Advanced Engineering Informatics 47 (2021) 101197

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Introduction of a time series machine learning methodology for the application in a production system

Martin Hennig [a,*], Manfred Grafinger [a], René Hofmann [b], Detlef Gerhard [c], Stefan Dumss [a], Patrick Rosenberger [a]

[a] TU Wien Institute of Engineering Design and Product Development – Research Area Mechanical Engineering Informatics and Virtual Product Development, Austria
[b] TU Wien – Institute for Energy Systems and Thermodynamics – Research Area Industrial Energy Systems, Austria
[c] Ruhr-Universität Bochum Digital Engineering Chair – Institute for Product and Service Engineering, Germany

A B S T R A C T

Machine learning methods are considered a promising approach for improving operations and processes in manufacturing. However, the application of machine learning often requires the expertise of a data scientist combined with thorough knowledge of the manufacturing processes. Small and medium-sized companies that specialize in certain high value-added, variant rich production processes often lack an in-house data scientist and therefore miss out on generating a deeper data-driven insight from their production data streams. This paper proposes a three-step machine learning methodology to empower process experts with limited knowledge in machine learning: 1) data exploration through clustering, 2) representation of the production systems behaviour through specially structured neural networks and 3) querying this representation through evolutionary algorithms to achieve decision support through online optimization or scenario simulation. The chosen algorithms focus on parameter-light, well-established, general use algorithms in order to lower knowledge requirements for their application.

1. Introduction, motivation

The manufacturing sector is experiencing an unprecedented increase in data availability. This data has a variety of formats, semantics, and quality levels as it comprises e.g. sensor data from a production line, machine tool parameters from production assets, material datasets from suppliers and overhead data [83]. The data sources are utilized in order to address increasing requirements for highly flexible production environments working in a decentralized interconnected manner [19]. Smart Factories are based on Internet of Things (IoT) and Cyber Physical Systems (CPS) and offer data sources and connectivity to other smart systems [47]. The growing accessibility and analytics of large quantities of data in volumes that call for different data handling and processing techniques is often referred to as Big Data. The availability of e.g. quality- or production-relevant Big Data offers the potential to sustainably improve process and product quality [41]. The process to obtain new helpful insights out of Big Data is called Data Mining [37]. Generally, it can be concluded that to utilize the growing availability of data a methodical support is needed to process non-linear relationships, high dimensionality and temporal changes.

Continuously growing demand for highly customized, smart products with short development cycles and high customer requirements concerning product quality and efficiency are some of the challenges manufacturers will be facing in the future [27]. This motivates companies to think about changes to their business processes and aim for new ways to optimize their production, further increase product quality and transform stakeholder requirements into reliable products [15].

Many researchers of the manufacturing research community [36,76,58,29] agree that the main challenges of manufacturing on an international and general level are driving the increasing complexity and dynamics of the manufacturing sector. The tasks not only affect the production processes but also products, production systems, businesses and networks of companies [1]. The field of large data volumes with high complexity is predestined for the application of machine learning (ML) methods, which offer the ability to mine insights from data that alternative methods (e.g. control charts for assessing process stability, monitoring statistical parameters and KPIs, model building for regression or simulation of processes) could not find as they either offer only limited possibilities (e.g. control charts or key performance indicators (KPI)) or require a sound knowledge of the process in order to be able to provide correct statements (e.g. simulations). In contrast, machine learning has the potential to address the new manufacturing challenges. Nevertheless, applying ML methods in a manufacturing environment also adds a challenge itself. The available data is usually characterized by high volume, velocity and variety as well as high dimensionality, partly inadequate quality, different formats, structures, and semantics. Data driven methods for optimizing manufacturing problems must be

* Corresponding author.
E-mail address: hennig1martin@gmail.com (M. Hennig).

https://doi.org/10.1016/j.aei.2020.101197
Received 11 October 2019; Received in revised form 24 September 2020; Accepted 26 October 2020
Available online 14 December 2020
1474-0346/© 2020 Published by Elsevier Ltd.

Nomenclature

DWT	Dynamic Time Wraping
KPI	Key Performance Indicator
LSTM	Long Short Term Memory
MIMO	Multiple Input Multiple Output
MISO	Multiple Input Single Output
ML	Machine Learning
MNN	Modular Neural Networks
MSE	Mean Squared Error
NARX	Nonlinear Autoregressive Neural Networks with External Input
NN	Neural Network
R	Correlation Coefficient
Sec	Seconds
SME	Small and Medium-sized Enterprises
TS	Time Series

able to function under these conditions or at least offer the capability to be combined with data transformation and cleansing algorithms that address these issues [80]. Hence, there is a broad expertise of available ML tools needed to find the right one. This lack of methodological knowledge poses a problem for small and medium sized enterprises (SMEs) in manufacturing which start to implement Industry 4.0 aspects and struggle to process the new data volumes for new insights [34]. Often SMEs already have knowledgeable process experts for their production assets but lack an on-site data scientist or ML engineer, who is needed to implement data mining methods from the very start. Hence a methodology is needed that offers predefined steps to ease ML requirements and workhours of an external ML expert, and it also needs to be able to manage data with hidden context structure and different characteristics.

Therefore, this paper presents a three-step time series ML methodology, which includes data exploration, a learned system representation, as well as an approach to run optimizations on this representation, for example for decision support. When applying the methodology, a production line's time series (TS) are clustered and subsequently modelled with a modular neural network (MNN). The model is then optimized by querying the MNN systematically in order to find data patterns for the production line's input variables that will result in an optimal objective function value which is use case depended.

The main scope of this paper are the following aspects:

1) Introduction of a ML methodology which combines a clustering of TS, representing them with MNNs and using this representation for optimization purposes like parameter tuning, in order to gain more benefits combined than the algorithms alone. The intention is to use the results of each step in the next one with the aim to achieve an advantage than from the application of each algorithm separately.
2) Presenting an approach to integrate TS cluster and plant topology information into an MNN.
3) The utilization of a grid search in preparation for an easier setup of NNs for ML novices.
4) Showcasing the clustering and representation step of the methodology on a real-world public dataset and demonstrating the optimization step on a synthetical optimization task on the same dataset.

The methodology aims to support process experts with limited knowledge in ML. Hence, it includes a set of and a recommendation for validated ML algorithms at each step of the methodology. The chosen algorithms all have in common that there is only little parameter tuning necessary which simplifies their application. For the parameterization of the neural network (NN) in the second step a spreadsheet interface is provided, which offers means to try out many parameter configurations systematically.

The practical application of the proposed methodology will be demonstrated by applying it to a public dataset of a test setup of an industrial winding process. The aim is to analyse the data to find structure, learn a representation of the system behaviour with the help of this structure and then use this representation in an optimization. This can be used for decision support purposes for the production line personnel for example when deciding which parameter settings to use to accomplish a desired output value (e.g. measured quality related outputs) with non-tunable input values (e.g. measured material properties). Although one specific use case is shown in detail the methodology is kept generic and offers many other benefits that can be employed likewise.

The following Section 2 gives an overview over the state of the art regarding ML as a tool for manufacturers and each step of the methodology. Section 3 presents the methodology concept in its three steps. In the first step, a data exploration through unsupervised TS ML is presented. The second step is about learning a system representation, which uses the structure that was found in the previous sections to train a neural network (NN). Then, in the third step the learned system representation is queried in an optimization with evolutionary algorithms. The paper continues with a proof of concept in Section 4 and closes with a conclusion followed by an outlook and future work in Section 5.

2. Related work

The concept of ML is known for several decades and its general understanding still holds. It is a set of methods for programming computers, to learn to solve problems or make predictions based on training data rather than a predefined procedure [69]. ML methods deliver a promising approach for addressing the key future challenges in the manufacturing sector especially in the context of industrial big data.

When ML is applied to industrial TS, the most popular approaches are unsupervised ML [79] and TS prediction with NNs [18,31]. Both mentioned research subfields have various application areas. In a manufacturing context unsupervised ML and prediction NNs are often used for predictive maintenance [81,20,13]and quality monitoring or anomaly detection [82]. The approaches taken in these application fields are often similar. At first the data is pre-processed and cleansed of outliers, before different features of the data are calculated which presumably correlate with the behaviour, that should be modelled (e.g. wear, normal / abnormal behaviour or quality defects). Then a ML algorithm is applied to these features and/or the data to either find structure in the data itself (unsupervised ML) or to model the relationship of a target behaviour to input data patterns (supervised ML).

2.1. Data exploration through clustering

TS ML is special because originally, ML algorithms were developed to analyse static or structured data, however, advances in technology made data acquisition and storage easy and cheap. These advances led to an increase in the acquisition of TS which are dynamic and often less structured by default [30]. Nearly every numerically describable aspect that changes over time can be viewed as a TS, but at first, popular ML techniques could not directly be applied to them. This restriction was due to the large size that TS can have, their high dimensionality, noise, or co-dependencies [80]. To address these problems of applying ML techniques to TS a suitable pre-processing, an aggregated data representation, and an adapted concept of similarity are needed.

