Hochaufgelöste Energieprofile durch hybride Simulation

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. The price of a commodity, as electricity, is determined on a commodity market. A market is efficient when the supply and demand in the market are at an equilibrium. Efficient markets run on information. Information can cause a spontaneous and instantaneous change within the supply and demand in a market. The market communicates this new equilibrium through the change of the price of a commodity. In the electricity market the supplier and consumer communicate through electrical load profiles. A load profile signals when and how much energy should be consumed within a certain time frame without causing a change in the price of electricity. Creating such load profiles is commonly done by the supplier of energy by means of standard load profiles. Here we propose a data-driven simulation-based method that allows for the consumer to create its own specific load profile, which potentially will bring down the cost of energy consumed.

Einführung

Lastprofile beschreiben den Energiebedarf eines Systems über einen bestimmten Zeitraum. Energieerzeuger bedienen sich Lastprofilen, um einem Konsumenten mitzuteilen, wann wieviel Energie abgerufen werden kann. Weicht der tatsächliche Energiebedarf eines Konsumenten von dem im Lastprofil beschriebenen Energieangebot ab, muss der Konsument entsprechend der Differenz ein erhöhtes Netzentgelt entrichten [1]. Standardlastprofile werden von Energieerzeugern für Abnehmer mit einem Jahresverbrauch von kleiner 100 MWh erstellt [2]. Für Abnehmer mit einem höheren Jahresverbrauch kann im Rahmen der registrierenden Leistungsmessung ein spezifisches Lastprofil für eine definierte Zeitperiode (täglich, wöchentlich, monatlich, …) erstellt werden [2, 3].

Dieses spezifische Lastprofil wird auf Basis der in der registrierenden Leistungsmessung erhobenen Daten erstellt. Das erstellte Lastprofil soll nun möglichst nah das Abrufverhalten von Energie eines Konsumenten beschreiben. Anhand eines zwischen Erzeuger und Konsument vereinbarten Lastprofils wird nun ein Lieferabkommen geschlossen. Weicht der Konsument in seinem Abrufverhalten vom vereinbarten Lastprofil ab, erklärt sich der Erzeuger weiter bereit ihm Energie zu liefern, diese ist aber in Abhängigkeit der erhöhten Volatilität höher bepreist. Dieses Vorgehen hat mehrere Nachteile:

1. Der Erzeuger hat hier einen Informationsvorsprung vor dem Konsumenten. Durch das Vorhandensein mehrerer Abnehmer, fällt diesem in der Regel die Erstellung des Lastprofils zu. Der Erzeuger seinerseits hat nun das Interesse die Lastprofile der Konsumenten möglichst an dem für ihn optimalen Abrufverhalten und nicht dem der Konsumenten zu orientieren.
2. Spezifische Lastprofile werden für definierte Perioden erstellt. Das optimale Abrufverhalten des Verbrauchers kann nun aber eine so starke Volatilität aufweisen, dass dieses nicht in einem periodischen Lastprofil abgebildet werden kann. Hier kann für den Konsumenten ein Optimierungsproblem der Zielgrößen Energiebezugskosten und bspw. der Gesamtanlageneffektivität (*Overall Equipment Effectiveness* – OEE) entstehen.
3. Der Konsument hat nur begrenzte Möglichkeiten auf die vor Abruf vereinbarten Lastprofile Einfluss zu nehmen. Dies liegt vorrangig daran, dass er im Falle von erwarteter starker Volatilität diese nicht beschreiben kann. Volatilität geht mit einer verringerten Prognosegüte in Abhängigkeit zum betrachteten Zeitraum einher.

Wir schlagen hier eine datengetriebene Simulationsmethode vor, um den oben beschriebenen Nachteilen entgegenzuwirken. Basis dieser Methode ist eine Hybride Simulation (HS) nach [4], welche eine diskret-ereignisorientiere Simulation (DES) mit einem Modell des Maschinellen Lernens (ML) verbindet. Bei dem ML-Modell handelt es sich hierbei um ein sog. Sequence-to-Se-

quence-Modell (Seq2Seq) [5, 6]. Seq2Seq-Modelle lernen zwei Sequenzen aufeinander abzubilden und werden standardmäßig für Übersetzungen verwendet. Die HS soll für eine mechanische spanende Fertigung mit mehreren parallelen CNC-Maschinen angewendet werden. Auf jeder Fertigungslinie soll anhand eines Produktionsplans eine Folge von Fertigungsaufträgen (FA) abgearbeitet werden. Der Produktionsplan enthält lediglich den Startzeitpunkt eines FA und eine Beschreibung des FA, wie den NC-Code, den Fertigungsmodus (Schruppen oder Schlichten) etc. Dieser Ansatz wurde schon für einen einzelnen FA vorgeschlagen [7], validiert und dessen Einbindung in DES diskutiert [8]. Der hier vorgeschlagene Ansatz soll sich nun mit der Erstellung von Lastprofilen für eine gesamte Fertigung beschäftigen.

