Using Dynamic Time Warping and Synthetic Data for Validating Seq2Seq in 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*

SNE 34(4), 2024, 203-213, DOI: 10.11128/sne.34.tn.10713
Selected ASIM WS 2023 Postconf. Publication: 2023-10-15
Rec. Improved English Vs.: 2024-08-03; Accepted: 2024-09-20
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

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.

Introduction

Sequence-to-Sequence (Seq2Seq) [3, 16] is a class of machine learning (ML) methods that enables mapping two sequences of different lengths and different descriptions to each other. Seq2Seq utilizes artificial neural networks and is categorized under deep learning [4].

In the context of simulation, Seq2Seq has already been successfully used to translate numerical control codes (NC codes) of a machine tool (MT) into a time series of the energy consumption of the same machine. For this, a training dataset is first created based on individual manufacturing orders (MO).

This dataset contains the NC code and the corresponding measured time series for each MO. The Seq2Seq model is then trained with this dataset. The trained model can now be activated with an NC code and subsequently outputs a time series based on the NC code [25]. Furthermore, the model can be used within a hybrid simulation to predict, for example, the duration of a MO or the energy consumption of machines [22, 26].

The Seq2Seq method suffers from several issues:

1. Training Data:

The acquisition of training data is cost and time-intensive [13, 21]. This effect is particularly strong for ML methods, as the amount of training data is equated with improved learning performance [5]. As a solution, we propose the use of synthetic data [13, 21].

2. Lack of Metrics for Ambiguous Datasets

Training data for machine learning must generally be unambiguous. *Unambiguous* here means that there is exactly one solution for each sample in the training dataset. This condition is violated for training data that includes NC codes and time series, where slightly different time series are measured for the same NC code. We propose introducing a new learning metric that compares the generated time series at the end of a training period with a comparison time series.

3. No Comparison Time Series:

Comparing generated time series with time series from training data seems trivial at first glance. The problem lies in two points. First, during the training of a Seq2Seq model each data point in the time series is compared individually to one another [3, 16]. A comparison of time series in their entirety does not take place. Second, for ambiguous datasets, there is no single comparison time series that can be used to compare entire time series. We propose solving this problem with *Dynamic Time Warping* (DTW) [1, 15].

The following chapters will propose the fundamentals of the suggested method and a concept based on it to solve all three of those problems. The core of the concept is to generate several time series based on all possible NC codes after each training run. Subsequently, the generated time series are compared with reference time series created by DTW. This is done iteratively at the end of each training run. Therefore, this approach is referred to as *Iterating over Metrics* (IOM).

Next, the components necessary for the implementation of the concept will be explained in detail, and the results of the derived method will be presented.

1 Fundamentals of the IOM-approach

For better understanding, the methods used in the presented IOM- approach will be briefly explained.

1.1 Sequence-to-Sequence

Artificial neural networks (ANNs) are used to identify patterns in complex data structures. When patterns change over time, this temporal sequence of patterns is understood as a sequence.

To process temporal patterns, recurrent connections in ANNs are necessary, allowing feedback of abstracted knowledge [27]. Such feedback or recurrent neural networks (RNNs) are particularly suitable for data in sequential form [4].

When the data consists of sequences, these are referred to as Sequence-to-Sequence (Seq2Seq) architectures. The input sequence is encoded through the input layer of an ANN.

When the input sequence is encoded into a neural layer, it is called an encoder. When a target sequence is generated from a neural layer, this part is referred to as a decoder [4].

Such a topology is also commonly referred to as recurrent encoder-decoder networks (RNN-ED) [4]. Figure 1 depicts an ANN for the application of NC codes and time series.

During training, all data of a sequence pair $\{X_i, Y_i\}$ are processed sequentially and mapped to each other. First, the input sequence X_i is encoded into a so-called context vector *C*. This context vector is then decoded into the output sequence Y_i.

The ML model learns the progression of Y_i by updating the state of the context vector for each entry $y_{i,T}$.

