Sequence to Sequence Modelle zur hochaufgelösten Prädiktion von Stromverbrauch

Benjamin Wörrlein, Steffen Straßburger

Fachgebiet Informationstechnik in Produktion und Logistik, Technische Universität Ilmenau, Max-Planck-Ring 12, 98693 Ilmenau, Deutschland; *benjamin.woerrlein@tu-ilmenau.de, steffen.strassburger@tu-ilmenau.de*

Abstract. Modelling power consumption for jobs on a machine tool is commonly performed by measuring the real power consumption of comparable jobs and machines. The so gathered data is then processed to represent the time-averaged sums of power consumptions of previous jobs. These values of power consumption are then used for upcoming comparable jobs. This approach allows for no high-resolution prediction of power consumption and further presumes static processing times of jobs. Here we propose a new approach to model power consumption that incorporates a Sequence-to-Sequence model, which generates time series according to dynamic data, that describes a numerical control code and environment settings such as state of tools, etc.

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* [8]. Die Kombination von diskret-ereignisgesteuerter und kontinuierlicher Simulation als eine Spielart der hybriden Simulation wird traditionell auch als „kombinierte Simulation“ bezeichnet [3].

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 [5, 12].

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 [10] vorgestellt. Er basiert auf der Grundidee der kombinierten Simulation, der z. B. auch in [9] vorgeschlagen wird. Während der Produktions- und Logistikanteil des Modells klassisch mit ereignisdiskreter Simulation abgebildet wird, wird in [10] 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 [10] erläuterten Ansatz keine Ursache-Wirkzusammenhänge zwischen Steuerparametern (z.B. halber Vorschub, langsamere Hochheizphase) und dem resultierenden Stromverbrauch darstellen.

Der Stromverbrauch einer Maschine aber geschieht über einen Zeitraum, in welchem kontinuierlich Energie benötigt wird, um einen vorher bestimmten Prozess abschließen zu können. Die Dauer dieses Prozesses wird von einer Vielzahl äußerer und innerer Faktoren bestimmt.

Am Beispiel einer Werkzeugmaschine wird deutlich, dass Faktoren wie der spezifische NC-Code eines Fertigungsauftrags maßgeblich darüber entscheiden, wann eine Maschine wieviel Strom verbraucht. Informationen

ARGESIM Report 59 (ISBN 978-3-901608-93-3), p 149-157, DOI: 10.11128/arep.59.a59021

wie der NC-Code beschreiben einen zukünftigen Prozess anhand einer festen Abfolge von Schritten, welche zur Bearbeitung des Prozesses nötig sind. Auch wenn die Abfolge von Schritten vorgegeben und vor dem tatsächlichen Prozessstart bekannt ist, so enthalten diese noch keine Beschreibung der Zeit, die benötigt wird, um die einzelnen Schritte durchzuführen.

Wir verwenden hier ein in [13] zuerst vorgeschlagenes Sequence-to-Sequence Modell zur Abbildung des Wirkzusammenhanges zwischen der Zeitreihe des Stromverbrauchs eines Fertigungsauftrags und alternativer Beschreibungen, wie dem NC-Code, Betriebsparametern, usw.

Der Schwerpunkt der Untersuchungen liegt hierbei auf dem Gebiet des maschinellen Lernens bzw. des *Deep Learnings*.

Ziel des vorgeschlagenen Verfahrens ist es, mittels entsprechend trainierten künstlichen neuronalen Netzen (KNN) Zeitreihen für den Stromverbrauch von Fertigungsaufträgen prognostizieren zu können.

Die Grundidee hierbei ist es, einem Sequence-to-Sequence Modell NC-Codes, sowie entsprechende Betriebszustände etc., und gemessene, hochaufgelöste Zeitreihen des Stromverbrauchs vorhergehender Fertigungsaufträge in einer Trainingsphase zu übergeben. In der Trainingsphase stellt das Sequence-to-Sequence Modell den Zusammenhang zwischen den beiden Beschreibungen her. Ist das Training abgeschlossen, muss dem nun gewichteten neuronalen Netz nur ein NC-Code, und gegebenenfalls weitere betrachtete Faktoren, wie Werkzeugstandzeit etc., übergeben werden, woraufhin dieses eine Zeitreihe des Stromverbrauchs eines Fertigungsauftrags anhand eines NC-Codes generiert.

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 Evaluierung vor und ist hierzu wie folgt gegliedert: Kapitel 1 führt in die benötigten theoretischen Grundlagen des Verwendeten maschinellen Lernverfahrend 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 Sequenzen unterschiedlicher Länge zueinander erlauben.

Hierauf aufbauend wird in Kapitel 2 ein Konzeptvorschlag der Gesamtarchitektur mit seinen Eingangs- und Zielsequenzen entwickelt. Das Konzept wird im Kontext eines Fertigungsauftrages (FA) an einer Werkzeugmaschine (WZM) erstellt. Als Eingangssequenz wird sich hier des NC-Codes und entsprechender Betriebszustände bedient. Anschließend wird der vektorisierte NC-Code auf aufgenommene Zeitreihen der Wirkleistung einer WZM in [kW], hier den Zielsequenzen, trainiert.

