Simulation in Produktion
und Logistik 2019
Matthias Putz & Andreas Schlegel (Hrsg.)
Wissenschaftliche Scripten, Auerbach 2019

Deep-Learning-basierte Prognose von Stromverbrauch für die hybride Simulation

Deep Learning based Prediction of Energy Consumption for Hybrid Simulation

Benjamin Wörrlein, Sören Bergmann, Niclas Feldkamp, Steffen Straßburger,
TU Ilmenau, Ilmenau (Germany), {benjamin.woerrlein, soeren.bergmann,
niclas.feldkamp, steffen.strassburger}@tu-ilmenau.de

Abstract: Modern production facilities need to prepare for changing market conditions within the energy market due to ongoing implementation of governmental policies. This results in higher volatility of the availability of energy and therefore energy costs. If a simulation model of a machinery model can estimate its own future consumption, and according time frames for said consumption, this information could be used for optimized scheduling of energy consuming jobs. This would result in lower procurement costs. To make said estimation about the dynamic behaviour of jobs, methods of time series prediction tend to be applied. Here a proposal is made to apply a Hybrid System Model incorporating a recurrent neural network (RNN)-Encoder-Decoder-Architecture, which returns a discrete times series when a behavioural sequence (such as an NC-Code) has been put into a neural net model of the respective machinery. Those discrete time series reflect the machines energy consumption for each job that it has been operated on. This neural net, if weighted and called, emits the length value of a job and an according time series which displays the quasi-continuous time consumption of said job. Such generative models combined with classic simulation paradigm qualify as potent applications of hybrid simulation approaches.

1 Einführung

Aufgabe der Simulation ist es, durch Nachbilden eines Systems Erkenntnisse über dessen Verhalten zu gewinnen. Das Verhalten eines Systems wird typischerweise anhand der dynamischen Veränderungen seines Systemzustands über ein diskret-ereignisgesteuertes oder kontinuierliches Simulationsparadigma in einem Simulationsmodell beschrieben. Kommen bei der Erstellung des Simulationsmodells mehrere verschiedene Modellierungsansätze (ggf. auch nur mehrere Weltsichten innerhalb eines der genannten Paradigmen) zum Einsatz, so spricht man von *hybrider Simulation* (Mustafee et al. 2017). Die Kombination von diskret-ereignisgesteuerter

und kontinuierlicher Simulation als eine Spielart der hybriden Simulation wird traditionell auch als „kombinierte Simulation“ bezeichnet (Eldabi et al. 2016).

Die Untersuchung von Fragestellungen der Energieeffizienz innerhalb der Simulation ist mittlerweile ein verbreiteter Untersuchungsansatz. Häufig basieren existierende Arbeiten auf der Berücksichtigung des Stromverbrauchs von Ressourcen (Maschinen, Öfen, …) anhand messtechnisch erfasster Betriebszustände, die über einen definierten Zeitraum als konstant angesehen werden (Haag 2013; Thiede 2012).

Der über einen Zeitraum gemittelte Stromverbrauch von Ressourcen wird hierbei einem Ressourcenzustand zugeordnet und kann dann statusbasiert mit ereignisdiskreten Simulationsansätzen abgebildet und analysiert werden.

Kritisch ist hierbei zu hinterfragen, für welche Anwendungsfälle diese quasi-statischen Betriebszustände genügend Realitätsnähe liefern. Zur Ermittlung und Glättung von Lastspitzen vieler Ressourcen bietet ein derartiger Ansatz keine ausreichende Realitätsnähe.

Ein Lösungsansatz hierfür wird in Römer et al. (2018) vorgestellt. Er basiert auf der Grundidee der kombinierten Simulation, der z. B. auch in Peter und Wenzel (2015) vorgeschlagen wird. Während der Produktions- und Logistikanteil des Modells klassisch mit ereignisdiskreter Simulation abgebildet wird, wird in Römer et al. (2018) für den Stromverbrauch der System-Dynamics-Ansatz angewendet. Hiermit können die Zeitreihen der real gemessenen Stromverbräuche hochaufgelöst in der Simulation reproduziert werden. Dies bietet den Vorteil eines hochaufgelösten Gesamtbilds des Stromverbrauchs der Produktion.

Nachteilig ist hierbei jedoch, dass nur der Stromverbrauch real gemessener Aufträge wiedergegeben werden kann. Ein Stromverbrauch unbekannter Auftragstypen kann nicht ohne vorherige Messung am Realsystem prognostiziert werden. Weiterhin lassen sich mit dem in Römer et al. (2018) erläuterten Ansatz keine Ursache-Wirkzusammenhänge zwischen Steuerparametern (z.B. halber Vorschub, langsamere Hochheizphase) und dem resultierenden Stromverbrauch darstellen.

