Klassifizierung Multidimensionaler Zeitreihen mithilfe von Deep Learning

Manuel Meyer

Hamburg University of Applied Sciences, Dept. Computer Science,
Berliner Tor 7
20099 Hamburg, Deutschland
Email: manuel.meyer@haw-hamburg.de

Zusammenfassung: Gegenstand dieser Arbeit ist die Vorstellung eines Lösungsansatzes zur Klassifikation und Vorhersage von multidimensionalen Zeitreihen auf Basis von tiefen neuronalen Netzen. Deep Learning (DL) Methoden haben sich bereits in der Bildverarbeitung, Spracherkennung und zur Verarbeitung von natürlicher Sprache als sehr effizient erwiesen. Dennoch existieren immer noch viele Probleme bei der Entwicklung von Lösungen insbesondere bei komplexen und nichtlinearen Systemen. Deshalb wurde in einer vorhergegangen Arbeit [22], [21] bereits ein erster Ansatz evaluiert, der es möglich gemacht hat, Deep Learning Techniken auch auf komplexe Daten anzuwenden. Dieses Modell soll nun optimiert und durch einen neuen Ansatz erweitert werden, um auch zukünftige Ereignisse vorherzusagen.

Schlüsselwörter: LSTM, Vorhersage, Zeitreihen, Klassifikation, Faltungsnetzwerke, deep learning, heterogene Sensordaten, multidimensional, dynamisches Verhalten

1 Einführung

Maschinen lernen Denken: Der Einsatz von Robotern, Sensortechnik, Big Data und künstlicher Intelligenz spielen eine entscheidende Rolle in der Transformation von Wirtschaft und Gesellschaft. Maschinen werden smarter als zuvor und sorgen für einen grundlegenden Strukturwandel - denn diese technischen Systeme sind lernfähig und zunehmend in der Lage, ihr bereits erlerntes Wissen auf neue Situationen zu übertragen und ihr Verhalten intelligent anzupassen. Sie können lernen neue Prozesse zu verstehen und zu planen, Prognosen treffen und sogar mit Menschen und anderen Maschinen interagieren. Sensoren gehören dazu sicherlich zu einem der wichtigsten Datenlieferanten, denn nur mit ihnen können Zustände erfasst und Aktionen ausgeführt werden. Besonderes Potenzial birgt zudem der Bereich *Gesundheitswesen* und die Vorhersage von Krankheitsverläufen und Diagnosewahrscheinlichkeiten. Es existieren bereits Ansätze, um den Arzt bei den Diagnosen und Therapien von Krankheiten zu unterstützen. Dabei sollen durch Angabe der relevanten Faktoren und Symptome mit den aktuellen gemessenen Daten, die Patientendaten durchsucht und kombiniert werden, um dem Arzt anschließend eine Liste bewerteter und möglicher Diagnosen

mitzuteilen. Im Bereich *Predictive Maintance* sollen so in vielen Petabytes an Sensordaten Muster erkannt und im Anschluss auf mögliche Fehlfunktionen oder den Ausfall von einzelnen Bauteilen hingewiesen werden - Maschinen reparieren, noch bevor sie defekt sind. In dieser Arbeit soll ein Ansatz vorgestellt werden, wodurch mittels Deep Learning Technik aus multidimensionalen Daten das dynamische Verhalten, als auch die temporalen Abhängigkeiten erkannt werden. Für die korrekte Erkennung ist eine effiziente Extraktion der Merkmalen aus den dynamischen Zeit und Seriendaten wichtig. Durch das Erlernen der temporalen Struktur der Sequenz in den komplexen Daten soll im Anschluss durch die reine Einbeziehung der Vergangenheitswerte eine Vorhersage und Klassifikation von zukünftigen Ereignissen stattfinden.

Die Arbeit lässt sich insgesamt in 7 Kapiteln untergliedern. Das nachfolgende Kapitel 2 befasst sich mit einer kurzen Beschreibung der Grundlagen, gefolgt Kapitel 3 mit einer Vorstellung und Präsentation über verwandte wissenschaftlichen Arbeiten. Abschnitt 4 beschreibt das Modell detaillierter, gefolgt von Kapitel 5 einer Beschreibung des verwendeten Datensatz. In Kapitel 6 wird auf das weiter Vorgehen beschrieben, auf das im Hauptprojekt für die Masterarbeit hingearbeitet wird. In Kapitel 6.1 werden Risiken aufgezeigt, die bei der Realisierung entstehen können. Abschließend wird es ein Fazit zu dieser Arbeit geben.

2 Hintergrund

In dieser Arbeit sollen die Faltungsnetzwerke (*engl.* Convolutional Neural Networks (CNN)) mit den rückgekoppelten Netzen (*engl.* Recurrent Neural Networks (RNN)) kombiniert werden. Dadurch soll ein neu vorgeschlagener Ansatz eines Modells zur Klassifikation und Vorhersage von multidimensionalen Zeitreihen mittels Deep Learning entstehen. Das vorgeschlagene Modell dazu kann dabei in zwei Hauptkomponenten unterteilt werden: Zum einen die Convolutional Neural Networks und zum anderen die Recurrent Neural Networks. Beide Arten von Neuronalen Netzen werden nachfolgend noch einmal kurz erläutert.

2.1 Convolutional Neural Networks (CNN)

Bei den Convolutional Neuronal Networks (CNN) (dt. „Faltungsnetzwerke“) handelt es sich um eine Sonderform des Multilayer Perceptron mit einer speziellen Architektur. Faltungsnetzwerke verzeichnen große Erfolge im Bereich der Objektklassifizierung, ein wichtiger Meilenstein ist dabei die berühmte *LeNet-5-Architektur* [16] (LeCun/Bottou/Bengio/Haffner 1998:2278-2324) gewesen. Gegenüber anderen Bildklassifizierungsverfahren unterscheiden sie sich insbesondere durch ihre Unempfindlichkeit gegen jegliche Objekttransformation, wie Skalierung, Rotation und Translation. Möglich machen das die Faltungskerne, die während des Trainings angelernt werden. Das führt folglich zu einer Reduktion der Trainingsmenge und -Zeit. Sie sind aber nicht nur auf Aufgaben der Bilderverarbeitung beschränkt, in anderen Bereichen wie Stimmerkennung und

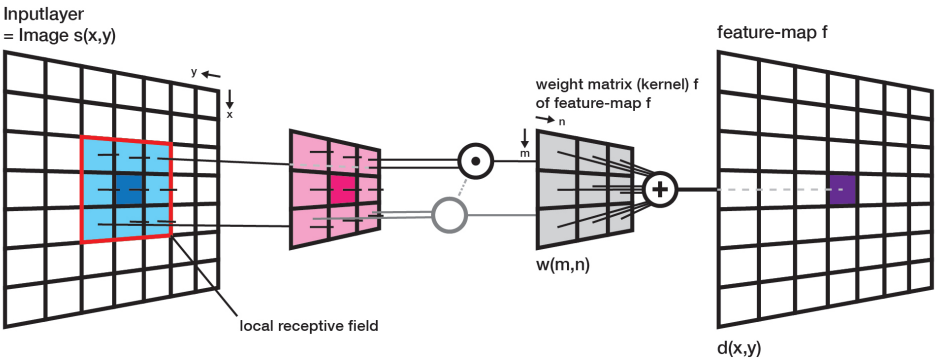

Abb. 1. *Eine Feature-Map f wird durch Faltung des Input-Layers mit der Gewichtsmatrix f berechnet,[19].*