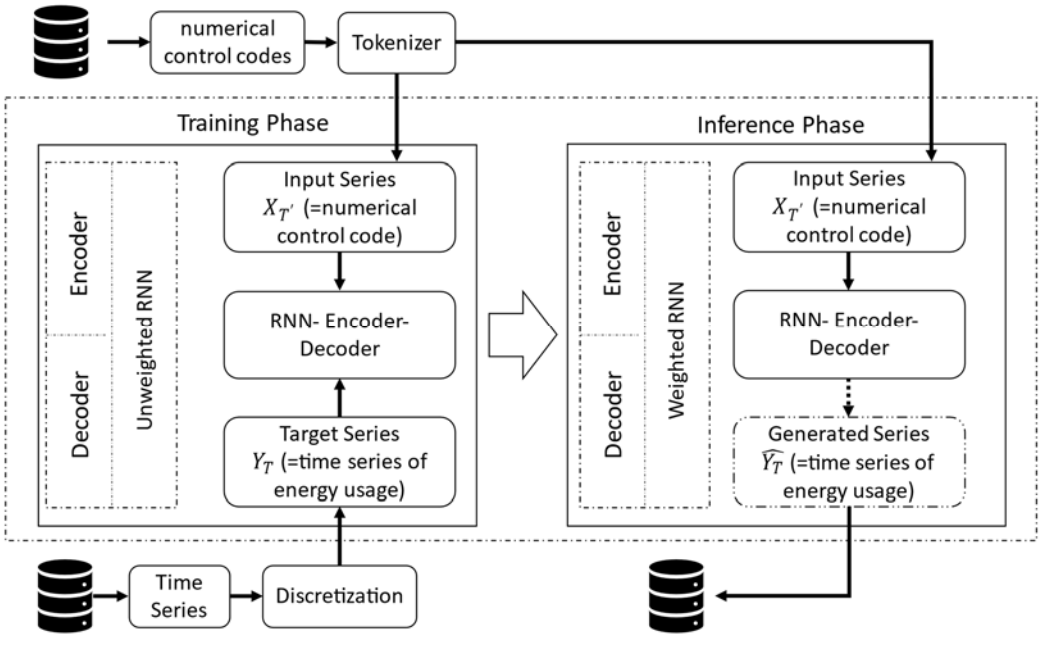

Figure 1: Components of an RNN Encoder-Decoder Topology.

The change in the state of the ANN is determined by a categorical cross-entropy loss function. The model then attempts to minimize this error [4].

The error value describes the difference between the target y_i and the predicted result $\hat{y}_i$. The prediction $\hat{y}_i$ is made in the final layer of the ANN, which uses a so-called *softmax* activation function. However, this has the drawback that it can only compare one entry of y_i with the generated time series at a time [4].

Sequence-to-Sequence models led to the development of *Large Language models* (LLM). This was made possible through the evolution from encode-decoder architectures to *Transformer* architectures [28].

Further explanations of the encoder-decoder network used here can be found in [3, 4, 16].

1.2 Synthetic data from Simulation

A fundamental problem in applying deep learning is the availability of training data. This data must be collected, checked for inconsistencies, preprocessed with analytical methods, and so on. This process often involves significant costs and time. This is particularly evident in the example of the dataset of NC codes and energy consumption time series for manufacturing orders.

If the machine tool in question does not have a suitable interface, the NC code and the period during which it operates must be manually recorded and transferred into a data format. The creation of the energy time series dataset is even more complex.

It requires identifying, implementing, and ultimately applying suitable measurement technology and software. This demands significant resources, including available personnel, as well as the availability of machine tools and measurement equipment.

Once the prerequisites for recording the NC code and time series are met, the next question is whether the machine tool in the observed period has sufficient utilization to generate a large number of training data points. Given the size of datasets typically used in deep learning (>10k), this seems unlikely.

Synthetic data can address this issue. Synthetic data are artificially generated data that resemble the original data in the relevant structure (length, features, feature frequency, etc.). The advantage of synthetic data is that they can be created cost-effectively, transparently, and reproducibly. Synthetic data can be generated through a simulation model and then used as a training basis for ML models.

The use of synthetic data in machine learning is not new and is an established research field [20]. For example, *Melo et al.* [13] demonstrated that synthetic data can be used to train image recognition models. Other applications of synthetic data in deep learning include non-destructive testing of steel [2], object detection [23], creating vehicle boundary frames for autonomous driving [20], and pedestrian detection in image data [7].

Creating synthetic data through simulation techniques is also not novel. Data farming has been used to create a data basis for deep learning in the simulation of production systems [9] or object detection for robots [19]. Additional applications include generating mobility data [8, 12], image data for heart tissue determination [10], or data for manufacturing planning and control [6].

1.3 Dynamic Time Warping

Time series describe the progression of a feature over time, typically for a fixed measurement interval, meaning the same interval between individual data points. As a result, a time series is essentially a collection of temporally ordered data points.

Comparing time series might initially seem like a trivial problem. For instance, if comparing time series of a fixed period, such as the temperature over a day, one could simply overlay the time series of two days, calculate their average to determine the mean temperature, or subtract them to find the temperature difference between two days.

However, comparing time series that do not follow a fixed period is more complex. For example, in the case of time series for the energy consumption of a MO. Machine tools are dynamic systems. This means their state changes with each event, resulting in different appearances for time series of the same NC code and the same machine. The differences can be observed in both dimensions of the time series: the number of data points and the feature values of the time series.

When comparing time series from dynamic systems, the following issues arise:

Firstly, the time series will have a different number of data points. The length of a time series is determined by its number of data points. If there is a discrepancy, the additional data points in one time series cannot be compared with the data points of the other time series because they are missing. If one were to remove the additional data points, it would make the time series comparable again, but potentially important data points could be lost.