A major part of the requirements can be addressed by searching for patterns in a TS that reoccur similarly at different occasions. These patterns are called TS motifs. Motifs are potential candidates in a TS clustering, since simply using all subsequences that can be extracted with a sliding window would result in a meaningless clustering. This problem was reported by Keogh, Lin et al. [49] and proven theoretically by Idé [45]. Motifs in an industrial manufacturing data stream can be

corresponded to typical process states that a production asset undergoes in a normal operation.

As it is possible to build a similarity matrix over all motifs, popular clustering techniques can be applied (e.g. k-means [54], OPTICS - Ordering Points To Identify the Clustering Structure[4], agglomerative linkage [73]. Those techniques do not have to be specific for dynamic data anymore since they work on a similarity matrix instead of raw data. Through the choice of clustering algorithms is broadened from specialized TS-specific algorithms to a wide variety of clustering algorithms. Mueen, Keogh et al. (2009) already presented a suitable algorithm for unsupervised TS ML that will be adapted for this paper. The adaption also follows principles of Keogh, Lin et al. [49] and other publications from that research group [50,78,84].

With these requirements of applying ML to TS met, there is still the problem of parameter tuning that hinders manufacturing process experts with limited ML experience to apply existing ML algorithms to their data. As a meta-study by Chiu [22] found, many ML algorithms exist and many claim better accuracy than rivalling methods in the tasks on which they were reported on. Often this good reported performance is due to cherry picking in the training, testing and validation data, due to a posteriori parameter tuning or due to the lack of a common test data basis. Validating claims made by proposed algorithms are difficult to investigate. It would be better to evaluate algorithms on well-known benchmark data or to use algorithms that only have few parameters. Therefore, parameter-free or parameter-light algorithms [78,50]are chosen for the scope of this paper and publicly available test data used by the research community [11,10,25].

2.2. Learned representation

As already mentioned in this chapter's introduction, predicting TS by using NNs is a popular task in applying ML techniques to industrial TS data[56].

Because NNs offer many degrees of freedom when creating them, practitioners struggle to find the right form of network to learn the behaviour of their data. Therefore, a technique is required that eases the task of structuring of a NN. A well-established approach of this is to split the NN into pieces that are more manageable and to mimic the problems structure in the model structure. To achieve this, modular neural networks (MNN) are used in this paper for predicting a production system's TS and modelling its behaviour. MNNs have been known in ML research for three decades [70,5,2]. Especially the aspects of MNNs with manual formation through expert knowledge as in Nardi, Togelius et al. [61] or Anand, Mehrotra et al. [3] are employed in this paper. This manual formation aspect helps to integrate the process expert knowledge about the plant topology in the NN training to improve the NN learning performance. The plant topology describes upstream and downstream dependencies and therefore cause-effect relationships in the production line [85].

TS cluster information are another information that can help the NN to better learn the structure which is present in the production data. For their integration, a dataset for the training of a NN is not split for cross validation at random, but cluster-wise. This reuse of cluster information for the NN training was adapted from Liu, Du et al. [53] as well as Nguyen [62]. Alternatively, to using cluster information for dataset segmentation is the provisioning of the cluster label as an additional input to the NN.

2.3. Optimization with evolutionary algorithms

To maximize the benefit of a predictive model one option is to query the model systematically to find optimal input parameters for a desired system output. This can be used for example to offer decision support for technical plant personnel in operation of the production assets by predicting the system behaviour. In the proposed methodology trained MNNs are used in optimization applications to find better production parameter combinations within those inputs, which are controllable by an operator. Because the MNN are based on discrete clusters and may have discrete input parameters, it might happen that the space of possible solutions for a given optimization task is discontinuous or has integer discrete steps. This difficulty together with the many and complex variables involved calls for an optimization method that can manage high dimensional problems with discontinuous solution spaces and non-linear system behaviour. This requirement reduces the choice of appropriate optimization methods. Because the learned representation is used as a data-driven simulation model of a production line the authors have researched simulation optimization methods.

The field of simulation optimization methods can be structured in different ways Fu [32] and Carson and Maria [17] classify them into response surface methodology, gradient-based search methods, stochastic optimization, statistical methods and heuristic methods. An approach to offer support in choosing the right simulation optimization method is presented in Jalali and Nieuwenhuyse [46]. There the main questions in guiding this decision is whether the decision variables are discrete or continuous, whether the objective function is differentiable and whether the set of feasible solutions is small or large to infinite. Following this study, the choice of possible simulation optimization methods for the proposed methodology is reduced to meta-heuristics and ordinal optimization. Ordinal optimization requires setting non-trivial problem-dependent parameters which is why it was not chosen here [42]. Meta-heuristics comprise many approaches like particle swarm optimization, genetic algorithms, evolutionary algorithms and artificial bee colony algorithms. They all have in common that they do not need a differentiable objective function and that work through balancing aspects of exploitation and exploration. Their disadvantage is that they converge more slowly, they need some kind of strategy parameters to be set and that optimality of found solutions is not guaranteed [21,63]. Nevertheless are they popular especially in multi-objective problems which cannot be scalarized to a single variable [46]. The authors chose to use an evolutionary algorithm because it offers possibilities to make problem specific strategy parameters (like the mutation rate) part of the solution variables which creates a meta-optimization for those values. This aspect has the advantage of accelerating the algorithm's convergence and eases the choice of parameter settings for a practitioner. Evolutionary strategies can work with these requirements and are they known to achieve great exploration and exploitation in a high dimensional solution space [67,71,65,28,64].

Evolutionary algorithms compile solutions of a problem at hand into genes, select fitting solutions, and then proceed to work with a population of solutions in a problem-independent way (e.g. by searching solutions through recombination and mutation). This is different from regular optimization algorithms, which work with single-point solutions in a problem-specific way (e.g. gradient descent).

An evolutionary algorithm's objective function either can be a variable, which is already part of the NN representation, or it can be a performance index, which is calculated by combining several variables of the trained network (e.g. a KPI). This indicator is used as a fitness indicator in the selection operation that guides the optimization. If several variables are used the user must consider how these are weighted and combined because a pareto optimality is likely to occur. This means that there is not a single best solution to be found but a pareto frontier, which is a hyperplane of (pareto optimal) solution combinations where no single criterion can be improved without making at least one other criterion worse.

3. Concept

The proposed methodology comprises three consecutive steps, as depicted in the graphical abstract. The first step is exploratory and concerned with clustering the TS. Structure is found by grouping reveal reoccurring similar patterns together and interpreting the groups in a meaningful way (e.g. a machine operation state). In the second step this

structure is utilized, training a NN as a model for the system's behaviour to obtain a learned representation. The structure of the NN is modularized and the training data is enriched with the cluster information to ease the training complexity and to represent the system topology more accurately. In the third step, this trained NN is queried in an optimization with evolutionary algorithms, which can be beneficial in online parameter- optimization. The learned representation can also be used in simulating the near future, detecting anomalies, and deriving a decision support in machine operation or predictive maintenance.

The aim of this methodology is to provide a set of tools and defined steps to explore relationships in data, train a self-learned model for the data and to query this model or to run optimizations on it. Separate consecutive ML steps are combined to achieve synergetic effects by integrating the results. In each step, new knowledge is derived through ML, which is then used in the following step. It is intended that the process expert's input and their critical interpretation is essential at some steps and that the methodology is by no means an automated artificial intelligence for manufacturing operation. It should lower the requirements for an external ML expert and help to empower existing process experts, although they have limited knowledge in ML, to derive insight and hidden relations from their collected manufacturing data.

The novelty of the methodology lies in the connection of these different algorithms and in the creation of a concept that is usable by practitioners in the manufacturing field who are not ML experts.

3.1. Data exploration through clustering

Before the application of the methodology the data is assumed to be pre-processed and reduced in dimensions. Although most of the later used algorithms can work with high dimensionality and outliers, the expected results improve after the variables are transformed with a principal component analysis and obvious outliers are removed.

Starting with an already pre-processed dataset, the proposed methodology begins with searching for patterns that reoccur similarly at unique occasions. These so called TS motifs are useful cluster candidates in a TS clustering, since they help the following algorithms to focus on the similarity relationships of the occurring data patterns [49]. For the extraction of TS motifs, which are used as features, the algorithm by Mueen, Keogh et al. [60] was be adapted for this paper. For the extraction of motifs, it has to be decided what size a typical TS pattern should have. This is also known as the TS motif length and is one of the few parameters the methodology needs. This is found either by examining the raw data or by process knowledge. It is also possible to do a grid search over possible motif lengths and evaluate which one produces meaningful results, but often the motif length is given as a process work cycle time.

With the extracted motif subsequences a similarity matrix has to be built in order to apply general clustering methods to the dataset. This is done by calculating the Dynamic Time Wrapping Distance [12] of each motif subsequence to each other and apply a agglomerative linkage [73] clustering to it. This produces a hierarchical clustering. The optimal cluster count can be determined by heuristics, by inspecting the clustering hierarchy visually or by production process knowledge. Through interpretation with a process expert the clusters can be associated with machine states which is a valuable information that helps to structure and enrich the production line's TS. After the machine state clusters are found a nearest neighbour classification can be used to assign incoming new data to known machine states. The choice of algorithms for each of these steps was the result of a large comparative calculation [38].