Sprachverarbeitung (NLP) sind sie ebenfalls erfolgreich. Bei den CNN handelt es sich um neuronale Netze, die eine Konvolution[1] (Faltung) anstelle von Matrix-Multiplikation nutzen in mindestens einer Schicht. Die Neuronen in der ersten Faltungsschicht sind dabei nicht mit jedem Pixel im Eingabebild verbunden, stattdessen nur mit den Pixel im lokalen Wahrnehmungsfeld (*eng.* local receptive field). Daraus folgt, dass jedes Neuron im zweiten CNN Layer ausschließlich mit Neuronen innerhalb eines kleinen Bereiches mit der ersten Schicht verbunden ist. Der Architektur ist es dadurch möglich, in der ersten verborgenen Schicht sich auf kleinteilige Merkmale zu fokussieren, welches sich wiederum in der nächsten verborgenen Schicht zu übergeordneten Merkmalen zusammensetzen und immer weiter so. Es findet in jedem Layer eine weitere Filterung statt und mit der steigenden Anzahl von Schichten steigert sich auch die Komplexität im Layer. Die Gewichte eines Neuronen lassen sich als kleines Bild in der Größe des *local receptive field* darstellen (sieh Abbildung 1). Die Gewichte werden auch Filter, Faltungsmaske, Faltungsmatrix oder kernel genannt, welche in Abbildung 1 als w gekennzeichnet sind. Die Abbildung 1 soll die Anwendung des Filter demonstrieren. Es zeigt wie ein Filter auf ein Wahrnehmungsfeld oder Bildbereich angewandt und durch die Berechnung von korrespondierenden Punkten zum Schluss ein neuer Pixel erzeugt wird. So bildet eine Schicht Neuronen mit dem gleichen Filter eine *Feature-Map* (dt. Abbildung von Merkmalen). Jeder Filter dient dazu, Merkmale von Objekten aus den entsprechenden Wahrnehmungsfeldern zu erkennen, z.B. Linien, Bögen, Rechtecke oder Kanten. Beschreiben lässt sich die

[1] Bei einer Konvolution (Faltung) handelt es sich um mathematische Operation und hängt eng mit Fourier-Transformation und der Laplace-Transformation zusammen. Sie finden häufig in der Signalverarbeitung Anwendung. Die Faltungsschichten verwenden Kreuzkorrelationen, die der Konvolutionen ähnlich sind.

Faltung wie folgt:

$$d(x,y) = b_f + \sum_{m=-b}^{a} \sum_{n=-b}^{b} w(m,n) \cdot s(x-m, y-n), \tag{1}$$

mit b_f als der Bias-Term der Feature-Map f. Während des Trainings werden die Filter trainiert und werden entsprechend immer komplexer und lernen auch neue komplexe Feature bzw. Muster zu identifizieren oder kombinieren. Jeder Faltungskern dient dazu, ein bestimmtes Merkmal von zu klassifizierenden Objekten innerhalb eines Bildes zu erkennen.

2.2 Long Short-Term Memory

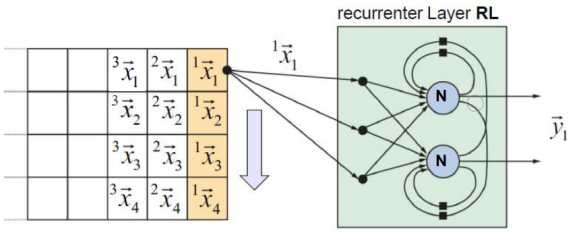

Abb. 2. *Ein Recurrent Layer (RL) mit einem TBPTT-Zeigenster mit 4 Inputvektoren (Timesteps),[20]*

Ein offensichtlicher Nachteil von neuronalen Netzen (NN) sind, dass die Eingaben eine feste Länge haben müssen. Damit eigenen sie sich nicht unbedingt zum Verarbeiten von Sequenzen mit beliebiger Länge von Vektoren. Hier kommen die rekurrenten neuronalen Netze ins Spiel. Sie eignen sich nicht nur zum Verarbeiten von einzelnen Eingaben, sondern auch von einer Abfolge von Eingaben, deren Länge vorher nicht festgelegt worden ist. Durch ihre rückwärts gerichteten Verbindungen sind sie in der Lage sich selbst zu beeinflussen, wodurch sie sich zu einem leistungsfähigen Werkzeug zur Modellierung von Sequenzen eignen. Die Abbildung 2 soll beispielhaft den Aufbau einer Schicht rekurrenter Neuronen (RL-Recurrent Layers) verdeutlichen. Es soll zeigen, das ein RNN nicht nur aus einem einzelnen rekurrenten Neuron besteht, sondern auch eine ganze Schicht mit rekurrenten Neuronen erstellt werden kann. Die nachfolgende Abbildung 3 geht einen Schritt weiter. Hier wurde das RNN entlang der Zeitachse dargestellt. Das nennt man auch das *„Netz entlang der Zeitachse aufrollen“*. Jeder RL kann dementsprechend auch als Zeitschritt (Timestep) interpretiert werden.Zu jedem *Ausführungszeitpunkt* t erhält der RL die Eingabe $\boldsymbol{x}_t$ sowie seine eigene Ausgabe aus dem vorhergegangen Zeitschritt $\boldsymbol{y}_{t-1}$. Um ein RNN zu trainieren wird auf das *backpropagation through time* (BPTT) zurückgegriffen. Der Trick bei diesem Verfahren liegt darin, die RNNs entlang der Zeitachse aufzurollen und

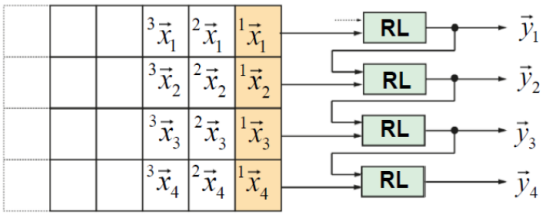

Abb. 3. *Zeitlich aufgerollter Recurrent Layer (RL) in 4 aufeinanderfolgenden Zeitschritten,[20]*

