



# **Entwicklung eines Verfahrens zur Multi Objekt Verfolgung auf Basis von maschinellem Lernen (Multi Object Tracking using Machine Learning)**

**Masterarbeit**

für die Prüfung zum  
**Master of Engineering (M. Eng.)**  
im Studiengang  
**Integrated Engineering**  
  
am Center for Advanced Studies  
der Dualen Hochschule Baden-Württemberg

Vorname und Nachname:  
Matrikelnummer:  
Bearbeitungszeitraum:  
Dualer Partner:  
Erstgutachter/-in:  
Zweitgutachter/-in:

Christian Alexander Holz  
8758851  
24.02.2023 - 23.08.2023  
Daimler Truck AG, Stuttgart  
Christian Bader  
Prof. Dr. rer. nat. Matthias Drüppel

Eingangsvermerk DHBW CAS

## **Sperrvermerk**

Die vorliegende Masterarbeit mit dem Titel

**Entwicklung eines Verfahrens zur Multi Objekt Verfolgung auf Basis von  
maschinellem Lernen (Multi Object Tracking using Machine Learning)**

enthält interne vertrauliche Daten der Daimler Truck AG.

Sie ist nur den Erst- und Zweitgutachtern sowie ggf. dem Prüfungsausschussvorsitzenden des Fachbereiches Technik zugänglich zu machen. Veröffentlichungen und Vervielfältigungen der Masterarbeit oder die Weitergabe der Masterarbeit im Gesamten oder in Teilen sowie das Anfertigen von Kopien oder Abschriften -auch in digitaler Form- sind grundsätzlich untersagt. Ausnahmen bedürfen der vorherigen schriftlichen Genehmigung der Daimler Truck AG.

## **Erklärung**

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Aus fremden Quellen direkt oder indirekt übernommene Gedanken habe ich als solche kenntlich gemacht. Die Arbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher auch nicht veröffentlicht.

Ich erkläre weiterhin, dass ich ein unverschlüsseltes digitales Textdokument der Arbeit (in einem der Formate .doc, .docx, .odt, .pdf, .rtf) in Moodle hochgeladen habe. Alle digital eingereichten Exemplare entsprechen in Inhalt und Wortlaut ausnahmslos der gedruckten Ausfertigung.

---

Stuttgart, 23. August 2023

Ort, Datum



---

Unterschrift

## Danksagung

Zuallererst möchte ich meinen Betreuer, Christian Bader, meinen tiefsten Dank aussprechen. Seine konstante Unterstützung, geduldige Anleitung und wertvollen Ratschläge haben maßgeblich zu der Qualität dieser Arbeit beigetragen.

Ein besonderer Dank gilt auch Prof. Dr. rer. nat. Matthias Drüppel für die wissenschaftliche Unterstützung und hilfreichen Diskussionen während der Entwicklung des Verfahrens. Seine Expertise war von unschätzbarem Wert.

Ich bin für die Zeit und das Engagement, die meine Betreuer in diese Arbeit investiert haben, zutiefst dankbar.

Ich bin zudem dem gesamten Advanced Driver Assistance System ([ADAS](#)) Bereich der Daimler Truck AG dankbar, das eine inspirierende Arbeitsumgebung geschaffen hat. Mein Dank gilt insbesondere Carsten Barth und Michael Berner für ihre wertvollen Feedbacks, gegebenen Freiheiten und anregenden Gespräche.

Des Weiteren danke ich der Dualen Hochschule Baden-Württemberg, die diese Arbeit ermöglicht hat.

Ein herzlicher Dank geht auch an meine Verlobte, Victoria. Ohne ihre anhaltende Unterstützung und Ermutigung wäre diese Arbeit nicht möglich gewesen. Schließlich möchte ich mich bei meinen Freunden bedanken, insbesondere bei Michael, der während dieser Zeit stets ein offenes Ohr für mich hatte und mir in stressigen Phasen zur Seite stand.

Diese Arbeit wäre ohne die Hilfe und Unterstützung all dieser Menschen nicht möglich gewesen und ich bin jedem Einzelnen von ihnen zutiefst dankbar.

# Inhaltsverzeichnis

<b>Abkürzungsverzeichnis</b>	<b>I</b>
<b>Abbildungsverzeichnis</b>	<b>III</b>
<b>Tabellenverzeichnis</b>	<b>V</b>
<b>1 Einführung</b>	<b>1</b>
1.1 Motivation und Problemstellung . . . . .	2
1.2 Zielsetzung . . . . .	3
1.3 Gliederung der Arbeit . . . . .	4
<b>2 Theoretische Grundlagen</b>	<b>6</b>
2.1 Filter- und Trackingverfahren . . . . .	8
2.1.1 Kalman Filter ( <b>KF</b> ) . . . . .	8
2.1.2 Datenassoziation . . . . .	12
2.2 Einführung in das maschinelle Lernen / Machine Learning ( <b>ML</b> ) . . . . .	15
2.2.1 Deep Learning ( <b>DL</b> ) und Neural Networks ( <b>NNs</b> ) . . . . .	16
2.2.2 Netzwerktraining von Feed-Forward Networks ( <b>FFNs</b> ) . . . . .	19
2.2.3 Recurrent Neural Networks ( <b>RNNs</b> ) . . . . .	20
2.3 Objekt Verfolgung / Single Object Tracking ( <b>SOT</b> ) . . . . .	24
2.3.1 <b>SOT</b> als Assoziationsproblem . . . . .	25
2.3.2 <b>SOT</b> als Problem der zeitlichen Vorhersage . . . . .	26
2.3.3 Erkenntnisgewinne . . . . .	27
2.4 Multi Objekt Verfolgung / Multi Object Tracking ( <b>MOT</b> ) . . . . .	28
2.4.1 <b>MOT</b> Grundlagen . . . . .	28
2.4.2 <b>MOT</b> Onlineverfahren . . . . .	30
2.4.3 Erkenntnisgewinne . . . . .	32
<b>3 Ergebnisse</b>	<b>33</b>
3.1 Vorgehensweise und Anforderungen . . . . .	33
3.2 KITTI Datensatz . . . . .	36
3.3 Single Prediction Network ( <b>SPENT</b> ) - Schätzung der Zustandsänderung erfasster Tracks . . . . .	38
3.3.1 Datenvorverarbeitung . . . . .	39
3.3.2 Netzwerkentwicklung . . . . .	42
3.3.3 Optimierungen . . . . .	47
3.3.4 Zusammenfassung . . . . .	56

3.4	Single Association Network ( <b>SANT</b> ) - Assoziation einer Messung zu erfassten Tracks . . . . .	58
3.4.1	Datenvorverarbeitung . . . . .	59
3.4.2	Netzwerkentwicklung . . . . .	61
3.4.3	Optimierungen . . . . .	64
3.4.4	Zusammenfassung . . . . .	68
3.5	Multi Association Network ( <b>MANTa</b> ) - Assoziation mehrerer Messungen zu erfassten Tracks . . . . .	70
3.5.1	Datenvorverarbeitung . . . . .	70
3.5.2	Netzwerkentwicklung . . . . .	72
3.5.3	Zusammenfassung . . . . .	74
3.6	Evaluierung des Multi Object Tracking ( <b>MOT</b> ) Verfahrens . . . . .	75
3.6.1	Optimal Sub-Pattern Assignment ( <b>OSPA</b> ) Metrik . . . . .	75
3.6.2	<b>MOT</b> Framework - Erläuterung der Funktionsweise . . . . .	76
3.6.3	Vergleich der Vorhersagemodelle im <b>MOT</b> Framework . . . . .	78
3.6.4	Vergleich Assoziationsmodelle im <b>MOT</b> Framework . . . . .	82
<b>4</b>	<b>Fazit</b>	<b>85</b>
4.1	Zusammenfassung . . . . .	85
4.2	Ausblick . . . . .	86
<b>Literatur</b>		<b>90</b>
<b>Anhang</b>		<b>93</b>

# Abkürzungsverzeichnis

<b>ADAM</b>	Adaptive Moment Estimation
<b>ADAS</b>	Advanced Driver Assistance System
<b>AI</b>	Artificial Intelligence
<b>BILSTM</b>	Bidirectional Long Short-Term Memory Netzwerk
<b>BNN</b>	Bayesian Neural Network
<b>BPTT</b>	Backpropagation Through Time
<b>CNN</b>	Convolutional Neural Network
<b>CV</b>	Computer Vision
<b>DA</b>	Data Association
<b>DGM</b>	Deep Generative Modeling
<b>DL</b>	Deep Learning
<b>DNN</b>	Deep Neural Network
<b>DT</b>	Daimler Truck
<b>ECU</b>	Electronic Control Unit
<b>EKF</b>	Extended Kalman Filter
<b>FC</b>	Fully Connected
<b>FFN</b>	Feed-Forward Network
<b>FN</b>	False Negative
<b>FoV</b>	Field of View
<b>FP</b>	False Positive
<b>GNN</b>	Global Nearest Neighbor
<b>GPS</b>	Global Positioning System
<b>GRU</b>	Gated Recurrent Unit
<b>GT</b>	Groundtruth
<b>HA</b>	Hungarian Algorithm
<b>HOTA</b>	Higher Order Tracking Accuracy
<b>IMU</b>	Inertial Measurement Unit
<b>IOU</b>	Intersection Over Union
<b>JPDA</b>	Joint Probabilistic Data Association
<b>KF</b>	Kalman Filter
<b>KIT</b>	Karlsruher Institut für Technologie
<b>LSTM</b>	Long Short-Term Memory
<b>MANTa</b>	Multi Association Network
<b>ML</b>	Machine Learning

<b>MOT</b>	Multi Object Tracking
<b>MOTA</b>	Multiple Object Tracking Accuracy
<b>MSE</b>	Mean Squared Error
<b>NN</b>	Neural Network
<b>NP</b>	Non Deterministic Polynomial Time
<b>OSPA</b>	Optimal Sub-Pattern Assignment
<b>PDA</b>	Probabilistic Data Association
<b>PKW</b>	Personen Kraft Wagen
<b>PO</b>	Punkt Objekt
<b>ReLU</b>	Rectified Linear Unit
<b>RMSE</b>	Root Mean Squared Error
<b>RNN</b>	Recurrent Neural Network
<b>ROLO</b>	Recurrent YOLO
<b>SANT</b>	Single Association Network
<b>SGD</b>	Stochastic Gradient Descent
<b>SO</b>	Sensor Objekt
<b>SoDa</b>	Soft Data Association
<b>SOT</b>	Single Object Tracking
<b>SPENT</b>	Single Prediction Network
<b>SRU</b>	Simple Recurrent Unit
<b>TbA</b>	Tracking by Attention
<b>TbD</b>	Tracking by Detection
<b>TbR</b>	Tracking by Regression
<b>TP</b>	True Positive
<b>TTIC</b>	Toyota Technological Institute at Chicago
<b>TUM</b>	Technische Universität München
<b>UKF</b>	Unscented Kalman Filter

# Abbildungsverzeichnis

2.1	Schematische Darstellung, Berechnungsschritte des Kalman Filter . . . . .	9
2.2	Schematische Darstellung eines Kalman Filterprozesses mit Track Management und Datenassoziation . . . . .	11
2.3	Schematische Darstellung der Assoziation detektierter Objekte . . . . .	12
2.4	Terminologie Mengendiagramm - Artificial Intelligence ( <b>AI</b> ), Machine Learning ( <b>ML</b> ) und Deep Learning ( <b>DL</b> ) . . . . .	15
2.5	Schematische Darstellung, künstliches Neuron . . . . .	16
2.6	Schematische Darstellung, Single Layer und Deep Neural Network . . . . .	17
2.7	Schematische Darstellung, Neuron mit Recurrence (Rückkopplung) . . . . .	21
2.8	Long Short-Term Memory ( <b>LSTM</b> ) Architektur, Gatter zur Steuerung des Informationsflusses . . . . .	22
2.9	Schematisch dargestellte Architektur GOTURN Modell . . . . .	25
2.10	Assoziationsprinzip des Verfahrens . . . . .	26
2.11	Architekturvorschlag der ROLO Modell Entwickler . . . . .	26
2.12	Kategorisierung einiger <b>MOT</b> Verfahren . . . . .	31
3.1	Schematische Darstellung der Architektur des genutzten <b>MOT</b> Frameworks . . . . .	33
3.2	Schematische Darstellung, <b>SPENT</b> Funktionsgebiet rot markiert . . . . .	38
3.3	3D-Plot, drei extrahierte Tracks des Gesamtdatensatzes . . . . .	39
3.4	KITTI, grafische Darstellung der Datensatzaufteilung . . . . .	40
3.5	<b>SPENT</b> po Analysedarstellung der Netzwerkarchitektur . . . . .	42
3.6	<b>LSTM</b> Netzwerktraining, Mini-Batch-Padding Effekt . . . . .	44
3.7	<b>SPENT</b> po, grafische Bewertung der Testtracks . . . . .	46
3.8	Testtrackbewertung, Sequenz: 7 / ID: 3 (li.), <b>SPENT</b> po (li.), <b>SPENT</b> dt (re.)	49
3.9	<b>SPENT</b> dt, Ergebnisplot, Trainingsprozess . . . . .	51
3.10	<b>SPENT</b> , Ergebnistabelle . . . . .	52
3.11	Single Prediction Network ( <b>SPENT</b> ), Input- Output Struktur . . . . .	56
3.12	Schematische Darstellung, <b>SANT</b> Funktionsgebiet rot markiert . . . . .	58
3.13	<b>SANT</b> , Struktur Inputdaten . . . . .	60
3.14	<b>SANT</b> , Variante LSTM1FC1, Analysedarstellung . . . . .	61
3.15	<b>SANT</b> , Variante LSTM1FC1, Confusion Matrix . . . . .	63
3.16	<b>SANT</b> , Ergebnistabelle, Auszug der untersuchten Netzwerkvarianten .	65
3.17	<b>SANT</b> , BilstmSeqLastFc1, Netzwerkanalyse . . . . .	65

3.18	SANT, BilstmSeqLastFc1, Confusionmatrix . . . . .	66
3.19	SANT, Untersuchung der Rauschintensität, Übersicht der Ergebnisse . .	67
3.20	Single Association Network ( <b>SANT</b> ), Input- Output Struktur . . . . .	68
3.21	MANTa, Datenstruktur, Input / Output . . . . .	71
3.22	MANTa, Schematische Darstellung der Netzwerkarchitektur . . . . .	72
3.23	MANTa, Verteilung der verfügbaren Daten, Trackanzahl pro Sample . .	73
3.24	Schematische Darstellung der Architektur des genutzten MOT Frameworks . . . . .	76
3.25	Schematische Darstellung, MOT Framework, KF und SPENT Integration	78
3.26	Vergleichsdiagramm OSPA, KITTI Datensatz, KF und SPENT, Global Nearest Neighbor (GNN) . . . . .	80
3.27	Vergleichsdiagramm, KF und SPENT Tracks, Sequenz 6 . . . . .	81
3.28	Schematische Darstellung, MOT Framework mit SANT Integration . . .	82
3.29	Vergleichsdiagramm OSPA, KITTI Datensatz, KF und SPENT, SANT . .	83

# Tabellenverzeichnis

2.1	SOT Verfahren, Zusammenfassung . . . . .	27
3.1	SPENTpo, Ergebnistabelle . . . . .	45
3.2	SPENT, Ergebnistabelle, Erweiterung der Zustandswerte . . . . .	48
3.3	SPENT, Ergebnistabelle Netzwerkevolution . . . . .	55
3.4	Ergebnistabelle, OSPA, KITTI Datensatz, MOT Framework . . . . .	84

# 1 Einführung

Advanced Driver Assistance System (**ADAS**) bzw. aktive Sicherheitssysteme für Fahrzeuge werden entwickelt, um den Fahrer bei der Fahraufgabe zu unterstützen. Das Ziel der Entwicklung von Fahrerassistenzsystemen zum teilautomatisierten Fahren (SAE J3016, Level 2) sind robuste mechatronische Systeme in die Fahrzeuge zu integrieren, um die Sicherheit, den Komfort und die Effizienz des Fahrerlebnisses zu verbessern.

Die Entwicklung teilautomatisierter Fahrerassistenzsysteme geht mit einer Vielzahl von Bedürfnissen einher. Einer der Hauptgründe ist die Verbesserung der Verkehrssicherheit. Durch die Übernahme bestimmter Fahraufgaben können Fahrerassistenzsysteme helfen, menschliche Fehler zu reduzieren und Unfälle zu vermeiden. Diese können beispielsweise bei der Notbremsung, beim Spurhalten, beim Einparken oder bei der Geschwindigkeitsregelung unterstützen. Darüber hinaus sollen teilautomatisierte Fahrerassistenzsysteme den Fahrkomfort erhöhen und somit dazu beitragen, die Fahrerermüdung zu reduzieren, indem sie repetitive oder monotone Aufgaben in definierten Situationen übernehmen. Ein weiterer Aspekt ist die Verbesserung der Fahrzeugeffizienz, z. B. durch optimierte Geschwindigkeitsregelung oder durch eine vorausschauende Anpassung des Fahrverhaltens an die Verkehrsbedingungen.

Die Entwicklung solcher komplexer Assistenzsysteme bringt entsprechende Herausforderungen mit sich. Ein wichtiger Aspekt ist die Gewährleistung der Zuverlässigkeit und Sicherheit. Die Assistenzsysteme müssen in der Lage sein, in verschiedenen Verkehrssituationen und unter unterschiedlichen Umgebungsbedingungen korrekt zu funktionieren. Dies erfordert eine umfangreiche und abgesicherte Verifikation (Simulationen) und Validation (reale Tests) der Soft- und Hardwaremodule sowie der Funktion im Gesamtsystem (Fahrzeug). Alle Hardwarekomponenten (bspw. Sensoren, Electronic Control Unit (**ECU**), Aktoren) müssen eine robuste Datenverarbeitung gewährleisten, um die Anforderung in diesem Bereich erfüllen zu können. In den meisten Fällen müssen Fahrerassistenzfunktionen Informationen in Echtzeit (online) aus verschiedenen Sensoren erfassen, interpretieren und angemessene Handlungen ableiten können. Die Basis hierfür bildet unter anderem die umfassende Erkennung der Umgebung und eine entsprechende Datenvorverarbeitung, um die Assistenzfunktionen darüber zu informieren, welche Objekte sich im Fahrzeugumfeld befinden.

## 1.1 Motivation und Problemstellung

Im Bereich der Advanced Driver Assistance System (**ADAS**) werden unterschiedliche Sensortechnologien für die Wahrnehmung des eigenen Umfelds eingesetzt. Fahrzeughersteller wie die Daimler Truck AG beziehen für die Entwicklung ihrer aktiven Sicherheitssysteme Sensoren von unterschiedlichen Lieferanten. Die Sensoren zur Erfassung des Umfelds (Bsp. Kamera, Radar, Lidar) haben in der Regel eine integrierte Electronic Control Unit (**ECU**), welche die Rohdaten verarbeiten und dem Fahrzeug unter anderem die detektierten Sensor Objekte (**SO**) (z.B. Fahrzeuge, Fußgänger, etc.) und deren Zustände zur Verfügung stellen (bspw. Position relativ zu einem Koordinatensystem, Geschwindigkeit auf der Ebene, Wahrscheinlichkeit der Existenz, etc.).

Die Herausforderung für Fahrzeughersteller besteht darin, die Zustandsinformationen auf der Sensor Objekt (**SO**) Ebene in der **ADAS ECU** so zu verarbeiten, dass real existierende Objekte verfolgt und plausible Trajektorien (Tracks) für die aktiven Sicherheitssysteme (Fahrzeugfeatures) bereitgestellt werden. Um diese Aufgabe zu realisieren, werden Trackingverfahren eingesetzt, welche nach dem Tracking by Detection (**TbD**) Paradigma entwickelt wurden (detailliert Einführung in [2.4](#)). Die Objektdetektion und Klassifizierung aller im Umfeld sensierten Objekte hat unabhängig vom Tracking bei **TbD** Verfahren bereits stattgefunden. Die Herausforderungen bei der Entwicklung kann in folgenden Teilaufgaben beschrieben werden:

- **Zustandsvorhersage** - Das Tracking übernimmt zum einen die Aufgabe der Prädiktion (Prediction) über die Zustände aller erkannten **SO** im nächsten Zeitschritt.
- **Assoziation** - Zum Anderen müssen die Zustände der **SO** (Messungen) den prädizierten Objektzuständen (Tracks) zugeordnet werden.
- **Update** - Abweichungen zwischen den vorhergesagten und gemessenen Zuständen müssen pro Zeitschritt entsprechend korrigiert werden.
- **Trackmanagement** - Unplausible, nicht mehr relevante bzw. nicht mehr existierende Tracks müssen entfernt und hinzukommende, bestätigte **SOs** als neue Tracks aufgesetzt werden.

Die Herausforderung steigt mit der zu erfassenden Anzahl und sich kreuzenden Objekten in einer Szene. Als State-of-the-Art Tracking bzw. Datenfusionsverfahren haben sich einige Filtervarianten (wie etwa das Kalman Filter (**KF**)) für die dynamische Zustands schätzung und einige Assoziationsverfahren (wie etwa Global Nearest Neighbor (**GNN**), Joint Probabilistic Data Association (**JPDA**)) etabliert (detaillierte Einführung in [2.1](#)).

Die Grenzen der eingesetzten Algorithmen zeigen sich allerdings durch die Notwendigkeit von heuristischen Regeln, welche in der Systementwicklung für unterschiedliche Szenen ermittelt werden, in denen das Tracking Schwächen aufzeigt. Die durchgeführten Analysen und entsprechend abgeleiteten Verfahrensvorschriften können jedoch dazu führen, dass Seiteneffekte auftreten. Werden beispielsweise Schwächen eines Sensors in einer bestimmten Szene erkannt, kann das Confidence Level für den jeweiligen Sensor herabgestuft werden. Das kann bedeuten, dass durch eine implementierte Regel der Algorithmus (bzw. das Tracking) in dem untersuchten Szenario eine verbesserte Performance zeigt, sich jedoch die Gesamtperformance auf einen Validierungsdatensatz (Testkatalog) verschlechtert hat. Auf dieser Basis können in vielen Fällen keine kontextbasierten (impliziten) Vorschriften ermittelt und implementiert werden. So können schon in scheinbar einfachen Szenarien wichtige Details, welche entscheidend für die explizite Lösung sind nicht vollumfänglich in Programmregeln abgebildet werden.

## 1.2 Zielsetzung

In der Problemstellung wurde allgemein dargestellt, welche Herausforderungen eine softwaregetriebene Multi Objekt Verfolgung mit sich bringt. Machine Learning (**ML**) Verfahren ermöglichen es, ein datengetriebenes Modell zu entwickeln, welches kontextbasierte Regeln auf Basis der verfügbaren Daten erlernt. Insgesamt können nach Alexander Amini [2] **ML** Verfahren viele Vorteile bieten, insbesondere in Bezug auf die Automatisierung, Robustheit, Skalierbarkeit, Flexibilität und kontinuierliche Verbesserung von Modellen. Diese sind ebenfalls in der Lage mit Störungen oder unvollständigen Daten umgehen zu können, wohingegen klassische Verfahren anfällig für Störungen dieser Art sind. Durch die Entwicklung einer entsprechenden Simulationsumgebung lassen sich **ML** Modelle datenbasiert und während des Fahrzeugentwicklungsprozesses mit steigender Datenverfügbarkeit optimieren, ohne dabei den Bedarf an Ressourcen für die Implementierung auf der Zielhardware zu erhöhen, Pallauf [27]. Es stellt sich somit die Frage, welche bisher eingesetzten Algorithmen bzw. Teilaufgaben des **TbD** basierten **MOT** Verfahrens (Zustandsvorhersage, Assoziation, Update, Trackmanagement) durch den Einsatz von **ML** gewinnbringend ersetzt werden können.

Ziel der Arbeit ist es daher, Machine Learning (**ML**) Verfahren für den Einsatz der Objekt Verfolgung zu entwickeln und mit Hilfe einer anerkannten Tracking Metrik zu bewerten (Konzeption, Implementierung und Simulation auf Basis eines etablierten Datensatzes). Es soll somit geprüft werden wie performant **ML** basierte Ansätze in einem bestehenden Kalman Filter (**KF**) Framework zu integrieren sind. Die Untersuchungen der Arbeit dienen zudem dem steigenden Interesse an datenbasierten Simulationen und

Softwareentwicklungsumfängen im **ADAS** Umfeld. Der Arbeitsumfang beinhaltet eine Recherche und theoretische Bewertung möglicher einzusetzender Verfahren, eine oder mehrerer Konzeptvorschläge für die Implementierung von **ML** Ansätzen sowie die Implementierung in einen **TbD** Framework, um die **MOT** Performance offline prüfen zu können. Als Simulationsumgebung für die Implementierung auf der Konzeptebebene wurde Matlab von der Firma Mathworks definiert.

## 1.3 Gliederung der Arbeit

Die erstellte Gliederung bietet einen Überblick über die beiden Hauptkapitel dieser Arbeit. In den folgenden Abschnitten sind die Inhalte der jeweiligen Unterkapitel in wenigen Sätzen zusammengefasst.

**Kapitel 2 Theoretische Grundlagen** bietet eine umfassende Einführung in die theoretischen Konzepte, die für das grundsätzliche Verständnis von Machine Learning (**ML**) und Multi Object Tracking (**MOT**) Verfahren notwendig sind.

2.1 beschäftigt sich mit klassischen Methoden zur Filterung und Verfolgung von Objekten. Es werden Algorithmen und Techniken vorgestellt, die es ermöglichen, die Position eines Objekts über die Zeit zu schätzen und zu verfolgen, basierend auf den Daten von Sensoren zur Erfassung des Umfelds eines Trucks.

In 2.2 wird das Ziel der Arbeit hervorgehoben, Machine Learning (**ML**) einzusetzen, um Teilaufgaben der Multi Objekt Verfolgung zu lösen. Es werden notwendige Begriffe und Themengebiete dieses umfangreichen Fachgebiets eingeführt und erläutert.

Das Unterkapitel 2.3 führt die Aufgabe der Einzel Objektverfolgung ein und zeigt unterschiedliche Lösungsmöglichkeiten. Es werden aktuelle Deep Neural Network (**DNN**) basierte Tracking Verfahren vorgestellt und für den Einsatz im Rahmen dieser Arbeit diskutiert.

Abschließend mit 2.4 Multi Objekt Verfolgung / Multi Object Tracking (**MOT**) werden die weiteren erforderlichen Begriffe eingeführt und Herausforderungen dargestellt. Es wird gezeigt, dass Multi Object Tracking (**MOT**) ein aktiver Forschungsbereich ist, insbesondere im Kontext von Advanced Driver Assistance System (**ADAS**) und des automatisierten Fahrens. Die Rechercheergebnisse werden diskutiert und die Verfahren vorgestellt, welche als Grundlage für die Entwicklung im Rahmen dieser Arbeit dienen.

**Kapitel 3 Ergebnisse** stellt eine detaillierte Übersicht zu den Entwicklungsergebnissen dar, welche im Rahmen dieser Arbeit erbracht wurden.

Kapitel 3.1 beschreibt die methodische Vorgehensweise, die Anforderungen und Einschränkungen für die Entwicklung innerhalb dieser Arbeit. Es wird gezeigt, welcher Multi Object Tracking (**MOT**) Framework der Arbeit als Entwicklungsrahmen dient und der logische Aufbau des Ergebniskapitels daran abgeleitet.

Der verfügbare Datensatz wird als Grundlage für die Entwicklung und Evaluierung eines **MOT** Verfahrens in 3.2 eingeführt. Vorhandene Einschränkungen werden diskutiert und mögliche Einflüsse auf die Entwicklung innerhalb dieser Arbeit betrachtet.

Das Kapitel 3.3 konzentriert sich auf die Entwicklung und Implementierung des Single Prediction Network (**SPENT**) zur Schätzung der Zustandsänderung erfasster Tracks. Das verfolgte Hauptziel ist es, ein **ML** Konzept zu entwickeln, das in den **TbD** Framework integriert werden kann. Der Entwicklungsablauf folgt dem bekannten **ML** Workflow für überwachtes Lernen, einschließlich Datenbeschaffung, Vorverarbeitung, Datensatzaufteilung, Modelltraining und -optimierung.

Analog zum vorangegangenen Kapitel, wird in 3.4 ein **ML** Modell zur Lösung des Datenassoziationsproblems vorgestellt. Dabei liegt der Fokus auf der Entwicklung eines Single Association Network (**SANT**), welches ohne zuvor gebildetes Abstandsmaß, einer Messung zu einem erfassten Track Set assoziiert. Das Vorgehen orientiert sich ebenfalls am bekannten **ML** Workflow.

Kapitel 3.5 zeigt die Weiterentwicklung von **SANT** zu einem Multi Association Network (**MANTa**). Es wird gezeigt, welche Daten und Netzwerkanpassung die Zielsetzung erfordern. Die dargestellten Einschränkungen auf Grund der Datenlage werden vorgestellt und experimentell verifiziert.

Das abschließende Kapitels 3.6 behandelt die Evaluierung der entwickelten Netzwerke innerhalb definierten **MOT** Frameworks. Dafür wird die **OSPA** Metrik eingeführt und die Funktionsweise des **MOT** Verfahrens erläutert. Es wird beschrieben, wie verschiedene Module des Frameworks zusammenarbeiten, um eine effektive Objektverfolgung zu ermöglichen. Die entwickelten Netzwerke aus Kapitel 3.3 und 3.4 werden entsprechend implementiert und die Integration beschrieben. Die unterschiedlichen Vorhersage- und Assoziationsverfahren werden anhand der **OSPA** Metrik miteinander verglichen.

## 2 Theoretische Grundlagen

In diesem Kapitel werden die fachlichen Grundlagen und Methoden eingeführt, welche in dieser Arbeit die Wissensbasis für den Einstieg in das jeweilige Themengebiet darstellen oder für den behandelten Anwendungsfall erweitert wurden. Hierzu wurde eine Recherche zum Stand der Technik und Wissenschaft im Bereich der Objekt Verfolgung bzw. Multi Object Tracking ([MOT](#)) und Deep Neural Networks ([DNNs](#)) durchgeführt.

Die Fahrzeugumfelderfassung mittels Sensoren bildet die Grundlage für die rechnergesteuerten Fahrerassistenzsystemen bzw. Advanced Driver Assistance Systems ([ADASs](#)). Die Sensordatenfusion kann im Allgemeinen als Methode zur kombinierten Auswertung von Sensordaten bezeichnet werden, wobei durch die gezielte Zusammenführung der Datenquellen eine höherwertige Information generiert werden kann. Es können mehrere unterschiedliche Sensorquellen genutzt werden, jedoch können auch zeitlich versetzte Messungen eines Sensorsystems kombiniert zu einem Informationsgewinn führen. In der Literatur werden nach Mitchell et al. [25] folgende Vorteile der Datenfusion zusammengefasst:

- **Repräsentanz** - Die aus der Fusion resultierende Information besitzt eine höhere Abstraktionsebene oder Granularität als die einzelnen Datenquellen.
- **Bestimmtheit** - Durch die Fusion  $VF$  wird eine Verstärkung der Aussagekraft oder Plausibilität  $p(V)$  eines Einzelsensors  $V$  mit  $p(VF) > p(V)$  erwartet.
- **Genaugigkeit** - Die Standardabweichung der Daten ist nach der Fusion kleiner als die der eingehenden Einzelsensoren.
- **Vollständigkeit** - Jeder neue Informationszusatz trägt zur Komplettierung der Umfeldwahrnehmung bei.

Die Verfolgung mehrerer Objekte kann als Problem der zusammenhängenden Schätzung der Zustände mehrerer Objekte während ihrer gesamten Existenz beschrieben werden. Somit besteht nach Milan et al. [24] die Aufgabe eines [MOT](#) Verfahrens darin, die Anzahl der in jedem Zeitschritt vorhandenen Objekte zu bestimmen und ihre eindeutige Identität zu wahren. Darüber hinaus werden die Zustände (States) pro Zeitschritt geschätzt und damit die Trajektorie eines Objekts erzeugt. Dabei wird ein verfolgtes Objekt gewöhnlich als Track bezeichnet, während die Beobachtung eines Objekts durch einen Sensor in der Regel als Messung bzw. Sensor Objekt ([SO](#)) bezeichnet wird.

Aktuelle MOT Ansätze können sich in folgenden Kategorien unterscheiden:

**Initialisierungsmethode** Innerhalb der Initialisierungsmethode wird in "detektionsbasierte und detektionsfreie Verfolgung" unterteilt. Bei der "detektionsbasierten Verfolgung" wird auch vom Tracking by Detection (**TbD**) Paradigma gesprochen, bei welchem der Trackingprozess die Detektionshypotesen aus jedem Zeitschritt mit den Trajektorien verknüpft. Als "detektionsfreie Verfolgungsansätze" werden Verfahren bezeichnet, bei denen die Lokalisierung und Verfolgung nicht getrennt voneinander durchgeführt werden bzw. die Detektion nicht die Basis für den Trackingalgorithmus darstellt. Diese Verfolgungsansätze können nach Leal-Taixé [18] in die beiden Tracking Paradigmen Tracking by Regression (**TbR**) und Tracking by Attention (**TbA**) eingeteilt werden.

**Verarbeitungsmodus** Der Verarbeitungsmodus wird unterteilt in "Online-Tracking", bei dem die Daten in jedem Zeitschritt sequentiell verarbeitet werden (kann auch als open loop bezeichnet werden) und "Offline-Tracking", bei dem der Berechnungsalgorithmus alle Messungen einer Zeitreihe gemeinsam verarbeitet (closed loop).

Die für **ADAS** am häufigsten verwendeten MOT Verfahren basieren auf probabilistischen Filtern, die sich für die Online-Verarbeitung eignen und in detektionsbasierten Systemen (**TbD**) verwendet werden können. In den letzten Jahren wurden jedoch auch immer mehr Arbeiten zu Verfolgungsalgorithmen veröffentlicht, welche auf Machine Learning (**ML**) basieren. Schindler et al. [29] beschreibt beispielsweise den Einsatz von Recurrent Neural Networks (**RNNs**) in einer Architektur, um einzelne Aufgaben eines Multi Object Tracking (**MOT**) Verfahrens zu lösen. In dieser Arbeit ist das **MOT** Verfahren ebenfalls als Problem definiert, welches als Filteranwendung verstanden wird. Die Einführung der Betrachtung findet im folgenden Kapitel statt.

## 2.1 Filter- und Trackingverfahren

Die bayes'sche Theorie ist die grundlegende Basis für viele Filteranwendungen. Diese beschreibt nach Dingler et al. [6] die Berechnung einer posterioren Wahrscheinlichkeitsfunktion (A-posteriori-Verteilung) eines Zustands  $x_m$  unter Verwendung des Priors und der Wahrscheinlichkeit von Beobachtungen bei einer Reihe von unbekannten Zuständen  $x_{0:m} = x_0, x_1, \dots, x_m$  und einer Reihe von Beobachtungen  $z_{0:n} = z_0, z_1, \dots, z_n$ . Aufgrund verschiedener Fehlerquellen werden die zu schätzenden Zustände in der Regel mit Hilfe von Wahrscheinlichkeitsverteilungen verfolgt, die den Grad der Unsicherheit widerspiegeln können.

Das Bayes'sche Filter oder Bayes-Filter ist ein rekursives probabilistisches Verfahren zur Schätzung von Wahrscheinlichkeitsverteilungen unbeobachteter Zustände eines Systems bei gegebenen Messungen. Ein Bayes-Schätzer ist somit eine Schätzfunktion, die zusätzlich zu den Messungen Vorwissen über einen zu schätzenden Zustandswert  $x_k$  berücksichtigt. Allgemein definiert man einen Bayes-Schätzer als denjenigen Wert, der den Erwartungswert einer Verlustfunktion unter der A-posteriori-Verteilung minimiert. Für eine quadratische Verlustfunktion ergibt sich dann gerade der A-posteriori-Erwartungswert als Schätzer.

### 2.1.1 Kalman Filter (KF)

Das Kalman Filter (KF) folgt der bayes'schen Theorie und kann daher nach Evans et al. [8] verwendet werden, um ein durch mehrere Zustände definiertes Objekt über die Zeit zu verfolgen und dessen Zustände mit entsprechenden Unsicherheiten zu schätzen. Es kann als rekursives Verfahren implementiert werden, so dass neue Messungen verarbeitet werden können, sobald diese eintreffen (open loop / online Vorhersage). Das KF ist ein optimaler bayes'scher Filter zur Schätzung von Zuständen linearer, gaußverteilter Systeme, d. h. es kann dazu verwendet werden Zustände aus indirekten, ungenauen und unsicheren Messungen zu schätzen, welche diesen Bedingungen entsprechen. Unter der Annahme, dass das gesamte Rauschen (Weißes Rauschen) gaußförmig ist, minimiert das KF nach Mitchell et al. [25] den mittleren quadratischen Fehler der geschätzten Zustände und stellt somit einen optimalen bayes'schen Filter zur Schätzung von Zuständen in linearen Systemen dar.

Soll das Kalman Filter (KF) für ein Multi Object Tracking (MOT) Verfahren eingesetzt werden, benötigt es daher ein lineares Zustandsraummodell, dass die Systemdynamik darstellt, und ein ebenfalls lineares Beobachtungsmodell, welches die Messungen beschreibt. Wenn man nur den Mittelwert und die Standardabweichung des Rauschens

kennt, ist das **KF** nach Dingler et al. [6] der beste lineare Schätzer. Die allgemeine Beschreibung eines zeitdiskreten Zustandsraumsystems wird für die Anwendung des Kalman Filter (**KF**) auf ein lineares zeitinvariantes System reduziert:

$$x_{k+1} = f(x_k, u_k) + w_k \quad (2.1)$$

$$z_k = h(x_k) + v_k \quad (2.2)$$

Dabei ist  $f(x_k, u_k)$  die Zustandsübergangsfunktion des Zustands  $x_k$  und der Steuerung  $u_k$  im Zeitschritt  $k$ . Im linearen System wird diese durch die dynamische Matrix  $Ax_k$  und die Steuerungsmatrix  $Bu_k$  ersetzt.  $h(x_k)$  ist die Messfunktion, die bei linearen Systemen durch die Messmatrix  $Hx_k$  ersetzt wird. Sowohl der Zustand  $x_{k+1}$  als auch die Messung  $z_k$  unterliegen dem voneinander unabhängigen weißen gaußschen Rauschen  $w_k$  und  $v_k$  mit Mittelwert Null. Sie sind durch die Kovarianzmatrix des Prozessrauschen  $Q_k$  und die Kovarianzmatrix des Messrauschen  $R_k$  definiert (siehe Formel 2.6 / 2.8).

$$x_{k+1} = Ax_k + Bu_k + w_k \quad (2.3)$$

$$z_k = Hx_k + v_k \quad (2.4)$$

Eine detaillierte Beschreibung und Herleitung der aufgeführten Gleichungen ist in Mitchell et al. [25] zu finden. Die Zustandsschätzung des **KF** erfolgt in zwei aufeinanderfolgenden Schritten, die als Vorhersage (Prediction) und Aktualisierung (Update) bezeichnet werden. Der Zustand wird durch den Mittelwert  $x_k$  und die Kovarianzmatrix  $P_k$  dargestellt. Abbildung 2.1 zeigt ein Schema der Berechnungsschritte des **KF** über drei Zeitschritte.