Für den vorliegenden Beitrag soll daher untersucht werden, ob es alternative Möglichkeiten zur hochaufgelösten Prognose des Stromverbrauchs gibt, die die genannten Nachteile überwinden können. Der Schwerpunkt der Untersuchungen liegt hierbei auf dem Gebiet des maschinellen Lernens bzw. des *Deep Learnings*, auf Basis dessen ein vielversprechendes Verfahren vorgeschlagen wird.

Ziel des vorgeschlagenen Verfahrens ist es, mittels entsprechend trainierten künstlichen neuronalen Netzen (KNN) Zeitreihen für den Stromverbrauch unbekannter Aufträge prognostizieren zu können. Die Grundidee hierbei ist es, ein KNN mit relevanten Steuerungsinformationen (hier: NC-Codes der Fertigungsaufträge einer Werkzeugmaschine) und den zu diesen Aufträgen gemessenen hochaufgelösten Zeitreihen der Stromverbräuche zu trainieren.

Perspektivisch kann das KNN dann zu beliebigen, ggf. auch unbekannten Aufträgen mit abweichenden NC-Codes eine Zeitreihe des erwarteten Stromverbrauchs prognostizieren. Diese ließen sich dann in hybriden Simulationen des gesamten Produktionssystems verwenden.

Der Beitrag stellt ein Lösungskonzept für das skizzierte Verfahren sowie eine prototypische Implementierung und Validierung vor.

Der Beitrag ist hierzu wie folgt gegliedert: Kapitel 2 führt in die benötigten theoretischen Grundlagen hybrider Simulation und maschineller Lernverfahren ein. Insbesondere werden Notwendigkeit und Grundidee einer *Deep-Learning*-Methode, welche Sequenzen unterschiedlicher Länge und Taktzeiten aufeinander abbilden kann, gesondert vorgestellt. Anschließend wird die prinzipielle Funktionalität von klassischen KNN zu zeitsensitiven, rekurrenten neuronalen Netzen (RNN) abgegrenzt. Aufbauend auf dieser Einführung in RNN wird auf *Sequence to Sequence* (Seq2Seq)- Modelle, insbesondere RNN-Encoder-Decoder-Architekturen (RNN-ED) eingegangen, welche eine Zuordnung unterschiedlich langer Sequenzen zueinander erlauben.

Hierauf aufbauend wird in Kapitel 3 ein Konzeptvorschlag der Gesamtarchitektur mit seinen Ein- und Ausgangssequenzen entwickelt. Das Konzept wird im Kontext eines Fertigungsauftrages (FA) an einer Werkzeugmaschine (WZM) erstellt. Als eingehende Zustandsfolge wird sich hier des NC-Codes bedient. Hierfür muss eine Methode definiert werden, die eine Eingangssequenz von Symbolen, welche nicht in der Ausgangssequenz vorkommen, in eine für KNN verständliche Form überführt. Dieser als Vektorisierung bezeichnete Vorgang wird anhand eines Word2Vec-*Tokenizers* durchgeführt und gibt so eine vektorisierte Form des NC-Codes aus. Anschließend wird der vektorisierte NC-Code auf aufgenommene Zeitreihen der Wirkleistung einer WZM trainiert.

Eine prototypische Umsetzung des Konzepts erfolgt in Kapitel 4. Dieses wird in R mit der API *Keras* und dem Backend *Tensorflow* umgesetzt. Nach erfolgreicher Trainingsphase erfolgt eine Vorstellung der Ergebnisse. Dies geschieht anhand einer Gegenüberstellung der Zeitreihen des Trainingsdatensatzes und der erzeugten Zeitreihe unter Zuhilfenahme eines Verfahrens zur Glättung von Zeitreihen. Hier sollen insbesondere die Trendkomponenten der beiden Zeitreihengruppen einander gegenübergestellt werden. Weiter wird die postulierte Seq2Seq-Architektur als *Timeout*-Funktion einer diskret-ereignisgesteuerten Simulation im Sinne einer hybriden Modellierung verwendet.

Eine kritische Betrachtung der Ergebnisse erfolgt in Kapitel 5.

2 Grundlage der vorgeschlagenen Methode

Die hier vorgeschlagene hybride Methode zeichnet sich dadurch aus, dass sie bekannte, asynchrone Zustands- bzw. Parameterverläufe, als eine Form von apriorischem Wissen, zur Modellierung von Systemverläufen ermöglicht. Weiter macht sie sich dazu Methoden des maschinellen Lernens zu Nutze. Vorteilig bei der Verwendung von Algorithmen des maschinellen Lernens ist es, dass diese sich, anschließend an die Modellierungsphase, anhand von Realdaten selbst parametrieren.

