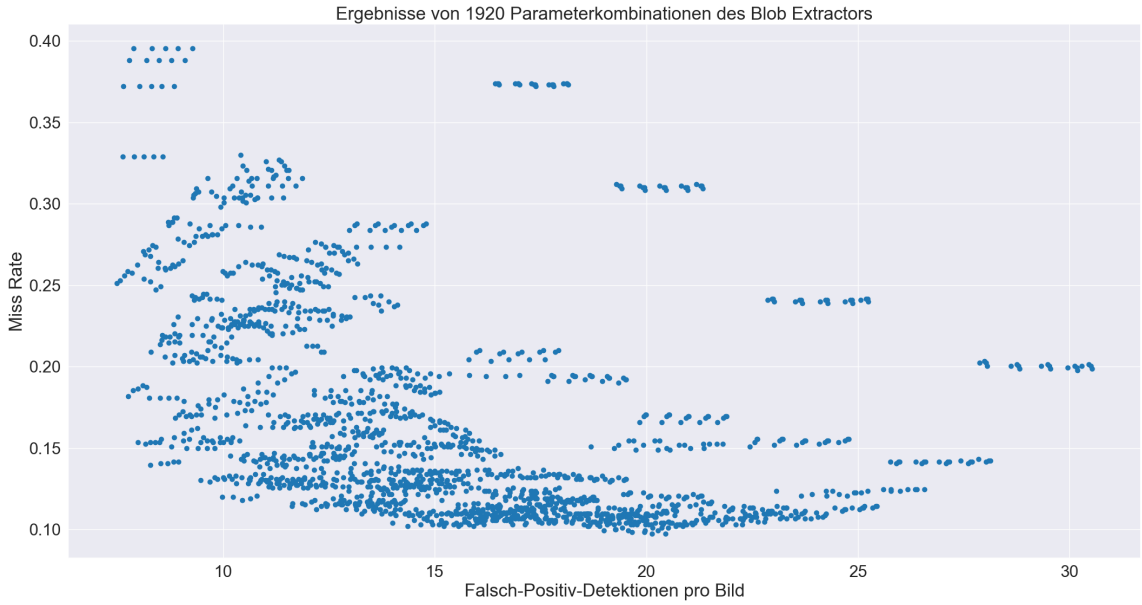


Abbildung A.9: Evaluationsergebnisse zum punktwolkenbasierten Kandidatengenerator

Erkennbar ist, dass bei geschickter Wahl der Parameter mit steigender Anzahl an akzeptierten Falsch-Positiven die Miss-Rate zuerst fällt und danach wieder steigt. Dabei sollten eben nur die Parameter genutzt werden, die für eine gegebene FPPI-Rate die minimale Miss-Rate in der Kurve ergeben. Die Kurve zeigt dabei zwei Minima, welche jedoch eine ähnliche Miss-Rate besitzen. Die im Rahmen dieser Masterarbeit verwendete Parameterkombination besitzt eine Miss-Rate von 11% bei 15 Falsch-Positiven pro Bild. Das zweite, globale Minimum verbessert die Miss-Rate nur um 0,2 Prozentpunkte, nimmt dafür aber etwa 20 Falschdetektionen in Kauf.

A.5 Tabellen und Werte

In diesem Abschnitt sind die numerischen Ergebnisse zu allen Experimenten dieser Masterarbeit abgebildet.

A.5.1 Baseline Personenklassifikation

In Tabelle A.1 sind die Ergebnisse für die Baseline-Experimente der Personenklassifikation (Kapitel 5.2.2) für die Netzwerke BeyerModRelu und ResNet18 dargestellt. Dabei wird noch nach der Verwendung der Augmentierung unterschieden. Die Angaben sind Mittelwerte und Varianzen über zwei Durchläufe.

Init. LR	**Netzwerk**	**Augm.**	**F1-Score SuPer**	**F1-Score TUI-Zuse**	**Recall Deep-Orien.**	**Recall SRL**
0,0001	ResNet18	nein	$81{,}31 \pm 0{,}33$	$94{,}76 \pm 1{,}03$	$99{,}98 \pm 0{,}02$	$93{,}86 \pm 1{,}14$
0,0001	ResNet18	ja	$82{,}03 \pm 0{,}72$	$94{,}99 \pm 0{,}14$	$99{,}99 \pm 0{,}0$	$94{,}82 \pm 0{,}17$
0,0001	BeyerModRelu	nein	$75{,}11 \pm 0{,}61$	$92{,}95 \pm 0{,}6$	$99{,}93 \pm 0{,}01$	$87{,}86 \pm 0{,}43$
0,0001	BeyerModRelu	ja	$77{,}26 \pm 0{,}25$	$92{,}83 \pm 0{,}25$	$99{,}94 \pm 0{,}02$	$86{,}6 \pm 0{,}82$
0,001	ResNet18	nein	$82{,}17 \pm 0{,}67$	$95{,}51 \pm 0{,}45$	$100{,}0 \pm 0{,}0$	$94{,}63 \pm 0{,}89$
0,001	ResNet18	ja	$82{,}5 \pm 0{,}35$	$95{,}82 \pm 0{,}22$	$100{,}0 \pm 0{,}0$	$95{,}44 \pm 0{,}31$
0,001	BeyerModRelu	nein	$78{,}43 \pm 0{,}72$	$93{,}49 \pm 0{,}95$	$99{,}95 \pm 0{,}03$	$86{,}3 \pm 0{,}4$
0,001	BeyerModRelu	ja	$79{,}78 \pm 0{,}29$	$94{,}05 \pm 0{,}02$	$99{,}97 \pm 0{,}0$	$86{,}05 \pm 1{,}74$
0,01	ResNet18	nein	$83{,}22 \pm 0{,}03$	$96{,}06 \pm 0{,}09$	$100{,}0 \pm 0{,}0$	$94{,}75 \pm 0{,}79$
0,01	ResNet18	ja	$85{,}24 \pm 0{,}28$	$95{,}73 \pm 0{,}15$	$100{,}0 \pm 0{,}0$	$94{,}51 \pm 0{,}44$
0,01	BeyerModRelu	nein	$80{,}42 \pm 0{,}3$	$94{,}68 \pm 0{,}41$	$99{,}98 \pm 0{,}0$	$87{,}23 \pm 1{,}16$
0,01	BeyerModRelu	ja	$82{,}19 \pm 0{,}53$	$94{,}45 \pm 0{,}4$	$99{,}97 \pm 0{,}01$	$85{,}36 \pm 0{,}06$

Tabelle A.1: Ergebnisse der Baselines für die Personenklassifikation auf den vier Testdatensätzen

A.5.2 Baseline Orientierungsschätzung

In Tabelle A.2 sind die Ergebnisse für die Baseline-Experimente der Orientierungsschätzung (Kapitel 5.2.2) für die Netzwerke BeyerModRelu und ResNet18 dargestellt. Die Trainings unterscheiden sich in der Verwendung von vortrainierten Gewichten, dem Einsatz einer Augmentierung der Daten und der Vorverarbeitung. Die Angaben sind Mittelwerte und Varianzen über zwei Durchläufe.

Init. LR	Netzwerk	Vortr.	Augm.	Vorverarb.	MAAD
0,0001	ResNet18	ja	ja	Norm.	12,75 ± 0,19
0,0001	ResNet18	ja	nein	Norm.	13,93 ± 0,14
0,0001	ResNet18	nein	ja	Skal.	22,2 ± 0,9
0,0001	ResNet18	nein	nein	Skal.	25,75 ± 0,66
0,0001	ResNet18-Additional-Dense	ja	ja	Skal.	12,42 ± 0,28
0,0001	ResNet18-Additional-Dense	ja	nein	Norm.	12,17 ± 0,15
0,0001	ResNet18-Additional-Dense	ja	nein	Skal.	12,36 ± 0,13
0,001	BeyerModRelu	nein	ja	Skal.	8,0 ± 0,3
0,001	BeyerModRelu	nein	nein	Skal.	8,94 ± 0,18
0,001	ResNet18	ja	ja	Skal.	8,62 ± 0,07
0,001	ResNet18	ja	ja	Norm.	8,64 ± 0,06
0,001	ResNet18	ja	nein	Skal.	9,81 ± 0,21
0,001	ResNet18	ja	nein	Norm.	9,79 ± 0,1
0,001	ResNet18	nein	ja	Skal.	14,71 ± 0,25
0,001	ResNet18	nein	nein	Skal.	16,34 ± 0,63
0,001	ResNet18-Additional-Dense	ja	ja	Skal.	7,76 ± 0,01
0,001	ResNet18-Additional-Dense	ja	nein	Skal.	8,43 ± 0,1
0,001	ResNet18-Additional-Dense	ja	nein	Norm.	8,33 ± 0,04
0,01	BeyerModRelu	nein	ja	Skal.	5,54 ± 0,23
0,01	BeyerModRelu	nein	nein	Skal.	6,06 ± 0,01
0,01	ResNet18	ja	ja	Skal.	7,19 ± 0,14
0,01	ResNet18	ja	ja	Norm.	7,03 ± 0,03
0,01	ResNet18	ja	nein	Skal.	8,57 ± 0,28
0,01	ResNet18	ja	nein	Norm.	7,94 ± 0,24
0,01	ResNet18	nein	ja	Skal.	11,29 ± 0,61
0,01	ResNet18	nein	nein	Skal.	12,29 ± 0,02
0,01	ResNet18-Additional-Dense	ja	ja	Skal.	6,07 ± 0,18
0,01	ResNet18-Additional-Dense	ja	nein	Skal.	7,02 ± 0,32
0,01	ResNet18-Additional-Dense	ja	nein	Norm.	6,89 ± 0,19
0,1	BeyerModRelu	nein	ja	Skal.	5,42 ± 0,08
0,1	BeyerModRelu	nein	nein	Skal.	5,82 ± 0,01
0,1	ResNet18	ja	ja	Skal.	6,67 ± 0,04
0,1	ResNet18	ja	nein	Skal.	8,43 ± 0,01
0,1	ResNet18-Additional-Dense	ja	nein	Skal.	7,7 ± 0,88

Tabelle A.2: Mittlerer absoluter Winkelfehler über den Testdaten Deep-Orientation für die Orientierungsbaseline-Experimente

A.5.3 Auswahlmethodik Multi-Task

Im Rahmen der Multi-Task-Vorexperimente musste in Kapitel 5.3.1 festgelegt werden, wie die beste Epoche im Multi-Task-Szenario bestimmt wird. In den folgenden Tabellen sind die Ergebnisse für die Auswahlmethoden *Nach Personenklassifikation* (Tabelle A.3), *Nach Orientierungsschätzung* (Tabelle A.4) und *Nach Rank-Level-Fusion* (Tabelle A.5) für das ResNet18-Single-Scale, Lernrate 0,01, dargestellt.