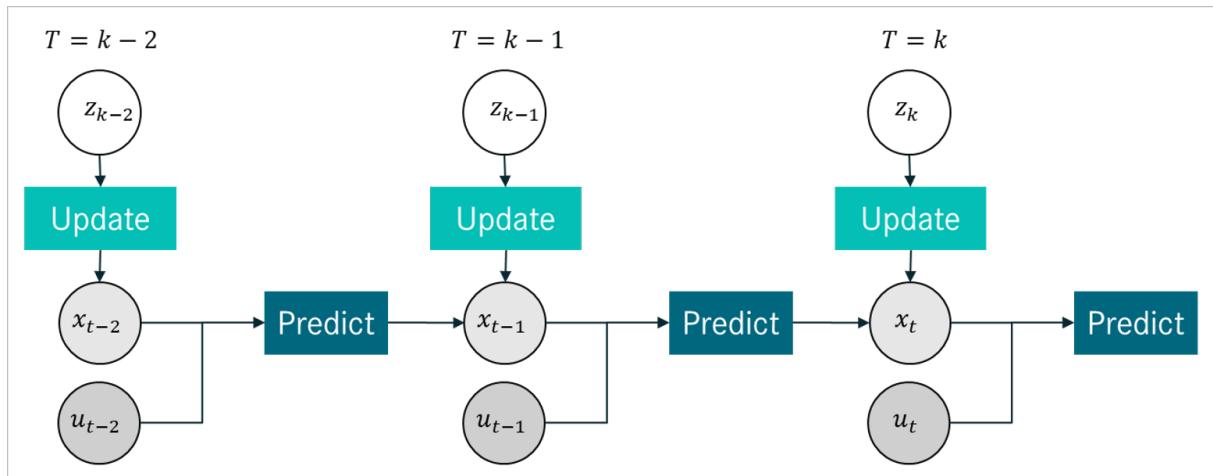


Abbildung 2.1: Schematische Darstellung, Berechnungsschritte des Kalman Filter<sup>1</sup>

---

<sup>1</sup> nach Mitchell et al. [25] *Multi-Sensor Data Fusion: An Introduction*

**Vorhersage (Prediction)** - Während des Vorhersageschrittes werden der Mittelwert  $\hat{x}_{k|k-1}$  und die Kovarianz  $P_{k|k-1}$  zum vorhergegangenen, für den aktuellen Zeitschritt  $k$  auf der Grundlage des posterioren Mittelwerts  $\hat{x}_{k-1|k-1}$  und der Kovarianz  $P_{k-1|k-1}$  des vorherigen Zeitschritts  $k - 1$  berechnet:

$$\hat{x}_{k|k-1} = A_k \hat{x}_{k-1|k-1} + B_k u_{k-1} \quad (2.5)$$

$$P_{k|k-1} = A_k P_{k-1|k-1} A_k^T + Q_k \quad (2.6)$$

Die Indizierungsschreibweise  $k|k - 1$  drückt für den Zustand zum Zeitpunkt  $t_k$  die Bedingtheit der Schätzung durch den vergangenen Zustand  $k - 1$  aus.

**Aktualisierung (Update)** - Der Aktualisierungsschritt korrigiert dann die vorherige Schätzung von Mittelwert und Kovarianz anhand einer Messung. Zunächst werden die Innovation  $\hat{z}_{k|k-1}$  und die Innovationskovarianz  $S_k$  berechnet, um die Kalman-Verstärkung  $K_k$  zu erhalten.

$$\hat{z}_{k|k-1} = H \hat{x}_{k|k-1} \quad (2.7)$$

$$S_k = H P_{k|k-1} H^T + R_k \quad (2.8)$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \quad (2.9)$$

Anschließend werden der aktuelle Mittelwert und die aktuelle Kovarianz unter Verwendung des vorherigen Wertes, der Innovation und der Kalman-Verstärkung berechnet:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - \hat{z}_{k|k-1}) \quad (2.10)$$

$$\hat{P}_{k|k} = \hat{P}_{k|k-1} - K_k H_k P_{k|k-1} \quad (2.11)$$

Die Kalman-Verstärkung bestimmt den Umfang der Korrektur während des Aktualisierungsschritts. Die Filtereigenschaften des **KF** werden hauptsächlich durch die beiden Kovarianzmatrizen  $Q_k$  und  $R_k$  bestimmt. Wenn ein hohes Prozessrauschen  $Q_k$  gewählt wird, ist die Kalman-Verstärkung hoch, was zu einer hohen Korrektur durch die Messungen führt. Die Verwendung eines hohen Messrauschens  $R_k$  führt zu einer niedrigen Kalman-Verstärkung und damit zu einer geringen Korrektur durch die Messungen.

**Erweiterungen des Kalman Filters** Es existieren mehrere Erweiterungen des Kalman-Filters, die auf nichtlineare Systeme angewendet werden können. Das Extended Kalman Filter (**EKF**) ist nach Evans et al. [8] auf Systeme mit nichtlinearen Zustandsübergangs- und Messfunktionen  $f(x_k, u_k)$  und  $h(x_k)$  anwendbar, indem die dynamische Matrix  $A_k$  und  $H_k$  bei jedem Zeitschritt mit Hilfe der Taylor-Reihen approximiert wird. Diese werden als Jacobians bezeichnet. Der vorherige Mittelwert und die Innovation können

mit Hilfe der nichtlinearen Funktionen mit Hilfe der Jacobians berechnet werden. Das Unscented Kalman Filter (**UKF**) ist eine weitere nichtlineare Version des Kalman Filter (**KF**), bei der die Jacobians nicht berechnet werden müssen. Stattdessen verwendet er die unscented Transformation, die mehrere Sigma-Punkte zur Schätzung der Kovarianzen verwendet. Dies führt oft zu einer höheren Genauigkeit im Vergleich zu **KF** und **EKF**, während die Rechenkomplexität weiterhin gering ist. Eine detaillierte Beschreibung der Erweiterungen des **KF** findet sich in Evans et al. [8].

**Kalman Filter (KF) für Multi Object Tracking (MOT)** Nachdem das mathematische Grundverständnis über das **KF** erarbeitet wurde, ist in Abbildung 2.2 abschließend eine schematische Darstellung einer Kalman Filter (**KF**) basierenden **MOT** Anwendung skizziert. Dabei werden die Zustände eines jeden erfassenden Objekts mit einem separaten **KF** geschätzt. Die Anwendung von **KF** auf **MOT** Verfahren kann im Wesentlichen als Erweiterung der Einzelzielverfolgung (Single Object Tracking (**SOT**)) verstanden werden und erfordert ein zusätzliches Track Management sowie eine Zuordnung der Messungen und Tracks (Data Association (**DA**)). Die Abbildung zeigt somit die notwendigen Berechnungsschritte bzw. Softwarekomponenten, die für **MOT** in einem **KF** Framework benötigt werden. In jedem Zeitschritt werden die Elemente  $z_{1:n} = z_n, z_2, \dots, z_n$  einer Detektionsmenge  $Z$  den vorhandenen Tracks zugeordnet. Die vom Trackmanagement bestätigte Trackmenge  $X$  mit den Elementen  $x_{1:m} = x_1, x_2, \dots, x_m$  kann innerhalb der **ADAS** Architektur entsprechend weitergeleitet werden.

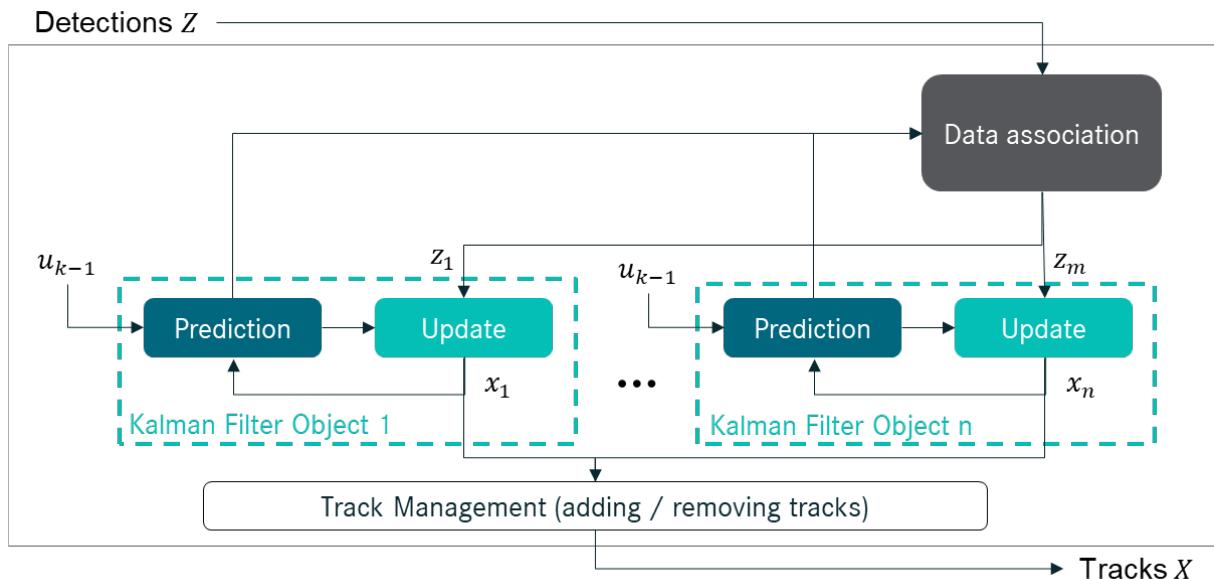


Abbildung 2.2: Schematische Darstellung eines Kalman Filterprozesses mit Track Management und Datenassoziation<sup>1</sup>

<sup>1</sup> nach Mitchell et al. [25] *Multi-Sensor Data Fusion: An Introduction*

## 2.1.2 Datenassoziation

In diesem Teilkapitel wird die zusätzliche Herausforderung eines MOT Systems, die Datenassoziation bzw. Data Association (**DA**), näher eingeführt. Die Aufgabe der Assoziation besteht im Kontext des Objekttrackings darin, Daten, die von gleichen oder verschiedenen Sensoren gemessen wurden, mit denselben physischen Objekten zu verbinden. Die Datenassoziation stellt somit eine Grundlage dar, um die Bewegungen und Interaktionen von mehreren Objekten in einer Szene zu verfolgen. Abbildung 2.3 zeigt eine schematische Szene, in welcher mehrere Detektionen ( $D_1$  und  $D_2$ ) zum Zeitpunkt  $t = 1$  bis zum Zeitpunkt  $t = 5$  vorliegen. Entsprechend der Farbmarkierung sind die Objektdaten zum jeweiligen Zeitschritt zuzuordnen.

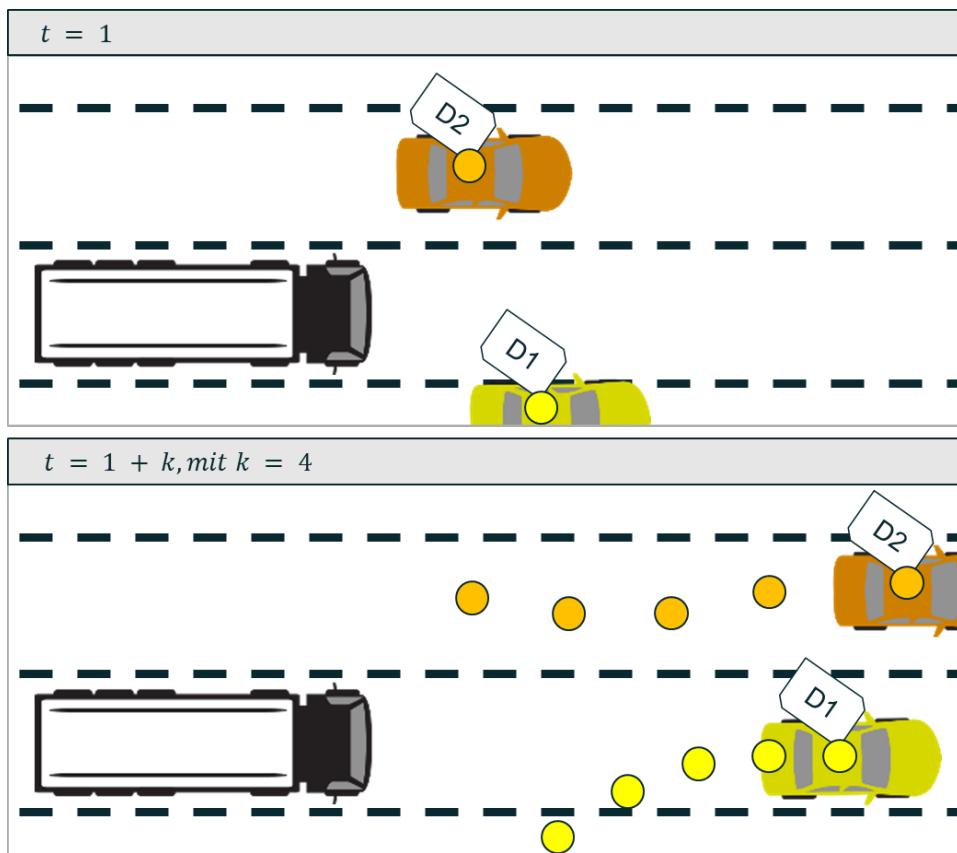


Abbildung 2.3: Schematische Darstellung der Assoziation detektierter Objekte

Die Aufgabe der Datenassoziation ist nach Wu et al. [37] ein grundlegendes Problem auf dem Gebiet der kombinatorischen Optimierung (The Assignment Problem). Die Zuordnung von Messungen zu Tracks ist ein Beispiel für ein kombinatorisches NP-hard Optimierungsproblem. Es existieren verschiedene Ansätze und Methoden zur Datenassoziation im Kontext des Multi Object Tracking (**MOT**). Ein grundlegendes Prinzip besteht darin, Merkmale oder Informationen von Sensor Objekten (**SO**) in einem Zeitschritt mit denen im Nächsten zu vergleichen, um mögliche Zuordnungen

zu bestimmen. Diese Merkmale werden in vielen Methoden für die DA auf SO Ebene durch ein Abstandmaß (euklidische-, mahalanobis-Distanz, etc.) gebildet. Aufgrund der beschränkten Informationsmenge bieten diese Verfahren den Vorteil eines geringen Rechenaufwands. Zusätzlich wurden nach Wu et al. [37] in der Vergangenheit Konzepte zur Optimierung der Rechenzeit eingeführt, welche mit Gates arbeiten. Yang et al. [39] beschreibt die verschiedene Methoden, welche eingesetzt werden können, um eine optimale Zuordnung finden zu können:

- **Nearest Neighbor Methode** - Die einfachste Methode der Datenassoziation besteht darin, die nächstgelegenen Datenpunkte aus verschiedenen Sensoren miteinander zu verbinden. Diese Methode ist sehr effizient, aber auch anfällig für Fehler, insbesondere wenn sich die Objekte in der Nähe voneinander befinden (überfüllten Szenen mit Clutter oder Verdeckungen).
- **Hungarian Algorithm (HA)** - Der ungarische Algorithmus (Hungarian algorithm) ist ein Polynominalzeit-Algorithmus, um die globale optimale Zuordnungsmatrix zu finden. Unter der Annahme, dass zum Zeitpunkt  $t$   $N$  Messungen existieren, kann der Algorithmus die Zuordnung zwischen Zielobjekt und Messung in einer Zeit von  $O(N^3)$  vornehmen.
- **Probabilistic Data Association (PDA)** - Ist eine Erweiterung des Kalman Filters und bietet eine Möglichkeit, Messungen mehreren Objekten zuzuordnen, wodurch eine höhere Tracking-Genauigkeit erreicht werden kann. Allerdings erfordert das Verfahren auch einen höheren Rechenaufwand als die aufgelisteten einfacheren Methoden (z.B. Nearest Neighbor Methode). Der Grundgedanke von PDA besteht darin, dass alle Messungen, die in die zugehörigen Gates einer vorhergesagten Position eines Tracks fallen, mit unterschiedlichen Wahrscheinlichkeiten vom Zielobjekt stammen können, und dass diese Wahrscheinlichkeiten verwendet werden, um die gewichtete Summierung der einzelnen effektiven Messungen zu berechnen. Das Verfahren ist jedoch nur für die Verfolgung eines einzelnen Zielobjekts unter Berücksichtigung von Fehlmessungen und verpassten Erkennungen geeignet.
- **Joint Probabilistic Data Association (JPDA)** - Der JPDA Algorithmus kombiniert die Idee der PDA mit einer gemeinsamen Wahrscheinlichkeitsverteilung, Bar-Shalom et al. [12]. Der Algorithmus modelliert die Wahrscheinlichkeiten für die Datenassoziation unter Berücksichtigung von Unsicherheiten in den Messungen und den Bewegungen der Objekte. Es werden die Wahrscheinlichkeiten dafür geschätzt, dass jede Messung zu jedem der verfolgten Objekte gehört. Der entscheidende Schritt des JPDA ist die Berechnung der gemeinsamen Wahrscheinlichkeitsverteilung. Diese Verteilung berücksichtigt alle möglichen Kombinationen

von Zuordnungen und gewichtet sie entsprechend ihrer Wahrscheinlichkeiten. Dadurch kann der Algorithmus die Unsicherheit in der Zuordnung von Daten reduzieren.

- **Soft Data Association (SoDa)** - Ist nach Hung et al. [41] eine Methode, die versucht, eine sinnvolle Verbindung von Datenpunkten zu finden, indem Wahrscheinlichkeiten berechnet werden, die die Verbindung der Datenpunkte mit verschiedenen Objekten modellieren. Dabei verwendet der vorgeschlagene Ansatz den Attention Mechanismus nach Vaswani et al. [15] in einem Netzwerkaufbau. Die vorgestellten experimentellen Ergebnisse deuten darauf hin, dass dieser Ansatz im Vergleich zum aktuellen Stand der Technik im visuellen MOT (Bilddatenebene) günstig abschneidet.

Jede der nach Rakai et al. [39] beschriebenen Methoden hat Vor- und Nachteile und die Wahl der richtigen Methode hängt von verschiedenen Faktoren ab, wie z.B. der Art der Daten, dem Anwendungsfall und den verfügbaren Ressourcen. In der Praxis werden oft mehrere Methoden kombiniert, um eine robuste und zuverlässige Datenassoziation zu erreichen.

## 2.2 Einführung in das maschinelle Lernen / Machine Learning (ML)

Ziel dieser Arbeit ist Machine Learning (ML) einzusetzen, um die beschriebene Herausforderung bzw. Teilaufgaben der Multi Objekt Verfolgung zu lösen. Im ersten Abschnitt dieses Kapitels werden daher notwendige Begriffe und Themengebiete dieses umfangreichen Fachgebiets eingeführt und erläutert.

Artificial Intelligence (AI) ist ein Oberbegriff, der sich auf alle Technologien und Methoden bezieht, die es Maschinen ermöglichen, menschenähnliche Intelligenzleistungen zu erbringen. Wie in Abbildung 2.4 schematisch dargestellt, ist Machine Learning (ML) nach Goodfellow et al. [9] eine Teildisziplin im Gebiet der künstlichen Intelligenz, welches sich darauf bezieht Abläufe und Funktionen von Maschinen aus Daten zu lernen, ohne explizit programmiert zu werden. Allgemein formuliert folgt das Konzept des maschinellen Lernens dem Ansatz, dass statt spezifischer Anweisungen einem Computer eine Menge an Daten gegeben wird, auf die er lernt Muster oder Zusammenhänge zu erkennen. Deep Learning (DL) ist wiederum ein Teilbereich des maschinellen Lernens, der sich auf künstliche neuronale Netze bezieht, welche eigenständig Merkmale (Features) in Daten erkennen, um die gewünschte Aufgabe ausführen zu können.

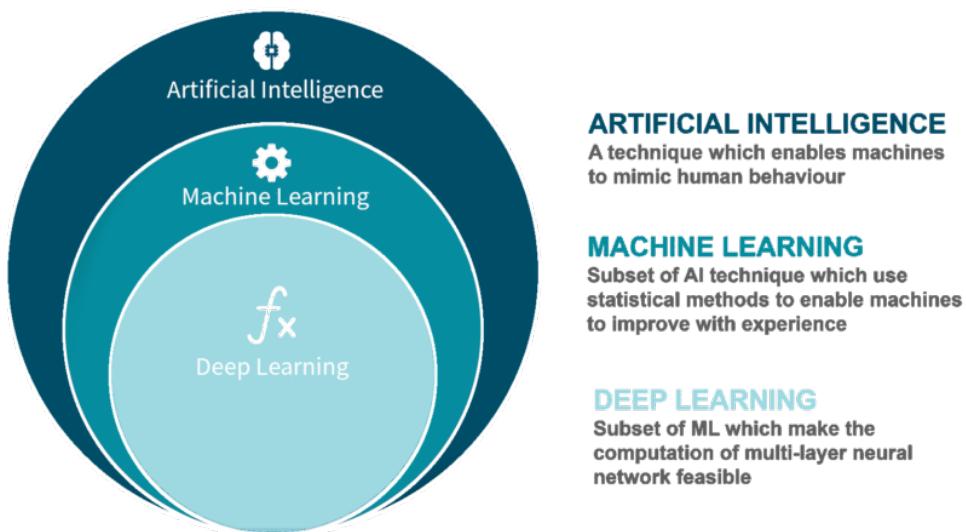


Abbildung 2.4: Terminologie Mengendiagramm - Artificial Intelligence (AI), Machine Learning (ML) und Deep Learning (DL)<sup>1</sup>

<sup>1</sup> nach Ava Soleimany [3] *Deep Sequence Modeling: MIT 6.S191*

Der Begriff Deep Learning (**DL**) hat sich mittlerweile zum Synonym der aktuellen Erfolgsgeschichte von **NNs** entwickelt und beschreibt den Trend zu vielschichtigen Architekturen, Deep Neural Networks (**DNNs**). Die komplexe Entscheidungsfindung von **DNNs** wird nach Goodfellow et al. [9] vielen realen Anwendungen gerecht. Vergangene Forschungsarbeiten haben das Potential von vielschichtigen Netzwerkarchitekturen wiederholt belegt und insbesondere in praxisnahen Bezug gestellt. Das Einsatzgebiet von **ML** ist nach Alexander Amini [2] längs in Alltagsprodukten und Anwendungen angekommen, nicht zuletzt aufgrund steigender Rechenleistung sowie effizienterer Modelle und Algorithmen.

### 2.2.1 Deep Learning (**DL**) und Neural Networks (**NNs**)

Die Modelle künstlicher neuronaler Netze sind dem menschlichen Gehirn bzw. Nervensystem nachempfunden und bestehen aus vielen Schichten von einzelnen Neuronen, die miteinander teilweise oder vollständig verbunden sind. Die Wirkungsweise eines künstlichen neuronalen Netzes basiert somit auf der Verarbeitung von Eingabedaten (Inputdata) durch viele Schichten von Neuronen. Jedes Neuron in einer Schicht erhält Signale von anderen Neuronen und führt daraufhin eine Berechnung durch, die dann an die nächste Schicht weitergeleitet wird. Nach Goodfellow et al. [9] können durch dieses systematische Vorgehen in den tieferen Schichten des Netzes komplexere Merkmale der Eingabedaten erkannt und verarbeitet werden. Am Ende des Netzes wird eine Vorhersage bzw. ein Ergebnis erzeugt, das auf den verarbeiteten Eingabedaten basiert. Abbildung 2.5 zeigt eine schematische Abbildung des Modells eines Neuronen (Perceptron) inklusive Signalverarbeitungskette.

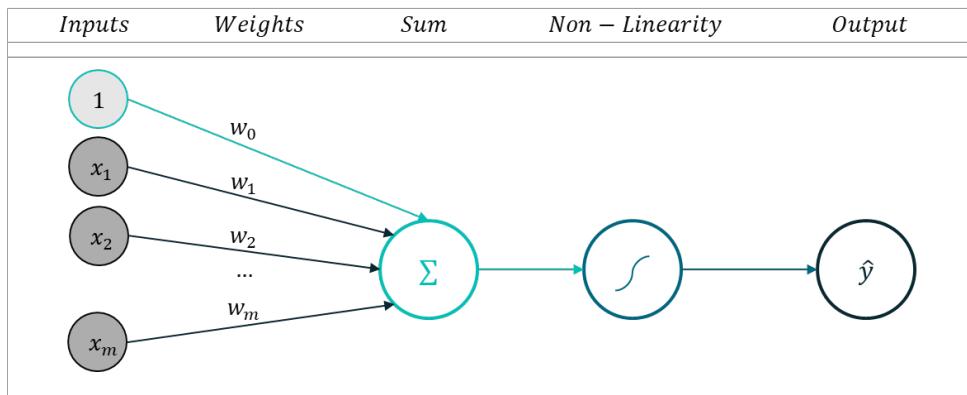


Abbildung 2.5: Schematische Darstellung, künstliches Neuron<sup>1</sup>

Jeder Eingabewert  $x$  (Inputs) wird mit einem Gewichtungsfaktor  $w$  (weights) multipliziert. Abhängig von den Vorzeichen der Gewichte kann eine Eingabe hemmend

<sup>1</sup> nach Alexander Amini [2] *Introduction to Deep Learning: MIT 6.S191*

(inhibitorisch) oder erregend (exzitatorisch) wirken. Ein Gewicht von 0 stellt somit eine nicht existente Verbindung dar. Die Übertragungsfunktion (Sum) bildet die Summe aller gewichteten Eingabewerte als Netzeingabe. Mit diesem Teil des Modells lassen sich eine beliebige Linearkombinationen bilden. Wie in Formel 2.12 zu sehen wird zusätzlich ein Wert für einen möglichen nachträglichen Versatz (Bias) vorgesehen. Das Ergebnis der summierten gewichteten Eingabewerte addiert mit dem Bias-Wert dient einer Aktivierungsfunktion  $g$  (activation function) als Inputwert.

$$\hat{y} = g(w_0 + \sum_{i=1}^m x_i w_i) \quad (2.12)$$

Damit kann eine nichtlineare Transformation realisiert werden. Als Aktivierungs- oder Transferfunktion können verschiedene Funktionstypen verwendet werden, abhängig von der verwendeten Netzwerkarchitektur. Um ein entsprechendes Netzwerk von künstlichen Neuronen zu erhalten, kann das beschriebene Modell in mehrfacher Ausführung nebeneinander zu einer Schicht angeordnet werden. In jedem Neuron findet der dreistufige Berechnungsprozess statt. Die einzelnen Schichten (Layer) können gestapelt werden (Stack) und somit eine tiefe Berechnungskette zwischen Input und Output modelliert werden. Abb. 2.6 zeigt ein single Layer bzw. shallow Network und ein tieferes Deep Neural Network (DNN).

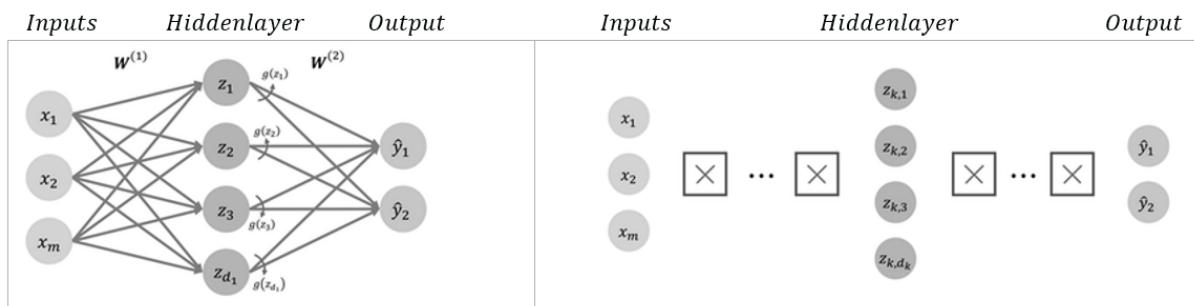


Abbildung 2.6: Schematische Darstellung Single Layer und Deep Neural Network<sup>1</sup>

**Aktivierungsfunktionen** Häufige nicht-lineare Aktivierungsfunktionen, welche ebenfalls in der Modellentwicklung innerhalb dieser Arbeit verwendet wurden sind mit den jeweiligen Formeln nach Goodfellow et al. [9] beschrieben:

- **Sigmoidfunktion (Sigmoid Function)** - Die Sigmoidfunktion bildet die eingehenden Eingaben auf einen Bereich zwischen 0 und 1 ab und eignet sich daher für Anwendungen, welche einen Wahrscheinlichkeitswert als Ausgabe erhalten

<sup>1</sup> nach [https://www.researchgate.net/figure/Single-Layer-Neural-Network-left-and-Deep-Neural-Network-right\\_\(Stand\\_21.08.2023\)](https://www.researchgate.net/figure/Single-Layer-Neural-Network-left-and-Deep-Neural-Network-right_(Stand_21.08.2023))

sollen. Zu beachten ist, dass die Aktivierung der Neuronen in Nähe 0 und 1 gesättigt wird. Das hat zur Folge, dass die Ableitung (entspricht dem Gradient der Verlustfunktion) sehr klein und damit die Aktualisierungen der Gewichte verhindert werden. Dieser Effekt kann somit den Lernprozess verlangsamen bzw. verhindern (vanishing gradient problem).

$$g(z) = \frac{1}{1 + e^{-z}} \quad (2.13)$$

- **Tanh-Aktivierungsfunktion (Hyperbolic Tangent)** - Die Funktion bildet eine reellwertige Zahl gemäß der Gleichung auf den Bereich -1 bis 1 ab.

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (2.14)$$

- **Gleichgerichtete Lineareinheit / Rectified Linear Unit (ReLU)** - Durch den Einsatz dieser Aktivierungsfunktion lassen sich Eingaben, die größer Null sind linear abbilden, kleinere Eingabewerte werden auf Null gesetzt. In der Praxis erzielt diese Funktion entsprechend geringe Rechenzeiten.

$$g(z) = \max(0, z) \quad (2.15)$$

- **Softmax** - Die Softmaxfunktion wandelt eine Anzahl an Werten  $z_i$  in einen Wahrscheinlichkeitsvektor mit  $i$  Werten. Dabei führt ein hoher numerischer Wert zu einer hohen Wahrscheinlichkeit im resultierenden Ausgabevektor (Basis für One-Hot-Encoding). Das hervorragende Merkmal dieser Funktion ist, dass die Summe der Ausgabewerte (Wahrscheinlichkeitswerten) immer kleiner oder gleich 1 ist.

$$g(z_i) = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}} \quad (2.16)$$

Der Einsatz von nicht linearen Aktivierungsfunktionen ist nach Goodfellow et al. [9] ein entscheidendes Kriterium für das Lösen von realen Problemen mittels **DL**. Erfasste Daten sind in vielen Anwendungen zu einem hohen Grad nichtlinear. Daher können vereinfachte bzw. Ableitungen auf linear implementierte Lösungsansätze zu einer akzeptablen Genauigkeit führen, je nach Anwendung jedoch auch eine nicht einsetzbare Performance erreichen. Nach dem universellen Approximationstheorem, beschrieben von White et al. [35], können Feed-Forward Networks (**FFNs**) beliebige Funktionen mit einem Hiddenlayer errechnen. Hierfür muss das Netzwerk entsprechend trainiert werden. In Abschnitt 2.2.2 wird das notwendige Trainingsverfahren näher erläutert.

## 2.2.2 Netzwerktraining von Feed-Forward Networks (FFNs)

Ein verbreitetes Verfahren zum Trainieren von FFNs ist das Backpropagation-Verfahren. Das Training mittels Backpropagation kann nach White et al. [35] in drei Phasen aufgeteilt und beschrieben werden:

1. **Phase** - Inputwerte werden vorwärts durch das Netz propagiert. Im Netzwerk werden alle Berechnungen vorwärtsgerichtet durchgeführt und ein Ergebnis entsprechend der Netzwerkarchitektur ausgegeben.
2. **Phase** - Das Ergebnis wird mit dem vorliegenden Erwartungswert verglichen und somit ein Fehler errechnet. Diese Abweichung wird auch als loss und die definierte Funktion als loss function (Cost function, Verlustfunktion) bezeichnet. Die Funktion wird für jeden Datenpunkt eines Datensatzes aufgerufen und somit das Ziel verfolgt die Gesamtabweichung zu minimieren.
3. **Phase** - Dieser berechnete Netzwerkfehler wird rückwärts durch das Netzwerk propagiert. Somit kann an jeder Neuronenverbindung ermittelt werden wie hoch die jeweilige Beteiligung an der Abweichung ist. Der jeweilige Neuronenfehler wird durch die Anpassung der Gewichte ( $w_i$ ) minimiert. Das Verfahren nähert sich somit an das erwartete Ergebnis pro Input an.

**Solververfahren für das Training von neuronalen Netzen** Beim Training von neuronalen Netzen gibt es verschiedene Optimierungsverfahren, auch als Solververfahren bezeichnet, die dazu dienen, die Gewichtungen und Parameter des Netzwerks anzupassen. Die Entwicklung von Solververfahren hat nach Ba et al. [4] im Laufe der Zeit eine Evolution durchlaufen, um die Konvergenzgeschwindigkeit und die Stabilität der Optimierung zu verbessern. Frühe Verfahren wie der stochastische Gradientenabstieg (Stochastic Gradient Descent (SGD)) und der Momentum-Algorithmus legten den Grundstein für adaptive Lernraten und Geschwindigkeitsanpassungen. Spätere Verfahren wie Adagrad und RMSProp führten weitere Verbesserungen ein, indem sie spezifische Herausforderungen der Konvergenz und Lernratenanpassung angegangen haben.

Ba et al. [4] stellt ein fortschrittliches Optimierungsverfahren vor, welches die Vorteile mehrerer der vorheriger Verfahren kombiniert. Adaptive Moment Estimation (ADAM) gilt heute als einer der effizientesten Solver für das trainieren von NN. Der Algorithmus berücksichtigt eine effiziente Nutzung der Moment-Schätzungen, eine Anpassung an unterschiedliche Lernraten, eine Skalierbarkeit auf Daten- und Netzwerkgrößen und zeigt in der Praxis weniger Anfälligkeit auf Abstimmung der Hyperparameter.

**ADAM** eignet sich ebenfalls für das Training von Recurrent Neural Networks (**RNNs**), welche in dem kommenden Kapitel 2.2.3 vorgestellt werden. Aufgrund der erläuterten allgemeinen Vorteile und der folgenden **RNN** spezifischen Überlegungen wurde **ADAM** innerhalb dieser Arbeit als Solververfahren für das Netzwerktraining ausgewählt:

- **Lernrate-Anpassung** - **ADAM** verwendet adaptive Lernraten, um die Anpassung der Gewichtungen während des Trainings zu steuern. Bei Recurrent Neural Networks (**RNNs**) kann dies hilfreich sein, um den Lernprozess zu stabilisieren und eine effiziente Konvergenz zu erreichen. Es sollen zeitliche Abhängigkeiten gelernt werden, wobei **ADAM** dazu beitragen könnte, die Lernraten in verschiedenen Abschnitten effektiv anzupassen.
- **Vanishing / Exploding Gradients** - **RNNs** sind anfällig für das Problem des Verschwindens oder Explodierens von Gradienten während des Trainings. **ADAM** kann dazu beitragen, diese Probleme zu verringern, indem adaptive Lernraten verwendet werden und somit eine effektivere Gewichtsanpassung ermöglicht.

### 2.2.3 Recurrent Neural Networks (**RNNs**)

Mit Recurrent Neural Network (**RNN**) wird eine weitere Art von künstlichen neuronalen Netzen vorgestellt, welche nach Ava Soleimany [3] in der Lage ist, Daten mit einer Sequenzstruktur zu verarbeiten. Im Gegensatz zu den bereits eingeführten Convolutional Neural Networks (**CNNs**) verfügen **RNNs** über Mechanismen, welche es den Modellen ermöglicht eine zeitliche Dimension in ihrer Funktionsapproximation zu berücksichtigen. Das Konzept von **RNNs** (rückgekoppelte neuronale Netze) wurde nach Goodfellow et al. [9] bereits in den 1980er Jahren entwickelt, jedoch waren die frühen Modelle aufgrund von Schwierigkeiten bei der Trainingsoptimierung und Behandlung von langen Abhängigkeiten in der Eingabe nicht sehr erfolgreich, Jonas Knupp et al. [14]. Seitdem wurden zahlreiche Varianten entwickelt, darunter Gated Recurrent Unit (**GRU**), Simple Recurrent Unit (**SRU**), Long Short-Term Memory (**LSTM**) und Attentional **RNNs**. Nach Goodfellow et al. [9] sind einige dieser Modelle heute ein wichtiger Bestandteil von **DL** und werden in einer Vielzahl von Anwendungen eingesetzt, einschließlich Spracherkennung, maschinellen Übersetzen, Text- und Bildgenerierung, welche alle zu dem Themengebiet Deep Generative Modeling (**DGM**) gehören. Um eine entsprechenden Zeitreihe in geordneter Folge zu behandeln wird die zeitliche Variable  $t$  bei **RNNs** als weitere Dimension genutzt. Ava Soleimany [3] erläutert, dass somit eine Lösung entwickelt wurde, isolierte Entscheidungen eines künstlichen Netzwerks zu einem gegebenen Zeitschritt von den vergangenen Daten abhängig zu machen (kontextbezogene Entscheidungen). Um diese Beziehung (recurrent bzw. rückgekoppelt) herzustellen

wurde die zusätzliche Wertematrix  $h$  eingeführt, welcher den Zustand des Netzwerks zu einem bestimmten Zeitschritt beschreibt (hiddenstate). Nach Goodfellow et al. [9] wurde die Funktion zur Berechnung des Ausgabewertes  $y_t$  erweitert, dass die Eingabedaten  $x_t$  und die Zustandsmatrix des vergangenen Zeitschritts  $h_{t-1}$  berücksichtigt wird. Die folgende Gleichung stellt diese Beziehung dar:

$$y_t = f(x_t, h_{t-1}) \quad (2.17)$$

Es kann ebenfalls ausgedrückt werden, dass die innere Zustandsmatrix  $h$  zum Zeitpunkt  $t$  von den vergangenen Werten der Zustandsmatrix  $h_{t-1}$  und den aktuellen Eingabewerten  $x_t$  abhängt:

$$h_t = f(x_t, h_{t-1}) \quad (2.18)$$

Dieser rekursive Ansatz benötigt einen Aktualisierungsschritt (update) der netzinternen Zustandswerte und wird in den meisten Anwendungsfällen mit der Hyperbeltangens oder lateinisch Tangens hyperbolicus ( $\tanh$ ) Funktion durchgeführt. Dabei kann der Updateschritt als reguläre **NN** Operation bezeichnet werden, welcher eine Weightmatrix  $W_{hh}^T$  mit der Zustandsmatrix  $h_{t-1}$  des vergangenen Zeitschritts multipliziert und eine weitere Weightmatrix  $W_{xh}^T$  multipliziert mit den aktuellen Eingabewerten  $x_t$  addiert:

$$h_t = \tanh(W_{hh}^T * h_{t-1} + W_{xh}^T * x_t) \quad (2.19)$$

In 2.7 ist dieser Verarbeitungsprozess dargestellt. Die Begriffe *folded* und *unfolded* der schematischen Darstellung beziehen sich auf die Auffaltung der Gleichung durch wiederholte Anwendung, Goodfellow et al. [9], Ava Soleimany [3].

