Universität Regensburg

Philosophische Fakultät III
Sprach-, Literatur- und Kulturwissenschaften
Institut für Information und Medien, Sprache und Kultur (I:IMSK)
Lehrstuhl für Informationswissenschaft

Modulprüfung Computational Intelligence
Modul: INF-BA-M09
SS 2021
Leitung: Stefan Kerscher/ Prof. Dr. Bernd Ludwig

Trainieren eines neuronalen Netzes zur Prädiktion der Trajektorien von Fußgängern

Pascal Strobel
Matrikelnummer: 2106133
6. Semester Medieninformatik/ Informationswissenschaft
E-Mail: pascal.strobel@stud.uni-regensburg.de

Abgegeben am 15.09.2021

Inhalt

1 Einleitung

Die Vorhersage der Trajektorien von Fußgängern gewinnt mit der ansteigenden Verbreitung von autonomem Fahren und sozialen Robotern, die den Menschen am Arbeitsplatz (Asoh, Hayamizu, Hara, Motomura, Akaho & Matsui, 1997), im Krankenhaus (King & Weiman, 1991), im Museum (Burgard, Cremers, Fox, Hähnel, Lakemeyer, Schulz, Steiner & Thrun, 1999), aber auch in den eigenen vier Wänden (Schaeffer & May, 1999) unterstützen sollen, zunehmend an Bedeutung. Durch präzise Vorhersagen des Verhaltens von Fußgängern in der näheren Umgebung können mögliche Kollisionen frühzeitig erkannt und verhindert werden.

In dieser Arbeit wird mithilfe eines neuronalen Netzes (NN) ein datengesteuerter Lösungsansatz vorgestellt, um auf Basis vergangener Positionsdaten die zukünftige Trajektorie eines einzelnen Fußgängers vorherzusagen. Derartig datengesteuerte Ansätze für dieses Themengebiet haben sich vor allem in den letzten Jahren durch vielversprechende Resultate etabliert.

Die Arbeit gibt in Kapitel 2 zunächst einen Überblick über den aktuellen Forschungsstand zum Thema Trajektorien-Prädiktion von Fußgängern. In Kapitel 3 folgt die Erläuterung der zugrundeliegenden Problemstellung sowie der vorgeschlagene Lösungsansatz. Dabei werden zuerst die angewandten Datenvorverarbeitungstechniken und anschließend die Architektur des neuronalen Netzes, inklusive grundlegender mathematischer Konzepte, dargelegt. In Kapitel 4 findet sich neben der Evaluation der Performance auch eine qualitative Analyse für Szenarien, in denen der vorgestellte Ansatz scheitert. Die Diskussion, in der Limitierungen und mögliche Verbesserungsvorschläge aufgezeigt werden, und ein schlussendliches Fazit, runden die Arbeit inhaltlich ab.

2 Verwandte Arbeiten

Das von Helbing & Molnár (1995) vorgestellte „Social Force Model“ bildet die Grundlage für physikbasierte Ansätze zur Prädiktion von Fußgänger-Trajektorien. Es beruht auf der Theorie, dass Fußgänger externen Kräften ausgesetzt sind und deren Verhalten durch diese mathematisch beschrieben und vorhergesagt werden kann. Ein weiterentwickeltes physikbasiertes Modell ist das von Kim, Guy, Liu, Wilkie, Lau, Lin, & Manocha (2015) vorgestellte BRVO, welches eine bereits bestehende Fußgänger-Simulationsmethode RVO (van den Berg, Guy, Lin & Manocha, 2011) mit Online-Lernen kombiniert, um so eine individualisierte Vorhersage für jeden Fußgänger zu treffen.

Durch das in den letzten Jahren starke Wachstum an Popularität von künstlichen neuronalen Netzen, aber auch durch die laut Zamboni, Kefato, Girdzijauskas, Norén & Dal Col (2022) gegebene Limitierung von physikbasierten Ansätzen, dass diese aufgrund handgefertigter Funktionen nur eine Teilmenge aller möglichen Verhaltensweisen von Fußgängern darstellen können, sind vor allem Deep Learning Ansätze, ganz besonders rekurrente neuronale Netze (RNN), stark in der Literatur vertreten.

RNNs bilden mit ihrem internen Gedächtnis eine sehr leistungsstarke Kategorie von künstlichen neuronalen Netzen und eignen sich besonders gut für die Arbeit mit sequenziellen Daten, beispielsweise bei der automatischen Spracherkennung (Chorowski, Bahdanau, Cho & Bengio, 2014), maschinellen Übersetzung (Bahdanau, Cho & Bengio, 2014), oder der Klassifikation von Bildern/ Videos (Cao, Liu, Yang, Yu, Wang, Wang, Huang, Wang, Huang, Xu, Ramanan & Huang, 2015).

Da die Aufgabe der Trajektorien-Prädiktion von Fußgängern als Zeitreihenproblem angesehen werden kann, ist auch in dieser Domäne die Verwendung von RNNs weit verbreitet, speziell die der „Long Short-Term Memory“ Zellen, vorgestellt von Hochreiter & Schmidhuber (1997). Beispiele für Arbeiten, die auf LSTM basieren sind das „Social LSTM“ Model (Alahi, Goel, Ramanathan, Robicquet, Fei-Fei & Savarese, 2016), die Vorhersage von

Trajektorien in sehr großen Menschenmengen (Shi, Shao, Guo, Wu, Zhang & Shibasaki, 2019) sowie die Arbeiten von Tao, Jiang, Duan & Luo (2020) und Pfeiffer, Paolo, Sommer, Nieto, Siegwart & Cadena (2017). Viele der Arbeiten zu diesem Thema beziehen neben den vergangen Positionsdaten eines Fußgängers zusätzliche Kontextinformationen, wie beispielsweise die Trajektorien anderer Fußgänger (Alahi et al., 2016; Shi et al., 2019; Tao et al., 2020) oder räumliche Informationen (Bartoli, Lisanti, Ballan & Bimbo, 2017; Pfeiffer et al., 2017), mit in die Vorhersage ein. In dieser Arbeit ist das nicht der Fall, die Vorhersage der zukünftigen Bewegungsbahn erfolgt nur mithilfe der bekannten, vergangenen Positionen.