2.1 Notwendigkeit hybrider Methoden

Die grundsätzliche Limitation ereignisdiskreter Simulationsansätze besteht darin, dass Zustandsänderungen zwischen zwei Ereignissen nicht abbildbar sind. Für eine Aktivität, d. h. die Zeitspanne zwischen zwei Ereignissen, kann jedoch die Notwendigkeit bzw. der Wunsch bestehen, einen Zustandsverlauf eines zur Aktivität gehörenden Merkmals (z. B. den Verlauf des Stromverbrauchs während der Bearbeitung) zu beschreiben. Hierfür könnte z. B. aus einem internen (in der

ereignisdiskreten Modellierung nicht betrachteten) Zustandsverlauf heraus zu bestimmten Taktzeiten eine Folge von Ausgaben erzeugt werden, welche das zeitliche Verhalten dieses Merkmals widerspiegeln (Lunze 2017).

Gerade aber der Zustandsverlauf eines technologischen Systems kann von einer Vielzahl äußerer, asynchroner Einflüsse bedingt werden. Dies bedeutet, dass Merkmalsbeschreibungen Y, die ab dem Start einer Aktivität ausgegeben werden sollen, in Relation zu etwaigen Einflussgrößen X verstanden werden müssen.

Es fehlt eine Methode, welche es erlaubt, Zielmerkmalsverläufe $Y_{T'}$ aus asynchronen, aber bekannten Parameter- bzw. Zustandsverläufen X_T abzubilden. Gerade für die Abbildung komplexer Merkmalsverläufe aufeinander bieten sich Methoden des maschinellen Lernens an, da diese einerseits Zusammenhänge zwischen Merkmalsverläufen selbstständig erkennen und andererseits lernen, diese aufeinander abzubilden. Weiter besitzen Methoden des maschinellen Lernens erst nach erfolgreicher Lernphase einen parametrierten Systemzustand, analog zum Systemzustand innerhalb einer Simulation, und setzen daher keinen bekannten Zustandsverlauf während der Aktivität voraus. Diese Eigenschaften machen Methoden des maschinellen Lernens perspektivisch zu einem potenten Verfahren innerhalb eines *Hybrid System Model* nach Mustafee et al. (2017).

2.2 Rekurrente Netze und Encoder-Decoder Architekturen

KNN werden zur Identifikation von Zusammenhängen in komplexen Datenstrukturen verwendet. Hierfür nehmen Aufnahmeschichten einer Netzarchitektur die Daten auf und leiten diese als abstrahierte Information durch die verdeckten Schichten eines KNN. Verdeckte Schichten bestehen wiederum aus verdeckten Einheiten, den eigentlichen Neuronen. Diese Neuronen sind sich selbst parametrierende Einheiten. Je mehr versteckte Schichten ein KNN hat, desto höher kann der Abstraktionsgrad der aufgenommenen Information sein. Besitzt ein KNN mehr als eine verdeckte Schicht, so kann es Abstraktionen, die in einer Schicht gewonnen wurden, in einer weiteren Schicht verknüpfen, und somit eine komplexere Abstraktion, mit jeder hinzugenommenen Schicht, erzeugen. Diese tiefe Staffelung von neuronalen Schichten wird als *Deep Learning* bezeichnet (Goodfellow et al. 2017).

Ändert sich ein Muster über die Zeit, wird diese zeitliche Abfolge des Musters als Sequenz verstanden. Damit ein KNN zeitliche Muster verarbeiten kann, müssen rekurrente Verbindungen in der Netztopologie vorhanden sein, welche eine Rückkopplung abstrahierten Wissens zulassen (Brause 1995; Zell 2003). Solche rückgekoppelten bzw. rekurrenten neuronalen Netze (RNN) eignen sich besonders für Daten, welche in sequentieller Form vorliegen (Goodfellow et al. 2017).

Für die zeitsensitive Verarbeitung von Sequenzen muss weiter eine neuronale Zelle bereitgestellt werden, welche einerseits ihren eigenen Zustand behält und diesen weitergeben kann, andererseits Zugriff auf Nachfolgezustände besitzt und diese einordnen kann (Zeng et al. 2017). Die Anforderungen an eine solche neuronale Zelle mit *Gedächtnis* werden durch neuronale *Long Short Term Memory* (LSTM)- Zellen (Hochreiter und Schmidhuber 1997), und deren vereinfachte Form *Gated Recurrent Unit* (GRU) (Chung et al. 2014) erfüllt.

