Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

A METHOD FOR PREDICTING HIGH-RESOLUTION TIME SERIES USING SEQUENCE-TO-SEQUENCE MODELS

Benjamin Woerrlein
Steffen Strassburger

Information Technology in Production & Logistics
Ilmenau University of Technology
P.O. Box 100 565
Ilmenau, 98684, GERMANY

ABSTRACT

With the increasing availability of data, the desire to interpret that data and use it for behavioral predictions arises. Traditionally, simulation has used data about the real system for input data analysis or within data-driven model generation. Automatically extracting behavioral descriptions from the data and representing it in a simulation model is a challenge of these approaches. Machine learning on the other hand has proven successful to extract knowledge from large data sets and transform it into more useful representations. Combining simulation approaches with methods from machine learning seems therefore promising to combine the strengths of both approaches. Representing some aspects of a real system by a traditional simulation model and others by a model incorporating machine learning**,** a hybrid system model (HSM) is generated. This paper suggests a specific HSM incorporating a deep learning method for predicting the anticipated power usage of machining jobs.

1 INTRODUCTION

For a computer simulation of a real system it is indispensable to create a model of this system. Systems models are generally abstractions of the real-world system under observation and will focus on the most relevant parts, or attributes thereof, which are of interest to the model designer. In traditional, monolithic modeling approaches, the modeler is bound and potentially limited by the chosen modeling paradigm.

Machine learning (ML), in contrast to simulation, is a set of algorithms that provides an effective way to aggregate rather big data sets and to find patterns within that data (Goodfellow et al. 2016). Those patterns can then further be used to describe the dynamic behavior of the system of interest that emitted the initial data points. Applications of ML are not limited to sets of static data, as the most prominent picture classification tasks, but can also be applied to dynamic data sets such as time series data. This duality results in machine learning methods being a promising match for hybrid systems modeling, since a chosen simulation methodology can be complemented by a machine learning method with a different methodology and vice versa.

While machine learning methods are no simulation technique by definition, they can be used to design a data driven model as a constituent part of a hybrid systems model (HSM) (Mustafee et al. 2017). In this paper we propose such a HSM that combines discrete event simulation (DES) with Sequence2Sequence (Seq2Seq) neural networks. This newly proposed HSM focuses on the realistic depiction of anticipated power usage of a job in a manufacturing cell that contains a waiting room and a machine tool.

The investigation of energy efficiency issues within simulation has become a widespread research approach. Existing studies are often based on the consideration of the power usage of resources (machines, furnaces, ...) by means of metrologically recorded operating conditions, which are regarded as constant over a defined period of time (Haag 2013; Thiede 2012). The power usage of resources averaged over a period of time is then assigned to a resource state and can be mapped and analyzed status-based with discrete-event simulation approaches. There are some application scenarios for which the usage of these quasi-static operating states provides sufficient accuracy, but others, for which this is clearly not sufficient. The determination and smoothing of load peaks in production systems with many resources is an example for the latter. A partial solution for this is presented in Römer et al. (2018). It is based on the basic idea of combined simulation, which is also proposed, for example, in Peter and Wenzel (2015). While the production and logistics part of the model is represented classically with discrete-event simulation, the system dynamics approach is applied for electricity usage in Römer et al. (2018). This allows the time series of the actually measured power usage to be reproduced in high resolution in the simulation. This offers the advantage of a high-resolution overall picture of the production power consumption.

However, the disadvantage is that only the power usage of measured jobs can be reproduced. Power usage of unknown job types cannot be predicted without prior measurement on the real system. Furthermore, the approach described in Römer et al. (2018) does not depict cause-effect relationships between control parameters (e.g., half feed, slower heating phase, ...) and the resulting power usage.

Within the scope of this paper, we will therefore examine whether there are alternative possibilities for high-resolution forecasting of electricity usage that can overcome the disadvantages mentioned above. The focus of the investigations is on the field of deep learning. More specifically, we propose a method that enables to predict time series for the power usage of upcoming jobs by means of appropriately trained artificial neural networks (ANN). The basic idea here is to train an ANN with relevant control information (here: numerical control codes of the production jobs of a machine tool and machine states they are operated in) on to a high-resolution time series of power consumption measured for previous jobs.

In perspective, the ANN can then forecast a time series of the expected power usage for any job, possibly even a job with deviating numerical control codes, etc. These time series could then be used in hybrid simulations of the entire production system.

This paper presents a framework for the outlined procedure as well as a prototypical implementation and validation. The paper is structured as follows: Section 2 discusses related work concerning the combination of simulation and ML and introduces the specific machine learning approach used. Here the basic idea of a deep learning method, which can map asynchronous sequences of different lengths to each other, is presented. Section 3 showcases the particular concept of such a Seq2Seq-model that predicts high-resolution time series, along with the data sequences of interest. Further will the integration of such values within a hybrid systems model be briefly discussed here as well. Building on this, Section 4 discusses the makeup and necessary preprocessing steps of the data, which are required to lead to conclusive results of the model. The results of the prototypical application of the Seq2Seq-model are demonstrated and evaluated in section 5. A critical review of the results and a discussion of future work is given in section 6.

2 RELATED WORK

2.1 Hybrid System Models

The need for data driven decision making in a dynamic environment results in a need for methods that allow simulation models to adapt over time by learning (Biller et al. 2017). Classical simulation approaches, such as discrete event simulation (DES), have traditionally used data about the real system either manually within the modeling process, e.g., in the context of input data analysis for modeling stochastic influences by fitting theoretical distributions to the real observations, or semi-automatically within data-driven model generation approaches for depicting structural aspects of the model (Bergmann et al. 2012). Automatically extracting behavioral descriptions from the data and representing it in a simulation model can be considered a weak point of automatic simulation model generation approaches (Bergmann and Strassburger 2010).