The data exploration provides the basis for the following process steps in the presented methodology, as it provides structure which helps in the following training of a regression model with NNs. Furthermore, it already delivers several utilization options as an intermediate result. Clustering TS data to reoccurring machine states generates metadata that can be used in further semantic processing of the data. This metadata makes it possible to view a TS as a sequence of meaningful discrete states additionally to the continuously changing point values. This can already be utilized for:

- Calculation of KPIs e.g. by linking clusters to value-adding and non-value-adding machine operations.
- A diagnostic support in machine monitoring (e.g. is the machine running in a typical state or has an anomaly) and (predictive) maintenance applications (e.g. through the association of certain clusters that occur in faulty or worn-out assets).
- If a TS is available that represents performance or quality data, a data exploration can deliver even more benefits. A further benefit is gained by finding relationships between certain process states or process parameters and product quality. By discovering this relationship, it becomes possible to classify if a current process state corresponds to a known cluster that is associated with a certain quality. In case, that a cluster is associated with sub-optimal quality and it exists a cluster for the current product but with a better corresponding quality, machine operators can be informed. Together with this better cluster, the information could suggest machine parameters that produced better results in the past.

3.2. Learned representation

A data-driven representation through NNs is utilized to model the system behaviour for enabling to predict a production line's output to a certain input which is beneficial amongst others for an optimization. NNs are a used for this step because they are an approach which has proven very good results in real-world applications that are high-dimensional, complex and governed by relationships which are difficult to describe with analytical formulas [56]. Through the structuring of networks into connected layers of neurons and the non-linear activation function, NNs offer remarkable information processing and learning capabilities. [9]

NNs are known for being able to learn very complex patterns in non-Gaussian data. With them is also possible to use a process' obvious input values as well as measured intermediate data streams as training input to learn the targeted process output data. This has already shown beneficial in learning industrial application data[44].

Usually setting up a NN is a tedious task because many parameters have to be decided which contradicts the methodology's claim to be parameter light and not require many workhours of a data scientist. Therefore, the authors created a MATLAB script through which the setup of a NN training can be done through filling out a spreadsheet table in excel and running the script. The script connects the dataset's dimensions with the NN inputs and outputs and helps to provide references to organize the separate networks into an MNN later. The spreadsheet table helps to set general training parameters (e.g. training function, training repetitions, max number of epochs) to often used default values. It also lets the user set network specific parameters (e.g. node count in input, hidden and recurrent layers, splitting the data into train / validation / test data). Because each network training configuration is only one spreadsheet-table row the configurations can be varied systematically. This opens up the possibility to evaluate a whole range of possible parameter configurations and utilize design of experiments approaches [59] to find the optimal configuration. In this context an experiment is a NN training, the parameters are the NN training configuration parameters which cannot be set to default values. Similar approaches have be used in many works before and proved to simplify the NN configuration process [75,77,8,51]. As a preliminary study it was investigated which Design of Experiments approach yielded a good balance between training calculation time and training accuracy of the found NN configuration for the use case presented in chapter 4. It was concluded that a full-factorial or fully-crossed design or at least an approach that uses a multilevel orthogonal array which is mapped to the network parameter range worked better than other experiment designs [74]. The code of the script and the spreadsheet table are available at

[39].

The learning model in the proposed methodology is an MNN, which uses the previously found cluster information on the one side and the plants' topology on the other in its training.

3.3. Integration of cluster information

Usually, the TS would be separated for cross validation using a random train-, validate, test-data split. But giving the NN unclustered data directly forces the network to learn desired standard operating conditions while they are mixed with frequent state transitions and possible anomalies. This results in poor accuracy or a necessity of more layers and neurons, which take more calculation effort in training. Here the separated motif subsequences from the previous data exploration are used to train the network instead. Liu, Du et al. [53] as well as Nguyen [62] showed that using clusters instead of unstructured data brings an improvement in training accuracy. This will be validated in the following proof of concept. Using the structure of previously identified clusters as training data simplifies the networks learning task because it reduces complexity and focuses the training on normal steady-state behaviour. Even if the network has to learn all clusters, knowing the patterns starting points and using these in the training of the network helps to learn the reoccurring structure. The differentiation between standard operating conditions and transitions or anomalies is hence not necessary to be learned in training because this structure is already given by the clustering.

3.4. Forming an MNN using the plant's topology

The values measured in a production line often originate from different machines and locations along the sequence in which a production line is passed by a workpiece and changes its states and value [82]. This flow can be described in a graph-like structure, the plant topology. A NN indirectly has to learn these relationships to represent the system behaviour. A useful tool to get an overview over the plant topology is an adjacency matrix which helps to document variable dependencies. In the modelling of the relationships circular dependencies should be avoided because the methodology in the current state is not able to model them.

Using the plant topology also helps in splitting an otherwise large network up into modular subnets that only represent one production asset and its connected process aspects. In simple terms this means that subnet represents a relevant machine in a production line. Through the training of subnets, the learning problem is much simpler than training a model that represents the whole system with all its inputs and measured outputs at once (in a Black Box manner) [2]. Training in a Black Box manner requires more computational effort because more nodes are needed, as the system behaviour is more complex. Even if the network node count would be the same as used in the subnets, there would be more connections and therefore trained weights because all inputs and outputs are considered and the network is fully connected. Because training a NN is an NP-complete problem, the training effort grows exponentially with the number of weights [14,23]. There are algorithmic and technical approaches, which accelerate the training for example by using stochastic gradient descent or executing the training on hardware which is optimized for matrix operations (e.g. graphical or tensor processing units). But the NP-complete problem characteristic is still a valid argument for looking into splitting a learning task in smaller easier to learn submodules.

This split makes it also possible to individually find the optimal network parameters in a parameter grid search for each subnet. Such a hyperparameter tuning identifies the optimal trade-off between training effort and accuracy. Instead of training one large and complex Black Box model, several smaller Black Box models are used. The smaller models only learn a small section of the system at once and are linked according to the production line topology.

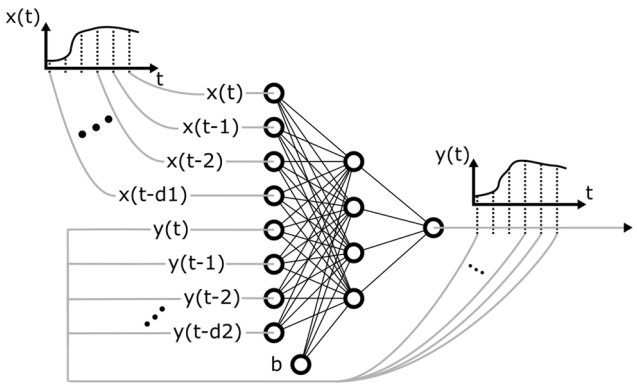

Fig. 1. Schematic of a NARX net. n stands for the node count in the hidden layer, d1 for the number of input delay nodes and d2 the number of feedback delay nodes.

Because the system's behaviour is expected to be dynamic and nonlinear a Nonlinear Autoregressive neural network with external input (NARX) is used to learn its behaviour. In several works, it was shown that NARX networks "[…] consistently outperform standard NN based predictors, such as the TDNN [feedforward time delay neural network]." [57]. NARX networks use their output values as an additional input to the network (as shown in Fig. 1). This enables them to not only from external input but also from their own state. This represents many dynamic processes better than simple feed-forward networks [52,26].

3.5. Optimization with evolutionary algorithms

Supplying a trained NN with an input and calculating an output according to its training is simple and fast, although the learned behaviour is complex. The fast network response enables operations that do many queries over a short time. Having a data-driven representation model of a production line's time series is beneficial but its full potential will only be exploited if this model is queried systematically. An exemplary use of this is to derive decision support in the production line's operation.

The general idea is that a population of different parameter-settings will be tried out with the previously created learned representation. Each individual of this population contains a set of valid input values whose feasibility is ensured through the creation rules in forming new genes. Since the trained MNN has recurrent nodes in its input layer it needs several timesteps to produce a valid output. With the concatenation of such networks the required number of timesteps increases. The set of valid input values must therefore consist of enough timesteps to get a valid output from the whole MNN. Furthermore some evolutionary algorithms take strategic optimization parameters like the mutation rate as part of the genome [7].

In a second step of the evolutionary algorithm each individual is processed as an input for the trained MNN and is assigned a fitness value according to an objective function and fitness mapping. This function is either a single network output or a combination of multiple (e.g. like a KPI). If the optimization goal or iteration threshold is not jet reached the individuals are sorted according to their fitness. A fitness-depended selection rule determines which of the individuals are selected as parents that create offspring and form a new generation of individuals. The selection rule aims to find a balance between convergence (choosing elites) and a large exploration of solution space (choosing less fit individuals in order to steer the population into a different region towards the global pareto optimum hypersurface). In order to utilize both contradicting effects the selection rule elitism (or selection pressure) will be low at first and increased over the generations. Each individual is evaluated and the ones representing the best solutions are getting the highest chance to be selected as parents for offspring.