Ein Seq2Seq-Modell erlernt den Zusammenhang zwischen der Beschreibung eines FA und dessen Zeitreihe des Energiebedarfs. Belegt nun in der DES der FA eine Fertigungslinie wird das trainierte Seq2Seq-Modell geladen und mit den Beschreibungen des FA aktiviert. Das Seq2Seq-Modell gibt nun eine hochaufgelöste Zeitreihe des Energiebedarfs des FA in Abhängigkeit seiner Beschreibung zum Aktivierungspunkt innerhalb der Simulation wieder. Somit kann für jede Fertigungslinie ein eigenes spezifisches Lastprofil erzeugt werden. Die Summe aller Fertigungslinien ergibt das spezifische Lastprofil des betrachteten Produktionssystems.

Ziel dieser Veröffentlichung ist es die Ergebnisse der oben beschriebenen HS vorzustellen und auf etwaige Schwächen und Stärken der beschriebenen Methode einzugehen. Die Validierung des Ansatzes geschieht anhand realer Daten eines Fertigungssystems für eine Fertigungsschicht.

1 Datengetriebene HS

1.1 Hybride Modelle zur Erzeugung von Lastprofilen

In einer klassischen DES können FA anhand ihres jeweiligen Startzeitpunktes im Produktionsplan abgebildet werden. Die Länge eines FA wird hier zuerst in Abhängigkeit zu einer aus den Daten ermittelten Verteilungsfunktionen bestimmt. Der eigentliche Energiebedarf wird anschließend meist statusbasiert über einen mittleren Leistungsbedarf bestimmt. Dieses Vorgehen kann nachteilig sein.

Erstens ändert sich ein Produktionsplan ständig in Abhängigkeit zu den Anforderungen an das Produktionssystem und kann über eine Fertigungsschicht hinaus kaum vorherbestimmt werden. Er eignet sich also kaum zur Erstellung eines Lastprofils über längere Perioden.

Zweitens bilden die Verteilungsfunktionen der Länge und der statusbasierte Abruf des Energiebedarfs der einzelnen FA lediglich aus den Daten ermittelte statistische Größen, wie Mittelwert und Standardabweichung, ab. Dies ist einerseits nachteilig, da sie von einer starken Vergleichbarkeit der FA ausgehen und andererseits nur einen gemittelten Energiebedarf wiedergeben. Etwaige Lastspitzen, wiederkehrende Muster oder andere Charakteristika des Energiebedarfs eines FA gehen somit verloren, und stehen bei der Erstellung eines hochaufgelösten Lastprofils nicht mehr zur Verfügung.

Als ursächlich hierfür kann die grundsätzliche Limitation ereignisdiskreter Simulationsansätze angesehen werden, dass Zustandsänderungen zwischen zwei Ereignissen nicht abbildbar sind. Somit wird der Energiebedarf der Einfachheit halber zwischen den Start- und Endereignissen eines Auftrages als konstant betrachtet.

Um diese Limitation zu überwinden, bietet sich der Ansatz der hybriden Simulation an [4], der es ermöglicht, den ereignisdiskreten Modellteil um ein weiteres Modellierungsparadigma zu ergänzen, welches eine höheraufgelöste Abbildung des Energiebedarfs ermöglich.

Eine mögliche Variante hierzu kann die kombinierte Simulation sein, also die Kombination eines ereignisdiskreten und eines kontinuierlichen Modellteils [9]. Wenn es die Natur des technischen Systems zulässt, kann dessen Energiebedarf über Differentialgleichungen modelliert werden [10]. Dies ist jedoch bei vielen Produktionsressourcen, wie z.B. CNC-Maschinen, nicht gegeben.

In derartigen Fällen greift man typischerweise auf hochaufgelöste gemessene Zeitreihen des Energiebedarfs zurück. Diese können direkt in der hybriden Simulation verwendet werden, indem sie z.B. über Tabellenfunktionen in Kombination mit dem System-Dynamics-Ansatz in die Simulation „eingespielt“ werden [11]. Über diesen Mechanismus lassen sich bereits Lastprofile ganzer Fertigungslinien erstellen.

Wesentliche Limitation dieses Ansatzes ist jedoch, dass nur bereits bekannte und gemessene Zusammenhänge wiedergegeben werden können. Eine Prognose des Energiebedarfs unbekannter Aufträge ist nicht möglich. Weiterhin ist keine Abbildung von Ursache-Wirkungszusammenhängen zwischen technologischen Steuerparametern (z.B. halbe Vorschubgeschwindigkeit, langsa-

mere Aufwärmphase) und dem resultierenden Energiebedarf möglich.

Eine Abhilfe hierfür könnte der in diesem Beitrag vorgestellte Ansatz zur hybriden Simulation darstellen, bei dem der ereignisdiskrete Modellteil um ein spezielles Modell des maschinellen Lernens ergänzt wird, das eine hochaufgelöste Zeitreihe des Energiebedarfs eines Auftrags auf Basis von dessen NC-Code prädiziert.