In Kapitel 3 werden die notwendigen Schritte zur Vorverarbeitung der Eingangs- und Zielsequenzen erläutert. Einerseits wird eine Methode definiert, die eine Eingangssequenz von Symbolen, wie den NC-Code, etc., in eine für ein 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. Andererseits werden empirische Erkenntnisse im Hinblick auf die Beschaffenheit der Zielsequenzen, hier Zeitreihen, vorgestellt und erläutert. Unsere Forschung weist darauf hin, dass die Verteilung der Häufigkeit einzelner Merkmalsausprägungen von entscheidender Bedeutung ist.

Eine prototypische Umsetzung des Konzepts erfolgt in Kapitel 4. Dieses wird 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. Hier sollen insbesondere die charakteristischen *features* der beiden Zeitreihengruppen einander gegenübergestellt werden.

Eine kritische Betrachtung der Ergebnisse und ein Ausblick über weitere Forschungsrichtungen erfolgt in Kapitel 5.

1 Sequenzmodellierung durch Sequence-to-Sequence

Die hier vorgeschlagene Methode der Sequenzmodellierung zeichnet sich dadurch aus, dass sie bekannte, asynchrone Zustands- bzw. Parameterverläufe, als eine Form von apriorischem Wissen, zur Modellierung von System-

verläufen ermöglicht. Verläufe werden als asynchron zueinander definiert, wenn sie über denselben Start- und Endzeitpunkt verfügen, sich die Taktung in ihren Einträgen aber voneinander unterscheidet. Um solche Verläufe einander zuzuordnen wird sich einer Methode des maschinellen Lernens zu Nutze gemacht. Vorteilig bei der Verwendung von Algorithmen des maschinellen Lernens ist es, dass diese sich, im Anschluss an eine Modellierungsphase, anhand von Realdaten selbst parametrieren.

1.1 Sequenzmodellierung als Ergänzung der ereignisdiskreten Simulation

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 [7].

Gerade aber der Zustandsverlauf eines technologischen Systems kann von einer Vielzahl äußerer 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 [8].

1.2 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 verdeckte 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 [4].

Ändern sich Muster über die Zeit, wird diese zeitliche Abfolge von Mustern als Sequenz verstanden. Damit ein KNN zeitliche Muster verarbeiten kann, müssen rekurrente Verbindungen in der Netztopologie vorhanden sein, welche eine Rückkopplung abstrahierten Wissens zulassen [1, 14]. Solche rückgekoppelten bzw. rekurrenten neuronalen Netze (RNN) eignen sich besonders für Daten, welche in sequentieller Form vorliegen [4].

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 sich an diesen referenzieren kann. Die Anforderungen an eine solche neuronale Zelle mit Gedächtnis werden durch neuronale *Long Short Term Memory* (LSTM)- Zellen [6] und deren vereinfachte Form *Gated Recurrent Unit* (GRU) [2] erfüllt.

Handelt es sich bei den Daten 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 Zielsequenz aus der Abstraktion einer neuronalen Schicht heraus generiert, so wird dieser Teil einer Netzwerktopologie als Decoder bezeichnet [4].

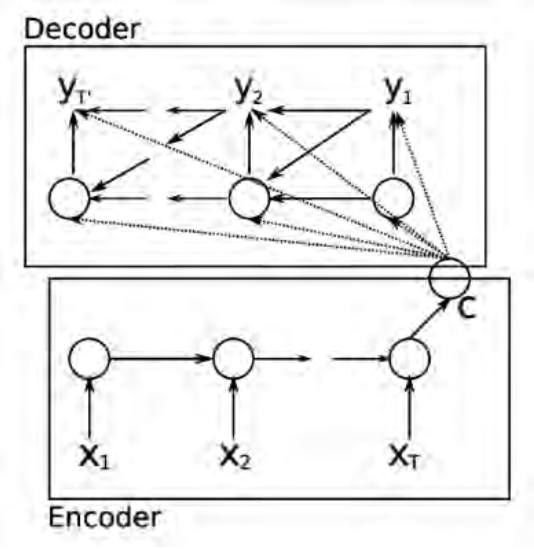

Abbildung 1: Encoder-Decoder Architektur mit den Sequenzen X_T und $Y_{T'}$ unterschiedlicher Länge T≠T' und dem Kontext C [2]

Ist es Aufgabe eines Seq2Seq-Modells, Sequenzen unterschiedlicher Länge auf einander abzubilden, werden solche Strukturen allgemein als rekurrente Encoder-Decoder-Netzwerke (RNN-ED) bezeichnet [4]. Sollen Sequenzen unterschiedlicher Länge und unterschiedlicher Attribute aufeinander abgebildet werden, so müssen diese um eine zusätzliche Beschreibungsart, einen *Kontext* (vgl. Kontextvektor *C* Abb. 1) erweitert werden. Der Kontext kann als Zwischenschicht zwischen den verdeckten Schichten eines Encoders und Decoders verstanden werden [2].