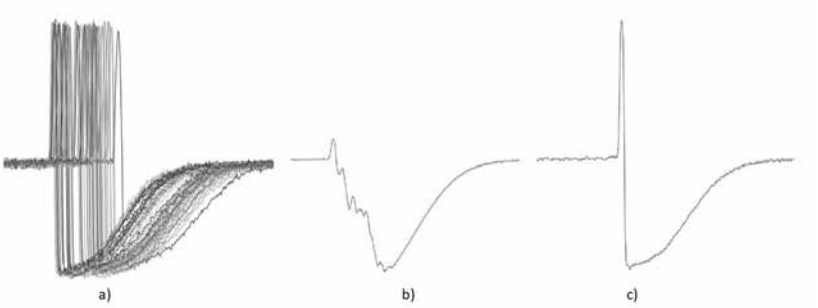

Figure 2: Comparison of a) a set of time series, b) the mean of that set and c) a time series that was generated from the set via DTW (Figures taken from [14]).

Alternatively, one could fill the shorter time series with values until it matches the length of the longer time series. The challenge here is finding a value that can fill the shorter time series without adversely affecting it.

Secondly, the characteristic patterns in the time series might shift due to missing data points. For example, these patterns could be a daily temperature peak at noon or, as in the mentioned case, recurring patterns in the tension profile of a machine tool.

Considering Figure 2, the above-mentioned problems are further illustrated. Figure 2 a) shows a set of time series, all of which are slightly different in length and whose characteristics (such as slope, gradient, and return to the starting value) differ in both position and magnitude. If the time series were simply filled with values (e.g. the mean of the time series), one could compute the mean of all the time series. Figure 2 b) shows the result of this averaged time series. It is evident that this averaged time series, except for its length, is no longer comparable to the original time series.

A method that can compare time series of different lengths with varying positions of their characteristics is called *Dynamic Time Warping* (DTW) [1, 15]. DTW is an algorithm that compares a set of time series using a so-called distance matrix.

The distance matrix compares the data points of two (or a set of) time series using a distance measure, e.g. *Euclidean* distance, *Manhattan* distance, etc. When comparing two data points, their difference is recorded in the distance matrix.

Then, a path is drawn through the distance matrix, known as the *Warping Path* (see the line in Figure 3). The optimal *Warping Path* is that path that, in total, has the shortest distance through the distance matrix compared to other paths.

If the data points are close to each other, the *Warping Path* increases uniformly in each dimension (see Figure 3, *Warping Path* – bottom left). If the data points of two time series deviate from each other, the *Warping Path* either flattens out or rises sharply.

Once a *Warping Path* is found, it can be used to create an averaged time series through DTW. For example, Figure 2 c) shows a time series that is clearly comparable in length and shape to the individual time series from Figure 2 a).

Further literature on Dynamic Time Warping can be found in [1, 14, 15, 17].

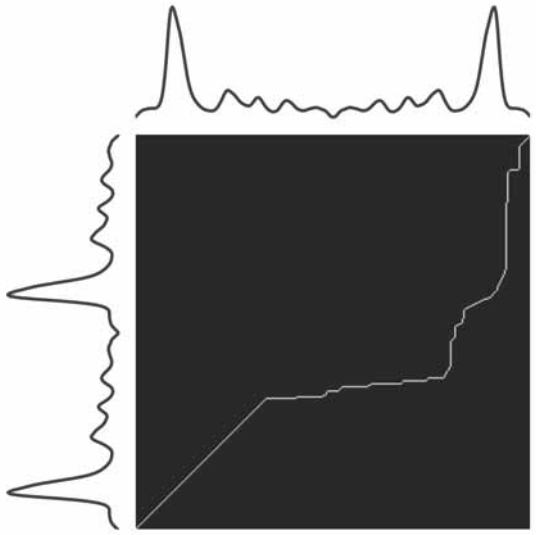

Figure 3: Distance matrix of two time series compared using *Dynamic Time Warping* (shown on the left and at the top) [17].

2 IOM-Concept

The IOM- approach consists of several steps (compare Figure 4). First, to address the cost and availability issues of training data, it will be generated using a simulation model.

In the second step, reference time series are created. These are used to make the time series generated by the model comparable.