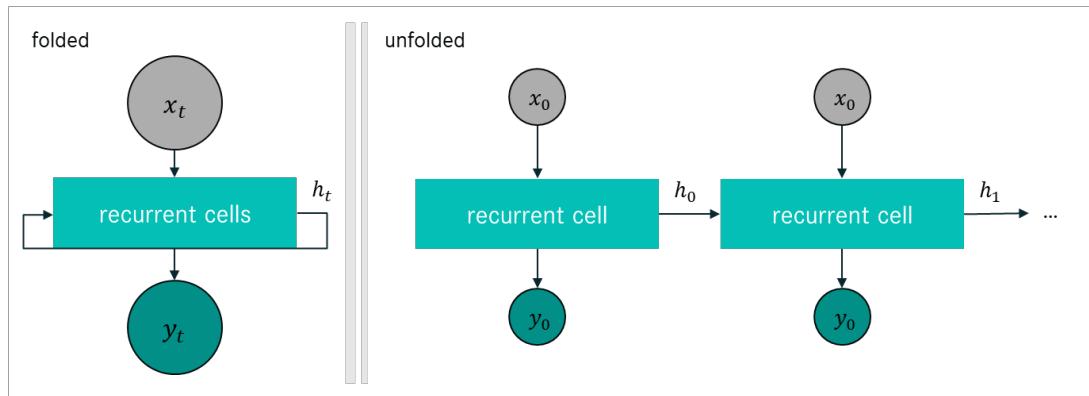


Abbildung 2.7: Schematische Darstellung, Neuron mit Recurrence (Rückkopplung)<sup>1</sup>

Der zusätzliche benötigte Speicher der Zustandsmatrix  $h$  wird auch als recurrent cell (teilweise auch als Gedächtnis oder Arbeitsspeicher) bezeichnet. Durch einen entspre-

---

<sup>1</sup> nach Ava Soleimany [3], Deep Sequence Modeling: MIT 6.S191

chenden Einsatz des inneren Zustandes können bewiesenermaßen Lösungen für die grundsätzlichen zeitlich gegliederten Problemstellungen ermittelt werden.

### Long Short-Term Memory (LSTM)

Hochreiter et al. [11] stellt eine Long Short-Term Memory (**LSTM**) Einheit (auch als Cell bezeichnet) vor, eine Variantenentwicklung auf Grundlage der gated **RNNs**. Gated **RNNs** oder auch Gated Recurrent Units (**GRUs**) adressieren die folgenden Probleme simpler **RNNs**. In der Regel werden **RNNs** durch Backpropagation trainiert, wobei entweder ein verschwindendes oder ein explodierendes Gradientenproblem auftreten kann (vanishing gradient problem). Dieser Effekt führt dazu, dass die Gewichte des Netzes entweder sehr klein oder sehr groß werden, was die Effektivität bei Anwendungen einschränkt, bei denen das Netz langfristige Beziehungen lernen soll.

Um dieses Problem zu lösen, nutzen **LSTMs** zusätzliche Gatter, um zu steuern, welche Informationen ( $h_t$ : Hidden state) der verborgenen Zelle für den nächsten Zeitschritt übergeben werden, Jonas Knupp [14]. Zusätzlich zu den Hidden state besteht die Architektur daher typischerweise aus einer Speicherzelle ( $g$ : Memory cell), einem Eingangsgatter ( $i$ : Input gate), einem Ausgangsgatter ( $o$ : Output gate) und einem Vergessensgatter ( $f$ : Forget gate).

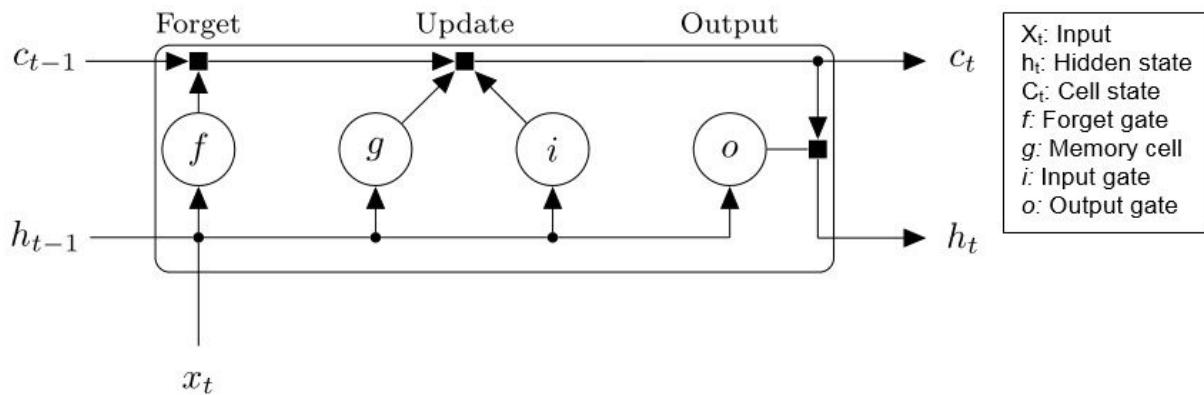


Abbildung 2.8: **LSTM** Architektur, Gatter zur Steuerung des Informationsflusses<sup>1</sup>

Das bereits vorgestellte Backpropagation-Trainingsverfahren wurde für **RNNs** entsprechend um die zeitliche Dimension erweitert. Das Backpropagation Through Time (**BPTT**) Trainingsverfahren ermöglicht die Entfaltung eines unfolded **RNNs** über die zurückliegenden Zeitschritte einer Input-Zeitreihe, indem das Netzwerk durch einen azyklischen Graphen (zurückliegender und aktueller Zeitschritt) trainiert wird, Jonas Knupp [14].

<sup>1</sup> <https://de.mathworks.com/help/deeplearning/long-short-term-memory-networks.html>, (Stand: 22.08.2023)

## Bidirectional Long Short-Term Memory Netzwerk (**BILSTM**)

Wie bereits erläutert kann das vanishing gradient problem durch den Einsatz von **LSTM** Schichten gelöst werden. Nach Hochreiter et al. [11] ermöglicht das Netzwerk damit einen konstanten Fehlerfluss entlang der Eingangssequenz. Ein konstanter Fehler bedeutet auch, dass das Netz besser in der Lage ist, langfristige Abhängigkeiten über die Eingabesequenz zu lernen. Durch die Kombination der Ausgänge von zwei **LSTM** Schichten, die die Informationen in entgegengesetzte Richtungen weitergeben, beschreibt Hochreiter et al. [11] die Möglichkeit, den Kontext von beiden Enden der Sequenz zu erfassen. Die daraus resultierende Architektur wird als Bidirektionales / Bidirectional Long Short-Term Memory Netzwerk (**BILSTM**) bezeichnet.

Zusammenfassend lässt sich festhalten, dass Gated Recurrent Units (**GRUs**) eine potentielle Methode für ein **ML** basiertes Trackingverfahren darstellen. Unter korrektem Einsatz kann eine **LSTM** / **BILSTM** Zelle folgende Anforderungen erfüllen und wird daher als **ML** Modell für die Entwicklung eines Multi Object Tracking (**MOT**) Verfahrens ausgewählt. Diese sind in der Lage:

- eine Funktion beliebiger Ordnung abzubilden
- Sequenzen unterschiedlichster Länge zu verarbeiten (zeitliche Existenz eines Tracks variiert je nach Situation)
- Kontextinformationen zu lernen (Trackabhängigkeiten über die zeitliche Abfolge zu erkennen)

In den beiden abschließenden Kapiteln der theoretischen Grundlagen, werden Begriffe und Verfahren aus dem Bereich Objekttracking eingeführt mit dem Fokus auf **ML** basierte Ansätze. Die Ergebnisse der Recherche zu aktuellen **ML** basierten Trackingverfahren werden in 2.3 und 2.4 vorgestellt. Einige der für diese Arbeit als relevant bewerteten Ansätze werden diskutiert und die Ergebnisse für die Entwicklung in Kapitel 3 genutzt.

## 2.3 Objekt Verfolgung / Single Object Tracking (SOT)

Die Aufgabe der Objekt Verfolgung bzw. Object-Tracking auf Basis von umfeldserfassenden Sensoren kann laut Mitchell et al. [25] im allgemeinen so definiert werden, dass ein Zielobjekt (Target) pro Zeitschritt den bereits erfassten Zuständen eines vergangenen Zeitschritts zugeordnet wie auch zukünftige Zustände prognostiziert werden. Ein Single Object Tracking (**SOT**) Verfahren wird Leal-Taixé et al. [18] dann eingesetzt, wenn mit einem oder mehreren Sensoren, ein Objekt im Umfeld eines Systems über den Bereich des physikalischen Erfassungsraums verfolgt werden soll. Für die initiale Erkennung eines Objektes werden in vielen Anwendungen Detektionsverfahren eingesetzt, um beispielsweise mit Hilfe einer Kamera einen Fußgänger in einer Bildaufnahme (Frame) zu erkennen (**TbD**). Je nach Umgebung und Störeinflüsse kann die Implementierung eines Verfahrens zur Lösung des **SOT** Problems entsprechend herausfordernd sein. Im Gesamtkontext einer Echtzeitanwendung ist das Tracking in Situationen wichtig in denen die Detektions-Verfahren kurzzeitig keine Informationen mehr liefern können, beispielsweise bei Verdeckungen und Störungen Leal-Taixé et al. [18], Leal-Taixé et al. [19]. Daher lässt sich die Idee des Trackings auch so formulieren, dass auf der Basis von aktuell erfassten Objekten und deren Zustand (ID, Position und gerichtete Geschwindigkeit im Raum) ein zukünftiger Zustand prädiziert werden soll. Betrachten wir eine Objekt Verfolgung als Anwendung für Advanced Driver Assistance Systems (**ADASs**) muss ebenfalls hervorgehoben werden, wie wichtig eine genaue Schätzung der Zustände der erfassten Objekte im Bereich des Fahrzeuges ist, da es sich um dynamische Systeme handelt, welche auf Basis dieser Zustandswerte entsprechende Sicherheitsfunktionen aktivieren.

Die folgenden Unterkapitel stellen eine Übersicht über einige aktuelle Deep Neural Networks (**DNNs**) basierten Tracking Verfahren dar. Es existieren eine Vielzahl von Lösungsverfahren, welche von der Industrie und Wissenschaft entwickelt wurden. Es wird gezeigt, dass **DNN** basiertes **SOT** als Assoziationsproblem (matching problem) oder auch als Problem der zeitlichen Vorhersage (temporal prediction problem) aufgestellt und entsprechend gelöst werden kann. Damit soll die Betrachtungsweise für die zu behandelte Problemstellung erweitert und mögliche Lösungsansätze mit dieser Wissensbasis entwickelt werden.

### 2.3.1 SOT als Assoziationsproblem

Bei dem Verfahren mit dem Namen GOTURN wird die Objekt Verfolgung von Savarese et al. [28] als Assoziationsproblem (matching problem) betrachtet. GOTURN nutzt Convolutional Neural Networks (CNNs) um die Bewegung eines Objekts offline zu lernen. Das GOTURN-Modell wird auf Tausenden von Videosequenzen trainiert und muss zur Laufzeit keinen Lernprozess durchführen. Der Ansatz hinter dem Modell ist eine simple Idee, um ein effizientes Tracking auf Bildebene zu realisieren. Das Modell versucht in zwei aufeinanderfolgenden Frames, innerhalb der definierten Teilbereiche (Crop), das selbe Objekt zu erkennen. Das Verfahren nutzt dafür zwei exakt identische Conv. Layers, welche den aktuellen Crop in Frame ( $t$ ) und den Crop des vorherigen Frames ( $t - 1$ ) verarbeiten. Das bedeutet, dass beide Conv. Layers identische Umrechnungen durchführen und somit für das gleiche Objekt gleiche Featurewerte berechnet werden. Der Crop dient dabei dazu den Suchbereich bzw. die Matrix für die Assoziation zu verringern. Die Assoziation wird von Fully-Connected Layers auf den beschränkten Bereich (Crop) durchgeführt. Die Tatsache, dass das Modell zum Zeitschritt  $t - 1$  den Suchbereich (Crop) definiert und im Zeitschritt  $t$  das Objekt entsprechend zuordnend, führt zu der Restriktion, dass die maximal Geschwindigkeit verfolgbarer Zielobjekte sehr stark von der Bildaufnahmerate der Kamera abhängig ist.

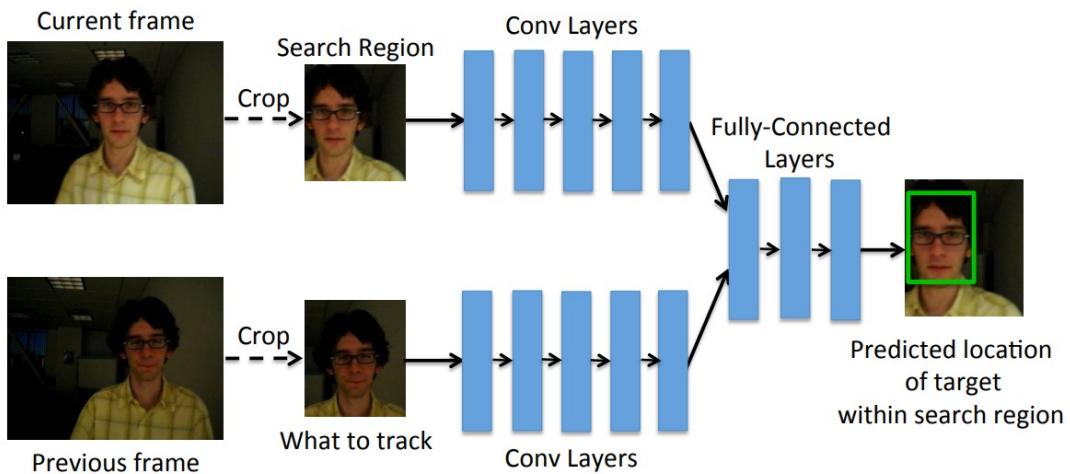


Abbildung 2.9: Schematisch dargestellte Architektur GOTURN Modell<sup>1</sup>

Efros et al. [7] beschreibt SOT ebenfalls als Assoziationsproblem. Der Ansatz des Verfahrens zielt darauf, dass eine zeitliche Verfolgung eines Objektes unabhängig ob diese vorwärts oder rückwärts gerichtet ist, das gleiche Ergebnis darstellen sollte. Wird beispielsweise die Position der im Bild dargestellten Person verfolgt, sollte sowohl im

<sup>1</sup> aus Savarese et al. [28] *Learning to Track at 100 FPS with Deep Regression Networks*

Forward-Track [ $t-2 \rightarrow t-1 \rightarrow t$ ] wie auch im Back-Track [ $t \rightarrow t-1 \rightarrow t-2$ ] die entsprechende aktuelle bzw. vergangene Position übereinstimmen. Das Verfahren ermöglicht es auf dieser Basis die Distanz differenz der Vorhersage zur detektierten Position zu berechnen. Die ermittelte Differenz stellt den Fehler der Vorhersage dar und wird als Loss genutzt um ein neuronales Netz zu trainieren. Dieses Verfahren nutzt somit eine nicht angeleitete (unsupervised) Trainingsmethode.



Abbildung 2.10: Assoziationsprinzip des Verfahrens<sup>1</sup>

### 2.3.2 SOT als Problem der zeitlichen Vorhersage

Wang et al. [33] zeigt eine weitere Betrachtungsvariante des Tracking Problems. Das vorgestellte Verfahren setzt dabei auf die Trennung der räumlichen Detektion und der zeitlichen Evolution eines Objektes. Die Autoren dieser Arbeit beschreiben die Möglichkeit pro Inputframe mit einem CNN visuelle Merkmale (Features) zu extrahieren und auf Basis dieser extrahierten Features mit Hilfe eines Long Short-Term Memory (LSTM) die Zustandsänderungen zu modellieren. Die Methode trägt die Bezeichnung Recurrent YOLO (ROLO), da das Modell einen YOLO-Detektor für die Featureextraktion und ein rekurrentes Netzwerk nutzt (siehe Kapitel 2.2.3).

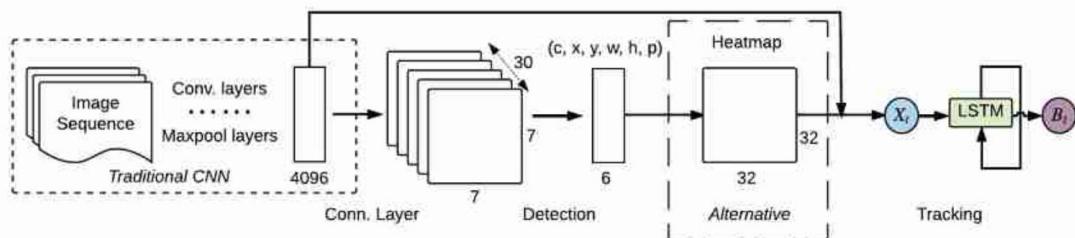


Abbildung 2.11: Architekturvorschlag der ROLO Modell Entwickler<sup>2</sup>

<sup>1</sup> aus Efros et al. [7] *Learning Correspondence from the Cycle-Consistency of Time*

<sup>2</sup> aus Wang et al. *Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking*

### 2.3.3 Erkenntnisgewinne

Die vorgestellten Verfahren zeigen einen Auszug der Variantenvielfalt an **DNN** basierten Lösungen zum Tracken eines einzelnen Objektes. Die in Tabelle 2.1 zusammengefassten Ergebnisse der Recherche beinhalten die Bewertung des Einsatzes eines ähnlichen Verfahrens in dieser Arbeit. Die Ebene des Trackings ist dabei entscheidend, da im Rahmen dieser Arbeit ein Verfahren auf der Objektebene entwickelt werden soll. Die beiden Verfahren nach Savarese et al. [28] und Efros et al. [7] nutzen den hohen Informationsgehalt auf der Bildebene und ermöglichen ein Tracking durch die Auswertung einer jeweils geschickt aufgebauten **CNN** basierten Verarbeitungspipeline. Hingegen ist das **ML** Verfahren, welches in ROLO nach Wang et al. [33] die Aufgabe des Tracking übernimmt ein Long Short-Term Memory (**LSTM**). Ein Modellansatz, welcher auch von einigen im nächsten Kapitel vorgestellten Multi Object Tracking (**MOT**) Systemen eingesetzt wird.

Tabelle 2.1: **SOT** Verfahren, Zusammenfassung

Einsatzgebiet	Bezeichnung des Verfahrens	Ebene des Trackings	Machine Learning Verfahren
SOT	GOTURN [28]	Bildebene	<b>CNN</b>
SOT	LCFTCOT [7]	Bildebene	<b>CNN</b>
SOT	ROLO [33]	Objektebene [Alternativ: Featureebene]	<b>LSTM</b>

## 2.4 Multi Objekt Verfolgung / Multi Object Tracking (MOT)

Nachdem die Grundlagen der Sensordatenfusion und das SOT inklusive einer Auswahl an ML basierten Verfahren vorgestellt wurden, wird in diesem Abschnitt der Begriff der Multi Objekt Verfolgung bzw. Multi Object Tracking (MOT) eingeführt. MOT ist ein Bereich der Computer Vision (CV), ADAS und Robotik, der sich nach Moon et al. [26] im Gegensatz zu SOT mit der Überwachung und Verfolgung von mehreren Objekten in einer Szene befasst. Bedingt durch die Anzahl mehrerer Objekte entstehen neue Herausforderungen für ein Tracking Verfahren, welche von Leal-Taixé et al. [18] beschrieben und in diesem Kapitel thematisiert werden.

### 2.4.1 MOT Grundlagen

Yang et al. [39] zeigt, dass Multi Object Tracking (MOT) Verfahren neben dem Einsatz in ADAS, in vielen Anwendungen verwendet werden, wie in der Überwachungssoftware von Sicherheitskameras, im Sport und in der Unterhaltungsbranche. In jedem dieser Bereiche ist es wichtig, die Bewegungen und Interaktionen von mehreren Objekten in Echtzeit zu verfolgen und zu überwachen, um relevante Informationen zu sammeln und entsprechende Entscheidungen zu treffen, bzw. Signale zu senden. Leal-Taixé et al. [18] zeigt weiter, das MOT-Systeme auf verschiedene Arten realisiert werden können, abhängig von den verfügbaren Sensoren und dem spezifischen Anwendungsfäll. Beispielsweise kann ein System auf der Verwendung von Daten aus einer einzigen Kamera basieren, während ein anderes System Daten von mehreren unterschiedlichen oder gleichartigen Sensoren kombiniert. Ein wichtiger Aspekt von MOT ist die Datenassoziation, d.h. die Verbindung von Datenpunkten, die von einem oder verschiedenen Sensoren gemessen werden, mit den erfassten physischen Objekten herzustellen. Insgesamt ist MOT ein aktiver Forschungsbereich, unter anderem für Advanced Driver Assistance Systems (ADASs) und das automatisierte Fahren.

An dieser Stelle ist zu erwähnen, dass die Forschung und Anwendung von MOT Verfahren seit 2019 ein Paradigmenwechsel erleben (meist Tracking auf Bildebene). Der Trend, welcher in Leal-Taixé et al. [19] von der Technische Universität München (TUM) untersucht wurde zeigt, dass Tracking Verfahren, welche das Tracking by Regression (TbR) Paradigma verwenden, eine bessere Performance bei den evaluierten MOT Challenge Datensätzen aufweisen. Ein weiteres Paradigma Tracking by Attention (TbA) hat wie TbR das Ziel die Detektion und Tracking Aufgabe in einem kombinierten Berechnungsverfahren zu lösen, jedoch auf Basis von unterschiedlichen ML Methoden (für weitere

Details siehe Leal-Taixé et al. [18]). Auf Grund der Zielsetzung wird jedoch in dieser Arbeit ausschließlich das **TbD** Paradigma verwendet und entsprechend eingeführt.

**Tracking by Detection (**TbD**) Paradigma** Die Entwicklung eines effizienten und zuverlässigen Tracking-Systems, das in der Lage ist, die Bewegungen und Interaktionen von mehreren Objekten in Echtzeit zu verfolgen und zu überwachen, stellt nach wie vor eine Herausforderung dar. Die meisten der entwickelten **MOT** Methoden nutzen laut Leal-Taixé et al. [18] das Tracking by Detection (**TbD**) Paradigma. Dieses Paradigma besteht im Wesentlichen aus zwei Schritten, der Erkennung und Verfolgung. Wird eine Kamera für die Umfeldserfassung verwendet, können die Teilaufgaben wie folgt beschrieben werden:

- In der Erkennungsphase werden alle Objekte in jedem Frame einer Bildsequenz erkannt. Dabei werden üblicherweise Objektdetektoren eingesetzt, die auf Basis von **ML** Techniken trainiert wurden. Diese können zum Beispiel auf Basis von Merkmalen wie Form, Farbe oder Textur lernen, wie sie ein bestimmtes Objekt in einem Bild erkennen können.
- In der Verfolgungsphase werden dann die Objekte, die in einem Frame erkannt wurden, im nächsten Frame durch die Verwendung von Verfolgungsalgorithmen wie Kalman Filter oder Partikel-Filter weiterverfolgt. Das Ziel ist es, sicherzustellen, dass jedes Objekt korrekt identifiziert und verfolgt wird, wenn es durch das Video oder die Bildsequenz bewegt wird.

Ein für diese Arbeit entscheidender Vorteil des **TbD** Paradigma ist, dass es flexibel mit verschiedenen Detektionsalgorithmen kombiniert werden kann. Dies bedeutet, dass der Programmteil der Detektion nicht entwickelt werden muss bzw. die entsprechenden Erkennungen (Sensorobjekte) den Erkennungsteil übernehmen können. Insgesamt bietet das Tracking-by-Detection Paradigma eine effektive Möglichkeit zur automatisierten Verfolgung von Objekten. Es ist flexibel, robust und kann eine große Anzahl von Objekten verfolgen.

## 2.4.2 MOT Onlineverfahren

Eine weitere Einteilung im Bereich der Entwicklung von Methoden für die Objekt Verfolgung kann nach Leal-Taixé et al. [17] [18] und im Bezug auf die Anwendung durchgeführt werden. Online Tracking-Verfahren werden zur Echtzeit-Verfolgung mehrerer Objekte in einer Videosequenz oder einem Datenstrom eingesetzt. Im Gegensatz zu Offline-Tracking-Verfahren, die die gesamte Videosequenz analysieren, um alle Objekte gleichzeitig zu verfolgen, funktionieren Online-Tracking-Verfahren, indem sie jedes Frame bzw. Sensorobjekt einzeln analysieren und versuchen, die Objekte zu verfolgen, während sie sich durch die Sequenz bewegen.

Ein wichtiges Merkmal von Online MOT Verfahren ist nach Yang et al. [38] und Wang et al. [34], dass sie in Echtzeit arbeiten (Real-Time, wird in vielen MOT Verfahren auch mit 'Online' bezeichnet) Das bedeutet, dass sie in der Lage sind, eine Sequenz in Echtzeit zu verarbeiten und die Objekte zu verfolgen, während sie sich bewegen. Dies ist besonders wichtig in der in dieser Arbeit behandelte Anwendung. Ein Objekt-Tracking für ADAS benötigt eine Echtzeit-Verfolgung, um eine schnelle Reaktion auf die Umgebung zu gewährleisten zu können. Ein weiteres Merkmal von Online MOT Verfahren ist ihre Fähigkeit zur inkrementellen Aktualisierung der Verfolgungsinformationen. Wenn neue Objekte in einer Sequenz eintreffen, verwenden Online-Tracking-Verfahren die Informationen aus den vorherigen Informationen, um die Verfolgung jedes Objekts zu aktualisieren und zu verbessern. Dadurch sind sie in der Lage, Objekte auch dann zu verfolgen, wenn sie teilweise oder vollständig vom Sichtfeld verdeckt werden.

Nach den umfassenden Zusammenfassungen von Moon et al. [26] und Yang et al. [39] existieren verschiedene Ansätze und Techniken, die in Online MOT Verfahren eingesetzt werden, darunter Kalman Filter (KF), Partikel Filter und Machine Learning (ML) Methoden. Die Wahl des geeigneten Algorithmus hängt von verschiedenen Faktoren ab, wie der Art der Datenquelle, der Anzahl der zu verfolgenden Objekte und der Anforderungen an die Echtzeit-Verarbeitung. Im nachfolgenden Teilabschnitt werden einige Arbeiten aus diesem Bereich vorgestellt, welche auf ML basieren.

### Machine Learning (ML) basierte Tracking Verfahren

Für die Recherche wurden unter anderem Suchmaschinen für wissenschaftliche Open-Acccess-Publikationen ([scholar.google.com](http://scholar.google.com), [connectedpapers.com](http://connectedpapers.com), [ieeexplore.ieee.org](http://ieeexplore.ieee.org)) und Citavi als Programm zur Literaturverwaltung und Wissensorganisation genutzt. Wissenschaftliche Zusammenfassungen aus dem Bereich Multi Object Tracking (MOT) (Fokus auf Machine Learning (ML) bzw. Deep Learning (DL) Verfahren) wurden ebenfalls als nützliche Quellen herangezogen, Moon et al. [26], Yang et al. [39]. Recherchergebnisse im Bereich der Tracking Verfahren, welche ML Methoden einsetzen sind in 2.12 dargestellt. Inspiriert von einer in Yang et al. [39] illustrierten Übersicht wurde die Kategorisierung der untersuchten MOT Verfahren entsprechend gewählt. Tracking Verfahren stützen sich oftmals auf einer Vielzahl an Modellen und Berechnungsverfahren, eine eindeutige Trennung ist daher nicht immer möglich. Die in farbig dargestellten Kreise stellen die grundsätzliche Trackingmethode der Verfahren dar. Auch die in Kapitel 2.3 vorgestellten Trackingansätze, werden teilweise in den Zusammenfassungen Moon et al. [26], Yang et al. [39] genannt und als modifizierbare Variante ebenfalls der Convolutional Neural Network (CNN) und Recurrent Neural Network (RNN) Methode zugeordnet (Einführung der DL Methoden in Kapitel 2.2.1).

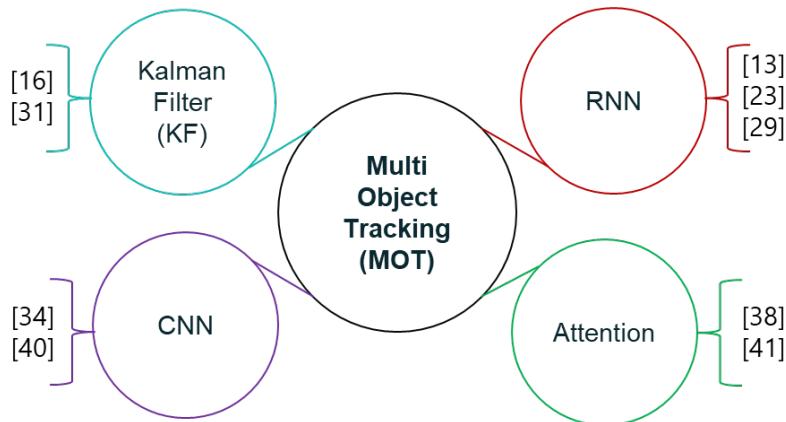


Abbildung 2.12: Kategorisierung einiger MOT Verfahren<sup>1</sup>

**Kalman Filter (KF)** nach Leal-Taixé et al. [16] und Upcroft et al. [31];

**CNN** nach Wang et al. [34] und Yu et al. [40];

**Attention** nach Yang et al. [38] und Yu et al. [41];

**RNN** nach Johannes Fitz et al. [13], Mertz et al. [23] und Schindler et al. [29]

Die Inhalte der Arbeiten wurden entsprechend untersucht. Für diese Arbeit als besonders relevant bewertete Verfahren sind im abschließenden Teilkapitel 2.4.3 näher beschrieben.

<sup>1</sup> nach Moon et al. [26] *Multiple Object Tracking in Deep Learning Approaches: A Survey*

### 2.4.3 Erkenntnisgewinne

Die Rechercheergebnisse zeigen, dass **ML** basierte **MOT** Verfahren, welche nach dem Tracking by Detection (**TbD**) Paradigma entwickelt wurden und somit auf der Sensor Objekt (**SO**) Ebene arbeiten Recurrent Neural Networks (**RNNs**) nutzen (Einführung in Kapitel 2.2.3).

Schindler et al. [29] stellt einen Ansatz für ein online Multi Object Tracking (**MOT**) Verfahren vor, welches auf Recurrent Neural Networks (**RNNs**) basiert. Nach eigenen Angaben liegt der Hauptvorteil dieses Ansatzes darin, dass keinerlei Dynamikmodelle implementiert werden müssen und daher lineare (vgl. Kalman-Filter), nicht lineare (vgl. Partikelfilter) und Abhängigkeiten höherer Ordnung erfasst werden können (in Abhängigkeit der verfügbaren Trainingsdaten).

Mertz et al. [23] beschreibt ein **LSTM** basiertes Assoziationsnetzwerk, welches für das Multi Object Tracking (**MOT**) in Umgebungen mit Störungen (Clutter) entwickelt wurde. Das Hauptziel des DeepDA Verfahrens besteht darin, die Zuordnung von Messungen zu den jeweiligen verfolgten Objekten zu verbessern und dabei die Herausforderungen durch Störungen und überlappende Objekte zu bewältigen. Als Inputdaten für das entwickelte Netzwerk bildet [23] eine Distanzmatrix auf Basis des euklidischen Abstandsmaßes. Das DeepDA-Netzwerk ersetzt somit einen Assoziationsalgorithmus wie z.B. den Hungarian Algorithmus. Es ist anzunehmen, dass bei der Erstellung der Groundtruth (**GT**) Trainingsdaten (Distanzmatrizen) sowie bei der Durchführung der Evaluierung das euklidische Abstandsmaß als Basisberechnung verwendet wurde.

In beiden Arbeiten wird die grundlegende Möglichkeit der datenbasierten Funktionsapproximation eines Deep Learning (**DL**) Netzwerks als geeignetes Mittel für ein Tracking Verfahren gewählt. Beide Verfahren verarbeiten Daten auf der Sensor Objekt (**SO**) Ebene und eignen sich daher als Ansätze für diese Arbeit.

Im nachfolgendem Kapitel 3 werden auf Basis der gewonnenen Erkenntnisse dieser Recherche entsprechende Netzwerke entwickelt, welche definierte Teilaufgaben des Multi Object Tracking (**MOT**) lösen sollen. Die Entwicklungen der Netzwerkaufgaben richtet sich dabei an den **MOT** Framework, welcher in Kapitel 3 eingeführt und dessen Funktionsweise in Kapitel 3.6 näher beschrieben wird.

# 3 Ergebnisse

Zu Beginn dieses Kapitels ist in 3.1 das geplante Vorgehen aufgeführt. Zudem beinhaltet es die Auflistung der Anforderungen und Einschränkungen an die Entwicklung **ML** basierter **MOT** Verfahren innerhalb dieser Arbeit. Weiter wird beschrieben welcher Logik der Aufbau dieses Kapitels folgt.

## 3.1 Vorgehensweise und Anforderungen

Ziel dieser Arbeit ist es **ML** Ansätze im Anwendungsgebiet des **MOT** zu untersuchen und Vorschläge für den Einsatz in einem bestehenden Framework zu entwickeln. Ba-Tuong Vo et al. [5] stellt einen Framework für die Untersuchung von Tracking Ansätzen zur Verfügung, welcher bereits als erweiterte Version mit Kalman Filter (**KF**) als Basis für die Entwicklungsumfänge vorliegt. Damit bietet der erweiterte Framework die Möglichkeiten **MOT** Verfahren zu entwickeln bzw. Filter mit definierten Schnittstellen miteinander zu vergleichen, welche nach dem **TbD** Paradigma konzipiert wurden. Abbildung 3.1 zeigt einer schematische Übersicht der einzelnen Programmmodulen. Eine detaillierte Beschreibung der Funktionsweise erfolgt in 3.6.2.

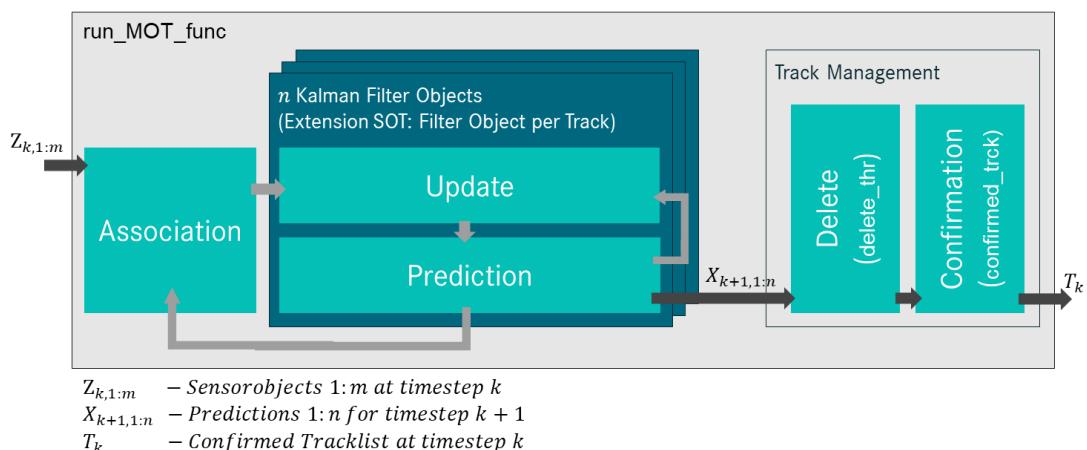


Abbildung 3.1: Schematische Darstellung der Architektur des genutzten **MOT** Frameworks

Die Entwicklung der einzelnen Netzwerke sowie der logische Aufbau dieses Kapitels richtet sich nach dem in Abbildung 3.1 dargestellten modularem Framework. Die schematischen Darstellung 3.1 zeigt, dass die Schätzung der Zustandsänderung eines

erfassten Tracks (Prediction), wie auch die Assoziation von Messungen zu bestehenden Tracks (Association) und die entsprechende Korrektur der prädizierten Zustände (Update) als eigener Funktionsblock betrachtet wird. So wird in Kapitel 3.3 ein Netzwerk entwickelt, mit dem Ziel die Aufgaben des Prediction Moduls zu lösen. Anschließend werden in 3.4 und 3.5 Netzwerkentwicklungen vorgestellt, welche das Ziel verfolgen eine datenbasierte Assoziationsfähigkeit zu erlernen. Die Netzwerke stellen somit einen jeweiligen Modulteil des MOT Frameworks dar. Innerhalb der entsprechenden Kapitel werden diese auf ihre Performance geprüft. Nach erfolgreicher Verifikation der Einzel-Netzwerk-Performance werden diese in 3.6 in den MOT Framework integriert und die implementierten Ansätze entsprechend evaluiert.

Die Strategie der Aufteilung des Gesamtproblems bietet dabei mehrere Vorteile für die Entwicklung von Multi Object Tracking (MOT) Verfahren:

- Einzelne Komponenten können während der Konzeptentwicklung effektiv isoliert und debuggt werden.
- Es wird ein modularer Rahmen mit definierten Schnittstellen eingehalten, welcher einen Austausch der Module (Funktionsblöcke) ermöglicht.
- Bezogen auf den Einsatz von Netzwerken für das jeweilige Teilproblem kann so jeder Block separat trainiert und eine generalisierte Lösung des Problems entwickelt werden. Das kann zur Beschleunigung des Lernprozesses führen bzw. notwendig sein um eine Konvergenz zu ermöglichen.

Abschließend sind in diesem Unterkapitel die Rahmenbedingungen für die Entwicklung eines MOT Verfahrens mit Machine Learning (ML) aufgelistet, um eine entsprechende Einordnung des Vorgehens zu ermöglichen.

#### **Anforderungen und Einschränkungen an die Entwicklung eines MOT Verfahrens mit Machine Learning (ML):**

- **Framework Kompatibilität** - die entwickelten ML basierten Ansätze müssen mit dem bestehendem Bays'schen Filter Framework kompatibel sein. Lösungsvorschläge für den Einsatz von ML wurden daher mit gegebenen Schnittstellen entwickelt, welche eine Integration der Algorithmen bzw. Modelle ermöglichen.
- **Tracking-by-Detection** - das Tracking findet auf der SO Ebene statt. Für das Training von entsprechenden ML Ansätzen wurden daher die Tracklets des KITTI Datensatzes (Einführung Folgekapitel) als Groundtruth (GT) Daten extrahiert und verrauscht als Sensorobjekte SO interpretiert.

- **Sensorobjekttypen / -klassen** - die Anzahl der Sensorobjektklassen wurde auf Grund der Datenverfügbarkeit auf PKW Fahrzeuge limitiert (KITTI Datenbezeichnung: 'Car' und 'Van').
- **Trackanzahl** - aktuelle State-of-the-Art Trackingverfahren in **ADAS** begrenzen die Anzahl an bestätigten Tracks auf Grund von Arbeitsspeicher- und / oder Rechenzeitlimitierungen in der Regel. Falls der KITTI Datensatz eine Anzahl von 30 Tracks pro Verarbeitungsschritt überschreitet, wird dieser Wert als Limit eingeführt.
- **Sensorobjektanzahl** - aus den selben Gründen ist in der Regel die Anzahl an neuen Beobachtungen pro Verarbeitungsschritt begrenzt. Falls der KITTI Datensatz eine Anzahl von 25 Sensorobjekten **SO** pro Verarbeitungsschritt überschreitet, wird auch entsprechend limitiert.
- **Evaluierung** - die offline Evaluierung wird mithilfe der Optimal Sub-Pattern Assignment (**OSPA**) Metrik durchgeführt, welche im Kapitel [3.6](#) eingeführt wird. Die **OSPA** Metrik ist eine gängige Bewertungsmetrik, die speziell für die Evaluierung von Multi Object Tracking (**MOT**) Systemen entwickelt wurde.

## 3.2 KITTI Datensatz

Für die Konzeptentwicklung des **ML** basierenden **MOT** Verfahrens wurde in dieser Arbeit der frei verfügbare KITTI Datensatz von Andreas Geiger et al. [32] verwendet. Er wurde vom Karlsruher Institut für Technologie (**KIT**) und der Toyota Technological Institute at Chicago (**TTIC**) entwickelt, um die Forschung im Bereich der autonomen Fahrzeuge und **ADAS** Funktionen zu unterstützen. Der KITTI-Datensatz besteht aus einer umfangreichen Sammlung von Sensordaten, die von einem speziell ausgestatteten Fahrzeug aufgenommen wurden und bietet insgesamt 6 Stunden Verkehrsszenarien. Die Daten wurden mit einer Vielzahl von Sensoren wie hochauflösende Farb- und Graustufen-Stereokameras, einem Velodyne 3D-Laserscanner und einem hochpräzisen Global Positioning System (**GPS**) / Inertial Measurement Unit (**IMU**) Trägheitsnavigationssystem mit einer Frequenz von 10-100 Hz aufgezeichnet. Die einzelnen Messdateien bilden reale Verkehrssituationen ab und reichen von Autobahnen über ländliche Gebiete bis hin zu innerstädtischen Szenen mit vielen statischen und dynamischen Objekten. Die Daten sind kalibriert, synchronisiert, mit einem Zeitstempel versehen und enthalten Objektbeschriftungen in Form von 2D und 3D Tracklets. Somit eignen sich die Messdaten für die Entwicklung von Funktionsalgorithmen auf Simulationsebene. Durch die Verwendung eines standardisierten Datensatzes können verschiedene Algorithmen leicht verglichen und evaluiert werden.