im Anschluss das gewöhnliche Backpropagation-Verfahren anzuwenden. Ein bereits bekanntes Problem ist, dass es keine Möglichkeit gibt Informationen *unmittelbar* weiterzugeben, denn alle Parameter spielen bei jedem Zeitschritt die gleiche Rolle, d.h. in jeden Berechnungsschritt werden die gleichen Parameter verwendet. Daraus folgt, dass der Einfluss von früheren Faktoren auf spätere Berechnungsschritte verloren gehen. Um dem Problem entgegenzuwirken führt man sog, *gates* ein, die bestimmte Information unmittelbar (d.h. unverändert) über beliebig viele Zwischenschritte weiterleiten können. Da es sich bei Sensordaten um Zeitreihen mit weitreichenden temporalen Abhängigkeiten handelt, wo der Einfluss von früheren Faktoren auf aktuelle Berechnungen mit berücksichtigt werden soll, wird das RNN zusätzlich um eine Langzeitgedächtniszelle oder auch *LSTM*-Zelle erweitert. Die *Long Short-Term Memory* (LSTM)-Zelle wurde im Jahr 1997 von Sepp Hochreiter und Jürgen Schmidhuber [11] vorgeschlagen und im Laufe der Jahre von mehreren Wissenschaftlern wie Alex Graves, Hasim Sak [27](2014:338-342) und vielen anderen verbessert. Man könnte das ganze auch automatentheoretisch auffassen und sich das RNN als Automat vorstellen: es gibt eine Folge von Eingabe $\overrightarrow{x}_1, ..., \overrightarrow{x}_n$ und nach jeder Eingabe wird ein neuer Zustand $\overrightarrow{s}_i$ erreicht. Daraus resultiert eine Zustandsfolge $\overrightarrow{s}_0 \overset{\overrightarrow{x}_1}{\Rightarrow} \overrightarrow{s}_1 \overset{\overrightarrow{x}_2}{\Rightarrow} ... \overset{\overrightarrow{x}_n}{\Rightarrow} \overrightarrow{s}_n$. Das Problem bei dieser Folge ist, dass an jedem Zeitpunkt t_i das Netz keinen Zugriff auf den unmittelbaren vorhergehenden Zustand, sowie die neue Eingabe hat. Damit bleiben Muster weiterhin unsichtbar, insbesondere voneinander entfernte weitreichende Abhängigkeiten zwischen Eingaben. Die Architektur einer LSTM-Zelle ist in Abbildung 4 dargestellt. Die Idee der Architektur ist, den Zustandsvektor $\overrightarrow{s}$ in zwei einzelne Vektoren aufzuteilen: $\boldsymbol{h_{(t)}}$ und $\boldsymbol{c_{(t)}}$ (≫c≪für cell). Dabei kann $\boldsymbol{h_{(t)}}$ sich als Kurzzeitgedächtnis und $\boldsymbol{c_{(t)}}$ als Langzeitgedächtnis vorgestellt werden. Der Grundgedanke ist, dass das Netz lernen kann, was im Langzeitgedächtnis gespeichert werden soll, was vergessen werden kann und wie es das Gedächtnis interpretieren könnte. Während der Langzeitzustand $\boldsymbol{c_{(t-1)}}$ Informationen *fast* unverändert weiterreichen kann, passiert er die entsprechenden *gates*, die die Weitergabe kontrollieren und optimieren. Der Zustand $\boldsymbol{h_{(t-1)}}$ wird an vier einzelnen vollständig verbundenen Schichten übergeben, komplett bearbeitet und durchläuft dabei verschiedene Rechenschritte. Näheres dazu kann in der Arbeit *„Long short-term memory recurrent neural network architectures for large scale acoustic modeling“* [27] (H. Sak/A. Senior/F. Beaufays 2014:338-

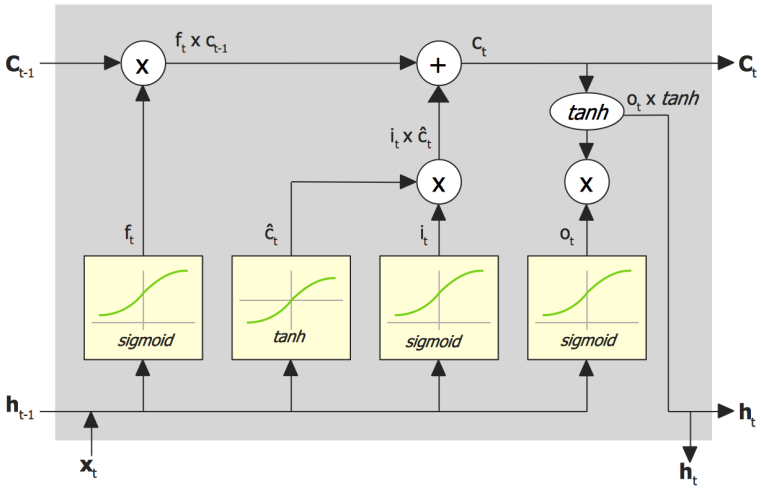

Abb. 4. *Einfache LSTM-Zelle, [5]*

342) nachgelesen werden. Die *gates* aus der Abbildung 4 können wie folgt beschrieben werden [28], [25]:

- ***Input Gate*:**
  Das Input Gate kontrolliert, welche Teile von $\tilde{\boldsymbol{c}}_{\boldsymbol{t}}$ in das Langzeitgedächtnis einfließen (wird von $\boldsymbol{i}_{(\boldsymbol{t})}$ gesteuert).
- ***Forget Gate*:**
  Das Forget Gate kontrolliert, welche Teile von dem Langzeitgedächtnis behalten werden soll (wird von $\boldsymbol{f}_{(\boldsymbol{t})}$ gesteuert).
- ***Output Gate*:**
  Das Output Gate kontrolliert welche Teile des Langzeitgedächtnis ausgelesen und ausgegeben werden sollen (ausgegeben an $\boldsymbol{h}_{(\boldsymbol{t})}$ und $\boldsymbol{y}_{(\boldsymbol{t})}$; von $\boldsymbol{o}_{(\boldsymbol{t})}$ gesteuert); gleichzeitig handelt es sich dabei auch um den Output des Zeitschrittes.

Die nachfolgenden Formel 2-7 sollen zusammengefasst noch einmal nachvollziehbar beschreiben, wie sich der Zustand des Langzeitgedächtnis, des Kurzzeitgedächtnis und die Ausgabe eines Zeitschrittes für einzelnen Datenpunkte berechnen lassen.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{4}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{5}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{6}$$

$$h_t = 0_t \cdot \tanh(C_t) \tag{7}$$

- $\boldsymbol{W_f, W_i, W_{\tilde{c}}, W_o}$ sind Gewichtsmatrizen des Eingabevektor $\boldsymbol{x_t}$ zu den vier Schichten, sowie Gewichtsmatrizen der Verbindungen des Kurzzeitgedächtnisses $\boldsymbol{h_{t-1}}$ zu den vier Schichten.
- $\boldsymbol{b_f, b_i, b_c, b_o}$ entsprechen die Bias-Terme der vier Schichten.

Es wurde gezeigt, warum LSTM-Architekturen so ausgesprochen erfolgreich beim Erkennen von weitreichenden und temporalen Abhängigkeiten sind, vor allem, die das Modellieren von Sequenzen erfordern. Sie liefern derzeit state-of-the-art Ergebnisse für eine Menge von Aufgabengebiete. Dennoch ist ein Problem dieser Architektur, dass sie relativ kompliziert ist und Ergebnisse nur schwer nachvollziehbar sind.