Bai, Kolter & Koltun (2018) zeigen, dass faltende neuronale Netze kanonische rekurrente Architekturen in einer breiten Anzahl von Aufgaben übertreffen und folgern das nötige Überdenken der Assoziation von Sequenzmodellierung und RNNs. Derartige Architekturen sind bei der Vorhersage von menschlichen Trajektorien derzeit allerdings nur spärlich vorzufinden. Aussichtsreiche Resultate der Arbeiten von Zamboni et al. (2022) und Nikhil & Morris (2019) bestärken jedoch die obige Schlussfolgerung und drängen zu weiterer Forschung in dem Bereich.

3 Methodik

In diesem Abschnitt wird zunächst die Problemstellung dargelegt und die informationswissenschaftliche Fragestellung abgeleitet. Anschließend wird ein möglicher Lösungsansatz präsentiert, welcher neben den verwendeten Vorverarbeitungstechniken der Daten und der Architektur des trainierten neuronalen Netzes auch Designentscheidungen und mathematischen Grundlagen erläutert.

3.1 Problemstellung

Die Prädiktion der Trajektorien von Fußgängern entspricht der Vorhersage des zukünftigen Bewegungspfades anhand einer bestimmten Anzahl zuvor beobachteter Positionen. Die informationswissenschaftliche Fragestellung, die sich daraus ergibt und für die diese Arbeit einen Lösungsansatz liefert, lässt sich demnach folgendermaßen formulieren:
„Wie kann anhand bekannter, vorheriger Positionen die zukünftige Trajektorie eines Fußgängers bestimmt werden?“

Zur Beantwortung der Frage soll auf Basis eines gegebenen Datensatzes, welcher Daten von sich bewegenden Fußgängern enthält, ein neuronales Netz trainiert und eine Regression durchgeführt werden. Jeder Datenpunkt im Datensatz besteht dabei aus einer Zeitreihe mit insgesamt 20 Zeitschritten und den dazugehörigen Werten für die aktuelle x- und y-Koordinate (in Metern) des Fußgängers. Die zeitlichen Intervalle zwischen zwei Zeitpunkten innerhalb eines Datenpunktes betragen durchgehend 400ms. Neben den Koordinaten sind keine weiteren kontextuellen Informationen über den Fußgänger oder seine Umgebung bekannt.

Das trainierte NN soll anschließend dazu genutzt werden, mithilfe der Positionsdaten aus den ersten acht Zeitpunkten eines Datenpunktes, die zukünftigen zwölf Positionen des Fußgängers vorherzusagen.

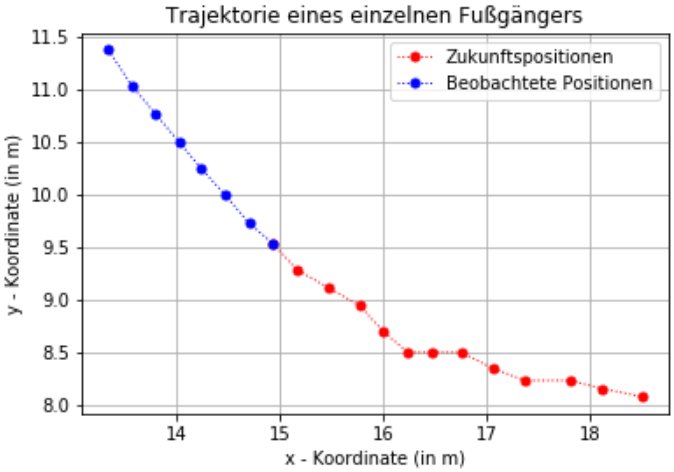

Abbildung 1: Beispiel-Trajektorie eines Fußgängers. Mithilfe der ersten acht beobachteten Positionen (blau) sollen die zukünftigen zwölf (rot) vorhergesagt werden. (eigene Darstellung)[1]

Diese acht-zu-zwölf-Verteilung ist gängig in der Literatur und beispielsweise in den Arbeiten von Alahi et al. (2016), Zamboni et al. (2022) und Nikhil & Morris (2019) zu finden. Die generelle Aufgabenstellung wird in Abb. 1 zusätzlich visualisiert.

3.2 Datenvorverarbeitung

Damit das neuronale Netz trainiert und ausgewertet werden konnte, mussten zunächst die gegebenen Daten entsprechend vorverarbeitet werden. Im Folgenden werden alle angewandten Schritte und Techniken dargelegt.

3.2.1 Normalisieren der Daten

Der ursprüngliche Datensatz enthält die Positionen der Fußgänger als absolute Koordinaten. Da allerdings kein Ursprung für diese festgelegt ist, können die Bewegungsdaten unterschiedlicher Fußgänger absolut weit auseinander liegen (Fußgänger 1 z. B. im Bereich 1 bis 10m, Fußgänger 2 im Bereich 90 bis 100m).

Zamboni et al. (2022) identifizieren neben dem naiven Ansatz, die absoluten Koordinaten ohne weitere Normalisierung zu verwenden, drei mögliche Normalisierungstechniken:

- Koordinaten haben den Ursprung im ersten beobachteten Zeitpunkt

[1] Abb. 1, 6, 7, 8, 9 und 10 wurden mithilfe der Matplotlib-Bibliothek (Straw et al., 2021) erstellt

- Koordinaten haben den Ursprung im letzten beobachteten Zeitpunkt
- Relative Koordinaten

Alle drei Ansätze führen dabei zu einer Performancesteigerung des in der Arbeit vorgestellten, faltenden neuronalen Netzes für unterschiedliche Datensätze, was den Nutzen der Normalisierung der Positionsdaten betont.

In dieser Arbeit werden die Daten gemäß dem zuletzt aufgeführten Stichpunkt normalisiert. Das bedeutet, dass das neuronale Netz nur mit den relativen Änderungen der x- und y-Koordinaten zwischen zwei Zeitpunkten arbeitet, welche im weiteren Verlauf der Arbeit als Delta-x (dx) und Delta-y (dy) bezeichnet werden. Zur Bestimmung der Delta Werte wurde für jeden Zeitpunkt eines Datenpunktes und für alle Datenpunkte im Datensatz die Differenz der x- und y-Koordinaten vom aktuellen Zeitpunkt t und zum vorherigen Zeitpunkt $t - 1$ berechnet.[2]

Das führt zu einer nötigen Spezialisierung der Anforderungen an das neuronale Netz: Aus den ersten acht Zeitschritten eines Datenpunktes werden je sieben dx und dy Werte berechnet. Das neuronale Netz soll demnach Zeitreihen der Länge sieben und mit den zwei Features dx und dy entgegennehmen und dafür die nächsten zwölf dx und dy Werte vorhersagen.