Previous work has therefore focused on combining ML with traditional simulation modeling for alleviating this weakness. Examples include Bergmann et al. (2014) and Bergmann et al. (2017) which present an approach for using trained artificial neural networks that can be called from material flow simulations to obtain a decision on which control strategy to apply within the simulation, depending on certain input parameters modeled in the simulation model.

Other examples include Rabe and Dross (2015), where Reinforcement Learning was used alongside a simulation-based Decision Support System for logistics networks. Here the actions of an agent were modeled through ML, to finally identify and select principles on which decision-making policies should be carried out by the agent.

In Morin et al. (2015) a set of machine learning classification techniques is proposed as a method to generate metamodels for the simulation of sawmilling processes. Here data driven models of the sawing process are generated and used to determine what sets of lumber are derived from breaking down the logs in a sawing mill.

Within patients care pathway design for hip fracture, ML was used to identify clusters of patients, and their underlying characteristics, and to use that insight in the development stage of a simulation model (Elbattah et al. 2018). ML was here used to cluster a set of patients into subgroups, that relate in risks of treatment for fractured hips.

Finally, ML is a key constituent in the modeling of a digital twin, as it is stipulated for symbiotic simulation approaches, as in Onggo et al. (2018). Here, ML enables a digital twin, that is a virtual representation of a physical system, as it allows the systems simulation model under observation to adapt according to the behavior of variables emitted by the physical system in question. Further such hybrid simulation-ML environments can be used to predict the changes in state variables of as system, as machine learning methods can be trained on past changes in the same system.

The aforementioned examples have in common, that they allow the representation of certain isolated decisions, in regard to events or entities, by a machine learning model and to include that decision within the simulation. A different, widely uninvestigated area is the inclusion of entire time-series data delivered from a ML model into a simulation model. This new approach is in contrast to classical time series data analytics and prediction in simulation modeling, that have been discussed extensively, e.g., in (Mustafee et al. 2018), where time series data was used to generate wait time predictions.

A key application of such approaches would be *peak shaving* of energy demands, where the objective is to smoothen the time series of energy consumption over a distinct time. Here the total consumption of an energy consuming process is considered to be equal to the sum of dynamic sub-processes of the system in question. A method as Sequence-to-Sequence can help to tackle the description of such subsystems as it allows to model dynamic sequences.

2.2 Sequence-to-Sequence Models

Artificial neural nets (ANN) are used to identify patterns in complex data structures. For this purpose, embedding layers of a neural network hold the data under observation and guide it as information through the hidden layers of an ANN. Hidden layers consist of hidden units, the actual neurons. These neurons are self-parameterizing units. The more hidden layers an ANN contains, the higher the degree of abstraction of the recorded information can be. If an ANN has more than one hidden layer, it can link abstractions gained in one layer to another layer, thus creating a more complex abstraction with each added layer. This deep staggering of neuronal layers is commonly referred to as deep learning (Goodfellow et al. 2016).

If patterns change over time, this temporal sequence of patterns is understood as a sequence. For an ANN to be able to process temporal patterns, recurrent connections must be present in the network topology that allow feedback of abstract knowledge (Brause 1995; Zell 2003). Such feedback or recurrent neural networks (RNN) are particularly suitable for data which is presented in sequential form (Goodfellow et al. 2016).

If the inputs to a deep learning method are sequences, one speaks of Sequence-to-Sequence (Seq2Seq) architectures (Cho et al. 2014; Sutskever et al. 2014). Here, the embedding layer of an RNN encodes an input sequence. If the input sequence, as abstraction, is encoded into a specific neuronal layer, one speaks of an encoder. If a target sequence is generated from the abstraction of a neuronal layer, this part of a network topology is called a decoder (Goodfellow et al. 2016).

The Encoder-Decoder model as described in Figure 1 encodes a sequence x_T of T values into a *summary* vector C that is then decoded into a sequence $y_{T'}$ of T' values. The Encoder and Decoder are conjoined by the fixed sized vector C (Cho et al. 2014).

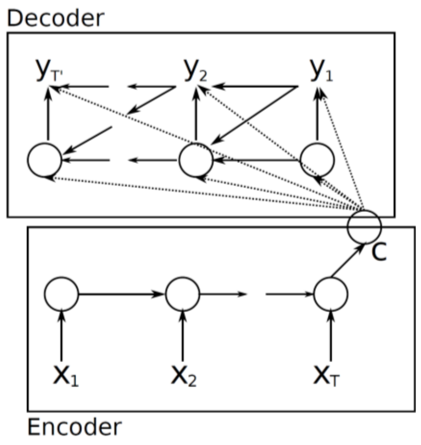


Figure 1: The Encoder-Decoder model as proposed by Cho et al. (2014). Note that *T´≠T*.

The general workflow to train such a model is to read the values of $X_i = (x_1, \dots, x_T)$ sequentially whilst updating the hidden state of the encoder accordingly (Cho et al. 2014; Sutskever et al. 2014).