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	6,799 ±0,503	9,278 ±1,188	9,936 ±3,91	6,216 ±0,18	9,377 ±3,547	10,296 ±3,306	10,325 ±2,34	9,297 ±0,773	18,427 ±4,135
F1-Score auf SuPer	0,826 ±0,009	0,829 ±0,006	0,84 ±0,015	0,845 ±0,008	0,843 ±0,009	0,845 ±0,008	0,847 ±0,003	0,84 ±0,007	0,845 ±0,009
Recall auf SRL	0,93 ±0,006	0,95 ±0,008	0,954 ±0,007	0,942 ±0,012	0,929 ±0,007	0,954 ±0,013	0,952 ±0,009	0,963 ±0,008	0,943 ±0,003
F1-Score auf TUI-Zuse	0,956 ±0,003	0,964 ±0,002	0,959 ±0,001	0,964 ±0,002	0,955 ±0,003	0,962 ±0,006	0,955 ±0,001	0,961 ±0,002	0,956 ±0,002

Tabelle A.3: Performancemaße für Netzwerkauswahl anhand der Klassifikations-Tasks
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Hierbei wurde die beste Epoche anhand des Klassifikations-Tasks ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	5,934 ±0,172	6,219 ±0,17	6,012 ±0,035	6,222 ±0,107	6,332 ±0,247	6,623 ±0,323	7,523 ±0,179	8,483 ±0,315	12,903 ±0,823
F1-Score auf SuPer	0,825 ±0,006	0,836 ±0,004	0,844 ±0,013	0,851 ±0,006	0,843 ±0,008	0,85 ±0,002	0,844 ±0,006	0,841 ±0,005	0,839 ±0,013
Recall auf SRL	0,916 ±0,016	0,939 ±0,012	0,948 ±0,001	0,936 ±0,009	0,951 ±0,008	0,957 ±0,009	0,941 ±0,002	0,962 ±0,005	0,953 ±0,006
F1-Score auf TUI-Zuse	0,949 ±0,003	0,958 ±0,003	0,957 ±0,002	0,958 ±0,003	0,962 ±0,003	0,959 ±0,005	0,952 ±0,005	0,961 ±0,001	0,96 ±0,004

Tabelle A.4: Performancemaße für Netzwerkauswahl anhand des Orientierungs-Tasks
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Hierbei wurde die beste Epoche anhand des Orientierungs-Tasks ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	5,978 ±0,15	6,348 ±0,199	6,002 ±0,102	6,348 ±0,067	6,574 ±0,352	6,695 ±0,3	7,67 ±0,314	8,557 ±0,21	13,037 ±0,632
F1-Score auf SuPer	0,824 ±0,011	0,83 ±0,004	0,843 ±0,014	0,846 ±0,008	0,845 ±0,007	0,847 ±0,003	0,844 ±0,005	0,84 ±0,006	0,843 ±0,009
Recall auf SRL	0,926 ±0,011	0,947 ±0,012	0,953 ±0,005	0,942 ±0,004	0,94 ±0,012	0,959 ±0,006	0,945 ±0,009	0,964 ±0,007	0,947 ±0,007
F1-Score auf TUI-Zuse	0,954 ±0,004	0,961 ±0,004	0,959 ±0,001	0,96 ±0,005	0,958 ±0,004	0,961 ±0,005	0,956 ±0,004	0,962 ±0,001	0,956 ±0,004

Tabelle A.5: Performancemaße für Netzwerkauswahl anhand der Rang-Fusion
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Hierbei wurde die beste Epoche anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

A.5.4 Datensatzzusammenstellung

Im Rahmen der Multi-Task-Vorexperimente musste in Kapitel 5.3.2 festgelegt werden, wie die Datenbasis am besten in Batches zusammengestellt wird. In den folgenden Tabellen sind die Ergebnisse für die Methoden *Konkatenieren* (Tabelle A.6), *50-50 Teilen* (Tabelle A.7) und *exakt 50-50 Teilen* (Tabelle A.8) dargestellt.

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	6,333 ±0,17	6,039 ±0,037	6,036 ±0,012	6,276 ±0,353	6,426 ±0,162	6,687 ±0,126	7,494 ±0,215	8,304 ±0,238	14,868 ±1,66
F1-Score auf SuPer	0,825 ±0,015	0,844 ±0,007	0,839 ±0,007	0,845 ±0,003	0,849 ±0,005	0,85 ±0,005	0,84 ±0,005	0,838 ±0,005	0,839 ±0,006
Recall auf SRL	0,932 ±0,016	0,932 ±0,003	0,944 ±0,023	0,944 ±0,008	0,94 ±0,004	0,941 ±0,019	0,952 ±0,018	0,942 ±0,009	0,964 ±0,002
F1-Score auf TUI-Zuse	0,951 ±0,003	0,957 ±0,002	0,961 ±0,005	0,963 ±0,004	0,96 ±0,003	0,956 ±0,003	0,96 ±0,005	0,957 ±0,004	0,966 ±0,004

Tabelle A.6: Performancemaße für Datenzusammenstellung anhand der Konkatenation

Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden konkateniert und zum Training verwendet. Hierbei wurde die beste Epoche anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	5,978 ±0,15	6,348 ±0,199	6,002 ±0,102	6,348 ±0,067	6,574 ±0,352	6,695 ±0,3	7,67 ±0,314	8,557 ±0,21	13,037 ±0,632
F1-Score auf SuPer	0,824 ±0,011	0,83 ±0,004	0,843 ±0,014	0,846 ±0,008	0,845 ±0,007	0,847 ±0,003	0,844 ±0,005	0,84 ±0,006	0,843 ±0,009
Recall auf SRL	0,926 ±0,011	0,947 ±0,012	0,953 ±0,005	0,942 ±0,004	0,94 ±0,012	0,959 ±0,006	0,945 ±0,009	0,964 ±0,007	0,947 ±0,007
F1-Score auf TUI-Zuse	0,954 ±0,004	0,961 ±0,004	0,959 ±0,001	0,96 ±0,005	0,958 ±0,004	0,961 ±0,005	0,956 ±0,004	0,962 ±0,001	0,956 ±0,004

Tabelle A.7: Performancemaße für Datenzusammenstellung anhand der Vorgehensweise 50-50
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Hierbei wurde die beste Epoche anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	6,196 ±0,145	6,091 ±0,162	6,16 ±0,174	5,965 ±0,046	6,785 ±0,404	7,085 ±0,106	7,799 ±0,445	9,146 ±0,283	16,061 ±0,092
F1-Score auf SuPer	0,829 ±0,004	0,842 ±0,008	0,841 ±0,0	0,848 ±0,001	0,842 ±0,002	0,845 ±0,005	0,85 ±0,007	0,851 ±0,001	0,853 ±0,006
Recall auf SRL	0,948 ±0,0	0,95 ±0,002	0,95 ±0,005	0,948 ±0,012	0,932 ±0,018	0,956 ±0,007	0,951 ±0,002	0,949 ±0,011	0,939 ±0,006
F1-Score auf TUI-Zuse	0,955 ±0,003	0,956 ±0,003	0,961 ±0,0	0,959 ±0,007	0,956 ±0,006	0,959 ±0,003	0,955 ±0,004	0,957 ±0,001	0,953 ±0,0

Tabelle A.8: Performancemaße für Datenzusammenstellung anhand der Vorgehensweise exakt 50-50
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Dabei wurde der Loss so ausmaskiert, dass beide Tasks mit der selben Batch Size trainieren. Hierbei wurde die beste Epoche anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

A.5.5 ResNet18-Single-Scale-Multi-Task

Im Rahmen der Multi-Task-Experimente zu den Single-Scale-Architekturen wurden in Kapitel 5.3.3 sowohl die ResNet18-Single-Scale-, als auch die BeyerModRelu-Single-Scale-Architektur analysiert. In den folgenden Tabellen sind die Ergebnisse für verschiedene Lernraten für die ResNet18-Architekturen dargestellt:

- Gemittelte Ergebnisse, Lernrate 0,05: Tabelle A.9
- Gemittelte Ergebnisse, Lernrate 0,01: Tabelle A.10
- Gemittelte Ergebnisse, Lernrate 0,001: Tabelle A.11
- Ungemittelte Ergebnisse, Lernrate 0,05: Tabelle A.12
- Ungemittelte Ergebnisse, Lernrate 0,01: Tabelle A.13
- Ungemittelte Ergebnisse, Lernrate 0,001: Tabelle A.14

Zusätzlich sind in den Abbildungen A.10 und A.11 die ungemittelten Ergebnisse für die Lernraten 0,05 und 0,001 als Diagramme dargestellt.