3 Verwandte Wissenschaftliche Arbeiten

Obwohl die Gebiete der Klassifikation und Vorhersage schon immer große Aufmerksamkeit erfahren haben, existieren immer noch viele Probleme bei der Entwicklung von Lösungen, insbesondere bei komplexen und nichtlinearen Systemen. Zu dieser Arbeit existieren nach dem aktuellen Stand keine direkt vergleichbare Studien. Es kommen lediglich einige frühere Arbeiten infrage, die zum Teil als Basis dienen könnten. Zudem sollen weiterhin auch andere Lösungen als konkurrierende Systeme betrachtet und kurz vorgestellt werden. Im Gebiet der komplexen Systeme in der Arbeit [18] geht es um eine Anomalieerkennung einer Kraftwerksanlage. Mittels Unterstützung durch SVM/SVR konnten aus den heterogenen Daten die zugrunde liegende Dynamik des Systems verstanden und entsprechend überwacht und die Zustände durch eine Klassifizierung interpretiert werden. Durch die ansteigende Entwicklung von IoT-Geräten erfahren auch viele anderer Bereich große Beachtung, insbesondere stehen dabei die Smart Home-Systeme im Fokus aber auch die Erkennung und Vorhersage menschlicher Aktivitäten. Die Autoren der Arbeit [1] beschäftigten sich mit dem Gebiet der Prognose im Smart Home-Systemen. Sie basierte auf An/Aus Zustände von Haushaltsgeräten und arbeitetet dabei auf der Grundlage von Markov-Modellen um durch Partial Matching (PPM) die nächste Aktivität einer Sequenz aus vorhergegangenen Sequenzen vorherzusagen. Dabei konnte eine Prognosegenauigkeit von ca. 88% erreicht werden. Eine weitere beliebte Methode sind die Bayes'schen Netze. In der Arbeit [24] nutzten die Autoren den Ort und die Zeit einer Aktivität, um daraus die nächste Aktivität vorauszusagen. Dabei betrachteten sie ein einfaches Smart Home-System und konzentrierten sich auf tägliche Routinen wie Essen, Baden, Fernsehen, Essen kochen etc.. Das Ergebnis der Arbeit ergab aber lediglich eine teilweise, aber nicht kontinuierliche erfolgreiche Identifizierung der nächsten Bewohneraktivitäten. Mittlerweile entwickelt sich auch Deep Learning zu einer der wichtigsten Technologien in einigen Bereichen (wie bspw. die Automobilindustrie, Energiemanagement etc.) und sind bereits einer der beliebtesten Methoden für KI-bezogenen Aufgaben. Dazu gehört neben der Bilderverarbeitung [17] auch die natürlich Sprachverarbeitung (NLP) [4] sowie Spracherkennung [6]. Exakt diese Ansätze wurden in

der Arbeit *„An Activity-Embedding Approach for Next-Activity Prediction in a Multi-User Smart Space“* [13] (Kim et al. 2017:1-6) verwendet, um die nächsten Aktivitäten von mehreren Benutzern in Büro- und Konferenzräumen zu generieren und vorherzusagen. Die Grundlage basiert auf einen bestimmten Worteinbettungsalgorithmus, welcher normalerweise für NLP verwendet wird. Es werden Aktivitäten einem Vektor im Vektorraum zugeordnet. Nach der Vektorisierung wird ein LSTM Netz entsprechend trainiert, um die nächste Aktivität zu prognostizieren. Es konnte in dieser Arbeit ein Genauigkeit von ca. 82% verzeichnet werden.
Eine weitere interessante und naheliegende Arbeit ist *„Effect of dynamic feature for human activity recognition using smartphone sensors“* [23] (Nakano/Chakraborty 2017:539-543). Es geht um die Erkennung menschlicher Aktivitäten aus heterogenen Sensordaten (sechs versch. Aktivitäten). Die Sensordaten stammen aus einem Beschleunigungssensor und Gyroskop. Der Grundgedanke ist gewesen, die bereits erfolgreichen Convolutional Neural Networks mit konkurrierenden aber herkömmlichen und vielversprechenden Klassifikatoren wie dem Multilayer Perceptron (MLP), Support-Vector-Machine (SVM) oder k-Nearest neighbour (KNN) Verfahren zu vergleichen.

Da die CNNs ihre Stärken in der Bildklassifizierung haben, wurden aus jeder Aktivität der Sensordaten ein *recurrence plot* [8] Plot erzeugt. Rekurrenzplot sind eine moderne Methode der nichtlinearen Datenanalyse und sollten dem CNN als Merkmal der Aktivität dienen, um dadurch die entsprechende Aktivität zu lernen und zu identifizieren. Bei der Evaluation überzeugten die SVM mit einer Genauigkeit von 96%. Das MLP, k-NN sowie das CNN erreichten lediglich eine Klassifikationsgenauigkeit von 90%. Das Ziel der folgende Veröffentlichung *„A Hybrid Deep Representation Learning Model for Time Series Classification and Prediction“* [10] (Guo/Wu/Ji 2017:226-231) war es ein Modell zu entwickeln, welches sich sowohl für die Klassifikation als auch für die Vorhersage eignet. Die Autoren setzten auf einen Zusammenschluss von CNN und RNN. Sie kombinierten die Modelle so miteinander (Anzahl Layer und Position), dass es am Ende drei unterschiedliche aber vergleichbare Entwürfe gab. Das Ziel ist es gewesen, kommentierte Videoaufzeichnungen von 5 Teilnehmern 4 unterschiedliche Aktivitäten aus den Frames zu klassifizieren und im Anschluss vorherzusagen. Sie veränderten lediglich die Art des Training um eine Aktivität vorherzusagen. Der Datensatz ist dabei fest sortiert gewesen und folgte immer derselben Reihenfolge von Aktivitäten. Letztendlich ist das Resultat der Arbeit eine Genauigkeit der Klassifikation von 87% und eine Vorhersage von 80% gewesen.
Die Grundprojektarbeit *„Klassifizierung Multidimensionaler Zeitreihen mithilfe von Deep Learning“* [21] (Meyer 2018:1-36) ist eine vorhergegangene Arbeit und noch ein andauerndes Projekt, welches sich auf komplexen und dynamische Datensätze insbesondere auf Zeitreihendaten fokussiert. Der erste Ansatz war es gewesen, aus multidimensionalen Daten, eine zuverlässige Klassifikation mithilfe von Deep Learning Techniken zu entwickeln. Der dabei verwendete Ansatz basiert auf eine Kombination von Faltungsnetzwerken und rückgekoppelten Netzen. Außerdem wurden Adäquat dazu ein unabhängiges CNN mit 2 Layer

und ein weiteres RNN mit LSTM-Zelle als Vergleichsfaktoren mit hinzugezogen. Die Ergebnisse der Evaluierung der Arbeit ergaben eine Genauigkeit des CNN+LSTM von 93%, LSTM ungefähr 90% und das CNN ca. 83%. Die Arbeit zeigte aber ebenfalls, dass noch viel Potenzial nach oben vorhanden ist.

Im nächsten Kapitel soll das in der eben vorgestellten Arbeit [21], Modell, welches sich bereits als Klassifikationslösung für multidimensionale Daten als erfolgreich erweisen konnte, durch die Fähigkeit auch zukünftige Ereignisse zu prognostizieren, erweitert und präsentiert werden.

4 Systemmodell

Die Verarbeitung multidimensionaler Zeitreihen zählt zu den kompliziertesten Aufgabengebiete, weshalb diese Arbeit dazu beitragen soll, die anspruchsvolle Aufgabe zu bewältigen. In vorhergegangenen Arbeiten konnte mit diesem Modell bereits bewiesen werden, dass es sich als Lösung zur Klassifikation von Zeitreihen eignet. Im nächsten Schritt soll ein weiterer, noch nicht existierender Ansatz das Modell erweitern, mit dem Ziel, auch zukünftige Ereignisse vorherzusagen. Das Modell besteht aus zwei Hauptkomponenten: Die Faltungsschichten und die LSTM bzw. rückkoppelten Schichten. Jede Komponente dieser Schicht verfolgt unterschiedliche Ziele. Außerdem wurde das Modell um einen weiteren Layer den sog. *consolidate recurrent Layer* mit zusätzlicher Rückkopplung erweitert. Die Komponenten sind nachfolgend genauer beschrieben.