Der Kontext C ist ein Vektor einer Sequenz, welche die Einheiten der verdeckten Schicht des Encoders aufnimmt und diese auf den Decoder abbilden soll [4]. Der Vektor selbst lässt sich anhand einer verdeckten Schicht beschreiben [4]. So wird bei der RNN-ED-Architektur der Kontext C als ein Resultat der finalen verdeckten Schicht des Encoders mit der Eingangssequenz X_T, beschrieben. Da der Encoder in der Trainingsphase seinen finalen verdeckten Zustand weitergibt, muss die ganze Sequenz X_T durchlaufen worden sein (siehe Abb. 1).

Weiterführende Erläuterungen zum hier verwendeten Encoder-Decoder können [2, 4, 11] entnommen werden.

2 Konzept

Wie eingehend erwähnt, wird zur konzeptuellen Überprüfung vorgeschlagen, eine Menge der Eingangssequenzen $\mathbb{X}$ und Zielsequenzen $\mathbb{Y}$ als Sequenzpaarungen $\{X_i, Y_i\}$ der Menge i zu verwenden, welche derselben zeitlich-räumlichen Entität angehören. Von einer zeitlich-räumlichen Entität wird hierbei 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 (siehe Abb. 2).

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 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.

Weiter wird der Beschreibung des FA entsprechende Betriebsmodi, wie Schruppen und Schlichten, angehangen. Der NC-Code, gemeinsam mit dem Betriebsmodus stellen hier die Eingangssequenz X_i eines RNN-ED dar.

Basis der Ausgangszeitreihen Y_i quasi-kontinuierlicher Ausgabewerte ist der Stromverbrauch [kW] desselben FA bei Durchlauf des NC-Codes im jeweiligen Betriebsmodus. 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 von 500 ms.

In der Trainingsphase wird ein ungewichtetes KNN, bestehend aus einem RNN-ED, anhand der Ein- und Zielsequenzen $\{X_i, Y_i\}$ parametriert (siehe Abb. 2).

Aufgabe der Inferenzphase ist es, ein aussagekräftiges Stromverbrauchsprofil $\hat{Y}_i$ explizit quasi-kontinuierlich über die Zeit auszugegeben.

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.

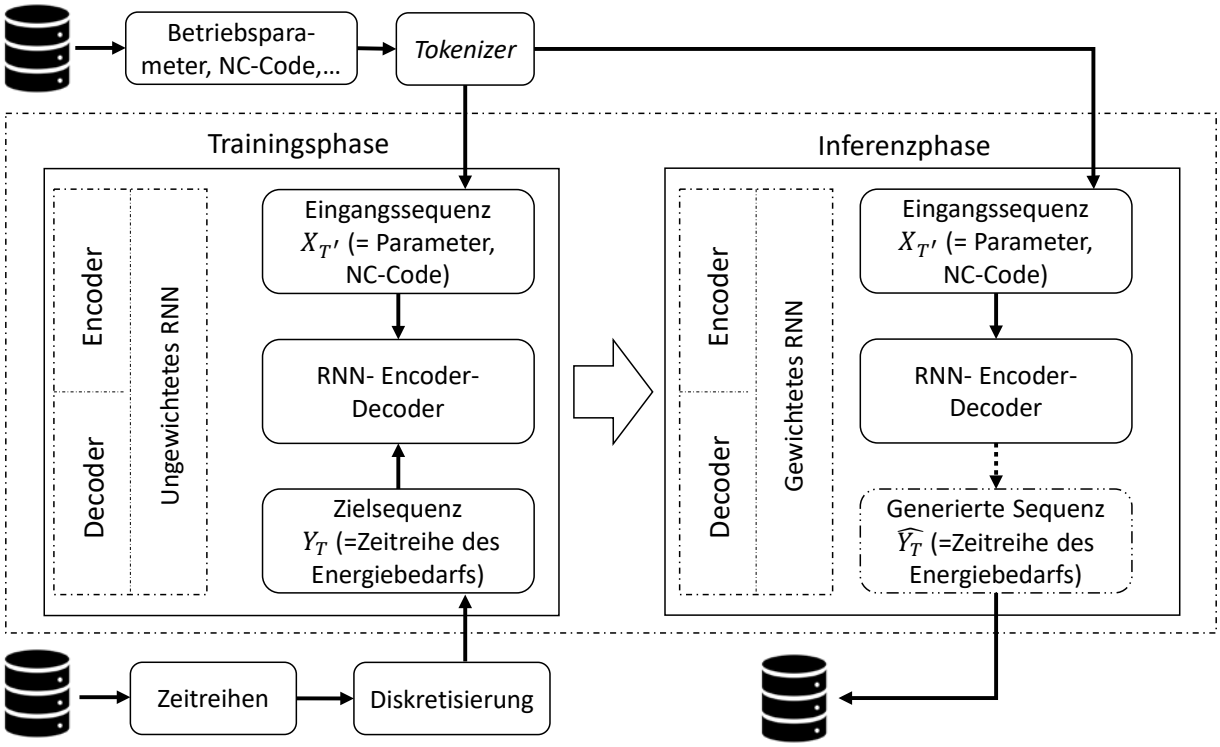

Abbildung 2: Bestandteile einer RNN-Encoder-Decoder Topologie für Sequenzen unterschiedlicher Länge und Symbolik während der Trainings- und Inferenzphase