1.2 Rekurrente Netze und Encoder-Decoder Architekturen

Künstliche neuronale Netze (KNN) werden zur Identifikation von Mustern in komplexen Datenstrukturen verwendet. 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 [12, 13]. Solche rückgekoppelten bzw. rekurrenten neuronalen Netze (RNN) eignen sich besonders für Daten, welche in sequentieller Form vorliegen [14].

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 in eine spezifische neuronale Schicht codiert, so ist dies ein Encoder. Wird eine Zielsequenz aus einer neuronalen Schicht heraus generiert, so wird dieser Teil einer Netzwerktopologie als Decoder bezeichnet [14].

Ist es Aufgabe eines Seq2Seq-Modells, Sequenzen unterschiedlicher Länge auf einander abzubilden, werden solche Strukturen allgemein als rekurrente Encoder-Decoder-Netzwerke (RNN-ED) bezeichnet [14]. Sollen Sequenzen unterschiedlicher Länge und unterschiedlicher Attribute aufeinander abgebildet werden, so müssen diese um eine zusätzliche Beschreibungsart, einen *Kontext* erweitert werden.

Der Kontext C ist der finale Zustandsvektor der verdeckten Schicht des Encoders und soll diesen auf den Decoder abbilden [14]. Der Vektor selbst lässt sich anhand einer verdeckten Schicht beschreiben [14]. 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 Zustand weitergibt, muss die ganze Sequenz X_T durchlaufen worden sein.

Weiterführende Erläuterungen zum hier verwendeten Encoder-Decoder können [5, 6, 14] entnommen werden.

2 Konzept

Ziel des Simulationsmodells ist es, für einen Fertigungsverbund von Maschinen (siehe Abb. 1) ein Gesamtlastprofil zu erzeugen. Die abgebildeten Maschinen sollen als Eingangsparameter lediglich Beschreibungen der Fertigungsaufträge erhalten, welche vor einer etwaigen Schicht verfügbar wären. Diese sind hier Startzeitpunkte, Betriebsmodi und NC-Code der FA auf jeder Maschine.

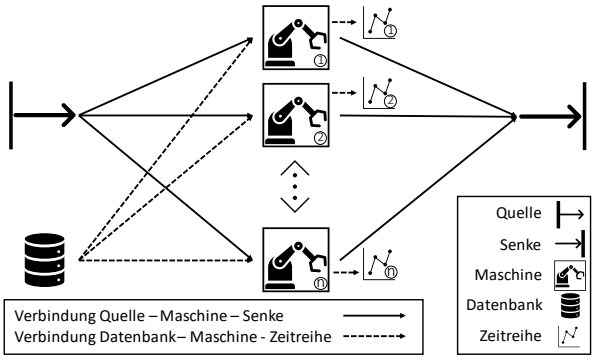

Abbildung 1: Modell des untersuchten Fertigungsverbunds

Erreicht die Simulationszeit die Startzeit des FA, wie in den Beschreibungen des FA hinterlegt, wird die Maschine belegt und arbeitet nun den FA ab. Hierbei wird die Länge des FA und die benötigte Energie vom trainierten Seq2Seq-Modell anhand der Beschreibung des spezifischen FA prognostiziert.

2.1 Generierung von Zeitreihen

Wie eingehend erwähnt wird eine Menge von Eingangssequenzen $\mathbb{X}$ und Zielsequenzen $\mathbb{Y}$ als Sequenzpaarungen $\{X_i, Y_i\}$ in einem Seq2Seq- Modell verwendet. Hierfür wird der Spanprozess eines FA auf einer Werkzeugmaschine (WZM) betrachtet.

Ein NC-Code beschreibt eine Abfolge notwendiger technologischer Schritte 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 die Beschreibung des FA um entspre-

chende Betriebsmodi, Schruppen und Schlichten, ergänzt. 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 Strombedarf [W] desselben FA bei Durchlauf des NC-Codes im jeweiligen Betriebsmodus. Der zeitliche Strombedarf gibt konkret Aufschluss darüber, wann mit wieviel benötigter Leistung gerechnet werden muss, sobald über die Einsteuerung eines FA entschieden werden muss.

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

Ist die Parametrierung des RNN-ED abgeschlossen, benötigt dieses lediglich die Eingangssequenz, um die Ausgangssequenz zu erzeugen. Aufgabe der Inferenzphase ist es nun, ein aussagekräftiges Strombedarfsprofil $\widehat{Y}_i$ auszugegeben. Inferenzierung ist hier das logische Schließen aus Daten.