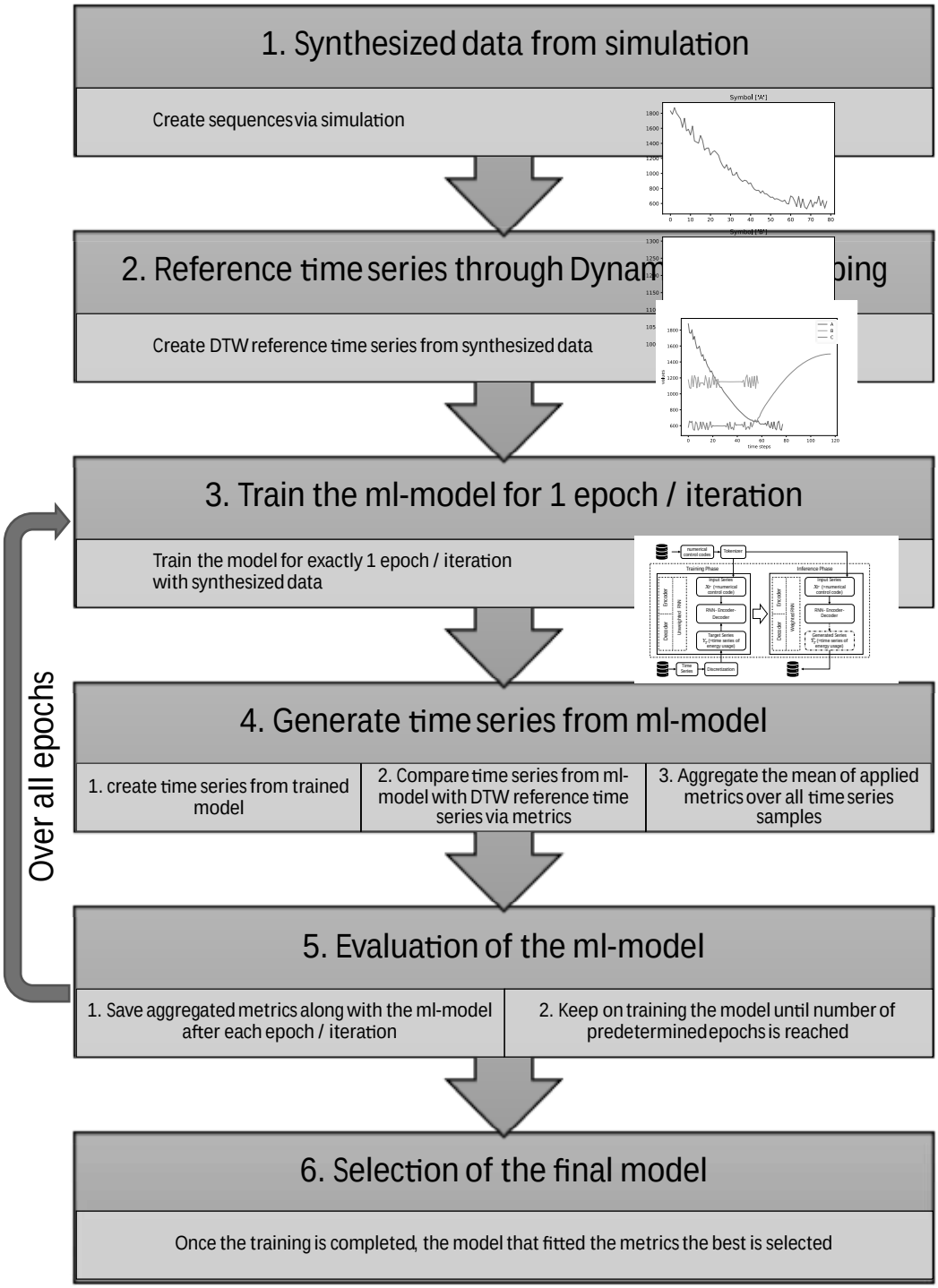

Figure 4: Concept of the IOM-approach

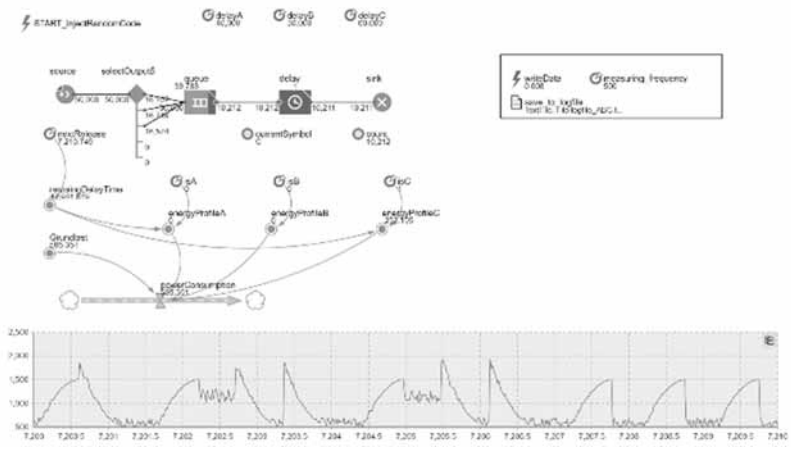

Figure 5: AnyLogic Simulation Model.

This is done using *Dynamic Time Warping*, with the data foundation being the previously generated synthetic data that will also be used for training.

Third, the training of the ML model is initiated. The model is trained for exactly one iteration (Note: To reduce computation limitations, 10 epochs are combined into one iteration). In the fourth step, the trained model is used to generate time series.

Comparing entire time series is necessary to address the problem described in Chapter 1.1, where the model's *softmax* function only compares individual data points step by step, not entire time series.

The quality of the generated time series is then assessed by comparing them with the reference time series produced by DTW. After metrics comparisons are made for all considered time series, these are aggregated. The average of each individual metric across all time series is computed.

In the subsequent fifth step, the model undergoes further training and repeats steps 3 and 4 until a predetermined number of epochs is reached. Again, a summary of metrics is created in each iteration, and the trained model is saved. In the sixth and final step, the model that achieved the best results in the evaluated metrics is selected.

3 Experimental Preparation

To implement the IOM- concept, several preparatory steps are necessary. For once there is the generation of synthetic time series data through the simulation model. Additionally, the creation of comparison time series using *Dynamic Time Warping*. Finally, the metrics defined in the previous chapter must be developed and integrated into the model training process.