Once all the values of X_i have been processed, the last hidden state is encoded into the summary vector C. The decoder now has two inputs C and $Y_i = (y_1, \dots, y_{T'})$ and learns the conditional distribution between them by updating its hidden state whilst reading in the values of Y_i and C accordingly. Here, the hidden state is linked to a *Softmax*-layer holding the unique tokens found in the training set $\mathbb{Y}$. Once training is finished the decoder can be initialized by a sequence X_i, which is mapped to C, and generates a sequence Y_i therefrom (Cho et al. 2014; Sutskever et al. 2014).

As the model learns to generate the next token y_{t+1} according to the previous token y_t and C, a stop condition needs to be added to keep the decoder from infinitely generating new tokens. This is commonly done by placing a unique *end-of-sequence* (EOS) token at the end of the sequences Y_i in the training set $\mathbb{Y}$. Then once the trained decoder generates an EOS token, the sampling of new tokens is terminated (Cho et al. 2014; Sutskever et al. 2014).

The introduction of Seq2Seq models has improved the translation of natural language dramatically and is further investigated intensively in fields of natural language processing like text translation (Cho et al. 2014; Dai and Le 2015; Gehring et al. 2017; Sutskever et al. 2014), text summarization (Keneshloo et al. 2019; Nallapati et al. 2016), speech recognition (Chiu et al. 2018), voice conversion (Zhang et al. 2019), etc.

Encoder-Decoder models are not limited to solve just a problem within a specific domain but have proven to be generally suitable in multi-task learning, where a multitude of encoders and decoders are linked together in a many-to-many setting to perform tasks as image captioning and translation in parallel (Luong et al. 2015). Use cases for Seq2Seq models can also be found in other domains of application as long as the data in question can be provided in the format of a tokenizable sequence. Applications from image processing derived accordingly within the fields of video-to-text translation (Venugopalan et al. 2015), visual-inertial odometry (Clark et al. 2017), video object segmentation (Xu et al. 2018), video captioning (Chen et al. 2019) or automated data visualization (Dibia and Demiralp 2019; Liu et al. 2018).

Using Seq2Seq models to predict the length and values of a dynamic time series has first been proposed in Wörrlein et al. (2019). We extend this work through increased data quality and quantity along with experimental findings on the influence of differing discretization parameters.

Further explanations of the encoder decoder used here can be found in Cho et al. (2014), Goodfellow et al. (2016), and Sutskever et al. (2014). The following section combines the approach presented here into a framework that shows how to model energy consumption of machining jobs and how to implement such a model within a discrete-event oriented model.

3 A HYBRID MODELING FRAMEWORK FOR PREDICTING TIME SERIES

3.1 Concept for using Seq2Seq models for time series prediction

Figure 2 shows the overall concept of the proposed method. In the training phase, an unweighted RNN, the Seq2Seq-model, is parameterized using the input and target sequences $\{X_i, Y_i\}$. For the training data, 51 in field measurements of the active power usage of a machine tool were taken. This time series data was recorded under field conditions while processing the same type of job.

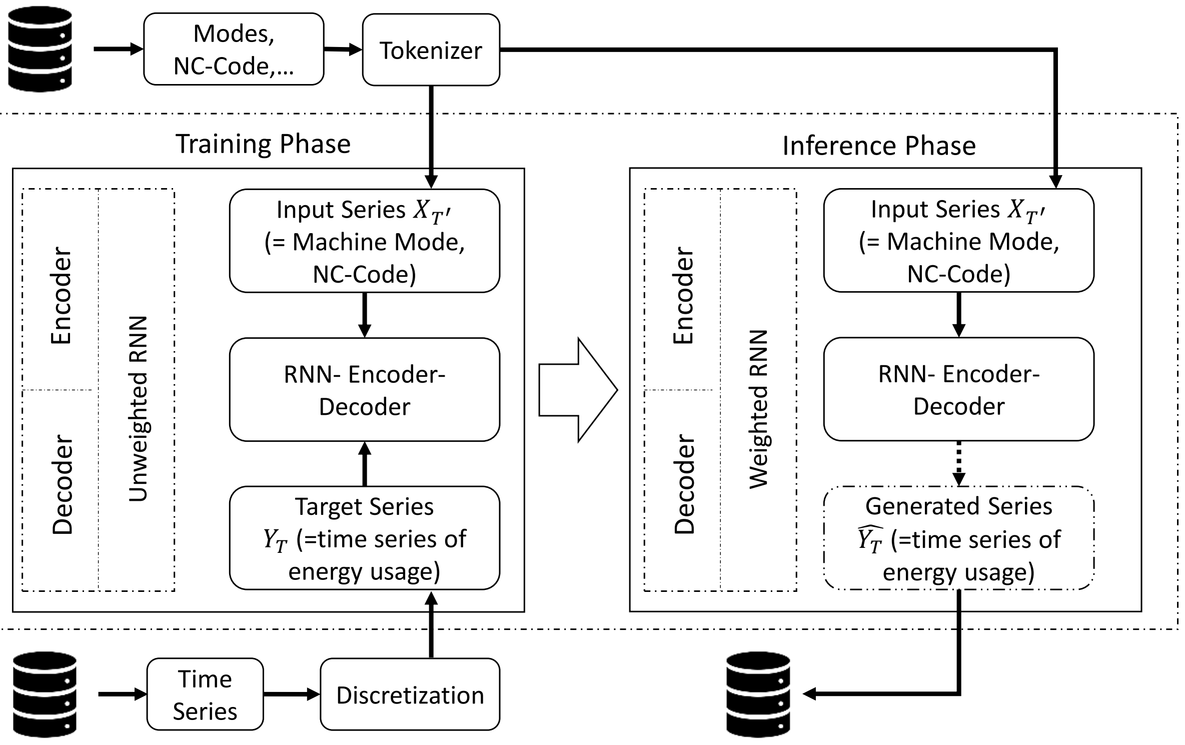

Modes, NC-Code,...
Tokenizer
Training Phase
Inference Phase
Encoder
Decoder
Unweighted RNN
Weighted RNN
Input Series $X_{T'}$ (= Machine Mode, NC-Code)
RNN- Encoder-Decoder
Target Series Y_T (=time series of energy usage)
Generated Series $\widehat{Y_T}$ (=time series of energy usage)
Time Series
Discretization