3 Versuchsvorbereitung

Dem Sequence-to-Sequence Modell werden in der Trainingsphase einerseits Eingangssequenzen, bestehend aus den jeweiligen NC-Codes und Betriebsmodi, andererseits Zielsequenzen, bestehend aus Zeitreihendaten, eines FA gegenübergestellt. Es wurde eine Menge ($i = 51$) Sequenzpaarungen aufgenommen. Diese müssen entsprechend der hier vorgestellten empirischen Ergebnisse vorverarbeitet werden, damit das Modell einen aussagefähigen Zusammenhang zwischen den beiden Mengen erlernen kann.

3.1 Vorbereitung der Eingangssequenzen $\mathbb{X}$

Die Eingangssequenzen $\mathbb{X}$ werden um verschiedene Betriebsmodi $\{x_{11}, x_{12}\}$, in denen die Werkzeugmaschine betrieben werden kann, erweitert. Diese Betriebsmodi spiegeln eine gängige Arbeitsroutine bei der Bearbeitung eines FA wider. Der NC-Code läuft zum ersten Mal $\{Schruppen = x_{11}\}$, um eine größere Menge an überschüssigem Material abzutragen und dem Material seine Form zu geben. Danach wird der gleiche NC-Code noch mehrere Male ausgeführt $\{Schlichten = x_{12}\}$, um die Oberfläche des nun in Form gebrachten Materials zu glätten. Diese beiden Modi resultieren in Zeitreihen der Leistungsaufnahme, die zwar in ihrer Länge vergleichbar sind, aber in ihren Merkmalsverläufen unterschiedliche Eigenschaften aufweisen. Die Eingangssequenz X_i wird entsprechend beschrieben als:

$$X_i = \{\{x_{11}, x_{12}\}, x_2\}$$

wobei x_2 der NC-Code ist.

Die Beschreibungen der Eingangssequenz müssen 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 usw. vorhanden sind, einen numerischen Wert bspw. anhand der Häufigkeit des betreffenden Symbols zu.

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

Weiter entfernt der Tokenizer Symbole bzw. Symbolbeschreibungen, welchen ein geringer Informationsgehalt, wie zum Beispiel Kommata und Groß/-Kleinschreibung, unterstellt wird. Werden alle Einträge eines (langen) NC-Codes übernommen, kann dies in einem Vektorraum resultieren, der zwar einen Prozess detailliert beschreibt, aber aufgrund seiner Größe nicht mehr rechentechnisch verarbeitet werden kann. Eine Möglichkeit, die Dimensionen des Vektorraums zu begrenzen, ist es, dem *Tokenizer* eine Anzahl maximal abbildbarer Symbolmengen, d. h. Wörter, anzuzeigen. Dies war jedoch im hier verwendeten Anwendungsfall, aufgrund der relativen Kürze des NC-Codes nicht nötig.

Im Anwendungsfall wurde ein *Word2Vec*-Tokenizer verwendet, welcher alle Symbole und Symbolmengen in den Vektor übernimmt. Die Symbole der Sequenzen von $\mathbb{X}$ werden abschließend in eine Sequenz von ganzzahligen Werten tokenisiert, wobei jedes eindeutige Wort durch genau eine ganze Zahl repräsentiert wird. Dies ermöglicht es, wiederkehrende Muster innerhalb eines NC-Codes zu modellieren.

3.2 Vorbereitung der Zeitreihen $\mathbb{Y}$

Die Grundlage der Werte für die quasi-kontinuierliche Zielzeitreihe $\mathbb{Y}$ bildet der reale Stromverbrauch eines FA beim Durchlauf eines NC-Codes. Eine äquidistante Messreihe gibt konkret Auskunft darüber, wann wie viel Verbrauch entstanden ist, sobald eine Entscheidung über die Bearbeitung eines Auftrags getroffen werden muss. Die Zeitreihendaten wurden unter Feldbedingungen aufgezeichnet und haben die gleiche Taktung Δt=500 ms, sowie eine mediane Länge von 2295 Zeitschritten für den Schruppprozess x_{11} und 2256 Zeitschritten für den Schlichtprozess x_{12}.

Der Energieverbrauch der Werkzeugmaschine und damit die Zeit, die für die Bearbeitung eines spezifischen Auftrags benötigt wird, wird zunächst bei jeder Bearbeitung eines Auftrags überwacht und als Zeitreihendatensatz gespeichert.