Gemittelte Ergebnisse ResNet18-Single-Scale-Multi-Task

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	5,925 ±0,233	6,03 ±0,191	5,882 ±0,171	7,63 ±1,59	5,68 ±0,06	6,39 ±0,252	7,828 ±0,487	8,848 ±0,553	14,976 ±0,918
F1-Score auf SuPer	0,811 ±0,018	0,839 ±0,003	0,836 ±0,008	0,816 ±0,034	0,866 ±0,006	0,857 ±0,007	0,848 ±0,007	0,848 ±0,009	0,853 ±0,003
Recall auf SRL	0,775 ±0,105	0,896 ±0,005	0,908 ±0,003	0,86 ±0,062	0,95 ±0,014	0,932 ±0,015	0,909 ±0,023	0,94 ±0,005	0,932 ±0,022
F1-Score auf TUI-Zuse	0,953 ±0,001	0,956 ±0,005	0,957 ±0,003	0,95 ±0,011	0,966 ±0,003	0,959 ±0,001	0,957 ±0,003	0,96 ±0,003	0,958 ±0,004

Tabelle A.9: Performancemaße für die ResNet18-Single-Scale-Multi-Task-Architektur mit Lernrate 0,05
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Die beste Epoche wurde anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	5,978 ±0,15	6,348 ±0,199	6,002 ±0,102	6,348 ±0,067	6,574 ±0,352	6,695 ±0,3	7,67 ±0,314	8,557 ±0,21	13,037 ±0,632
F1-Score auf SuPer	0,824 ±0,011	0,83 ±0,004	0,843 ±0,014	0,846 ±0,008	0,845 ±0,007	0,847 ±0,003	0,844 ±0,005	0,84 ±0,006	0,843 ±0,009
Recall auf SRL	0,926 ±0,011	0,947 ±0,012	0,953 ±0,005	0,942 ±0,004	0,94 ±0,012	0,959 ±0,006	0,945 ±0,009	0,964 ±0,007	0,947 ±0,007
F1-Score auf TUI-Zuse	0,954 ±0,004	0,961 ±0,004	0,959 ±0,001	0,96 ±0,005	0,958 ±0,004	0,961 ±0,005	0,956 ±0,004	0,962 ±0,001	0,956 ±0,004

Tabelle A.10: Performancemaße für die ResNet18-Single-Scale-Multi-Task-Architektur mit Lernrate 0,01
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Die beste Epoche wurde anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	7,236 ±0,211	7,58 ±0,177	7,462 ±0,245	7,697 ±0,13	8,371 ±0,113	9,628 ±0,101	11,62 ±0,196	12,999 ±0,301	23,499 ±0,884
F1-Score auf SuPer	0,775 ±0,011	0,824 ±0,006	0,833 ±0,006	0,823 ±0,009	0,83 ±0,004	0,836 ±0,012	0,838 ±0,005	0,829 ±0,002	0,843 ±0,008
Recall auf SRL	0,862 ±0,014	0,919 ±0,014	0,927 ±0,012	0,943 ±0,002	0,941 ±0,017	0,955 ±0,004	0,947 ±0,004	0,944 ±0,011	0,942 ±0,002
F1-Score auf TUI-Zuse	0,941 ±0,003	0,952 ±0,005	0,952 ±0,004	0,955 ±0,004	0,958 ±0,005	0,955 ±0,005	0,955 ±0,004	0,96 ±0,002	0,955 ±0,002

Tabelle A.11: Performancemaße für die ResNet18-Single-Scale-Multi-Task-Architektur mit Lernrate 0,001
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Die beste Epoche wurde anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

Ungemittelte Ergebnisse ResNet18-Single-Scale-Multi-Task

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	5,815	6,291	6,116	6,068	5,763	6,445	7,14	8,476	16,216
	5,712	5,837	5,818	7,011	5,658	6,058	8,209	9,63	14,025
	6,249	5,961	5,711	9,812	5,62	6,669	8,135	8,438	14,686
F1-Score auf SuPer	0,831	0,84	0,824	0,838	0,871	0,86	0,858	0,836	0,856
	0,814	0,835	0,842	0,842	0,869	0,863	0,845	0,858	0,849
	0,787	0,842	0,841	0,767	0,858	0,847	0,842	0,851	0,853
Recall auf SRL	0,864	0,902	0,91	0,904	0,929	0,954	0,929	0,935	0,902
	0,834	0,895	0,905	0,904	0,957	0,925	0,922	0,938	0,95
	0,627	0,891	0,91	0,773	0,962	0,918	0,877	0,948	0,945
F1-Score auf TUI-Zuse	0,955	0,949	0,953	0,959	0,962	0,961	0,953	0,964	0,952
	0,952	0,961	0,958	0,957	0,966	0,958	0,959	0,957	0,963
	0,953	0,958	0,96	0,935	0,969	0,96	0,959	0,96	0,958

Tabelle A.12: Performancemaße für die ResNet18-Single-Scale-Multi-Task-Architektur mit Lernrate 0,05

Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Die beste Epoche wurde anhand der Rang-Fusion ausgewählt. Die Angaben sind Einzelergebnisse der drei Trainings.

Taskwichtung									
Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf	5,811	6,385	5,859	6,264	6,093	6,771	7,908	8,259	12,41
Deep-Orientation	6,175	6,089	6,062	6,351	6,704	6,296	7,875	8,711	12,799
	5,949	6,572	6,086	6,428	6,925	7,019	7,225	8,7	13,903
F1-Score auf	0,824	0,83	0,827	0,855	0,848	0,851	0,851	0,832	0,842
SuPer	0,81	0,835	0,86	0,848	0,835	0,846	0,84	0,839	0,833
	0,837	0,825	0,843	0,835	0,852	0,843	0,841	0,848	0,854
Recall auf	0,919	0,961	0,958	0,936	0,955	0,953	0,951	0,971	0,945
SRL	0,941	0,932	0,947	0,946	0,94	0,967	0,932	0,967	0,957
	0,917	0,949	0,952	0,945	0,925	0,958	0,951	0,954	0,94
F1-Score auf	0,949	0,966	0,96	0,953	0,963	0,955	0,961	0,962	0,953
TUI-Zuse	0,958	0,957	0,959	0,963	0,957	0,967	0,954	0,963	0,961
	0,955	0,962	0,959	0,965	0,954	0,962	0,951	0,961	0,953

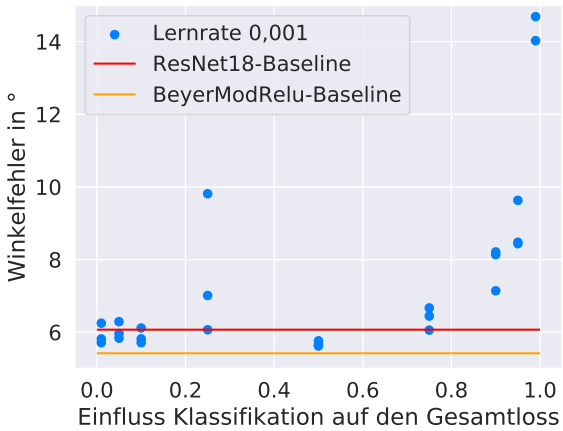
Tabelle A.13: Performancemaße für die ResNet18-Single-Scale-Multi-Task-Architektur mit Lernrate 0,01

Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Die beste Epoche wurde anhand der Rang-Fusion ausgewählt. Die Angaben sind Einzelergebnisse der drei Trainings.

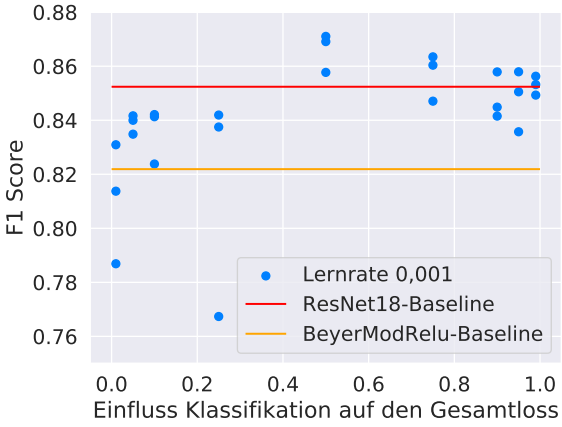
Taskwichtung									
Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf	6,94	7,809	7,122	7,529	8,411	9,539	11,603	12,737	22,256
Deep-Orientation	7,421	7,379	7,69	7,845	8,218	9,577	11,869	13,421	24,229
	7,348	7,553	7,575	7,716	8,486	9,769	11,389	12,84	24,013
F1-Score auf	0,79	0,818	0,83	0,815	0,829	0,819	0,84	0,832	0,853
SuPer	0,765	0,821	0,841	0,818	0,836	0,844	0,83	0,827	0,834
	0,77	0,832	0,828	0,835	0,826	0,845	0,842	0,828	0,843
Recall auf	0,845	0,903	0,932	0,941	0,92	0,956	0,951	0,951	0,945
SRL	0,879	0,938	0,911	0,942	0,945	0,96	0,95	0,928	0,942
	0,863	0,916	0,939	0,945	0,96	0,949	0,941	0,953	0,941
F1-Score auf	0,937	0,946	0,95	0,952	0,951	0,961	0,958	0,962	0,952
TUI-Zuse	0,943	0,958	0,949	0,961	0,963	0,954	0,95	0,957	0,954
	0,942	0,954	0,957	0,953	0,962	0,95	0,957	0,96	0,958

Tabelle A.14: Performancemaße für die ResNet18-Single-Scale-Multi-Task-Architektur mit Lernrate 0,001

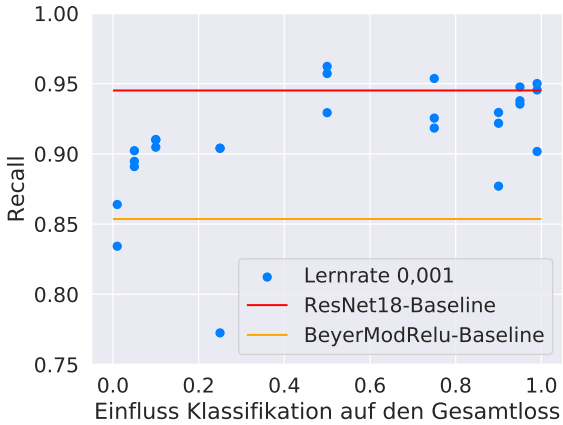
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Die beste Epoche wurde anhand der Rang-Fusion ausgewählt. Die Angaben sind Einzelergebnisse der drei Trainings.



