Dynamic Time Warping und Synthesedaten zur Validierung von Seq2Seq für die 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. Seq2Seq is a machine learning method that allows to translate sequences into other sequences. This method has been tried in hybrid simulation of machine tools. The method has been used to generate time series of energy consumption of jobs from the corresponding numerical control code that runs on a machine tool. Seq2Seq suffers from various problems. Firstly, the creation of training data is costly. Secondly, standard Seq2Seq metrics only allow for the evaluation of a prediction of one timestamp at a time, not an entire time series. Thirdly, training metrics are failing when vanilla data is used, as two identical numerical control codes can result in deviating time series. This causes confusion for the model in the training loop, as it is not clear which time series should be considered correct. Here we propose a holistic framework to all three problems, that contains synthetic data, additional metrics for time series and dynamic time warping.

Einführung

Sequence to Sequence (Seq2Seq) [3, 16] ist eine Klasse von Methoden des maschinellen Lernens (ML), welche es ermöglicht zwei Sequenzen unterschiedlicher Länge und unterschiedlicher Beschreibungen aufeinander abzubilden. Seq2Seq bedient sich hierbei künstlicher, neuronaler Netz und ist dem Deep Learning zugeordnet [4].

Im Kontext der Simulation wurde Seq2Seq schon erfolgreich verwendet um *numerical control codes* (NC-Codes) einer Werkzeugmaschine (WZM) in eine Zeitreihe des Energiebedarfs derselben Maschine zu übersetzen. Hierfür wird zuerst ein Trainingsdatensatz auf Basis einzelner Fertigungsaufträge (FA) erstellt. Dieser enthält pro FA einmal den NC-Code und weiter die gemessene Zeitreihe. Mit ebendiesem Datensatz wird das Seq2Seq-Modell trainiert. Das trainierte Modell kann nun mit einem NC-Code aktiviert werden und gibt anschließend eine Zeitreihe anhand des NC-Codes aus [25]. Weiter kann das Modell innerhalb einer hybriden Simulation verwendet werden, um beispielweise die Länge eines Fertigungsauftrages oder den Energiebedarf von Maschinen zu prognostizieren [22, 26].

Die Seq2Seq-Methode leidet an mehreren Problemen:

1. Trainingsdaten:

Die Beschaffung von Trainingsdaten ist kosten- und zeitintensiv [13, 21]. Gerade bei ML-Methoden ist dieser Effekt besonders stark spürbar, da die Menge an Trainingsdaten mit einem verbesserten Lernverhalten gleichgesetzt wird [5]. Als Lösungsansatz schlagen wir hier die Verwendung von synthetischen Daten vor [13, 21].

2. Fehlende Metriken für mehrdeutige Datensätze:

Trainingsdaten für maschinelles Lernen müssen im Regelfall eindeutig sein. Eindeutig heißt hier, dass für jede Stichprobe des Trainingsdatensatzes genau eine Lösung vorliegt. Diese Bedingung wird bei der Erstellung von Trainingsdaten, die NC-Codes und Zeitreihen erhalten, verletzt, wenn bei gleichem NC-Code eine (leicht) unterschiedliche Zeitreihe gemessen wird. Wir schlagen hier vor eine neue Lernmetrik einzuführen, welche die generierten Zeitreihen am Ende einer Trainingsperiode mit einer Vergleichszeitreihe vergleicht.

3. Keine Vergleichszeitreihen:

Generierte Zeitreihen mit Trainingszeitreihen zu vergleichen, scheint auf den ersten Blick trivial. Das Problem liegt in zwei Punkten begründet. Erstens vergleicht das Seq2Seq-Modell während des Trainings nicht komplette Zeitreihen, sondern jeden Datenpunkt in den Zeitreihen einzeln [3, 16]. Ein Vergleich von Gesamtzeitreihen findet nicht statt. Zweitens gibt es bei mehrdeutigen Datensätzen keine eindeutige Vergleichszeitreihe, welche verwendet werden könnte, um ganze Zeitreihen miteinander zu vergleichen. Wir schlagen vor, dieses Problem durch *Dynamic Time Warping* (DTW) [1, 15] zu lösen.

In den folgenden Kapiteln werden die Grundlagen der vorgeschlagenen Methode und ein darauf aufbauendes Konzept zur Lösung aller drei Probleme vorgeschlagen. Kern des Konzeptes ist es, nach jedem Trainingslauf eine

Anzahl von Zeitreihen anhand aller möglichen NC-Codes zu generieren. Anschließend werden die generierten Zeitreihen mit durch DTW erzeugten Referenzzeitreihen verglichen. Dies geschieht iterativ am Ende jedes Trainingslaufes. Der Ansatz wird daher hier als *Iterating over Metrics* (IOM) bezeichnet.

Im Anschluss werden die zur Implementierung notwendigen Bestandteile des Konzepts näher erläutert und die Ergebnisse der daraus abgeleiteten Methode vorgestellt.

1 Grundlagen des IOM-Ansatzes

Zum besseren Verständnis wird kurz auf die verwendeten Methoden des vorgestellten IOM-Ansatzes eingegangen.

1.1 Sequence-to-Sequence

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. Um zeitliche Muster verarbeiten zu können, müssen rekurrente Verbindungen im KNN vorhanden sein, welche eine Rückkopplung abstrahierten Wissens zulassen [27]. Solche rückgekoppelten bzw. rekurrenten neuronalen Netze (RNN) eignen sich besonders für Daten, welche in sequentieller Form vorliegen [4].

Handelt es sich bei den Daten 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 neuronale Schicht codiert, so ist dies ein Encoder. Wird eine Zielsequenz aus einer neuronalen Schicht heraus generiert, so wird dieser Teil als Decoder bezeichnet [4].