Handelt es sich bei Ein- und Ausgängen eines KNN um Sequenzen, werden diese als *Sequence to Sequence* (Seq2Seq) Architekturen bezeichnet. Durch die

Aufnahmeschicht eines KNN findet eine Codierung der Eingangssequenz statt. Wird die Eingangssequenz als Abstraktion, in eine spezifische neuronale Schicht codiert, so ist dies ein Encoder. Wird eine Sequenz aus der Abstraktion einer neuronalen Schicht heraus generiert, so wird dieser Teil einer Netzwerktopologie als Decoder bezeichnet (Goodfellow et al. 2017).

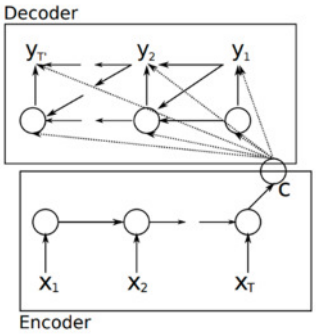

Abbildung 1: *Encoder-Decoder Architektur mit den Sequenzen X_T und $Y_{T'}$ unterschiedlicher Länge T≠T' und dem Kontext C (Cho et al., 2014)*

Ist es Aufgabe eines Seq2Seq-Modells, Sequenzen unterschiedlicher Länge auf einander abzubilden, werden solche Strukturen allgemein als Encoder-Decoder-Netzwerke bezeichnet (Zeng et al. 2017). Sollen Sequenzen unterschiedlicher Länge und unterschiedlicher Attribute aufeinander abgebildet werden, so müssen diese um eine zusätzliche Beschreibungsart, einen *Kontext* (vgl. Kontext **Abbildung 1**), erweitert werden. Der Kontext kann als Zwischenschicht zwischen den verdeckten Schichten eines Encoders und Decoders verstanden werden (Cho et al. 2014).

Der Kontext C ist ein Vektor einer Sequenz, welche die Einheiten der verdeckten Schicht des Encoders aufnimmt und diese auf den Decoder abbilden soll (Goodfellow et al. 2017). Der Vektor selbst lässt sich anhand einer verdeckten Schicht beschreiben (Goodfellow et al. 2017). So wird bei der RNN-ED-Architektur der Kontext C als ein Resultat der finalen verdeckten Schicht des Encoders mit der Zustandssequenz X_T, beschrieben. Da der Encoder in der Trainingsphase seinen finalen verdeckten Zustand weitergibt, muss die ganze Sequenz X_T durchlaufen worden sein (siehe **Abbildung 1**).

Weiterführende Erläuterungen zum hier verwendeten Encoder-Decoder können (Sutskever et al. 2014; Cho et al. 2014; Goodfellow et al. 2017) entnommen werden.

3 Konzept

Zur konzeptuellen Überprüfung wird vorgeschlagen, als Ein- und Ausgangsequenz zwei Sequenzen zu verwenden, welche derselben zeitlich-räumlichen Entität angehören. Als Merkmal einer zeitlich-räumlichen Entität wird hier von einem Prozess ausgegangen, welcher am selben Ort und zur selben Zeit stattfindet. Hierfür wurde das technologische Verfahren des Spanens eines Fertigungsauftrages (FA) auf einer Werkzeugmaschine (WZM) identifiziert.

Ein NC-Code beschreibt eine Abfolge notwendiger technologischer Prozesse bis zur Beendigung eines FA und kann somit als eine konkrete Beschreibung einer dem Prozess zugrundeliegender Zustandsfolge verstanden werden. Der NC-Code eines FA stellt hier die Eingangssequenz X_T eines RNN-ED dar (siehe Abbildung 2). Der NC-Code bestimmt also maßgeblich das Verhalten innerhalb des Spanraums einer WZM. Weiter gilt ein FA erst als abgeschlossen, wenn der NC-Code einmal komplett durchlaufen wurde.

Der NC-Code muss hierfür erst in eine Abfolge numerischer Werte übersetzt werden, welche die Struktur der Eingangsfolge beibehält. Dies wird durch einen sog. *Tokenizer* realisiert. Ein *Tokenizer* weist jedem Symbol bzw. jeder Menge von Symbolen, welche im NC-Code vorhanden sind, einen numerischen Wert bspw. anhand der Häufigkeit des betreffenden Symbols zu.

$$[\dots G\ 00, X0\ Y0\ Z0, \dots] \xrightarrow{Tokenizer} [\dots 1\ 2\ 3\ 4\ 5\ \ \dots]$$

Weiter entfernt der *Tokenizer* Symbole bzw. Symbolbeschreibungen, welchen ein geringer Informationsgehalt, wie zum Beispiel Kommata und Groß/-Kleinschreibung, unterstellt wird. Eine Möglichkeit, die Dimensionen des Vektorraums zu begrenzen, ist es, dem *Tokenizer* eine Anzahl maximal abbildbarer Symbolmengen, d. h. Wörter, anzuzeigen. Im Anwendungsfall wurde ein *Word2Vec-Tokenizer* verwendet, welcher nur die häufigsten Symbole und Symbolmengen in den Vektor übernimmt.