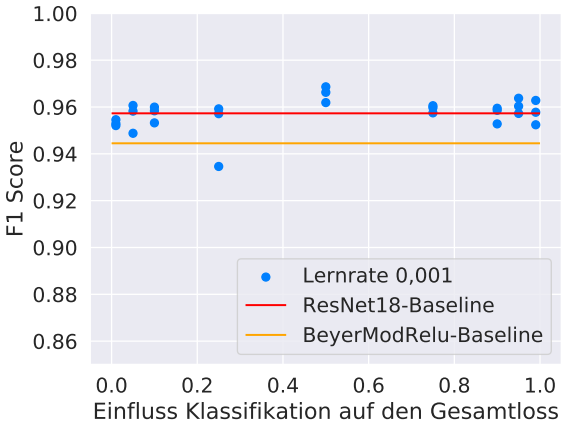
(a) Winkelfehler auf Deep-Orientation



(b) F1 Score auf SuPer



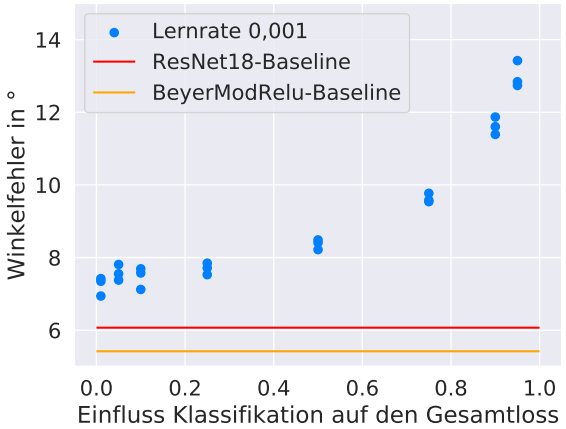
(c) Recall auf SRL



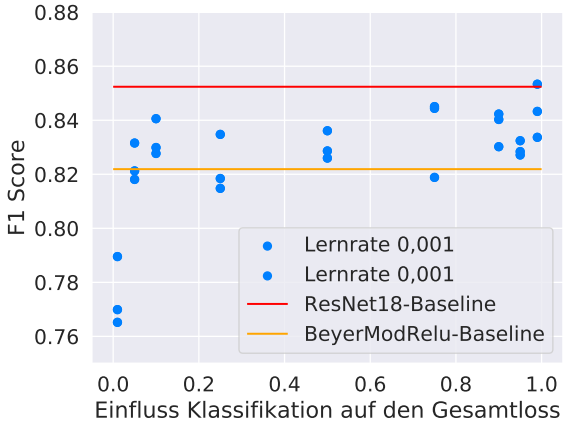
(d) F1 Score auf TUI-Zuse

Abbildung A.10: Ungemittelte Netzwerkperformance der ResNet18-Single-Scale-Multi-Task-Architektur

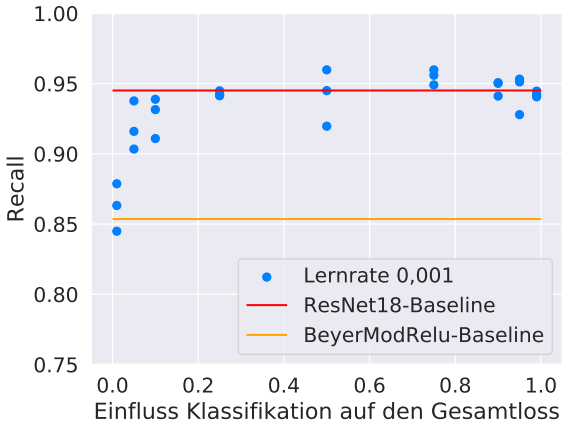
Dargestellt sind die Ergebnisse auf den Testdaten für die ResNet18-Single-Scale-Architektur mit initialer Lernrate 0,05. Dabei wurden je Taskwichtung drei Trainings durchgeführt, welche hier ungemittelt angetragen sind.



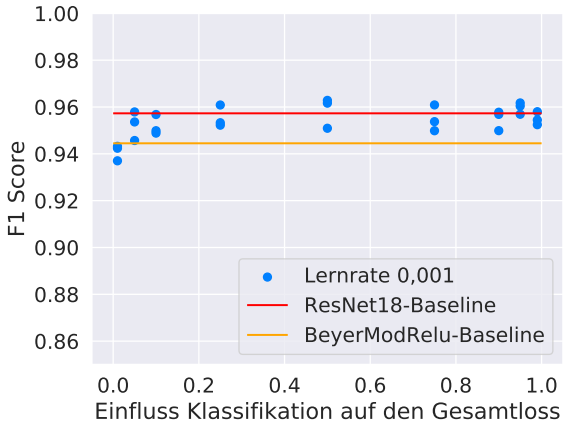
(a) Winkelfehler auf Deep-Orientation



(b) F1 Score auf SuPer



(c) Recall auf SRL



(d) F1 Score auf TUI-Zuse

Abbildung A.11: Ungemittelte Netzwerkperformance der ResNet18-Single-Scale-Multi-Task-Architektur

Dargestellt sind die Ergebnisse auf den Testdaten für die ResNet18-Single-Scale-Architektur mit initialer Lernrate 0,001. Dabei wurden je Taskwichtung drei Trainings durchgeführt, welche hier ungemittelt angetragen sind.

A.5.6 BeyerModRelu-Single-Scale-Multi-Task

Nach den ResNet18-Single-Scale-Architekturen folgen nun die BeyerModRelu-Single-Scale-Multi-Task-Architekturen (Kapitel 5.3.3). In den folgenden Tabellen sind die Ergebnisse für verschiedene Lernraten für die BeyerModRelu-Architekturen dargestellt:

- Gemittelte Ergebnisse, Lernrate 0,05: Tabelle A.15
- Gemittelte Ergebnisse, Lernrate 0,01: Tabelle A.16
- Gemittelte Ergebnisse, Lernrate 0,001: Tabelle A.17

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	5,584 ±0,106	5,924 ±0,202	5,965 ±0,567	6,262 ±0,097	6,489 ±0,457	7,615 ±0,194	11,285 ±1,944	15,102 ±0,543	23,928 ±0,22
F1-Score auf SuPer	0,786 ±0,006	0,81 ±0,008	0,813 ±0,008	0,826 ±0,004	0,822 ±0,008	0,826 ±0,002	0,815 ±0,003	0,827 ±0,002	0,818 ±0,007
Recall auf SRL	0,819 ±0,023	0,869 ±0,006	0,879 ±0,029	0,857 ±0,029	0,868 ±0,019	0,838 ±0,033	0,853 ±0,03	0,852 ±0,018	0,852 ±0,038
F1-Score auf TUI-Zuse	0,944 ±0,006	0,949 ±0,009	0,958 ±0,007	0,959 ±0,002	0,961 ±0,004	0,957 ±0,002	0,957 ±0,002	0,949 ±0,003	0,952 ±0,004

Tabelle A.15: Performancemaße für die BeyerModRelu-Single-Scale-Multi-Task-Architektur mit Lernrate 0,05
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Die beste Epoche wurde anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	6,27 ±0,369	6,567 ±0,347	6,314 ±0,232	6,545 ±0,126	8,051 ±1,103	9,59 ±1,661	11,437 ±1,665	15,588 ±0,148	26,151 ±0,775
F1-Score auf SuPer	0,763 ±0,013	0,808 ±0,004	0,808 ±0,011	0,804 ±0,006	0,818 ±0,006	0,82 ±0,003	0,811 ±0,008	0,819 ±0,003	0,816 ±0,007
Recall auf SRL	0,841 ±0,004	0,871 ±0,049	0,865 ±0,02	0,892 ±0,03	0,826 ±0,039	0,863 ±0,025	0,87 ±0,021	0,868 ±0,031	0,852 ±0,015
F1-Score auf TUI-Zuse	0,928 ±0,01	0,952 ±0,009	0,954 ±0,005	0,96 ±0,003	0,955 ±0,001	0,955 ±0,006	0,952 ±0,003	0,945 ±0,003	0,944 ±0,001

Tabelle A.16: Performancemaße für die BeyerModRelu-Single-Scale-Multi-Task-Architektur mit Lernrate 0,01
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Die beste Epoche wurde anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	8,033 ±0,333	8,03 ±0,576	8,333 ±0,831	10,335 ±2,391	10,978 ±1,813	13,779 ±0,74	19,995 ±3,014	24,823 ±1,482	37,985 ±0,698
F1-Score auf SuPer	0,657 ±0,036	0,768 ±0,003	0,777 ±0,007	0,789 ±0,008	0,792 ±0,007	0,809 ±0,002	0,785 ±0,006	0,799 ±0,004	0,792 ±0,003
Recall auf SRL	0,664 ±0,013	0,823 ±0,038	0,895 ±0,03	0,918 ±0,006	0,894 ±0,028	0,887 ±0,018	0,892 ±0,002	0,884 ±0,006	0,878 ±0,022
F1-Score auf TUI-Zuse	0,887 ±0,003	0,929 ±0,011	0,945 ±0,004	0,947 ±0,009	0,948 ±0,001	0,943 ±0,004	0,944 ±0,001	0,946 ±0,002	0,937 ±0,003

Tabelle A.17: Performancemaße für die BeyerModRelu-Single-Scale-Multi-Task-Architektur mit Lernrate 0,001
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Die beste Epoche wurde anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

A.5.7 Baselines ResNet18-Multi-Scale

Die Ergebnisse für die GAP- und LAP-Baselines für die beiden ResNet18-Multi-Scale-Netzwerke aus Kapitel 5.3.4 sind in den Tabellen A.18 (Klassifikation) und A.19 (Orientierungsschätzung) dargestellt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