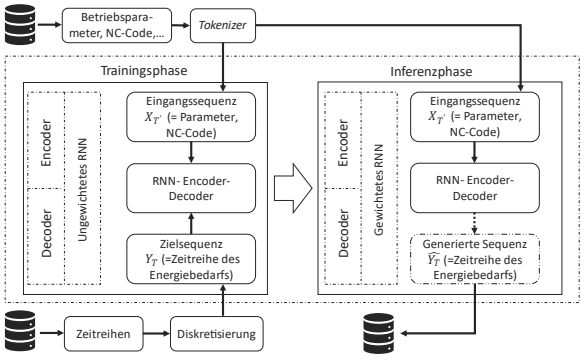

Abbildung 1: Bestandteile einer RNN-Encoder-Decoder Topologie [24].

Eine solche Topologie wird auch allgemein als rekurrente Encoder-Decoder-Netzwerke (RNN-ED) bezeichnet [4]. Abb. 1 beschreibt ebenso ein KNN für den Anwendungsfall von NC-Codes und Zeitreihen.

Beim Training werden sequenziell alle Daten eines Sequenzpaares $\{X_i, Y_i\}$ durchlaufen und aufeinander abgebildet. Zuerst wird die Eingangssequenz X_i in einen sogenannten Kontextvektor C encodiert. Dieser wird anschließend auf die Ausgangssequenz Y_i decodiert.

Das ML-Model lernt hierbei den Verlauf von Y_i in dem es den Zustand des Kontextvektors für jeden Eintrag $y_{i,T}$ aktualisiert. Die Änderung des Zustands des KNN wird hierbei durch eine Fehlerfunktion bestimmt. Das Modell versucht anschließend den Fehler zu minimieren. Der Fehlerwert beschreibt den Unterschied zwischen dem zu lernenden y_i und dem vorhergesagten Ergebnis $\hat{y}_i$. Die Vorhersage $\hat{y}_i$ geschieht in der letzten Schicht des KNN. In dieser wird eine sog. *Softmax*-Aktivierungsfunktion verwendet. Diese hat zum Nachtteil, dass sie je nur einen Eintrag von y_i mit der generierten Zeitreihe vergleichen kann [4].

Weiterführende Erläuterungen zum hier verwendeten Encoder-Decoder können [3, 4, 16] entnommen werden.

1.2 Synthesedaten durch Simulation

Ein grundlegendes Problem bei der Anwendung von Deep Learning ist die Verfügbarkeit von Trainingsdaten. Diese müssen gesammelt, auf Unstimmigkeiten hin untersucht, mit analytischen Methoden vorverarbeitet usw. werden. Dies geht teils mit hohen Kosten und Zeitaufwand einher. Im hier zugrundeliegenden Beispiel des Datensatzes von NC-Codes und Zeitreihen des Energiebedarfs von Fertigungsaufträgen wird dies deutlich.

Verfügt die betrachtete Werkzeugmaschine über keine passende Schnittstelle, muss der NC-Code und der Zeitraum, in dem dieser läuft, händisch aufgenommen und in ein Datenformat übertragen werden.

Noch aufwendiger verhält sich die Erstellung des Datensatzes der Energiezeitreihen. Hierfür muss eine geeignete Messtechnik und Software identifiziert, implementiert und schlussendlich angewandt werden. Dies stellt hohe Anforderung an verfügbares Personal, sowie die Verfügbarkeit von WZM und Messtechnik.

Sind die Voraussetzungen für eine Aufnahme des NC-Codes und der Zeitreihen erfüllt, stellt sich anschließend die Frage, ob die WZM im betrachteten Zeitraum eine Auslastung hat, die so hoch ist, dass damit eine

große Anzahl von Trainingsdaten erzeugt werden kann. Gerade im Hinblick auf die Größe der im Deep Learning verwendeten Datensätze (>10k) scheint dies unwahrscheinlich.

Abhilfe kann hier die Verwendung von Synthesedaten schaffen. Synthesedaten sind künstlich erzeugte Daten, welche in Struktur (Länge, Merkmalen, Merkmalshäufigkeit, etc.) den Originaldaten ähneln. Der Vorteil von Synthesedaten ist, dass diese kostengünstig, transparent und reproduzierbar erstellt werden können. Synthesedaten können durch ein Simulationsmodel generiert werden und anschließend einem ML-Model als Trainingsgrundlage dienen.

Die Verwendung von Synthesedaten im Maschinellen Lernen ist nicht neu und stellt ein eigenes Forschungsfeld dar [20]. So haben *Melo et. al.* [13] gezeigt, dass Synthesedaten zum Training von Bilderkennungsmodellen verwendet werden können. Weitere Anwendungsfälle für die Anwendung von Synthesedaten im Deep Learning sind bspw. die zerstörungsfreie Prüfung von Stahl [2], Objekterkennung [23], zur Erstellung von Fahrzeugbegrenzungsrahmen im autonomen Fahren [20] oder Fußgängererkennung in Bilddaten [7].

Auch die Erstellung von Synthesedaten durch Simulationstechniken ist kein Novum. So wurde Data Farming verwendet, um eine Datenbasis für Deep Learning in der Simulation von Produktionssystemen [9] oder der Objekterkennung für Roboter [19], zu schaffen. Weitere Anwendungsfälle waren bspw. die Erstellung von Mobilitätsdaten [8, 12], Bilddaten zur Herzgewebebestimmung [10] oder Daten der Fertigungsplanung und -steuerung [6].

1.3 Dynamic Time Warping

Zeitreihen beschreiben den Verlauf eines Merkmals über die Zeit, meist für eine feste Messstrecke, d.h. einem gleichen Intervall zwischen den einzelnen Messpunkten. Im Resultat ist eine Zeitreihe lediglich eine Sammlung von zeitlich sortierten Datenpunkten.

Der Vergleich von Zeitreihen miteinander erscheint zuerst wie ein triviales Problem. Vergleicht man die Zeitreihen einer festen Periode miteinander, beispielsweise der Temperaturverlauf über einen Tag, so könnte man die Zeitreihen für zwei Tage einfach übereinanderlegen, den Mittelwert dieser bilden, um das Temperaturmittel zu bestimmen; oder sie voneinander abziehen, um die Temperaturdifferenz zwischen zwei Tagen zu bestimmen.