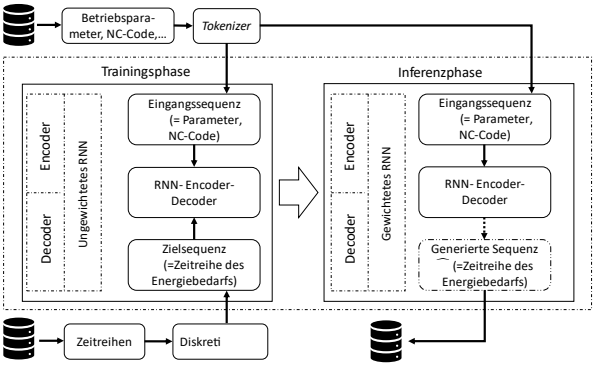

Abbildung 2: Bestandteile einer RNN-Encoder-Decoder Topologie für Sequenzen unterschiedlicher Länge und Symbolik während der Trainings- und Inferenzphase [8].

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

2.2 Inferenz zur Laufzeit

Um die Generierung der Zielsequenz $\widehat{Y}_i$ innerhalb einer Simulation zu gewährleisten, muss die Inferenzmethode als Funktion hinterlegt sein (siehe Code 1).

```
def seq2seq(job):
   load seq2seq model
   seq2seq(job)
   return ts_job
```

Code 1: Pseudocode der Inferenzmethode

Hierfür muss lediglich das parametrierte Modell *seq2seq model* geladen werden und diesem die Beschreibungen des spezifischen FA *job* als Parameter übergeben werden. Die Inferenzmethode kann nun innerhalb des Simulationsmodells einer WZM aufgerufen werden und gibt eine Zeitreihe *ts_job* zurück.

Das hier verwendete Modell der Fertigung (siehe Code 2) besteht aus einem Maschinenmodell auf welchem sich zwei Prozesse, Warten und Fertigen, abwechseln.

```
class machine(name):
   self.name=name
   self.parts_made=0
   self.jobs=[list of jobs and descriptions]
   self.ts_energy= empty series

   def waiting(self):
       timeout_job=
       self.jobs[parts_made[start time]]
       yield timeout(timeout_job)
       yield machining
   def machining(self):
       seq2seq(jobs[parts_made[description])
       yield timeout(len(ts_job))
       self.parts_made +=1
       concate(ts_energy,ts_job)
       yield waiting
```

Code 2: Pseudocode des Simulationsmodells

Die Maschine wird hier in SimPy, einem objektorientiertem Simulationsparadigma (vgl. [15]), beschrieben. Das Objekt, die Maschine, wird durch mehrere Variablen beschrieben. Sie verfügt über einen Namen, einen Teilezähler, eine Liste von geplanten FA, sowie über ein, anfangs leeres, Datenobjekt, in welchem die Zeitreihen der einzelnen FA gespeichert werden. Weiter verfügt die Maschine über zwei Funktionen, Warten und Fertigen. Die Maschine wartet so lange bis die Simulationszeit der Startzeit eines FA entspricht. Darauffolgend wird Fertigungsprozess begonnen.

Der Fertigungsprozess (*machining*) wird anhand der Seq2Seq Funktion beschrieben. Die Funktion erhält die Beschreibung des aktuellen FA und gibt eine Zeitreihe dessen Energiebedarfs *ts_job* zurück. Die Länge des FA

ist gleich der Anzahl der einzelnen Elemente der Zeitreihe. Die Maschine wird nun für die Länge der Zeitreihe blockiert. Anschließend wird die Zeitreihe im Datenobjekt *ts_energy* gespeichert, die Zählvariable der abgeschlossenen FA um 1 erhöht und die Maschine beginnt wieder zu warten. Dies geschieht, bis alle FA abgearbeitet wurden.

3 Versuchsvorbereitung

3.1 Vorbereitung der Seq2Seq-Methode

Dem Seq2Seq-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 an einer WZM über mehrere Tage aufgenommen. Diese müssen entsprechend der in [16] vorgestellten empirischen Ergebnisse vorverarbeitet werden, damit das Modell einen aussagefähigen Zusammenhang zwischen den beiden Mengen erlernt.

Die Eingangssequenzen $\mathbb{X}$ werden um verschiedene Betriebsmodi $\{x_1, x_2\}$, in denen die WZM betrieben werden kann, erweitert. Diese Betriebsmodi spiegeln eine gängige Arbeitsroutine bei der Bearbeitung eines FA wider. Der NC-Code läuft zum ersten Mal $\{Schruppen = x_1\}$, 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 $\{Schlichten = x_2\}$, um die Oberfläche des nun in Form gebrachten Materials zu glätten.

Die Grundlage der Werte für die Zielzeitreihe $\mathbb{Y}$ bildet der reale Strombedarfs eines FA beim Durchlauf eines NC-Codes. Die Zeitreihendaten wurden unter Feldbedingungen aufgezeichnet und haben die gleiche Taktung Δt=500 ms.

Der Energiebedarf der WZM 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. Anschließend wird das Seq2Seq Modell mit den oben beschriebenen Trainingsdaten parametriert und für die Weiterverwendung in der Simulation gespeichert [8].