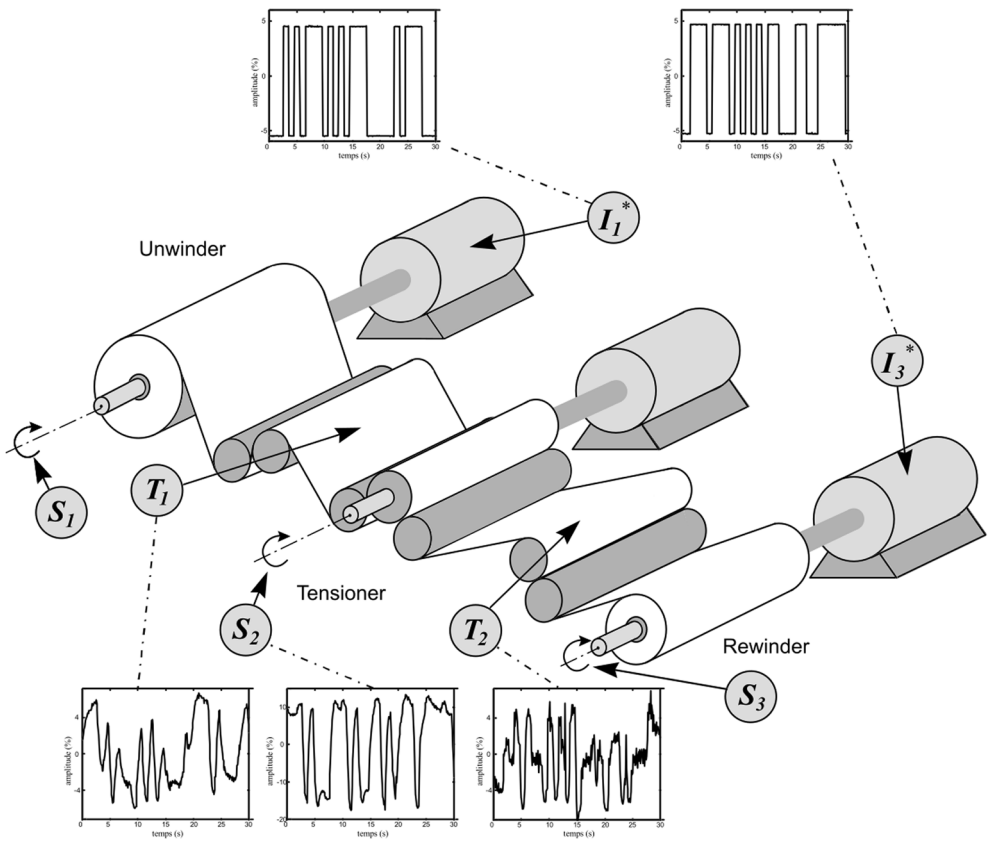

Fig. 2. Process schematic of the proof of concept use case. Taken, translated and adapted from [11,10].

To generate offspring either a solution is mutated or two parents are recombined by creating new individuals that are located in a valid area between the two parent values. The recombination used here is an extended version of the original line recombination [66] which acts like an interpolation but with random weighting and a chance to "shoot" a bit over the line of interpolation. Mutation is done by adding zero-mean gaussian noise to the variables. Because the mutation step size is part of the optimization variables the mutation operator will cause an accelerated convergence towards high-fitness solutions, otherwise it will contribute to the search for alternative and new solutions. After this the new individuals together with some parents are selected depending on their fitness and create the new generation. This ensures that the overall fitness rises while also many different solutions are tried before converging to a pareto optimal one. The pseudo code for a generic evolutionary algorithm is shown in Algorithm 1. For clarity, the transition from and to between solution-space and an embedded space is omitted.

In preliminary research many operators for selection rules and population compositions were researched and compared. The details of the comparison are beyond the scope of this paper, but as a result for the methodology a recommendation for operators to be used was determined. They are as follows:

- It is favourable to create a population consisting of 20% parents and 80% children [6].
- The determination of an individual's breeding probability should be done by mapping it to a linear fitness ranking [63]
- The selection rule for recombination should be tournament selection with the size of three individuals [35].
- As an evolutionary algorithm is a stochastic process the optimization should be repeated several times to get a more robust result. Only the best solution over all runs should be utilized.

Algorithm 1. (*Pseudo code for an evolutionary algorithm*)

```
Define the optimization problem by setting the objective function,
  goal, input variables and their ranges
Define evolutionary operators and optimization parameters
Initialize a population of random individuals within the range of
  the input parameters
Do
  Adjust selection pressure
  Calculate the individual's fitness through evaluation of the
  objective function
  Transform the individual's fitness into a selection-likelihood by
  sorting and ranking them, taking selection pressure into account
  Crossover: Apply a selection rule to the individuals to choose
  individuals for crossover at random
    Apply crossover operator to them
    Calculate their fitness and selection-likelihood
  Mutation: Apply a selection rule to the individuals to choose
  individuals for mutation at random
    Apply mutation operator to them
    Calculate their fitness and selection-likelihood
  Merge crossover and mutation results to children population
  Form a new generation according to the population composition
  Sort population and find best individual
  Decrease mutation step size
While (goal.reached != true or population.converged != true or
  population.counter < max_populations)
```

The benefit of an optimization in this methodology is mostly use case depended and even if a reasonable objective function can be formulated, there remains the question of choosing a starting system state to originate the population from.

If an evolutionary algorithm is chosen for auto parameter tuning the problem of choosing starting conditions is omitted because the starting condition is already set by adopting the current system state. Once an

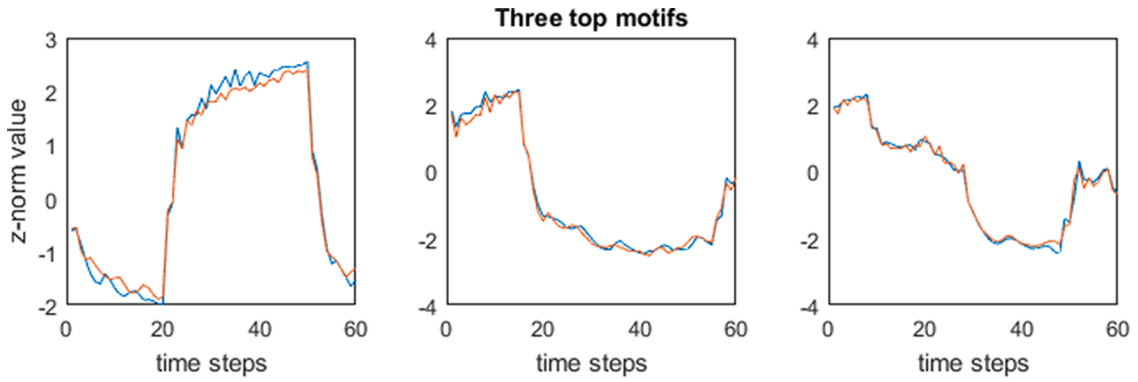

Fig. 3. Result of the TS motif search step of the methodology.

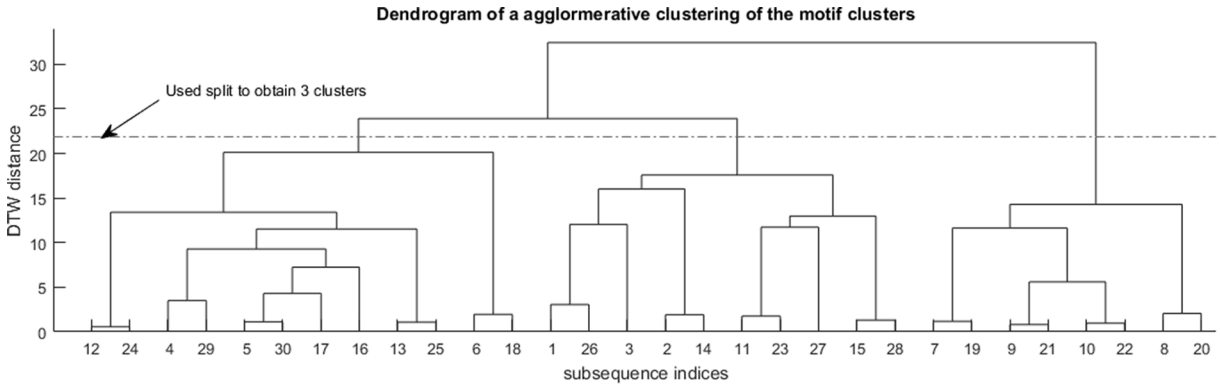

Fig. 4. Result of the TS clustering step of the methodology.

objective function is defined and boundary conditions are set the optimization can be executed in a loop with the currently measured system state as an initial optimization point. This would offer a decision support for online process parameter tuning without the need to define an optimization scenario by searching of a realistic starting point.