**Einschränkungen des KITTI Datensatzes** Obwohl der KITTI-Datensatz eine wichtige Ressource für die Entwicklung von Algorithmen im Bereich des autonomen Fahrens und entsprechenden **ADAS** Funktionsumfängen ist, hat er auch einige Einschränkungen, die bei der Verwendung des Datensatzes berücksichtigt werden müssen.

Eine der größten Einschränkungen des KITTI-Datensatzes ist, dass er in einem begrenzten räumlichen Bereich (Karlsruhe, Deutschland) aufgenommen wurde und die Aufnahmen nur in urbanen Umgebungen gemacht wurden. Dies bedeutet, dass der Datensatz nicht unbedingt repräsentativ für andere Umgebungen ist, wie z.B. ländliche Gebiete oder andere Städte in anderen Ländern. Ebenfalls wurden die Aufnahmen ausschließlich mit konstanten Wetterbedingungen an 5 Tagen im Jahr 2011 erfasst (trocken, kein Regen oder Schnee). Der Datensatz stellt somit keinerlei Einflüsse unter schlechten Wetterverhältnissen dar. Eine weitere Einschränkung Datensatzes ist, dass er keine Szenarien enthält, die für die Entwicklung von Algorithmen **ADAS** kritisch sein können, wie z.B. Unfälle oder unerwartete Ereignisse. Außerdem sind die Szenarien in dem Datensatz meistens von einer bestimmten Perspektive aus aufgenommen worden, was möglicherweise nicht für alle Anwendungsfälle geeignet ist (Beispiel Kameraperspektive Truck). Schließlich kann die begrenzte Größe des KITTI-Datensatzes auch

### *3 Ergebnisse*

---

ein Problem sein, da **ML** Modelle in der Regel eine große Menge an Trainingsdaten benötigen. Es kann daher notwendig sein, den Datensatz zu erweitern oder mit anderen Datensätzen zu kombinieren, um bessere Ergebnisse für bestimmte Anwendungen zu erzielen.

Für die Entwicklung eines Verfahrens zur Multi Objekt Verfolgung mit maschinellem Lernen auf Ebene der Sensorobjekte ist zu erwarten, dass die Einschränkung der nicht repräsentativen Umgebung weniger schwerwiegend ausfällt als bei einer Anwendung auf Sensorrohdatenebene. Davon ist auszugehen, da nicht die Sensorrohdaten für die Entwicklung verwendet werden, sondern die Tracklets, welche die Groundtruth (**GT**) Daten der Sensorobjekte **SO** repräsentieren. Eine grundsätzliche Untersuchung bzw. Konzeptentwicklung von **MOT** Verfahren und Algorithmen ist im Rahmen dieser Arbeit auf Basis des anerkannten KITTI Datensatzes möglich. Für die Entwicklung steht der veröffentlichte Datensatz von KITTI zur Verfügung. Ein weiterer KITTI Testdatensatz kann nach der Implementierung eines Verfahrens und Veröffentlichung eines entsprechenden Abstract angefordert werden. Im Rahmen dieser Arbeit wurde ein Split des veröffentlichten Datensatzes (Training, Validierung, Test) verwendet. Die Einschränkungen des Datensatzes sollten für Folgeentwicklungen behoben werden, durch die Aufnahme von anwendungsspezifischen Daten (Truck) und globalen, zeitlich variierenden Messkampagnen.

### 3.3 Single Prediction Network (SPENT) - Schätzung der Zustandsänderung erfasster Tracks

In Anlehnung an die in Kapitel 2.4 untersuchten Multi Object Tracking (MOT) Verfahren werden entsprechende Machine Learning (ML) Konzepte entwickelt, implementiert und in den vorgestellten Tracking by Detection (TbD) Framework integriert. Dieser Abschnitt dokumentiert die Entwicklung eines Netzwerks für die Schätzung der Zustandsänderung erfasster Tracks. Zu Beginn werden die Sensor Objekte SO als Punkt Objekte (PO) betrachtet, um das Datenhandling gering zu halten. Sobald das Modell die Vorhersage von Punktobjekten mit einer ausreichend geringen Abweichung umsetzen kann, soll der Zustandsvektor erweitert werden, um die Informationsmenge bzgl. der Objektzustände von aktuellen Seriensensoren zu verarbeiten.

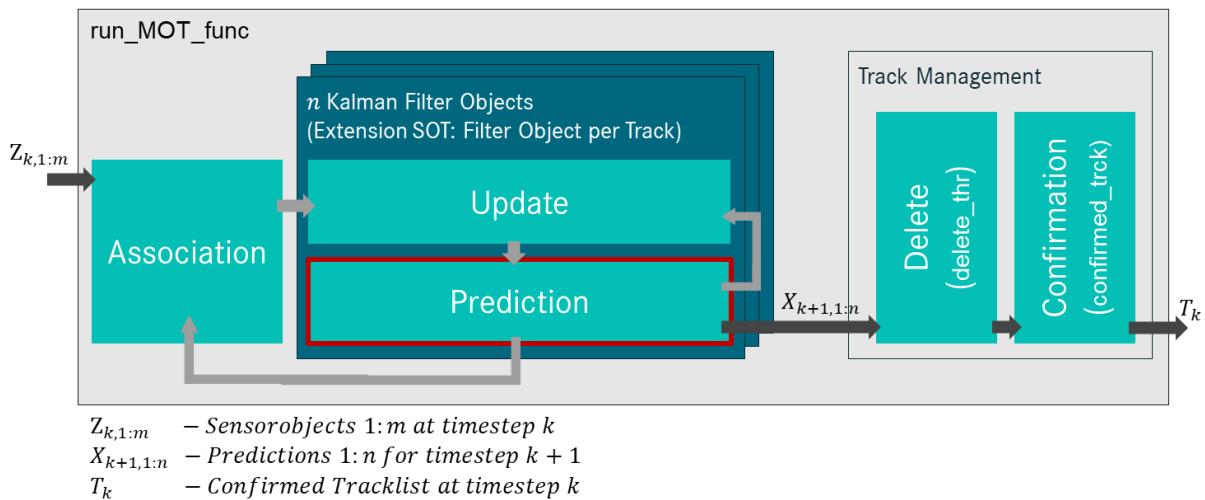


Abbildung 3.2: Schematische Darstellung, SPENT Funktionsgebiet rot markiert

Der Workflow für die Entwicklung des Single Prediction Networks (SPENTS) ist im Anhang dargestellt. Der Entwicklungsablauf ist angelehnt an den allgemein bekannten Machine Learning (ML) Workflow für supervised Learning Aufgaben und lässt sich wie folgt zusammenfassen:

- Beschaffung von Daten
- Vorverarbeitung und Aufteilung des Datensatzes
- Aufbau und Training des Modells
- Auswertung der Modellperformance
- Optimierungen bzw. Abstimmung der Hyperparameter

#### 3.3.1 Datenvorverarbeitung

In der ersten Konzeptphase wird das Ziel verfolgt, ein Netzwerk für die vorhersage der Position eines SOs relativ zum Egofahrzeug zu entwickeln. Entsprechend wurden folgende Zustandswerte pro Zeitschritt dem jeweiligen Track in Abhängigkeit der ID des gelabelten KITTI Datensatzes zugeordnet:

- x Koordinate (KITTI Kamerakoordinatensystem: z)
- y Koordinate (KITTI Kamerakoordinatensystem: -x)
- Gierwinkel (KITTI Kamerakoordinatensystem: ry)

Wie im eingeführten Workflow dargestellt, wurde die Datenvorverarbeitung für die SPENT Modellentwicklung in drei Aufgabengebiete aufgeteilt. Jede Lösung dieser Aufgabengebiete wurde in einem separaten Skript implementiert, um einen übersichtlichen und anpassbaren Preprocessing Workflow aufzubauen. Das Ergebnis der ersten Vorverarbeitung ist in Abbildung 3.3 visuell dargestellt. Es wurden pro Sequenz drei zufällig ausgewählte extrahierte Tracks in einem 3D-Plot für die visuelle Überprüfung ausgegeben. Die extrahierten Zustände (x und z Koordinate) sind pro Track auf der y und z Achse über die Zeit aufgetragen. Mit Hilfe dieser Darstellung lässt sich ein Eindruck über die Varianz des Bewegungsprofils der Tracks gewinnen. Der mittlere Plot stellt den zeitlich längsten Track der drei Ausgaben dar. Das Objekt wurde mit einem longitudinalen Abstand von ca. 40m und einem lateralen Abstand von ca. -5m im ersten Zeitschritt einer Sequenz entdeckt. Mit dem letzten Zeitschritt dieses Tracks ist der Abstand in longitudinaler / lateraler Richtung 0m / -3m. Das Objekt befindet sich zu diesem Zeitpunkt seitlich (links) neben dem Egofahrzeug (wurde vom Egofahrzeug überholt, Verifikation mittels Videosequenz).

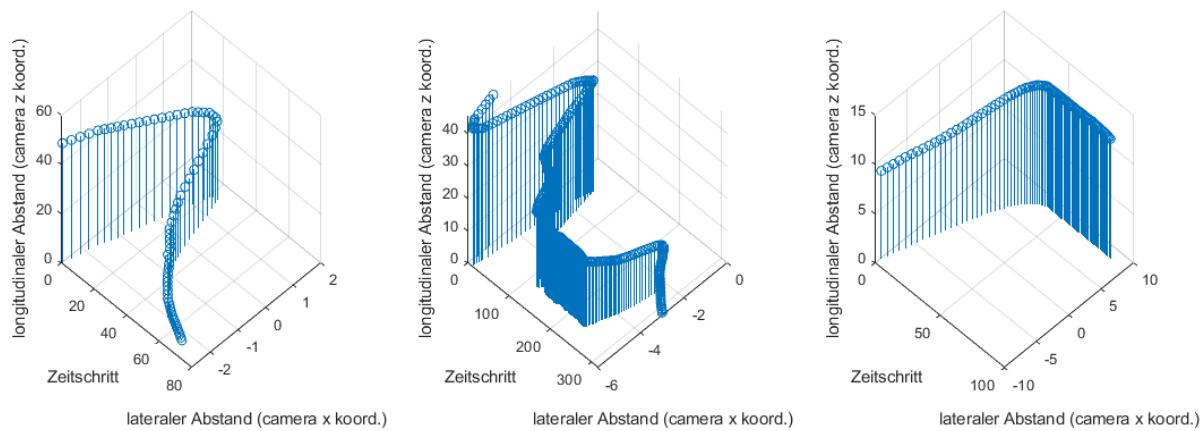


Abbildung 3.3: 3D-Plot, drei extrahierte Tracks des Gesamtdatensatzes

### 3 Ergebnisse

Betrachtet man die Tracks als mathematische Funktionen, lassen sich Anforderungen für ein entsprechendes Vorhersagemodell ableiten. Das Modell zur Zustandsvorhersage sollte somit die Fähigkeit der universellen Funktionsannäherung für zeitstrukturiertere Informationen besitzen. Wie bereits von Schindler et al. [29] beschrieben, ist eine datenbasierte Prädiktionsoptimierung, eine Sequenz-zu-Sequenz-Regression für die Zustandsvorhersage geeignet. Im zweiten Skript der Vorverarbeitung wurde ein Programm geschrieben, um alle Tracks der ausgewählten Objektklassen (Cars and Vans) pro Sequenz zu extrahieren und in einem gemeinsamen Datensatz abzuspeichern. Mit diesem iterativen Vorgehen konnten 635 Tracks (Cars and Vans) aus dem gesamten KITTI Datensatz gewonnen werden. Tracks mit einer geringen zeitlichen Länge wurden mit einem Threshold von 3 gefiltert, sodass 624 Tracks für die weitere Vorverarbeitung existieren. Die Tracklänge variiert von nun minimal 4 Frames bis zu einem Track mit 643 Frames (Sequenz 20, ID 12). Im dritten Skript des Preprocessing wird unter anderem die Varianz der Tracklänge betrachtet.

Abbildung 3.4 zeigt die Aufteilung der extrahierten Tracks inklusive Detaildarstellung in den unteren Plots. Das Kuchendiagramm gibt an, dass 90% (560 Tracks) in einem Trainingsdaten- und jeweils 5% (32 Tracks) in einem Verifikations- bzw. Testdatendatatz verwendet wurden. [www](http://www)

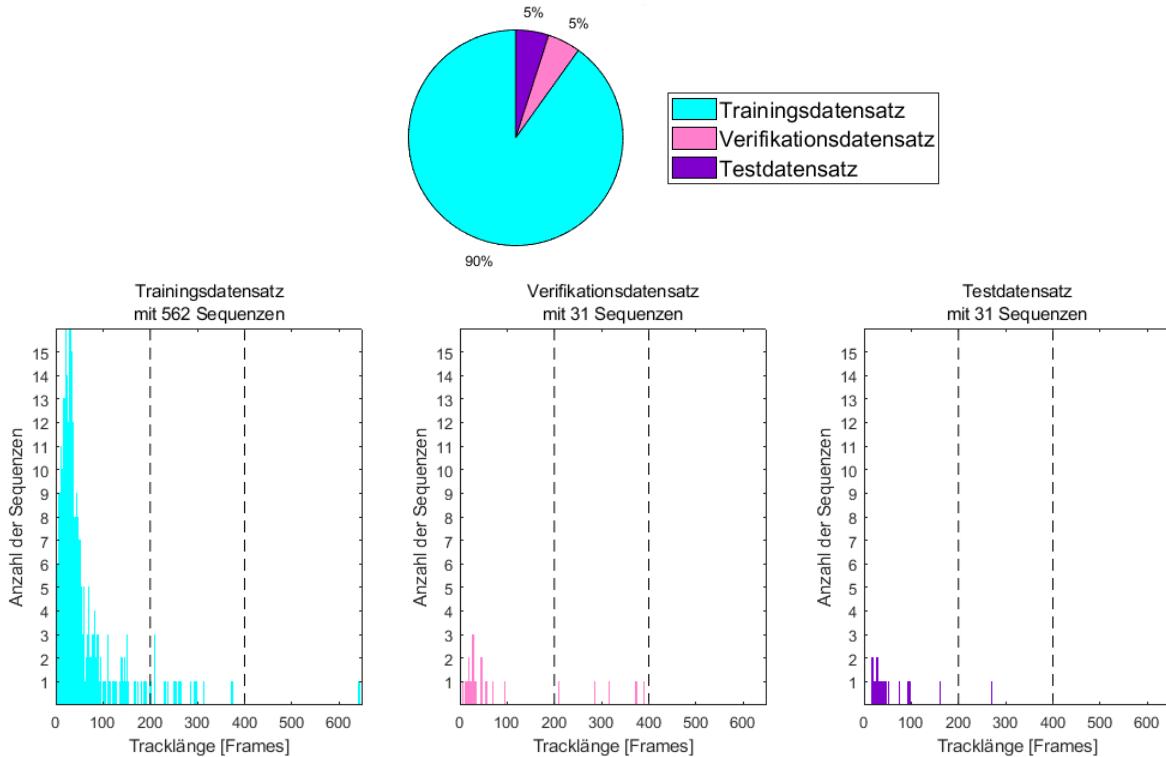


Abbildung 3.4: KITTI, grafische Darstellung der Datensatzaufteilung

Bei der Aufteilung der Daten wurde auf folgende Kriterien geachtet:

- **Trainingsdatensatz** - Enthält einen Großteil der Daten. Inklusive einer / mehrerer Repräsentationen der am geringsten vertretenen Beispiele. In diesem Fall sind die Tracks mit einer Tracklänge bis 200 Frames überrepräsentiert, Tracks mit 200-400 Frames unterrepräsentiert und lediglich ein Track mit einer Länge von über 400 Frames vorhanden (643 Frames). Entsprechend wurde darauf geachtet, dass diese im Trainingsdatensatz auftauchen.
- **Verifikationsdatensatz** - Die Aufteilung wurde trotz der geringen Anzahl von 31 Samples repräsentativ gewählt. Aus den Bereichen bis 400 Frames sind Tracks vertreten, welche für die Validation während des Netzwerktrainings verwendet werden.
- **Testdatensatz** - Auf Grund der geringen Datenmenge wurden keine Kriterien für den Testdatensatz gestellt, welcher keinen Einfluss auf die Netzwerkperformance haben wird. Damit wurde in Kauf genommen, dass die Daten nach abgeschlossenem Netzwerktraining nur noch bedingt eine Aussagekraft über die Gesamtperformance liefern können.

Um eine bessere Generalisierung zu erreichen und um die Chance zu erhöhen, dass das Training konvergiert, sollten die Prädiktoren (Trackzustandswerte zum Zeitpunkt t) und Ziele (Trackzustandswerte zum Zeitpunkt t+1) normalisiert werden. Nach Alexander Amini [2] kann eine Normalisierung durchgeführt werden, sodass die Prädiktoren und Ziele einen Mittelwert von Null und eine Einheitsvarianz besitzen. Die Berechnung des Mittelwerts  $\mu$  und der Standardabweichung  $\sigma$  wurde mit Hilfe der folgenden Formeln über alle Tracks durchgeführt:

$$\mu = \frac{1}{N} \sum_{i=1}^N A_i \quad (3.1)$$

$$S = \sqrt{\frac{1}{N-1} \sum_{i=1}^N |A_i - \mu|^2} \quad (3.2)$$

Im Entwicklungsverlauf wurde der Datensatz erweitert und entsprechendes Rauschen hinzugefügt. Das allgemeine Ziel solcher Maßnahmen ist es die Netzwerkperformance und -robustheit zu steigern. Details zum Vorgehen und Ergebnisse sind im Kapitel 3.3.3 beschrieben. Abschließen wurden die Tracks nach ihrer Tracklänge sortiert. Für das Netzwerktraining von Zeitreiheninformationen besitzt die Reihenfolge der Trainings-samples einen Einfluss. Details und eine Darstellung der unsortierten bzw. sortierten Daten folgen im kommenden Abschnitt (siehe Abbildung 3.6).

### 3.3.2 Netzwerkentwicklung

In diesem Abschnitt ist dokumentiert, wie der Funktionsblock der Prädiktion des in Abb. 3.1 dargestellten MOT Frameworks durch ein entwickeltes Prädiktionsnetzwerk ersetzt werden kann. Dazu wird die in Mathworks.inc [21] beschriebene Open-Loop Methode und eine LSTM Layer genutzt. Das bedeutet, dass das Netzwerk Werte für einen zukünftigen Zeitschritt anhand bisher erhaltener Daten vorhersagt. Für den Einsatz in einem online MOT Verfahren ist das Netzwerk somit in der Lage mit erhaltenen Messdaten bzw. Zustandswerten eines SO eine Vorhersage über die wahrscheinlich nächsten Zustandswerte zu treffen. Dabei werden die internen Werte (Hiddenstates) der LSTM Schicht pro Zeitschritt anhand der erhaltenen Messdaten aktualisiert. Durch diese Aktualisierung erfolgt somit eine interne Korrektur über den Sequenzverlauf.

**Konzeptentwicklung für die Prädiktion der Zustandswerte** In der ersten Konzeptphase wurde ein Netzwerk entwickelt für die vorhersage der Position eines SO relativ zum Egofahrzeug. Es wird somit ein Punkt Objekt (PO) angenommen, welches sich im Raum bewegt. Jeder Zustandswert ( $x, y$ , Gierwinkel) bildet dabei ein eigens Inputsignal bzw. Features für das Netzwerk. Wie in der Analysedarstellung 3.5 aufgelistet besteht SPENTpo aus vier Schichten (siehe LayerGraph Architektur, links). Die Sequenzinput-Schicht verarbeitet drei Features zu einem Zeitschritt und gibt diese an eine Long Short-Term Memory (LSTM)-Schicht weiter, welche die Zustandsabhängigkeiten zwischen den Zeitschritten mit Hilfe der Hiddenstates berücksichtigt. Die Schicht führt additive Interaktionen durch, die dazu beitragen können, den Gradientenfluss über lange Sequenzen während des Trainings zu verbessern. Für ein erstes Netzwerktraining werden 128 Hiddenunits gewählt (Speichereinheiten der Hiddenstates).

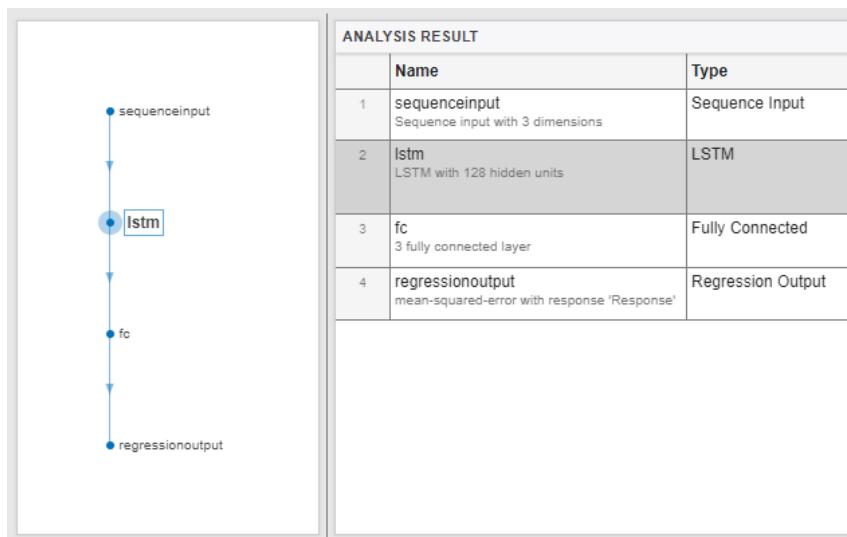


Abbildung 3.5: SPENTpo Analysedarstellung der Netzwerkarchitektur

Die Anzahl der Hiddenunits bestimmt, wie viele Informationen von der Schicht gelernt werden können (Hiddenstates und Cellstates). Die Verwendung von mehr versteckten Einheiten kann genauere Ergebnisse liefern, kann aber auch zu einem Overfitting führen. Die Bewertung und Optimierung der Parameter bzw. der Netzwerkarchitektur findet in einem weiteren Schritt statt. Ziel ist es Sequenzen mit gleicher Anzahl von Signalkanälen wie die Inputdaten auszugeben. Dazu muss eine vollständig verknüpfte Schicht (fully connected layer) mit einer Ausgangsgröße eingefügt werden, die der Anzahl der Features entspricht. Eine vollständig verknüpfte Schicht multipliziert die Eingabe mit einer Gewichtsmatrix und fügt dann einen Bias-Vektor hinzu. Abschließend wird noch eine Regressionsschicht eingefügt. Insgesamt besitzt das einfache bzw. flache (shallow) Netzwerk 67971 lernfähige Parameter, welche während der Backpropagation im Trainingsprozess anhand der Ergebnisse der Lossfunktion angepasst werden.

**SPENT Kostenfunktion (lossfunction)** Die verwendetet Regressionsschicht (regressionoutput) berechnet den Verlust des mittleren quadratischen Fehlers für Regressionsaufgaben. Der mittlere quadratische Fehler (Mean Squared Error (**MSE**)) gibt den Durchschnitt der quadrierten Differenz zwischen Modellvorhersage und Zielwert an. Dieser Wert kann als Maß für die Qualität eines Schätzers verwendet werden. Für eine einzelne Beobachtung ist der mittlere quadratische Fehler gegeben durch:

$$MSE = \sum_{i=1}^R \frac{(t_i - y_i)^2}{R} \quad (3.3)$$

wobei  $R$  die Anzahl der Antworten,  $t_i$  die Zielausgabe und  $y_i$  die Vorhersage des Netzes für Sample  $i$  ist. Bei Sequenz-zu-Sequenz-Regressionsnetzen wie **SPENT** ist die Verlustfunktion der Regressionsschicht der halbe mittlere quadratische Fehler der Vorhersagen für jeden Zeitschritt, normiert durch  $S$ , nicht durch  $R$ :

$$loss = \frac{1}{2S} \sum_{i=1}^S \sum_{j=1}^R (t_{ij} - y_{ij})^2 \quad (3.4)$$

wobei  $S$  die Sequenzlänge ist. Beim Training wird der mittlere Verlust über die Beobachtungen im Mini-Batch berechnet. Womit  $S$  die Mini-Batch-Länge darstellt.

**SPENT Trainingsoptionen** Die von Mathworks implementierten Methoden zur Erstellung des beschriebenen Netzwerks haben eine Vielzahl von Parametern und Funktionen als Default hinterlegt. Diese vereinfachen bzw. ermöglichen das erfolgreiche Training von Deep Neural Networks (DNNs). Eine vollständige Liste der betrachteten Trainingsoptionen ist im Anhang zu finden. Ein Auszug einer untersuchten Option ist in diesem Abschnitt beschrieben. Das Single Prediction Network (**SPENT**) soll eine Sequenz-zu-Sequenz Aufgabe lösen, welche keine definierte Sequenzlänge (Tracklänge) vorschreibt. Daher ist das Netzwerktraining mit dem folgenden **LSTM** spezifischen Sequenz-Optionen parametriert worden.

**SequenceLength = 'longest'** - Option zum Auffüllen, Abschneiden oder Teilen von Eingabesequenzen für das Netzwerktraining. Mit Auswahl der Option 'longest' werden die Sequenzen in jedem Mini-Batch so aufgefüllt, dass sie die gleiche Länge der längsten Sequenz besitzen. Bei dieser Option werden im Vergleich zum Kürzen keine Daten verworfen. Wie in Mathworks.inc [20] beschrieben, kann das Auffüllen zu Rauschen im Netzwerk führen, jedoch ist eine Sequenzanpassung pro Mini-Batch für das Training von **LSTM** Netzwerken erforderlich. Um den Effekt möglichst gering zu halten wurden die Sequenzen der Trainingsdaten daher vor dem Mini-Batch-Padding (Auffüllen) der Länge nach sortiert. In Abbildung 3.6 ist deutlich zu erkennen, dass das Padding (türkisfarbener Bereich) durch einen sortierten Trainingsdatensatz pro Mini-Batch minimiert wird. Wie stark die Netzwerkperformance durch die Anpassung der Mini-Batch Größe beeinflusst wird, ist im weiteren Verlauf dieses Kapitels dokumentiert.

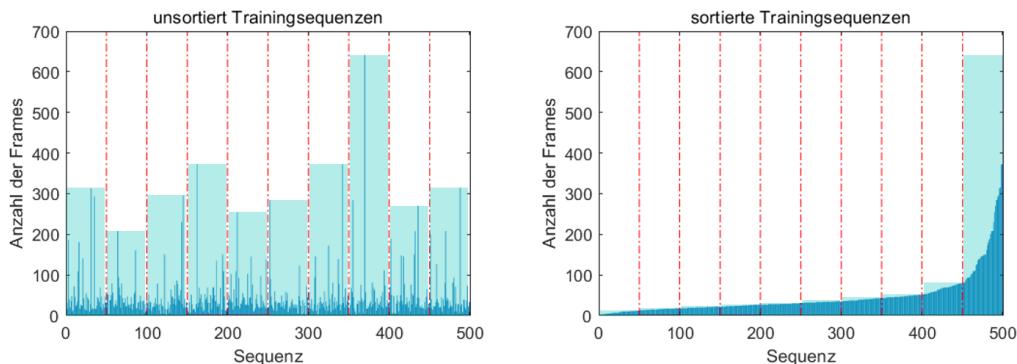


Abbildung 3.6: **LSTM** Netzwerktraining, Mini-Batch-Padding Effekt

### **SPENT Ergebnisanalyse: Prädiktion der Zustandswerte von Punkt Objekt (PO)**

Auf Basis des Testdatensatzes (Abbildung 3.4) konnten weitere Detailanalysen durchgeführt und die Netzwerkperformance überprüft werden. In Tabelle 3.1 ist der Root Mean Squared Error (RMSE) für die aufgeteilten Datensätze zusammengefasst. SPENTpo ist in dieser Tabelle mit den gewählten Eigenschaften in der Spalte 'Netzwerkvariante' aufgelistet: 128 Hiddenunits (HU), einer LSTM Schicht und einer Fully-Connected (FC) Schicht. Der berechnete RMSE stellt den Mittelwert über alle Sequenzen für die drei Zustandswerte der Position (x/y/yaw) dar.

Tabelle 3.1: SPENTpo, Ergebnistabelle

Netzwerkvariante	Datensatz: extrahierte GT			
	Training (x/y/yaw), 562 Tracks	Validation (x/y/yaw), 31 Tracks	Test (x/y/yaw), 31 Tracks	RMSE
HU:128, LSTM:1, FC:1	0.14	0.27	0.31	
-	-	-	-	

**Detailanalyse: höchster und geringster RMSE Track** Aus dem gesamten Datensatz wurde der Track mit der höchsten Abweichung ermittelt und die Prädiktionsergebnisse (rot) pro Zeitschritt in Abbildung 3.7 auf der linken Seite dargestellt. In blau sind die tatsächlichen Zustandswerte pro Zeitschritt aufgetragen. Der untersuchte Track besitzt die ID 3 aus der Sequenz 7 des KITTI Datensatzes. In dieser Sequenz fährt das Egofahrzeug durch eine Wohnsiedlung. Das Fahrzeug mit der ID 3 steht am Seitenrand auf der rechten Seite und wird vom Egofahrzeug während eines Abbiegevorgangs entdeckt. Die Abweichung der SPENT Prädiktion in den ersten Zeitschritten könnte auf Grund eines unausgeglichenen Datensatzes bestehen. Entsprechende Entdeckungen während einer Kurvenfahrt sind unterrepräsentiert. Die Abweichung wurde nach 6 Zeitschritten mit dem Erhalt der tatsächlichen Position (Messung) von SPENT korrigiert. Im weiteren Verlauf der Netzwerkentwicklung wird der Track als Referenzobjekt verwendet.

Der auf der rechten Seite dargestellte Track zeigt ein sehr gutes SPENT Prädiktionsverhalten. Die Einzelanalyse der Tracks verdeutlicht eine erfolgreiche Implementierung des datenbasierten Lernprozesses. Die entwickelte Netzwerkarchitektur ist für eine open-loop Prädiktion der Zustandswerte geeignet. Die RMSE Werte der längeren Sequenzen sind auf Grund der höheren Anzahl an Elementen in der Regel geringer. Im nachfolgenden Abschnitt werden mögliche Optimierungsmaßnahmen für das SPENT Netzwerk untersucht.

### 3 Ergebnisse

---

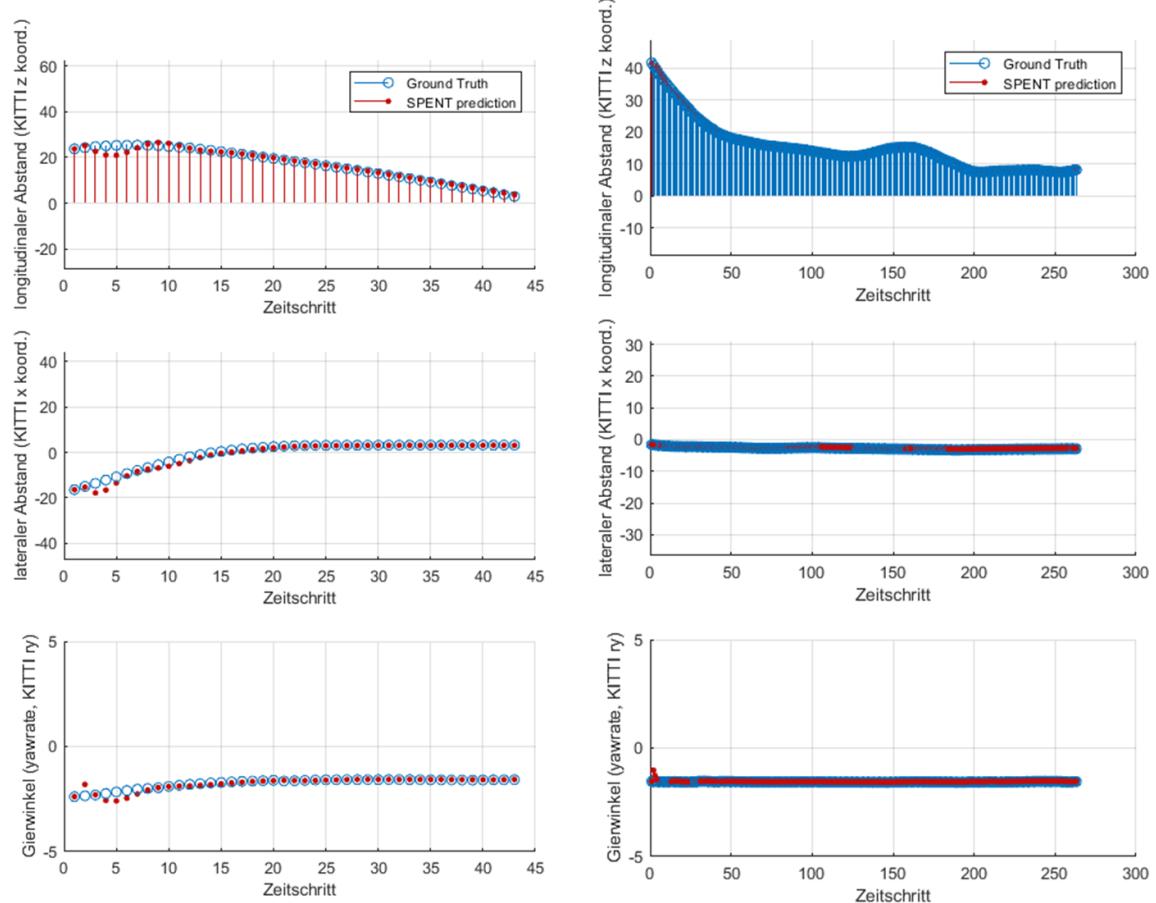


Abbildung 3.7: SPENTpo, grafische Bewertung der Testtracks

höchster RMSE = 0.396, Sequenz: 7 / ID: 3 (li.),

geringster RMSE = 0.033, Sequenz: 18 / ID: 2 (re.)

### 3.3.3 Optimierungen

Für die Single Prediction Network (**SPENT**) Optimierung wurde drei Entwicklungs-umfänge definiert. Das strukturierte Vorgehen dient der Übersicht mit dem Ziel die Performance zu steigern. Die Optimierungsmaßnahmen im Überblick:

1. **Erweiterung Netzwerkinputdaten** - von 3 Zustandswerten (X,Y Koordinate und Gierwinkel) hin zu 7 Zustandswerten eines jeden Objekts (X,Y Koordinate, Gierwinkel, Länge, Breite und relative Geschwindigkeit des Objekts in X und Y Richtung)
2. **Netzwerkvarianten** - empirische Ermittlung der Netzwerkarchitektur (Netzwerktiefe, -breite und Layeraufbau) sowie Optimierung der Netzwerk- und Trainingsparameter
3. **Datenerweiterung (data augmentation)** - Untersuchung der Rauschintensität und Möglichkeit der Erweiterung der extrahierten Tracks für das Netzwerktraining

#### Optimierungsmaßnahme 1 - Erweiterung der Netzwerkinputdaten

In einer ersten Optimierungsphase wurde das Single Prediction Network (**SPENT**) erweitert, sodass eine hohe Anzahl an Zustandswerten als Sequenzinputdaten verarbeitet werden. Der Erweiterungsumfang orientiert sich dabei an Daimler Truck (**DT**) serientypischen Zustandsvektoren eines gängigen Sensors zur Erfassung des Truckumfelds. Aktuelle Sensoren liefern unter anderem folgende relevante Daten auf dem Sensorobjektinterface, welche auch im KITTI Datensatz extrahiert werden, um ein Netzwerktraining und -evaluierung implementieren zu können:

- x Koordinate / posX (KITTI Kamerakoordinatensystem: z)
- y Koordinate / posY (KITTI Kamerakoordinatensystem: -x)
- Gierwinkel / Yaw (KITTI Kamerakoordinatensystem: ry)
- Objektlänge in x / dimX (KITTI z Richtung)
- Objektbreite in y / dimY (KITTI x Richtung)
- relative Geschwindigkeit in x / vrelX (KITTI z Richtung)
- relative Geschwindigkeit in y / vrelY (KITTI -x Richtung)

### 3 Ergebnisse

---

Das Netzwerk erhält somit eine höhere Informationsmenge pro Zeitschritt über das einzelne **SO**. Die Geschwindigkeiten stellen dabei einen berechneten Wert in Abhängigkeit der erfassten Positionen dar, werden jedoch auch in klassischen Tracking Verfahren gerne zur Stabilisierung verwendet. Es ist zu erwarten, dass das Netzwerk die Abhängigkeiten (Gierwinkel, Geschwindigkeiten und Position im Raum) im Training lernt und somit eine genauere Schätzung der Position ermittelt werden kann.

Tabelle 3.2 umfasst die Ergebnisse der beiden bisherigen Netzwerkentwicklung **SPENT<sub>po</sub>** und **SPENT<sub>dt</sub>** im Vergleich. Damit ein Vergleich zur ersten Variante möglich ist, wird der Root Mean Squared Error (**RMSE**) ebenfalls über die prädizierten Zustandswerte x, y, yaw (KITTI Koordinatensystem) gebildet. Die Ergebniswerte stellen einen Mittelwert über 10 Netzwerktrainingsdurchläufe dar. Die Werte zeigen, dass die Performance durch die Erhöhung der Anzahl der Zustandswerte (Features) verbessert werden konnte. Es konnte gezeigt werden, dass das Netzwerk die höhere Datenmenge bzw. Informationsdichte verknüpfen und somit genauere Vorhersagen treffen kann.

Tabelle 3.2: **SPENT**, Ergebnistabelle, Erweiterung der Zustandswerte

Netzwerkvariante	Datensatz: extrahierte GT		
	Training, 562 Tracks	Validation, 31 Tracks	Test, 31 Tracks
RMSE (x/y/yaw)			
<b>SPENT<sub>po</sub>:</b> HU:128, LSTM:1, FC:1	0.14	0.27	0.31
<b>SPENT<sub>dt</sub>:</b> HU:128, LSTM:1, FC:1	0.11	0.19	0.20

Im folgenden Abschnitt sind die Ergebnisse der Einzel-Track-Untersuchung zusammengefasst. Die Analyse zeigt die Änderung, durch die Zustandsvektorerweiterung.

**Detailanalyse: Track mit höchstem RMSE** Die Performance (gesamter Datensatz) bezogen auf die Positionsschätzung konnte verbessert werden (**RMSE**: von 0,141 zu 0,116). Der Track mit der höchsten Abweichung ist nun ein sehr kurzer (4 Zeitschritte) mit einem **RMSE** von 0,386. Ein Wert der für Tracks mit wenigen Zeitschritten in einem sehr guten Bereich liegt. Die Prädiktionen des Tracks mit der ID 3 aus der Sequenz 7 weist mit **SPENT<sub>po</sub>** den höchsten **RMSE** auf. In Abbildung 3.8 sind die Ergebnisse von **SPENT<sub>po</sub>** (li.) und **SPENT<sub>dt</sub>** (re.) zu diesem Track nebeneinander abgebildet. Es ist zu sehen, dass das zuvor beschriebene Verhalten in den ersten Zeitschritten verbessert wurde.