3.2 Vorbereitung der Referenzzeitreihen

Zur Validierung des durch HS erzeugten Lastprofils soll parallel hierzu ein Lastprofil rein aus den Daten des zugrundeliegenden Systems erstellt werden (Siehe Abb. 3). Hierzu wird die Messzeitreihe von 6 Tagen tageweise auf 6 Maschinen umgelegt.

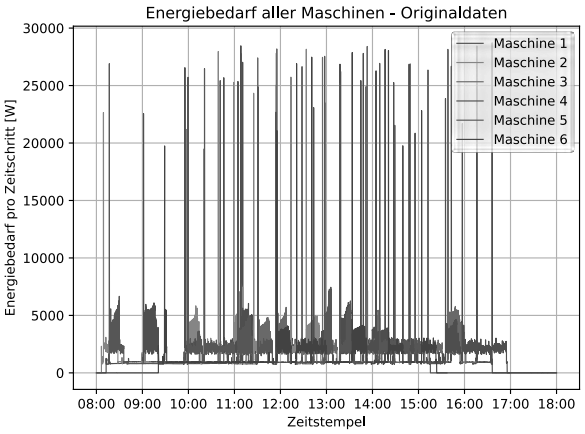

Abbildung 3: Originaldaten des Strombedarfs für alle Maschinen der Fertigung einer Schicht.

Die Daten müssen hierfür vorverarbeitet werden. Einerseits betrachtet die später verwendete Seq2Seq Methode nur den Energiebedarf bei Ausführung eines FA. Die Rohdaten müssen daher um die Zeiträume, in denen keine Fertigung stattfindet, bereinigt werden. Andererseits soll aus den Daten ein Gesamtlastprofil erstellt werden. Hierfür müssen die Zeitreihen anhand eines gemeinsamen Index aufsummiert werden. Die Rohdaten wurden mit der gleichen Messstrecke von 500 ms aufgenommen. Die einzelnen Zeitpunkte unterlegen jedoch einer Messtoleranz, welche zu einem leicht unterschiedlichen Zeitstempel führt.

Weiter werden Lastprofile zur Kommunikation des zeitlichen Energiebedarfs zwischen Verbraucher und Versorger verwendet. Da jeder Versorger über eine unterschiedliche Energieinfrastruktur verfügt (Speicher, Kraftwerke, Netze, etc.) muss das erzeugte Lastprofil an den Bedürfnissen des Versorgers ausgerichtet sein. Diese Bedürfnisse entsprechen der Möglichkeit des jeweiligen Versorgers effizienter Energie in seinem Verteilnetzsystem zur Verfügung zu stellen. Die Lastprofile müssen daher unterschiedliche Fragen nach bspw. mittlerem, minimalem oder maximalem Energiebedarf für unterschiedliche Zeiträume beantworten. Ein Mangel an öffentlicher Information über die versorgerseitige Erstellung von Lastprofilen trägt hierzu weiter bei [17].

Diese Probleme werden dadurch gelöst, dass die Zeitreihen einer *Resampling* Methode unterzogen werden.

Resampling ist der Prozess der Aggregation eines Intervalls an Datenpunkten zu einem neuen Datenpunkt, samt neuem Zeitstempel. Die *Resampling* Methode benötigt hierfür 2 Parameter, die Schrittlänge des Intervalls und die Aggregationsmethode, welche beide anhand des Anwendungsfall bestimmt werden können.

Die Schrittlänge sollte sich einerseits am Betrachtungshorizont des Lastprofils orientieren. Diese wird bedingt durch die registrierende Leistungsmessung auf 15 Minuten Intervalle gelegt. Für die Aggregationsmethode wird eine Mittelwertberechnung, als robustes Vergleichsmaß für den mittleren Energiebedarf, gewählt.

Weiter soll der Minimalbedarf während eines Intervalls bestimmt werden, um eine genauere Aussage über das stromverbrauchende System zu treffen. Hierfür wird wieder ein Intervall von 15 Minuten gewählt, aber dieses Mal mit einer Minimalwertberechnung.

Die Energiebezugskosten sind nicht allein von der Menge des benötigten Stroms, sondern auch vom Zeitpunkt des Bedarfs, in Form vom Versorger zur Verfügung gestellter Kapazität, abhängig. Daher wird die Betrachtung der Zeitreihen noch um ein weiteres *Resampling* erweitert. Hier wird eine Maximalwertmethode angewandt, um den maximalen Bedarf an Energie in einem Intervall darzustellen. Weiter wird das Intervall auf 1 Minute gesetzt. Dies liegt hier darin begründet, dass der maximale Energiebedarf nicht nur über einen längeren Zeitraum dargestellt werden soll, sondern auch in Form von Lastspitzen. Wird ein FA um einen Zeitraum verschoben, verschieben sich auch die Lastspitzen des FA um eben diesen. Dieser Zeitraum ist wiederrum abhängig von der Regelstrecke des Bezugsystems. Diese wurde vor Datenerhebung für den Werker der Maschine auf 1 Minute gesetzt.