Bei Zeitreihen, die keiner festen Periode folgen gestaltet sich der Vergleich von Zeitreihen schwieriger. Beispielsweise im hier genannten Anwendungsfall von Zeitreihen des Energiebedarfs eines FA. Werkzeugmaschinen sind dynamische Systeme, d.h. dass diese ihren Zustand bei jedem Ereignis ändern, was zur Folge hat, dass die Zeitreihen des gleichen NC-Codes und der gleichen Maschine unterschiedlich aussehen werden. Der Unterschied ist hier in beiden Dimensionen der Zeitreihe, der Anzahl von Datenpunkten und der Merkmalsausprägung der Zeitreihe festzustellen.

Möchte man nun solche Zeitreihen dynamischer Systeme miteinander vergleichen, stellen sich folgende Probleme.

Erstens haben die Zeitreihen eine unterschiedliche Anzahl an Datenpunkten. Die Länge einer Zeitreihe wird durch ihre Anzahl an Datenpunkten bestimmt. Besteht hier eine Abweichung können die zusätzlichen Datenpunkte einer Zeitreihe nicht mit den Datenpunkten der anderen Zeitreihe verglichen werden, da diese nicht vorhanden sind. Streicht man nun die zusätzlichen Datenpunkte, so könnte man die Zeitreihen wieder miteinander vergleichen, aber es würde potenziell wichtige Datenpunkte dabei verloren gehen. Alternativ könnte man die kürzere Zeitreihen mit Werten befüllen, bis sie die Länge der längeren Zeitreihe erreicht hat. Das Problem hierbei ist es, einen Wert zu finden mit dem man die kürzere Zeitreihe auffüllt, ohne dabei die Zeitreihe nachteilig zu ändern.

Zweitens können sich charakteristische Verläufe auf den Zeitreihen durch die fehlenden Datenpunkte in ihrer Lage verschoben haben. Solche Verläufe wären bspw. der Temperaturverlauf über den Tag mit einem Hoch zur Mittagszeit oder wie im hier genannten Anwendungsfall wiederkehrende Muster im Spannverlauf einer WZM.

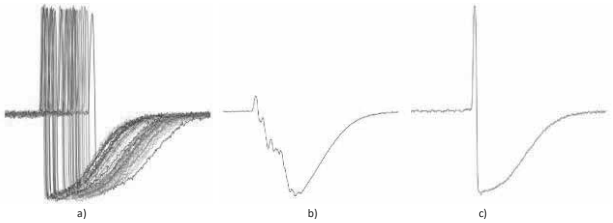

Abbildung 2: Vergleich von a) einer Menge an Zeitreihen, b) deren gemittelte Zeitreihe und c) eine durch DTW gemittelte Zeitreihe (Abbildungen entnommen aus [14]).

Betrachtet man Abb. 2 werden die oben genannten Probleme noch einmal verdeutlicht. Abb. 2 a) stellt hier

eine Menge an Zeitreihen dar, welche alle leicht unterschiedlich lang sind und deren Charakteristika (Steigung, Gefälle, Anstieg auf Ausgangswert) sich in Lage und Ausprägung unterscheiden. Würde man die Zeitreihen nur mit Werten auffüllen (hier dem Mittelwert der Zeitreihen) könnte man den Mittelwert aller Zeitreihen bilden. Abb. 2 b) stellt das Ergebnis dieser gemittelten Zeitreihe dar. Es ist erkennbar, dass diese gemittelte Zeitreihe, außer in der Länge, nicht mehr mit den Ausgangszeitreihen vergleichbar ist.

Eine Methode die Zeitreihen unterschiedlicher Länge mit unterschiedlicher Lage ihrer Charakteristika miteinander vergleichen kann heißt *Dynamic Time Warping* (DTW) [1, 15]. DTW ist ein Algorithmus, welcher eine Menge an Zeitreihen miteinander über eine sog. Distanzmatrix vergleicht. Die Distanzmatrix vergleicht die Datenpunkte der beiden Zeitreihen durch ein Distanzmaß, wie euklidische Distanz, Manhattan Distanz usw. Beim Vergleich von zwei Datenpunkten wird deren Unterschied in der Distanzmatrix notiert. Anschließend wird ein Pfad in der Distanzmatrix aufgespannt, der sog. *Warping Path* (siehe Linie in Abb. 3). Der optimale *Warping Path* ist der Pfad, der in Summe die geringste Distanz durch die Distanzmatrix aufweist.

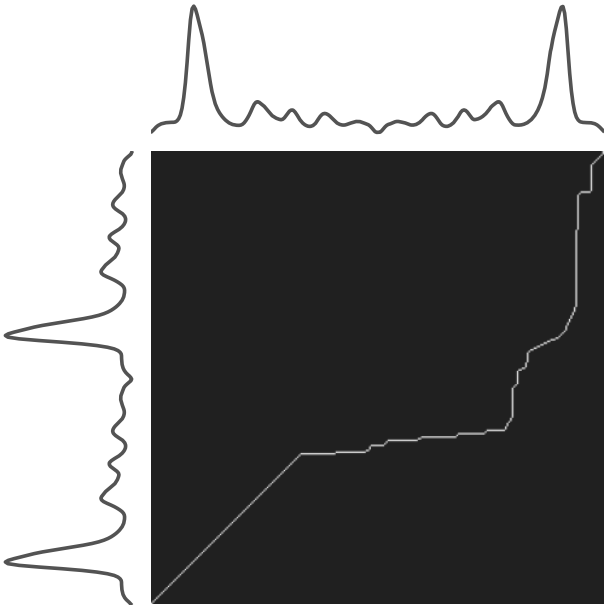

Abbildung 3: Distanzmatrix von zwei durch Dynamic Time Warping verglichenen Zeitreihen (links und oben zu sehen) [17].

Liegen die Datenpunkte nah beieinander, steigt der *Warping Path* in jeder Dimension gleich stark an (vgl. Abb. 3 *Warping Path* – links unten).

Weichen die Datenpunkte der beiden Zeitreihen voneinander ab, so flacht sich der *Warping Path* ab oder steigt stark an.