3.1 Creation of synthetic time series data

To create synthetic time series, the simulation tool *AnyLogic* (see Figure 5) is used. In *AnyLogic*, a discrete-event simulation model is set up. In the model's source contains manufacturing orders (A, B, or C) that are generated randomly.

These orders enter a queue and then move to a waiting area. Only one (MO) can be in the waiting area at a time. Each MO is assigned its own waiting time, which follows a continuous uniform distribution of [0.95;1].

Once the waiting area is occupied by an MO a function associated with that MO is executed. These functions differ for each of the three MO types (see Figure 6).

They are mathematical functions that use the remaining waiting time of the MO as an input parameter. The outputs of these functions thus vary over time. The computed value is then altered through another uniform distribution function and finally output.

The uniform distribution in the waiting area and in the output functions ensures that the time series for the same type of MO will slightly differ each time. This addresses the problem described initially, where two measured time series of the same MO never match exactly in length and value progression. Thus, it is ensured that two MO with the same label are represented with comparable, but not identical time series. Figure 6 illustrates samples of the generated time series. The uniform distribution in the output values is clearly visible here.

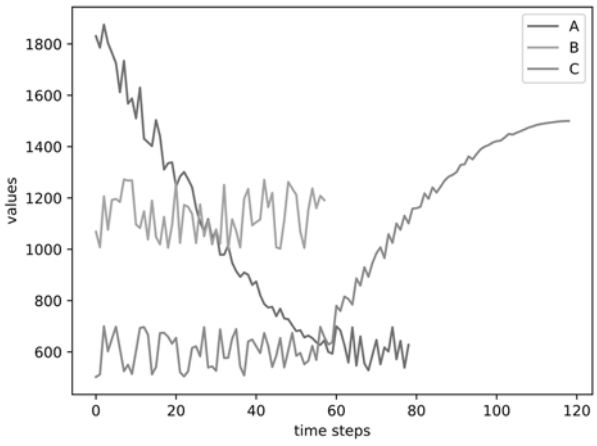

Figure 6: Sample of time series created by simulation model.

3.2 Generation of reference time series through Dynamic Time Warping

To create reference time series, *Dynamic Time Warping* is used. DTW generates a reference time series from a set of time series, which exhibits minimal overall error in length and structure compared to the original set.

The calculation is performed using the Python library *tslearn* [17]. The specific DTW algorithm used here is *softDTW* [11]. *softDTW* has a higher error tolerance in the creation of the *Warping Path*, which positively affects the generation of averaged time series [11].

Figure 7 shows a time series generated by *softDTW* compared to two samples of manufacturing order C. The *softDTW* time series effectively summarizes the samples but does not merely replicate them. Instead, it emulates their shape and length.

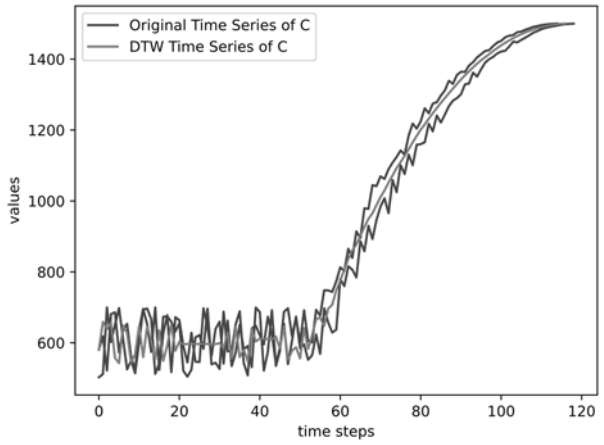

Figure 7: Comparison of 2 synthetic time series of type C with the time series created for type C through softDTW.

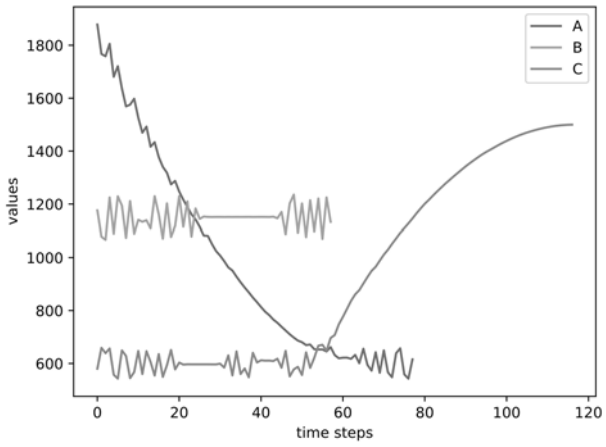

Figure 8: Sample of time series created via softDTW.

Figure 8 shows the *softDTW* time series for all three manufacturing orders. Compared to Figure 6, the difference between the original and DTW-generated time series is evident. The time series all exhibit a similar length and shape but differ slightly, as they are derived from the entire set of time series.

3.3 Implementation of metrics into the Sequence-to-Sequence Model

During model training, the generated time series are compared with the DTW-generated time series as a reference. Subsequently, the values of the metrics for the individual time series are averaged over an entire training epoch. Two metrics were used for this purpose.