Im Anschluss an das *Resampling* werden alle 3 so erzeugten Zeitreihen für alle Maschinen aufsummiert (siehe Abb. 4). Die Methode der Synthese eines Lastprofils aus Einzellastprofilen ist nicht neu und wurde schon bei [2, 18] diskutiert.

Die gewählten *Resampling* Parameter werden für die mit der HS erzeugten Zeitreihen wiederverwendet, um eine Vergleichbarkeit der beiden Datensätze zu erreichen.

4 Versuchsdurchführung

Die HS wird in Python mit SimPy [15] modelliert und durchgeführt. Als Framework für die verwendete Seq2Seq- Methode wird Tensorflow [19] verwendet. Für die GPU Berechnung des Tensorflow Modells stand eine RTX 2080 Ti mit 4352 CUDA Kernen zur Verfügung. Die Parametrierung des Modells dauert auf dieser ca. 6 Tage. Die Inferenzierung eines FA dauert hingegen nur wenige Sekunden.

4.1 Ergebnisse der HS

Die HS wird nun gestartet und gibt Zeitreihen nach der in Code 2 beschriebenen Methode aus. Diese werden noch zur Laufzeit für jede Maschine in einem eigenen Datensatz gespeichert. Anschließend werden die inferenzierten Zeitreihen nach, der in Kapitel 3.2 beschriebenen Methode, aggregiert. Die nun erzeugten Lastprofile (vgl. Abb. 5) stellen den Strombedarf eines Fertigungssystems dar, welche nur durch ein Seq2Seq- Modell, Beschreibungen und Startzeitpunkte der FA erzeugt wurden.

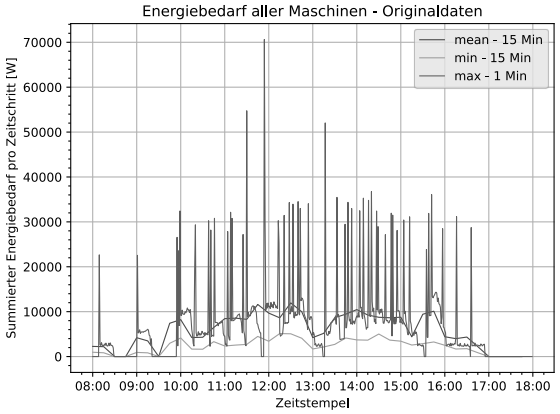

Abbildung 4: Ergebnisse der bereinigten Daten, nach Resampling mit einer Mittel-, Minimal- und Maximalwertmethode.

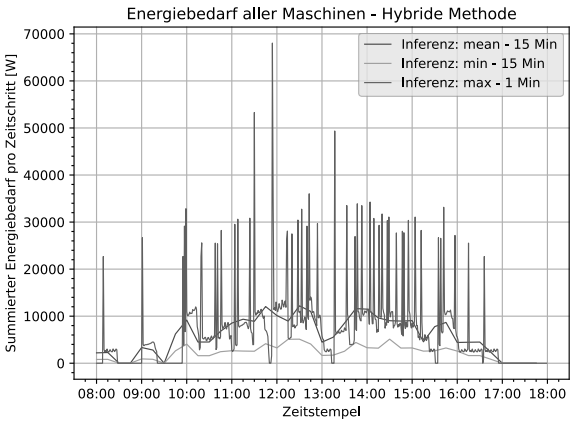

Abbildung 5: Ergebnisse der durch Seq2Seq erzeugten Daten, nach Resampling mit einer Mittel-, Minimal- und Maximalwertmethode.