Wurde ein Warping Path gefunden, kann dieser verwendet werden, um eine durch DTW gemittelte Zeitreihe zu erstellen. Als Beispiel sei hier Abb. 2 c) angeführt. Die hier gezeigte Zeitreihe ist in ihrer Länge und Form klar mit den einzelnen Zeitreihen aus Abb. 2 a) vergleichbar.

Weiterführende Literatur zu Dynamic Time Warping kann [1, 14, 15, 17] entnommen werden.

2 Konzept

Der IOM-Ansatz besteht aus mehreren Schritten (vgl. Abb. 4).

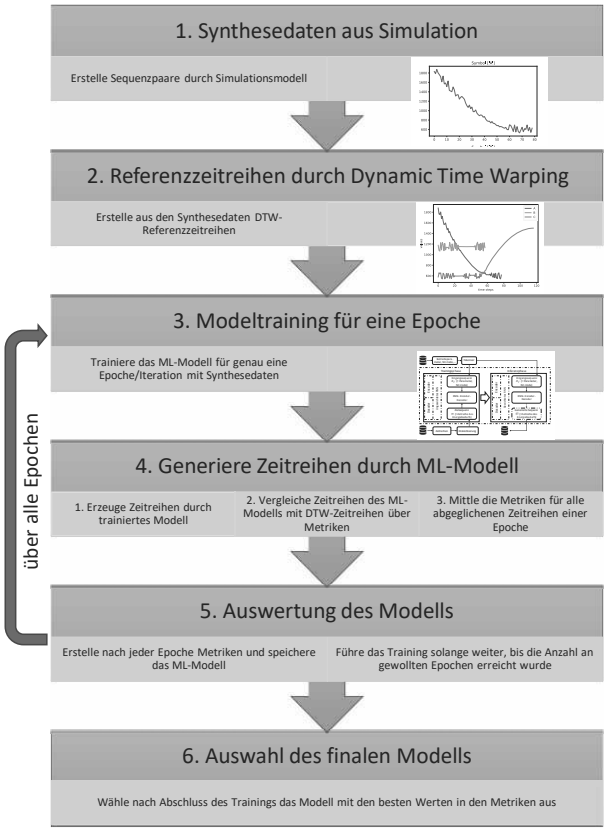

Abbildung 4: Konzept des IOM-Ansatzes

Zuerst, um dem Kosten- und Verfügbarkeitsproblem von Trainingsdaten zu entgehen, werden diese über ein Simulationsmodel erstellt.

Im zweiten Schritt werden Referenzzeitreihen erstellt. Diese dienen dazu, durch das Model erzeugte Zeitreihen vergleichbar zu machen. Dies geschieht anhand von *Dynamic Time Warping*, wobei die Datengrundlage die zuvor erzeugten Synthesedaten darstellen.

Drittens wird nun das Training des ML-Models gestartet. Das Model trainiert genau für eine Iteration (Anm.: Zur Verringerung der Berechnungszeiten wurden 10 Epochen in einer Iteration zusammengefasst).

Im vierten Schritt wird das trainierte Model verwendet, um Zeitreihen zu generieren. Der Vergleich von kompletten Zeitreihen ist nötig um mit dem in Kapitel 1.1 beschrieben Problem umzugehen, dass die im Model verwendete *Softmax*-Funktion nur schrittweise einzelne Datenpunkte vergleicht, nicht aber eine gesamte Zeitreihe.

Die Güte der generierten Zeitreihen wird anschließend über einen Vergleich mit den durch DTW erzeugten Referenzzeitreihen bestimmt. Wurden über Metriken Vergleiche für alle betrachteten Zeitreihen erstellt, werden diese im Anschluss zusammengefasst. Hierbei wird der Mittelwert jeder einzelnen Metrik für alle Zeitreihen gebildet.

Im anschließenden fünften Schritt geht das Model wieder ins Training über und wiederholt Schritt 3 und 4 so lange bis eine vorher festgesetzte Anzahl an Epochen erreicht wurde. Auch hier wird wieder bei jedem Durchlauf eine Zusammenfassung der Metriken erstellt und das trainierte Model gespeichert.

Im sechsten und letzten Schritt wird das Modell ausgewählt, welches die besten Resultate in den betrachteten Metriken hatte.

3 Versuchsvorbereitung

Zur Umsetzung des IOM-Konzeptes ist es nötig verschiedene Vorbereitungsschritte durchzuführen. Diese sind einerseits die Erstellung von Synthesezeitreihen. Weiter müssen Vergleichszeitreihen über *Dynamic Time Warping* erstellt werden. Anschließend müssen die im vorhergehenden Kapitel genannten Metriken erstellt und in das Modeltraining eingebunden werden.

3.1 Erstellung Synthesezeitreihen

Zur Erstellung von Synthesezeitreihen wird das Simulationstool *Anylogic* (vgl. Abb. 5) verwendet. In *Anylogic* wird ein diskret ereignisorientiertes Simulationsmodel angelegt. In der Quelle des Models werden zufällig Fertigungsaufträge (A, B oder C) erzeugt. Diese gelangen über eine Warteschlange in einen Warteraum. Es kann sich immer nur ein FA im Warteraum befinden. Jedem FA wird eine eigene Wartezeit zugeordnet. Die Wartezeit im Warteraum folgt einer stetigen Gleichverteilung von [0.95;1]. Sobald der Warteraum mit einem FA belegt ist, wird eine dem FA zugehörige Funktion ausgeführt. Diese Funktionen unterscheiden sich für alle drei Typen von FA (vgl. Abb.6). Es handelt sich hier um mathematische Funktionen, welche als Eingangsparameter die verbleibende Wartezeit des FA erhalten. Die Ausgaben der Funktionen ändern sich daher über die Zeit. Der berechnete Wert wird anschließend durch eine weitere Gleichverteilungsfunktion geführt und ausgegeben.