4. Proof of concept

This use case was chosen because it has defined input and output TS and a known logical dependence of the production equipment components (e.g. plant topology). The process is a test setup of an industrial winding process as depicted in Fig. 2. The data is publicly available and intended for testing system identification approaches. In the proof of concept use case a plastic web is unwound from a first reel, tracked over a traction reel and finally rewound on a third reel. Tachometers measure the angular speed of each reel (S1, S2, S3) and tension meters measure the tension between the reels (T1, T3). There are DC-motors and reduction gears coupled to the reels. Each DC-motor is controlled by a regulator which adjusts the motor current (I1 and I3) while also monitoring the angular speed. For optimization purposes it is estimated that the desired objective property of the process is a consistent mid-level tension (T1 and T3) on the plastic web. The angular speeds are the process variables and the motor currents (I1 and I3) are the controllable input variables of the system. The plastic web tensions are the objective variables for which a model training and an optimization is conducted [11,10,25].

4.1. Data exploration through clustering

After preprocessing a first aspect to consider is the selection of a suitable motif length. In this example upon inspection, it could be seen that similar patterns were occurring with a typical length of 60 timesteps, which could be verified by a grid search. After data cleansing, reconciliation and dimensionality reduction with a principal component analysis, the first principal component is analysed with a motif-search and clustering algorithm. The algorithms used here are the classic MK_Motif Search Algorithm [60]to find the candidates, Dynamic Time Warping (DTW) [12] to build a distance matrix of those candidates, and “Weighted Pair Group Method with Arithmetic Mean” [73] to create an bottom-up agglomerative clustering to find a hierarchy in the candidates.

The MK_Motif Search Algorithm found motifs of 60 timesteps lengths, stored the most similar TS subsequence pair and excluded both subsequences from the further search of the remaining TS. This process is repeated until no full motif could be found any more. The resulting motif pairs were found to be often 1000 timesteps apart, which can be the indication of a process repetition or super-cycle. Fig. 3 shows the top three motif pairs, which were all 1000 timesteps apart. After the search algorithm, the resulting pairs were separated and with each individual TS subsequence a similarity matrix was build, that contains the DWT distance from each subsequence to every other. This matrix served as an input for the clustering that followed.

Table 1
Adjacency matrix of the use case topology.

		Inputs							
Outputs		I1	I3	S1	T1	S2	S3	T2	
	I1								
	I3								
Unwinder Reel	S1	1							
Tensioner Reel	T1	1		1					Global Output
	S2	1		1					
Rewinder Reel	S3	1	1		1	1			
	T2	1	1		1	1			Global Output
		Global Inputs							

to train a classifier.

4.2. Learned representation

To prepare the use case's "production line" topology it is useful to describe the dependency of the variables with an adjacency matrix. It helps in structuring relationships and makes clear which variables are determined by which. In this use case a reel's rotational speed is assigned as an output to the reel and the net tension between two reels is assigned as an output to the latter.

After the adjacency matrix as in Table 1 is built a layout of an NN can be derived from it. In this layout, there is a subnet for each aggregate. As can be seen in Fig. 5 (also compare to Fig. 2) the only input aspects into the modular network are I1 and I3. Reel rotation speeds are assigned as output aspects for the subnets that represent the respective reel. Tension aspects that result from the interaction between two reels were assigned to the downstream (later) reel. Circular dependencies or upstream interactions are either omitted or simplified into parallel placed subnets, because otherwise a training of separate networks would be very complicated and would require a co-training of both networks at the same time.

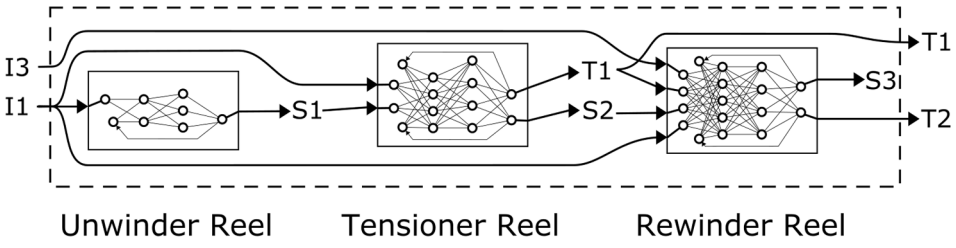

Fig. 5. A modular neural network (MNN) is formed according to the known topology. Each reel is represented by a separate subnetwork. Each reel's subnetwork is dependent on the amperage setpoint of the reels driving motor. Each reel's subnetwork has the reels angular speed as objective. Tension between two reels is trained as an output of the downstream (later) reel. The dashed box indicates that the whole "production line" and therefore the resulting has MNN only 2 controllable inputs and 2 outputs, which are relevant for product quality. The number of nodes in the visualization is not the actually used node count.

Table 2
The influence on using cluster information for the training of NNs. R is the Target-Prediction Correlation Coefficient, which indicates the error the neural network makes in its test prediction.

		S1	T1	S2	S3	T2	Improvement compared to Benchmark
NN Configuration*		n = 25 d1 = 1:5 d2 = 1:3	n = 15 d1 = 1:5 d2 = 1:3	n = 20 d1 = 1:5 d2 = 1	n = 35 d1 = 1:15 d2 = 1:8	n = 20 d1 = 1:5 d2 = 1:4	
One network per Output + Random Blocks (Benchmark)	R	0.561	0.909	0.315	0.499	0.924	Higher is better
	MSE	0.00878	0.0123	0.0280	0.0194	0.00353	Lower is better
	Calculation time in sec	1.103	0.860	1.207	150.6	1.963	Lower is better
One network per Output + Use all clusters as blocks	R	0.801	0.943	0.630	0.777	0.940	40.79%
	MSE	0.00345	0.00752	0.01499	0.00536	0.00194	111.4%
	Calculation time in sec	1.092	1.014	1.0181	153.2	2.127	−1.75%

* In the NN configuration n stands for the node count in the hidden layer, d1 for the number of input delay nodes and d2 the number of feedback delay nodes as shown in Fig. 1.

The search for a good cluster count[1] suggests a high number of clusters that indicates a chaotic behaviour. The dendrogram[2] as seen in Fig. 4 brings a more insight and lets the practitioner see that, despite the dissimilarity of motif pairs, he or she could choose a cluster count of three or alternatively two to structure the data with coarse clusters and

For each network mentioned in this paper, a two-stage[3] grid search over all reasonable parameters was conducted to find suitable parameters for each net. Suitable means that the calculation time for training the network stayed below 3000 s and that the networks prediction performance on an held-out dataset did not improve upon adding more

[1] This means the median of the suggested cluster count produced by Calinski-Harabasz criterion [16], Davies-Bouldin-Evaluation index [24] and Silhouette values [68]

[2] A dendrogram is a visualisation format for hierarchies and often used in biology. In cluster analysis it is used to show which cluster vertical lines are similar by the vertical lines. Similar clusters can be merged into superclusters. Their distance is represented by horizontal lines. Superclusters can be merged again with other supercluster or with unmerged single clusters at the displayed distances. This visualizes a higher structure in the data. It is recommended to "cut" this tree structure if the desired cluster count is reached or if there is a sharp increase merged distance for the next higher supercluster in order to find the data cluster count.

[3] Two-stage means that a grid search was first conducted with large step intervals followed by a smaller grid search in the neighbourhood of the first stage's best candidates but using smaller intervals and more repetitions.

Table 3
Improvements of using one network per output vs one network per production asset.

		S1	T1	S2	S3	T2	Improvement compared to Benchmark
NN Configuration[4]		n = 35 d1 = 1:20 d2 = 3	n = 30 d1 = 1:40 d2 = 3	n = 30 d1 = 1:20 d2 = 5			
One network per Reel + clusters as blocks	R	0.561	0.939	0.726	0.769	0.924	37.6%
	MSE	0.00878	0.0111	0.00352	30.4%		
	Calculation time in sec	1.103	4.196	401.0	−162.9%		

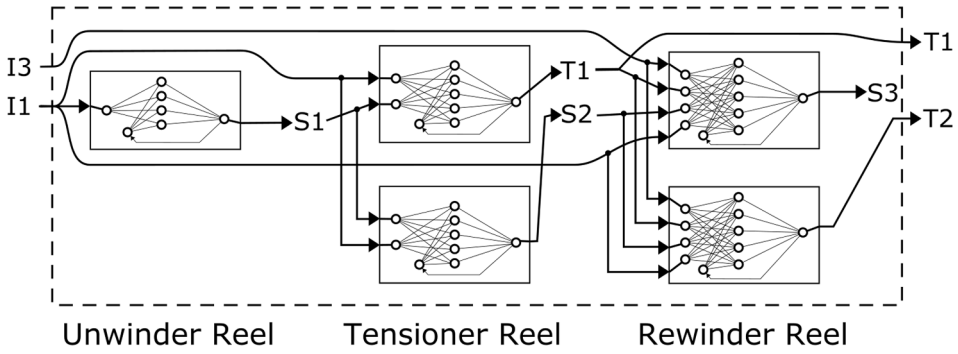

Fig. 6. An MNN in accordance with the known topology and Fig. 5, but with a separate network for each training objective. This way the training tasks are simplified. Each subnet is dimensioned with a parameter grid search to find an adequate and required neural network complexity. Therefore, some subnetworks are simpler than others are if, for example, they have fewer inputs and their objective is easier to learn.

nodes, similar to Setiono [72]. To rate the networks prediction result two indicators are employed. R as the Target-Prediction Correlation Coefficient and the Mean Squared Error (MSE), which both indicate the error the NN makes in its prediction on the test dataset. Because some networks have multiple output elements which can have largely different ranges the MSE is scaled relative to the original objective data to the range [−1,1]. Also, the time duration for training each network is reported as calculation time. For rating purposes, the relative improvement to the benchmark network (using one subnetwork per output and randomly chosen training datasets see first row of Table 2) is recorded. Since for R higher is better and for the MSE and the calculation time lower is better, the ratio to the benchmark is subtracted with one for the relative improvement of R and subtracted from one for the relative improvement of the MSE and calculation time.