4.1 Notation

Eine Zeitreihe ist eine Sequenz von Datenpunkten zu einem Zeitpunkt mit D verschiedenen Sensorkanälen. Die Rohdaten x_{ti} der Sensoren eines Zeitpunktes t mit einem multidimensionalen Vektor i der als Tupel der Form $(ti, \overrightarrow{V}_i)$ beschrieben werden kann, wobei ti einem Zeitpunkt mit einem D-dimensionalen Vektor $(\overrightarrow{V}_i) \in \mathbb{R}^D$ von Messpunkten entspricht. Außerdem wurde ein Schiebefenster der Länge l auf die Sensordaten angewandt und mit einer Überlagerung von $l/2$ (50%) segmentiert. Jedes Segment entspricht einem Zustand, welcher einer Klasse zugeordnet werden kann. Jede Eingaben des Modells muss eindeutig bezeichnet werden. Die Reihenfolge für jeden Zustand wird daher als $P = \{p1, p2, p3, ..., pn\}$, mit $pi = \{xt1, xt3, ..., xtl\}$ und repräsentiert einen Zustand. Die Eingabewerte des Modells können mit der entsprechenden Form (N, L, D) dargestellt werden. In vorhergegangenen Arbeiten wurde ein Klassifikationsproblem mit $Y_i = \{y_1, y_2, y_2, ..., y_n\}$ als Ausgangswert betrachtet. Für die Vorhersage zukünftiger Ereignisse geht es um eine Prognose und es wird $Y_i = \{y_{1+v}, y_{2+v}, y_{3+v}, ..., y_{n+v}\}$ als Ausgangswert betrachtet, wobei v als Offset gesehen werden kann. Mit $v = 1$ wird versucht den nächsten Zustand vorherzusagen.

4.2 Das Ensemble CNN-LSTM DL-Modell

Bei Ensemble-Methoden geht es um die Möglichkeit Systeme bzw. Modelle miteinander zu kombinieren, die die besten Leistungen erbracht haben. Die Schwierigkeit bei Sensordaten, vor allem multidimensionalen Daten, ist ihr dynamisches Verhalten. Deshalb können Ensembles dementsprechend sehr nützlich für die Analyse komplexer Datensätze sein und zu einer verbesserten Performance führen. Die Faltungsnetzwerke erwiesen sich bereits mehrmals als eine sehr effektive Methode in der Bildverarbeitung und sollen sich daraus resultierend in dieser Kombination, die Aufgabe zur Extraktion von Merkmalen und übergeordnete Merkmale bewähren. Im Anschluss findet eine Überführung der Daten durch den *consolidate recurrent Layer* in das RNN statt. Die wiederkehrenden Schichten sollen die zeitliche Dynamik aus (diesen Feature Maps) den multidimensionalen Daten modellieren. Dazu sollen sie zusammen mit der LSTM-Zelle ausgestattet werden und als Langzeitgedächtnis fungieren, um zeitliche Muster bzw. temporale Abhängigkeiten zu erkennen. Das vollständige Modell kann der Abbildung 5 entnommen werden und wird nachfolgende noch genauer beschrieben.

Convolutional Neural Networks (CNN)
Für das Erlernen von Merkmalen und übergeordneten Merkmalen in multidimensionalen Daten sollen die Faltungsnetzwerke (*engl.* Convolutional Neural Networks) als Feature-Extraktoren fungieren und abstrakte Darstellungen der heterogenen Sensordaten in Feature Maps liefern. pi ist ein $L0 \times D0$ großer Tensor, mit $L0$ als Sequenz der Länge des Sliding Window l und $D0$ entspricht gleich der Anzahl Sensorkanäle D. Zu jedem Zeitpunkt t wird eine Matrix pi in die CNN-Architektur eingespeist. Im ersten Ansatz wird sich genauer auf die Sensorkanäle konzentriert. Hier sollen insbesondere Merkmale und Muster von Sequenzen extrahiert werden. Zur Umsetzung werden 2D Filter (kernel) k der Form (k,1) auf die Zeitreihen angewendet. Dadurch werden Feature Maps f erzeugt, die sich mit jedem weiteren CNN-Layer verdoppeln. So werden für jede Faltungsschicht des Modells weitere Filter der Sequenz unter Verwendung der ReLU als Aktivitätsfunktion gelernt. Ein anschließender Pooling Vorgang erfolgt dabei nicht. Der Ausgang der Faltungsschichten m hat die Form (L_m, f), wobei f die Anzahl der Feature Maps sind, die aus den Sensorkanälen $D0$, während der Faltungen gelernt wurden.

Long Short-Term Memory
Mit den Long Short Term Memory (LSTM) soll sich auf das Erkennen von temporalen Abhängigkeiten in den Sequenzen bzw. Merkmalen fokussiert werden. Darunter zeichnen sich LSTM Netzwerke besonders dadurch aus, dass sie auch weitreichende zeitliche Abhängigkeiten in Sequenzen erkennen können, ohne dabei den Einfluss von früheren Faktoren (auf spätere Berechnungen) zu verlieren. Die LSTM Architektur bekommt als Eingabewert die Matrix $D1$ der Größe $L_m \times f$, mit L_m als Vektor der Größe l des Sliding Window und m für die Anzahl der durchlaufenden CNN Layer, f ist ein Vektor der Größe mit der Anzahl der aus den unterschiedlichen Sensorkanälen erlernten Feature Maps, welche sich mit