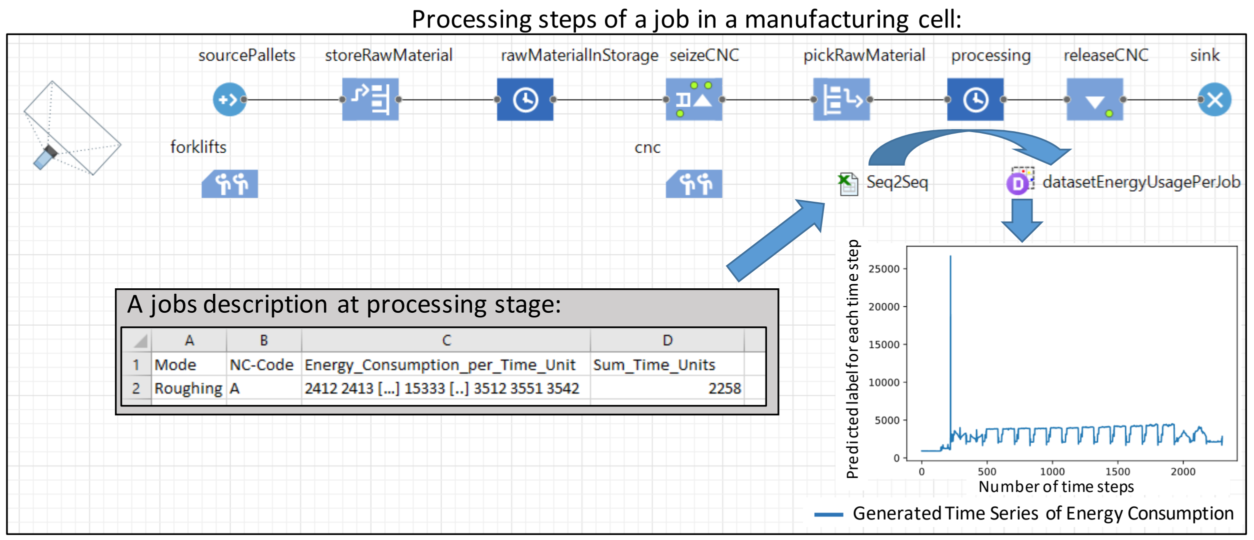
Figure 2: Implementation and components of an RNN-Encoder-Decoder-Architecture for asynchronous and asymbolic series.

The target sequences represent quasi-continuous recordings of the active power usage. The input sequence represents the numerical control code, along the machine states, of the same jobs. A tokenizer was used to generate the vectorized input sequence therefrom. The time series data of the machine tool's usage of energy is then trained to the numeric control code, along with the machines operating mode, within a Seq2Seq neural net. Once the training is finished, the trained neural net can inference time series $\hat{Y}$ according to an input X.

The task of the inference phase is to provide a meaningful power consumption profile $\widehat{Y_T}$, explicitly quasi-continuously over time (see Figure 2). The parametrized model can further be saved and implemented in a machine tool's DES-model to predict the time it takes to finish a job and the quasi-continuous energy usage while a job is manufactured.

3.2 Integration of Seq2Seq into DES

To provide the information generated at the inference stage to a simulation model, two approaches, that both qualify as HSM, can be considered that generally lead to the same effect. Firstly the weighted neural net can directly be called from within the simulation tool, to generate the values in question directly at runtime in the simulation stage. This approach allows for more flexibility, but requires an increased implementation effort up front. The second approach would be to create a data storage that holds generated values for all combinations that are possible at simulation start. Figure 3 showcases the latter approach in more detail.



	A	B	C	D
1	Mode	NC-Code	Energy_Consumption_per_Time_Unit	Sum_Time_Units
2	Roughing	A	2412 2413 [...] 15333 [..] 3512 3551 3542	2258

Figure 3: The values for the time it takes to finish a job, and the continuous values that describe the energy consumption at processing stage, are here provided to the model in from of an .XLS chart. To visualize the generated time series a time plot and data sets block, holding the time series data, were used.

Figure 3 shows the processing steps of a job within a discrete event oriented simulation modeled as a $\langle \mathcal{M} | \mathcal{G} | 1 \rangle$ queuing system. Here $\mathcal{M}$ refers to the waiting time (*rawMaterialInStorage*) of a job being described as a stochastic Markovian process that is defined by a Poisson distribution. $\mathcal{G}$ describes the generated, specific time distribution (*processing*) for each job at processing stage according to its parameter settings. As this paper focuses on the generation of the values for $\mathcal{G}$, instead of their integration, the simulation at runtime will no further be discussed here.

4 IMPLEMENTATION DETAILS

This section focuses on the preprocessing steps necessary to set up a Seq2Seq model, which can be integrated into a HSM. The Seq2Seq model, at training stage, contains input sequences X_i of sets of numerical control codes along with machine parameters and the target sequences Y_i of time series of the power consumption at process stage. These two sets of sequences need to be preprocessed to allow the model to learn a meaningful conditional joint distribution.