Basis der Ausgangszeitreihen $Y_{T'}$ quasi- kontinuierlicher Ausgabewerte ist der Stromverbrauch desselben FA bei Durchlauf des NC-Codes (siehe **Abbildung 2**). Der zeitliche Stromverbrauch soll konkret Aufschluss darüber geben, wann mit wieviel Verbrauch gerechnet werden muss, sobald über die Einsteuerung eines FA entschieden werden muss. Die Zeitreihen wurden unter Feldbedingungen aufgenommen und verfügen über dieselbe Taktung.

In der Trainingsphase wird ein ungewichtetes KNN, bestehend aus einem RNN-ED, anhand der Ein- und Ausgangssequenzen $\langle X_T | Y_{T'} \rangle$ parametriert.

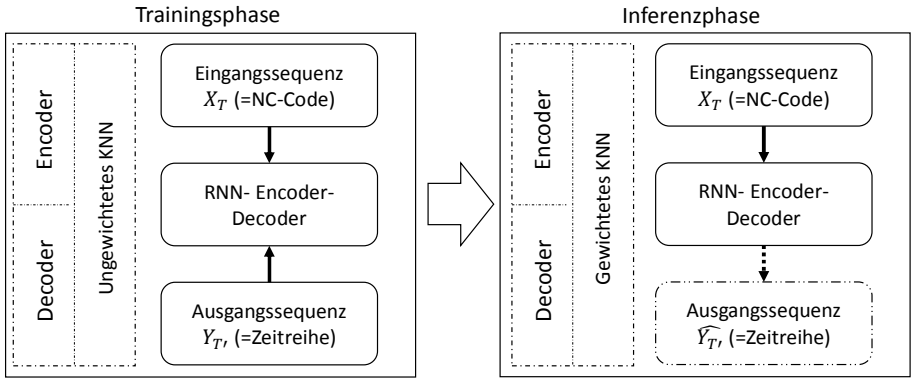

Abbildung 2: *Bestandteile einer RNN-Encoder-Decoder Topologie für Sequenzen unterschiedlicher Länge und Symbolik während der Trainings- und Inferenzphase*

Aufgabe der Inferenzphase ist es, ein aussagekräftiges Stromverbrauchsprofil $\widehat{Y_{T'}}$ explizit quasi- kontinuierlich über die Zeit auszugegeben (siehe **Abbildung 2**).

Die so postulierte Methode der Erzeugung quasi-kontinuierlicher Zeitreihen stellt, wenn sie in Kombination mit den Modellierungsmöglichkeiten diskret-ereignisgesteuerter Simulationssysteme verwendet wird, eine Methode der hybriden Simulation dar.

4 Testszenario und Ergebnisse

Zur Umsetzung des *Tokenizers* und des Encoder-Decoder RNN wird *Tensorflow* verwendet, welches hier in der Programmiersprache *R*, mit der IDE *RStudio* (RStudio Team 2015) bedient wird. Um auf *Tensorflow* in der IDE *RStudio* zugreifen zu können, wurde auf die API *Keras* zurückgegriffen. Die API *Keras* (Chollet 2015) als Schnittstelle zu *Tensorflow* wurde ausgewählt, da diese Netzarchitekturen auf höherem Abstraktionsgrad verständlich darstellt und somit ein hohes Maß an Übersichtlichkeit bereitstellt. Für die ereignisdiskrete Modellierung wird das R-Packet *rSimmer* (Lawson und Leemis 2015) genutzt.

Als Trainingsdatensätze der Ausgangssequenz werden 3 reale Messreihen der Wirkleistung einer WZM verwendet. Diese wurden unter Feldbedingungen bei Bearbeitung eines FA aufgenommen. Die Ausgangssequenzen stellen Zeitreihenaufnahmen der Wirkleistung mit dem äquidistanten Abstand von 2 Sekunden dar. Die Eingangssequenz stellt der NC-Code des FA dar. Bei diesem wird ein *Tokenizer* eingesetzt, der die vektorisierte Eingangssequenz erzeugt.