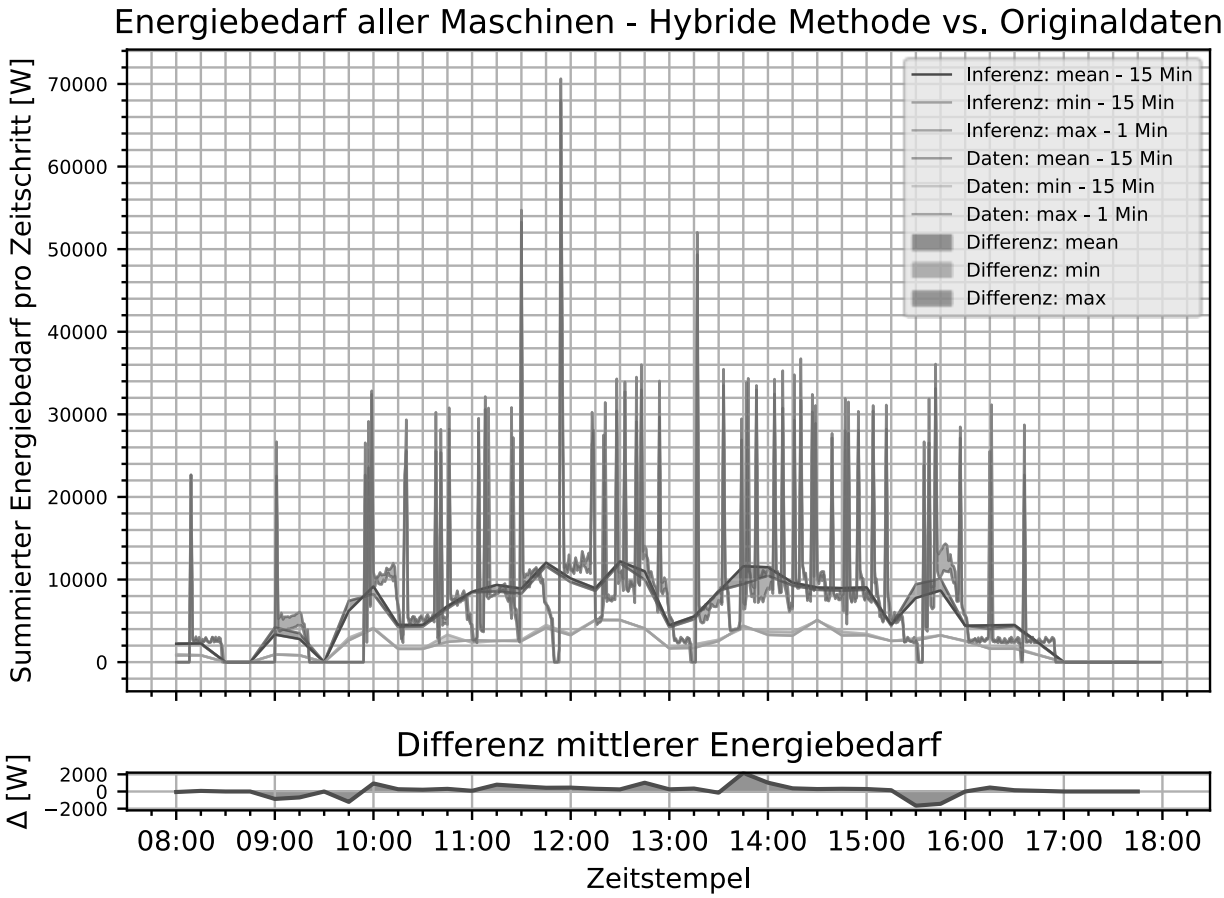

Abbildung 4: Oben: Visueller Abgleich der Ergebnisse der hybriden Methode mit denen der abgespielten Daten. Auffällig ist die starke Übereinstimmung der Trainings- und Inferenzdaten, die auch aus der unten dargestellten Differenz des mittleren Energiebedarfs über die Zeit abzulesen ist.

4.2 Validierung der HS am realen Datensatz

Vergleicht man die verschiedenartig erzeugten Lastprofile von Abb. 4 und Abb. 5 so ist eine deutliche Ähnlichkeit auffällig. Diese Ähnlichkeit wird weiter unterstrichen durch einen visuellen und analytischen Abgleich der beiden Lastprofilmengen.

Im direkten Vergleich (vgl. Abb. 6- oben) sind die Unterschiede, in Form von farbig schattierten Flächen, kaum wahrzunehmen. Dies betrifft alle gegenübergestellten Lastprofile, sowie die Lage der konkreten Lastspitzen. Besonders deutlich wird dies in der über die Zeit abgetragenen Differenz des mittleren Energiebedarfes (vgl. Abb. 6- unten).

Vergleicht man die Lastprofile analytisch anhand ihres konkreten Energiebedarfes über die Gesamtzeit in Kilowattstunden [kWh], so wird dies noch weiter untermauert (vgl. Tab. 1).

Hierzu wird die *Riemann'sche* Summe aller einzelnen Lastprofile gebildet und miteinander verglichen. Die Methode der *Riemann'schen* Summen [20] wird dann angewandt, wenn das Integral über einer Funktion über diskrete Datenpunkte approximiert werden soll.

[kWh]	Seq2Seq	Original-daten	$\frac{Seq2Seq}{Daten}$
Mean	59.47	58.07	1.02
Min	22.18	23.19	0.96
Max	69.02	69.67	0.99

Tabelle 1: Vergleich der Energiemengen der unterschiedlichen Methoden und Lastprofile.

5 Kritische Betrachtung

Die grundlegende Funktionalität des beschriebenen hybriden Simulationsmodells wurde im Testszenario bestätigt.

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 und bezieht sich nur auf einen Fertigungsauftrag in unterschiedlichen Bearbeitungsmodi. Eine methodische Erweiterung der Seq2Seq-Methode ist daher notwendig, um auch unterschiedliche Fertigungsaufträge mit unterschiedlichen NC-Codes für das Training verwenden zu können. Ein Ansatzpunkt hierzu könnte die Verwendung synthetischer Trainingsdaten sein, welche kostengünstig und transparent zu generieren wären.

Im Erfolgsfall könnte dann eine Lösung entwickelt werden, die auf Basis nicht in den Trainingsdaten vorhandener NC-Codes plausible Stromverbrauchsprognosen für neue FAs generiert. Dies hätte ein hohes praktisches Potenzial und wäre auch aus wissenschaftlicher Sicht ein bedeutender Erfolg.