### 3 Ergebnisse

Das Ergebnis bestätigt ebenfalls die aufgestellte Theorie, dass das Netzwerk durch die Featureerweiterung Abhängigkeiten unter den Daten lernt und somit eine präzisere Prädiktion bereits in den ersten Zeitschritten ausgegeben werden kann. ([RMSE ID 3, Seq. 7: von 0,396 zu 0,188](#)).

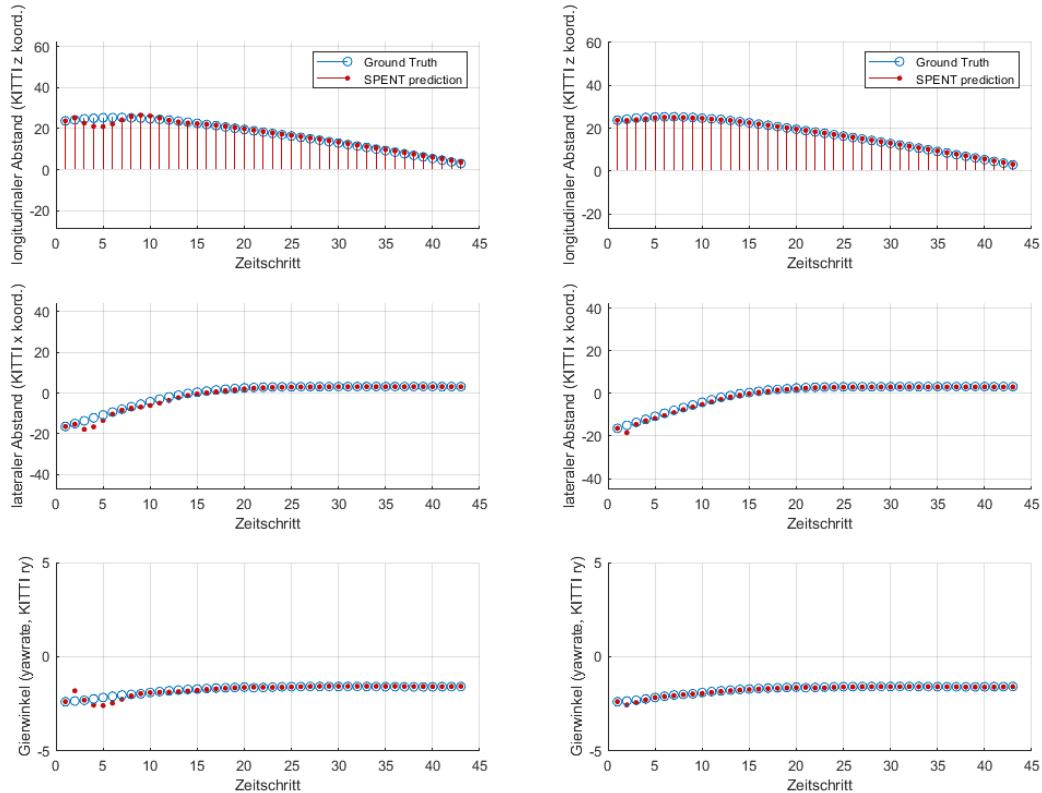


Abbildung 3.8: Testtrackbewertung, Sequenz: 7 / ID: 3 (li.), **SPENT**po (li.), **SPENT**dt (re.)

#### Optimierungsmaßnahme 2 - Netzwerkvarianten

Nachdem die Performance mit dem erweiterten Daimler Truck (**DT**) Zustandsvektor des Single Prediction Network (**SPENT**) verbessert werden konnte, dient das entwickelte Netzwerk (**SPENTdt**, HU:128, LSTM:1, FC:1) und der in Kapitel [3.3.1](#) beschriebene Datensatz als Basisreferenz für die Netzwerkvariantenbildung. Der Entwicklungsprozess wurde für die experimentelle Netzwerkentwicklung in folgenden Teilbereiche gegliedert:

- **Variation der rekursiven Schichten** - die meist verwendeten gated Recurrent Neural Network (**RNN**) Schichten gängiger Sequenzmodelle wurden implementiert und Ihre Performance bewertet. Daher wurden **GRU**, **LSTMProjectedLayer**, **LSTM** und **BILSTM** Schichten untersucht.
- **Netzwerkbreite und -tiefe** - die Breite des Netzwerks kann über die Anzahl der Neuronen bzw. Hiddenunits der rekursiven Netzwerkschichten bestimmt werden. Wie in Kapitel [2.2.1](#) beschrieben, wird die Netzwerktiefe über die Anzahl der Schichten definiert. Entsprechend wurden Netzwerke mit unterschiedlichen Netzwerkbreiten und -tiefen entwickelt, trainiert und dessen Performance bewertet.

Es wurden eine Vielzahl an Experimenten durchgeführt. Für die Dokumentation in dieser Arbeit hat bereits eine Selektion zu Gunsten der Übersicht stattgefunden. Alle Optimierungen wurden mit Hilfe der Bayes'sche Optimierung berechnet und hatten das Ziel den Verlust auf Basis des Validationsdatensatzes (Validation Loss) zu minimieren.

Im Ergebnisplot [3.9](#) ist der Trainingsfortschritt anhand der aufgezeichneten Metriken des **SPENTdt** Netzwerks dargestellt. Im Allgemeinen werden Plots zur Überwachung des Trainingsprozesses dazu eingesetzt, um auf visueller Ebene eine Bewertung über den Erfolg des Trainings abzugeben. So lässt sich beispielsweise feststellen, ob und wie schnell sich die Genauigkeit des Netzwerks verbessert und ob das Netzwerk beginnt, die Trainingsdaten zu stark anzupassen (overfitting). Jede Iteration stellt eine Schätzung der Gradienten dar und eine entsprechende Aktualisierung der anpassbaren Netzparameter. Das obere Diagramm zeigt den geglätteten (smoothed) Root Mean Squared Error (**RMSE**) über die Iterationen des gesamten Optimierungsprozesses, bezogen auf die Trainingsdaten. In diesem Training wurde ein Minimum des Validation Loss angestrebt. Der Verlauf der Loss Werte ist im unteren Diagramm gezeichnet (dunkles Orange: Trainingsdaten). In beiden Ergebnisplots ist jede Trainingsepoch durch einen schattierten Hintergrund gekennzeichnet. Eine Epoche ist ein vollständiger Durchlauf mit allen Beispielen des Trainingsdatensatzes. Ebenfalls dargestellt sind die Zeitpunkte

### 3 Ergebnisse

---

der Bewertung der Netzwerkperformance, welche mit dem Setupparameter 'ValidationFrequency' festgelegt wurden (schwarze Punkte mit gestrichelter Verbindungslien). Die Bewertung wird mit dem Validationsdatensatz durchgeföhrt.

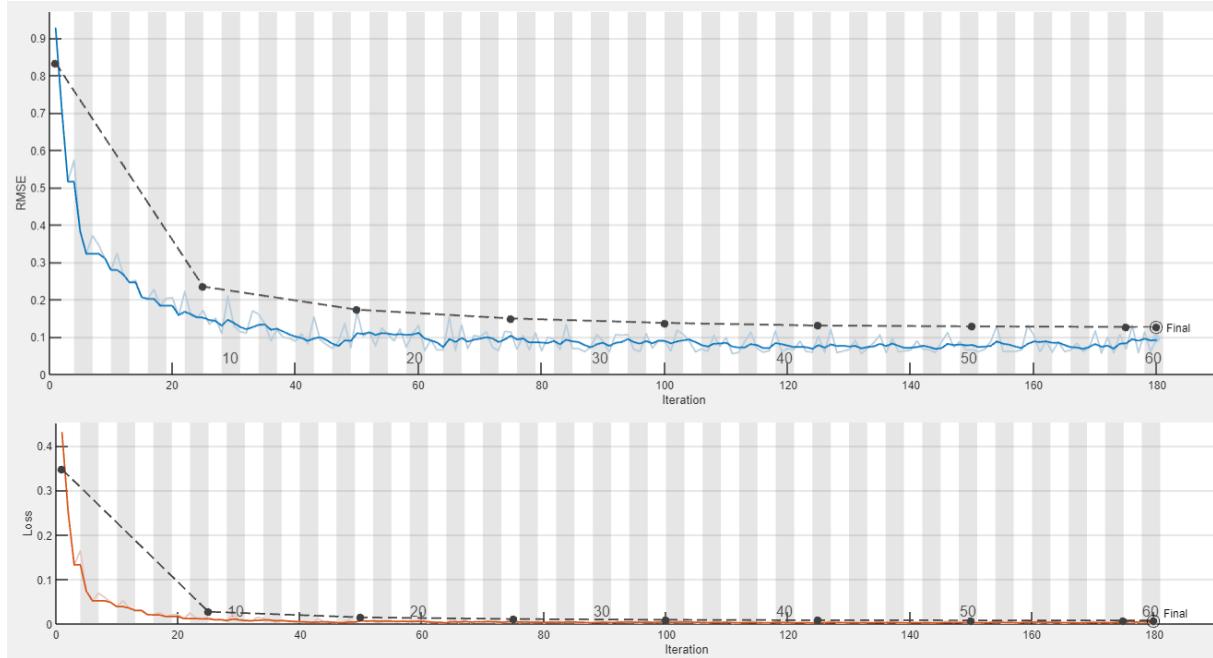


Abbildung 3.9: SPENTdt, Ergebnisplot, Trainingsprozess

Der Ergebnisplot kann als gut angepasste Lernkurve interpretiert werden (good fit). Eine gute Anpassung wird durch einen Trainings- und Validierungsverlust gekennzeichnet, der mit einem minimalen Abstand zwischen den beiden endgültigen Verlustwerten abnimmt. Der Verlust des Modells wird im Trainingsdatensatz fast immer niedriger sein als im Validierungsdatensatz. Die Generalisierungslücke (Abweichung zwischen Trainingsdatensatz und Validierungsdatensatz) ist ebenfalls gering, womit die Datensatzaufteilung bewertet werden kann.

Es wurden Netzwerke mit zusätzlichen Aktivierungsfunktions- und Normalisierungsschichten implementiert und trainiert. Die Auszüge der Ergebnistabelle 3.10 zeigen, die Ergebnisse einiger Varianten im Vergleich zum Referenznetzwerk (dunkelgrau hinterlegt). Keines der Netzwerke mit zusätzlichen Aktivierungs- und Normalisierungsschichten konnte mit den gewählten Trainingsoptionen bessere Ergebnisse erzielen. Der Verlust der Netzwerkarchitektur mit einer rekursiven Schicht und zwei Fully Connected (**FC**) Schichten konnte in diesen Experimenten am besten minimiert werden (gelb hinterlegt). Die Anzahl der Neuronen des rekursiven Layer wurde im ersten **FC** Layer halbiert und mit einem **FC** verbunden, mit einer Neuronenanzahl entsprechend der Ausgabedaten (Vorhersagewerte für 7 Zustandswerte).

### 3 Ergebnisse

Trial	NETWORKARCHITECTURE	NUMHIDUNITS	Training RMSE	Validation RMSE	Validation Loss
3	bilstm1FC2	192	0.0835	0.1050	0.0055
6	bilstm1FC2	160	0.0607	0.1304	0.0085
16	bilstm1FC1	160	0.1274	0.1678	0.0141
1	bilstm1FC1	128	0.0641	0.1738	0.0151
...					
12	LSTM1FC2	192	0.1040	0.2225	0.0247
4	LSTM1FC2	128	0.1330	0.2243	0.0252
3	LSTM1FC1	192	0.1374	0.2262	0.0256
8	LSTM1FC1	128	0.1142	0.2315	0.0268
...					
1	LSTMprojected1FC2	128	0.4411	0.5234	0.1370
2	LSTMprojected1FC2	192	0.4860	0.5637	0.1589
10	LSTMprojected1FC1	128	0.5730	0.5736	0.1645
20	LSTMprojected1FC1	96	0.6636	0.6672	0.2226
...					

Abbildung 3.10: SPENT, Ergebnistabelle

Die Werte der Ergebnistabelle 3.10 zeigen, dass die BILSTM Netzwerke zu einer besseren Repräsentation der Zeitreihe führen. Das kann mit ihrer Fähigkeit begründet werden, Informationen sowohl von vergangenen als auch zukünftigen Zeitpunkten berücksichtigen zu können. Die bidirektionale Verarbeitung hilft dabei, den Kontext besser zu lernen und Abhängigkeiten in beide Richtungen zu modellieren. Dies erhöht jedoch auch die Berechnungskomplexität und Modellgröße.

Gated Recurrent Unit (GRU) und LSTMprojectedLayers besitzen eine reduzierte Anzahl an trainierbaren Parametern und somit ebenfalls eine geringere Modellkapazität. Sie eignen sich somit für Anwendungen welche eine effiziente Implementierung erfordern. Die Beschränkungen können jedoch zu begrenzten Fähigkeiten der erlernten (langfristigen) Abhängigkeiten führen. Die Ergebnisse zeigen, dass es im betrachteten Fall zu Einschränkung bei der Vorhersage der Zustandswerte kommt.

Trotz der hohen Modellkapazität werden im Rahmen dieser Arbeit keine Einschränkungen zu Ungunsten der Performance gewählt und daher das optimierte BILSTM und LSTM Netzwerk mit der geringsten Fehlerabweichung für die weiteren Untersuchungen gewählt. Die bisher flache Netzwerkstruktur wird in einem weiteren Experiment um die aufgelisteten Schichten erweitert:

- **dropoutLayer** - Eine Dropout-Schicht setzt Eingabeelemente mit einer bestimmten Wahrscheinlichkeit zufällig auf Null (Wahrscheinlichkeit 25-75% wurden getestet, Ergebnistabelle mit Wahrscheinlichkeitswert = 25%). Durch diesen Vorgang wird nach Hinton et al. [10] die zugrunde liegende Netzarchitektur zwischen den Iterationen effektiv verändert und eine Überanpassung (overfitting) des Netzes

verhindert. Ein höherer Wahrscheinlichkeitswert führt dazu, dass mehr Elemente während des Trainings verworfen werden. Zum Zeitpunkt der Vorhersage ist die Ausgabe der Schicht gleich der Eingabe.

- **layerNormalizationLayer** - Für jedes Merkmal (d. h. jede Dimension) in der Aktivierung wird ein eigener Normalisierungsfaktor berechnet. Dies ermöglicht eine unabhängige Normalisierung der Aktivierungen für jedes Merkmal und verbessert die Robustheit des Netzwerks gegenüber unterschiedlichen Skalen und Verteilungen. Die Schicht wird nach Ba et al. [4] typischerweise nach der Aktivierungsfunktions- bzw. nach den lernfähigen Schichten verwendet.
- **reluLayer** - Eine Rectified Linear Unit (**ReLU**) Schicht führt für jedes Element der Eingabe eine Schwellenwert-Operation durch, bei der jeder Wert kleiner als Null auf Null gesetzt wird. Auf Normalisierungsschichten folgt in der Regel eine nichtlineare Aktivierungsfunktion wie **ReLU** (gleichgerichtete lineare Einheit), die durch eine entsprechende Schicht spezifiziert wird.

Durch den Einsatz der zusätzlichen Layer konnte mit dem aktuellen Setup keine weiteren Optimierungen erzielt werden (Validation loss in einem ähnlichen Bereich). In dieser Untersuchung wurde die Performance des Netzwerks als einzelne Softwarekomponente betrachtet. Die Netzwerkarchitektur weist für die Integration in einen Gesamtframework somit noch Optimierungspotenzial auf. Es ist daher an dieser Stelle zu empfehlen weitere Optimierungen in Gesamtkontext des Trackings durchzuführen.

#### Optimierungsmaßnahme 3 - Datenerweiterung

Das Themengebiet der Datenerweiterung (data augmentation) bildet ein weiteres Forschungsgebiet im Computer Vision (**CV**) Bereich. Die Datenerweiterung ist im allgemeinen eine Technik zur künstlichen Vergrößerung des Trainingssatzes durch die Erstellung modifizierter Kopien eines Datensatzes unter Verwendung vorhandener Daten. Nach Goodfellow et al. [9] werden dabei die erweiterten Daten aus den Originaldaten mit einigen geringfügigen Änderungen gewonnen. Bei der Objekterkennung auf Bildebene sind gängige Methoden der Datenerweiterung bspw. geometrische und Farbraumtransformationen welche vorgenommen werden, um die Größe und Vielfalt der Trainingsmenge zu erhöhen. Hingegen werden synthetische Daten (synthetic data) künstlich erzeugt, ohne weitere Basisdaten. Diese Untersuchung beinhaltet ausschließlich die Methode der Datenerweiterung. Ziel ist es daher auf Basis von erweiterten Trainingsdaten die Vorhersagegenauigkeit für einen unveränderten Groundtruth (**GT**) Validationsdatensatz zu erhöhen.

Die Erweiterung der **GT** Trainingsdaten für das **SPENT** Netzwerk wird in zwei kombinierten Optimierungsstufen durchgeführt:

- **Flipped Tracks** - jeder **GT** Track wird in umgekehrter Reihenfolge als Erweiterung des Trainingsdatensatzes hinzugefügt.
- **Rauschintensität** - jeder **GT** Track wird mit unterschiedlicher Rauschintensität modifiziert und ebenfalls dem Trainingsdatensatz hinzugefügt.

Die Flipped Tracks Erweiterung ist über eine switch-case Implementierung realisiert. Die Rauschintensität (0 bis 5%, 1% Schritte) wurde ebenfalls über eine entsprechende Abfrage implementiert. Für jeden case (bsp. Rauschintensität = 5%) wird eine implementierte Funktion aufgerufen, welche den aktuellen Wert eines jeden Zustands um einen pseudo-zufälligen Wert aus dem gewählten Rauschbereich (bsp. +/-5%) modifiziert (genutzte Matlab Funktion: `normrnd(0,sigma)`). Die Implementierung wurde auf Basis der Annahme realisiert, dass mit steigenden Werten die Genauigkeit sinkt. Dies ist beispielsweise in vielen Kamerasystem oft der Fall.

Mit diesem Vorgehen sollte sichergestellt werden, dass das Netzwerk Daten erhält, welche nicht konstant über einen prozentualen Offset verfügen. Dieses einfache Muster würde sonst sehr Wahrscheinlich vom Netzwerk erlernt werden. Es ist anzunehmen, dass somit die Generalisierungsfähigkeit nicht weiter verbessert werden würde. Die erweiterten (verauschten) Daten besitzen daher pro Zeitschritt einen Offset, welcher zu einer hohen Wahrscheinlichkeit vom vorherigen Zeitschritt abweicht und somit eine verausachte Sensorsignal darstellt.

### 3 Ergebnisse

---

Der erweiterte Trainingsdatensatz besitzt je nach Kombination der beiden sequenziell ausgeführten Modifikationen eine unterschiedliche Anzahl an Tracks. Die Anzahl der Tracks im Validierungsdatensatz bleibt gleich, da hier ausschließlich unverrauschte GT Daten zur Bewertung der Performance zum Einsatz kommen. Durch die Erweiterung des beschriebene Trainingsdatensatzes konnte mit dem entwickelten BILSTM und LSTM Netzwerkaufbau eine ähnliche Performance erreicht werden. Tabelle 3.3 zeigt das Ergebnis aller Untersuchungen auf Basis der LSTM Netzwerkevolution.

Durch einen zusätzlichen geflippten und um 5% verrauschten Trainingsdatensatz konnten die Netzwerkperformance bezogen auf den unveränderten GT Datensatz noch einmal verbessert werden. Zusätzlich wurden die RMSE Ergebnisse eines implementierten Kalman Filters (KFs) aufgelistet (Skriptauszug im Anhang). Die KF Parameter wurden in dieser Arbeit nicht auf jeden Test angepasst. Der Vergleich zeigt, dass trotz der geringen Anzahl an Samples das trainierte Single Prediction Network (SPENT) eine vergleichbar geringe Fehlerquote erzielen kann. Der Vergleich zielt dabei darauf ab eine Einordnung der SPENT Konzeptentwicklung zu ermöglichen und stellt dar, dass mit dem entwickelten Netzwerk- / Daten- / Trainingskonzept ein Zustandsvorhersagemodell mit Potenzial entwickelt worden konnte. Es wird deutlich, dass mit steigender Datenverfügbarkeit ein Netzwerk wie SPENT, die Fähigkeiten der universellen Funktionsapproximation nach White et al. [35] erlernen kann. Ein Vergleich der beiden Vorhersagemodele im MOT Framework findet in Kapitel 3.6 statt.

Tabelle 3.3: SPENT, Ergebnistabelle Netzwerkevolution

Netzwerkvariante	Datensatz: extrahierte GT		
	Training, 562 Tracks	Validation, 31 Tracks	Test, 31 Tracks
RMSE (x/y/yaw)			
SPENT <sub>po</sub> : HU:128, LSTM:1, FC:1	0.14	0.27	0.31
SPENT <sub>dt</sub> : HU:128, LSTM:1, FC:1	0.11	0.19	0.20
SPENT <sub>dt</sub> : HU:128, LSTM:1, FC:2 (+ flipped Tracks, + noisy Tracks)	0.025	0.029	0.023
Kalman Filter (KF)	0.066	0.065	0.066

### 3.3.4 Zusammenfassung

Kapitel 3.3 stellt das Vorgehen und einige Ergebnisse der Experimente dar, welche im Rahmen der Entwicklung eines Single Prediction Network (**SPENT**) durchgeführt wurden. In Abb. 3.11 ist das entwickelte Netzwerk inklusive der Input- und Output Datenstruktur noch einmal übersichtlich dargestellt.

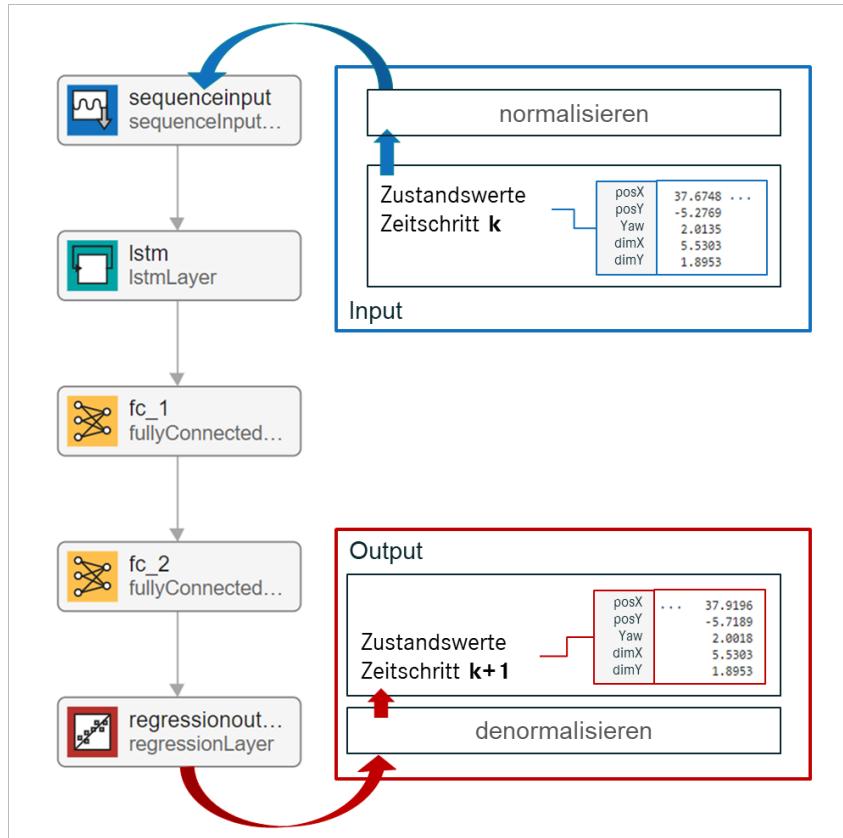


Abbildung 3.11: Single Prediction Network (**SPENT**), Input- Output Struktur

Zusammengefasst wurden folgende Arbeiten durchgeführt bzw. Ergebnisse realisiert:

- Die grundsätzliche Entwicklungsstrategie für ML Anwendungen konnte für das Aufgabengebiet erfolgreich genutzt werden.
- Die verfügbaren Daten wurden analysiert und entsprechend aufbereitet, um ein konvergierendes Netzwerktraining zu realisieren.
- Um eine bessere Generalisierung zu erreichen, wurde vorgeschlagen, die Prädiktoren und Ziele zu normalisieren, sodass sie einen Mittelwert von Null und eine Einheitsvarianz haben.

- Verschiedene Netzwerkarchitekturen wurden implementiert und trainiert, einschließlich solcher mit zusätzlichen Aktivierungsfunktions- und Normalisierungsschichten.
- Die besten Ergebnisse konnten mit Netzwerkarchitekturen erzielt werden, welche **LSTM** und **BILSTM** Schichten für die Verarbeitung der kontextbezogenen Informationen der Tracks (Zeitreihen) verwenden.
- Es konnte nachgewiesen werden, dass diese Modelle zur real time Zustandsvorhersage geeignet sind.
- Mit der Annahme der Verschlechterung der Genauigkeit für hohe Distanzen, wie es beispielsweise bei Kameras oft der Fall ist, wurden das Netzwerk erfolgreich auf einen bis zu 5% verrauschten Datensatz trainiert. (Abweichung prozentual auf den Absolutwert pro Zeitschritt)
- Das entwickelte Modell operiert auf **SO** Ebene und ermöglicht somit eine Integration in ein bestehendes **TbD MOT** Framework. Der Vergleich mit einem **KF** zeigt, dass **SPENT** bei einer entsprechenden Trainingsdatenbasis genauso gute oder bessere Ergebnisse erzielen kann.

**Ausblick** Die durchgeführten Optimierungsmaßnahmen stellen einen Auszug der Möglichkeiten in der **ML** Entwicklung dar. Es könnte vorteilhaft sein, weitere Experimente mit unterschiedlichen Netzwerkarchitekturen und Hyperparametern durchzuführen. Die Verwendung von fortschrittlicheren Techniken zur Datenanreicherung und -vorverarbeitung könnte ebenfalls untersucht werden. Eine tiefere Analyse der Fehlerfälle und die Entwicklung von Strategien zur Adressierung dieser Fehler könnten ebenfalls als potenzielle Weiterentwicklungsbereiche betrachtet werden.

## 3.4 Single Association Network (SANT) - Assoziation einer Messung zu erfassten Tracks

In diesem Kapitel ist der Entwicklungsprozess eines **ML** Modells zur Lösung des Datenassoziationsproblem beschrieben. Wie von Mertz et al. [23] wurde auch in dieser Arbeit das Ziel verfolgt einen datenbasierten Ansatz zu entwickeln, welcher lernen kann, das kombinatorische Non Deterministic Polynomial Time (**NP**) hard Optimierungsproblem der Datenassoziation vollständig zu lösen. Die bisherigen Ergebnisse dieser und der in Kapitel 2.4.3 untersuchten Arbeiten zeigen, dass sich ein **LSTM** bzw. **BILSTM** Netzwerk als **ML** Methode ebenfalls für die Aufgabe der Datenassoziation eignen kann. Diese Aussage kann auf Grund ihrer Fähigkeiten der nichtlinearen Transformationen und Speicherkomponente zur Berücksichtigung der zeitlichen Abhängigkeiten begründet werden. Darstellung 3.12 zeigt die Einordnung im **MOT** Framework.

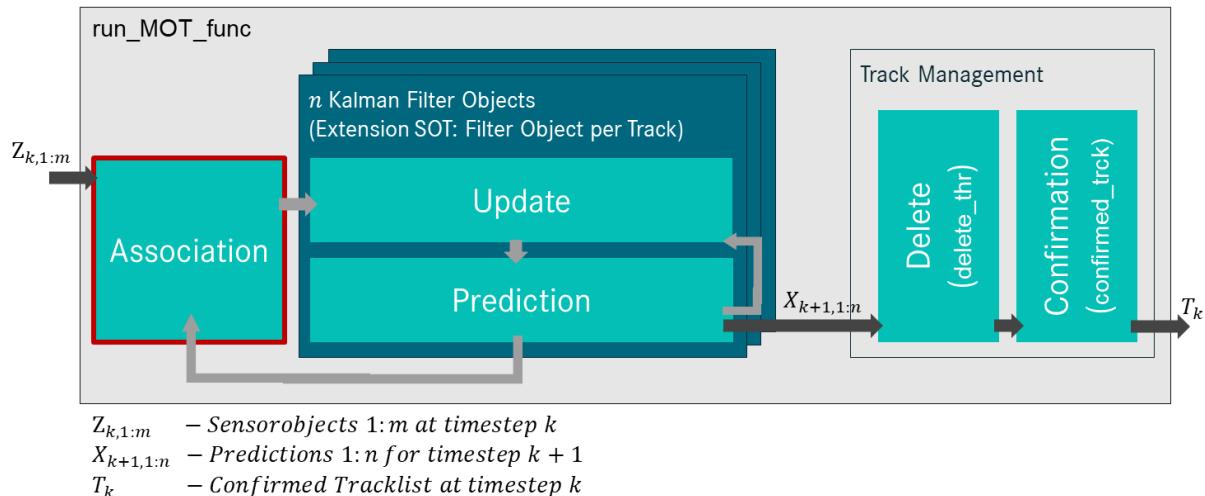


Abbildung 3.12: Schematische Darstellung, **SANT** Funktionsgebiet rot markiert

Wie bereits in Kapitel 2.4.3 beschrieben, nutzen Mertz et al. [23] als Inputdaten für das entwickelte **DA** Netzwerk eine Distanzmatrix auf Basis des euklidischen Abstandsmaßes, und ersetzt somit einen Assoziationsalgorithmus wie z.B. den Hungarian Algorithm (**HA**) (siehe Kapitel 2.1.2). Es ist anzunehmen, dass bei der Erstellung der Groundtruth (**GT**) Trainingsdaten (Distanzmatrizen), sowie bei der Evaluierung, das euklidische Abstandsmaß als Basisberechnung verwendet wurde. Dies ist jedoch nicht explizit aufgeführt. Es lässt sich somit die Behauptung aufstellen, dass durch diesen Berechnungsschritt dem Netzwerk die Möglichkeit genommen wird, einer anderen Assoziationslogik zu folgen bzw. diese datenbasiert zu lernen.

Im Rahmen dieser Arbeit wurde daher die These aufgestellt, dass durch ein nicht definiertes Abstandsmaß ein **GRU** basiertes Assoziationsnetzwerk die Zuordnung von der

zeitlichen Speicherkomponente und somit von der Historie verstrt gebildet werden kann. In diesem Kapitel der Arbeit wird daher das Ziel verfolgt ein Assoziationsnetzwerk zu entwickeln, welches die Zuordnung eines **SO** zu einer bestehenden Anzahl an Tracks ohne definiertes Abstandsma len soll.

Das Problem der Assoziation neuer Sensor Objekt (**SO**) und Tracks wird in diesem Kapitel unter folgender Betrachtung als Machine Learning (**ML**) Problem eingeordnet:

- **Klassifikation** - das Problem kann nicht als klassisches Klassifikationsproblem betrachtet werden, da keine definierten Klassen gelernt bzw. gelabelt werden knnen. Jedes Sensor Objekt (**SO**) ist zum Zeitpunkt des Erscheinens von der Umgebung und anderen Beobachtungen abhig.
- **Multi Zuordnung** - das erste implementierte Konzept ordnet immer genau eine neue Messung bzw. ein neues **SO** einem bestehenden Set von Tracks zu. Es besteht ein Zuordnungsproblem pro Zeitschritt, welches abhig von den aktuellen und bereits erfassten Messdaten (Tracks) ist.

Somit kann eine einfache Datenassoziation als zeitlich gegliedertes Problem (Sequenz-zu-Vektor) betrachtet und durch eine entsprechende Datenvorverarbeitung und ein entsprechendes Trainingsverfahren ein Netzwerk fr die Lsungsfindung trainiert werden. Der Workflow orientiert sich an gngigen **ML** Entwicklungsprozessen und der bereits vorgestellten Entwicklung des **SPENT** Modells (siehe Anhang). Ziel der Single Association Network (**SANT**) Entwicklung ist im ersten Schritt die folgenden Zuordnungsmglichkeiten zu len:

- **1 zu n** - ein **SO** zu  $n$  Tracks
- **0 zu n** - keine **SO** zu  $n$  Tracks

#### 3.4.1 Datenvorverarbeitung

Fr eine erste Netzwerkentwicklung wird das Datenassoziationsproblem als Zuordnung von einem **SO** zu einem Set von Tracks betrachtet. Die entsprechenden Tracks wurden aus dem KITTI Datensatz der gelabelten Kameraaufzeichnungen extrahiert. Allerdings existieren fr das betrachtete Problem der Assoziation keine realen **GT** Daten. Um sicherzustellen, dass die Zuordnung eines Sensor Objekts (**SOs**) zu einem Track eines Tracksets genau eine korrekte Lsung besitzt, wurde das **SO** in einem Zeitschritt aus dem bestehenden Trackset eines Zeitschritts knstlich erzeugt.

**Generierung der Groundtruth (GT) Daten** Die in Abbildung 3.13 dargestellte Struktur der Inputdaten wurde für das Training des Single Association Network (**SANT**) gewählt. Die Größe der Matrix entspricht  $m * n$ . Mit  $m = 2 * \text{AnzahlZustandswerte}$  und  $n = \text{Trackanzahl}(t)$ . Der obere markierte Wertebereich der Matrix ist für die Zustandsvektoren der im aktuellen Zeitschritt bestätigten Tracks reserviert (hier: 10 Tracks). Der Trackindex bleibt über die Zeit der Existenz bestehen. Der untere markierte Bereich zeigt ein erzeugtes Sensor Objekt (**SO**), welches einem bestehendem Track zugeordnet werden soll. Der Groundtruth (**GT**) Wert entspricht dem Index der existierenden Trackliste (hier: 10, Ausgabe im unteren Bildschnitt). Die Anzahl der Größe  $n$  ist abhängig von der Trackanzahl und variiert somit über die Zeit. Die gewählte Datenstruktur erfordert das Auffüllen der nicht relevanten Spalten im Bereich der Messung mit Nullen. Die Form bietet keinen Vorteil für das Netzwerktraining und wurde ausschließlich mit dem Fokus auf die indexbasierte Trackzuordnung gewählt. Ebenfalls erhöht sich die Datengröße unnötig, weshalb für eine Weiterentwicklung des **SANT** Modells die Struktur der Inputdaten optimiert werden könnte. Die aktuelle Darstellung entspricht eher einer optimierten Datenstruktur für eine menschliche Zuordnung.

	<pre>timestep = 542 disp(truth.data_single_noisy{timestep});</pre>																																																																																																					
Tracks	<table border="1"> <thead> <tr> <th></th><th>posX</th><th>posY</th><th>Yaw</th><th>dimX</th><th>dimY</th><th>20.3384</th><th>3.6731</th><th>11.5837</th><th>18.2327</th><th>15.0802</th><th>27.4272</th><th>26.8184</th><th>38.0321</th><th>46.6030</th><th>34.0985</th></tr> </thead> <tbody> <tr> <td>posX</td><td>0.0834</td><td>-4.2530</td><td>-8.2726</td><td>-8.2118</td><td>-3.9022</td><td>-8.6829</td><td>-4.5987</td><td>-5.1736</td><td>-4.9422</td><td>-12.7658</td><td></td><td></td><td></td><td></td><td></td></tr> <tr> <td>posY</td><td>-1.5588</td><td>-1.5720</td><td>-1.5768</td><td>-1.5730</td><td>-1.5712</td><td>-1.5838</td><td>-1.5708</td><td>-1.5740</td><td>-1.5777</td><td>-1.5847</td><td></td><td></td><td></td><td></td></tr> <tr> <td>Yaw</td><td>3.8404</td><td>6.1654</td><td>3.9744</td><td>3.3710</td><td>3.9243</td><td>4.0531</td><td>3.6252</td><td>3.4943</td><td>4.3322</td><td>4.5225</td><td></td><td></td><td></td><td></td></tr> <tr> <td>dimX</td><td>1.7604</td><td>2.0163</td><td>1.7027</td><td>1.6578</td><td>1.6691</td><td>1.6099</td><td>1.6919</td><td>1.5765</td><td>1.6378</td><td>1.6949</td><td></td><td></td><td></td><td></td></tr> <tr> <td>dimY</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </tbody> </table>												posX	posY	Yaw	dimX	dimY	20.3384	3.6731	11.5837	18.2327	15.0802	27.4272	26.8184	38.0321	46.6030	34.0985	posX	0.0834	-4.2530	-8.2726	-8.2118	-3.9022	-8.6829	-4.5987	-5.1736	-4.9422	-12.7658						posY	-1.5588	-1.5720	-1.5768	-1.5730	-1.5712	-1.5838	-1.5708	-1.5740	-1.5777	-1.5847					Yaw	3.8404	6.1654	3.9744	3.3710	3.9243	4.0531	3.6252	3.4943	4.3322	4.5225					dimX	1.7604	2.0163	1.7027	1.6578	1.6691	1.6099	1.6919	1.5765	1.6378	1.6949					dimY													
	posX	posY	Yaw	dimX	dimY	20.3384	3.6731	11.5837	18.2327	15.0802	27.4272	26.8184	38.0321	46.6030	34.0985																																																																																							
posX	0.0834	-4.2530	-8.2726	-8.2118	-3.9022	-8.6829	-4.5987	-5.1736	-4.9422	-12.7658																																																																																												
posY	-1.5588	-1.5720	-1.5768	-1.5730	-1.5712	-1.5838	-1.5708	-1.5740	-1.5777	-1.5847																																																																																												
Yaw	3.8404	6.1654	3.9744	3.3710	3.9243	4.0531	3.6252	3.4943	4.3322	4.5225																																																																																												
dimX	1.7604	2.0163	1.7027	1.6578	1.6691	1.6099	1.6919	1.5765	1.6378	1.6949																																																																																												
dimY																																																																																																						
Messung	<table border="1"> <thead> <tr> <th></th><th>posX</th><th>posY</th><th>Yaw</th><th>dimX</th><th>dimY</th><th>36.4012</th><th>0</th><th>0</th><th>0</th><th>0</th><th>0</th><th>0</th><th>0</th><th>0</th><th>0</th></tr> </thead> <tbody> <tr> <td>posX</td><td>-13.0804</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr> <td>posY</td><td>-1.6071</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr> <td>Yaw</td><td>4.7688</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr> <td>dimX</td><td>1.7846</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr> <td>dimY</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </tbody> </table>												posX	posY	Yaw	dimX	dimY	36.4012	0	0	0	0	0	0	0	0	0	posX	-13.0804	0	0	0	0	0	0	0	0	0	0	0	0	0	posY	-1.6071	0	0	0	0	0	0	0	0	0	0	0	0	0	Yaw	4.7688	0	0	0	0	0	0	0	0	0	0	0	0	0	dimX	1.7846	0	0	0	0	0	0	0	0	0	0	0	0	0	dimY														
	posX	posY	Yaw	dimX	dimY	36.4012	0	0	0	0	0	0	0	0	0																																																																																							
posX	-13.0804	0	0	0	0	0	0	0	0	0	0	0	0	0																																																																																								
posY	-1.6071	0	0	0	0	0	0	0	0	0	0	0	0	0																																																																																								
Yaw	4.7688	0	0	0	0	0	0	0	0	0	0	0	0	0																																																																																								
dimX	1.7846	0	0	0	0	0	0	0	0	0	0	0	0	0																																																																																								
dimY																																																																																																						
	<pre>fprintf('GT Zuordnung: %d',truth.label_single{timestep});</pre>																																																																																																					
	<pre>GT Zuordnung: 10</pre>																																																																																																					

Abbildung 3.13: **SANT**, Struktur Inputdaten

Die erzeugten **SO** der **GT** Daten wurden zu Beginn der Netzwerkentwicklung mit einem zusätzlichem Rauschen von maximal 7.5% vom aktuellen Zustandswert verändert (Rauschintensitätsuntersuchung folgt in 3.4.3). Die Werte des erzeugten **SO** stimmen somit nicht exakt mit den Werten des **GT** Tracks überein, wodurch das Netzwerk eine Zuordnung über den Kontext (historischen Daten der Sequenz) und einen naheliegenden Wertevergleich lernen sollte.