Der RNN-ED nimmt den vektorisierten NC-Code als Anfangssequenz auf und trainiert diesen anschließend auf die dem FA zugehörigen Zeitreihen des Stromverbrauchs. Das trainierte Netz und seine Gewichtungen werden anschließend gespeichert. Um das gewichtete RNN-ED nun in der Inferenz zu nutzen, wird wieder der vektorisierte NC-Code des FA in die Encoder-Schicht eingegeben. Dies führt zur Ausgabe einer Zeitreihe des Stromverbrauchs aus einem trainierten Encoder-Decoder-RNN heraus.

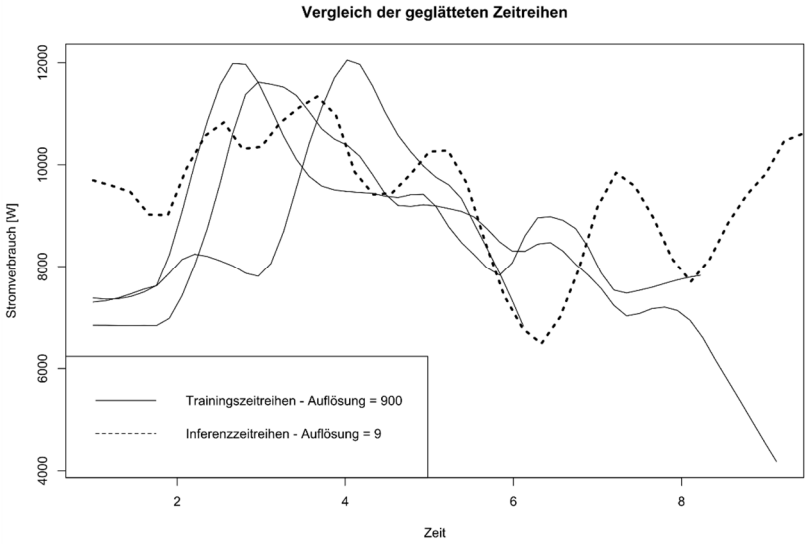

Abbildung 3: *Trend der Trainingszeitreihen* $\boldsymbol{Y_{T,}}$ *und Inferenzzeitreihe* $\widehat{\boldsymbol{Y_{T,}}}$ *bei Durchlauf desselben NC-Codes mit einer Auflösung von 900 bzw. 9*

In **Abbildung 3** werden die Trainings- und Inferenzzeitreihen des Stromverbrauchs geglättet einander gegenübergestellt. Eine geglättete Zeitreihe, auch Trend genannt, wird verwendet, um ein generelles Verhalten einer Zeitreihe sichtbar zu machen, wobei die Auflösung eines Trends den Grad der beabsichtigten Glättung bestimmt. Dies ist notwendig, da hier ohne eine solche Visualisierungshilfe kein Verhalten im Verlauf der Zeitreihe auszumachen ist.

Werden nun die Zeitreihen in **Abbildung 3** betrachtet, so fällt auf, dass diese ein stochastisches Verhalten über ihren gesamten Verlauf aufweisen. Dies ist im Wesentlichen auf Variationen im technologischen Fertigungsprozess des Zerspanens, wie im NC-Code beschrieben, zurückzuführen. Durch eine Anpassung der Auflösung der Trainings- und Inferenzzeitreihen ist es möglich, deren generelles Verhalten vergleichbar zu machen. So ist mit der in **Abbildung 3** gewählten Auflösung ein Verhalten der Inferenzzeitreihe sichtbar, welches grundsätzlich auf ein Funktionieren des vorgeschlagenen Verfahrens des RNN-Encoder-Decoders hindeutet. Weitere Tests zur Validierung des Verfahrens, abseits einer subjektiven, optischen Gegenüberstellung, sind jedoch notwendig.

Weiter wird die Ausgabe aus dem Encoder-Decoder als Ausgabe- und Zeitfortschrittsfunktion in einer Methode der hybriden Simulation (vgl. *Hybrid Systems Model* in Mustafee et al. 2017) verwendet. Hierbei wird der Merkmalsverlauf (die „Trajektorie“) des Stromverbrauchs innerhalb des Bedienraums der WZM mit dem beschriebenen Verfahren charakterisiert.