First, the mean squared error (*MSE*) is applied. This compares the time series by calculating the difference between each individual data point and then squaring these differences.

All differences are then summed to provide a measure of the overall quality of the time series. Squaring the differences has two advantages. First, all differences become positive, which means that negative and positive error values do not cancel each other out in the subsequent addition. Second, squaring gives more weight to large errors compared to small ones.

The second metric, *sigma length*, considers only one dimension of the time series, more precisely its length. This metric calculates the number of data points in both time series and then divides one by the other. The closer the result is to 1, the more similar the generated time series is to the reference time series in terms of length.

```
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 of the MES and *sigma length* metrics to compare two time series. *ml_ts* marks the time series that was created through the machine learing model. *dtw_ts* is the reference time series that was created via DTW.

When a training run is completed, time series are generated with the trained model. These are then compared using the metrics described in Code 1, and the results for each time series are saved.

Next, a summary for the entire iteration is calculated. This aims to provide a comprehensive assessment of the model at a particular training stage. Hence, all *MSE* and *sigma length* results are averaged.

The mean allows for a more robust statement about the model's ability to generate time series than examining individual time series.

4 Experiment Execution

The creation of synthetic time series was carried out in *AnyLogic*. Python was used as the programming language. The calculation of DTW-generated time series was performed in *tslearn* [17], and the creation of the ML model in TensorFlow [18].

The hardware available included a Ryzen 7 2700 X processor, an RTX 2080Ti graphics card with 4352 CUDA cores, and a 480 GB SSD.

The IOM- approach involves saving the ML model after each epoch. However, this could not be implemented with the available storage capacity. Therefore, it was decided to save the trained the model after 10 epochs (here: equal to 1 iteration) and only collect metrics for the last epoch of an iteration.

The model was trained for 1000 epochs (subsequently 100 iterations).

The training data as described in Chapter 3 consisted of labeled sequence pairs formed from 2 manufacturing orders each to increase the variations in the dataset. The dataset was comprised of 10,000 labeled sequence pairs.

4.1 Comparison of metrics

As described in the IOM- concept, averaged metrics are to be generated at the end of each iteration. The progression of these metrics is plotted in Figure 9. The results of *MSE* and *sigma length* are shown.

To select the optimal model, the optimum of both metrics is determined. The optimum of a metric is the value at which the metric achieves the best result. For *MSE*, this would be 0, as *MSE* considers the difference between all data points. For *sigma length*, it would be 1, as this metric examines the ratio of the length of one time series to another.

To illustrate, the optimum for *sigma length* is graphically represented (blue line in Figure 9). All iterations that lie on the blue line represent models that have achieved the optimal value in the *sigma length* metric.

In a second step, iterations where *sigma length* $= 1$ are examined for their value in the *MSE* metric. The green line in Figure 9 at iteration 91 (=epoch 910) corresponds to the iteration where the values for both metrics are the lowest. Therefore, the model from epoch 910 is therefor selected.