Netzwerk	**Init. Lernrate**	**F1-Score SuPer**	**F1-Score TUI-Zuse**	**Recall SRL**
ResNet18-MultiScale-GAP	0,001	$82{,}92 \pm 0{,}28$	$95{,}28 \pm 0{,}1$	$95{,}29 \pm 0{,}61$
	0,01	$84{,}53 \pm 1{,}05$	$95{,}49 \pm 0{,}22$	$93{,}55 \pm 0{,}6$
	0,05	$85{,}35 \pm 1{,}08$	$95{,}98 \pm 0{,}38$	$93{,}69 \pm 1{,}8$
ResNet18-Multi-Scale-LAP	0,001	$82{,}45 \pm 0{,}32$	$95{,}75 \pm 0{,}38$	$95{,}61 \pm 0{,}52$
	0,01	$84{,}39 \pm 0{,}42$	$95{,}6 \pm 0{,}51$	$94{,}74 \pm 1{,}19$
	0,05	$81{,}27 \pm 0{,}85$	$95{,}14 \pm 0{,}39$	$88{,}45 \pm 2{,}66$

Tabelle A.18: Ergebnisse der Multi-Scale-Baselines für die Personenklassifikation auf den drei Testdatensätzen

Architektur	**Initiale Lernrate**	**MAAD auf Deep-Orientation**
ResNet18-Multi-Scale-GAP	0,001	$7{,}27 \pm 0{,}16$
	0,01	$6{,}24 \pm 0{,}26$
	0,05	$5{,}72 \pm 0{,}09$
ResNet18-Multi-Scale-LAP	0,001	$7{,}29 \pm 0{,}19$
	0,01	$6{,}4 \pm 0{,}34$
	0,05	$6{,}34 \pm 0{,}54$

Tabelle A.19: Mittlerer absoluter Winkelfehler über den Testdaten Deep-Orientation für die Multi-Scale-Orientierungsbaseline

A.5.8 ResNet18-Multi-Scale-GAP-Architektur

Für die Multi-Scale-Netzwerkarchitektur ResNet18-Multi-Scale-GAP aus Kapitel 5.3.4 wurde aufgrund begrenzter Zeit für die Experimente nur die Lernrate 0,01 evaluiert. Die Ergebnisse dieser Multi-Task-Architektur sind in Tabelle A.20 angegeben. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	5,847 ±0,164	6,169 ±0,059	6,057 ±0,26	6,501 ±0,38	6,22 ±0,02	6,789 ±0,573	7,661 ±0,051	8,383 ±0,354	13,944 ±1,822
F1-Score auf SuPer	0,822 ±0,006	0,833 ±0,011	0,845 ±0,002	0,842 ±0,012	0,845 ±0,006	0,839 ±0,006	0,853 ±0,011	0,85 ±0,005	0,84 ±0,004
Recall auf SRL	0,93 ±0,008	0,936 ±0,006	0,946 ±0,015	0,946 ±0,009	0,953 ±0,007	0,951 ±0,015	0,961 ±0,005	0,958 ±0,009	0,944 ±0,008
F1-Score auf TUI-Zuse	0,958 ±0,003	0,956 ±0,003	0,958 ±0,004	0,955 ±0,004	0,96 ±0,004	0,958 ±0,007	0,961 ±0,001	0,962 ±0,005	0,956 ±0,002

Tabelle A.20: Performancemaße für die ResNet18-Multi-Scale-GAP-Architektur mit Lernrate 0,01
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Die beste Epoche wurde anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

A.5.9 ResNet18-Multi-Scale-LAP-Architektur

Für die Multi-Scale-Netzwerkarchitektur ResNet18-Multi-Scale-LAP aus Kapitel 5.3.4 wurde aufgrund begrenzter Zeit für die Experimente nur die Lernrate 0,01 evaluiert. Die Ergebnisse dieser Multi-Task-Architektur sind in Tabelle A.21 angegeben. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	6,233 ±0,164	6,15 ±0,207	6,153 ±0,155	6,175 ±0,129	6,526 ±0,367	6,674 ±0,149	8,179 ±0,557	8,779 ±0,235	14,357 ±1,604
F1-Score auf SuPer	0,82 ±0,004	0,833 ±0,005	0,826 ±0,012	0,842 ±0,003	0,836 ±0,012	0,841 ±0,003	0,842 ±0,002	0,842 ±0,014	0,838 ±0,005
Recall auf SRL	0,923 ±0,007	0,938 ±0,015	0,928 ±0,007	0,952 ±0,008	0,939 ±0,019	0,956 ±0,007	0,963 ±0,006	0,947 ±0,006	0,947 ±0,008
F1-Score auf TUI-Zuse	0,955 ±0,003	0,961 ±0,004	0,957 ±0,005	0,957 ±0,005	0,96 ±0,007	0,963 ±0,003	0,962 ±0,004	0,957 ±0,003	0,959 ±0,002

Tabelle A.21: Performancemaße für die ResNet18-Multi-Scale-LAP-Architektur mit Lernrate 0,01
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Die beste Epoche wurde anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

A.5.10 BeyerModRelu-Multi-Scale-Architektur

Im Rahmen der Multi-Task-Experimente zu den Multi-Scale-Architekturen wurde in Kapitel A.3 die BeyerModRelu-Multi-Scale-Architektur analysiert. In den folgenden Tabellen sind die Ergebnisse für verschiedene Lernraten für diese dargestellt:

- Gemittelte Ergebnisse, Lernrate 0,025: Tabelle A.22
- Gemittelte Ergebnisse, Lernrate 0,01: Tabelle A.23
- Gemittelte Ergebnisse, Lernrate 0,0025: Tabelle A.24

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	5,924 ±0,073	6,66 ±0,108	6,353 ±0,627	6,25 ±0,31	6,936 ±0,195	7,846 ±0,56	12,993 ±1,525	15,844 ±3,121	23,806 ±0,783
F1-Score auf SuPer	0,784 ±0,022	0,8 ±0,002	0,807 ±0,007	0,806 ±0,007	0,817 ±0,004	0,826 ±0,001	0,816 ±0,002	0,825 ±0,001	0,817 ±0,002
Recall auf SRL	0,876 ±0,023	0,879 ±0,004	0,883 ±0,002	0,855 ±0,018	0,843 ±0,059	0,849 ±0,022	0,814 ±0,011	0,864 ±0,031	0,851 ±0,004
F1-Score auf TUI-Zuse	0,941 ±0,012	0,951 ±0,007	0,95 ±0,004	0,955 ±0,005	0,947 ±0,013	0,951 ±0,003	0,943 ±0,002	0,942 ±0,007	0,946 ±0,001

Tabelle A.22: Performancemaße für die BeyerModRelu-Multi-Scale-Multi-Task-Architektur mit Lernrate 0,025
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Die beste Epoche wurde anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	6,0 ±0,137	6,63 ±0,276	6,77 ±0,294	7,013 ±0,265	7,26 ±0,115	9,026 ±0,111	10,44 ±0,444	15,411 ±0,238	25,479 ±0,205
F1-Score auf SuPer	0,792 ±0,0	0,79 ±0,003	0,802 ±0,001	0,808 ±0,005	0,817 ±0,001	0,816 ±0,001	0,809 ±0,002	0,807 ±0,004	0,809 ±0,002
Recall auf SRL	0,825 ±0,068	0,819 ±0,042	0,84 ±0,023	0,853 ±0,004	0,86 ±0,009	0,89 ±0,012	0,859 ±0,008	0,871 ±0,022	0,846 ±0,011
F1-Score auf TUI-Zuse	0,933 ±0,007	0,946 ±0,008	0,953 ±0,004	0,952 ±0,003	0,952 ±0,003	0,956 ±0,003	0,947 ±0,003	0,95 ±0,002	0,946 ±0,0

Tabelle A.23: Performancemaße für die BeyerModRelu-Multi-Scale-Multi-Task-Architektur mit Lernrate 0,01
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Die beste Epoche wurde anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

Taskwichtung Klassifikation	0,01	0,05	0,1	0,25	0,5	0,75	0,9	0,95	0,99
MAAD auf Deep-Orient.	7,12 ±0,147	7,432 ±0,383	7,697 ±0,433	7,325 ±0,201	8,542 ±0,336	14,38 ±0,025	14,517 ±1,36	16,976 ±0,18	30,588 ±0,417
F1-Score auf SuPer	0,685 ±0,011	0,776 ±0,002	0,791 ±0,006	0,81 ±0,001	0,797 ±0,001	0,8 ±0,012	0,803 ±0,005	0,81 ±0,006	0,812 ±0,0
Recall auf SRL	0,717 ±0,016	0,902 ±0,011	0,894 ±0,012	0,894 ±0,0	0,913 ±0,01	0,881 ±0,003	0,871 ±0,028	0,865 ±0,02	0,849 ±0,003
F1-Score auf TUI-Zuse	0,902 ±0,014	0,944 ±0,007	0,953 ±0,001	0,954 ±0,002	0,956 ±0,001	0,946 ±0,007	0,942 ±0,002	0,941 ±0,002	0,94 ±0,0

Tabelle A.24: Performancemaße für die BeyerModRelu-Multi-Scale-Multi-Task-Architektur mit Lernrate 0,0025
Für die vier Datensätze sind die Performancemaße (Orientierungsschätzung und Klassifikation der Personen) angegeben. Die Datensätze SuPer und Deep-Orientation wurden im Verhältnis 50-50 kombiniert und zum Training verwendet. Die beste Epoche wurde anhand der Rang-Fusion ausgewählt. Die Angaben sind Mittelwerte und Varianzen über drei Durchläufe.