Darüber hinaus fehlt dem vorgeschlagenen Seq2Seq-Modell eine geeignete Bewertungsmethode für einzelne Zeitreihen eines FA [8]. Es gilt zu diskutieren, ob dies auch der Fall für die hier vorgestellten Gesamtzeitreihen ist. Der Abgleich in Abbildung 6 hat gezeigt, dass ein Gesamtlastprofil, welche aus einzelnen inferenzierten Lastprofilen erstellt wurde, mit Lastprofilen, welche rein aus dessen Trainingsdaten erstellt wurde, verglichen werden kann. Da Trainingsdaten zwingend für Parametrieren des ML-Modells vorhanden sein müssen, könnten diese in ihrer Gesamtheit, wie hier vorgestellt, zur Validierung des Lernverhaltens verwendet werden. Weiter weisen die generierten Gesamtlastprofile die gleiche Struktur im Hinblick auf ihren Gesamtzeitraum und Zeitstempel auf und können somit durch Methoden der Zeitreihenanalyse wie dem *Mean Absolute Error* oder der *Riemann'schen* Summe verglichen werden.

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] Bundesnetzagentur. 2005. Verordnung über die Entgelte für den Zugang zu Elektrizitätsversorgungsnetzen (Stromnetzentgeltverordnung -StromNEV). StromNEV.

[2] Benjamin Jacobsen und Maximilian Stange. 2020. Vorgehensmodell zur Simulation von gebündeltem Energiebedarf. In *Proceedings ASIM SST 2020*. ARGESIM Publisher Vienna, 295–301. DOI: https://doi.org/10.11128/arep.59.a59042.

[3] Bundesnetzagentur. 2005. *Verordnung über den Zugang zu Elektrizitätsversorgungsnetzen (Stromnetzzugangsverordnung - StromNZV)*. *StromNZV*.

[4] Navonil Mustafee, Sally Brailsford, Anatoli Djanatliev, Tillal Eldabi, Martin Kunc, und 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.

[5] Ilya Sutskever, Oriol Vinyals, und 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*. MIT Press, Cambridge, MA, USA.

[6] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, und 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.

[7] Benjamin Wörrlein, Sören Bergmann, Niclas Feldkamp, und 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.

[8] Benjamin Wörrlein und Steffen Straßburger. 2020. On the Usage of Deep Learning for Modelling Energy Consumption in Simulation Models. *SNE* 30, 4, 165–174. DOI: https://doi.org/10.11128/sne.30.tn.10536.

[9] Anna C. Römer, Martina Rückbrod, und Steffen Straßburger. 2018. Eignung kombinierter Simulation zur Darstellung energetischer Aspekte in der Produktionssimulation. In *ASIM 2018 : 24. Symposium Simulationstechnik*. ARGESIM/ASIM, Wien, 73–80.

[10] Thorsten Pawletta, Artur Schmidt, und Peter Junglas. 2017. A Multimodeling Approach for the Simulation of Energy Consumption in Manufacturing. *SNE* 27, 2, 115–124. DOI: https://doi.org/10.11128/sne.27.tn.10377.

[11] Anna C. Roemer und Steffen Strassburger. 2019 - 2019. Hybrid System Modeling Approach for the Depiction of the Energy Consumption in Production Simulations. In *2019 Winter Simulation Conference (WSC)*. IEEE, 1366–1377. DOI: https://doi.org/10.1109/WSC40007.2019.9004772.

[12] Rüdiger W. Brause. 1995. *Neuronale Netze. Eine Einführung in die Neuroinformatik* (2., überarbeitete und

erweiterte Auflage). Leitfäden der Informatik. Vieweg+Teubner Verlag, Wiesbaden.

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

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

[15] Klaus Müller und Tony Vignaux. 2002. *Simpy. a process-based discrete-event simulation framework* (2002). URL: http://simpy.readthedocs.org.

[16] Benjamin Wörrlein und Steffen Straßburger. 2020. Sequence to Sequence Modelle zur hochaufgelösten Prädiktion von Stromverbrauch. In *Proceedings ASIM SST 2020*. ARGESIM Publisher Vienna, 149–157. DOI: https://doi.org/10.11128/arep.59.a59021.

[17] C. M. Colson und M. H. Nehrir. 2009. An alternative method to load modeling for obtaining end-use load profiles. In *41st North American Power Symposium*. IEEE, 1–5. DOI: https://doi.org/10.1109/NAPS.2009.5484036.

[18] Noah D. Pflugradt. 2016. *Modellierung von Wasser und Energieverbräuchen in Haushalten*. Dissertation. Universitätsbibliothek Chemnitz, Chemnitz.

[19] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* (2015). URL: https://www.tensorflow.org.

[20] Shigeru Takahashi. 1955. Notes on the Riemann-sum. *Proc. Japan Acad. Ser. A Math. Sci.* 31, 1. DOI: https://doi.org/10.3792/pja/1195525845.