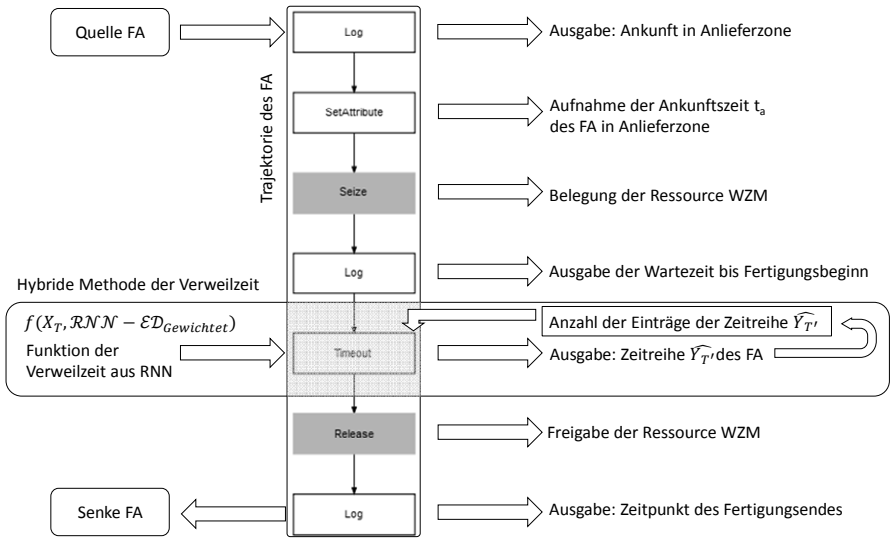

Abbildung 4: *Zustandsraumbeschreibung eines FA in rSimmer*

Hierzu wird in der Simulation bei Belegung der WZM durch einen FA die Ausgabefunktion des RNN-ED aufgerufen (siehe Abb. 4). Die Erzeugung der Verweilzeit eines FA in der WZM aus den NC-Codes wird also durch die Zuordnung zur *Timeout*-Funktion der Trajektorie erreicht. Dieses ordnet dem FA beim Durchlaufen des Bedienraums eine Zeitreihe zu, welche erzeugt wird, sobald der FA *Timeout* erreicht. Dies geschieht nachdem der FA die Ressource WZM blockiert hat. Diese bleibt nun solange blockiert, bis die Verweilzeit im *Timeout* erreicht ist. Anschließend gibt die Trajektorie des FA die Ressource WZM wieder frei und diese kann vom nächsten FA belegt werden. Bei einem Simulationsdurchlauf mit der beschriebenen hybriden Methode wird die gleiche Zeitreihe, wie sie bei alleiniger Inferenz erzeugt wird (vgl. $\widehat{Y_{T'}}$ in **Abbildung 3**), ausgegeben.

5 Zusammenfassung und Ausblick

Die prinzipielle Funktionsfähigkeit des beschriebenen Lösungsansatzes konnte im Rahmen des Testszenarios bestätigt werden. Die generierten Zeitreihen sind jedoch noch kritisch zu hinterfragen und in weiteren Forschungsarbeiten zu validieren. Hierzu fehlt es einerseits noch an Evaluierungsmethoden für generative Modelle des maschinellen Lernens, um die erzeugten Zeitreiheneinträge auf Sinnhaftigkeit ihrer Einträge zu überprüfen. Dies geschieht zum momentanen Zeitpunkt über die Betrachtung und den Abgleich der erzeugten Zeitreihen durch einen Experten des Anwendungsfalles durch optische Sichtung (Goodfellow et al. 2017) wie in **Abbildung 3**. Andererseits vermochte es das trainierte Modell noch nicht, Zeitreihen stochastischer Länge auszugeben und somit das zeitlich-dynamische Verhalten eines FA wiederzugeben. Ein Grund hierfür kann in der unzureichenden Qualität und insbesondere Quantität der Eingangsdaten liegen. Eine generelle Eignung der verwendeten Methoden wird aber dennoch unterstellt, da das System auf hohem

Abstraktionsniveau ein plausibles Lernverhalten aufweist, auch wenn der Inhalt des Erlernten noch kritisch zu betrachten ist.

Für eine abschließende Bewertung der verwendeten Methoden ist es angeraten, die qualitative und quantitative Datenbasis des RNN-Encoder-Decoder Modells zu erhöhen. Weiterhin muss dem Lösungsvorschlag eine geeignete Evaluierungsmethode angefügt werden. Da diese sich nicht an einer (nicht existenten) Ideal-Zeitreihe orientieren kann, wird vorgeschlagen, diese wieder in die Eingangssequenz, hier den NC-Code, zurückzuübersetzen und die Differenz zwischen der Eingangssequenz und der aus der Ausgangssequenz generierten Eingangssequenz als Evaluierungsmethode zu verwenden. Dies liegt in der Annahme begründet, dass der NC-Code als eine Form formaler Sprache (vgl. Chomsky Hierarchien der Sprache, Lunze 2017) interpretiert werden kann, für welche es schon Ansätze einer Evaluierung gibt (vgl. BLEU-Score, Cho et al. 2014). Die Weiterentwicklung des dargestellten maschinellen Lernverfahrens und dessen Nutzung für die hybride Simulation sind Gegenstand aktuell laufender Forschung. Bei erfolgreicher Etablierung und Validierung der Methode könnte eine Lösung entstehen, die auch für NC-Codes unbekannter Fertigungsaufträge plausible Prognosen des Stromverbrauchs erstellt. Dies hätte ein hohes Praxispotential und wäre auch aus wissenschaftlicher Sicht ein Durchbruch. Die Übertragung der grundsätzlichen Idee auf andere Beschreibungsformen der Steuerung und Zeitreihen anderer Messwerte ist ebenfalls denkbar und ein möglicher Gegenstand weiterer Untersuchungen.