4.1 Setup of Input Sequences $\mathbb{X}$

The input sequences X_i hold the numerical control code, that controls the machine's action for a particular job. A numerical control code describes a sequence of necessary technological process steps up to the completion of a job. It can be understood as a concrete description of a sequence of states underlying the process of machining said job. The numerical control code thus decisively determines the behavior within the machining room of a machine tool. Furthermore, a job is only considered to be completed once the numerical control code has been completely run through.

The numerical code must first be translated into a sequence of numerical values that retains the structure of the targeted input sequence. This is realized by a so-called tokenizer. A tokenizer assigns a numeric value to each symbol or set of symbols present in the numerical code, e.g., based on the frequency of the symbol concerned.

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

The tokenizer also removes symbols or sets of symbols that are assumed to have a low information content, such as commas, upper/lower case letters and so on. Another way to limit the dimensions of the vector space is to dictate the tokenizer a maximum number of symbol sets, i.e., words, that can be mapped to a numerical vector. In the case presented here, a word to vector (Word2Vec) tokenizer was used, which translates all symbols and sets thereof into a vector.

The input sequences are further extended with different modes $\{x_{11}, x_{12}\}$ in which the machine tool can be operated on. Those modes reflect a common work routine in machining a job. The numerical control code runs for the first time $\{roughing = x_{11}\}$ to chip a larger amount of excess material off and give the material its shape. Afterwards the same numerical control code is run for several times $\{smoothing = x_{12}\}$ to smooth the surface of the now shaped material. Those two modes are reflected in time series of power consumption that are comparable in length but show very different characteristics in their features. The input sequence X_i is described accordingly as:

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

with x_2 being the numerical control code. The sequences of $\mathbb{X}$ further are tokenized to a list of integer values, where any unique word is represented by exactly one integer. This allows to model recurring patterns within the numerical control code.

4.2 Setup of Set of Time Series $\mathbb{Y}$

The basis of values for the quasi-continuous time target series Y_i is the power current consumption of the same jobs when the numerical control code is processed. The temporal power consumption gives concrete information about when how much consumption must be anticipated as soon as a decision has to be made about the machining of a job. The time series data was recorded under field conditions and has the same clocking of Δt, that is 500 [*ms*].

The machine tool's usage of energy, and inherently the time it took to process said job, is initially monitored every time a job is processed on it and saved as time series data. The set of time series $\mathbb{Y}$ has further been discretized. Discretization is the process of portioning continuous values into some new discrete groups of values or *bins*, that resemble the original values of the data. This was necessary as the empirical results presented in (Wörrlein et al. 2019) led to the conclusion that a uniform or long tail distribution, i.e., where the tail tends towards a discrete uniform distribution, P^f of value frequencies prevents the Seq2Seq model from learning a meaningful joint distribution.

To find the right parameter, as to which degree needed to be discretized, several runs of training with alternative discretization parameters were conducted. The different discretization parameters were applied on the whole data set of time series and then classified according to the frequency f of the discretized values (see Figure 4). To do so, the distribution of frequencies P^f was analyzed using a gaussian kernel density estimator-KDE.

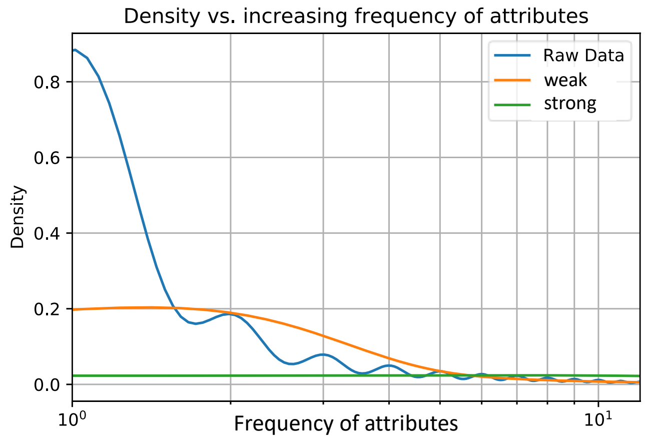


Figure 4: The KDE-plot shows that the raw data contained mostly unique values, while a strong discretization results in a uniform distribution, where the probability of a value belonging to any frequency is the same as for any other frequency. A weak discretization results in a heavy tail distribution of frequencies.

The proposed concept has been tried for all three frequency distributions and results only for the *weak* discretization in satisfying results.

5 RESULTS AND EVALUATION

Metrics to compare the generated time series $\widehat{Y_i}$ and Y_i are the median length $len(\bar{Y}_i)$ and average $sum(\bar{Y}_i)$ of time series as found in the training set. Further, the time series have been visualized and features of characteristic patterns or labels have been added to those visualizations (see Figure 5). Adding features helps to compare the time series $\widehat{Y_i}$ and Y_i more intuitively on a visual level. The result for the raw data, which has not been discretized, aligns with the non-conclusiveness of (Wörrlein et al. 2019). The sequences created showed no meaningful course of values and further failed to produce an EOS-token.

The results for the strong discretization as shown in Figure 4 disappoint. An EOS token was created, as well as most other features, yet the generated series can clearly be distinguished from the training data (samples shown in Figure 6) and results in low scores in the metrics accordingly.