Because the TS would usually be separated for cross validation at random into training, validation and testing datasets, this configuration together with a one-network-per-output approach is used as a benchmark. To incorporate the results from the data exploration through clustering step the cluster structure information is used to split the dataset instead of using a random split. In doing so it has to ensured that, if possible, the new data split contains instances of all clusters in all three datasets without repetition. This is achieved by a simple recursive algorithm that evaluates cluster instance occurrences as split point candidates beginning from seldomly occurring clusters and proceeding with more frequent ones. The claim that this improves the training accuracy is confirmed for the use case in Table 2, which shows that the training using the motifs indices as training dataset blocks is more accurate than using random TS blocks. The MSE could be halved and R improved by 40%. Therefore, for the following NN and MNN, motifs from all clusters are used as training, validation and test dataset blocks (but without using a motif twice.)

The use of a new NN for each cluster was also evaluated but resulted in poor results because of sparse training data. With a much larger dataset, this could change, but for SMEs, this is often infeasible, as they either would need months' worth of data records or multiple identical processes running in parallel.

Table 3 shows the impact of using a separate network for each aspect as shown in Fig. 6. On average, it is better to employ single output nets than to make a network learn to predict multiple outputs correctly. It is also more time consuming since the multiple input multiple output network have many weights to train. It is interesting to see that S2 (the TS that seems to be the most difficult to learn) is predicted better for a network that also predicts the second reels other measured value as an additional output. The reason could be that the behaviour of S2 is chaotic but has some correlation to T1 that is more predictable.

The investigated alternative to an MNN are a two MIMO and one MISO NN (as shown in Fig. 7). These are named Black Box A, B and C. There are several alternatives, as it has also been examined whether the

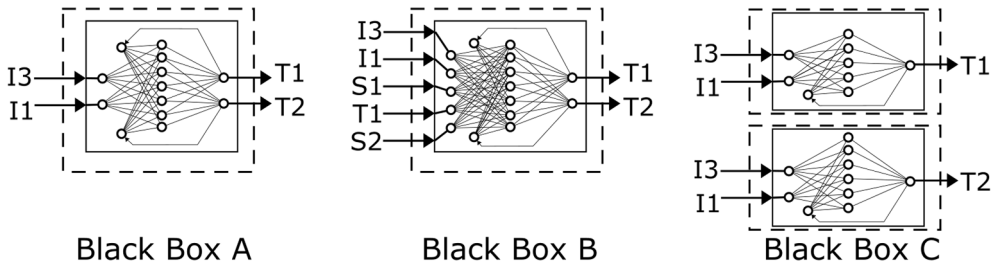

Fig. 7. The Modular Neural Network (Fig. 6) is tested against variants of a Black Box Model NN. These variants are schematically depicted here. Black Box A is a MIMO NN that uses only the controllable aspects as inputs. Black Box B is also a MINO NN but uses all non-output aspects as input, even if they cannot be controlled. Black Box C is a MISO NN. It is similar to Black Box A but uses parallel subnetworks in order to have only one network per objective aspect.

Table 4
Training one big network with a flat structure vs training a modular network with a structure like the plant topology. NN configuration in line with Table 2.

		T1	T2	Improvement compared to Benchmark (see Table 2)
Modular network from single output networks according to the plant topology [Grey-Box, see Fig. 6] Network configuration same as "One network per Output + Use all clusters as blocks" but with connected networks	R	0.950	0.914	1.71%
	MSE	0.00700	0.00241	37.41%
	Calculation time in sec	158.5		−1.777%
MIMO NN with the controllable aspects as inputs and the objective aspects as output [Black Box A, see Fig. 7] trainbr n = 35; d1 = 1:20; d2 = 1:2	R	0.936	0.872	−1.329%
	MSE	0.00399		27.21%
	Calculation time in sec	118.9		23.65%
MIMO NN with all measured aspects as input and the target aspects as output [Black Box B, see Fig. 7] trainbr n = 35; d1 = 1:20; d2 = 1:10	R	0.988	0.935	4.94%
	MSE	0.00318		42.02%
	Calculation time in sec	2588.8		−1562.3%
MISO NN with the controllable aspects as input [Black Box C, see Fig. 7] trainbr n = 35; d1 = 1:20; d2 = 1 trainlm n = 35; d1 = 1:20; d2 = 1:2	R	0.963	0.718	−8.14%
	MSE	0.00508	0.0139	−117.3%
	Calculation time in sec	154.4	322.2	−206.04%

use of a multi output networks (Black Box A, B) over a multi-output network (the proposed MNN but also Black Box C) is again beneficial. In addition, the influence on using only the controllable as input (Black Box A, C) versus using all measured aspects (except the objective aspects) as inputs (Black Box B) was investigated. Using these aspects as inputs too is arguably feasible in a subsequent evolutionary optimization (methodology step 3) by excluding those, which are not controllable from the gene structure, and setting those with current measured values.

As depicted in Table 4, the usage of an MNN created by connecting individually trained smaller subnets according to the plant's topology brings some improvement in accuracy. The MSE and R values are better than those of a Black Box NN using the same input aspects to predict all objective aspects in one (Black Box A) or separate NNs (Black Box C) but worse than a NN that uses all controllable and measured aspects (Black Box B).

Because the MNN uses subnetwork predictions as input in later subnets, the subnetwork prediction error is also propagated. That is why the MNN performs worse than individually trained MISO NN from Table 2 or Black Box B. Nevertheless, in online production use of a NN the intermediate process aspects are not known beforehand which is why the MNN and Black Box A & C are the only NNs which are suited for this use case. Block Box B or single output NNs that predict objective values from intermediate aspects can be used in production, but they need all input as well as intermediate aspects to predict the next time-steps target aspects. NNs which use intermediate aspects as well are therefore restricted to use cases that work with historical data or production environment that have intermediate process aspects instantly available.

It needs to be mentioned that the results are not completely comparable because through the concatenation of NARX networks with their feedback inputs there are more timesteps needed before the MNN produces an output than the Black Box model even if it uses more feedback input neurons. This longer response time shortens the test period used to evaluate the accuracy.

4.3. Optimization with evolutionary algorithms

To utilize the trained representation for an optimization task the evolutionary approach, as shown in Fig. 8, is intended to be used.

An optimization is often use case depended. This means even if data is available and the plant topology is known - which is rare with public datasets - it still needs to be determined what the optimization goal (in other words the objective function) and the boundary conditions are. As this is unknown for this use case and could not be found for any other real-world industrial dataset, the authors resorted to a synthetic optimization scenario to showcase the third methodology step.

The optimization goal in this use case is to find an input data pattern which results in an smooth web tension (T1, T2) that is kept close to 1, while keeping the inputs in the ranges that they demonstrated in the

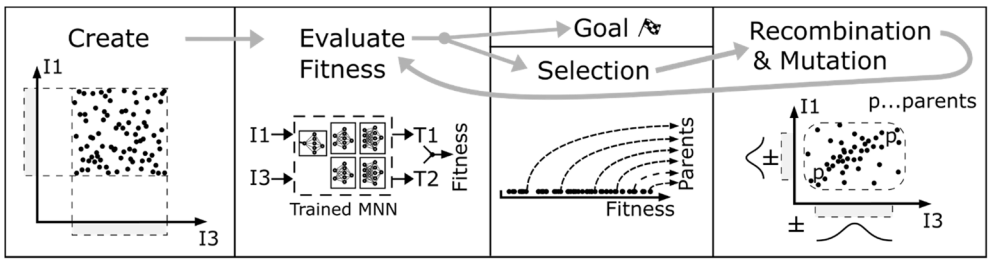

Fig. 8. Basic steps of optimizing the eligible inputs for a desired combination of T1 and T2 through evolutionary algorithms. First, an initial population of feasible solutions (e.g. value tuples of I1 and I3) is created at random. After that each solutions fitness (e.g. KPI consisting of T1 and T2) is evaluated by querying the trained MNN, that represents the system behaviour, with the tuples. In the following Selection step, solutions are chosen as parents of a next generation, with fitter solutions have a better chance of selection. To form next generation selected parent solutions are recombined and mutated at random to form offspring solutions that replace part of the old generation. This is looped until a stopping criterion is met.