Die Menge von Zeitreihen $\mathbb{Y}$ muss weiter diskretisiert werden. Diskretisierung ist der Prozess der Portionierung kontinuierlicher Werte in diskrete Gruppen von Werten oder *bins*, die den ursprünglichen Werten der Daten ähneln. Dies stellte sich als notwendig heraus, da die hier vorgestellten empirischen Ergebnisse zu der Schlussfolgerung führten, dass eine Verteilung von Merkmalshäufigkeiten P^f, welche einer diskreten Gleichverteilung $P^{d.u.d.}$ oder einer *long tail*- Verteilung $P^{long\ tail}$, bei welcher der *tail* zu einer diskreten Gleichverteilung tendiert, das Seq2Seq-Modell darin behindern, eine sinnvolle gemeinsame Verteilung von $\{X_i, Y_i\}$ zu erlernen.

Um den richtigen Diskretisierungsparameter zu finden, d.h. den Grad der Diskretisierung zu bestimmen, wurden mehrere Trainingsläufe mit alternativen Diskretisierungsparametern durchgeführt. Die verschiedenen Diskretisierungsparameter wurden auf den gesamten Zeitreihendatensatz angewendet und anschließend entsprechend der Merkmalsfrequenz f der diskretisierten Werte klassifiziert (siehe Abb. 3). Dazu wurde die Verteilung der Merkmalshäufigkeiten P^f mit einem gaussschen Kerndichteschätzer (KDE) analysiert.

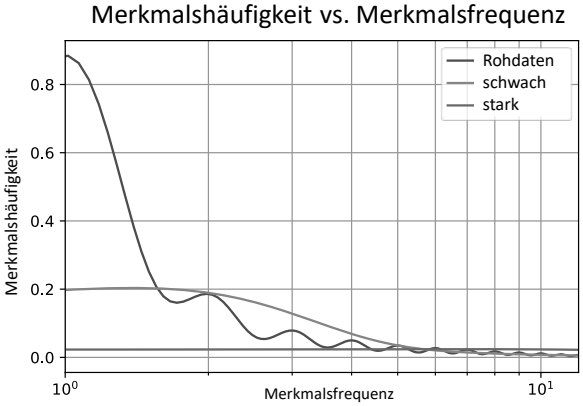

Abbildung 3: Der KDE-Plot zeigt, dass die Rohdaten meist einmalige Werte enthielten, während eine *starke* Diskretisierung zu einer gleichmäßigen Verteilung führt, bei der die Wahrscheinlichkeit, dass ein Wert zu einer beliebigen Frequenz gehört, die gleiche ist wie bei jeder anderen Frequenz. Eine *schwache* Diskretisierung führt zu einer *heavy-tail* Verteilung der Frequenzen.

Das vorgeschlagene Konzept wurde für alle drei Häufigkeitsverteilungen erprobt und führt nur bei der schwachen Diskretisierung zu zufriedenstellenden Ergebnissen.

4 Versuchsdurchführung

Als Metriken zum Vergleich der erzeugten Zeitreihen $\widehat{Y}_i$ und den Trainingszeitreihen Y_i werden die mittlere Länge $len(\bar{Y}_i)$ und die durchschnittliche Summe $sum(\bar{Y}_i)$ der Zeitreihen, wie sie im Trainingsset gefunden werden, herangezogen. Außerdem wurden die Zeitreihen visualisiert und Merkmale charakteristischer Muster (*features*) zu diesen Visualisierungen hinzugefügt

(siehe Abb. 5). Das Hinzufügen von *features* hilft, die Zeitreihen $\widehat{Y}_i$ und Y_i intuitiver auf visueller Ebene zu vergleichen.

Die auf Basis der Rohdaten, welche nicht diskretisiert wurden, erzeugte Zeitreihe stimmt mit der geringen Aussagekraft, wie in [13] beschrieben, überein. Die erzeugten Sequenzen zeigten keinen sinnvollen Werteverlauf und versäumten es weiterhin, ein EOS-*Token* zu erzeugen, d. h. die Methode erzeugt infinit neue Zeitreiheneinträge bis ein generisches Abbruchkriterium gefunden wurde.

Die Ergebnisse für die starke Diskretisierung, wie in Abbildung 4 dargestellt, stellen eine Verbesserung dar. Es wurde ein EOS-*Token* erstellt, wie auch die meisten anderen Zeitreihenmerkmale, doch die generierte Zeitreihe kann deutlich von denen der Trainingsdaten unterschieden werden (vgl. Stichprobenbeispiele in Abbildung 5) und führen schlussendlich zu niedrigen Werten in den Metriken.