3.2.2 Aufteilen der Daten in Trainings-, Validierungs- und Testdaten

Die normalisierten Datenpunkte wurden nach dem Zufallsprinzip in Trainings- (80%), Validierungs- und Testdaten (je 10%) aufgeteilt. Zur Vermeidung von Data-Leakage erfolgte diese Aufsplittung vor der Standardisierung.

3.2.3 Standardisieren der Daten

Um sicherzustellen, dass die Wertebereiche der zwei Merkmale dx und dy nicht zu unterschiedlich sind, werden diese mithilfe der z-Transformation standardisiert, sodass für jedes Merkmal ein Erwartungswert von 0 und eine Standardabweichung von 1 vorliegt. Der z-Score des Samples x lässt sich mit folgender Formel berechnen:

[2] Die erneute Umwandlung in die absoluten Koordinaten erfolgt durch die einfache Addition der prädiktierten Delta-Werte auf die vorangehenden, absoluten Koordinatenwerte

$$z = \frac{x - \bar{x}}{s}$$

$\bar{x}$ ist dabei das arithmetische Mittel und s die Standardabweichung aller Trainingssamples.

Eine Verlagerung der Mittelwerte der Inputvariablen gegen null, wie es bei der z-Transformation der Fall ist, wird auch von LeCun, Bottou, Orr & Müller (2012) empfohlen, da jede Verschiebung des durchschnittlichen Inputs weg von null die Updates der Gewichte in eine bestimmte Richtung verzerrt und damit den Lernvorgang des neuronalen Netzes verlangsamt.[3]

3.2.4 Formatieren der Daten

Zur Bestimmung der nächsten zwölf Positionen eines Fußgängers wird in dieser Arbeit die Strategie der sequenziellen Vorhersage angewandt. Das bedeutet, das NN prädiziert basierend auf der Sequenz von je sieben dx und dy Werten immer nur den nächsten dx und dy Wert, statt alle zwölf auf einmal. Die neugewonnenen Informationen nach einer Vorhersage werden anschließend an die vorherigen sechs Inputdaten angehängt (der erste Wert fällt weg, da das NN auf Zeitreihen der Länge sieben als Input trainiert wurde), was wiederum den Input für die nächste Prädiktion bildet. Dieser Vorgang wird zwölfmal wiederholt.

[3] Die z-Transformation der Trainingsdaten, sowie die später folgende Umkehrung davon, erfolgte mithilfe des StandardScalers der Scikit-learn Bibliothek (Pedregose et al., 2011)

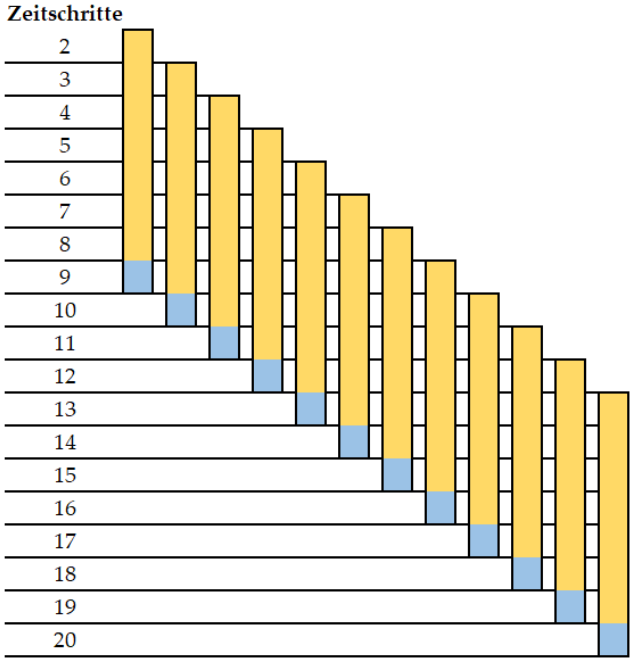

Abbildung 2: Prinzip der Zerlegung eines Datenpunktes mit 20 Zeitschritten in zwölf Trainingssamples (dargestellt durch farbige Balken). Jedes Trainingssample besteht aus einer Inputsequenz (orange) und aus dem Orakel (blau). Der erste Zeitschritt ist irrelevant, da für diesen keine Delta Werte berechnet werden können. (eigene Darstellung)

Damit das NN für diese Aufgabe trainiert werden konnte, mussten die Trainings- und Validierungsdaten entsprechend angepasst werden: Jeder Datenpunkt des Trainingsdatensatzes wurde in zwölf Trainingssamples zerlegt. Die Delta Werte von sieben aufeinanderfolgenden Zeitschritten bilden dabei immer den Input und der darauffolgende dx und dy Wert den gewünschten Output/ das Orakel zum Input. Abb. 2 verdeutlicht das Prinzip der Zerlegung.

3.3 Trainieren des neuronalen Netzes

In diesem Kapitel werden Hintergründe für grundlegende Designentscheidungen, sowie die verwendete Netzwerkarchitektur, vorgestellt und erläutert.

3.3.1 Rekurrente neuronale Netze (RNNs)

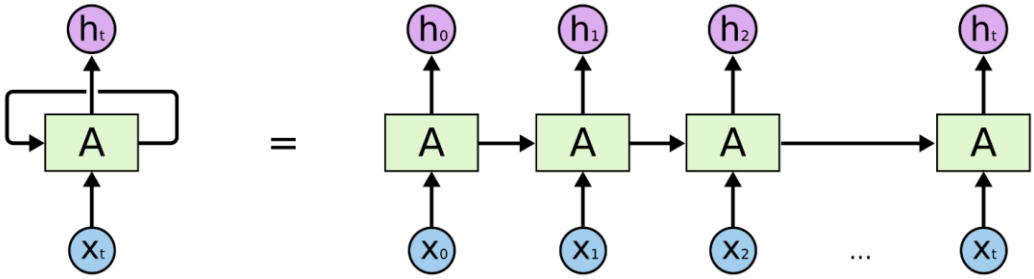

Abbildung 3: Ausrollen eines RNN. Links: Darstellung des RNN mit seinen Schleifen. Rechts: Offenbarung der Kettenform durch Ausbreiten des RNN. x_t kennzeichnet den Input, h_t den Output und A ein einzelnes Modul des RNN. (Oinkina, 2015)

Rekurrente neuronale Netze eignen sich durch ihre kettenähnliche Natur (siehe Abb. 3) sehr gut für das Arbeiten mit sequenziellen Daten und liefern in unterschiedlichen Domänen aussichtsreiche Resultate. Im Bereich der Prädiktion von Fußgänger-Trajektorien haben sie sich sogar als state-of-the-art Ansatz etabliert.