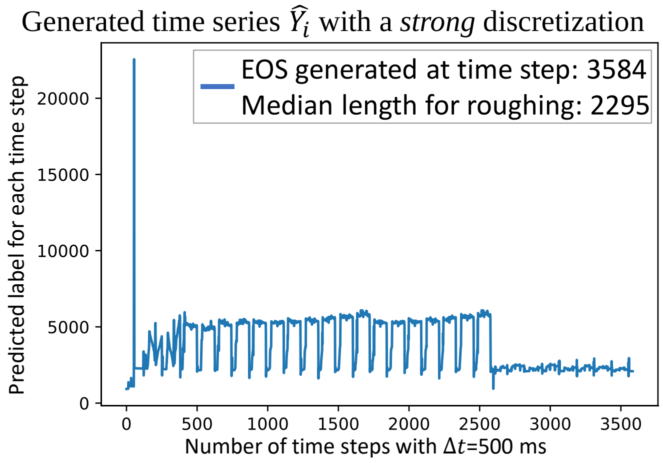


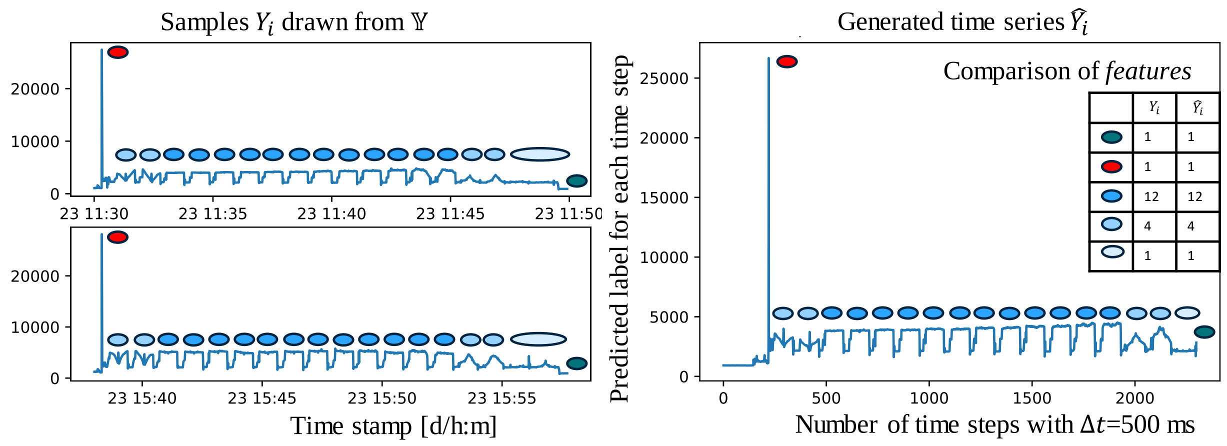
Figure 5: Result for a strong discretization and parameter setting $\{x_{11}, x_2\}$. An EOS token was created, as well as most other features, yet the generated series can clearly be distinguished from the training data (see Figure 6) and results in low scores in the metrics.

The *weakly* discretized time series on the other hand shows high values in comparison to $len(\bar{Y}_i)$ and $sum(\bar{Y}_i)$:

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

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

On closer inspection of the time series $\widehat{Y}_i$ and Y_i for $\{x_{11}, x_2\}$, as displayed in Figure 6, a striking resemblance can be seen.



	Y_i	$\widehat{Y}_i$
●	1	1
●	1	1
●	12	12
●	4	4
●	1	1

Figure 6: Comparison of samples Y_i drawn from the training set $\mathbb{Y}$ and the generated time series $\widehat{Y}_i$ for the weak discretization parameter and the sequence combination $\{x_{11}, x_2\}$. The sequences displayed from the two sets clearly show the same patterns over the course of labels. To facilitate the visual comparison between distinct sequences, features have been added to local patterns or points of interest. The table shown on the right side compares the count of features against each other.

The time series generated accomplishes to mimic the course of labels as shown in the training set with a remarkable precision. Not only achieves it to reproduce an EOS token, that matches the length of the time series found in the training set (green dot), a distinct peak-feature (red dot), a string of subsequences (the dots of changing shades of blue), but also to generate them in the right order and dimensionality.

The time series $\widehat{Y}_i$ and Y_i for $\{x_{12}, x_2\}$, displayed in Figure 7, also clearly show that the Seq2Seq-model succeeded in catching the course of labels within the training set, even though the time series from the training set contained few features that could be learned in the first place.

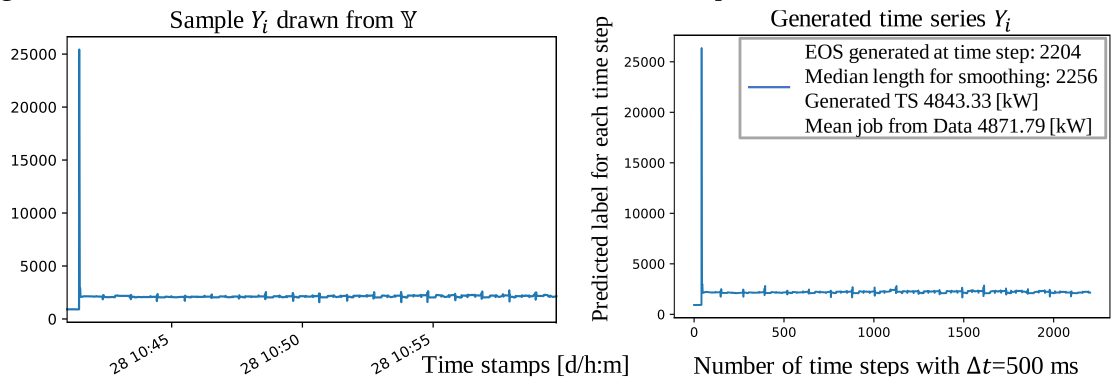


Figure 7: Comparison of samples Y_i drawn from the training set $\mathbb{Y}$ and the generated time series $\widehat{Y}_i$ for the weak discretization and the sequence combination $\{x_{12}, x_2\}$. No features were added as the time series holds few characteristics.