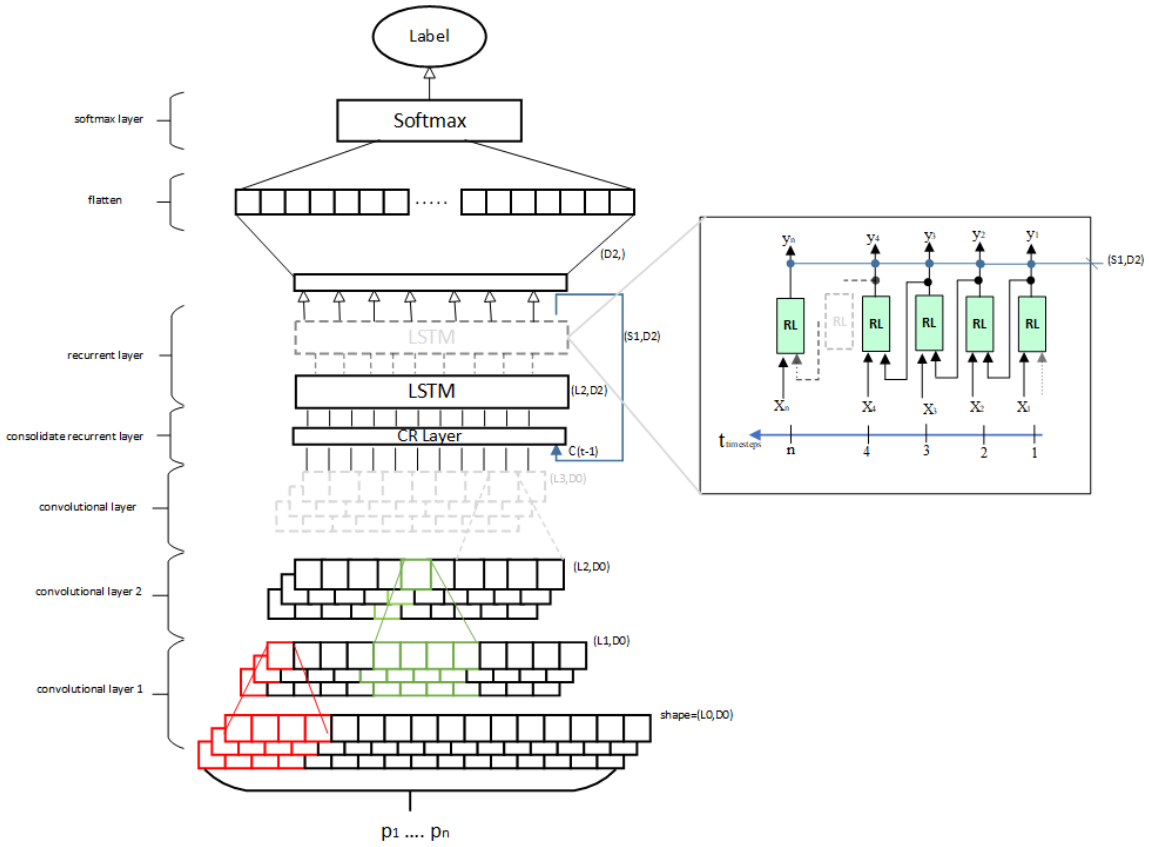

Abb. 5. *Das Deep Learning Architekturmodell*

jedem CNN Layer verdoppelten. Nach der Verarbeitung durch das RNN erhält man im letzten Zeitschritt (Timestep) am Ausgang einen Vektor der Größe u (Units), wobei u die Anzahl der rekurrenten Neuronen im rekurrenten Layer sind. Für die Ausgabe einer Schicht für einen ganzen Mini-Batch entspräche es bspw. eine Matrix der Größe $n \times u$, mit n der Anzahl der Datenpunkte im Mini-Batch zum Zeitpunkt t für jeden Datenpunkt im Mini-Batch. Der Output des LSTM zum Zeitpunkt t wird in der Abbildung als $D2$ beschrieben, wobei $D2 = u$ entspricht. Im Anschluss werden die Ergebnisse der Operationen zu jedem Zeitschritt dem *consolidate recurrent Layer* zurückgeführt und der Softmax-Regressionsschicht zugeführt und das entsprechende zukünftige Ereignis y_{i+1} prognostiziert und klassifiziert.

Consolidate Recurrent Layer

Der CR-Layer verfolgt das Ziel den Einfluss der Prognosegenauigkeit durch Einfluss der Vergangenheitswerte zu verbessern. Wie der Abbildung 5 zu entnehmen

ist, befindet sich der Layer zwischen dem CNN und dem LSTM Layer.
Bei der Überführung der Daten von dem CNN in das LSTM, sollen die Merkmale von den aktuellen Daten mit den zurückgeführten zusammengeführt werden. Die temporale Struktur der Sequenz soll somit durch die Einbeziehung neuer Gewichte, durch Vergangenheitswerte den Erfolg Zukünftige Ergebnisse vorherzusagen, erhöhen. Im Kapitel 6 wird es einen weiteren Ausblick geben wie der CR-Layer durch ein Sliding Window erweitert werden soll.
Zur Umsetzung muss die Matrix $D1$ der Ausgabe des CNN in $L2 \times f$ und $P1$ beschrieben, umgeformt werden. Dasselbe gilt für die Daten der Zurückführung C_{t-1} der Form $(S1, D2)$, welche in $(S1 \times D2)$ umgewandelt und mit $P2$ gekennzeichnet werden. $S1$ kann als die Anzahl der Timesteps beschrieben werden und $D2$ entspricht u, die Anzahl der rekurrenten Neuronen im einem RL. Der CR-Layer enthält außerdem zwei Sätze Gewichte: einen für $P1$ und einen Satz für $P2$. Die Gewichtsmatrizen werden als W_{P1} und W_{P2} bezeichnet. Die Ausgabe des CR-Layer lässt sich wie in Formel 9 berechnen (b ist der Bias-Term und $\varphi(\cdot)$ ist die Aktivierungsfunktion, in dieser Anwendung $tanh$ [2])

$$D3 = \Phi(P1_t \cdot W_{p1} + P2_t \cdot W_{p2} + b) \tag{8}$$

$$= \Phi([P1_t \; P2_{t-1}] \cdot W + b) \; mit \; W = \begin{bmatrix} W_{P1} \\ W_{P2} \end{bmatrix} \tag{9}$$

Damit das Resultat $D3$ der Gleichung 9 in das RNN eingespeist werden kann, muss es in die entsprechenden RNN Datenform umgewandelt werden. Dafür wird $D3$ in eine Matrix der Form $D1$ mit der Größe $L2 \times f$ umgeformt und entspricht der Form $(L2, f)$.

4.3 Modell Training

Die Implementation findet mit Python mittels TensorFlow als Framework zur Unterstützung von Deep Learning Modelle statt. Für die Durchführung von Datentransformationen und Vorverarbeitungen kommen die Frameworks Pandas und NumPy zum Einsatz. Für 2D und 3D Datenvisualisierung wird sowohl auf Matlab als auch auf TensorBoard zurückgegriffen, ebenfalls wird scikit-learn verwendet, weil es eine gute Unterstützung für weitere viele nützliche Funktionen für das maschinelle Lernen (Machine Learning) bereitstellt.
Die Grundlage der Arbeit basiert auf 3x leistungsstarke NVIDIA Tesla P100 mit 16 GB Arbeitsspeicher. Die Hardware sollte genug Performance aufbringen können, um auch mit einer großen Anzahl und Rechenintensiven Berechnungen die entsprechende benötigten Ressourcen bereitstellen zu können und somit den Anforderungen entsprechen.

[2] Vor allem bei Recurrent Neural Networks bevorzugen Wissenschaftler den Tangens hyperbolicus (tanh) als Aktivierungsfunktion anstatt die ReLU. Der dazugehörige Artikel ≫*Dropout Improves Recurrent Neural Networks for Handwriting Recognition*≪[26] von Vu Pham et al. Die Verwendung von ReLU finden aber dennoch ebenfalls sehr oft Anwendung, wie im Artikel ≫*A Simple Way to Initialize Recurrent Networks of Rectified Linear Units*≪von Quoc V.Le et al. [15] deutlich.