Wie auch klassische Feedforward-, oder faltende neuronale Netze, werden Trainingsdaten zum Lernen benötigt. Der Unterschied liegt im „Gedächtnis“ von RNNs, welches ermöglicht, dass Informationen aus früheren Inputs verwendet werden können, um die aktuellen Inputs und Outputs zu beeinflussen. Zudem hängt die Ausgabe von RNNs von den vorherigen Elementen innerhalb der Sequenz ab, während konventionelle tiefe neuronale Netze die Unabhängigkeit von Input und Output annehmen. Eine weitere spezielle Eigenschaft ist, dass sie sich innerhalb jeder Schicht des Netzwerkes den gleichen Gewichtungsparameter teilen, und nicht über jeden Knoten hinweg unterschiedliche Gewichtungen aufweisen.

RNNs nutzen den Backpropagation-Through-Time (BPTT) Algorithmus, welcher sich vom herkömmlichen Backpropagation Algorithmus in der Hinsicht unterscheidet, dass er die Fehler bei jedem Zeitschritt summiert, was bei konventionellen Feedforward-Netzwerken nicht der Fall ist, da hier die Gewichtungsparameter nicht über die Schichten geteilt werden.

Das führt dazu, dass RNNs vom Problem des verschwindenden Gradienten betroffen sein können. Das Problem tritt auf, wenn der Gradient, welcher der Steigung der Verlustfunktion entlang der Fehlerkurve entspricht, zu klein ist. Ist dies der Fall, so wird er kontinuierlich kleiner, die Gewichtungsparameter werden so lange aktualisiert, bis sie verschwindend gering sind, und das führt wiederum zur Stagnation des Lernvorgangs des Netzes. (IBM Cloud Education, 2020)

3.3.2 Long Short-Term Memory (LSTM)

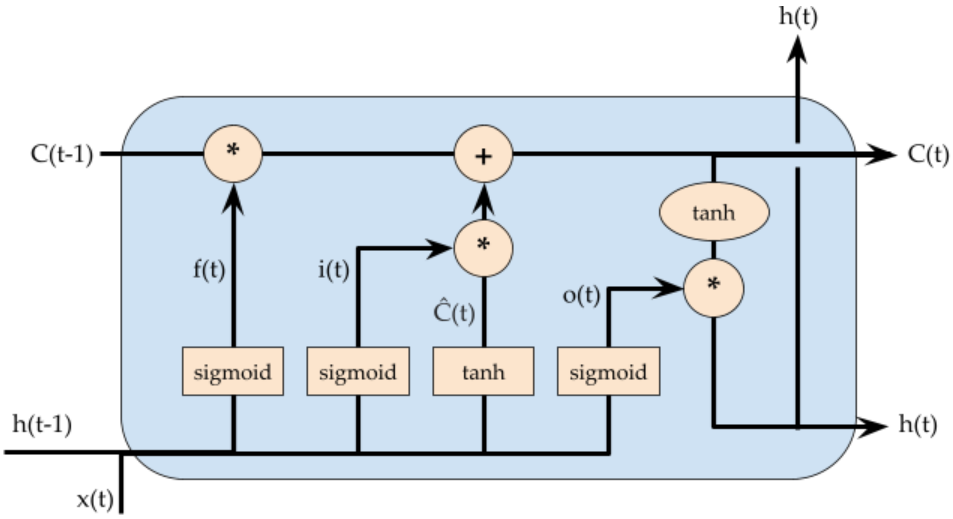

Abbildung 4: Aufbau eines einzelnen LSTM Moduls (In Anlehnung an Oinkina, 2015)

Das von Hochreiter & Schmidhuber (1997) eingeführte LSTM ist eine beliebte RNN-Architektur, welche das Problem des verschwindenden Gradienten behandelt. Wie auch bei traditionellen RNNs lässt sich LSTM als eine Kette von sich wiederholenden Modulen von neuronalen Netzen ansehen. Der Unterschied ist, dass ein Modul nicht nur eine, sondern vier Netzwerkschichten enthält (siehe Abb. 4). Das besondere an LSTM ist der Zellenzustand, mit dem Informationen aus der Eingangssequenz im Gedächtnis behalten werden können. Durch drei Gatter (Input-, Forget- und Output-Gatter), welche sich aus einer Sigmoid Netzschicht und einer punktweisen Multiplikation zusammensetzen, ist die Möglichkeit gegeben, Informationen zum

Zellenzustand hinzuzufügen oder zu entfernen. Jedes LSTM Modul arbeitet dabei folgende Gleichungen ab:

$$f_t = \sigma\,(W_f \cdot [h_{t-1}, x_t] + b_f),$$

$$i_t = \sigma\,(W_i \cdot [h_{t-1}, x_t] + b_i),$$

$$\hat{C}_t = tanh\,(W_C \cdot [h_{t-1}, x_t] + b_C),$$

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t,$$

$$o_t = \sigma\,(W_o \cdot [h_{t-1}, x_t] + b_o),$$

$$h_t = o_t * \tanh(C_t),$$

wobei x_t der Inputvektor zur Zeitinstanz t ist. h bezeichnet den versteckten Zustand, C den Zellenzustand und $\hat{C}$ den kandidierenden Zellenzustand. W_f, W_i, W_C, W_o sind die Gewichtungsmatrizen, um den Input-Gatter Vektor i_t, Output-Gatter Vektor o_t und Forget-Gatter Vektor f_t zu berechnen. b_i, b_o, b_f und b_C sind die Verzerrungsvektoren. $*$ repräsentiert die punktweise Multiplikation und σ die Sigmoid Funktion. (Oinkina, 2015)