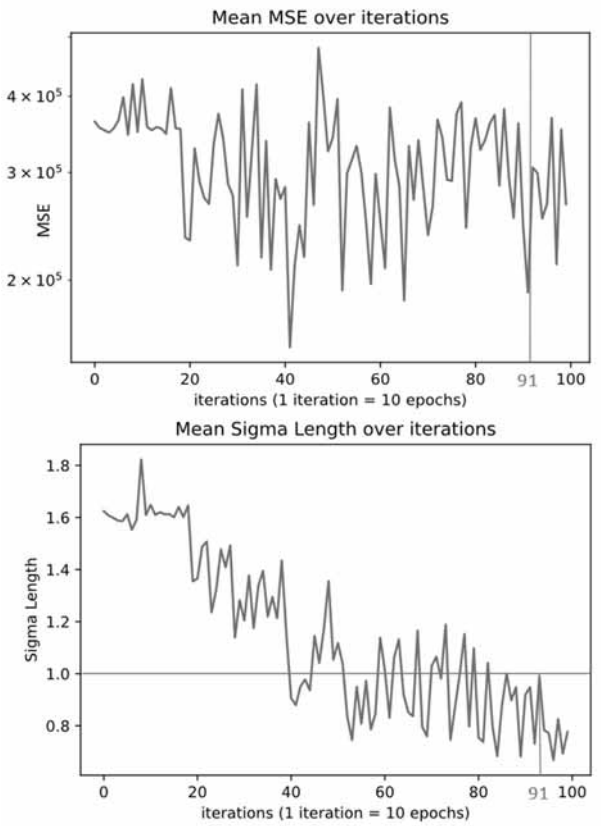

Figure 9: Metrics of the IOM Approach. Top: Averaged *MSE*. Bottom: Averaged *sigma length*. The blue line marks the optimum of *sigma length* (=1). The green line indicates the selected iteration.

4.2 Comparison of metrics

The model can now be used to generate time series. Figure 10 shows all 9 possible variants on which the ML model was trained.

The blue time series was generated by the ML model. The orange time series is the reference time series created by DTW. The visual comparison confirms the functionality of the IOM- approach.

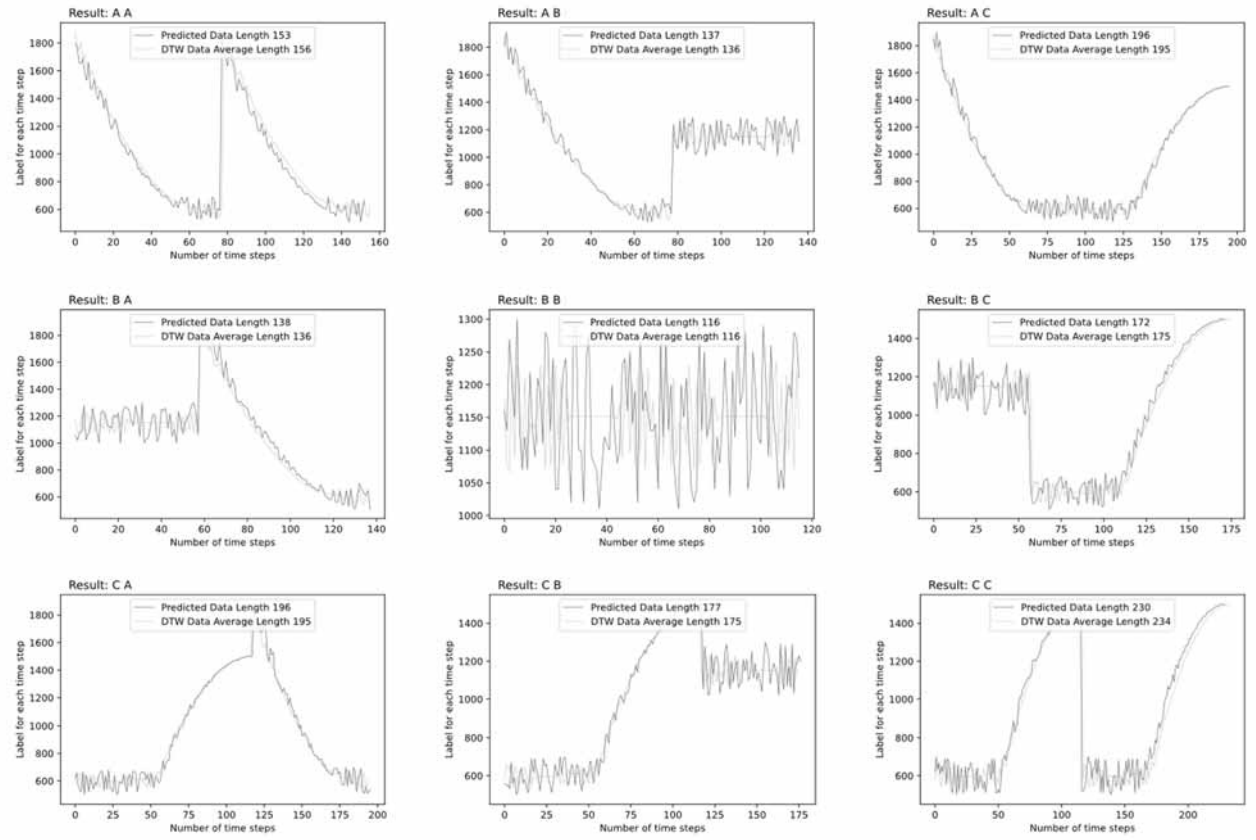

Figure 10: Comparison of all time series created at epoch 910 with the DTW reference time series. The legend in the variants indicates the lengths of the two time series.

5 Critical Review

The presented IOM- approach was able to meet the requirements set for it.

On the one hand, it was shown that synthetic data generated through simulation can be used for training ML models. These synthetic data could be generated cost-effectively, transparently, and appropriately for the specific application.

Furthermore, the implementation of specialized metrics ensured that entire time series could now be compared, not just individual data points via the *softmax* function as in the Seq2Seq base model.

It was subsequently demonstrated that DTW time series can be used to form reference time series for comparison via the metrics. The use of DTW was necessary to address the issue of ambiguity in sets of time series with identical descriptions (e.g., the same NC code).

Finally, it was shown that time series can be generated that hardly differ from the analytically generated *Dynamic Time Warping* time series. The time series are almost identical in form and length.

The method itself offers potential for further research questions. For instance, it could be further validated by significantly enlarging the dataset or generating several different manufacturing orders in the simulation.

Additionally, the ML model here only generated time series from NC codes that were already present in the training data. It remains to be investigated to what extent the model can generate time series in the case of an unknown NC code. This is a general challenge for all generative AI applications as they lack labels for unlabelled data to compare its output to.

A disadvantage of the presented method is the requirement for computational and storage resources. A model must be saved for each iteration, although after determining the optimal model, all others can be deleted. This drawback can be mitigated by applying the comparison of metrics during the training and not just once afterwards. This way models that underperform could be identified and removed sooner. Also, the calculation of the metrics exceeds the time spent here for training the model when done on an epoch-by-epoch basis. This offers room for improvement aswell.

References

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

[2] Boikov A, Payor V, Savelev R, Kolesnikov A. Synthetic Data Generation for Steel Defect Detection and Classification Using Deep Learning. 2021, Symmetry 13, 7, 1176. DOI 10.3390/sym13071176.

[3] Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 2014, in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, Doha, Qatar.

[4] Goodfellow I, Bengio Y, Courville A. 2016, Deep Learning. MIT Press.

[5] Hestness J, Narang S, Ardalani N, Diamos G, Jun H, Kianinejad H, Patwary MA, Yang Y, Zhou Y. 2017, Deep Learning Scaling is Predictable, Empirically.

[6] Jain S, Narayanan A, Lee YT. Infrastructure for Model Based Analytics for Manufacturing. 2019, in 2019 Winter Simulation Conference (WSC). IEEE. DOI 10.1109/wsc40007.2019.9004893.