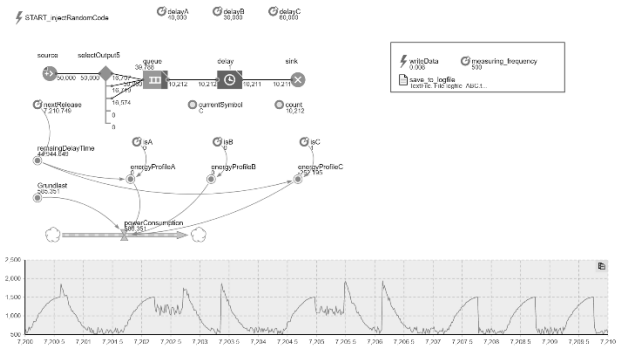

Abbildung 5: *Anylogic* Simulationsmodel.

Durch die Gleichverteilung im Warteraum und in den Ausgabefunktionen wird sichergestellt, dass die Zeitreihen gleicher FA sich jedes Mal leicht unterscheiden. Dies trägt dem eingangs beschriebenen Problem Rechnung, dass sich zwei gemessene Zeitreihen des gleichen FA nie exakt in Länge und Werteverlauf gleichen. Abb. 6 stellt Stichproben der erzeugten Zeitreihen dar. Die Gleichverteilung in den Ausgabewerten ist hier klar zu erkennen.

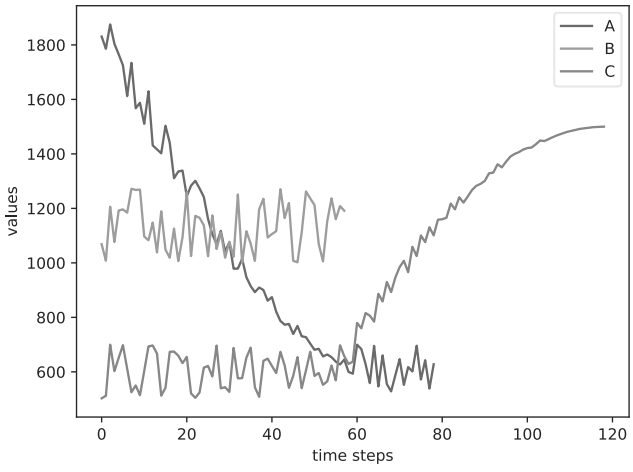

Abbildung 6: Stichprobe der durch *Anylogic* erzeugten Zeitreihen.

3.2 Generierung der Referenzzeitreihen durch Dynamic Time Warping

Zur Erstellung von Referenzzeitreihen wird *Dynamic Time Warping* verwendet. DTW erzeugt aus einer Menge

von Zeitreihen eine Referenzzeitreihe, welche in Länge und Struktur einen minimalen Gesamtfehler zur Ausgangsmenge aufweist. Die Berechnung erfolgt anhand der Python Bibliothek *tslearn* [17]. Der spezifisch hier verwendete DTW Algorithmus ist softDTW [11]. softDTW weist eine höhere Fehlertoleranz in der Erstellung des *Warping Path* auf, welche sich positiv auf die Erstellung von gemittelten Zeitreihen auswirkt [11].

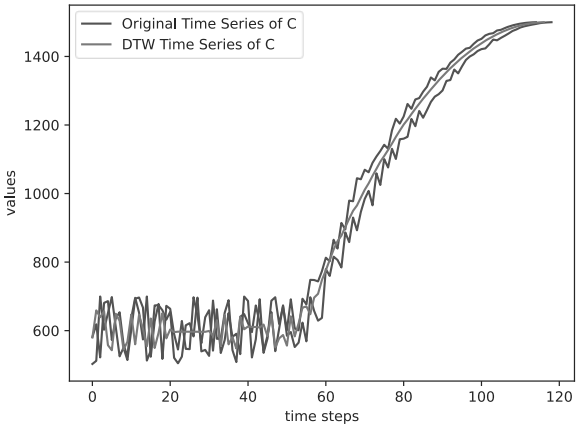

Abbildung 7: Vergleich von 2 Zeitreihen des Fertigungsauftrags C mit der für C durch softDTW erzeugten Zeitreihe.

Abb. 7 zeigt eine durch softDTW erzeugte Zeitreihe im Vergleich zu zwei Stichproben des Fertigungsauftrages C. Es ist deutlich zu erkennen, dass die softDTW Zeitreihe die Stichproben gut zusammenfasst, diese aber nicht einfach kopiert, sondern in Form und Länge nachahmt.

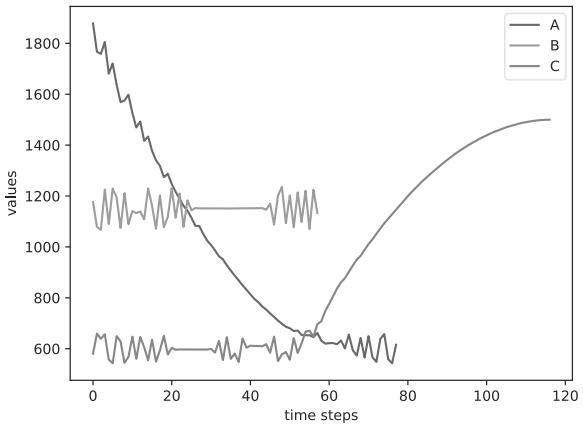

Abbildung 8: Mit softDTW erzeugte Zeitreihen der drei Fertigungsaufträge.

Abb. 8 zeigt die softDTW Zeitreihen für alle 3 FA. Im Vergleich mit Abb. 6 wird der Unterschied zwischen den Ausgangs- und DTW-Zeitreihen deutlich. Die Zeitreihen weisen alle eine ähnliche Länge und Form auf, unterscheiden sich aber leicht, da sie aus der gesamten Menge an Zeitreihen berechnet wurden.

3.3 Einbinden der Metriken in das Seq2Seq-Modell

Während des Modelltrainings sollen die generierten Zeitreihen mit den DTW-Zeitreihen als Referenz verglichen werden. Im Anschluss daran werden die Werte der Metriken der einzelnen Zeitreihen für eine gesamte Trainingsepoche gemittelt.