3.3.3 Netzwerkarchitektur

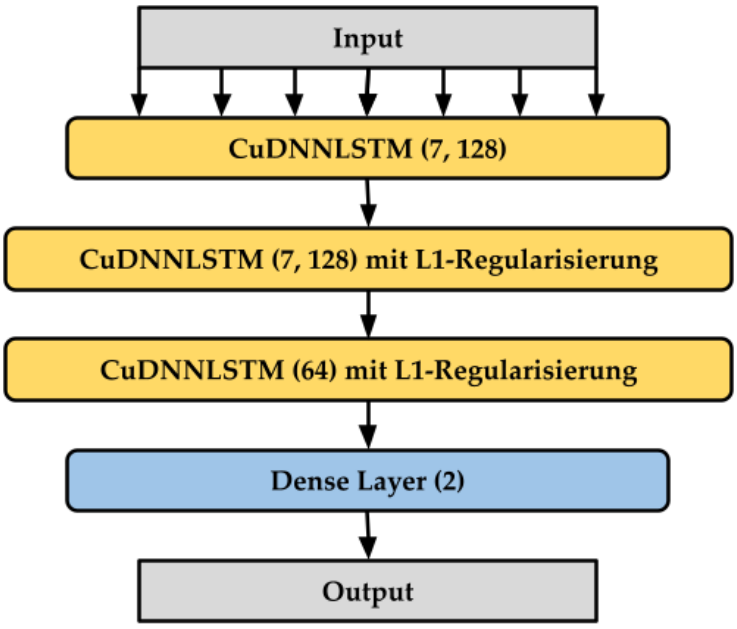

Abbildung 5: Architektur des verwendeten Models. Klammern enthalten die Form des Outputs einer Schicht. (eigene Darstellung)

Beruhend auf den aufgeführten Eigenschaften von RNNs und LSTM wurde mithilfe der Keras Bibliothek (Chollet et al., 2015) ein sequenzielles Model,

bestehend aus vier versteckten Schichten (Hidden Layer), erstellt (siehe Abb. 5). Die ersten drei Hidden Layer basieren auf dem traditionellen LSTM, bzw. CuDNNLSTM[4], und besitzen 128 (ersten zwei CuDNNLSTM-Schichten) und 64 Neuronen (dritte CuDNNLSTM-Schicht).

Als Gegenmaßnahme gegen Overfitting des Models wurde bei der zweiten und dritten CuDNNLSTM-Schicht L1-Regularisierung mit einem Regulierungsfaktor von 0.01 angewandt. Die finale Hidden Layer ist ein Dense Layer mit zwei Neuronen, wobei alle Neuronen mit allen Inputs und Outputs verbunden sind. Zudem besitzt der Dense Layer eine lineare Aktivierungsfunktion, da sowohl positive als auch negative Werte (Fußgänger können sich in negative x- bzw. y-Richtung fortbewegen) vom neuronalen Netz als Ergebnis ausgegeben werden sollen.

Als Input erwartet das neuronale Netz die vorverarbeiten Positionsdaten eines Fußgängers. Ein Input ist somit eine Sequenz aus sieben standardisierten, relativen Positionsänderungswerten (dx und dy).

Ausgegeben werden zwei Werte, nämlich der z-Score des prädiktierten dx und dy Wertes. Um die eigentlichen Deltas zu erhalten, müssen die ausgegebenen Werte mithilfe des zuvor gefitteten StandardScalers (siehe Kapitel 3.2.3) zurücktransformiert werden. Die vorhergesagten, absoluten Koordinaten berechnen sich anschließend aus der einfachen Addition der Deltas auf die jeweils vorangehenden Positionskoordinaten des Fußgängers.

3.3.4 Training

Das Model wurde mit einer Lernrate von 0,00005 und einer Ladungsgröße von 16 für insgesamt 30 Epochen trainiert. Die Wahl der verwendeten Hyperparameter erfolgte manuell und mithilfe der Validierungsdaten. Als Verlustfunktion wurde die mittlere quadratische Abweichung, welche sich durch folgende Formel bestimmen lässt, gewählt:

$$MQA = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

[4] CuDNNLSTM ist eine optimierte LSTM Implementierung der NVIDIA CUDA Deep Neural Network (CuDNN) Bibliothek

Dabei entspricht n der Anzahl der Trainingssamples, Y_i der Grundwahrheit und $\hat{Y}_i$ der Vorhersage für das aktuelle Trainingssample i.

4 Evaluation

In diesem Kapitel erfolgt die Auswertung der Performance des vorgestellten Netzwerks anhand von zwei gängigen Metriken. Zudem werden auf Basis einer qualitativen Analyse mehrere Szenarien identifiziert, für die der vorgestellte Ansatz scheitert.

4.1 Metriken

Ebenso wie in verwandter Literatur von bspw. Zamboni et al. (2022) oder Nikhil & Morris (2019) werden zur Auswertung der Performance die Metriken „Average Displacement Error" (ADE), erstmalig eingeführt von Pellegrini, Ess, Schindler & van Gool (2009), und „Final Displacement Error" (FDE), eingesetzt.

Der ADE eines einzelnen Fußgängers ist der durchschnittliche Fehler zwischen den prädizierten Positionen und der Ground-Truth für alle vorhergesagten Zeitschritte. Der ADE lässt sich mit folgender Formel berechnen:

$$ADE = \frac{\sum_{t=T_{beo}+1}^{T_{final}} \left\| Y_t - \hat{Y}_t \right\|}{T_{final} - T_{beo}}$$

T_{beo} ist der Zeitschritt der letzten Beobachtung des Fußgängers (in unserem Fall gilt T_{beo} = 8), T_{final} der final zu vorhersagende Zeitschritt eines Datenpunktes (in unserem Fall gilt T_{final} = 20). $\hat{Y}_t$ ist die vorhergesagte und Y_t die tatsächliche Position zum Zeitpunkt t. || repräsentiert die euklidische Distanz.

Der FDE eines einzelnen Fußgängers ist der Fehler zwischen der prädizierten Position und der zugehörigen Ground-Truth zum finalen Zeitpunkt T_{final} und lässt sich folgendermaßen berechnen:

$$FDE = \left\| Y_{T_{final}} - \hat{Y}_{T_{final}} \right\|$$

4.2 Resultate

Die Performance des trainierten RNN wurde anhand des zu Beginn abgespaltenen Testdatensatzes (10% der Gesamtdaten), welcher die Trajektorien von insgesamt 540 Fußgängern umfasst, ausgewertet, indem für jeden Datenpunkt sowohl ADE als auch FDE bestimmt wurde. Damit eine Gesamtaussage über die Performance des Systems für alle Testdaten getroffen werden konnte, muss der durchschnittliche ADE und FDE über alle Fußgänger berechnet werden, was zu folgendem Ergebnis führt:

Das RNN erzielt für die Testdaten einen ADE von 0,763 und einen FDE von 1,561.