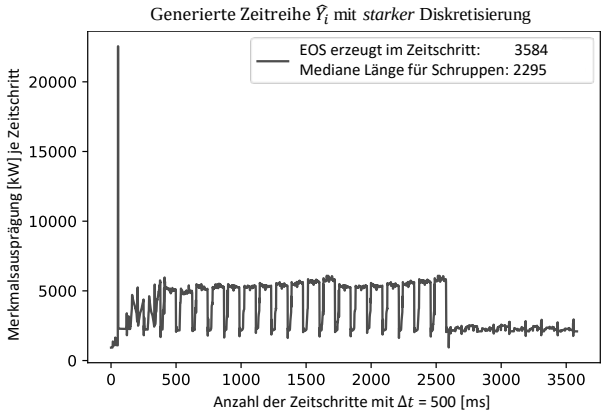

Abbildung 4: Ergebnis für eine *starke* Diskretisierung und Parametereinstellung $\{x_{11}, x_2\}$. Ein EOS-*Token* wurde, wie die meisten anderen Merkmale, erstellt, doch die generierte Serie kann klar von den Trainingsdaten unterschieden werden (siehe Abbildung 5).

Die *schwache* diskretisierte Zeitreihe hingegen zeigt einerseits hohe Werte für $len(\bar{Y}_i)$ und $sum(\bar{Y}_i)$:

$\{x_{11}, x_2\}$: $\frac{len(\hat{y}=2258)}{len(\bar{y}_i=2295)} = 98.4\ \%$; $\frac{sum(\hat{y}=6847.7)}{sum(\bar{y}_i=6927.9)} = 98.8\ \%$

$\{x_{12}, x_2\}$: $\frac{len(\hat{y}=2204)}{len(\bar{y}_i=2256)} = 97.7\ \%$; $\frac{sum(\hat{y}=4843.3)}{sum(\bar{y}_i=4871.8)} = 99.4\ \%$

Andererseits ist bei näherer Betrachtung der Zeitreihen $\widehat{Y}_i$ und Y_i für $\{x_{11}, x_2\}$, wie in Abbildung 5 dargestellt, eine auffällige Ähnlichkeit zu erkennen. Die generierte Zeitreihe schafft es, den Verlauf der *features*, wie in den Stichproben gezeigt, mit einer bemerkenswerten Präzision nachzuahmen. Sie vermag nicht nur, einen EOS-*Token*, der der Lage der im Trainingsset gefundenen Zeitreihe entspricht (grünes *feature*), eine ausgeprägte Lastspitze (rotes *feature*), eine Folge von Untersequenzen (*features* wechselnder Blautöne) zu reproduzieren, sondern auch, diese in der richtigen Reihenfolge und Dimensionalität zu erzeugen.

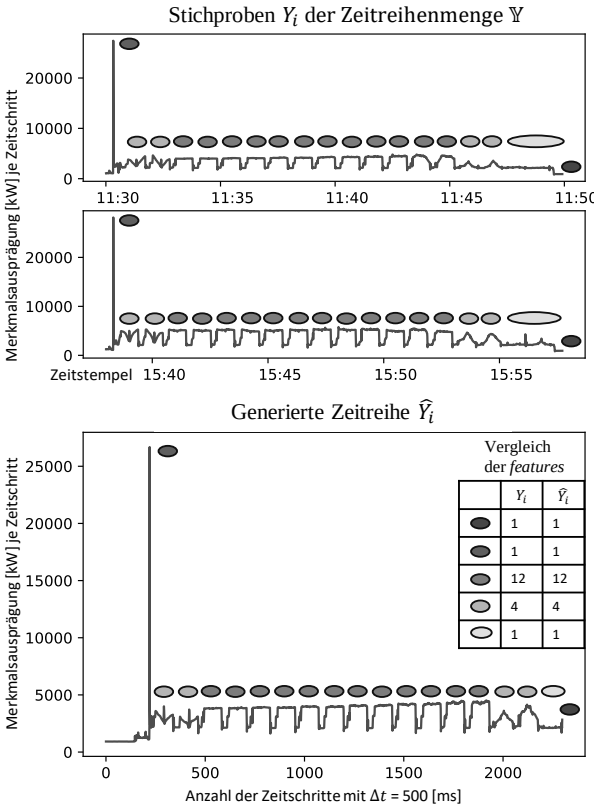

Abbildung 5: Vergleich der Proben Y_i aus dem Trainingsset $\mathbb{Y}$ und der generierten Zeitreihe $\widehat{Y}_i$ für den *schwachen* Diskretisierungsparameter und die Sequenzkombination $\{x_{11}, x_2\}$. Die angezeigten Sequenzen zeigen deutlich die gleichen Muster in ihrem Verlauf. Um den visuellen Vergleich zwischen verschiedenen Sequenzen zu erleichtern, wurden den lokalen Mustern, *Points of Interest- features* hinzugefügt. Die Tabelle auf der rechten unteren Seite vergleicht die Anzahl der *features* miteinander.

Die in Abbildung 6 dargestellten Zeitreihen $\widehat{Y}_i$ und Y_i für $\{x_{12}, x_2\}$ zeigen ebenfalls deutlich, dass es dem Seq2Seq-Modell gelungen ist, den Verlauf der Labels innerhalb des Trainingssets zu erfassen, obwohl die Trainingszeitreihe nur wenige Merkmale enthielt, die überhaupt erlernt werden konnten.