Das Modell wird anfänglich wie im Grundprojekt *„Klassifizierung Multidimensionaler Zeitreihen mithilfe von Deep Learning“* [21] (Meyer 2018:1-36) mit 2 Convolutional Layer und 1 Recurrent Layer trainiert. Im Anschluss soll die Anzahl der Convolutional Layer, sowie die Recurrent Layer erhöht und erneut evaluiert werden. Die Arbeiten in Kapitel 3 konnten erfolgreich demonstrieren, dass der Erfolg der Vorhersage mit der Steigerung der Layer optimiert werden kann. Außerdem soll weiterhin als konkurrierendes und vergleichbares Modell ein reines RNN mit LSTM-Zelle, als auch das Modell zur Klassifikation von Zeitreihen in der vorhergegangenen Arbeit [21] herangezogen werden.
Trainiert und getestet wird im Mini-Batch Verfahren. Die Gewichte werden mit zufälligen Werten initialisiert und die Optimierung der Parameter werden durch die Minimierung der Verlustfunktion mit Hilfe des Mini-Batch Gradientenverfahren und der Adam-Update Regel [14] mit einer Lernrate von *1e-3* justiert und durchgeführt. Außerdem wird beim Training auf die weit verbreitete Technik der Kreuzvalidierung (engl. Cross-Validation) zurückgegriffen, um sicherzustellen, dass das Modell auf die Daten gut verallgemeinert. Bei Cross-Validation wird der Testdatensatz solange zurückgehalten, bis die Qualität des Modells zuversichtlich eingeschätzt werden kann.

5 Der Datensatz

Der in dieser Arbeit verwendete Datensatz stammt aus dem Forschungsbereich *Human-Centered Computing* und basiert auf der Idee menschliches Verhalten im Zusammenhang mit ihrem sozialen Kontext zu verstehen. Eine der neuesten, anspruchsvollsten und ansprechendsten Anwendungen in diesem Rahmen besteht darin, mithilfe von am Körper getragenen Sensoren wie z.B. Smartphones, Bewegungen des menschlichen Körpers zu erfassen, um Kontextinformationen über menschliche Aktionen zu sammeln und einen Bezug herzustellen. In diesem Zusammenhang wurde von der Forschungsgruppe (Davide Anguita et al.2013:26-33) [2] eine *Human Activity Recognition* Datenbank aufgebaut und Aufzeichnungen von 30 Personen im Alter von 19 bis 48 Jahren aufgenommen und gespeichert. Der Datensatz wurde anschließend auf bekannte Online-Repositories veröffentlicht. Dazu gehören unter anderen das *UCI Machine Learning Repository*[7] sowie *Kaggle*[9] eine Plattformen welche verschiedene Wettbewerbe rund um das Maschinenlernen, Datamining und Vorhersagemodelle veranstaltet. Der Aufbau, Vorverarbeitungen des Datensatzes sind nachfolgend näher beschrieben.

5.1 Aufbau und Vorverarbeitung

Die Erkennung von menschlichen Aktivitäten wurde in den folgenden sechs Klassifikationen aufgeteilt: *walking, sitting, laying, standing, walking upstairs, walking downstairs*. Insgesamt handelt es sich dabei um ca. 10.000 aufgezeichnete Daten. Als Basis wurden dreiachsige, lineare Beschleunigungs- und Winkelgeschwindigkeitssignale verwendet, welche mittels Beschleunigungssensoren und Gyroskop erfasst worden sind. Die Abtastrate der Sensoren beträgt 50 Hz. Die

Daten wurden mit einem *sliding window* der festen Breite von 2.56 *sec* und mit einer Überlappung von 50% segmentiert [3]. Daraus ergab sich ein Vektor mit einer Fenstergröße von 128 Signalen ($2.56\ sec \times 50\ \text{Hz} = 128\ cycles$). Um sicherzustellen, dass das entwickelte Modell nicht nur aus bereits erlernten Daten korrekte Ergebnisse liefern kann, wurde der Datensatz zusätzlich in Trainings- und Testdaten im Verhältnis 70 / 30 unterteilt. Da die Sensoren (Beschleunigungssensor und Gyroskop) bereits dieselbe Abtastraten besitzen, konnte auf Vorverarbeitungsschritte wie die Synchronisierung der Datenströme verzichtet werden. Des weiteren wurden die Signale zur Rauschreduzierung mit einem Medianfilter und ein Butterworth-Tiefpassfilter der Ordnung 3 mit einer Grenzfrequenz von 20 Hz vorverarbeitet (Diese Grenzfrequenz ist in diesem Fall ausreichend für das Erfassen Menschliche Bewegungen, da sich 99% der Energie sich unter 15 Hz befinden [12].)

6 Weiteres Vorgehen für das Hauptprojekt

Das weitere Vorgehen im Hauptprojekt besteht zunächst aus der Umsetzung und Evaluierung der in Kapitel 4 beschriebenen Ansätze zur Vorhersage und Klassifikation zukünftiger Ereignisse, um anschließend die Ergebnisse und neu gesammelten Erkenntnisse in der Masterthesis entsprechend umzusetzen. Das vorgestellte System bzw. Modell soll anfänglich mit einer einfachen Sinus Funktion evaluiert und untersucht werden. Im weiteren Verlauf soll im Anschluss der Schwierigkeitsgrad durch Erhöhung der Komplexität bspw. durch eine quasiperiodische Bewegung zunehmend gesteigert. Nach der Analyse und Auswertung soll das Modell auf den eigentlichen Datensatz, die multidimensionalen Daten angewandt werden. Wie in der vorhergegangenen Arbeit wird bei der Evaluierung ein unabhängiges LSTM-Netz als Vergleichsfaktor hinzugezogen. Zu untersuchen gilt es unter anderen die Anzahl der trainierten Schichten zusammen mit der Größe der *Timesteps*, als auch die Grenzen in dem die temporale Struktur der Sequenz noch erfasst und erlernt werden kann. Außerdem gilt es in dem Bezug zu untersuchen, inwieweit Ereignisse unter Berücksichtigung der Komplexität der Zeitreihendaten prognostiziert und klassifiziert werden können. Zur Ermittlung der Grenzwerte der Timesteps[3] könnte ein *sliding window* unterstützen. Das Anwenden der *sliding window* Technik hat zu Folge, dass neben den aktuellen Werten auch die entsprechenden notwendigen Vergangenheitswerte beinhaltet sein müssen. Außerdem sollen für das weitere Vorgehen des Projekts die Trainingsdaten signifikant erhöht, als auch ein weiterer Datensatz hinzugezogen werden.

6.1 Risiken

Da es sich bei Sensordaten um einen komplexen Datensatz handelt, birgt es auch einige Risiken mit sich. Für das System kann es sich sehr schwierig gestalten,

[3] Die Untersuchung der Grenzwerte der Timesteps ist deshalb so wichtig, da man Gefahr laufen kann, durch zu viele Timesteps auf Probleme wie VG/EG(vanishing/exploding gradient) zu treffen.

die zugrunde liegenden Muster und Dynamik des Systems zu erkennen. Was letztendlich zu einer schlechten Verallgemeinerung des Fehlers und zu einer ungenauen Vorhersage führt. Das kann zum einen an der schlechten Qualität der Trainingsdaten liegen bspw. durch Fehler, Ausreißer, Rauschen oder an den zu wenig relevanten Merkmale in den Daten liegen. Zur Minimierung der Risiken gehört deshalb wie oben bereits erwähnt, die Beschaffung und Erhöhung neuer Trainingsdaten zu den wichtigste Schritten.