4.3 Fälle des Scheiterns des Ansatzes

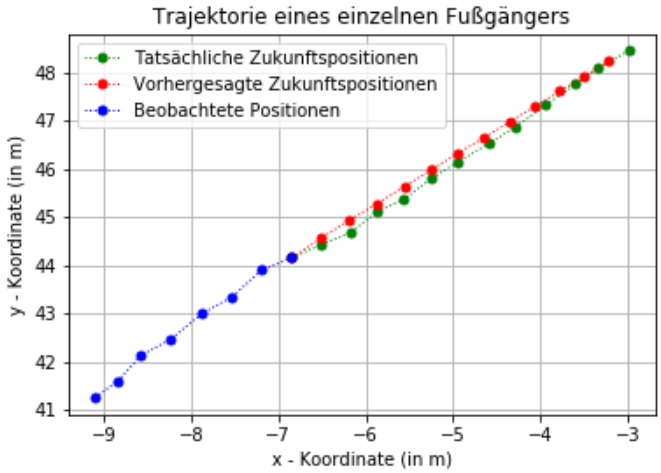

Abbildung 6: Präzise Vorhersagen durch triviales Fußgängerverhalten (eigene Darstellung)

Nicht für alle Fußgänger ist die Vorhersage der zukünftigen Trajektorie so trivial wie die aus Abb. 6. Abrupte Richtungswechsel oder Geschwindigkeitsänderungen sorgen für ein oft unberechenbares Verhalten von Fußgängern und damit zu Einbußen bei der Performance. Unzuverlässige Prädiktionen des vorgestellten Systems sind vor allem bei den folgenden drei Szenarien zu beobachten.

1. Fußgänger ändert plötzlich Richtung oder Geschwindigkeit, ohne einen Hinweis darauf:

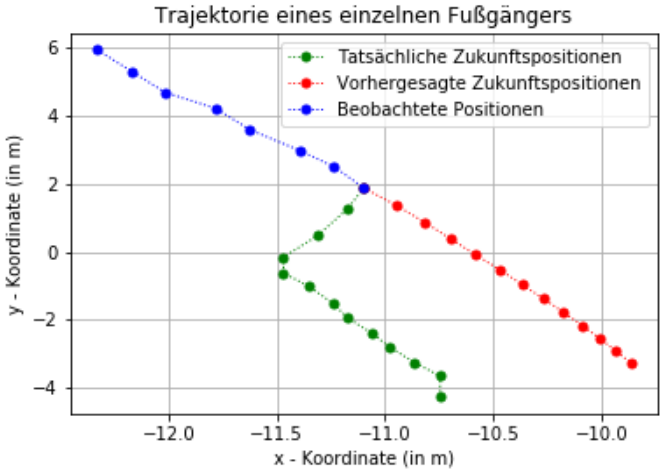

Abbildung 7: Überraschende Richtungsänderung (eigene Darstellung)

Die beobachtete Trajektorie des Fußgängers in Abb. 7 gibt keinerlei Hinweise auf eine mögliche Richtungsänderung. Die bestmögliche Vorhersage, die vom System (aber auch vom Menschen) getroffen werden kann, ist das Weiterführen der Bewegungsbahn. Damit derartige Bewegungsmuster besser prädiziert werden können, sind zusätzliche Kontext-Informationen (bspw. Straßenverlauf, Aufeinandertreffen mit anderen Fußgängern, o. ä.) notwendig.

2. Fußgänger ändert mehrfach seine Richtung oder Geschwindigkeit:

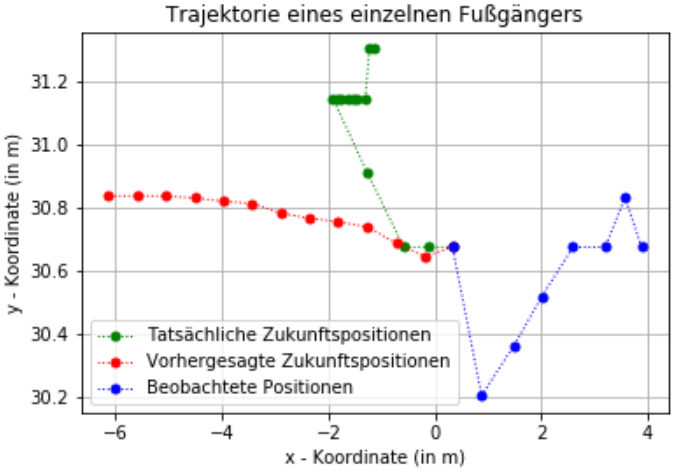

Abbildung 8: Mehrfache Richtungs- und Geschwindigkeitsänderungen (eigene Darstellung)

Nicht nur plötzliche, unerwartete, sondern auch häufige Richtungs- und Geschwindigkeitsänderungen in den beobachteten Positionen, führen zu Schwierigkeiten bei der Vorhersage (siehe Abb. 8). Ein richtiges Muster ist dabei

auch für den Menschen schwer erkennbar. Wie auch schon im ersten Fall, sind zusätzliche Kontext-Informationen nötig, um eine zuverlässigere Prädiktion zu ermöglichen.

3. Fußgänger ändert seine Position nicht:

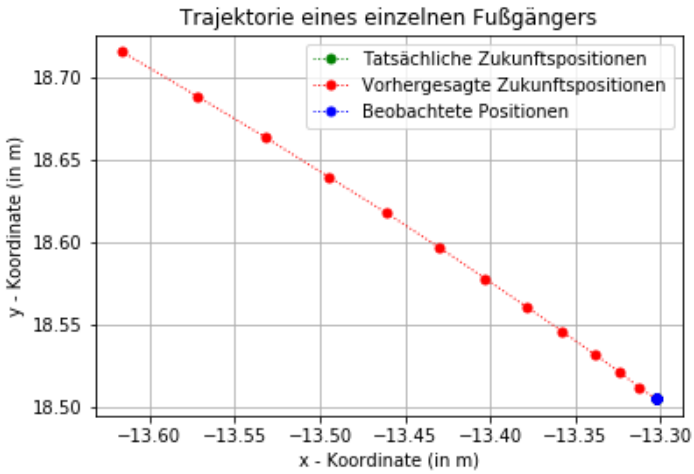

Abbildung 9: Keine Positionsänderung (eigene Darstellung)

Ebenfalls schlechte Vorhersagen des Systems sind zu beobachten, wenn sich der Fußgänger nicht bewegt (siehe Abb. 9). In derartigen Szenarien scheint das System nicht gelernt zu haben, keinerlei Bewegung in x- und y-Richtung vorherzusagen. Stattdessen werden Werte für dx und dy prädiziert, deren Betrag geringfügig von Null abweicht. Durch das sequenzielle Vorgehen bei der Bestimmung der nächsten zwölf dx und dy Werte akkumuliert sich dann der Fehler und führt zu zunehmenden Abweichungen zwischen Vorhersage und Ground-Truth.