Hierfür wurden zwei Metriken verwendet. Einmal wird der *mean squared error* (MSE) angewandt. Dieser vergleicht die Zeitreihen, indem er die Differenz zwischen jedem einzelnen Datenpunkt berechnet und diese anschließend quadriert. Im Anschluss werden alle Differenzen miteinander addiert, um eine Aussage über die Güte der Gesamtzeitreihe zu erhalten. Die Quadratur der Differenz hat zwei Vorteile. Erstens werden alle Differenzen positiv, was in der anschließenden Addition dazu führt, dass sich ein negativer und positiver Fehlerwert nicht gegenseitig auslöschen. Zweitens werden durch die Quadratur große Fehler stärker gewichtet als einzelne.

Die zweite Metrik *sigma length* betrachtet lediglich eine Dimension der Zeitreihe, nämlich ihre Länge. Hier wird die Anzahl der Datenpunkte in beiden Zeitreihen ermittelt und anschließend durcheinander geteilt. Desto näher das Ergebnis an 1 liegt, desto ähnlicher sind sich generierte und Referenzzeitreihe in ihrer Länge.

```
def results(ml_ts, dtw_ts, iteration):

  mean_squared_error_accuracy =
  mean_squared_error(ml_ts, dtw_ts)

  sigma_length_accuracy =
  len(ml_ts) / len(dtw_ts)

  results =
  {"Iteration": iteration,
  "MSE": mean_squared_error_accuracy,
  "Sigma_Length": sigma_length_accuracy}

  return results
```

Code 1: Pseudocode der Metriken MES und *sigma length* für eine Zeitreihe (ml_ts steht für die durch das ML-Model erzeugten Zeitreihen; dtw_ts für die durch DTW erzeugten.

Wird ein Trainingslauf beendet, werden mit dem trainierten Modell Zeitreihen erzeugt. Diese werden anschließend mit denen in Code 1 beschriebenen Metriken

verglichen und die Resultate der Metriken für jede Zeitreihe gespeichert. Anschließend wird nun eine Zusammenfassung für die gesamte Iteration berechnet. Dies soll eine ganzheitliche Aussage über das Modell zu einem bestimmten Trainingsstand ermöglichen. Hierbei werden alle MSE und *sigma length* Resultate gemittelt. Der Mittelwert erlaubt eine robustere Aussage über die Fähigkeit des Modells Zeitreihen zu erzeugen als die Betrachtung von Einzelzeitreihen.

4 Versuchsdurchführung

Die Erstellung der Synthesezeitreihen wurde in *Anylogic* durchgeführt. Als Programmiersprache wurde *Python* verwendet. Die Berechnung der DTW-Zeitreihen geschah in *tslearn* [17], die Erstellung des ML-Models in *Tensorflow* [18]. Als Hardware standen zur Verfügung ein *Ryzen* 7 2700 X Prozessor, eine RTX 2080Ti Grafikkarte mit 4352 Cuda Kernen, sowie eine SSD mit 480 GB Speicher.

Der IOM-Ansatz sieht vor das ML-Model nach jeder Epoche zu speichern. Dies konnte aber mit der vorhandenen Speicherkapazität nicht umgesetzt werden. Daher wurde entschieden das Model je 10 Epochen (=1 Iteration) trainieren zu lassen und nur für die letzte Epoche einer Iteration das Modell zu speichern und Metriken zu erheben. Das Modell trainierte für 1000 Epochen.

Bei den Trainingsdaten handelt es sich um den in Kapitel 3 beschriebenen, wobei die Sequenzpaare aus jeweils 2 FA gebildet wurden, um die Varianten im Datensatz zu erhöhen. Der Datensatz hatte eine Größe von 10000 Sequenzpaaren.

4.1 Abgleich der Metriken

Wie im IOM-Konzept beschrieben, sollen am Ende jeder Iteration gemittelte Metriken gebildet werden. Der Verlauf dieser Metriken wurde in Abb. 9 aufgetragen. Hier werden einmal Ergebnisse des MSE und des sigma length dargestellt. Um nun das optimale Model auszuwählen, wird das Optimum der beiden Metriken bestimmt. Das Optimum einer Metrik ist der Wert, in dem die Metrik das beste Resultat erzielt. Bei MSE wäre dies 0, da MSE die Differenz zwischen allen Datenpunkten betrachtet. Bei *sigma length* hingegen wäre es 1, da hier ein Verhältnis der Länge einer Zeitreihe zur anderen untersucht wird. Zur Veranschaulichung wurde das Optimum bei *sigma length* grafisch eingetragen (blaue Linie). Alle Iterationen, die nun auf der blauen Linie liegen, stellen Modelle dar, die in der *sigma length* Metrik den optimalen Wert erreicht haben.

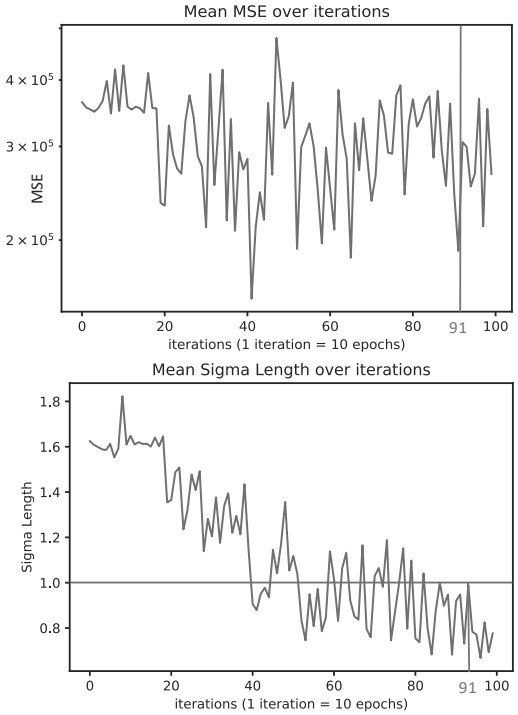

Abbildung 9: Metriken des IOM-Ansatzes. Oben: gemittelter MSE. Unten: gemittelter *sigma length*. Die blaue Linie markiert das Optimum von sigma length (=1). Die grüne Linie ist die ausgewählte Iteration.