A.5.11 Vorexperimente

Im Folgenden sind die Ergebnisse für die Vorexperimente zur Personenklassifikation aus Kapitel A.1 dargestellt. Hierbei wird für die Netzwerkarchitekturen (ResNet18, ResNeXt50, BeyerModRelu) für verschiedenen Lernraten und Vorverarbeitungen auf den vier Datensätzen jeweils die Personenklassifikationsleistung angegeben. In den Tabellen finden sich folgende Experimente:

- ResNeXt50, 400 Epochen auf SuPer, Normalisierung und Skalierung: Tabelle A.25
- ResNet18, 400 Epochen auf SuPer, Normalisierung: Tabelle A.26
- ResNet18, 400 Epochen auf SuPer, Skalierung: Tabelle A.27
- ResNet18, 15 Epochen auf SuPer, Normalisierung: Tabelle A.28
- ResNet18, 15 Epochen auf SuPer, Skalierung: Tabelle A.29
- BeyerModRelu, 400 Epochen auf SuPer, Normalisierung und Skalierung: Tabellen A.30 und A.31

Netzwerk	**Init. LR**	**Vorverarbeitung**	**F1-Score SuPer**	**F1-Score TUI-Zuse**	**Recall Deep-Orien.**	**Recall SRL**
ResNeXt50	0,0001	Normalisierung	86,06	97,17	99,97	97,48
ResNeXt50	0,0001	Skalierung	82,74	97,35	99,99	97,17
ResNeXt50	0,0001	Skalierung	85,13	97,06	99,96	97,17
ResNeXt50	0,001	Normalisierung	87,34	96,95	99,93	91,6
ResNeXt50	0,001	Skalierung	86,41	96,99	99,95	97,45
ResNeXt50	0,001	Skalierung	87,64	96,7	99,95	96,01
ResNeXt50	0,01	Normalisierung	83,65	97,59	99,97	96,57
ResNeXt50	0,01	Skalierung	84,53	97,25	99,94	96,21

Tabelle A.25: Ergebnisse der Vorexperimente für die Personenklassifikation

Netzwerk	Init. LR	Vorverarbeitung	F1-Score SuPer	F1-Score TUI-Zuse	Recall Deep-Orien.	Recall SRL
ResNet18	0,000025	Normalisierung	76,11	96,07	99,7	85,58
ResNet18	0,000025	Normalisierung	79,64	95,56	99,74	93,91
ResNet18	0,000025	Normalisierung	78,1	96,22	99,76	90,43
ResNet18	0,00005	Normalisierung	78,8	96,11	99,87	93,22
ResNet18	0,00005	Normalisierung	77,69	96,48	99,78	88,72
ResNet18	0,00005	Normalisierung	78,87	96,37	99,86	86,0
ResNet18	0,0001	Normalisierung	77,85	96,8	99,93	93,04
ResNet18	0,0001	Normalisierung	79,19	95,93	99,81	88,46
ResNet18	0,0001	Normalisierung	80,18	96,14	99,79	93,51
ResNet18	0,00025	Normalisierung	80,41	96,46	99,89	91,85
ResNet18	0,00025	Normalisierung	79,44	96,46	99,86	91,17
ResNet18	0,00025	Normalisierung	81,62	95,3	99,58	80,98
ResNet18	0,0005	Normalisierung	81,93	96,05	99,74	91,02
ResNet18	0,0005	Normalisierung	78,76	97,07	99,91	90,57
ResNet18	0,0005	Normalisierung	81,49	96,51	99,78	90,78
ResNet18	0,001	Normalisierung	81,11	96,2	99,81	92,16
ResNet18	0,001	Normalisierung	79,76	96,71	99,91	92,81
ResNet18	0,001	Normalisierung	81,41	96,49	99,87	86,13
ResNet18	0,0025	Normalisierung	82,03	96,49	99,77	89,0
ResNet18	0,0025	Normalisierung	81,17	96,78	99,83	93,07
ResNet18	0,0025	Normalisierung	83,15	96,23	99,74	88,58
ResNet18	0,005	Normalisierung	81,67	96,99	99,86	94,8
ResNet18	0,005	Normalisierung	82,21	96,69	99,9	94,87
ResNet18	0,005	Normalisierung	81,05	96,27	99,66	89,66
ResNet18	0,01	Normalisierung	80,74	96,94	99,87	91,89
ResNet18	0,01	Normalisierung	83,51	95,78	99,74	78,13
ResNet18	0,01	Normalisierung	79,48	96,83	99,92	86,52
ResNet18	0,025	Normalisierung	81,46	96,66	99,91	84,69
ResNet18	0,025	Normalisierung	82,02	95,61	99,82	82,98
ResNet18	0,025	Normalisierung	84,04	96,45	99,87	81,46
ResNet18	0,05	Normalisierung	81,93	96,4	99,8	84,86
ResNet18	0,05	Normalisierung	79,58	96,53	99,73	82,8
ResNet18	0,05	Normalisierung	81,97	96,38	99,82	83,33
ResNet18	0,1	Normalisierung	74,5	95,57	98,88	62,5
ResNet18	0,1	Normalisierung	73,34	94,75	98,95	62,65
ResNet18	0,1	Normalisierung	72,35	93,75	95,15	45,92

Tabelle A.26: Ergebnisse der Vorexperimente für die Personenklassifikation *ResNet18, 400 Epochen, Normalisierung*

Netzwerk	Init. LR	Vorverarbeitung	F1-Score SuPer	F1-Score TUI-Zuse	Recall Deep-Orien.	Recall SRL
ResNet18	0,00005	Skalierung	80,17	94,4	99,47	94,6
ResNet18	0,00005	Skalierung	79,06	95,14	99,78	94,78
ResNet18	0,00005	Skalierung	78,84	95,62	99,85	95,83
ResNet18	0,0001	Skalierung	80,51	95,61	99,74	95,36
ResNet18	0,0001	Skalierung	78,9	95,66	99,85	95,9
ResNet18	0,0001	Skalierung	78,61	95,54	99,71	95,11
ResNet18	0,00025	Skalierung	79,31	96,26	99,83	96,31
ResNet18	0,00025	Skalierung	81,08	95,9	99,66	95,42
ResNet18	0,00025	Skalierung	81,08	96,09	99,81	95,52
ResNet18	0,0005	Skalierung	80,92	95,46	99,62	94,54
ResNet18	0,0005	Skalierung	79,47	96,46	99,87	96,5
ResNet18	0,0005	Skalierung	79,67	96,36	99,86	96,19
ResNet18	0,001	Skalierung	79,91	95,95	99,78	96,23
ResNet18	0,001	Skalierung	81,52	95,3	99,3	93,81
ResNet18	0,001	Skalierung	80,42	95,8	99,84	96,03
ResNet18	0,0025	Skalierung	82,22	96,03	99,81	95,85
ResNet18	0,0025	Skalierung	82,61	95,79	99,84	96,01
ResNet18	0,0025	Skalierung	79,55	96,71	99,9	96,89
ResNet18	0,005	Skalierung	82,71	96,08	99,77	94,57
ResNet18	0,005	Skalierung	81,77	96,29	99,71	95,04
ResNet18	0,005	Skalierung	80,73	96,01	99,83	95,62
ResNet18	0,01	Skalierung	80,92	96,39	99,82	94,47
ResNet18	0,01	Skalierung	80,13	96,55	99,92	96,36
ResNet18	0,01	Skalierung	83,09	95,93	99,7	93,81
ResNet18	0,025	Skalierung	81,69	95,46	99,91	94,81
ResNet18	0,025	Skalierung	80,23	96,81	99,99	97,48
ResNet18	0,025	Skalierung	82,87	95,62	99,87	95,14
ResNet18	0,05	Skalierung	80,42	95,65	99,75	91,44
ResNet18	0,05	Skalierung	82,86	95,8	99,89	94,54
ResNet18	0,05	Skalierung	81,29	95,49	99,91	93,29

Tabelle A.27: Ergebnisse der Vorexperimente für die Personenklassifikation *ResNet18, 400 Epochen, Skalierung*

Netzwerk	Init. LR	Vorverarbeitung	F1-Score SuPer	F1-Score TUI-Zuse	Recall Deep-Orien.	Recall SRL
ResNet18	0,000025	Normalisierung	56,39	94,98	98,52	82,81
ResNet18	0,000025	Normalisierung	59,96	94,52	98,86	92,12
ResNet18	0,000025	Normalisierung	62,51	93,4	98,78	90,11
ResNet18	0,00005	Normalisierung	65,08	94,93	99,41	93,1
ResNet18	0,00005	Normalisierung	65,21	94,45	99,07	82,01
ResNet18	0,00005	Normalisierung	63,15	95,25	99,2	81,8
ResNet18	0,0001	Normalisierung	68,54	95,51	99,52	86,05
ResNet18	0,0001	Normalisierung	70,25	95,9	99,79	86,76
ResNet18	0,0001	Normalisierung	70,56	95,08	99,67	92,06
ResNet18	0,00025	Normalisierung	76,99	95,85	99,85	91,3
ResNet18	0,00025	Normalisierung	74,05	95,92	99,8	91,05
ResNet18	0,00025	Normalisierung	77,73	95,21	99,62	80,63
ResNet18	0,0005	Normalisierung	76,84	96,58	99,82	93,15
ResNet18	0,0005	Normalisierung	78,38	96,01	99,83	83,84
ResNet18	0,0005	Normalisierung	75,67	96,23	99,74	90,07
ResNet18	0,001	Normalisierung	79,82	95,76	99,79	92,0
ResNet18	0,001	Normalisierung	80,41	95,7	99,71	87,34
ResNet18	0,001	Normalisierung	79,73	96,04	99,81	83,96
ResNet18	0,0025	Normalisierung	80,86	95,99	99,59	86,35
ResNet18	0,0025	Normalisierung	77,89	96,98	99,88	93,32
ResNet18	0,0025	Normalisierung	80,22	96,78	99,85	92,0
ResNet18	0,005	Normalisierung	79,8	96,96	99,86	94,34
ResNet18	0,005	Normalisierung	80,88	96,64	99,81	93,49
ResNet18	0,005	Normalisierung	78,33	96,84	99,82	92,1
ResNet18	0,01	Normalisierung	81,8	95,97	99,72	88,13
ResNet18	0,01	Normalisierung	82,27	95,9	99,78	80,96
ResNet18	0,01	Normalisierung	79,48	96,83	99,92	86,52
ResNet18	0,025	Normalisierung	83,69	95,76	99,82	81,73
ResNet18	0,025	Normalisierung	82,02	95,61	99,82	82,98
ResNet18	0,025	Normalisierung	78,85	97,1	99,79	84,99
ResNet18	0,05	Normalisierung	82,57	96,2	99,81	87,44
ResNet18	0,05	Normalisierung	80,2	95,07	99,38	69,55
ResNet18	0,05	Normalisierung	81,66	96,01	99,45	78,11
ResNet18	0,1	Normalisierung	75,22	95,17	97,45	50,86
ResNet18	0,1	Normalisierung	65,53	94,01	98,38	59,71
ResNet18	0,1	Normalisierung	64,62	91,86	94,97	51,92