5 Diskussion

Das vorgestellte System liefert zufriedenstellende Ergebnisse für triviale Bewegungsbahnen. Allerdings deuten sich einige Schwierigkeiten bei der Vorhersage von komplexerem Fußgängerverhalten, was nicht untypisch für Fußgänger ist, an.

Eine Möglichkeit zur Optimierung des Systems ist die automatische Feinabstimmung von Hyperparametern, da dies bisher manuell durchgeführt

wurde und somit nur eine kleine Anzahl an Hyperparameterwerten und Kombinationen getestet werden konnte.

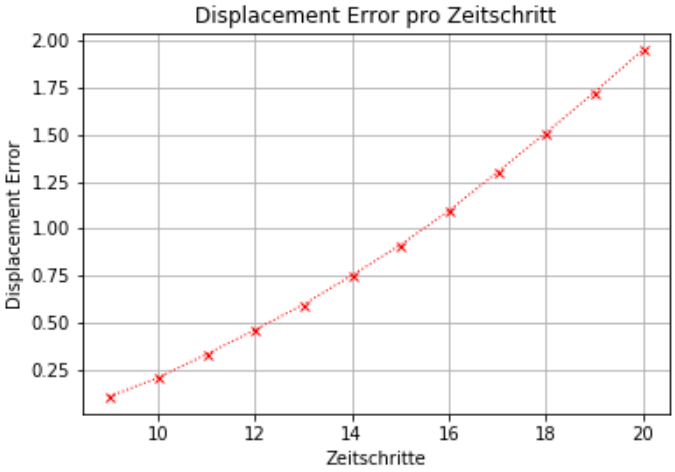

Abbildung 10: Akkumulierung des Fehlers für ferner in der Zukunft liegende Vorhersagen. Zeitschritt 9 ist der Zeitschritt der ersten und Zeitschritt 20 der der letzten Prädiktion. (eigene Darstellung)

Des Weiteren ist eine Verbesserung der Performance zu erwarten, wenn die zukünftigen Positionen nicht sequenziell, sondern per Multi-Output-Prädiktion auf einen Schlag vom neuronalen Netz bestimmt werden. Bei sequenziellen Vorhersagen akkumuliert sich der Fehler, deutlich erkennbar in der vorhergesagten Trajektorie aus Abb. 9. D. h. je weiter die Vorhersage in der Zukunft liegt, desto größer der Fehler (siehe Abb. 10), da diese auf möglichen vorherigen Fehlern basiert. Ein Multi-Output Netzwerk ist diesbezüglich resistenter.

Auch eine Erweiterung der verfügbaren Trainingsdaten durch das Sammeln weiterer Daten oder durch „Data Augmentation“ (bspw. durch das Rotieren/ Spiegeln/ Anwenden von Gaußschen Rauschen auf Fußgänger-Trajektorien (Zamboni et al., 2022)) kann sich positiv auf die Performance auswirken. Eine größere Anzahl an Trainingsbeispielen für bestimmte Fußgängerverhaltensweisen führt dazu, dass das neuronale Netz diese besser erlernen kann.

Weiterhin ist anzumerken, dass das vorgestellte System nur für einen Datensatz trainiert und getestet wurde. Damit ein Vergleich zu bereits

bestehenden Ansätzen aus verwandten Arbeiten gezogen werden kann, muss die Performance auch für andere, in der Literatur etablierte Datensätze, wie bspw. ETH (Ess, Leibe & van Gool, 2007), UCY (Lerner, Chrysanthou & Lischinski, 2007) oder TrajNet (Becker, Hug, Hübner & Arens, 2018), bewertet werden.

6 Fazit

Diese Arbeit präsentiert einen möglichen Lösungsansatz zur sequenziellen Prädiktion von Fußgänger-Trajektorien mithilfe eines rekurrenten neuronalen Netzes auf Basis von LSTM. Dabei wurden alle angewandten Datenvorverarbeitungsschritte, die Architektur des trainierten Netzes, sowie Designentscheidungen und mathematische Grundlagen aufgeführt und erläutert. Das System wurde anschließend bezüglich seiner Performance evaluiert und erzielt einen ADE von 0,763 und einen FDE von 1,561. Eine qualitative Analyse offenbart aussichtsreiche Ergebnisse für triviale Bewegungsbahnen, nicht-triviale führen allerdings zu unzuverlässigen Vorhersagen, was die Notwendigkeit der Optimierung des Systems betont. Die Arbeit zeigt zudem Limitierungen sowie mögliche Lösungsansätze zur Optimierung auf.

Literaturverzeichnis

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L. & Savarese, S. (2016, 27.–30. Juni). Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (S. 961–971). IEEE. https://doi.org/10.1109/CVPR.2016.110

Asoh, H., Hayamizu, S., Hara, I., Motomura, Y., Akaho, S. & Matsui, T. (1997). Socially embedded learning of the office-conversant mobile robot jijo-2. *IJCAI*(2), 880–887.

Bahdanau, D., Cho, K. & Bengio, Y. (2014, 1. September). *Neural Machine Translation by Jointly Learning to Align and Translate*. http://arxiv.org/pdf/1409.0473v7

Bai, S., Kolter, J. Z. & Koltun, V. (2018, 4. März). *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. http://arxiv.org/pdf/1803.01271v2

Bartoli, F., Lisanti, G., Ballan, L. & Bimbo, A. D. (2017, 6. Mai). *Context-Aware Trajectory Prediction*. http://arxiv.org/pdf/1705.02503v1

Becker, S., Hug, R., Hübner, W. & Arens, M. (2018, 20. Mai). *An Evaluation of Trajectory Prediction Approaches and Notes on the TrajNet Benchmark*. http://arxiv.org/pdf/1805.07663v6

Burgard, W., Cremers, A. B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W. & Thrun, S. (1999). Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, *114*(1-2), 3–55. https://doi.org/10.1016/S0004-3702(99)00070-3

Cao, C., Liu, X., Yang, Y., Yu, Y., Wang, J., Wang, Z., Huang, Y., Wang, L., Huang, C., Xu, W., Ramanan, D. & Huang, T. S. (2015, 7.–13. Dezember). Look and Think Twice: Capturing Top-Down Visual Attention with Feedback Convolutional Neural Networks. In *2015 IEEE International Conference on Computer Vision (ICCV)* (S. 2956–2964). IEEE. https://doi.org/10.1109/ICCV.2015.338

Chollet, F. & others. (2015). *Keras [Computer software]*. https://keras.io/

Chorowski, J., Bahdanau, D., Cho, K. & Bengio, Y. (2014, 4. Dezember). *End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First Results*. http://arxiv.org/pdf/1412.1602v1

Chun, W. H. & Wolfe, W. J. (Hrsg.) (1991). *Mobile Robots V. SPIE Proceedings*. SPIE.