In einem zweiten Schritt werden nun Iterationen, in denen $sigma\ length = 1$ gilt auf ihren Wert in der MSE-Metrik hin untersucht. Die grüne Linie in Abb. 9 bei Iteration 91 (=Epoche 910) entspricht der Iteration, bei der die Werte für beide Metriken am niedrigsten sind. Es wird daher das Model der Epoche 910 ausgewählt.

4.2 Visualisierung der Inferenz- und Dynamic Time Warping Zeitreihen

Das Model kann nun verwendet werden, um Zeitreihen zu generieren. Abb. 10 zeigt alle 9 möglichen Varianten auf die das ML-Modell trainiert wurde. Die blaue Zeitreihe wurde vom ML-Modell generiert. Die orange Zeitreihe ist die durch DTW erzeugte Referenzzeitreihe. Der visuelle Abgleich bestätigt das Funktionieren des IOM-Ansatzes.

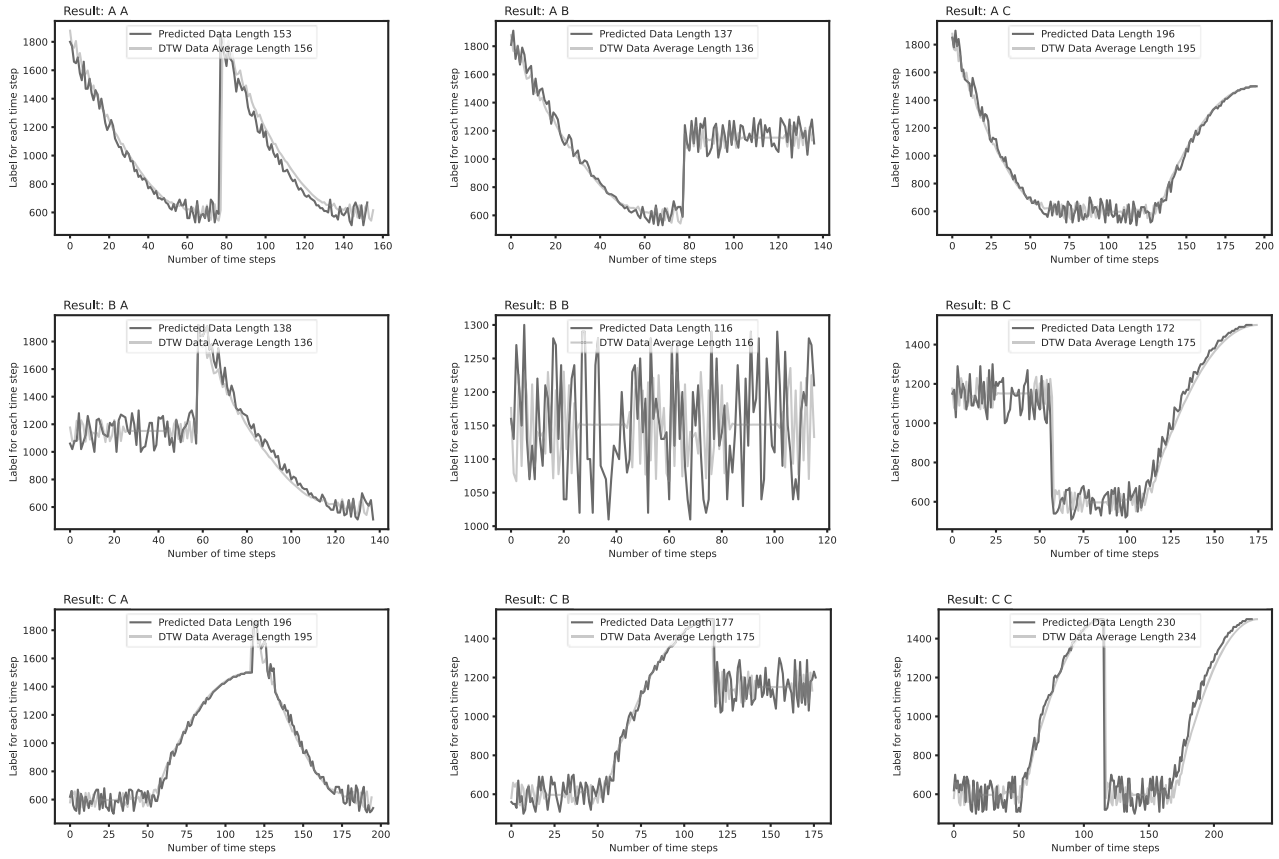

Abbildung 10: Vergleich aller bei Epoche 910 erstellten Zeitreihen mit den DTW-Referenzzeitreihen. Die Legende in den Varianten gibt die Längen der beiden Zeitreihen wieder.

5 Kritische Betrachtung

Der vorgestellte IOM-Ansatz konnte die an ihn gestellten Anforderungen erfüllen.

Einerseits wurde gezeigt, dass durch Simulation erzeugte Synthesedaten für das Training von ML-Modellen verwendet werden können. Diese konnten kostengünstig, transparent und auf den Anwendungsfall passend erzeugt werden.

Weiter wurde durch die Implementierung von speziellen Metriken erreicht, dass nun Gesamtzeitreihen miteinander verglichen werden konnten und nicht nur wie im Seq2Seq Basis Model einzelne Datenpunkte über die *Softmax*-Funktion.

Daraufhin wurde gezeigt, dass DTW-Zeitreihen dazu verwendet werden können, Referenzzeitreihen für den Abgleich in den Metriken zu bilden. Die Verwendung von DTW war nötig, um das Problem der Nichteindeutigkeit in einer Menge von Zeitreihen, mit gleichen Beschreibungen (bspw. gleicher NC-Code) zu lösen.

Abschließend konnten so Zeitreihen generiert werden, welche sich zu den analytisch erzeugten Dynamic Time Warping Zeitreihen kaum unterscheiden. Die Zeitreihen sind in Form und Länge fast identisch.