Tabelle A.28: Ergebnisse der Vorexperimente für die Personenklassifikation *ResNet18, 15 Epochen, Normalisierung*

Netzwerk	Init. LR	Vorverarbeitung	F1-Score SuPer	F1-Score TUI-Zuse	Recall Deep-Orien.	Recall SRL
ResNet18	0,00005	Skalierung	64,37	94,8	99,85	94,5
ResNet18	0,00005	Skalierung	65,84	93,94	99,78	91,79
ResNet18	0,00005	Skalierung	64,95	95,1	99,9	95,32
ResNet18	0,0001	Skalierung	72,55	94,6	99,63	93,93
ResNet18	0,0001	Skalierung	71,48	94,81	99,87	93,66
ResNet18	0,0001	Skalierung	70,06	94,52	99,62	93,57
ResNet18	0,00025	Skalierung	76,93	95,88	99,81	95,78
ResNet18	0,00025	Skalierung	78,24	95,44	99,7	95,69
ResNet18	0,00025	Skalierung	76,66	94,96	99,67	93,82
ResNet18	0,0005	Skalierung	80,05	94,03	99,45	93,8
ResNet18	0,0005	Skalierung	79,96	95,42	99,76	95,74
ResNet18	0,0005	Skalierung	78,91	95,61	99,75	94,91
ResNet18	0,001	Skalierung	78,29	95,31	99,6	95,65
ResNet18	0,001	Skalierung	79,44	95,37	99,34	93,75
ResNet18	0,001	Skalierung	79,8	95,33	99,74	95,68
ResNet18	0,0025	Skalierung	82,07	95,7	99,75	94,62
ResNet18	0,0025	Skalierung	79,18	96,54	99,92	96,94
ResNet18	0,0025	Skalierung	79,31	96,19	99,83	96,22
ResNet18	0,005	Skalierung	79,91	96,51	99,89	96,03
ResNet18	0,005	Skalierung	81,52	96,0	99,63	93,82
ResNet18	0,005	Skalierung	80,27	95,85	99,79	95,35
ResNet18	0,01	Skalierung	79,51	96,45	99,84	94,33
ResNet18	0,01	Skalierung	77,09	96,94	99,95	97,04
ResNet18	0,01	Skalierung	82,22	96,16	99,75	93,84
ResNet18	0,025	Skalierung	81,69	95,46	99,91	94,81
ResNet18	0,025	Skalierung	83,81	95,77	99,89	95,18
ResNet18	0,025	Skalierung	82,0	95,72	99,88	95,32
ResNet18	0,05	Skalierung	80,42	95,65	99,75	91,44
ResNet18	0,05	Skalierung	83,55	95,11	99,76	92,21
ResNet18	0,05	Skalierung	80,95	95,06	99,94	90,85

Tabelle A.29: Ergebnisse der Vorexperimente für die Personenklassifikation *ResNet18, 15 Epochen, Skalierung*

Netzwerk	Init. LR	Vorverarbeitung	F1-Score SuPer	F1-Score TUI-Zuse	Recall Deep-Orien.	Recall SRL
BeyerModRelu	0,000025	Normalisierung	52,68	91,83	98,98	79,17
BeyerModRelu	0,000025	Normalisierung	57,28	92,65	98,01	90,94
BeyerModRelu	0,000025	Normalisierung	59,02	92,3	98,37	89,02
BeyerModRelu	0,000025	Skalierung	65,22	93,76	98,97	93,21
BeyerModRelu	0,000025	Skalierung	62,11	93,73	99,4	95,03
BeyerModRelu	0,000025	Skalierung	62,69	93,15	98,05	95,0
BeyerModRelu	0,00005	Normalisierung	63,6	93,26	99,0	92,14
BeyerModRelu	0,00005	Normalisierung	61,4	93,51	99,41	89,53
BeyerModRelu	0,00005	Normalisierung	65,35	92,65	98,86	76,74
BeyerModRelu	0,00005	Skalierung	65,31	93,65	98,25	92,52
BeyerModRelu	0,00005	Skalierung	66,75	92,7	98,37	91,92
BeyerModRelu	0,00005	Skalierung	66,62	93,72	98,81	93,6
BeyerModRelu	0,0001	Normalisierung	70,47	92,05	99,02	74,75
BeyerModRelu	0,0001	Normalisierung	69,46	93,6	98,76	84,05
BeyerModRelu	0,0001	Normalisierung	69,63	93,66	99,25	80,81
BeyerModRelu	0,0001	Skalierung	69,4	93,1	98,65	95,2
BeyerModRelu	0,0001	Skalierung	69,83	93,86	98,92	93,96
BeyerModRelu	0,0001	Skalierung	71,04	93,17	98,22	93,03
BeyerModRelu	0,00025	Normalisierung	71,79	92,98	99,06	78,59
BeyerModRelu	0,00025	Normalisierung	68,91	94,8	99,38	74,51
BeyerModRelu	0,00025	Normalisierung	69,47	93,99	98,55	67,27
BeyerModRelu	0,00025	Skalierung	73,3	93,68	98,52	93,91
BeyerModRelu	0,00025	Skalierung	70,66	94,23	98,6	94,11
BeyerModRelu	0,00025	Skalierung	70,47	94,52	99,07	95,23
BeyerModRelu	0,0005	Normalisierung	69,51	94,26	98,99	62,7
BeyerModRelu	0,0005	Normalisierung	69,61	94,17	98,69	66,53
BeyerModRelu	0,0005	Normalisierung	70,63	94,37	99,35	82,05
BeyerModRelu	0,0005	Skalierung	73,14	93,99	98,79	92,35
BeyerModRelu	0,0005	Skalierung	71,89	93,14	98,39	90,25
BeyerModRelu	0,0005	Skalierung	74,46	93,26	98,5	91,19
BeyerModRelu	0,001	Normalisierung	73,84	93,43	98,34	68,78
BeyerModRelu	0,001	Normalisierung	73,26	94,19	98,79	62,94
BeyerModRelu	0,001	Normalisierung	73,43	93,87	98,61	65,44
BeyerModRelu	0,001	Skalierung	72,96	94,42	98,96	93,65
BeyerModRelu	0,001	Skalierung	74,48	93,56	98,9	92,98
BeyerModRelu	0,001	Skalierung	67,69	95,22	98,79	92,63

Tabelle A.30: Ergebnisse der Vorexperimente für die Personenklassifikation *BeyerModRelu, 400 Epochen Lernraten 0,000025 bis 0,001*

Netzwerk	Init. LR	Vorverarbeitung	F1-Score SuPer	F1-Score TUI-Zuse	Recall Deep-Orien.	Recall SRL
BeyerModRelu	0,0025	Normalisierung	71,46	94,76	98,61	79,98
BeyerModRelu	0,0025	Normalisierung	74,09	93,89	98,18	55,37
BeyerModRelu	0,0025	Normalisierung	73,32	94,56	99,21	60,68
BeyerModRelu	0,0025	Skalierung	75,06	94,6	98,74	92,24
BeyerModRelu	0,0025	Skalierung	73,53	94,49	99,01	93,85
BeyerModRelu	0,0025	Skalierung	76,21	92,83	97,93	86,71
BeyerModRelu	0,005	Normalisierung	74,12	94,51	98,53	59,65
BeyerModRelu	0,005	Normalisierung	74,14	94,26	98,86	65,18
BeyerModRelu	0,005	Normalisierung	74,38	94,22	98,34	64,54
BeyerModRelu	0,005	Skalierung	75,9	94,22	98,65	89,29
BeyerModRelu	0,005	Skalierung	74,57	94,68	99,24	89,63
BeyerModRelu	0,005	Skalierung	73,79	94,52	98,59	89,36
BeyerModRelu	0,01	Normalisierung	78,06	93,67	98,54	66,57
BeyerModRelu	0,01	Normalisierung	75,28	95,0	98,81	64,6
BeyerModRelu	0,01	Normalisierung	75,65	94,17	97,51	62,85
BeyerModRelu	0,01	Skalierung	73,84	96,13	99,3	89,63
BeyerModRelu	0,01	Skalierung	75,45	95,1	99,24	90,8
BeyerModRelu	0,01	Skalierung	78,62	94,19	99,12	86,83
BeyerModRelu	0,025	Normalisierung	75,93	94,68	98,81	68,76
BeyerModRelu	0,025	Normalisierung	77,22	93,92	97,71	70,52
BeyerModRelu	0,025	Normalisierung	75,53	94,82	97,98	71,54
BeyerModRelu	0,025	Skalierung	76,09	95,07	99,46	86,98
BeyerModRelu	0,025	Skalierung	75,26	95,78	99,7	89,12
BeyerModRelu	0,025	Skalierung	77,23	95,3	99,45	88,35
BeyerModRelu	0,05	Normalisierung	75,81	95,2	98,6	57,01
BeyerModRelu	0,05	Normalisierung	75,2	94,6	98,77	73,27
BeyerModRelu	0,05	Normalisierung	74,87	94,07	98,05	59,03
BeyerModRelu	0,05	Skalierung	77,47	94,87	99,42	89,51
BeyerModRelu	0,05	Skalierung	78,32	94,01	98,79	85,93
BeyerModRelu	0,05	Skalierung	76,37	94,46	99,11	82,34
BeyerModRelu	0,1	Normalisierung	73,23	93,89	98,25	59,45
BeyerModRelu	0,1	Normalisierung	73,06	94,79	97,97	71,91
BeyerModRelu	0,1	Normalisierung	74,38	94,27	98,53	61,5
BeyerModRelu	0,1	Skalierung	73,11	95,75	99,58	89,59
BeyerModRelu	0,1	Skalierung	74,65	95,89	99,63	89,85
BeyerModRelu	0,1	Skalierung	73,72	95,84	99,51	86,72

Tabelle A.31: Ergebnisse der Vorexperimente für die Personenklassifikation *BeyerModRelu, 400 Epochen Lernraten 0,0025 bis 0,1*