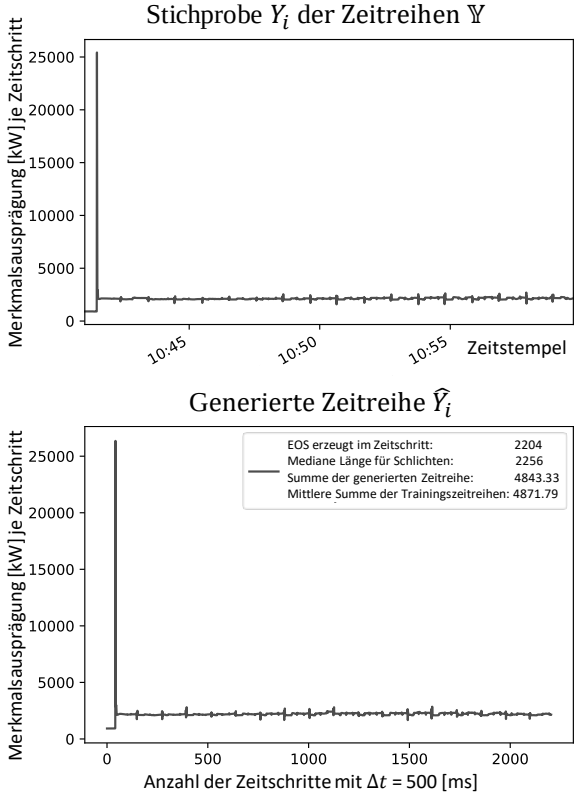

Abbildung 6: Vergleich der Proben Y_i aus dem Trainingsset $\mathbb{Y}$ und der generierten Zeitreihe $\hat{Y}_i$ für die schwache Diskretisierung und die Sequenzkombination $\{x_{12}, x_2\}$. Es wurden keine Merkmale hinzugefügt, da die Zeitreihe nur wenige Merkmale aufweist.

5 Kritische Betrachtung

Die grundlegende Funktionalität der beschriebenen Methode wurde im Testszenario bestätigt. Die generierten Zeitreihen müssen jedoch in weiteren Forschungsarbeiten noch kritisch hinterfragt und validiert werden. Zum einen fehlen Evaluationsmethoden für generative Modelle des maschinellen Lernens, um die generierten Zeitreiheneinträge auf die Aussagekraft ihrer Einträge zu überprüfen. Dies geschieht derzeit durch Analyse und den Vergleich der generierten Zeitreihen durch einen Experten des Anwendungsfalles mittels optischer Inspektion [4], wie in Abbildung 5 dargestellt.

Für eine abschließende Bewertung der verwendeten Methoden ist es ratsam, die qualitative und quantitative Datenbasis des Seq2Seq-Modells zu erweitern. Der hier verwendete Datensatz ist von geringer Größe. Dennoch ist die Größe des Datensatzes beispielhaft für reale Situationen, die sich schnell und in kurzer Zeit ändern können. Algorithmen des maschinellen Lernens hingegen konvergieren in der Regel aber umso besser, je mehr Daten vorhanden sind, aus denen gelernt werden kann.

Eine Methode, in der der Trainingsdatensatz um synthetische Zeitreihen erweitert wird, die so beschaffen sind, dass sie einen *gemittelten* Verlauf der Zeitreihen des Trainingsdatensatzes darstellen, könnte dieses Problem lösen. *Dynamic Time Warping* könnte verwendet werden, um solche *Ground Truth*-Zeitreihen zu erzeugen, die dann iterativ dem Trainingsset hinzugefügt werden könnten, bis ein vorteilhaftes Lernverhalten erreicht wird.

Zusätzliche *End-of-Sequence-Token* könnten zur Beschreibung von Ereignissen wie Maschinenausfall verwendet werden. Die hier verwendeten EOS-*Token* markierte lediglich das Ende eines abgeschlossenen FA. Jedoch sind einige Aufträge aufgrund von Systemveränderungen wie z.B. Verschleiß des Werkzeugs anfällig für Werkzeugbruch, etc. Das Hinzufügen eines alternativen EOS-*Tokens* zum Trainingsdatensatz, der einen Maschinenausfall markiert, zusammen mit zum Zustandsdaten der Werkzeuge usw., könnte auch die Frage beantworten, ob ein Auftrag im Hinblick auf die bekannten Systemgrößen erfolgreich ausgeführt werden kann.

Das hier vorgestellte Seq2Seq-Modell erlaubt es außerdem, Zeitreihen auf Basis faktorieller Kombinationen zu generieren, welche in den Trainingsdaten nicht vorhanden sind. Da der Decoder nicht direkt auf die in der Eingangssequenz gefundenen Beschreibungen parametrisiert wird, sondern auf eine Zusammenfassung ebendieser in Form eines *Kontext*vektors, können faktorielle Kombinationen von Eingangsparametern verwendet werden, die im ursprünglichen Trainingsdatensatz nicht repräsentiert sind. Sobald die jeweiligen Eingabeparameter und ihre spezifische Wirkung auf die Zeitreihe modelliert worden sind, können beliebige Kombinationen von Eingangsparametern verwendet werden. Dies würde zu Faktorkombinationen führen, die für einen Simulationsexperten von hohem Interesse sind und die in einer generischen Simulationsstudie nur mit hohem Aufwand modelliert werden könnten.