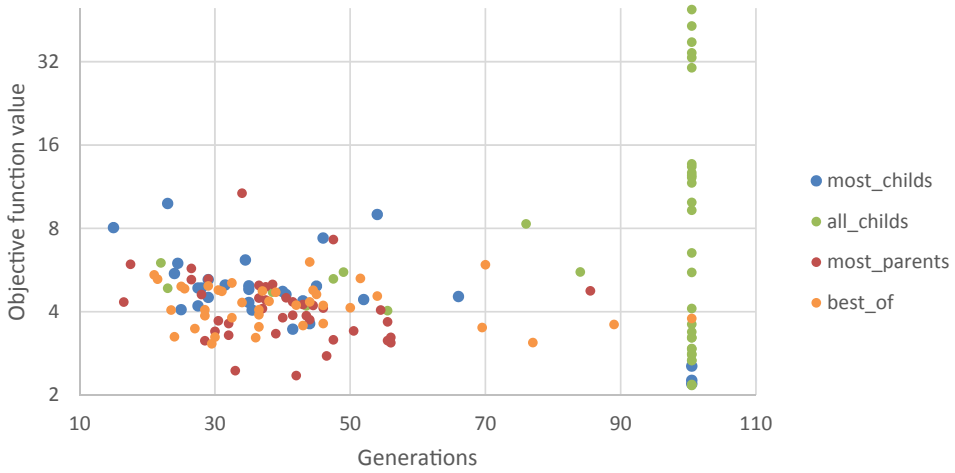

Fig. 9. Resulting objective values and the generations needed for each optimization. Population composition, selection rules and fitness mappings were varied systematically. Here only the result aspect population composition is annotated. Most_childs is 80%, all_childs 100%, most_parents 20% children in each new population. Best_of means that both parents and children compete in the getting into the next population.

Table 5
Parameters and operator choices for the optimization problem of the use case.

evolutionary algorithm aspect	Parameter and operator choices for the evolutionary algorithm
Variables with defined value range limit and boundary conditions	The variables are the input variables of the MNN and their timesteps. The number of timesteps should be at least to required timesteps of the MNN to fill all recurrent nodes and produce a valid output. In this case 60 timesteps were determined.
An objective function that can be calculated using the variables	The objective function is the sum of the squared difference of the web tensions to the desired level of one. The objective function is minimal if the web tensions are both close to one and increase with the power of two if it deviates from it.
An optimization goal or termination criterion	The optimization goal is a minimization of the objective function and the termination criterion is a maximum number of generations of 200 or a solution convergence. Here a solution is converged if it does not change over 15 generations.
The choice of an optimization algorithm and its execution parameters	Repetition: An optimization consisted of 5 full runs. Population size 200 Individuals Crossover-Operator: Extended Line Recombination with an extension of 10% Crossover-Probability 70% Mutation-Operator: uncorrelated Mutation with a generic and dimension specific mutation rate of 0.1 with a mutation rate damping of 1% per generation Mutation-Probability for individuals 30% but only 10% of the genes from the selected individuals get mutated Selection rules and population compositions were set as described in chapter 3.3.

training dataset. The latter aspect is ensured through an adequate gene value range and the first is achieved by minimizing the following objective function.

$$objective_{minimization} = \sum_{1}^{n=60\ timesteps} \left(1 - T_{1,n}\right)^2 + \left(1 - T_{2,n}\right)^2 \quad (1)$$

In this case the parts of the objective function are equally weighted with a factor of 1. If a pareto optimum should be placed towards a variable, then its weight has to be adjusted.

In a real optimization scenario, there would likely be further boundary conditions to the input data, for example a restriction of a maximum change rate for certain variables. But as this is not known here it is assumed that the production line's inertance is low enough to allow all sudden changes within the boundaries.

The basis for the implementation of the evolutionary strategies is "Yarpiz Evolutionary Algorithms Toolbox (YPEA)" by Heris [40] under MIT license. This is a MATLAB framework for the definition and solution of optimization problems with evolutionary algorithms and heuristics. The genetic algorithm of this framework was adapted by the authors to use continuously real-valued gene values with a coding space from 0 to 1 instead of binary coded genes. With this adaption the genetic algorithm was reprogrammed to an evolutionary strategy.

To define an optimization problem in YPEA the following aspects need to be defined:

These values are either taken from the defaults that were set in YPEA or found in literature [63]. The choice of operators, selection rules and population compositions were tried out in a board empirical study with this use case to find out with settings provided robustly good results. The details of this comparison study are beyond the scope of this paper but as an excerpt Fig. 9 shows the resulting objective values and the required generations over the variation of the population composition. The empirical study resulted in the recommendation for default settings for the application of the methodology which are used here and displayed in Table 5.

As the optimization goal is synthetic the performance in this use case is not representative for the methodology. Nonetheless the characteristic of the optimization problem is realistic although further boundary conditions are likely required in a real optimization use case. The resulting input pattern scored on average an objective function value of 4.05 while random guessing averaged on 54. This cannot be benchmarked against general numeric optimization methods as the MNN cannot directly be differentiated and is therefore not suited for most numeric optimizations as already discussed in Section 3.3.

To transform this optimization into an online optimization YPEA was adapted even further. Instead of initializing the population randomly distributed over the whole value range of input values it gets initialized with a gaussian distribution around the currently measured system state. Although the population starts from the current system state, measures have to be taken to keep the first few timesteps of the optimized solution close to this state, if the system has restriction on maximum change rates. This is achieved by expanding the objective function with a penalty term, which penalizes the objective function value if the difference from the current actual system state to the first timesteps of the optimal input data pattern is too large.

For the use case this was implemented and tested with the available training data as an initialization state, but no real measured system state was available.

5. Conclusion and outlook

This this work a ML methodology was introduced which combines a clustering of TS, representing them with MNNs and using this representation for optimization purposes like parameter tuning, in order to gain more benefits combined than the algorithms alone

From the proof of concept with a real-world production dataset, it is evident that finding reoccurring motifs in TS data first and using them in the training of a NN for regression leads to advantages in accuracy. Furthermore, utilizing the plant topology to structure the NNs into an MNN was a beneficial approach in this use case. Having a subnetwork for each production asset did ease the individual network's learning task. Though asset cross correlations might be lost, this MNN approach helps to build networks that resemble production assets closely.

A spreadsheet interface was implemented and provided to easily set-up multiple runs of a NN training in MATLAB, including interconnecting the NNs inputs and outputs to form a MNN. Through this interface it is possible to run a grid search for a NN training in order to find optimal network parameters. In preliminary work the authors investigated the integration of design of experiments approaches into the grid search but found no conclusive result which would significantly reduce the search effort.

Although the proof of concept with industrial data showed an improvement in accuracy for all objectives, more investigation and benchmarks need to be conducted to assess, if the identified advantages can be confirmed in various real-world situations. Especially the use of a plant topology in networks in this manner needs to be proven and further extended for more complex topologies.

In terms of future work, the learned representation could also be achieved with LSTM NNs [43,55]and their deeper extensions [48]which both showed good performance in literature even without the constraint of fixed subsequence lengths. They are able to find longer-term patterns and decide for themselves, which subsequences influence the long-term memory. This advantage means less effort in feature creation (in this case motifs search) and expert process knowledge is necessary, which according to Gamboa [33] makes them more universal. However, the trade-off to the NNs of this paper would be that they are harder to train and require a certain level of ML expertise. Applying advanced NNs would mean a more Black-Box-like modelling behaviour. This would also impede a steering of the learning towards the previously identified meaningful machine states (interpreted motifs) and plant topologies. To verify that it is indeed an acceptable trade-off to a possibly more accurate TS representation the authors aim to compare accuracy and amount of parameter tuning for a LSTM NN to the application of a NARX in future work.

Regarding the data exploration, there are many algorithms available for feature extraction, clustering as well as TS representation. Therefore, the authors intend to perform a comparison of different algorithm combinations over a large database with sample datasets from different applications. An early summary of the comparison was already published in [38]. Again, the focus there was on parameter-light implementations, like in Yeh, Zhu et al. [84]. The data exploration algorithms used the early preliminary outcome of this comparison study. The aim of further investigations is to find a combination of feature extraction and clustering algorithms that discover clusters in the data accurately and within a feasible amount of calculation effort. Further research is planned on the transformation of the data into an aggregated TS representation to speed up the calculation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors would like to acknowledge and express their sincere gratitude to the Austrian Research Promotion Agency (FFG) through grant project number 865898 and the Institute of Engineering Design and Product Development, Department Mechanical Engineering Informatics and Virtual Product Development for granting the research activities as part for a pre-doc position.