Pro Zeitschritt konnte aus jedem existierenden Track ein Sample generiert werden. Der so erzeugte **SANT** Datensatz besteht aus 25216 Samples. Der KITTI Datensatz liefert für die gewählte Fahrzeugklassen Cars und Vans eine maximale Anzahl an 16 Tracks in einem Zeitschritt. Innerhalb des Datensatzes wurden zusätzlich zu den bestehenden **SOs** welche indexbasiert einem Track zuzuordnen sind, **SOs** erzeugt mit Zustandswerten,

welche keinem der existierenden Tracks zugeordnet werden sollten (Indexklasse 99). Das Ziel dieser Erweiterung ist es dem Netzwerk die Fähigkeit zu geben ein neu erscheinendes Sensor Objekt (**SO**) als unbekannt zu klassifizieren und somit keinem der erfassten Tracks zuzuweisen. Der Datensatz wurde iterativ erweitert mit dem Ziel Abhängigkeiten im Sequenzverlauf zu einem großen Teil nicht zu beeinflussen. Daher wurden die neuen **SO** mit dem **GT** Label 99 mit anteilig ca. 5% verteilt. Inklusive Erweiterung der 99er Klasse wurde der Gesamtdatensatz (27100 Samples) in Trainings- (80%, 21680 Samples), Validierungs- (10%, 2710 Samples) und Testdaten (10%, 2710 Samples) aufgeteilt.

#### 3.4.2 Netzwerkentwicklung

Wie bei der **SPENT** Entwicklung wurde zu Beginn ein flaches, leicht zu trainierendes Netzwerk für die Aufgabe der Datenassoziation konzipiert. Darstellung 3.14 zeigt den Netzaufbau, bestehend aus fünf sequentiell angeordneten Schichten.

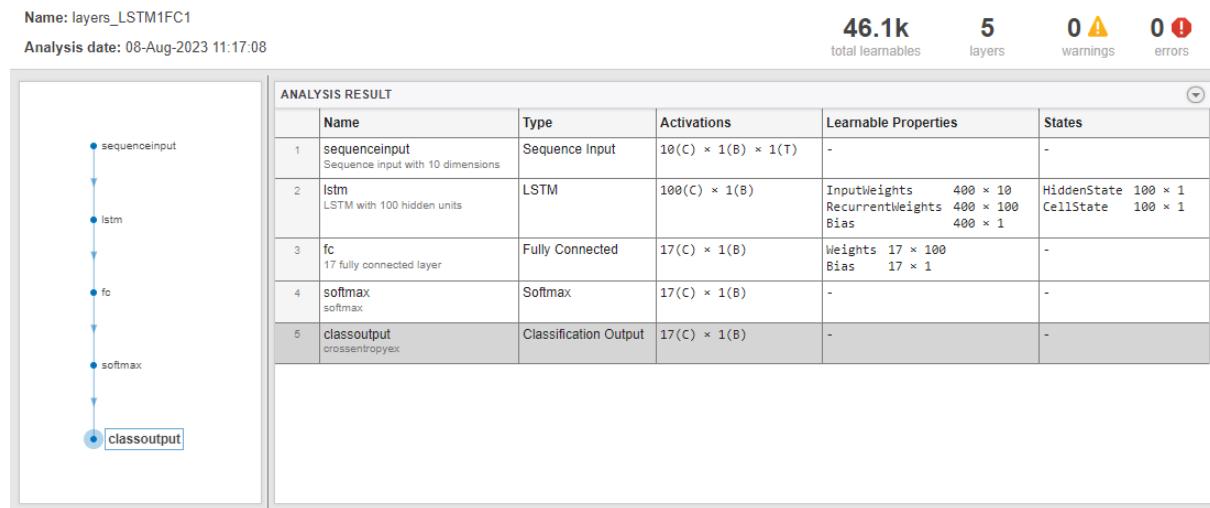


Abbildung 3.14: **SANT**, Variante LSTM1FC1, Analysedarstellung

Um die Fehlerquellen zu Beginn der Modellentwicklung und die notwendige Trainingszeit zu reduzieren, wurde **SANT** mit den Zustandswerten x, y Koordinaten, Gierwinkel, Länge und Breite des jeweiligen Objekts entworfen. Auf die berechneten Relativgeschwindigkeiten wurde verzichtet. Entsprechend sind 10 Features (5 Zustandswerte Tracks, 5 Zustandswerte **SO**) in der Inputschicht als Channel (C) definiert. Für ein erstes Netzwerktraining ist ein **LSTM** (mit 100 Hiddenunits) und ein **FC** Layer als lernfähiger Netzwerkteil definiert worden. Das Netzwerk ist als Sequenz-to-Classification ausgelegt, wobei eine Sequenz eine  $m * n$  (siehe 3.13) Zuordnung darstellt. Ziel ist es eine Klassifizierung durchzuführen, somit einen Ausgabewert pro Zuordnungssequenz

(Zeitschritt) als Netzwerkoutput zu erzeugen. Nach Mathworks [20] muss das LSTM Layer entsprechend mit dem Outputmode 'last' parametert werden. Die vollständig verknüpfte Schicht FC gibt über die Anzahl der Ausgabewerte die Klassenanzahl vor. Die 17 Klassenwerte werden in der Softmax Schicht durch Anwendung der Softmaxfunktion in eine eindeutige Wahrscheinlichkeitsverteilung berechnet (wie in Kapitel 2.2.1 beschrieben). Durch Anwendung der Cross-Entropie-Operation in der Ausgabeschicht (Classout) kann so ein Netzwerk auf einen korrekten Zuordnungswert (GT Klassenindex) trainiert werden.

**SANT Kostenfunktion (lossfunction)** Die Cross-Entropie-Kostenfunktion (Kreuzentropie) berechnet den Cross-Entropie-Verlust zwischen Netzvorhersagen und Zielwerten für die eindeutige Zuordnungsaufgabe (für sich gegenseitig ausschließende Klassen). Dabei wird mittels One-Hot-Kodierung die Klasse binär in einem Vektor dargestellt und somit ein 1-zu-n Code generiert. Nach folgender Formel werden die Crossentropie-Verlustwerte für jeden Eingabewert  $Y_j$  und zugehörigen Zielwert (Targetvalue)  $T_j$  elementweise berechnet:

$$loss_j = -(T_j \ln Y_j + (1 - T_j) \ln(1 - Y_j)) \quad (3.5)$$

Um einen Skalar  $loss$  zu erhalten werden alle Verlustwerte  $loss_j$  aufsummiert und durch die Anzahl der Samples  $N$  geteilt. Optional können die Verlustwerte von definierten Samples mit dem Gewichtungsfaktor  $w_j$  gewichtet werden:

$$loss = \frac{1}{N} \sum loss_j w_j \quad (3.6)$$

Eine entsprechende Nutzung des Gewichtungsfaktors  $w_j$  kann bei Datensätzen mit unausgewogenen (imbalanced) Klassenverteilung hilfreich sein

**SANT Ergebnisanalyse** Die Trainingsoptionen wurden entsprechend der Netzwerk-aufgabe gewählt. Eine Liste aller betrachteten Optionen inklusive Skriptauszug sind im Anhang zu finden. Der Trainingsdatensatz weist 2.5% Noise auf. Die Zuordnungsergebnisse des trainierten Single Association Network (SANT) sind in Abbildung 3.15 in Form einer Confusionmatrix dargestellt.

Die Y-Achse beschreibt den wahren Zuordnungsindex (True Class), die X-Achse von SANT assoziierten Index (Predicted Class). Mit dieser Darstellungsform können die korrekt zugeordneten Samples, sowie die falsch assoziierten Samples quantifiziert werden. Für den Trackindex 13 besitzt der Trainingsdatensatz beispielsweise 145 Samples (Summe der True Class Reihe). Die Auswertungstabelle auf der rechten Seite beinhaltet

### 3 Ergebnisse

---

die prozentuale Verteilung wie viele Samples einer Klasse korrekt (blau = True Positive (**TP**)) und fälschlicherweise einer Anderen (rot = False Positive (**FP**)) zugeordnet wurden. Die im unteren Bildbereich aufgetragene Auswertungstabelle liefert die prozentuale Verteilung der richtig (blau = **TP**) und fälschlich einem betrachteten Trackindex assoziierten **SO** (rot = False Negative (**FN**)).

True Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
	5030	181	21	28	57	40	18	9	25	7	4	6	6	2			
1	4	181	21	28	57	40	18	9	25	7	4	6	6	2			
2	8	9	3876	54	10	1	2										
3	8	5	64	2994	60	4	2	2									
4	2	13	8	187	2175	64	8	3									
5	1		1	23	354	1275	46	15	7								
6		3	4	7	39	534	614	81	23	1		1					
7		6		2	9	109	268	516	106	7		2	2				
8			2		10	21	38	231	437	16	9	20	4				
9		1	4			6	6	45	284	137	45	45	6				
10		2					9	59	62	117	138	9	2				
11				1				12	11	28	203	52		2			
12					2				4	3	8	112	77	1	33		
13						1			4			28	22	14	76		
14							2		1			37	15	23	12		
15							5					20			2		
16																	

99.5%	81.9%	96.8%	90.8%	79.4%	61.9%	60.9%	56.6%	45.4%	56.1%	55.5%	33.2%	39.9%	57.5%	60.8%	100.0%	
0.5%	18.1%	3.2%	9.2%	20.6%	38.1%	39.1%	43.4%	54.6%	43.9%	44.5%	66.8%	60.1%	42.5%	39.2%		
1	99	2	3	4	5	6	7	8	9	10	11	12	14	13	15	16

Predicted Class

Abbildung 3.15: SANT, Variante LSTM1FC1, Confusion Matrix

#### 3.4.3 Optimierungen

Die Zuordnungsperformance der ersten **SANT** Konzeptentwicklung lässt darauf schließen, dass das Netzwerk die umzusetzende Aufgabe bisher nicht erlernen konnte. Es werden daher die folgenden Maßnahmen mit dem Ziel der Performanceoptimierung implementiert:

1. **Datenerweiterung** - umfasst die Erweiterung der Samples pro Zeitschritt
2. **Netzwerkvarianten** - empirische Ermittlung der Netzwerkarchitektur (Netzwerktiefe, -breite und Layeraufbau) sowie Optimierung der Netzwerk- und Trainingsparameter

Zum Abschluss der Entwicklung wird eine Untersuchung der Rauschintensität durchgeführt. Diese dient dazu die Robustheit der erlernten Zuordnungsfähigkeit zu prüfen.

#### Optimierungsmaßnahme 1 - Datenerweiterung

Im ersten Preprocessing wurde pro Zeitschritt die in Abbildung 3.13 dargestellt Inputdatenstruktur erzeugt. Dafür wurden jeweils die Zustandswerte eines pseudo-zufällig ausgewählten Tracks verauscht und als Sensor Objekt (**SO**) gespeichert. Pro Zeitschritt existieren je nach Beobachtungsaufkommen mehrere Tracks. Somit konnte der Gesamtdatensatz für das Single Association Network (**SANT**) auf 39315 Samples erhöht werden. Es werden somit alle verfügbaren Tracks pro Zeitschritt genutzt.

#### Optimierungsmaßnahme 2 - Netzwerkvarianten

Der erweiterte Datensatz dient der experimentellen Netzwerkentwicklung als Datenbasis. Der Datensatz wurde mit einem Splitt 90 / 5 / 5 aufgeteilt und die Sensorobjekte **SO** des Trainingsdatensatzes um 5% wie beschrieben pseudo-zufällig um den aktuellen Zustandswert verauscht.

Die Ergebnisse zeigen, dass durch die Optimierungsmaßnahmen eine Validationsgenauigkeit von ca. 95% erreicht werden konnte. Die Netzwerkvariante mit zwei **BILSTM** Schichten und einer Fully Connected (**FC**) Schicht liefert in diesem Setup die besten Ergebnisse. Der Aufbau ist in der Ergebnistabelle 3.17 abgebildet. Das Netzwerk besteht aus insgesamt 6 Schichten. Dabei ist die erste der beiden **BILSTM** Schichten mit dem Ausgabemodus 'sequence' parametriert. Das bedeutet, dass diese Schicht den gesamten Inputvektor verarbeitet und an die darauffolgende Schicht übergibt. Dieser Ansatz wurde untersucht, da während des Trainings kontextbezogene Informationen über die

### 3 Ergebnisse

Hiddenstates in dieser Schicht gebildet werden können. Die folgende Abbildung zeigt einen Auszug der untersuchten Netzwerkvarianten.

Trial	NUMHIDUNITS	NETWORK	Training Accuracy (%)	Training Loss	Validation Accuracy (%)	Validation Loss
8	100	BILSTM_seq_last_FC1	97.7860	0.0683	96.1993	0.1150
2	200	BILSTM_seq_last_FC1	97.0480	0.1420	95.7565	0.1321
5	200	BILSTM_seq_last_FC1	97.0480	0.1420	95.7565	0.1321
7	150	BILSTM_seq_last_FC1	92.9889	0.2828	95.7196	0.1724
11	100	BILSTM1_FC1	82.2878	0.4210	82.5461	0.4884
10	150	BILSTM1_FC1	90.0369	0.3598	82.2878	0.4406
9	200	LSTM1_FC1	71.5867	0.7682	74.1328	0.6525
4	150	LSTM1_FC1	69.7417	0.8507	70.6642	0.7598

Abbildung 3.16: SANT, Ergebnistabelle, Auszug der untersuchten Netzwerkvarianten

Nach Mathworks.inc [20] werden je nach definierter Größe des Minibatches die trainierbaren Parameter der Schicht angepasst. In der zweiten **BILSTM** Schicht wurde der Ausgabemodus 'last' konfiguriert. Diese Schicht ist damit in der Lage eine Sequenz als Input zu erhalten, und einen einzelnen Wert bzw. Wertevektor auszugeben. Diese Form der Dimensionsreduzierung ist notwendig, um eine entsprechende Klassifizierung durchzuführen. Die weitere Verarbeitung der Ausgabe erfolgt wie bereits beschrieben durch den bekannten *FC > softmax > classout* Stack.

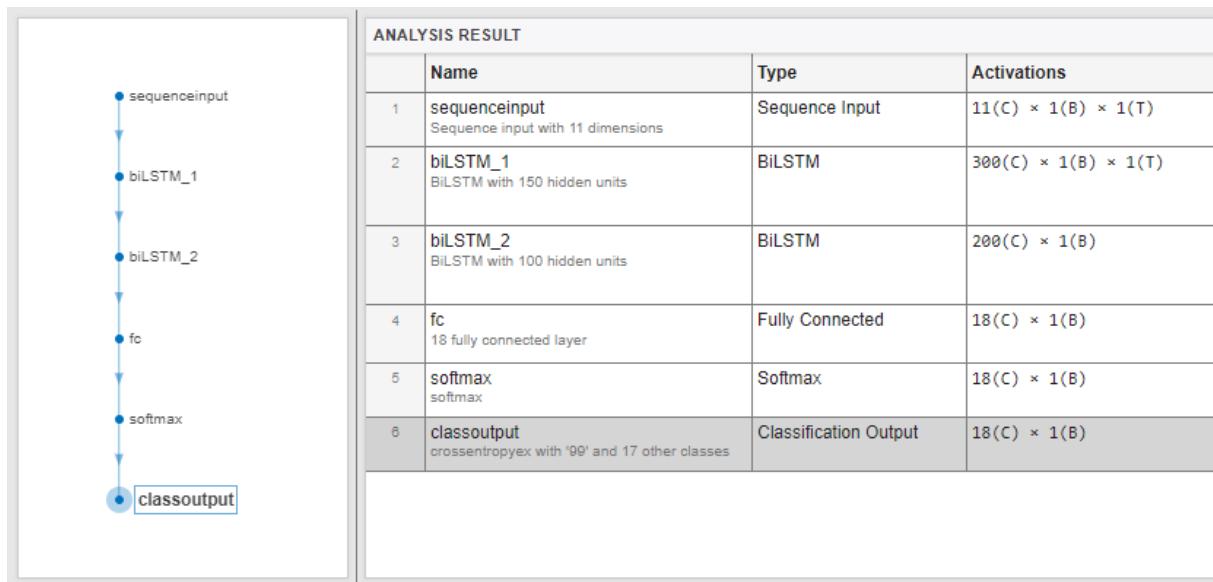


Abbildung 3.17: SANT, BilstmSeqLastFc1, Netzwerkanalyse

Abbildung 3.18 zeigt weiter die Confusionmatrix auf Basis der 5% verrauschten Validationsdaten. Dabei ist zu erwähnen, dass die Netzwerkperformance von 95% Validation Accuracy mit einem verrauschten Trainingsdatensatz von 1-5% erzielt wurde (bezogen auf einen gemittelten Wert an 10 Trainingsdurchläufen). Die Confusionmatrix zeigt, dass eine generelle Zuordnungsfähigkeit gelernt wurde. Tracks mit der idx 1 bis einschließlich 6, besitzen ein vorkommen von 10.30 - 14.61% (siehe Auflistung im unteren

### 3 Ergebnisse

Bereich der Abbildung). Insgesamt repräsentieren diese somit ca. 76% der Samples des Verifikationsdatensatzes. In diesem Bereich gab es lediglich einmal eine falsche Zuordnung (True Class 99, Predicted Class 13). Index 99 steht dabei für ein neues SO. Der restliche Anteil teilt sich zwischen den Trackindizes 7 - 14 auf. Der Validationsdatensatz beinhaltet keine Samples mit gleichzeitig existierenden Tracks von einer Anzahl 15 oder 16. Die geringe Anzahl der verfügbaren Tracks bzw. Realdaten wird mit dieser Analyse noch einmal deutlich. Wie bereits in Kapitel 3.3.1 beschrieben, wurden die Samples mit sehr geringem Vorkommen für den Trainingsdatensatz ausgewählt.

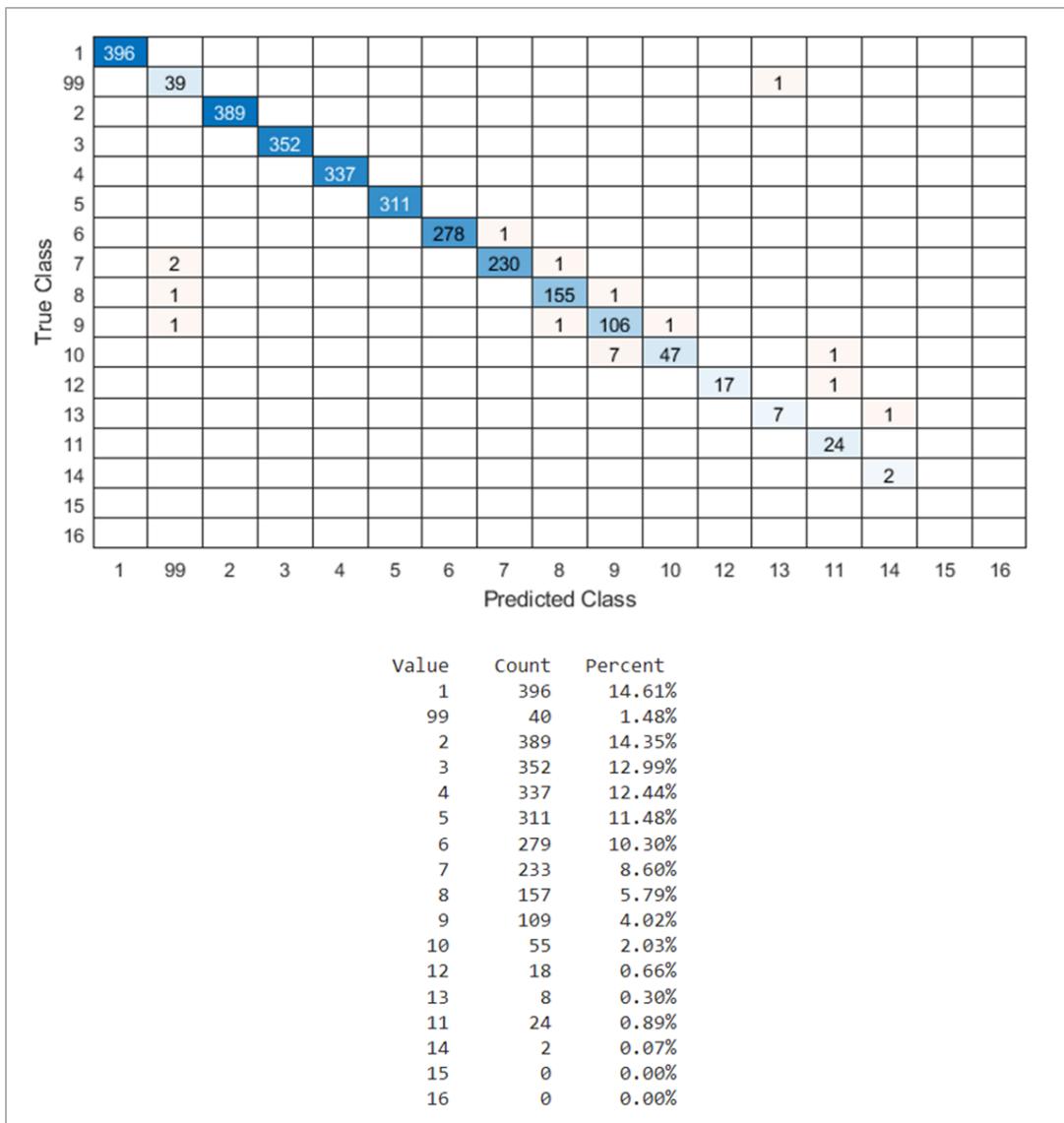


Abbildung 3.18: [SANT](#), BilstmSeqLastFc1, Confusionmatrix

**Untersuchung der Rauschintensität der GT Daten** Des Weiteren wurde untersucht, wie robust das Netzwerk auf die Rauschintensität der Sensor Objekte SO reagiert (Abweichung zwischen GT und Messwert). Die Trainingsdaten wurden dazu wie in Kapitel

### 3 Ergebnisse

---

3.4.1 beschrieben verrauscht. D.h. das **SO**, welches das Netzwerk einem Track einer bestehenden Trackliste zuordnen soll, besitzt mit steigender Rauschintensität eine höhere absolute Werteabweichung. Die erzeugten **GT** Daten des Validierungsdatensatzes bleiben weiterhin unverrauscht. Der Ergebnisauszug der Rauschintensitätsuntersuchung ist in Tabelle 3.19 abgebildet. Es ist ein Auszug aus der Ergebnistabelle dargestellt. Die Spalte *PERCENTAGERANGE* gibt an in welchem Bereich die **SO** der Trainingsdaten verrauscht wurden (0.16 entspricht 16%).

Trial	PERCENTAGERANGE	Validation Accuracy (%)	Validation Loss
7	0.16	85.1292	0.4653
8	0.32	67.1956	0.9995
3	0.64	34.7232	1.8197

Abbildung 3.19: **SANT**, Untersuchung der Rauschintensität, Übersicht der Ergebnisse

Die Ergebnisse dieser Untersuchung zeigen, dass die Zuordnungsfähigkeit von **SANT** mit der Zunahme des Rauschens abnimmt. Damit konnte nachgewiesen werden, dass das Netzwerk eine realistische Zuordnung für den verrauschten Datensatz bis 5% lernen kann. Bei einem Trainingsdatensatz mit 10 bis 15% verrauschten Daten, sinkt die Validation Accuracy auf ca. 85%. Das Verhalten ist nachvollziehbar und wird weiter mit dem Extremfall von 64% begründet. Die Performance (34.7% Validation Accuracy) zeigt, dass auf Basis dieser Trainingsdaten weder eine kontextbezogene oder absolute Zuordnungslogik gelernt werden kann. Ein Positions倅 von 10m kann in diesem Datensatz beispielsweise um bis zu 6.4m vom **GT** Wert abweichen. Dieses gelernte (willkürlich erscheinende) Zuordnungsverfahren setzt das Netzwerk dann ebenfalls für den unverrauschten Validierungsdatensatz um. Zusätzlich wurde ermittelt, dass kein Rauschen in mehreren Trainingsdurchläufen zu overfitting führte (Training Accuracy 100%, Validation Accuracy 90%).

Für die Evaluation in einem Multi Object Tracking (**MOT**) Framework wird das Single Association Network (**SANT**) verwendet, welches die in Abbildung xx dargestellten Ergebnisse erzielt hat. Damit wird das **SANT** verwendet, welches mit einem 1-5% verrauschten Datensatz trainiert wurde und eine Validation Accuarcy von 93-97% aufweist.

### 3.4.4 Zusammenfassung

Kapitel 3.4 stellt das Vorgehen und einige Ergebnisse der durchgeföhrten Experimente dar, welche im Rahmen der Entwicklung eines Single Association Network (**SANT**) durchgeföhr wurden. In Abb. 3.20 ist das entwickelte Netzwerk inklusive der Input- und Output Datenstruktur noch einmal übersichtlich dargestellt.

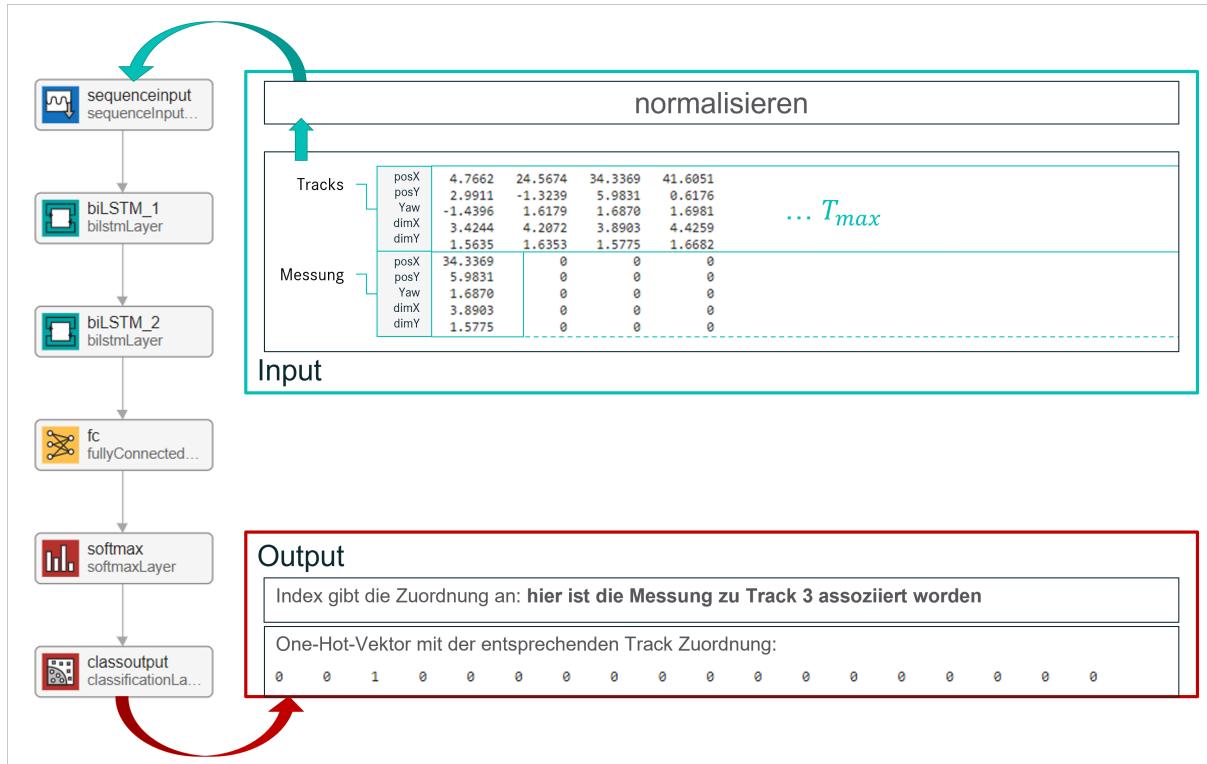


Abbildung 3.20: Single Association Network (**SANT**), Input- Output Struktur

Zusammengefasst wurden folgende Arbeiten durchgeföhr bzw. Ergebnisse realisiert:

- Es wurde eine geeignete Datenstruktur erarbeitet, um im Kontext des verwendeten **ML** Ansatzes das Problem der Datenassoziation lösen zu können.
- Die verfügbaren Daten wurden analysiert und entsprechend aufbereitet, um ein konvergierendes Netzwerktraining zu realisieren.
- Verschiedene Netzwerkarchitekturen wurden implementiert und trainiert, einschließlich solcher mit zusätzlichen Aktivierungsfunktions- und Normalisierungsschichten.
- Die Ergebnisse zeigen, dass das entwickelte Netzwerk eine hohe Zuordnungsfähigkeit aufweist. Mit der Annahme der Verschlechterung der Genauigkeit für hohe Distanzen, wie es beispielsweise bei Kameras oft der Fall ist, wurden das Netzwerk erfolgreich auf einen bis zu 5% verrauschten Datensatz trainiert. Das

Rauschen bezieht sich auf Grund dieser Annahme auf den Absolutwert. Bezogen auf den Zustandswert der x Position ist beispielsweise bei einem Wert von 80m eine Abweichung von bis zu 4m im Datensatz möglich. Die Zuordnungsfähigkeit bezieht sich auf die zu Beginn definierten 1 zu  $n$ , sowie 0 zu  $n$  Zuordnung.

- Damit konnte die zu Beginn aufgestellte These bestätigt werden, dass eine Zuordnung ohne vorgegebenes Distanzmaß realisiert werden kann.
- Bezogen auf den Bereich der untersuchten Netzwerkvarianten, zeigen Netzwerkvarianten mit zwei **BILSTM** Schichten und einer **FC** Schicht die besten Ergebnisse, mit einer Validationsgenauigkeit von ca. 95%.
- Das entwickelte Modell operiert auf **SO** Ebene und ermöglicht somit eine Integration in ein bestehendes **TbD MOT** Framework.

**Ausblick** Die durchgeführten Optimierungsmaßnahmen stellen einen Auszug der Möglichkeiten in der **ML** Entwicklung dar. Es könnte vorteilhaft sein, weitere Experimente mit unterschiedlichen Netzwerkarchitekturen und Hyperparametern durchzuführen. Besonders zu empfehlen ist die Erweiterung des Datensatzes. Gelabelte Sensorrealdaten sollten für den weiteren Entwicklungsprozess berücksichtigt werden. Die Verwendung von fortschrittlicheren Techniken zur Datenanreicherung und -vorverarbeitung könnte ebenfalls untersucht werden.

Eine tiefere Analyse der Fehlerfälle und die Entwicklung von Strategien zur Adressierung dieser Fehler kann ebenfalls als potenzielle Weiterentwicklungsbereiche betrachtet werden. Untersuchungen bzgl. der kontextbezogenen Zuordnungsfähigkeiten sind zusätzlich empfehlenswert. Es könnte beispielsweise ein Testkatalog mit entsprechenden Testdaten definiert werden, welcher eine gezielte Optimierung spezifischer Situationen ermöglicht. Dabei ist darauf zu achten, dass die Gesamtperformance mit einem Hauptkatalog geprüft wird, welcher alle für den Einsatz relevanten Daten in ausreichender Menge und Verteilung darstellt. Das kontextbezogene Lernen ist in jedem Fall zu optimieren, indem Sequenzen mit entsprechender Länge (Minibatchsize) für die Entwicklung verwendet werden können. Des weiteren kann untersucht werden, ob eine Track zu Messungen zuordnen für eine Integration in einen bestehenden **MOT** Framework von Vorteil sein könnte, im Vergleich zu der implementierten Messung zu Track Zuordnung.

## 3.5 Multi Association Network (**MANTa**) - Assoziation mehrerer Messungen zu erfassten Tracks

Mit dem entwickelten Single Association Network (**SANT**) konnte gezeigt werden, dass eine datenbasierte Assoziationslogik von einem Modell gelernt werden kann. Es wurde weiter das Ziel verfolgt, ein Netzwerk zu entwickeln, um eine Anzahl von  $m$  Sensorobjekten **SO** einer bestehenden Anzahl an  $n$  Tracks in einem Operationsschritt zuzuordnen. Ziel ist ein Multi Association Network (**MANTa**) zu entwickeln, welches die folgenden Zuordnungsmöglichkeiten lösen kann:

- **1 zu n** - ein **SO** zu  $n$  Tracks
- **m zu 1** -  $m$  **SOs** zu einem Track
- **m zu n** -  $m$  **SO** zu  $n$  Tracks
- **m zu 0** -  $m$  **SO** zu keinen Tracks
- **0 zu n** - keine **SO** zu  $n$  Tracks
- **0 zu 0** - keine **SO** zu keinen Tracks

Hinsichtlich der Integration eines Multi Association Network (**MANTa**) in einen Multi Object Tracking (**MOT**) Framework kann die Frage gestellt werden, ob eine 0 zu  $n$  und 0 zu 0 Zuordnung eine zu lösende Aufgabe ist. Bei keinem neu erfassten **SO** kann die Prädiktion des letzten Operationsschritts so lange weitergeführt werden bis der Track auf Basis der sinkenden Existenzwahrscheinlichkeit innerhalb des Trackmanagement-moduls gelöscht wird. Bei keinen erfassten Tracks und Sensorobjekten **SO** muss der Assoziationsalgorithmus bzw. das **MANTa** nicht aufgerufen werden. Obwohl diese Zuordnungsmöglichkeiten somit über die Programmstruktur im **MOT** Framework gelöst werden können, werden diese Optionen ebenfalls berücksichtigt. Somit soll erreicht werden, dass das Netzwerk zusätzlich mit nicht mehr vorhandenen **SO** und Tracks umgehen zu lernt.

### 3.5.1 Datenvorverarbeitung

Der Datensatz für das Training und die Validierung von **MANTa** wurde entsprechend der beschriebenen Zielsetzung erstellt. Abbildung 3.21 zeigt die Inputdatenstruktur mit entsprechenden Zuordnungsaufgaben im dargestellten Zeitschritt (85) der Sequenz 20 des KITTI Datensatzes. Über alle Sequenzen wurden die jeweiligen Tracks pro Zeitschritt extrahiert. Die extrahierten Tracks wurden jeweils mit einem Rauschen von

### 3 Ergebnisse

0 bis 7.5% modifiziert und im Anschluss normiert. Die so erhaltenen Werte wurden wie in Abb. 3.21 dargestellt als Messungen in pseudo-zufälliger Reihenfolge pro Zeitschritt einer  $F * T_{max}$  Matrix zugeordnet. Wobei  $F$  die Anzahl der Feature darstellt (hier 10). Die Featureanzahl ergibt sich aus den Zustandswerten pro Track und SO Set.  $T_{max}$  steht für die maximale Anzahl an existierenden Tracks pro Zeitschritt. Für den definierten Untersuchungsfall mit Cars und Vans ergibt sich aus dem KITTI Datensatz ein  $T_{max} = 16$ . Im dargestellten Zeitschritt der Sequenz existieren 7 Tracks. Jeder Track erhält in diesem Zeitschritt eine neue Messung, zusätzlich wurde ein weiteres Objekt entdeckt (neues SO). Die Groundtruth (GT) Zuordnung ist im unteren Teil des Ausschnitts ausgegeben. Die Werte zeigen den korrekten Index, der Zuordnung für 1 bis 16 ( $T_{max}$ ) bezogen auf das existierende Trackset. Der Wert 99 wurde als Klassenindex für neue bzw. nicht zuzuordnende Messungen definiert. Entsprechend ist die Messung in der zweiten Spalte ein neues Objekt und sollte keinem bestehenden Track zugeordnet werden.

Das Outputformat von MANTa ist im unteren Bildrand abgebildet. Das Multi Association Network (MANTa) soll mehrere Messungen zu mehreren Tracks zuordnen. Entsprechend wurde ein Ausgabevektor (One-Hot-Vektor) mit 288 Elementen pro Zeitschritt erzeugt. Die Größe von  $OH_{max} = 288$  ergibt sich aus der maximalen Trackanzahl  $T_{max} = 16$  und der Anzahl der möglichen Zuordnungsklassen  $classes = 18$ . Die Zuordnungsklassen ergeben sich aus der beschrieben Indexklasse 1 bis 16, 99 und 0. 0 stellt den Fall dar, dass keine Messung existiert bzw. zugeordnet werden sollte. In Abb. 3.21 entsprechend die Spalten 9,10 und die nicht dargestellten restlichen Spalten bis  $T_{max}$ .

	<pre>timestep = 85 ━━━━ ;</pre> <pre>fprintf('MANTa Inputdatenstruktur:');</pre>
	<pre>MANTa Inputdatenstruktur:</pre> <pre>disp(truth.MANTa_noisy_norm_TrackAndSoSet_cellarray{timestep}(1:10,:))</pre>
Tracks	<pre>posX posY Yaw dimX dimY</pre> <pre>-0.7281 1.1261 1.1341 1.0073 0.8762 1.0519 2.7315 0 0 0</pre> <pre>0.6114 0.8854 2.1352 3.1433 4.1329 -1.3647 0.1746 0 0 0</pre> <pre>-0.7684 1.9539 -1.5988 -1.5957 -1.6044 0.3685 1.2819 0 0 0</pre> <pre>0.3706 0.0171 0.1318 1.0860 -0.1352 -0.9380 0.0755 0 0 0</pre> <pre>-0.4095 -0.4492 -0.0866 -0.0866 0.3964 -0.4169 -2.5164 0 0 0</pre>
	$\dots T_{max}$
Messungen	<pre>posX posY Yaw dimX dimY</pre> <pre>1.4396 2.3841 -0.5813 1.3003 2.3371 0.8622 0.6363 1.2752 0 0</pre> <pre>2.3656 -0.2022 0.7101 -1.4649 0.1399 0.7960 3.7876 3.4269 0 0</pre> <pre>-1.6660 1.1993 -0.7789 0.4082 1.1575 1.7778 -1.4919 -1.6901 0 0</pre> <pre>0.7725 -1.0819 1.0208 -0.4333 -0.5219 -0.5735 -0.7097 1.7757 0 0</pre> <pre>0.9214 -2.3539 0.5630 0.5142 -3.2912 -1.3873 -0.6054 0.8940 0 0</pre>
	$\dots T_{max}$
	<pre>fprintf('GT Zuordnung:');</pre>
	<pre>GT Zuordnung:</pre>
One-hot-Vektor 1:17 (von 1:288)	<pre>disp(truth.MANTa_labels{timestep});</pre> <pre>3 99 1 6 7 2 5 4</pre> <pre>n = 1 ━━━━ ; disp(truth.MANTa_OneHot_cellarray{timestep}((n-1)*truth.max_</pre> <pre>0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 ... OH_{max}</pre>

Abbildung 3.21: MANTa, Datenstruktur, Input / Output

### 3.5.2 Netzwerkentwicklung

Die schematische Darstellung 3.22 zeigt eine Variante der entwickelten MANTa Netzwerkarchitekturen. Der Graphplot der MANTa Architektur ist auf der rechten Seite abgebildet. Auf der linken Seite ist der einfache Schichtaufbau von SANT dargestellt.

Die beiden BiLSTM Schichten verarbeiten die Inputdaten wie bereits im Kapitel 3.4.1 eingeführt. Die Aufgabe der Assoziation von einer SO-Liste zu einer Track-Liste erfordert pro Track einen eigenen Netzwerkelement. Diese Erweiterung ist in der Grafik 3.22 in rot gekennzeichnet. Pro Track (von 1 bis  $T_{max}$ ) ist das MANTa mit dem bekannten Fully Connected (FC) → softmax Stack entwickelt worden. Jeder softmax Output besteht aus einem Vektor mit  $classes = 18$  Elementen, welcher die Wahrscheinlichste Zuordnung darstellt (siehe 2.2.1). Somit kann pro Track eine Einzelzuordnung realisiert werden. Die Vektoren 1 bis  $T_{max}$  werden im Concatenation Layer miteinander verknüpft. Somit wird ein Vektor mit 288 Elementen erzeugt, wobei jeweils 18 Elemente die wahrscheinlichste Zuordnung einer Messung zu einem Track darstellen.