Darüber hinaus muss dem vorgeschlagenen Modell eine geeignete Bewertungsmethode hinzugefügt werden, da die Validierung der generierten Zeitreihen sich nicht an einer (nicht vorhandenen) idealen Zeitreihe orientieren kann.

Weiter könnte bei erfolgreicher Etablierung und Validierung der Methode eine Lösung entwickelt werden, die auf Basis nicht in den Trainingsdaten vorhandener

NC-Codes, plausible Stromverbrauchsprognosen für neue FA‘s generiert. Dies hätte ein hohes praktisches Potenzial und wäre auch aus wissenschaftlicher Sicht ein Durchbruch.

Die Weiterentwicklung der oben beschriebenen Methode des maschinellen Lernens und ihre Anwendung für hybride Simulationsmodelle ist derzeit Gegenstand laufender Forschung.

Die Übertragung der Grundidee auf andere Formen von Eingangssequenzen und Zeitreihen anderer Messwerte ist ebenfalls denkbar und ein möglicher Gegenstand weiterer Untersuchungen.

Literaturverzeichnis

[1] Rüdiger W. Brause. 1995. *Neuronale Netze. Eine Einführung in die Neuroinformatik* (2., überarbeitete und erweiterte Auflage). Leitfäden der Informatik. Vieweg+Teubner Verlag, Wiesbaden.

[2] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar.

[3] Tillal Eldabi, Mariusz Balaban, Sally Brailsford, Navonil Mustafee, Richard E. Nance, Bhakti S. Onggo, and Robert G. Sargent. 2016. Hybrid Simulation. Historical lessons, present challenges and futures. In *Proceedings of the 2016 Winter Simulation Conference (WSC). Simulating complex service systems*. IEEE, Piscataway, NJ, 1388–1403. DOI: https://doi.org/10.1109/WSC.2016.7822192.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

[5] Holger Haag. 2013. *Eine Methodik zur modellbasierten Planung und Bewertung der Energieeffizienz in der Produktion*. Zugl.: Stuttgart, Univ., Diss., 2013. Stuttgarter Beiträge zur Produktionsforschung, 11. Fraunhofer Verlag, Stuttgart.

[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8, 1735–1780.

[7] Jan Lunze. 2017. *Ereignisdiskrete Systeme*. De Gruyter, Berlin, Boston.

[8] Navonil Mustafee, Sally Brailsford, Anatoli Djanatliev, Tillal Eldabi, Martin Kunc, and Andreas Tolk. 2017. Purpose and benefits of hybrid simulation: Contributing to the convergence of its definition. In *Proceedings of the 2017 Winter Simulation Conference (WSC)*. IEEE Press, Piscataway, NJ, 1631–1645. DOI: https://doi.org/10.1109/WSC.2017.8247903.

[9] T. Peter and S. Wenzel. 2015. Simulationsgestützte Planung und Bewertung der Energieeffizienz für Produktionssysteme in der Automobilindustrie. In *Simulation in production and logistics 2015. 16. ASIM Fachtagung Simulation in Produktion und Logistik, Dortmund, 23. - 25. September 2015 ; Tagungsband*, Markus Rabe and Uwe Clausen, Eds. ASIM-Mitteilung, 157. Fraunhofer Verl., Stuttgart, 535–544.

[10] Anna C. Römer, Martina Rückbrod, and Steffen Straßburger. 2018. Eignung kombinierter Simulation zur Darstellung energetischer Aspekte in der Produktionssimulation. In *ASIM 2018 : 24. Symposium Simulationstechnik, 4. bis 5. Oktober 2018, HafenCity Universität Hamburg : Tagungsband*. ARGESIM/ASIM, Wien, 73–80.

[11] Ilya Sutskever, Oriol Vinyals, and Quoc Le V. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS’14: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. MIT Press, Cambridge, MA, USA.

[12] Sebastian Thiede. 2012. *Energy efficiency in manufacturing systems*. Zugl.: Braunschweig, Techn. Univ., Diss., 2011. Sustainable production, life cycle engineering and management. Springer, Berlin.

[13] Benjamin Wörrlein, Sören Bergmann, Niclas Feldkamp, and Steffen Straßburger. 2019. Deep-Learning-basierte Prognose von Stromverbrauch für die hybride Simulation. In *Simulation in Produktion und Logistik 2019*. Verlag Wissenschaftliche Scripten, Auerbach, 121–131.

[14] Andreas Zell. 2003. *Simulation neuronaler Netze* (4., unveränd. Nachdr). Oldenbourg, München.