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Abu-Mostafa, 1990] Abu-Mostafa, Y. S. (1990). *Learning from Hints in Neural Networks*. Journal of Complexity., 6(2):192–198. *(Seite: 20)*

[Argyriou et al., 2008] Argyriou, Andreas, T. Evgeniou und M. Pontil (2008). *Convex multi-task feature learning*. Machine Learning, 73(3):243–272. *(Seite: 23)*

[Benenson et al., 2012] Benenson, R., M. Mathias, R. Timofte und L. Van Gool (2012). *Pedestrian detection at 100 frames per second*. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 2903–2910. *(Seite: 96)*

[Beyer et al., 2015] Beyer, Lucas, A. Hermans und B. Leibe (2015). *Biternion Nets: Continuous Head Pose Regression from Discrete Training Labels*. In: *Pattern Recognition*, Bd. 9358 d. Reihe *Lecture Notes in Computer Science*, S. 157–168. Springer. *(Seite: 10, 11, 52)*

[Bornstein, 2018] Bornstein, Aaron (2018). *Silver, Gold and Electrum: 3 Data Techniques for Multi-Task Deep Learning*. `https://towardsdatascience.com/silver-gold-electrum-3-data-techniques-for-multi-task-deep-learning-2655004970a2`. Abgerufen am 22.08.2019. *(Seite: 34)*

[Cao et al., 2017] Cao, Zhe, T. Simon, S.-E. Wei und Y. Sheikh (2017). *Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. *(Seite: 96, 106)*

[Caruana, 1997] Caruana, Rich (1997). *Multitask Learning*. Mach. Learn., 28(1):41–75. *(Seite: 1, 18)*

[Chen et al., 2017] Chen, Zhao, V. Badrinarayanan, C. Lee und A. Rabinovich (2017). *GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks*. CoRR, abs/1711.02257. *(Seite: 32, 33, 60, 116)*

[Dollar et al., 2012] Dollar, P., C. Wojek, B. Schiele und P. Perona (2012). *Pedestrian Detection: An Evaluation of the State of the Art*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 34(4):743–761. *(Seite: 13, 94)*

[Evgeniou et al., 2005] Evgeniou, Theodoros, C. A. Micchelli und M. Pontil (2005). *Learning Multiple Tasks with Kernel Methods*. Journal of Machine Learning Research, 6:615–637. *(Seite: 23)*

[Felzenszwalb et al., 2010] Felzenszwalb, P. F., R. B. Girshick und D. McAllester (2010). *Cascade object detection with deformable part models*. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 2241–2248. *(Seite: 96)*

[Gross, 2018] Gross, H.-M. (2018). *Angewandte Neuroinformatik*. Vorlesungsskript Sommersemester 2018. Fachgebiet Neuroinformatik und Kognitive Robotik, TU Ilmenau. *(Seite: 60)*

[Gross, 2019] Gross, H.-M. (2019). *Neuroinformatik*. Vorlesungsskript Sommersemester 2019. Fachgebiet Neuroinformatik und Kognitive Robotik, TU Ilmenau. *(Seite: 57)*

[Guo et al., 2018] Guo, Michelle, A. Haque, D.-A. Huang, S. Yeung und L. Fei-Fei (2018). *Dynamic Task Prioritization for Multitask Learning*. In: *European Conference on Computer Vision (ECCV). (Seite: 105)*

[He et al., 2016] He, Kaiming, X. Zhang, S. Ren und J. Sun (2016). *Deep Residual Learning for Image Recognition*. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 770–778. *(Seite: 7, 9, 21, 26, 51, 67, 70, 94, 106)*

[Ilmenau, 2019] Ilmenau, Technische Universität (2019). *Konrad und Suse*. `https://www.tu-ilmenau.de/fileadmin/media/neurob/research/robots/konrad_suse`. Abgerufen am 22.12.2019. *(Seite: 144)*

[Ioffe und Szegedy, 2015] Ioffe, Sergey und C. Szegedy (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. In: *International Conference on Machine Learning (ICML)*, S. 448–456. JMLR.org. *(Seite: 8)*

[Jafari et al., 2014] Jafari, Omid Hosseini, D. Mitzel und B. Leibe (2014). *Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras*. International Conference on Robotics and Automation (ICRA), S. 5636–5643. *(Seite: 3, 42, 43, 96)*

[Jiang et al., 2019] Jiang, Angela H., D. L. K. Wong, G. Zhou, D. G. Andersen, J. Dean, G. R. Ganger, G. Joshi, M. Kaminksy, M. Kozuch, Z. C. Lipton und P. Pillai (2019). *Accelerating Deep Learning by Focusing on the Biggest Losers*. arXiv preprint arXiv:1910.00762. *(Seite: 105)*

[Joo et al., 2017] Joo, Hanbyul, T. Simon, X. Li, H. Liu, L. Tan, L. Gui, S. Banerjee, T. S. Godisart, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara und Y. Sheikh (2017). *Panoptic Studio: A Massively Multiview System for Social Interaction Capture*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI). *(Seite: 96)*

[Kokkinos, 2017] Kokkinos, Iasonas (2017). *UberNet: Training a 'Universal' Convolutional Neural Network for Low-, Mid-, and High-Level Vision using Diverse Datasets and Limited Memory*. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (Seite: 1, 2, 35)*

[Krizhevsky et al., 2012] Krizhevsky, Alex, I. Sutskever und G. E. Hinton (2012). *ImageNet Classification with Deep Convolutional Neural Networks*. In: Pereira, F., C. J. C. Burges, L. Bottou und K. Q. Weinberger, Hrsg.: *Advances in Neural Information Processing Systems 25*, S. 1097–1105. Curran Associates, Inc. *(Seite: 26)*

[Lawin et al., 2016] Lawin, Felix Järemo, P.-E. Forssén und H. Ovrén (2016). *Efficient multi-frequency phase unwrapping using kernel density estimation*. In: *European Conference on Computer Vision (ECCV)*, S. 170–185. Springer. *(Seite: 44)*

[Lewandowski et al., 2019a] Lewandowski, Benjamin, J. Liebner, T. Wengefeld, S. Müller und H.-M. Gross (2019a). *A Fast and Robust 3D Person Detector and Posture Estimator for Mobile Robotic Applications.* International Conference on Robotics and Automation (ICRA). *(Seite: 4, 41, 44, 45, 66, 94, 96)*

[Lewandowski et al., 2019b] Lewandowski, Benjamin, D. Seichter, T. Wengefeld, L. Pfennig, H. Drumm und H.-M. Gross (2019b). *Deep Orientation: Fast and Robust Upper Body Orientation Estimation for Mobile Robotic Applications.* International Conference on Intelligent Robots & Systems (IROS). *(Seite: 2, 4, 8, 34, 41, 42, 44, 48, 50, 52, 61, 64, 66, 67, 69, 70, 72, 91, 94)*

[Linder et al., 2015] Linder, T., S. Wehner und K. O. Arras (2015). *Real-time full-body human gender recognition in (RGB)-D data.* In: *International Conference on Robotics and Automation (ICRA)*, S. 3039–3045. *(Seite: 4, 44, 47)*

[Long und Wang, 2015] Long, Mingsheng und J. Wang (2015). *Learning Multiple Tasks with Deep Relationship Networks.* CoRR, abs/1506.02117. *(Seite: 20, 22)*

[Lu et al., 2017] Lu, Yongxi, A. Kumar, S. Zhai, Y. Cheng, T. Javidi und R. Feris (2017). *Fully-Adaptive Feature Sharing in Multi-Task Networks with Applications in Person Attribute Classification.* In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 1131–1140. *(Seite: 30, 32, 33, 35, 60, 115, 116)*

[Misra et al., 2016] Misra, I., A. Shrivastava, A. Gupta und M. Hebert (2016). *Cross-Stitch Networks for Multi-task Learning.* In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 3994–4003. *(Seite: 19, 28, 29, 30, 151)*

[Ranjan et al., 2019] Ranjan, R., V. M. Patel und R. Chellappa (2019). *HyperFace: A Deep Multi-Task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 41(1):121–135. *(Seite: 25, 26, 27, 52, 55, 151)*

[Ren et al., 2015] Ren, Shaoqing, K. He, R. Girshick und J. Sun (2015). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* In:

Cortes, C., N. D. Lawrence, D. D. Lee, M. Sugiyama und R. Garnett, Hrsg.: *Advances in Neural Information Processing Systems 28*, S. 91–99. Curran Associates, Inc. *(Seite: 106)*

[Ruder, 2017] Ruder, Sebastian (2017). *An Overview of Multi-Task Learning in Deep Neural Networks*. CoRR, abs/1706.05098. *(Seite: 16, 17, 18, 19, 20, 21, 22, 24, 31)*

[Ruder et al., 2017] Ruder, Sebastian, J. Bingel, I. Augenstein und A. Søgaard (2017). *Sluice networks: Learning what to share between loosely related tasks*. arXiv preprint arXiv:1705.08142. *(Seite: 28, 31, 151)*

[Simonyan und Zisserman, 2015] Simonyan, Karen und A. Zisserman (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. In: *International Conference on Learning Representations (ICLR). (Seite: 7, 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, 15:1929–1958. *(Seite: 8)*

[Tan und Le, 2019] Tan, Mingxing und Q. V. Le (2019). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. CoRR, abs/1905.11946. *(Seite: 106)*

[Xie et al., 2017] Xie, Saining, R. Girshick, P. Dollar, Z. Tu und K. He (2017). *Aggregated Residual Transformations for Deep Neural Networks*. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 5987–5995. *(Seite: 9, 94)*

[Zhang et al., 2014] Zhang, Zhanpeng, P. Luo, C. C. Loy und X. Tang (2014). *Facial Landmark Detection by Deep Multi-task Learning*. In: Fleet, David, T. Pajdla, B. Schiele und T. Tuytelaars, Hrsg.: *European Conference on Computer Vision (ECCV)*, S. 94–108, Cham. Springer International Publishing. *(Seite: 18, 25, 26, 37, 38, 104, 151)*