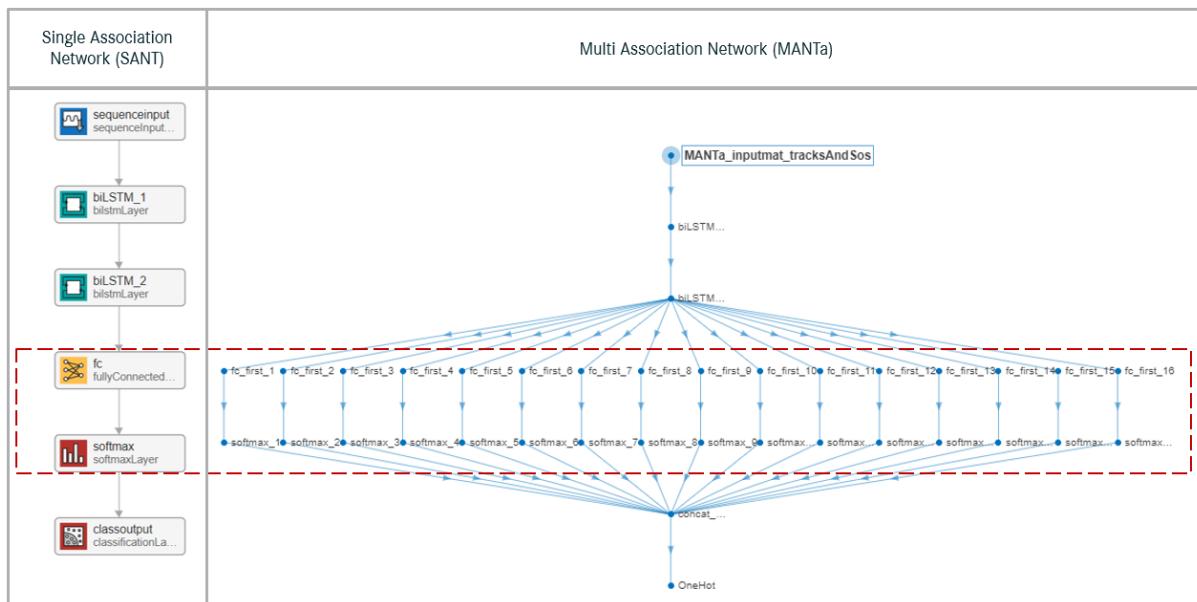


Abbildung 3.22: MANTa, Schematische Darstellung der Netzwerkarchitektur

**Kostenfunktion** Entsprechend des Formats der GT und Output Daten des Netzwerks wurde die Kostenfunktion implementiert. Das Ziel pro Track genau eine wahrscheinlichste Zuordnung zu erhalten wurde dabei mit der folgenden Erweiterung umgesetzt. Die bereits eingeführt Cross-Entropie-Kostenfunktion wurde verwendet, und mittels zusätzlicher Iteration pro Zeitschritt durch die jeweiligen Trackblöcke eine eindeutige Zuordnung zu realisieren. Nach folgender Formel werden die Crossentropie-Verlustwerte

für jeden Eingabewert  $Y_i$  und zugehörigen Zielwert (Targetvalue)  $T_i$  elementweise berechnet. Um einen Skalar pro Zeitschritt  $loss_k$  zu erhalten werden alle Verlustwerte aufsummiert und durch die Anzahl  $classes$  geteilt. Somit erhält man den durchschnittlichen loss pro Zeitschritt für alle Tracks.

$$loss_k = \frac{1}{classes} \sum_{i=1}^{classes} -(T_i \ln Y_i + (1 - T_i) \ln(1 - Y_i)) \quad (3.7)$$

Anschließend werden alle erhaltenen Skalare pro Zeitschritt aufsummiert und durch die Anzahl der Samples  $N$  eines Minibatches geteilt:

$$loss = \frac{1}{N} \sum loss_k \quad (3.8)$$

**Ergebnisanalyse** Durch das beschriebenen Vorgehen und der im Anhang aufgelisteten Trainingsoptionen konnte ein Multi Association Network (**MANTa**) trainiert werden, welches in einem Operationsschritt eine Liste an Sensor Objekten **SO** einer Liste an Tracks zuordnet. Zum Abschluss dieser Arbeit erreicht das **MANTa** eine durchschnittliche Zuordnungsgenauigkeit von 70% bezogen auf den gesamten KITTI Datensatz. Der Hauptgrund für diese Limitierung konnte durch eine Analyse der extrahierten Daten ausgemacht werden. Abb. 3.23 zeigt eine Verteilung über die Anzahl an existierenden Tracks pro Zeitschritt. So kann bspw. ausgemacht werden, dass Zeitschritte, welche einen Track beinhalten mit 29.9% und einer absoluten Anzahl an 2315 Samples fast ein drittel des gesamten verfügbaren Datensatzes ausmachen.

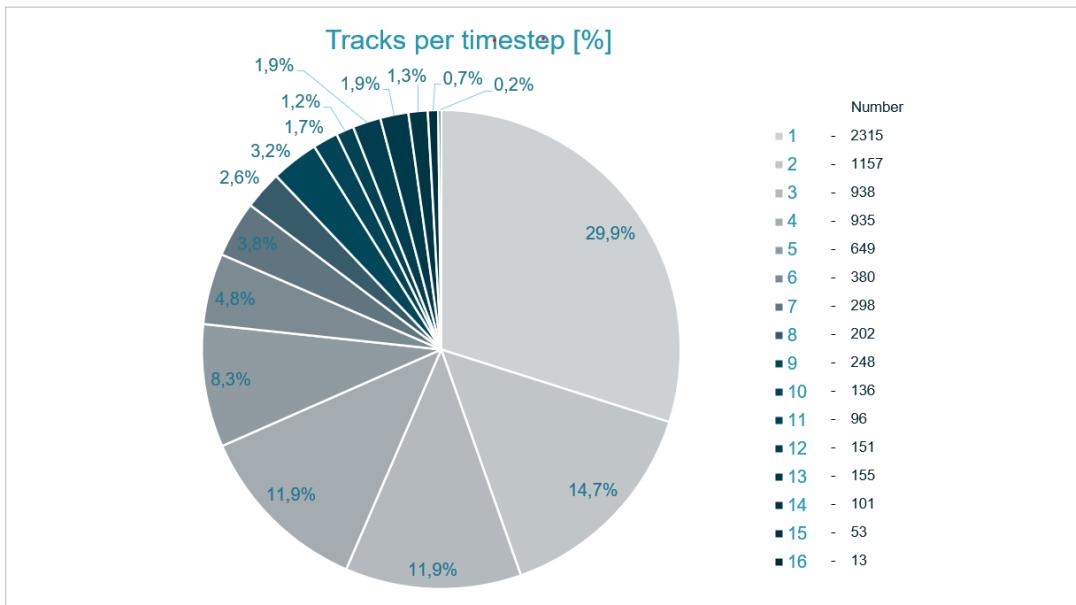


Abbildung 3.23: **MANTa**, Verteilung der verfügbaren Daten, Trackanzahl pro Sample

Zeitschritte welche ein bis sechs Tracks beinhalten, bilden zusammen 81.5% der Samples. Entsprechend wurden Tests mit einem reduzierten Datensatz durchgeführt, um die Multi-Assoziationsfähigkeit des Netzwerks unter Beweis zu stellen. So ordnet **MANTa** den Datensatz mit Zeitschritten, welche ein bis sechs Tracks beinhalten zu 95% korrekt zu. Damit bestätigt **MANTa** die Zuordnungsfähigkeit bei entsprechender Datenlage.

#### 3.5.3 Zusammenfassung

Kapitel 3.5 umfasst die Weiterentwicklung des in Kapitel 3.4 präsentierten Single Association Network (**SANT**) hin zu einem Multi Association Network (**MANTa**). Zusammengefasst konnten folgende Ziele erreicht werden:

- Es wurde ein Konzept entwickelt, um unter Anderem eine 1 zu n, m zu 1 und m zu n Sensorobjekte **SO** zu Tracks zu realisieren.
- Die verfügbaren Daten wurden analysiert und entsprechend aufbereitet, um eine konvergierendes Netzwerktraining zu realisieren. In Experimenten konnte bewiesen werden, dass der unausbalancierte Datensatz der limitierende Faktor für das Multi Association Network (**MANTa**) darstellt.
- **MANTa** erzielt auf einen ausbalancierten Datensatz mit bis zu sechs Tracks pro Zeitschritt eine Zuordnungsgenauigkeit von 95%. Somit konnte eine Netzwerk entwickelt werden, welches in einem Operationsschritt eine Liste an Sensor Objekten **SO** einer Liste an Tracks zuordnet ( $m$  zu  $n$ ).
- Das entwickelte Modell operiert auf **SO** Ebene und ermöglicht somit eine Integration in einen bestehenden **MOT** Framework, welcher nach dem Tracking by Detection (**TbD**) konzipiert wurde. Von einer Evaluierung im **MOT** Framework wird abgesehen, auf Grund der Limitierungen der Datenlage und Netzwerkperformance.

Die Ergebnisse zeigen das Optimierungspotenzial der **MANTa** Weiterentwicklung. In einem ersten Schritt muss die Datenlage entsprechend der Aufgabenkomplexität verbessert werden. Wie bereits für **SPENT** und **SANT** beschrieben, kann es anschließend von Vorteil sein weitere Experimente mit unterschiedlichen Netzwerkarchitekturen und Hyperparametern durchzuführen.

## 3.6 Evaluierung des Multi Object Tracking (MOT) Verfahrens

Die Entwicklung von Bewertungsmethoden für Multi Object Tracking (MOT) Verfahren bildet einen eigenen Forschungsbereich. Dementsprechend werden heute unterschiedliche Metriken für die Evaluierung von entwickelten MOT Verfahren herangezogen, um die Performance anhand einer Kennzahl (Score) vergleichen zu können. Für den KITTI-Datensatz werden häufig die Multiple Object Tracking Accuracy (MOTA) nach Stiefelhagen et al. [30] und die Higher Order Tracking Accuracy (HOTA) Metrik nach [17] et al. [17] verwendet. Beide Metriken benötigen die Intersection Over Union (IOU) als Basisdistanz. Neben diesen beiden Metriken gibt es jedoch auch Andere, wie die Optimal Sub-Pattern Assignment (OSPA) Metrik, welche für beliebige Mengen und Wahrscheinlichkeitsverteilungen entwickelt wurde. OSPA wird in diesem Kapitel als Metrik verwendet und im folgenden Unterkapitel eingeführt.

### 3.6.1 Optimal Sub-Pattern Assignment (OSPA) Metrik

Die OSPA Metrik kann als statistischer Abstand zwischen einer Menge von Groundtruths (GTs) mit  $m$  Elementen und einer Menge von Tracks mit der Elementanzahl  $n$  betrachtet werden. Zur Berechnung werden zunächst die vorhandenen Tracks  $n$  und  $m$  GTs mithilfe eines Global Nearest Neighbor (GNN) Algorithmus einander zugeordnet. Eine entsprechende Zuordnung muss hergestellt werden, da die Metrik eine übereinstimmende Anzahl von Objekten in beiden Mengen benötigt, um eine Distanz berechnen zu können (Optimal Subpattern). Sobald die Zuordnung berechnet ist, unterteilt die Metrik den Gesamtabstand in zwei Teilkomponenten:

**Lokalisierungskomponente (Loc)** - erfasst die Fehler, die sich aus der Genauigkeit der Zustandsschätzung ergeben. Ein hoher Wert der Lokalisierungskomponente zeigt somit an, dass die prädizierten Zustände der zugewiesenen Tracks über- bzw. unterschätzt werden.

**Kardinalitätskomponente (Card)** - erfasst die Auswirkungen von redundanten, fälschlich aufgesetzten und verpassten Tracks. Ein hoher Wert der Kardinalitätskomponente deutet somit auf das Vorhandensein von verpassten Zielen und falschen oder redundanten Tracks hin.

Die OSPA Metrik berücksichtigt nicht die zeitliche Entwicklung der Tracks, d. h. die Zuordnungen aus dem vorherigen Schritt haben keinen Einfluss auf die Metrik im aktuellen Schritt. Daher werden Effekte wie ID-Switches in der klassischen OSPA Metrik

nicht erfasst. Die klassische **OSPA** Metrik eignet sich trotz der beschriebenen Einschränkung für die Evaluierung von **MOT** Verfahren. Diese wird in den kommenden Unterkapiteln als Metrik herangezogen, um eine erste quantifizierte Bewertung über die entwickelten **ML** Modelle innerhalb eines **MOT** Programmablaufs abgeben zu können. Damit kann das Potenzial einer Weiterentwicklung quantifiziert bewertet werden.

#### 3.6.2 MOT Framework - Erläuterung der Funktionsweise

In diesem Kapitel wird der bereits eingeführte Tracking by Detection (**TbD**) Multi Object Tracking (**MOT**) Framework näher beschrieben (Abb. 3.1 und Abb. 3.24 sind identisch). Die Integration der entwickelten Netzwerke erfordert ein tieferes Verständnis über den Programmablauf der schematisch dargestellten Multi-Objekt-Verfolgung. Die Beschreibung stellt dabei keinen vollständigen Programmvorschlag dar, sondern gibt eine Übersicht über die Funktionsweise der einzelnen Programmteile bzw. Modulblöcke.

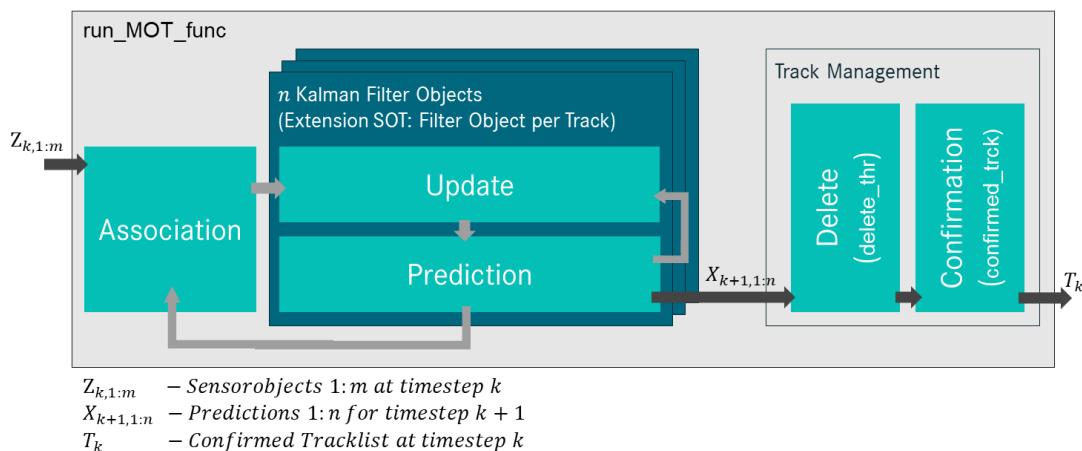


Abbildung 3.24: Schematische Darstellung der Architektur des genutzten MOT Frameworks

Der ursprüngliche Framework wurde für ein Kalman Filter (**KF**) basiertes Tracking mit Data Association (**DA**) implementiert. Für den Vergleich wird eine Global Nearest Neighbor (**GNN**) Datenassoziation verwendet, welche auf Grundlage der Mahalanobis-Distanz ein Übereinstimmungsmaß und mit Hilfe des Hungarian Algorithm (**HA**) eine Zuordnung berechnet. Die Beschreibung der vier Hauptmodule erfolgt auf der folgenden Seite.

**Association** Die Datenassoziation wird mit einem Global Nearest Neighbor ([GNN](#)) Verfahren durchgeführt. Dafür wird im ersten Schritt die Mahalanobis-Distanz als Maß für die Übereinstimmung zwischen den vorhergesagten  $x_{k+1,n}$  und gemessenen Zuständen  $z_{k,m}$  berechnet. Sie quantifiziert somit, wie gut eine Detektion mit einem bereits verfolgten Sensor Objekt ([SO](#)) übereinstimmt. Es wird eine Distanzmatrix über alle Tracks  $n$  und alle verfügbaren Messungen  $m$  pro Zeitschritt gebildet, welche die Kosten der Zuordnungen zwischen Tracks und Messungen repräsentiert. Des weiteren wird der Hungarian Algorithm ([HA](#)) verwendet, um die optimale Zuordnung mit minimalen Gesamtkosten zu finden. Je höher die Kosten, desto weniger wahrscheinlich ist die Zuordnung. Der [HA](#) gibt somit eine Assoziationsmatrix aus, welche die Zuordnung Messungen zu Tracks pro Zeitschritt darstellt.

**Initialisierung und Update** Die Initialisierung eines Kalman Filter ([KF](#)) Objekts für neu getrackte [SO](#) beinhaltet die Zuweisung des Anfangszustands. Bereits verfolgte [KF](#) Objekte werden mit den entsprechenden Messungen aktualisiert.

**Prediction** Alle Zustände der verfolgten Objekte werden unter Verwendung eines Dynamikmodells vorhergesagt. Das Dynamikmodell beschreibt, wie sich der Zustand eines Objekts im Laufe der Zeit ändert. Für diese Untersuchung wurde ein konstantes Geschwindigkeitsmodell verwendet. Der Prozess wird für jeden Zeitschritt iteriert, wobei die [KF](#) Zustände aktualisiert und vorhergesagt werden und die Assoziationsmatrix erneut berechnet wird.

**Track-Management** Die Überwachung und Verwaltung der Lebensdauer der Tracks erfolgt im Programmteil Track-Management. Für jeden Track wird bei der Initialisierung eine Existenzwahrscheinlichkeit zugewiesen. Die Berechnung der Existenzwahrscheinlichkeit erfolgt nach Aeberhard [1]. Wird einem Track in einem Folgezeitschritt ein [SO](#) zugeordnet, wird ebenfalls die Existenzwahrscheinlichkeit aktualisiert. Erhält ein Track kein [SO](#) sinkt diese entsprechend der Berechnung mit im Modell hinterlegten survival/death Parametern. Tracks mit einer Existenzwahrscheinlichkeit unter einem Schwellenwert werden innerhalb des Track-Managements gelöscht, während Tracks mit einem Wert über einem Schwellenwert als bestätigte Tracks  $T_k$ , weiter verfolgt werden. Innerhalb einer [ADAS](#) Softwarearchitektur werden die bestätigten Tracks des Trackers von den [ADAS](#) Features verwendet, um beispielsweise Warnungen bzw. Fahrzeugreaktionen auszulösen. In diesem Kapitel werden daher die bestätigten Tracks für die Bewertung des Trackers genutzt. Die Bewertung erfolgt auf Basis der [OSPA](#) Metrik.

### 3.6.3 Vergleich der Vorhersagemodelle im MOT Framework

Die in diesem Kapitel dokumentierte Auswertung vergleicht die Fähigkeit der Prädiktion von Multiobjektzuständen im vorgestellten Multi Object Tracking (**MOT**) Framework auf Basis der **OSPA** Metrik. Dafür wurde das Single Prediction Network (**SPENT**) wie in Abbildung 3.25 dargestellt in den bestehenden Programmablauf integriert. Aufbauend auf die beschriebenen Funktionsweise des **MOT** Frameworks, ist die Implementierung des **SPENT** innerhalb dieses Frameworks erläutert. 3.25 stellt schematisch dar, dass **SPENT** für die Evaluierung in den bestehenden Framework ergänzend zu der bestehenden Funktionalität implementiert wurde.

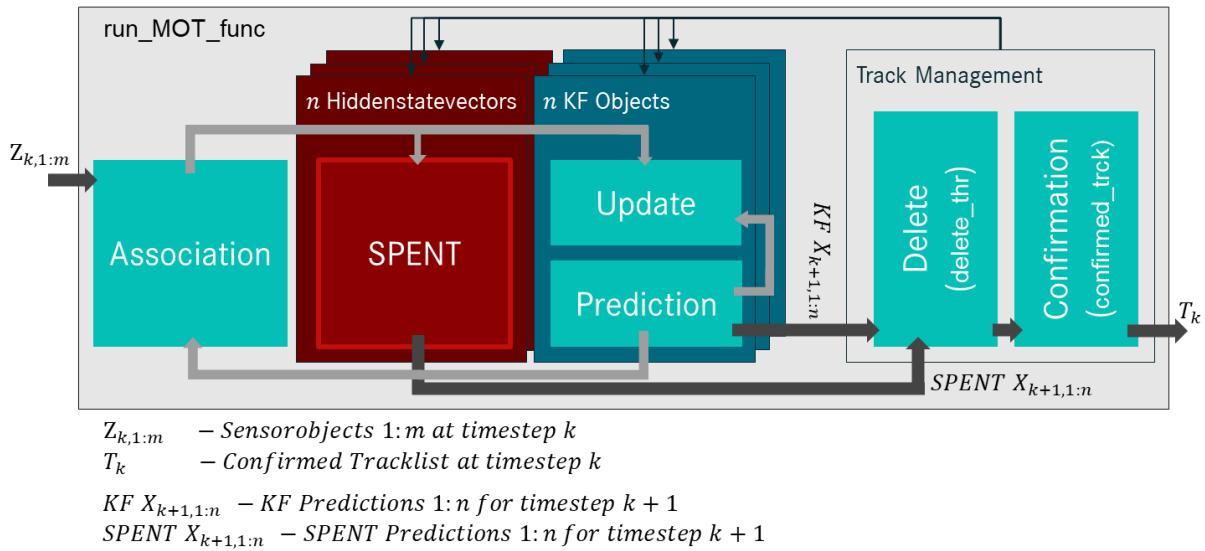


Abbildung 3.25: Schematische Darstellung, MOT Framework, KF und SPENT Integration

Die Anzahl der Verfolgungen  $n$  stellt die Anzahl der Kalman Filter (**KF**) Objekte und **SPENT** Hiddenstate-Vektoren dar. Abb. 3.25 zeigt zudem, dass **SPENT** über keinen externen Updateprozess verfügt. Die Hiddenstates des jeweiligen Tracks  $n$  werden pro Zeitschritt mit Hilfe der erhaltenen Messungen angepasst. Auf Basis der Modellentwicklung und -evaluierung in Kapitel 3.3.3 wurde bereits bewiesen, dass dieser interne Updateprozess zu genauen Zustandsvorhersagen führen kann. Die Behandlung der unterschiedlichen Fälle wurde während der Entwicklung empirisch ermittelt. Folgender Programmablauf ist für die Evaluierung implementiert:

- **Initialisierung von neuen Sensorobjekten SO** – die Zustandswerte einer neuen Entdeckungen  $z_{k,1}$  werden genutzt um einen Hiddenstatevektor  $H_1$  zu Initialisieren. Für die Assoziation im folgenden Zeitschritt  $k = 2$  wird der initiale Zustandswert  $z_{k,1}$  verwendet. Damit wurde sichergestellt, dass **SPENT** nach dem zweiten internen Update der Hiddenstates eine Prädiktion ausgibt.

- **Assoziierte Sensorobjekte SO** - wird einem existierendem Track  $T_{k,3}$  ein SO  $z_{k,m}$  zugeordnet wird der entsprechende Hiddenstatevektor  $H_3$  durch die Zustandswerte des SO  $z_{k,m}$  aktualisiert und eine Prädiktion  $x_{k,3}$  ausgegeben. In der schematischen Darstellung 3.25 ist zu erkennen, dass die Zustandsvorhersagen des KF für die Bildung der Distanzmatrix als Eingabewerte für die GNN Datenassoziation genutzt werden. Das Verfahren wurde so implementiert, dass auch die Vorhersagen von SPENT für diesen Programmteil übernommen werden können.
- **Tracks ohne assoziiertes Sensor Objekt (SO)** - SPENT erhält in diesem Fall die Vorhersage des vorherigen Zeitschritts  $x_{k-1,m}$  als neue Messung. (Track-Management bestimmt die Existenzdauer).

Der erweiterte Framework gibt wie beschrieben die Tracks des Kalman Filters (KFs) und von SPENT aus, welche für die Auswertung genutzt werden. Der Vergleich verwendet die 21 verfügbaren Sequenzen des KITTI Datensatzes mit den Objektklassen Car und Van. Die Simulation beinhaltet somit den Gesamtdatensatz der Netzwerkentwicklung. Die Tracklets wurden für den Vergleich mit 5% Rauschen als Sensorobjekte SO  $z_{k,m}$  simuliert. Die Zustandswerte variieren entsprechend um 5% um den GT Wert. Das Ergebnis der Untersuchung ist auf der folgenden Seite zusammengefasst.

### 3 Ergebnisse

Die implementierte Simulationsumgebung ermöglicht einen Vergleich der beiden Prädiktionsverfahren pro Sequenz des Datensatzes. Um einen durchschnittlichen Wert pro Sequenz bilden zu können, verlangt die **OSPA** Metrik einen Cutoff-Wert. Die Untersuchung wurde mit einem Cutoff-Wert von  $c = 2$  durchgeführt. Alle Objekte innerhalb der "DontCareRegionen oder außerhalb des vom KITTI-Datensatz definierten Field of View (**FoV**) werden von der Auswertung ausgeschlossen. Diagramm 3.26 zeigt die durchschnittlichen **OSPA** Werte pro Sequenz. Zu beachten ist, dass Sequenz 17 keinerlei Fahrzeugdaten enthält und daher keine Auswertebalken aufgetragen sind. Es wird deutlich, dass die Ergebnisse der beiden Prädiktionsmethoden sich nur minimal in der **OSPA Loc** Komponente unterscheiden. Die Ergebnisse zeigen weiter, dass **SPENT** erfolgreich in den bestehenden **MOT** Framework integriert wurde. Die Implementierung weist jedoch noch Optimierungspotenzial auf, da die Erwartungshaltung auf Grund der Ergebnisse der Einzel-Modul-Verifizierung aus Kapitel 3.3.3 in dieser Analyse nicht bestätigt werden kann.

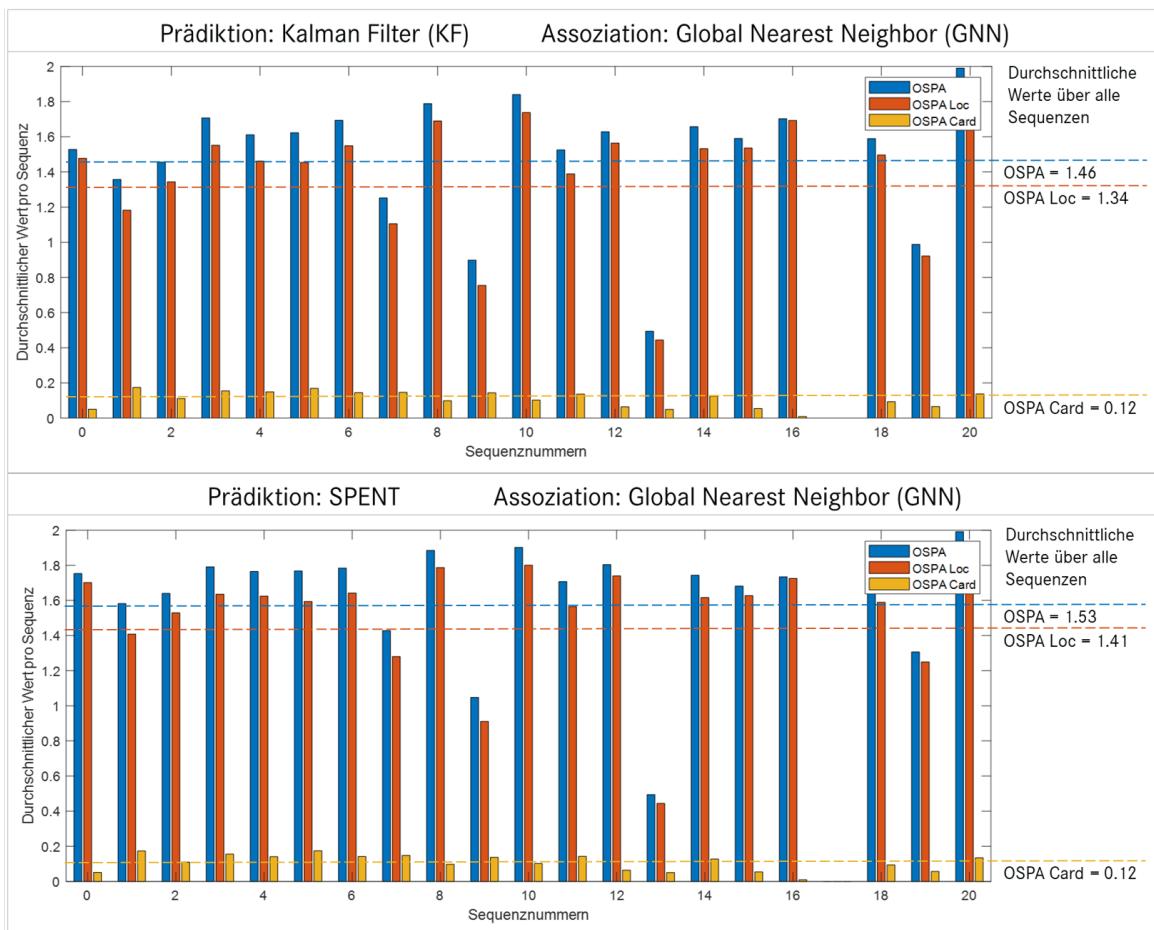


Abbildung 3.26: Vergleichsdiagramm **OSPA**, KITTI Datensatz, **KF** und **SPENT**, **GNN**

In der Sequenzanalyse (hier Sequenz 6) wird deutlich, dass beide Vorhersagemodelle auf Basis des selben Assoziations- und Trackmanagementmodul arbeiten, jedoch unterschiedliche Zustandsvorhersagen treffen. Abbildung 3.27 zeigt den GT Verlauf der vorhandenen Tracks, sowie die Vorhersagen von KF und SPENT. Kreise stellen die Initialisierung der Tracks (Track Birth) dar, Dreiecke stehen entsprechend für den Verlust eines Tracks. Dabei ist zu beachten, dass die beiden Vorhersagemodelle um 5% verrauschte Messdaten als Eingabe erhalten haben. Beide Implementierungen der Vorhersagemodelle weißen das Potenzial der Optimierung auf.

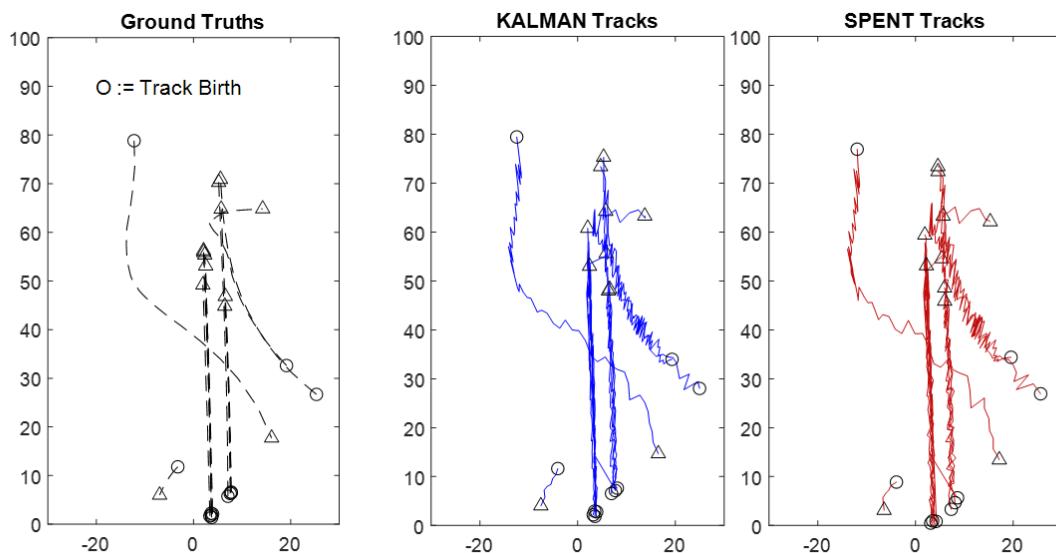


Abbildung 3.27: Vergleichsdiagramm, KF und SPENT Tracks, Sequenz 6

Die Analyse zeigt, dass das entwickelte Single Prediction Network (SPENT) eine Machine Learning (ML) Methode darstellt, welche datenbasiert die Zustandsvorhersage von Personen Kraft Wagen (PKW) erlernt hat und in einem TbD Framework erfolgreich integriert werden konnte.

Wie Schindler et al. [29] beschreibt, kann im Rahmen dieser Untersuchung bestätigt werden, dass einer der Vorteile des Einsatzes von RNN für die Zustandsvorhersage darin liegt, dass keinerlei Dynamikmodelle implementiert werden müssen und daher lineare (vgl. Kalman-Filter), nicht lineare (vgl. Partikelfilter), und Abhängigkeiten höherer Ordnung erfasst werden können. Natürlich in Abhängigkeit von verfügbaren Trainingsdaten. In Kapitel 4.1 werden noch einmal die Ergebnisse der Arbeit zusammengefasst, sowie Chancen und Risiken für die Entwicklung und den Einsatz von Neural Network (NN) im ADAS Kontext dargestellt.

### 3.6.4 Vergleich Assoziationsmodelle im MOT Framework

Dieser Abschnitt der Evaluierung beschäftigt sich mit dem Vergleich des entwickelten Single Association Network (**SANT**) zu einem klassischen **GNN** Assoziationsverfahren. Die Optimal Sub-Pattern Assignment (**OSPA**) Metrik wird weiterhin als Tracking-Metrik verwendet. Wie in Abb. 3.28 schematisch dargestellt, ersetzt **SANT** das Assoziationsmodul. Für die Assoziation benötigt **SANT** das zuzuordnende Sensorobjekt **SO**  $z_{k,m}$  sowie die existierenden Tracks  $T_k$  im jeweiligen Zeitschritt (siehe Inputstruktur, Kapitel 3.4.1). **SANT** ersetzt somit die beiden Algorithmen der **GNN** Assoziation, die Berechnung eines Übereinstimmungsmaßes und den Hungarian Algorithm (**HA**).

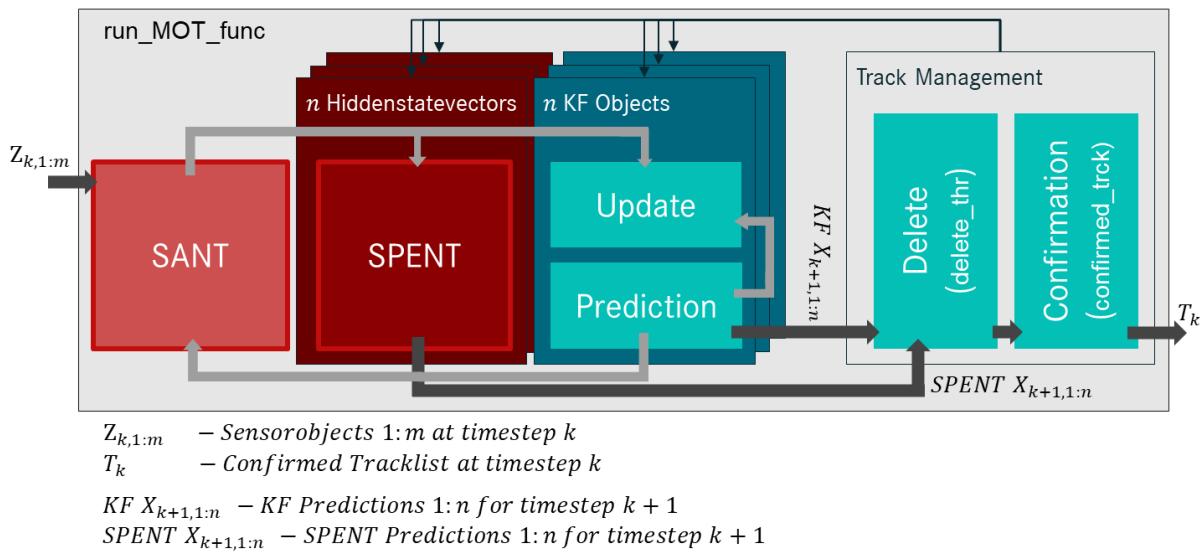


Abbildung 3.28: Schematische Darstellung, MOT Framework mit **SANT** Integration

Das Single Association Network (**SANT**) ist für diese Evaluierung wie folgt in den Programmablauf des **MOT** Frameworks integriert:

- **Initialisierung Assoziationsmatrix** - pro Zeitschritt  $k$  wird entsprechend der Schnittstellen des **MOT** Frameworks eine Assoziationsmatrix mit der Größe  $n * m$  Null initialisiert. Wobei  $n$  die Anzahl der existierenden Tracks und  $m$  die Anzahl der Sensor Objekte **SO** in diesem Zeitschritt darstellt.
- **SANT Inputstruktur** - sobald  $m$  ungleich Null ist, wird nach der Initialisierung der Assoziationsmatrix, über die Anzahl der **SO** iteriert. Pro Iterationsschritt wird die bereits in Kapitel 3.13 vorgestellte Inputstruktur für **SANT** erzeugt. Der Framework ist für die Evaluierung so angepasst, dass je nach Parametrierung die **SPENT** oder **KF** Zustandsvorhersagen als Tracks für die Inputstruktur übernommen werden können.

### 3 Ergebnisse

- **Einzel-Zuordnung** - das Verfahren wurde so implementiert, dass **SANT** pro **SO** eine Zuordnungswert ausgibt. Zusätzlich wird der Wahrscheinlichkeitswert des softmaxlayers ermittelt.
- **Integration ohne Distanzmatrix** - der Wahrscheinlichkeitswert des Zuordnungswerts wird entsprechend der Schnittstellen des Frameworks dazu verwendet eine Zuordnung in Abhängigkeit eines Thresholds durchzuführen. Im klassischen **GNN** Verfahren ist an dieser Stelle ein Modelparameter als Threshold hinterlegt, welcher sich auf das berechnete Abstandsmaß bezieht. Wird das **SO** nicht assoziiert, folgt die Initialisierung eines neuen Tracks.

Abbildung 3.29 zeigt die durchschnittlichen **OSPA** Werte pro Sequenz des gesamten KITTI Datensatzes. Die Assoziation wurde in beiden Simulationen mit **SANT** durchgeführt. Entsprechend variiert der Einsatz des Vorhersagemodells. Die Ergebnisse zeigen weiter, dass **SANT** erfolgreich in den bestehenden **MOT** Framework integriert wurde. Der **OSPA Card** Wert ist in beiden Simulationen bei ca. 0.24.