Ess, A., Leibe, B. & van Gool, L. (2007, 14.–21. Oktober). Depth and Appearance for Mobile Scene Analysis. In *2007 IEEE 11th International Conference on Computer Vision* (S. 1–8). IEEE. https://doi.org/10.1109/ICCV.2007.4409092

Helbing & Molnár (1995). Social force model for pedestrian dynamics. *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics*, *51*(5), 4282–4286. https://doi.org/10.1103/PhysRevE.51.4282

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

IBM Cloud Education. (2020). *Recurrent Neural Networks.* IBM. https://www.ibm.com/cloud/learn/recurrent-neural-networks

Kim, S., Guy, S. J., Liu, W., Wilkie, D., Lau, R. W.H., Lin, M. C. & Manocha, D. (2015). BRVO: Predicting pedestrian trajectories using velocity-space reasoning. *The International Journal of Robotics Research*, *34*(2), 201–217. https://doi.org/10.1177/0278364914555543

King, S. J. & Weiman, C. F. R. (1991). HelpMate autonomous mobile robot navigation system. In W. H. Chun & W. J. Wolfe (Hrsg.), *SPIE Proceedings, Mobile Robots V* (S. 190–198). SPIE. https://doi.org/10.1117/12.48088

Leal-Taixé, L. & Roth, S. (Hrsg.). (2019). *Lecture Notes in Computer Science*. *Computer Vision – ECCV 2018 Workshops*. Springer International Publishing. https://doi.org/10.1007/978-3-030-11015-4

LeCun, Y. A., Bottou, L., Orr, G. B. & Müller, K.-R. (2012). Efficient BackProp. In G. Montavon, G. B. Orr & K.-R. Müller (Hrsg.), *Lecture Notes in Computer Science. Neural Networks: Tricks of the Trade* (Bd. 7700, S. 9–48). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_3

Lerner, A., Chrysanthou, Y. & Lischinski, D. (2007). Crowds by Example. *Computer Graphics Forum*, *26*(3), 655–664. https://doi.org/10.1111/j.1467-8659.2007.01089.x

Montavon, G., Orr, G. B. & Müller, K.-R. (Hrsg.). (2012). *Lecture Notes in Computer Science*. *Neural Networks: Tricks of the Trade*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35289-8

Nikhil, N. & Morris, B. T. (2019). Convolutional Neural Network for Trajectory Prediction. In L. Leal-Taixé & S. Roth (Hrsg.), *Lecture Notes in Computer Science. Computer Vision – ECCV 2018 Workshops* (Bd. 11131, S. 186–196). Springer International Publishing. https://doi.org/10.1007/978-3-030-11015-4_16

Oinkina. (2015). *Understanding LSTM Networks*. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Pedregose, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*(12), 2825–2830.

Pellegrini, S., Ess, A., Schindler, K. & van Gool, L. (2009, 29. September – 2. Oktober). You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision* (S. 261–268). IEEE. https://doi.org/10.1109/ICCV.2009.5459260

Pfeiffer, M., Paolo, G., Sommer, H., Nieto, J., Siegwart, R. & Cadena, C. (2017, 25. September). *A Data-driven Model for Interaction-aware Pedestrian Motion Prediction in Object Cluttered Environments*. http://arxiv.org/pdf/1709.08528v2

Schaeffer, C. & May, T. (1999). Care-o-bot-a system for assisting elderly or disabled persons in home environments. *Assistive technology on the threshold of the new millenium*, 3.

Shi, X., Shao, X., Guo, Z., Wu, G., Zhang, H. & Shibasaki, R. (2019). Pedestrian Trajectory Prediction in Extremely Crowded Scenarios. *Sensors (Basel, Switzerland)*, *19*(5). https://doi.org/10.3390/s19051223

Tao, C., Jiang, Q., Duan, L. & Luo, P. (2020). Dynamic and Static Context-aware LSTM for Multi-agent Motion Prediction. *ECCV 2020*. http://arxiv.org/pdf/2008.00777v1

Thomas A Caswell, Michael Droettboom, Antony Lee, Elliott Sales de Andrade, Tim Hoffmann, John Hunter, Jody Klymak, Eric Firing, David Stansby, Nelle Varoquaux, Jens Hedegaard Nielsen, Benjamin Root, Ryan May, Phil Elson, Jouni K. Seppänen, Darren Dale, Jae-Joon Lee, Damon McDougall, Andrew Straw, . . . Paul Ivanov. (2021). *matplotlib/matplotlib: REL: v3.5.0b1 [Computer software]*. Zenodo.

van den Berg, J., Guy, S. J., Lin, M. & Manocha, D. (2011). Reciprocal n-Body Collision Avoidance. In B. Siciliano, O. Khatib, F. Groen, C. Pradalier, R. Siegwart & G. Hirzinger (Hrsg.), *Springer Tracts in Advanced Robotics. Robotics Research* (Bd. 70, S. 3–19). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-19457-3_1

Zamboni, S., Kefato, Z. T., Girdzijauskas, S., Norén, C. & Dal Col, L. (2022). Pedestrian trajectory prediction with convolutional neural networks. *Pattern Recognition*, *121*, 108252. https://doi.org/10.1016/j.patcog.2021.108252

Erklärung zur Urheberschaft

Wir haben die Arbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie alle Zitate und Übernahmen von fremden Aussagen kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt.

Die vorgelegten Druckexemplare und die vorgelegte digitale Version sind identisch.

Regensburg, 14.09.2021 

Ort, Datum Unterschrift