6 CONCLUSION AND FUTURE WORK

The functionality of the described approach was confirmed in the use case by chosen metrics. However, the generated time series still must be critically questioned and validated in further research work. On the one hand, there is still a lack of evaluation methods for generative models of ML to check the generated time series entries for the meaningfulness of their entries. This is done at the moment by the observation and comparison of the generated time series through an expert of the application case by optical inspection (Goodfellow et al. 2016) as shown in Figure 6.

For a final evaluation of the methods used, it is advisable to increase the qualitative and quantitative data basis of the Seq2Seq-model. The data set used here is of a small size. Yet the set size is exemplary for real world settings that might change rapidly and in short periods of time. Machine learning algorithms on the other hand tend to work better given that there is a lot of data to learn from. A framework in which the training set is extended by time series which have been altered to represent a ground truth of the training set of time series could solve that problem. *Dynamic time warping* could be used to generate such ground truth time series, which could then iteratively be added to the training set until an advantageous learning behavior could be displayed.

Additional *end-of-sequence* tokens could be used to describe events like machine failure. The EOS token used here simply marked the end of a finished job. Yet some jobs are prone to break due to system changes like wear and tear experienced by the tool. Adding an alternative EOS, indicating machine failure, to the training set, along with data for the state of tools etc., might also answer the question if a job can be executed given the current settings.

The method further allows to generate time series according to factorial combinations not found in the training data. As the decoder is not parametrized directly on the input sequences found in the training set, but on a summary thereof in form of the context vector, factorial combinations of input parameters can be used that are not represented in the initial training set. If the respective input parameters and their distinctive effect on the time series has been modeled, any combination thereof could be used. This would result in factor combinations of high interest to a simulation expert, that could not be modeled in a generic simulation setting. Furthermore, a suitable evaluation method must be added to the proposed solution, since validation cannot be guided by a (non-existent) ideal time series.

The further development of the machine learning method described above and its use for hybrid simulation models is currently the subject of ongoing research. Also, if the method is successfully established and validated, a solution could be developed that produces plausible power consumption forecasts for unknown jobs, e.g., based on their numerical control codes. This would have a high practical potential and would also be a breakthrough from a scientific point of view. The transfer of the basic idea to other forms of control code and time series of other measured values is also conceivable and a possible subject for further investigations.

REFERENCES

Bergmann, S., and S. Strassburger. 2010. “Challenges for the Automatic Generation of Simulation Models for Production Systems”. In *Proceedings of the 2010 Summer Computer Simulation Conference*, July 11th-14th, San Diego, California, 545–549.

Bergmann, S., S. Stelzer, and S. Strassburger. 2014. “On the Use of Artificial Neural Networks in Simulation-based Manufacturing Control”. *Journal of Simulation* 8(1):76–90.

Bergmann, S., N. Feldkamp, and S. Strassburger. 2017. “Emulation of Control Strategies through Machine Learning in Manufacturing Simulations”. *Journal of Simulation* 11(1):38–50.

Bergmann, S., S. Stelzer, S. Wüstemann, and S. Strassburger. 2012. “Model Generation in SLX using CMSD and XML Stylesheet Transformations”. *In Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, 1–11. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Biller, B., S. R. Biller, O. Dulgeroglu, and C. G. Corlu. 2017. “The Role of Learning on Industrial Simulation Design and Analysis”. In *Proceedings of the 2017 Winter Simulation Conference*, edited by W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, 3287–3298. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Brause, R. W. 1995. *Neuronale Netze: Eine Einführung in die Neuroinformatik*. 2nd ed. Wiesbaden, Germany: Vieweg+Teubner Verlag.

Chen, S., T. Yao, and Y.-G. Jiang. 2019. "Deep Learning for Video Captioning: A Review". In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, edited by T. Eiter and S. Kraus, 6283–6290. California, USA: International Joint Conferences on Artificial Intelligence Organization.

Chiu, C., T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani. 2018. "State-of-the-Art Speech Recognition with Sequence-to-Sequence Models". In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, 4774–4778. Calgary, Canada: Institute of Electrical and Electronics Engineers, Inc.

Cho, K., B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". *arXiv:1406.1078.*

Clark, R., S. Wang, H. Wen, A. Markham, and N. Trigoni. 2017. "VINet: Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem". *arXiv:1701.08376.*

Dai, A. M., and Q. V. Le. 2015. "Semi-supervised Sequence Learning". In *Advances in Neural Information Processing Systems 28*, edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, 3079–3087. Montreal, Canada: Curran Associates, Inc.

Dibia, V., and Ç. Demiralp. 2019. "Data2Vis: Automatic Generation of Data Visualizations Using Sequence-to-Sequence Recurrent Neural Networks". *IEEE Computer Graphics and Applications* 39(5):33–46.

Elbattah, M., O. Molloy, and B. P. Zeigler. 2018. "Designing Care Pathways using Simulation Modeling and Machine Learning". In *Proceedings of the 2018 Winter Simulation Conference*, edited by M. Rabe, A.A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, 1452–1463. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Gehring, J., M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. 2017. "Convolutional Sequence to Sequence Learning". *In Proceedings of the 34th International Conference on Machine Learning*, August 6th–11th, Sydney, Australia, 1243-1252.