Die Methode selbst bietet Potential für weitere Forschungsfragen. So könnte sie weiter validiert werden, indem der Datensatz um ein Vielfaches vergrößert oder mehrere verschiedene FA in der Simulation erzeugt werden. Weiter hat das ML-Model hier nur Zeitreihen von NC-Codes erzeugt, welche schon in den Trainingsdaten vorhanden waren. Es gilt zu untersuchen inwieweit das Model Zeitreihen generieren kann, im Falle eines unbekannten NC-Codes.

Nachteil der vorgestellten Methode ist die Anforderung an Rechen- und Speicherleistung. So muss für jede Iteration ein Modell gespeichert werden, obwohl nach Bestimmung des optimalen Modells alle anderen gelöscht werden können. Auch die Berechnung der Metriken übersteigt bei epochenweiser Erstellung die hier für das Training des Modells aufgewandte Zeit.

6 Literaturverzeichnis

[1] Donald J. Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series. In *KDD workshop*, 359–370.

[2] Aleksei Boikov, Vladimir Payor, Roman Savelev, and Alexandr Kolesnikov. 2021. Synthetic Data Generation for Steel Defect Detection and Classification Using Deep Learning. *Symmetry* 13, 7, 1176. DOI: https://doi.org/10.3390/sym13071176.

[3] 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.

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

[5] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md M. A. Patwary, Yang Yang, and Yanqi Zhou. 2017. *Deep Learning Scaling is Predictable, Empirically*.

[6] Sanjay Jain, Anantha Narayanan, and Yung-Tsun T. Lee. 2019. Infrastructure for Model Based Analytics for Manufacturing. In *2019 Winter Simulation Conference (WSC)*. IEEE. DOI: https://doi.org/10.1109/wsc40007.2019.9004893.

[7] Hadi Keivan Ekbatani, Oriol Pujol, and Santi Segui. 2017. Synthetic Data Generation for Deep Learning in Counting Pedestrians. In *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods*. SCITEPRESS - Science and Technology Publications. DOI: https://doi.org/10.5220/0006119203180323.

[8] Amir T. Kutyev and Yana A. Bekeneva. 2021. Simulation Software for Generating Data in Monitoring Systems. In *Proceedings of the 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). January 26-28, 2021 : St. Petersburg and Moscow, Russia, 2021*. IEEE, Piscataway, NJ, 486–489. DOI: https://doi.org/10.1109/ElConRus51938.2021.9396098.

[9] Tobias Lechler, Martin Sjarov, and Jörg Franke. 2021. Data Farming in Production Systems - A Review on Potentials, Challenges and Exemplary Applications. *Procedia CIRP* 96, 230–235. DOI: https://doi.org/10.1016/j.procir.2021.01.156.

[10] Michael Loecher, Luigi E. Perotti, and Daniel B. Ennis. 2021. Using synthetic data generation to train a cardiac motion tag tracking neural network. *Medical image analysis* 74, 102223. DOI: https://doi.org/10.1016/j.media.2021.102223.

[11] Marco Cuturi and Mathieu Blondel. 2017. Soft-DTW: a Differentiable Loss Function for Time-Series. In *Proceedings of the 34th International Conference on Machine Learning*. Proceedings of Machine Learning Research. PMLR, 894–903.

[12] Pasqual Martí, Jaume Jordán, Javier Palanca, and Vicente Julian. 2022. Charging stations and mobility data generators for agent-based simulations. *Neurocomputing* 484, 196–210. DOI: https://doi.org/10.1016/j.neucom.2021.06.098.

[13] Celso M. de Melo, Antonio Torralba, Leonidas Guibas, James DiCarlo, Rama Chellappa, and Jessica Hodgins. 2022. Next-generation deep learning based on simulators and synthetic data. *Trends in cognitive sciences* 26, 2, 174–187. DOI: https://doi.org/10.1016/j.tics.2021.11.008.

[14] François Petitjean, Germain Forestier, Geoffrey I. Webb, Ann E. Nicholson, Yanping Chen, and Eamonn Keogh. 2016. Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. *Knowl Inf Syst* 47, 1, 1–26. DOI: https://doi.org/10.1007/s10115-015-0878-8.

[15] Petitjean François, Forestier Germain, Webb Geoffrey I., Nicholson Ann E., Chen Yanping, and Keogh Eamonn. 2014. Dynamic Time Warping Averaging of Time Series Allows Faster and More Accurate Classification. In *2014 IEEE International Conference on Data Mining*, 470–479. DOI: https://doi.org/10.1109/ICDM.2014.27.

[16] 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*. MIT Press, Cambridge, MA, USA.

[17] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. 2020. Tslearn, A Machine Learning Toolkit for Time Series Data. *Journal of Machine Learning Research* 21, 118, 1–6.

[18] TensorFlow Developers. 2022. *TensorFlow*. Zenodo.

[19] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World*.

[20] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil,

Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. 2018. *Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization*.

[21] Aleksei Triastcyn and Boi Faltings. 2018. *Generating Artificial Data for Private Deep Learning*.

[22] Benjamin Woerrlein and Steffen Strassburger. 2020. A Method for Predicting High-Resolution Time Series Using Sequence-to-Sequence Models. In *2020 Winter Simulation Conference (WSC)*. IEEE, 1075–1086. DOI: https://doi.org/10.1109/WSC48552.2020.9383969.

[23] Matthew Z. Wong, Kiyohito Kunii, Max Baylis, Wai H. Ong, Pavel Kroupa, and Swen Koller. 2019. Synthetic dataset generation for object-to-model deep learning in industrial applications. *PeerJ. Computer science* 5, e222. DOI: https://doi.org/10.7717/peerj-cs.222.

[24] Benjamin Wörrlein and 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.

[25] Benjamin Wörrlein and 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.

[26] Benjamin Wörrlein and Steffen Straßburger. 2022. Hochaufgelöste Energieprofile durch hybride Simulation. In *ASIM SST 2022 Proceedings Langbeiträge. 26. ASIM Symposium Simulationstechnik, 25.07.-27.07.2022, TU Wien*. ASIM Mitteilung, 180. ARGESIM Verlag, Wien, 243–251. DOI: https://doi.org/10.11128/arep.20.a2004.

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