Literatur

Brause, R.W.: Neuronale Netze: Eine Einführung in die Neuroinformatik. Wiesbaden: Vieweg-Teubner 1995.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y.: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: Moschitti, A.; Pang, B.; Daelemans, W. (Hrsg.): Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 2014, S. 1724–1734.

Chollet, F., 2015: Keras. Online verfügbar https://keras.io/. Zugriff: 08.05.2019

Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. NIPS 2014 Deep Learning and Representation Learning Workshop (2014).

Eldabi, T.; Balaban, M.; Brailsford, S.; Mustafee, N.; Nance, R.E.; Onggo, B.S.; Sargent, R.G.: Hybrid Simulation - Historical lessons, present challenges and futures. In: Roeder, T.M.; Frazier, P.I.; Szechtman, R.; Zhou, E. (Hrsg.): Proceedings of the 2016 Winter Simulation Conference, Washington, DC, USA, 2016, S. 1388–1403.

Goodfellow, I.; Bengio, Y.; Courville, A.: Deep learning. Cambridge, Mass.: The MIT Press 2017.

Haag, H.: Eine Methodik zur modellbasierten Planung und Bewertung der Energieeffizienz in der Produktion. Stuttgart: Fraunhofer 2013.

Hochreiter, S.; Schmidhuber, J.: Long Short-Term Memory. Neural Computation 9 (1997) 8, S. 1735–1780.

Lawson, B.; Leemis, L.M.: Discrete-event simulation using R. In: Yilmaz, L.; Chan, W.K.; Moon, I.; Roeder, T.M.; Maca, C.; Rossetti, M.D. (Hrsg.): Proceedings of

the 2015 Winter Simulation Conference, Huntington Beach, CA, USA, 2015, S. 3502–3513.

Lunze, J.: Ereignisdiskrete Systeme: Modellierung und Analyse dynamischer Systeme mit Automaten, Markovketten und Petrinetzen. Berlin, Boston: De Gruyter Oldenbourg 2017.

Mustafee, N.; Brailsford, S.; Djanatliev, A.; Eldabi, T.; Kunc, M.; Tolk, A.: Purpose and benefits of hybrid simulation - Contributing to the convergence of its definition. In: Chan, W.Kin; D'Ambrogio, A.; Zacharewicz, G.; Mustafee, N.; Wainer, G.; Page, E.H. (Hrsg.): Proceedings of the 2017 Winter Simulation Conference, Las Vegas, NV, USA, 2017, S. 1631–1645.

Peter, T.; Wenzel, S.: Simulationsgestützte Planung und Bewertung der Energieeffizienz für Produktionssysteme in der Automobilindustrie. In: Rabe, M.; Clausen, U., (Hrsg.): Simulation in Production and Logistics; Dortmund, Stuttgart: Fraunhofer 2015, S. 535–544.

Römer, A.; Rückbrod, M.; Straßburger, S.: Eignung kombinierter Simulation zur Darstellung energetischer Aspekte in der Produktionssimulation. In: C. Deatcu, T. Schramm, Zobel, K. (Hrsg.): 24. Symposium Simulationstechnik ASIM 2018; Hamburg, Wien: ARGESIM 2018, S. 73–80.

RStudio Team, 2015: RStudio: Integrated Development Environment for R. Online verfügbar unter http://www.rstudio.com/. Zugriffsdatum: 08.05.2019

Sutskever, I.; Vinyals, O.; V. Le, Q.: Sequence to Sequence Learning with Neural Networks. Advances in Neural Information Processing Systems 2014.

Thiede, S.: Energy Efficiency in Manufacturing Systems. Berlin, Heidelberg: Springer 2012.

Zell, A.: Simulation neuronaler Netze. München: Oldenbourg 2003.

Zeng, T.; Wu, B.; Zhou, J.; Davidson, I.; Ji, S.: Recurrent Encoder-Decoder Networks for Time-Varying Dense Prediction. In: Raghavan, V. (Hrsg.): 17th IEEE International Conference on Data Mining, New Orleans, LA, 2017, S. 1165–1170.