7 Fazit

In dieser Arbeit wurde ein Modell mit einem Ansatz zur Vorhersage und Klassifikation von Zeitreihen vorgestellt. Es wurde aufgezeigt, dass immer noch viele Probleme bei der Entwicklung von Lösungen insbesondere bei multidimensionalen Daten mit zeitlichen Abhängigkeiten existieren. Aus diesem Grund wurde in Kapitel 4 ein vielversprechender Lösungsweg vorgeschlagen, der die Entwicklung in diesen Bereich unterstützt und es möglich machen soll, DL-Modell auf komplexen Zeitreihendaten anzuwenden. Obwohl DL-Techniken bereits viele Erfolge zu verzeichnen haben, zeigte sich in Kapitel 3, dass ein Vergleich mit anderen Verfahren den DL-Techniken in vielen Bereichen ebenwürdig (bspw. Bayesschen Netze) sogar teilweise überlegen (SVM) sind. Es wurde aber auch deutlich, dass sie noch erhebliches Potenzial aufweisen, wie die Convolutional Netze bereits gezeigt haben. Abschließend gab es noch einen kurzen Ausblick über das weitere Vorgehen und die Risiken, die während der Umsetzung entstehen können.

Literatur

[1] Alam, M. R. ; Reaz, M. B. I. ; Ali, M. A. M.: SPEED: An Inhabitant Activity Prediction Algorithm for Smart Homes. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 42 (2012), July, Nr. 4, S. 985–990. http://dx.doi.org/10.1109/TSMCA.2011.2173568. – DOI 10.1109/TSMCA.2011.2173568. – ISSN 1083–4427

[2] Anguita, D. ; Ghio, A. ; Oneto, L. ; Parra, X. ; L Reyes-Ortiz, J. : A Public Domain Dataset for Human Activity Recognition using Smartphones. (2013), 01

[3] BenAbdelkader, C. ; Cutler, R. ; Davis, L. : Stride and cadence as a biometric in automatic person identification and verification. In: *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, 2002, S. 372–377

[4] Dahl, G. E. ; Yu, D. ; Deng, L. ; Acero, A. : Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. In: *IEEE Transactions on Audio, Speech, and Language Processing* 20 (2012), Jan, Nr. 1, S. 30–42. http://dx.doi.org/10.1109/TASL.2011.2134090. – DOI 10.1109/TASL.2011.2134090. – ISSN 1558–7916

[5] datennerd.de: *lstm-intro*. http://datennerd.de/post/lstm-intro/. Version: 2018. – Zuletzt besucht am 15.08.2018

[6] Deng, L. ; Platt, J. : Ensemble deep learning for speech recognition. (2014), 01, S. 1915–1919

[7] Dheeru, D. ; Karra Taniskidou, E. : *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml. Version: 2017

[8] Eckmann, J.-P. ; Kamphorst, S. O. ; Ruelle, D. : Recurrence Plots of Dynamical Systems. In: *EPL (Europhysics Letters)* 4 (1987), Nr. 9, 973. http://stacks.iop.org/0295-5075/4/i=9/a=004

[9] Goldbloom CEO, A. ; Hamner CTO, B. ; Moser Chief Architect, J. : *KAGGLE Repository*. https://www.kaggle.com/. Version: 2018

[10] Guo, Y. ; Wu, Z. ; Ji, Y. : A Hybrid Deep Representation Learning Model for Time Series Classification and Prediction. In: *2017 3rd International Conference on Big Data Computing and Communications (BIGCOM)*, 2017, S. 226–231

[11] Hochreiter, S. ; Schmidhuber, J. : Long Short-term Memory. In: *Neural Comput.* 9 (1997), Nov., Nr. 9, 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735. – DOI 10.1162/neco.1997.9.8.1735. – ISSN 0899–7667

[12] Karantonis, D. M. ; Narayanan, M. R. ; Mathie, M. ; Lovell, N. H. ; Celler, B. G.: Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. In: *IEEE Transactions on Information Technology in Biomedicine* 10 (2006), Jan, Nr. 1, S. 156–167. http://dx.doi.org/10.1109/TITB.2005.856864. – DOI 10.1109/TITB.2005.856864. – ISSN 1089–7771

[13] Kim, Y. ; An, J. ; Lee, M. ; Lee, Y. : An Activity-Embedding Approach for Next-Activity Prediction in a Multi-User Smart Space. In: *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2017, S. 1–6

[14] Kingma, D. P. ; Ba, J. : Adam: A Method for Stochastic Optimization. In: *CoRR* abs/1412.6980 (2014). http://arxiv.org/abs/1412.6980

[15] Le, Q. V. ; Jaitly, N. ; Hinton, G. E.: A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. In: *CoRR* abs/1504.00941 (2015). http://arxiv.org/abs/1504.00941

[16] LeCun, Y. ; Bottou, L. ; Bengio, Y. ; Haffner, P. : Gradient-Based Learning Applied to Document Recognition. In: *Proceedings of the IEEE* 86 (1998), November, Nr. 11, S. 2278–2324

[17] Lee, H. ; Grosse, R. ; Ranganath, R. ; Ng, A. Y.: Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM (ICML '09). – ISBN 978–1–60558–516–1, 609–616

[18] Liu, B. ; Chen, H. ; Sharma, A. ; Jiang, G. ; Xiong, H. : Modeling heterogeneous time series dynamics to profile big sensor data in complex physical systems. In: *2013 IEEE International Conference on Big Data*, 2013, S. 631–638

[19] Meisel, A. : *ConvolutionalNeuralNetworks (CNN)*. Vorlesungsskript 'Modellierung Dynamischer Systeme', 2017

[20] Meisel, A. : *Recurrent-neural-Networks*. Vorlesungsskript 'Modellierung Dynamischer Systeme', 2017

[21] Meyer, M. : Klassifizierung Multidimensionaler Zeitreihen mithilfe von Deep Learning. In: *2018 Master Informatik Grundprojekt (minf'18)*, 2018, S. 1–36

[22] Meyer, M. : Klassifzierung Multidimensionaler Zeitreihen mithilfe von Deep Learning. In: *2015 Master Informatik Seminar (minf'15)*, 1-12

[23] Nakano, K. ; Chakraborty, B. : Effect of dynamic feature for human activity recognition using smartphone sensors. In: *2017 IEEE 8th International Conference on Awareness Science and Technology (iCAST)*, 2017, S. 539–543

[24] Nazerfard, E. ; Cook, D. : CRAFFT: An Activity Prediction Model based on Bayesian Networks. 6 (2014), 01

[25] Olah, C. : *Understanding LSTM Networks*. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`. Version: 2018. – Zuletzt besucht am 30.06.2018

[26] Pham, V. ; Kermorvant, C. ; Louradour, J. : Dropout improves Recurrent Neural Networks for Handwriting Recognition. In: *CoRR* abs/1312.4569 (2013). `http://arxiv.org/abs/1312.4569`

[27] Sak, H. ; Senior, A. ; Beaufays, F. : Long short-term memory recurrent neural network architectures for large scale acoustic modeling. (2014), 01, S. 338–342

[28] Wurm, C. : *Neuronale Netze und Tiefe Architekturen*. `https://user.phil.hhu.de/~cwurm/wp-content/uploads/2018/01/deep_architectures.pdf`. Version: 2018. – Zuletzt besucht am 30.06.2018