[7] Ekbatani HK, Pujol O, Segui S. Synthetic Data Generation for Deep Learning in Counting Pedestrians. 2017, in Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods. SCITEPRESS - Science and Technology Publications. DOI 10.5220/0006119203180323.

[8] Kutyev AT, Bekeneva YA. Simulation Software for Generating Data in Monitoring Systems. 2021, 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 10.1109/ElConRus51938.2021.9396098.

[9] Lechler T, Sjarov S, Franke J. Data Farming in Production Systems - A Review on Potentials, Challenges and Exemplary Applications. 2021, Procedia CIRP 96, 230–235. DOI 10.1016/j.procir.2021.01.156.

[10] Loecher M, Perotti LE, Ennis DB. Using synthetic data generation to train a cardiac motion tag tracking neural network. 2021, Medical image analysis 74, 102223. DOI 10.1016/j.media.2021.102223.

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

[12] Martí P, Jordán J, Palanca J, Vicente Julian V. Charging stations and mobility data generators for agent-based simulations. Neurocomputing 484, 2022, 196–210. DOI 10.1016/j.neucom.2021.06.098

[13] de Melo CM, Torralba A, Guibas L, DiCarlo J, Chellappa R, Hodgins J. Next-generation deep learning based on simulators and synthetic data. 2022, Trends in cognitive sciences 26, 2, 174–187. DOI 10.1016/j.tics.2021.11.008

[14] Petitjean F, Forestier G, Webb GL, Nicholson AE, Chen Y, Keogh E. Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. 2016, Knowl Inf Syst 47, 1, 1–26. DOI 10.1007/s10115-015-0878-8.

[15] Petitjean F, Forestier G, Webb GL, Nicholson AE, Chen Y, Keogh E. Dynamic Time Warping Averaging of Time Series Allows Faster and More Accurate Classification. 2014, in 2014 IEEE International Conference on Data Mining, 470–479. DOI 10.1109/ICDM.2014.27.

[16] Sutskever I, Vinyals O, Quoc LV. Sequence to Sequence Learning with Neural Networks. 2014, in NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems. MIT Press, Cambridge, MA, USA.

[17] Tavenard R, Faouzi J, Vandewiele G, Divo F, Androz G, Holtz C, Payne M, Yurchak R, Rußwurm M, Kolar K, Woods E. Tslearn, A Machine Learning Toolkit for Time Series Data. 2020, Journal of Machine Learning Research 21, 118, 1–6.

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

[19] Tobin J, Fong R, Ray A, Schneider J, Zaremba W, Abbeel P. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World, 2017.

[20] Tremblay J, Prakash A, Acuna D, Brophy M, Jampani V, Anil C, To T, Cameracci E, Boochoon S, Birchfield S. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. 2018.

[21] Triastcyn A, Faltings B. Generating Artificial Data for Private Deep Learning. 2018. http://arxiv.org/pdf/1803.03148v3

[22] Woerrlein B, Strassburger S. A Method for Predicting High-Resolution Time Series Using Sequence-to-Sequence Models. 2020, in 2020 Winter Simulation Conference (WSC). IEEE, 1075–1086. DOI 10.1109/WSC48552.2020.9383969.

[23] Wong MZ, Kunii K, Baylis M, Ong WH, Kroupa P, Koller S. Synthetic dataset generation for object-to-model deep learning in industrial applications. 2019. *PeerJ. Computer science* 5, e222. DOI 10.7717/peerj-cs.222

[24] Woerrlein B, Strassburger S. On the Usage of Deep Learning for Modelling Energy Consumption in Simulation Models. 2020, SNE 30, 4, 165–174. DOI 10.11128/sne.30.tn.10536.

[25] Wörrlein B, Strassburger S. Sequence to Sequence Modelle zur hochaufgelösten Prädiktion von Stromverbrauch. 2020, in Proceedings ASIM SST 2020. ARGESIM Publisher Vienna, 149–157. DOI 10.11128/arep.59.a59021.

[26] Wörrlein B, Strassburger S. Hochaufgelöste Energieprofile durch hybride Simulation. 2022, 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 10.11128/arep.20.a2004.

[27] Zell A. Simulation neuronaler Netze. 2003, 4. unveränd. Nachdruck, Oldenbourg, München

[28] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I. Attention Is All You Need. DOI 10.48550/arXiv.1706.03762