Goodfellow, I., Y. Bengio, and A. Courville. 2016. *Deep Learning*. Boston, Massachusetts: MIT Press.

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

Keneshloo, Y., T. Shi, N. Ramakrishnan, and C. K. Reddy. 2019. "Deep Reinforcement Learning for Sequence-to-Sequence Models". *IEEE Transactions on Neural Networks and Learning Systems* 31(7):2469-2489.

Liu, X., Z. Han, Y.-S. Liu, and M. Zwicker. 2018. "Point2Sequence: Learning the Shape Representation of 3D Point Clouds with an Attention-based Sequence to Sequence Network". *arXiv:1811.02565.*

Luong, M.-T., Q. Le V, I. Sutskever, O. Vinyals, and L. Kaiser. 2015. "Multi-task Sequence to Sequence Learning". *arXiv:1511.06114.*

Morin, M., F. Paradis, A. Rolland, J. Wery, and F. Laviolette. 2015. "Machine learning-based Metamodels for Sawing Simulation". In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, 2160–2171. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Mustafee, N., J. H. Powell, and J. H. Harper. 2018. "RH-RT: A Data Analytics Framework for Reducing Wait Time at Emergency Departments". In *Proceedings of the 2018 Winter Simulation Conference*, edited by M. Rabe, A.A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, 100–110. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Mustafee, N., S. Brailsford, A. Djanatliev, T. Eldabi, M. Kunc, and A. Tolk. 2017. "Purpose and Benefits of Hybrid Simulation: Contributing to the Convergence of its Definition". In *Proceedings of the 2017 Winter Simulation Conference*, edited by W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, 1631–1645. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Nallapati, R., B. Xiang, and B. Zhou. 2016. "Sequence-to-Sequence RNNs for Text Summarization". *arXiv:1602.06023v1*.

Onggo, B. S., N. Mustafee, A. Smart, A. A. Juan, and O. Molloy. 2018. "Symbiotic Simulation System: Hybrid Systems Model meets Big Data Analytics". In *Proceedings of the 2018 Winter Simulation Conference*, edited by M. Rabe, A.A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, 1358–1369. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Peter, T., and S. Wenzel. 2015. "Simulationsgestützte Planung und Bewertung der Energieeffizienz für Produktionssysteme in der Automobilindustrie". In *Simulation in Production and Logistics 2015*, edited by M. Rabe and U. Clausen, 535–544. Stuttgart: Fraunhofer Verlag.

Rabe, M., and F. Dross. 2015. "A Reinforcement Learning approach for a Decision Support System for logistics networks In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, 2020–2032. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Römer, A. C., M. Rückbrod, and S. Straßburger. 2018. "Eignung kombinierter Simulation zur Darstellung energetischer Aspekte in der Produktionssimulation". In *24. Symposium Simulationstechnik*, edited by C. Deatcu, T. Schramm, and K. Zobel, 73–80. Vienna: Arbeitsgemeinschaft Simulation.

Sutskever, I., O. Vinyals, and Q. Le V. 2014. "Sequence to Sequence Learning with Neural Networks". *arXiv:1409.3215*.

Thiede, S. 2012. *Energy Efficiency in Manufacturing Systems*. Berlin, Germany: Springer Verlag.

Venugopalan, S., M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. 2015. "Sequence to Sequence - Video to Text". *arXiv:1505.00487*.

Wörrlein, B., S. Bergmann, N. Feldkamp, and S. Straßburger. 2019. “Deep-Learning-basierte Prognose von Stromverbrauch für die hybride Simulation”. In *Simulation in Produktion und Logistik 2019*, edited by M. Putz and A. Schlegel, 121–131. Auerbach: Verlag Wissenschaftliche Scripten.

Xu, N., L. Yang, Y. Fan, J. Yang, D. Yue, Y. Liang, B. Price, S. Cohen, and T. Huang. 2018. “YouTube-VOS: Sequence-to-Sequence Video Object Segmentation”. *arXiv:1809.00461*.

Zell, A. 2003. *Simulation neuronaler Netze*. 4th ed. Munich, Germany: Oldenbourg.

Zhang, J.-X., Z.-H. Ling, L.-J. Liu, Y. Jiang, and L.-R. Dai. 2019. “Sequence-to-Sequence Acoustic Modeling for Voice Conversion”. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27(3):631–644.

AUTHOR BIOGRAPHIES

BENJAMIN WOERRLEIN holds a master degree in Industrial Engineering from the Ilmenau University of Technology and works currently as a doctoral student in the Group for Information Technology in Production and Logistics at the Ilmenau University of Technology. His research interests include machine learning (Deep learning), data science and hybrid simulation systems. His email address is benjamin.woerrlein@tu-ilmenau.de.

STEFFEN STRASSBURGER is a professor at the Ilmenau University of Technology and head of the Group for Information Technology in Production and Logistics. Previously he was head of the “Virtual Development” department at the Fraunhofer Institute in Magdeburg, Germany and a researcher at the DaimlerChrysler Research Center in Ulm, Germany. He holds a Doctoral and a Diploma degree in Computer Science from the University of Magdeburg, Germany. His research interests include distributed simulation, automatic simulation model generation, and general interoperability topics within the digital factory and Industry 4.0 context. His email address is steffen.strassburger@tu-ilmenau.de.