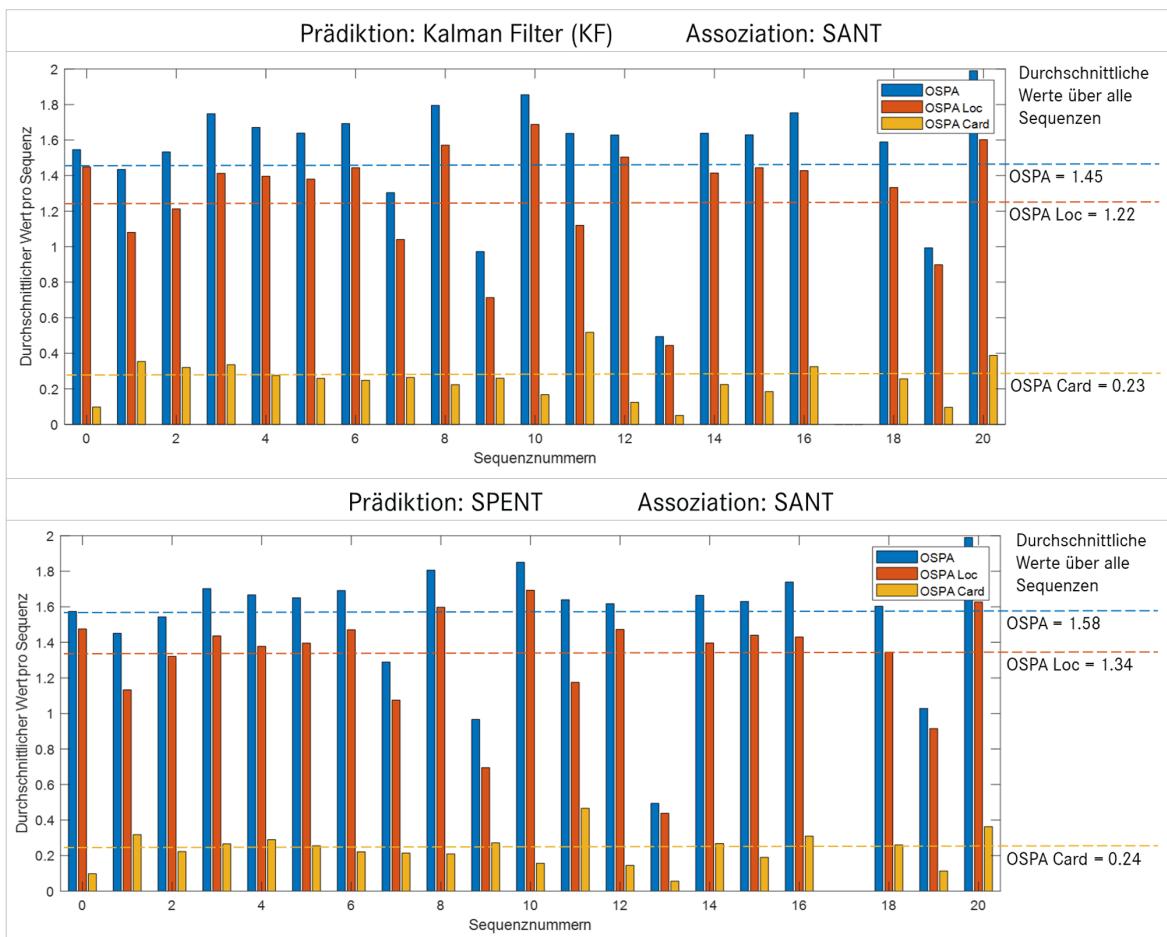


Abbildung 3.29: Vergleichsdiagramm **OSPA**, KITTI Datensatz, **KF** und **SPENT**, **SANT**

### 3 Ergebnisse

---

Abschließend sind die durchschnittlichen **OSPA** Werte über alle Sequenzen pro Kombinationsmöglichkeit tabellarisch zusammengefasst. Tabelle 3.4 verdeutlicht, dass die beiden entwickelten Netzwerke innerhalb des **MOT** noch Schwächen zeigen. Bezogen auf die **OSPA** Card Ergebnisse lässt sich ableiten, dass die Assoziationsergebnisse des klassischen **GNN** in diesem Vergleich besser ausfallen.

Tabelle 3.4: Ergebnistabelle, **OSPA**, KITTI Datensatz, **MOT** Framework

Prediction + Association	OSPA	OSPA Loc	OSPA Card
KF + GNN	1.46	1.34	0.12
SPENT + GNN	1.53	1.41	0.12
KF + SANT	1.45	1.22	0.23
SPENT + SANT	1.58	1.34	0.24

# 4 Fazit

Das vorliegende Kapitel bildet den Schlussstein dieser wissenschaftlichen Arbeit und stellt einen umfassenden Überblick über die erreichten Ergebnisse sowie einen Ausblick auf mögliche zukünftige Entwicklungen dar.

Im Verlauf der vorangegangenen Kapitel wurden unterschiedliche Aspekte und Fragestellungen behandelt, wobei die vorliegende Arbeit darauf abzielte, Machine Learning (**ML**) basierte Ansätze für Multi Object Tracking (**MOT**) mit Sensor Objekt (**SO**) Schnittstelle zu entwickeln. Die nachfolgende Zusammenfassung bietet eine kompakte Darstellung der Hauptergebnisse und betont dabei die signifikanten Erkenntnisse, die im Rahmen dieser Untersuchung gewonnen wurden.

Darauf aufbauend richtet der Ausblick den Blick nach vorne und gibt einen Ausdruck der Potenziale und Herausforderungen, die sich aus den erzielten Erkenntnissen ergeben. Dies ist entscheidend, um den Beitrag der vorliegenden Arbeit zu kontextualisieren und ihre Relevanz für die zukünftige Forschung oder Praxis zu verdeutlichen.

## 4.1 Zusammenfassung

Die Arbeit hat sich intensiv mit der Entwicklung und Implementierung von Machine Learning (**ML**) basierten Ansätzen für Multi Object Tracking (**MOT**) mit Sensor Objekt (**SO**) Schnittstelle beschäftigt. Im Fokus standen dabei die Entwicklung des Single Prediction Network (**SPENT**), Single Association Network (**SANT**) und Multi Association Network (**MANTa**). Die Netzwerke stellen jeweils einen Lösungsansatz dar, um eine Teilaufgabe des vorgegebenen Tracking by Detection (**TbD**) **MOT** Frameworks datenbasiert zu lösen.

Die Lösung ist modular und wurde in Matlab geschrieben. Verschiedene Netzwerkarchitekturen wurden getestet, wobei die besten Ergebnisse Architekturen erzielten, die **LSTM** und **BILSTM** Schichten für die Verarbeitung der Zeitreiheninformationen verwenden. Diese konnten in einen bestehenden Framework integriert werden und ersetzten somit klassische heuristische Algorithmen innerhalb des **MOT** Verfahrens.

Dabei ersetzt Single Prediction Network (**SPENT**) die Zustandsvorhersagen des Kalman Filters **KF**. Das trainierte Netzwerk schätzt die Vorhersagen pro Zeitschritt ohne

den Bedarf eines Dynamikmodells. Die Implementierung erlaubt dem rekurrenten Netzwerk ein Update der internen Hiddenstates pro Zeitschritt, womit eine genaue Zustandsvorhersage ohne eine externe Korrektur erreicht wurde. Das Modell eignet sich für den Einsatz in real time Anwendungen und stellt einen alternativen Ansatz zu klassischen Vorhersageverfahren dar. Die Netzwerkverifizierung zeigt einen RMSE von 0.026 gemittelt auf den Trainings-, Validations-, und Testdatensatz. Bezogen auf die Vorhersagen der Positionen eines Objekts in X Koordinate, entspricht das einer durchschnittlichen Abweichung von 0.42m. Bezogen auf die Position in Y Richtung relativ zum Egofahrzeug liegt die durchschnittliche Abweichung der durchgeföhrten Verifizierung bei 0.23m.

Für eine weitere Hauptaufgabe des Tracking by Detection (TbD) MOT Verfahrens, der Datenassoziation, wurde das Single Association Network (SANT) entwickelt, um einen Ersatz für das klassische Global Nearest Neighbor (GNN) Verfahren darzustellen. Das trainierte Netzwerk assoziiert dabei jedes Sensor Objekt (SO) einzeln zu einem bestehenden Track Set. Damit kann SANT die Algorithmen der Berechnung einer Distanzmetrik und Zuordnungsverfahren wie den Hungarian Algorithm (HA) durch die erlernte, Trainingsdaten basierte Zuordnungslogik ersetzen. Bezogen auf einen definierten Validationsdatensatz mit ca. 2700 Samples, erzielt das Single Association Network (SANT) eine Genauigkeit von 95%.

Das Multi Association Network (MANTa) stellt eine Weiterentwicklung von SANT dar und adressiert dabei die Einschränkung der Einzelzuordnung. Es konnte eine Netzwerkerweiterung implementiert werden, welche eine Menge an Sensor Objekten SO  $m$  zu einer Menge Tracks  $n$  zuordnet. Dabei wurde die Datenlage genauer untersucht und als Limitierung für die Netzwerkperformance ausgemacht. Die durchgeföhrte Verifizierung zeigt, dass MANTa eine Zuordnungsgenauigkeit von 95% erzielt, bezogen auf die sechs am häufigsten vorkommenden Assoziationsmengen.

## 4.2 Ausblick

Die in dieser Masterthesis durchgeföhrten Untersuchungen haben wichtige Erkenntnisse im Bereich des Machine Learning (ML) basierten Multi Object Tracking (MOT) auf Sensor Objekt (SO) Ebene geliefert. Trotz der erzielten Fortschritte gibt es jedoch noch zahlreiche offene Fragen und Potenziale für zukünftige Forschungsarbeiten.

**Vertiefung der aktuellen Forschung** - Als erster Schritt sollten die Netzwerke um einige Featureeingabe erweitert werden. Informationen des Egofahrzeugs wie die Beschleunigungs- und Geschwindigkeitswerte sollten dem Netzwerk als zusätzliche

Informationsbasis dienen. Es ist mit einer Performancesteigerung auf Grund der zusätzlichen Information zu rechnen. Ebenfalls können die Entwicklungsergebnisse als Basis dazu dienen, Netzwerke auf weitere Verkehrsteilnehmer, wie beispielsweise Fußgänger zu trainieren. Liefert das Sensor Objekt (**SO**) zusätzlich die Information über seine Klasse, könnte ein Netzwerk unterschiedliche Hiddenstates Initialisieren und so klassenspezifische Zustandsvorhersagen treffen. Des weiteren könnten die Ergebnisse dieser Arbeit durch eine Erweiterung der Datengrundlage weiter validiert werden. Im ersten Schritt könnten zusätzliche frei verfügbare Datensätze herangezogen werden wie z.B. der Waymo Open Dataset<sup>1</sup> oder nuscenes Open Dataset<sup>2</sup>.

Um die Untersuchungen für eine real time Anwendungen im Truck vorzubereiten, sollten zudem entsprechende Daten mit einem realen Sensorsetup bezogen werden. Die entwickelten Netzwerke könnten mit einer Vielzahl an Daten trainiert werden, wobei als Validationdataset ein anwendungsnaher Truck Datensatz verwendet werden sollte, um den Einfluss auf die Zielanwendung bewerten zu können. Eine weitere Möglichkeit der Datengenerierung könnte zudem betrachtet werden. So könnten Ergebnisse von Simulationen mit in einen Trainingsdatensatz einfließen und so eine deutliche Erhöhung der Datenmengen ermöglichen. Ein weiterer nächster Entwicklungsschritt könnte die Erweiterung auf den Bereich der Multi-Sensor **MOT** Verfahren sein. Die in dieser Arbeit entwickelten Modelle könnten in einem ersten Schritt auf die unterschiedlichen sensorspezifischen Daten getestet werden, um entsprechende Entwicklungsmaßnahmen abzuleiten.

**Erweiterung des Untersuchungsbereichs** - Während diese Arbeit sich auf die Netzwerkentwicklung mit **RNN** Layern konzentrierte, könnte zukünftige Forschung den Fokus erweitern. Wie bereits im Kapitel der theoretischen Grundlagen erarbeitet, zeigen aktuelle Forschungen einen großen Erfolg mit Attention Modellen [41], [38], [40]. Erste Experimente, welche innerhalb dieser Arbeit durchgeführt wurden, bestätigen die Einsatzmöglichkeit des Self Attention Mechanismus nach Kaiser et al. [15] für ein Tracking auf **SO** Ebene. Ein weiteres gewinnbringendes Themengebiet stellt die Entwicklung der Bayesian Neural Network (**BNN**) dar. Im erweiterten Ausblick wird das Ziel der **BNN**, die Unsicherheitsquantifizierung näher vorgestellt.

Abschließend lässt sich sagen, dass das Machine Learning (**ML**) basierte Multi Object Tracking (**MOT**) zwei dynamisches Forschungsfeld mit vielen unerforschten Möglichkeiten vereint. Die in dieser Masterthesis präsentierten Ergebnisse bieten dabei einen soliden Ausgangspunkt. Es wird erwartet, dass zukünftige Arbeiten in diesem Bereich weiterhin wertvolle Beiträge zur Wissenschaft und Praxis leisten werden.

---

<sup>2</sup> <https://waymo.com/open/>

<sup>2</sup> <https://www.nuscenes.org/>

## Erweiterter Ausblick: Bayesian Neural Network (BNN) - Unsicherheitsquantifizierung

Wie bereits in 2.1 eingeführt, nutzen klassische Filter- und Trackingverfahren wie das Kalman Filter (KF) die Varianz und den Mittelwert der Normalverteilung, um unter Anderem das Messrauschen quantifiziert für die Berechnung zu berücksichtigen.

Die Ergebnisse dieses Kapitels zeigen, dass präzise Zustandsvorhersagen mit SPENT getroffen werden können, jedoch suggerieren alle Vorhersagen bisher eine hohe (absolute) Sicherheit. Im Rahmen dieser Arbeit wurde zudem eine vielversprechende Methode untersucht, welche als Ausblick für eine Weiterentwicklung des Single Prediction Network (SPENT) betrachtet wurde. Die untersuchte Methode stellt einen ML basierten Ansatz dar, welche es dem Netzwerk ermöglicht die Unsicherheit über eine Prädiktion auszugeben.

Das Gebiet der Modellierung und bzw. Berücksichtigung von Datenunsicherheiten in NN bildet einen eigenen Forschungsbereich, welcher bereits mehrere vorgeschlagenen Methoden hervorgebracht hat. Wierstra et al. [36] stellt das Verfahren Bayes by Backprop vor. Es wird ein Backpropagation-kompatibler Algorithmus gezeigt, welcher das Ziel verfolgt Bayes'sche Inferenz in neuronalen Netzwerken NN zu ermöglichen. Der Ansatz zielt darauf Unsicherheiten in den Gewichten und damit auch in Vorhersagen zu quantifizieren.

Im Gegensatz zu den bisher genutzten deterministischen neuronalen Netzwerkschichten (RNNs, FCs), die einen einzelnen Satz von Gewichten verwenden, die durch Backpropagation und Gradientenabstieg während des Trainings erlernt werden, führt Bayes by Backprop Unsicherheit in die Gewichte ein, indem es Wahrscheinlichkeitsverteilungen für die Gewichte definiert. Dadurch können verschiedene Realisierungen (Samples) der Gewichte entstehen. Das Verfahren verwendet Bayes'sche Inferenz, um die Posterior-Verteilungen der Gewichte zu schätzen. Es wird davon ausgegangen, dass die Posterior-Verteilungen der Gewichte eine bestimmte parametrische Form besitzt (z.B. Normalverteilung). Um die Posterior-Verteilungen zu schätzen, wird Variational Inference angewendet. Dabei wird ein variationaler Ansatz verwendet, um eine approximative Posterior-Verteilung zu definieren, die der wahren Posterior-Verteilung ähnlich ist. Das Ziel ist es, die Parameter dieser approximativen Verteilung zu optimieren, um sie der wahren Posterior-Verteilung möglichst nahe zu bringen. [22] stellt die Variational Inference Methode im Detail vor.

Bayes by Backprop nutzt diese Methode, um während des Trainings die Gewichtsunsicherheit durch Backpropagation und stochastischen Gradientenabstieg zu optimieren. Dabei werden Samples aus der approximativen Posterior-Verteilung der Gewichte

gezogen, um die Gewichte für jeden Mini-Batch zu aktualisieren. Dies ermöglicht die Modellierung der Unsicherheit in den Gewichten und somit eine Aussage über die Unsicherheit in den Vorhersagen.

**Vorgeschlagener Entwicklungsansatz** Eine mögliche Implementierung eines Bayesian Neural Network (**BNN**) für die Vorhersage von Zeitreihen kann beispielsweise durch folgende Schritte realisiert werden:

- **Modellarchitektur festlegen** - Das **BNN** kann eine ähnliche Architektur wie ein herkömmliches **NN** haben. Der entscheidende Unterschied liegt jedoch in der Verwendung von probabilistischen Schichten, die unsicherheitsbehaftete Gewichtungen ermöglichen.
- **Prior- und Posterior-Verteilungen festlegen** - Für jedes Gewicht in den Schichten des **BNNs** müssen Prior- und Posterior-Verteilungen festgelegt werden. Die Prior-Verteilungen repräsentieren das Vorwissen über die Gewichte, bevor das Modell auf die Daten angepasst wird. Typischerweise werden hierbei bestimmte Wahrscheinlichkeitsverteilungen, wie die Normalverteilung, verwendet. Die Posterior-Verteilungen hingegen repräsentieren das Wissen über die Gewichte nach der Anpassung an die Daten. Diese werden während des Trainings des **BNNs** entsprechend aktualisiert.
- **Bayes'sches Lernen** - Das Training beinhaltet das Schätzen der Posterior-Verteilungen der Gewichte, basierend auf den Trainingsdaten und unter Verwendung von Bayes'scher Statistik. Dies kann durch Techniken wie den Bayes'schen Gradientenabstieg erreicht werden, McAuliffe et al. [22].

Nach dem Training könnte das mit einer **BNN** Schicht erweiterte **SPENT** verwendet werden, um wie bisher Vorhersagen für Zustandswerte zu machen, Mathworks.inc [20]. Zusätzlich können Unsicherheitsmaße wie das Konfidenzintervall berechnet werden, da die Gewichtungen durch Wahrscheinlichkeitsverteilungen dargestellt werden. Dies ermöglicht es, die Vorhersagen mit der entsprechenden Unsicherheit zu quantifizieren. Die Vorstellung des Bayesian Neural Networks (**BNNs**) zeigt somit eine leistungsstarke Methode zur Vorhersage von Zeitreihen mit der Möglichkeit, Unsicherheiten zu quantifizieren und fundierte Entscheidungen zu treffen.

# Literatur

- [1] Michael Aeberhard. *Object-Level Fusion for Surround Environment Perception in Automated Driving Applications*. 2017.
- [2] Alexander Amini. *Introduction to Deep Learning*: MIT 6.S191. 2023.
- [3] Ava Soleimany. *Deep Sequence Modeling*: MIT 6.S191. 2022.
- [4] Diederik Kingma; Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014.
- [5] Ba-Tuong Vo. *code set for research use: Multi-Sensor Multi-Target Tracking*. 2013.
- [6] Reiner Marchthaler; Sebastian Dingler. *Kalman-Filter: Einführung in die Zustands-schätzung und ihre Anwendung für eingebettete Systeme*. 2017.
- [7] Xiaolong Wang; Allan Jabri; Alexei Efros. *Learning Correspondence from the Cycle-Consistency of Time*. 2019.
- [8] Subhash Challa; Mark Morelande; Robin Evans. *Fundamentals of Object Tracking*. 2011.
- [9] Ian Goodfellow. *Deep Learning - Grundlagen, aktuelle Verfahren und Algorithmen, neue Forschungsansätze*. 2018.
- [10] Nitish Srivastava; Geoffrey Hinton. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. 2014.
- [11] Schmidhuber Hochreiter. *Long short-term memory*. 1997.
- [12] Yaakov Bar-Shalom; Fred Daum; Jim Huang. *The probabilistic data association filter*. 2009.
- [13] Johannes Fitz. *Datenassoziation für Multi-Objekt-Verfolgung mittels Deep Learning*. 2020.
- [14] Jonas Knupp. *Einführung in Deep Learning – LSTM & CNN: Proseminar Data Mining*. 2017.
- [15] Ashish Vaswani; Noam Shazeer; Niki Parmar; Jakob Uszkoreit; Llion Jones; Aidan Gomez; Łukasz Kaiser. *Attention Is All You Need*. 2017.
- [16] Jenny Seidenschwarz; Guillem Brasó; Victor Serrano; Ismail Elezi; Laura Leal-Taixé. *Simple Cues Lead to a Strong Multi-Object Tracker*. 2022.
- [17] Jonathon Luiten; Patrick Dendorfer; Philip Torr; Andreas Geiger; Laura Leal-Taixé. *HOTA: A Higher Order Metric for Evaluating Multi-object Tracking*. 2021.

- [18] Laura Leal-Taixé. *Shifting Paradigms in Multi-Object Tracking*. 2021.
- [19] Patrick Dendorfer; Daniel Cremers; Ian Reid; Stefan Roth; Laura Leal-Taixé. *MOT-Challenge: A Benchmark for Single-Camera Multiple Target Tracking*. 2020.
- [20] Mathworks.inc. *long-short-term-memory-networks: LSTM Neural Network Architecture*. 2023.
- [21] Mathworks.inc. *Time Series Forecasting Using Deep Learning*. 2023.
- [22] David Blei; Alp Kucukelbir; Jon McAuliffe. *Variational Inference: A Review for Statisticians*. 2017.
- [23] Huajun Liu; Hui Zhang; Christoph Mertz. *DeepDA: LSTM-based Deep Data Association Network for Multi-Targets Tracking in Clutter*. 2019.
- [24] Wenhan Luo; Junliang Xing; Anton Milan. *Multiple object tracking: A literature review*. 2021.
- [25] Henry Bedinger Mitchell. *Multi-Sensor Data Fusion: An Introduction*. 2007.
- [26] Yesul Park; Minh Dang; Sujin Lee; Dongil Han; Hyeonjoon Moon. *Multiple Object Tracking in Deep Learning Approaches: A Survey*. 2021.
- [27] Johannes Pallauf. „Objektsensitive Verfolgung und Klassifikation von Fußgängern mit verteilten Multi-Sensor-Trägern“. Diss. 2016.
- [28] David Held; Sebastian Thrun; Silvio Savarese. *Learning to Track at 100 FPS with Deep Regression Networks*. 6.04.2016.
- [29] Anton Milan; Seyed Rezatofighi; Anthony Dick; Ian Reid; Konrad Schindler. *Online Multi-Target Tracking Using Recurrent Neural Networks*. 2016.
- [30] Keni Bernardin; Rainer Stiefelhagen. *Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics*. 2008.
- [31] Alex Bewley; Zongyuan Ge; Lionel Ott; Fabio Ramos; Ben Upcroft. *Simple Online and Realtime Tracking*. 2016.
- [32] Andreas Geiger; Philip Lenz; Christoph Stiller; Raquel Urtasun.
- [33] Guanghan Ning; Zhi Zhang; Chen Huang; Zhihai He; Xiaobo Ren; Haohong Wang. *Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking*. 2016.
- [34] Zhongdao Wang; Liang Zheng; Yixuan Liu; Yali Li; Shengjin Wang. *Towards Real-Time Multi-Object Tracking*. 2019.
- [35] Kurt Hornik; Maxwell Stinchcombe; Halbert White. *Multilayer Feedforward Networks are Universal Approximators*. 1989.
- [36] Charles Blundell; Julien Cornebise; Koray Kavukcuoglu; Daan Wierstra. *Weight Uncertainty in Neural Networks*. 2015.

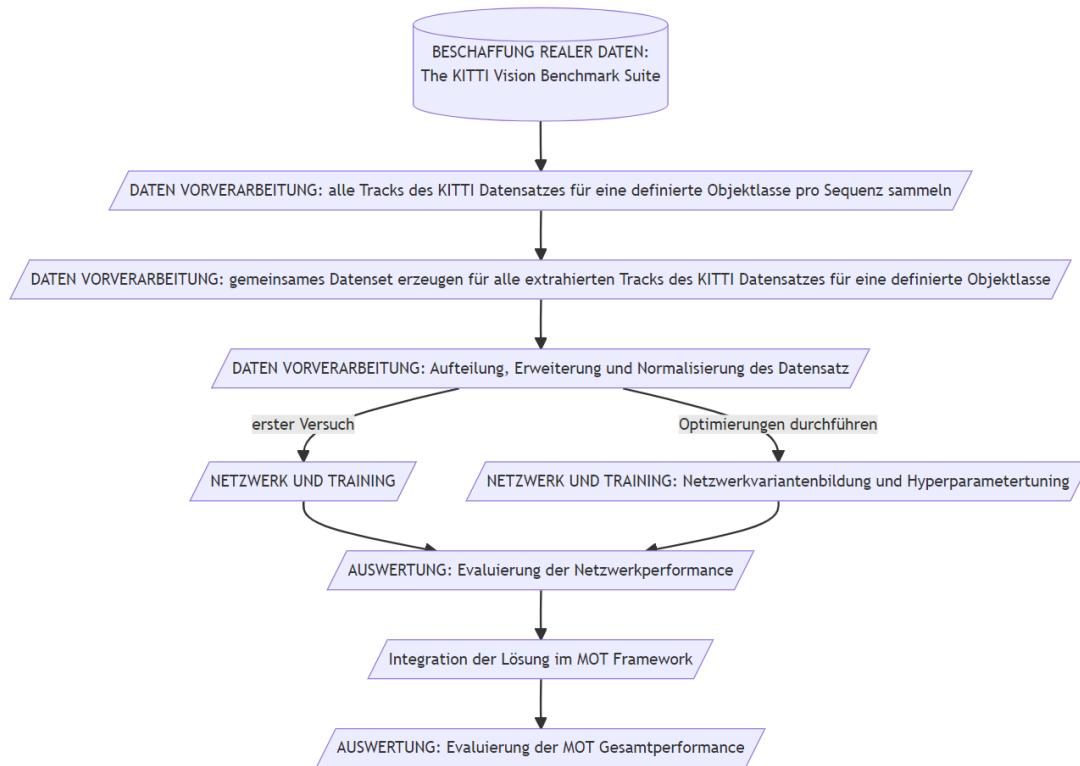
- [37] Margrit Betke; Zheng Wu. *Data Association for Multi-Object Visual Tracking*. 2016.
- [38] Ji Zhu; Hua Yang; Nian Liu; Minyoung Kim; Wenjun Zhang; Ming-Hsuan Yang. *Online Multi-Object Tracking with Dual Matching Attention Networks*. 2019.
- [39] Lionel Rakai; Huansheng Song; ShiJie Sun; Wentao Zhang; Yanni Yang. *Data association in multiple object tracking: A survey of recent techniques*. 2022.
- [40] Qi Chu; Wanli Ouyang; Hongsheng Li; Xiaogang Wang; Bin Liu; Nenghai Yu. *Online Multi-Object Tracking Using CNN-based Single Object Tracker with Spatial-Temporal Attention Mechanism*. 2017.
- [41] Wei-Chih Hung; Henrik Kretzschmar; Tsung-Yi Lin; Yuning Chai; Ruichi Yu. *SoDA: Multi-Object Tracking with Soft Data Association*. 2020.

# Anhang

- A. zu 3.3, Single Prediction Network (SPENT)
- B. zu 3.4, Single Association Network (SANT)
- C. zu 3.5, Multi Association Network (MANTa)

## A. zu 3.3, Single Prediction Network (SPENT)

### A.01 Workflow Netzwerkentwicklung und Integration in MOT Framework



## A.02 Auflistung der betrachteten Trainingsoptionen

Für das **IstmLayer** werden folgende Funktionen beim Training standardmäßig mit ausgeführt:

- **StateActivationFunction** - Aktivierungsfunktion zur Aktualisierung der Zelle und des verborgenen Zustands: 'tanh' - Verwendung der hyperbolischen Tangensfunktion (tanh).
- **GateActivationFunction** - Aktivierungsfunktion, die auf die Gates anzuwenden ist: 'sigmoid' - Verwendung der Sigmoidfunktion.
- **CellState** - Zustand der Zelle: Leerer Vektor, keine definierten Werte.
- **HiddenState** - Zustand der Hiddenstates: Leerer Vektor, keine definierten Werte.

Für das **IstmLayer** und auch für das **fullyConnectedLayer** werden folgende Funktionen beim Training standardmäßig mit ausgeführt:

- **WeightsInitializer** - Funktion zur Initialisierung von Gewichten: Die Initialisierung der Gewichte wird per Default ('glorot') mit dem Glorot-Initialisierer (auch bekannt als Xavier-Initialisierer) ausgeführt. Der Glorot-Initialisierer nimmt unabhängig Stichproben aus einer Gleichverteilung mit Mittelwert Null und Varianz  $2 / (\text{InputSize} + \text{OutputSize})$ .
- **BiasInitializer** - Funktion zur Initialisierung der Bias: Per Default werden alle Biaswerte mit Nullen initialisiert ('zeros').
- **WeightLearnRateFactor / BiasLearnRateFactor** - Lernratenfaktor für Gewichte / Biaswerte: Lernratenfaktor für die Gewichte / Biaswerte, angegeben als positiver Skalar (Default = 1). Wenn z. B. WeightLearnRateFactor den Wert 2 hat, ist die Lernrate für die Gewichte in dieser Schicht doppelt so hoch wie die aktuelle globale Lernrate. Die globale Lernrate wird in den Trainingskonfigurationen (mit der Funktion `trainingOptions`) angeben.

Das Netzwerktraining wurde mit dem folgenden **Solver-Optionen** durchgeführt:

- **Solver = 'adam'** - als bewährtes Verfahren für gradientenbasierte Optimierung stochastischer Kostenfunktionen (lossfunctions) erster Ordnung wurde Adaptive Moment Estimation (**ADAM**) gewählt [4].
- **InitialLearnRate = 0.001** - Initiale Lernrate: Anfängliche Lernrate, die für das Training verwendet wird (Standardwert für **ADAM**).

- **LearnRateSchedule = 'piecewise'** - Option zum Verringern der Lernrate während des Trainings (Standard = 'none'). Durch Auswahl des Parameters 'piecewise' wird die globale Lernrate nach einer bestimmten Anzahl von Epochen (LearnRateDropPeriod) durch Multiplikation mit einem bestimmten Faktor (LearnRateDropFactor) während des Trainings verringert.

**LearnRateDropPeriod = 10**

**LearnRateDropFactor = 0.1**

- **L2Regularization = 0.0001** - Das Hinzufügen eines Regularisierungsterms für die Gewichte zur Verlustfunktion ist eine Möglichkeit, Overfitting zu reduzieren (der Regularisierungsterm wird auch als 'weight decay' bezeichnet). Der gewählte Wert entspricht dem Standardwert.
- **GradientDecayFactor = 0.9** - Abklingrate des gleitenden Gradientenmittelwerts für **ADAM**, angegeben als positiver Skalar kleiner als 1. Der Standardwert 0.9 ist für die meisten Aufgaben gut geeignet [4].
- **Epsilon = 1e-8** - **ADAM** fügt den Offsetwert Epsilon während der Aktualisierungen der Netzparameter zum Nenner hinzu, um eine Division durch Null zu vermeiden.

Das Netzwerktraining wurde mit dem folgenden **Mini-Batch-Optionen** durchgeführt:

- **MaxEpochs = 30** - Maximale Anzahl der für das Training zu verwendenden Epochen. Eine Epoche ist ein vollständiger Durchlauf des Trainingsalgorithmus über den gesamten Trainingsdatensatz.
- **MiniBatchSize = 50** - Größe des für jede Trainingsiteration zu verwendenden Mini-Batch. Ein Mini-Batch ist eine Teilmenge der Trainingsmenge, die zur Auswertung des Gradienten der Verlustfunktion und zur Aktualisierung der Gewichte der Netzwerkschichten verwendet wird. Dabei ist darauf zu achten, dass der gewählte Wert MiniBatchSize ein Teilen der Samples des Trainingsdatensatz (hier: 500) ohne Rest ermöglicht.

Das Netzwerktraining wurde mit dem folgenden **Validation-Optionen** durchgeführt:

- **ValidationData** - Der erzeugte Datensatz wird während des Trainings verwendet, um die Validierungsgenauigkeit (validation accuracy) und den Validierungsverlust (validation loss) zu berechnen. Die Daten werden nicht für die Anpassung der Netzwerkparameter genutzt, sondern prüfen die Performance durch die Anpassungen auf Basis des Trainingsdatensatzes.

- **ValidationFrequency = 50** - Der Wert gibt die Anzahl der Trainingsiterationen an, welche zwischen den Bewertungen (Netzvalidierung) durchgeführt werden.
- **ValidationPatience = 5** - gibt an, wie oft der validation loss größer oder gleich dem zuvor kleinsten Verlust sein kann, bevor das Netztraining abgebrochen wird.
- **OutputNetwork = 'best-validation-loss'** - gibt das Netzwerk mit den Parametern zurück, welches während der Trainingsiteration mit dem geringsten Validierungsverlust erzeugt wurde.

Das Netzwerktraining wurde mit dem folgenden **LSTM** spezifischen **Sequenz-Optionen** durchgeführt:

- **SequenceLength = 'longest'** - Option zum Auffüllen, Abschneiden oder Teilen von Eingabesequenzen für das Netzwerktraining. Mit Auswahl der Option 'longest' werden die Sequenzen in jedem Mini-Batch so aufgefüllt, dass sie die gleiche Länge der längsten Sequenz besitzen. Bei dieser Option werden im Vergleich zum Kürzen keine Daten verworfen. Wie in [20] beschrieben, kann das Auffüllen zu Rauschen im Netzwerk führen, jedoch ist ein Sequenzanpassung pro Mini-Batch für das Training von **LSTM** Netzwerken erforderlich. Um den Effekt möglichst gering zu halten wurden die Sequenzen der Trainingsdaten daher vor dem Mini-Batch-Padding (Auffüllen) der Länge nach sortiert. In Abbildung 3.6 ist deutlich zu erkennen, dass das Padding (türkisfarbener Bereich) durch einen sortierten Trainingsdatensatz pro Mini-Batch minimiert wird. Wie weit die Netzwerkperformance durch die Anpassung der Mini-Batch Größe beeinflusst wird, ist im weiteren Verlauf der Arbeit dokumentiert.

## A.03 genutzter Kalman Filter (KF) für den Vergleich mit SPENT

Parameters for kalman filter

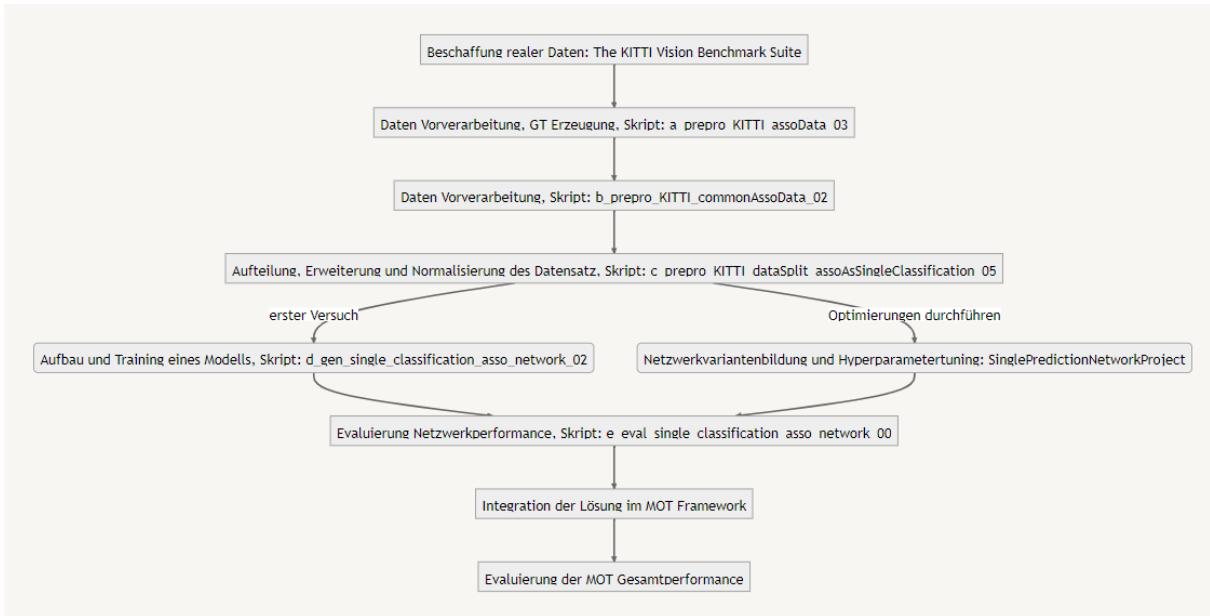
```
kittiParams = kitti_parameters()

kittiParams = struct with fields:
    p_d: 0.8678
    p_s: 0.9950
    birthrate: 4
    birth_x: [10 10 30 30]
    birth_y: [-7 7 -7 7]
    p_b: 0.0500
    lambda_c: 2
    range_c: [3x2 double]
    v_pos: 0.3162
    w_pos: 0.1732

model.T= 0.1;                                %sampling period
model.A0= [ 1 model.T; 0 1 ];                  %transition matrix
model.F= [ model.A0 zeros(2,3); zeros(2,2) model.A0 zeros(2,1); 0 0 0 0 1];
model.B0= [ (model.T^2)/2; model.T ];
model.B= [ model.B0 zeros(2,1); zeros(2,1) model.B0 ];
model.B_birth(:,:,1)= diag([ 10; 10; 10; 10; 1 ]);
model.sigma_v = kittiParams.v_pos;
model.Q= (model.sigma_v)^2* model.B*model.B';      %process noise covariance
model.Q = [model.Q zeros(4,1); 0 0 0 0 kittiParams.v_pos^2];
model.H= [ 1 0 0 0 0 ; 0 0 1 0 0; 0 0 0 0 1 ];      %observation matrix
model.D= diag([ kittiParams.w_pos; kittiParams.w_pos; kittiParams.w_pos ]);
model.R= model.D*model.D';                      %observation noise covariance
```

## B. zu 3.4, Single Association Network (SANT)

### B.01 Workflow Netzwerkentwicklung und Integration in MOT Framework



## B.02 SANT Trainingsoptionen

Der Skriptauszug zeigt die Implementierung für die ersten Durchläufe des Single Association Network (**SANT**). Das Objekt *options* wird wie dargestellt initialisiert an die Trainingsfunktion übergeben.

```
num_vec = 1:100000;
pos_dividor = num_vec(mod(size(featuresTrain,2),num_vec) == 0)
max_idx_pos_dividor = size(pos_dividor,2);
MBS = size(featuresTrain,2)/pos_dividor( 4 —————— )
ME = 60
VF = round(ME/MBS*100,2,"significant")
options = trainingOptions("adam", ...
    InitialLearnRate = 0.01, ...
    LearnRateSchedule = "piecewise", ...
    LearnRateDropFactor = 0.5 ,...
    LearnRateDropPeriod = 10, ...
    GradientDecayFactor = 0.9, ...
    MaxEpochs = 100, ...
    MiniBatchSize = MBS, ...
    Shuffle = "never", ...
    ValidationData = {featuresVerifi,labelsVerifi_mat}, ...
    ValidationFrequency = 50, ...
    Plots = "training-progress", ...
    Verbose = 0);
```

## C. zu 3.5, Multi Association Network (MANTa)

### C.01 MANTa Trainingsoptionen

```
num_vec = 1:1000000;
pos_dividor = num_vec(mod(size(MANTa_input_noisy_norm_TrackAndSoSet_training,2),num_vec) == 0)
max_idx_pos_dividor = size(pos_dividor,2);
MBS = size(MANTa_input_noisy_norm_TrackAndSoSet_training,2)/pos_dividor( 3 —————— )
ME = 30
VF = round(ME/MBS*[100 —————— ],2,"significant")
% VF = 100
options = trainingOptions( ...
    ... % solver options %
    "adam", ...
    InitialLearnRate = 0.01, ...
    LearnRateSchedule = "piecewise", ...
    LearnRateDropPeriod = 10, ...
    LearnRateDropFactor = 0.5 ,...
    L2Regularization = 0.0001, ...
    GradientDecayFactor = 0.9, ...
    SquaredGradientDecayFactor = 0.999, ...
    Epsilon = 1e-8, ...
    ... % validation options %
    ValidationFrequency = VF, ...
    ValidationData = {MANTa_input_noisy_norm_TrackAndSoSet_validation,MANTa_onehot_validation_mat}, ...
    ... % mini-batch options %
    MiniBatchSize = MBS, ...
    MaxEpochs = ME, ...
    OutputNetwork = 'best-validation-loss', ...
    Shuffle = 'never',...
    Plots = "training-progress", ...
    ... % display options %
    Verbose = 1);
```