References

[1] B. Alkan, et al., Complexity in manufacturing systems and its measures: a literature review, Eur. J. Ind. Eng. 12 (1) (2018) 116–150.
[2] M. Amer, T. Maul, A review of modularization techniques in artificial neural networks, Artif. Intell. Rev. 52 (1) (2019) 527–561.
[3] R. Anand, et al., Efficient classification for multiclass problems using modular neural networks, IEEE Trans. Neural Networks 6 (1) (1995) 117–124.
[4] M. Ankerst, et al., OPTICS: ordering points to identify the clustering structure. ACM Sigmod record, ACM, 1999.
[5] G. Auda, M. Kamel, Modular neural networks: a survey, Int. J. Neural Syst. 9 (02) (1999) 129–151.
[6] T. Bäck, F. Hoffmeister, Extended selection mechanisms in genetic algorithms, 1991.
[7] T. Bäck, H.-P. Schwefel, An overview of evolutionary algorithms for parameter optimization, Evol. Comput. 1 (1) (1993) 1–23.
[8] P.P. Balestrassi, et al., Design of experiments on neural network's training for nonlinear time series forecasting, Neurocomputing 72 (4–6) (2009) 1160–1178.
[9] I.A. Basheer, M. Hajmeer, Artificial neural networks: fundamentals, computing, design, and application, J. Microbiol. Methods 43 (1) (2000) 3–31.
[10] T. Bastogne, Identification des systèmes multivariables par les méthodes des sous-espaces. Application à un système d'entraînement de bande, Nancy, 1997.
[11] T. Bastogne, et al., Application of subspace methods to the identification of a winding process. 1997 European Control Conference (ECC), IEEE, 1997.
[12] D.J. Berndt, J. Clifford, Using Dynamic Time Warping to Find Patterns in Time Series. KDD workshop, Seattle, WA, 1994.
[13] A. Bey-Temsamani, et al., A practical approach to combine data mining and prognostics for improved predictive maintenance, Data Min. Case Stud. 36 (2009).
[14] A. Blum, R.L. Rivest, Training a 3-node neural network is NP-complete, Adv. Neural Inf. Process. Syst. (1989).
[15] M. Brettel, et al., How virtualization, decentralization and network building change the manufacturing landscape: an Industry 4.0 Perspective, Int. J. Mech. Ind. Sci. Eng. 8 (1) (2014) 37–44.
[16] T. Caliński, J. Harabasz, A dendrite method for cluster analysis, Commun. Stat.-Theory Methods 3 (1) (1974) 1–27.
[17] Y. Carson, A. Maria, Simulation optimization: methods and applications. Proceedings of the 29th conference on Winter simulation, 1997.
[18] K. Chakraborty, et al., Forecasting the behavior of multivariate time-series using neural networks, Neural Netw. 5 (6) (1992) 961–970.
[19] B.T. Chen, et al., Smart factory of industry 4.0: key technologies, application case, and challenges, IEEE Access 6 (2018) 6505–6519.
[20] C.S. Cheng, H.P. Cheng, Identifying the source of variance shifts in the multivariate process using neural networks and support vector machines, Expert Syst. Appl. 35 (1–2) (2008) 198–206.
[21] R. Cheng, M. Gen, Genetic Algorithms and Engineering Design, John Wiley, 1997.
[22] B. Chiu, et al., Probabilistic discovery of time series motifs. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003.
[23] B. Dasgupta, et al., On the complexity of training neural networks with continuous activation functions, IEEE Trans. Neural Netw. 6 (6) (1995) 1490–1504.
[24] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. 1 (2) (1979) 224–227.
[25] B.L.R. De Moor, DaISy: Database for the Identification of Systems. [Used dataset: Data from a test setup of an industrial winding process, section Process Industry Systems,97-003], 2019. Retrieved 8.8.2019, from http://homes.esat.kuleuven.be/~smc/daisy/.
[26] E. Diaconescu, The use of NARX neural networks to predict chaotic time series, Wseas Trans. Comput. Res. 3 (3) (2008) 182–191.
[27] U. Dombrowski, et al., Concept for a cyber physical assembly system, in: Assembly and Manufacturing (ISAM), 2013 IEEE International Symposium on, IEEE, 2013.
[28] E. Elbeltagi, et al., Comparison among five evolutionary-based optimization algorithms, Adv. Eng. Inf. 19 (1) (2005) 43–53.
[29] R. Erkki, P. Johnsson, Quality Data Management in the Next Industrial Revolution: A Study of Prerequisites for Industry 4.0 at GKN Aerospace Sweden, 2018.
[30] P. Esling, C. Agon, Time-series data mining, ACM Comput. Surv. (CSUR) 45 (1) (2012) 12.
[31] R.J. Frank, et al., Time series prediction and neural networks, J. Intell. Rob. Syst. 31 (1–3) (2001) 91–103.
[32] M.C. Fu, Optimization via simulation: a review, Ann. Oper. Res. 53 (1) (1994) 199–247.

[33] J.C.B. Gamboa, Deep learning for time-series analysis. arXiv preprint arXiv: 1701.01887, 2017.
[34] D. Gerhard, Product lifecycle management challenges of CPPS, in: S. Biffl, A. Lüder (Eds.), Multi-Disciplinary Engineering for Cyber-Physical Production Systems: Data Models and Software Solutions for Handling Complex Engineering Projects, Springer, 2017, pp. 89–113.
[35] D.E. Goldberg, et al., Messy genetic algorithms: Motivation, analysis, and first results, Complex Syst. 3 (5) (1989) 493–530.
[36] J. Gordon, S.A. Sohal, Assessing manufacturing plant competitiveness-An empirical field study, Int. J. Oper. Prod. Manage. 21 (1/2) (2001) 233–253.
[37] J. Han, et al., Data Mining: Concepts and Techniques, Elsevier, 2011.
[38] M. Hennig, et al., Comparison of time series clustering algorithms for machine state detection. 53rd CIRP Conference on Manufacturing Systems, 2020.
[39] M. Hennig, M. Schreiner, Create MATLAB neural networks from excel. xls_to_MATLAB_NN, 2020. Retrieved 21. april 2020, 2020, from https://github.com/mhennig-TUW/xls_to_MATLAB_NN.
[40] S.M.K. Heris, Yarpiz Evolutionary Algorithms Toolbox. YPEA. Retrieved 27, 2019. April 2019, 2019, from https://www.github.com/smkalami/ypea.
[41] M. Hermann, et al., Design principles for industrie 4.0 scenarios, in: 2016 49th Hawaii international conference on system sciences (HICSS), 2016, IEEE.
[42] Y.-C. Ho, An explanation of ordinal optimization: Soft computing for hard problems, Inform. Sci. 113 (3–4) (1999) 169–192.
[43] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[44] R. Hofmann, et al., Comparison of a physical and a data-driven model of a Packed Bed Regenerator for industrial applications, J. Storage Mater. 23 (2019) 558–578.
[45] T. Idé, Why does subsequence time-series clustering produce sine waves?. European Conference on Principles of Data Mining and Knowledge Discovery Springer, 2006.
[46] H. Jalali, I.V. Nieuwenhuyse, Simulation optimization in inventory replenishment: a classification, IIE Trans. 47 (11) (2015) 1217–1235.
[47] H. Kagermann, et al., Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group, Forschungsunion, 2013.
[48] F. Karim, et al., LSTM fully convolutional networks for time series classification, IEEE Access 6 (2018) 1662–1669.
[49] E. Keogh, et al., Clustering of time-series subsequences is meaningless: implications for previous and future research, in: ICDM 2003. Third IEEE International Conference on Data Mining, IEEE, 2003.
[50] E. Keogh, et al., Towards parameter-free data mining, in: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, Seattle, WA, USA, 2004, pp. 206–215.
[51] W. Laosiritaworn, N. Chotchaithanakorn, Artificial neural networks parameters optimization with design of experiments: an application in ferromagnetic materials modeling, Chiang Mai J. Sci. 36 (1) (2009) 83–91.
[52] T. Lin, et al., Learning long-term dependencies in NARX recurrent neural networks, IEEE Trans. Neural Netw. 7 (6) (1996) 1329–1338.
[53] F. Liu, et al., Use clustering to improve neural network in financial time series prediction, in: Third International Conference on Natural Computation (ICNC 2007), IEEE, 2007.
[54] S. Lloyd, Least squares quantization in PCM, IEEE Trans. Inf. Theory 28 (2) (1982) 129–137.
[55] P. Malhotra, et al., Long short term memory networks for anomaly detection in time series. Proceedings, Presses universitaires de Louvain, 2015.
[56] M.R. Meireles, et al., A comprehensive review for industrial applicability of artificial neural networks, IEEE Trans. Ind. Electron. 50 (3) (2003) 585–601.
[57] J.M.P. Menezes, G.A. Barreto, Long-term time series prediction with the NARX network: an empirical evaluation, Neurocomputing 71 (16–18) (2008) 3335–3343.
[58] L. Monostori, Cyber-physical production systems: Roots, expectations and R&D challenges, Procedia CIRP 17 (2014) 9–13.
[59] D.C. Montgomery, Design and Analysis of Experiments, John Wiley & Sons, 2017.
[60] A. Mueen, et al., Exact Discovery of Time Series Motifs, (2009) 473–484.
[61] R.d. Nardi, et al., Evolution of neural networks for helicopter control: why modularity matters, in: 2006 IEEE International Conference on Evolutionary Computation, 2006.
[62] H.D. Nguyen, A data-driven framework for remaining useful life estimation, Vietnam J. Sci. Technol. 55 (5) (2017) 557.
[63] V. Nissen, Einführung in evolutionäre Algorithmen: Optimierung nach dem Vorbild der Evolution, Springer-Verlag, 2013.
[64] V. Oduguwa, et al., Evolutionary computing in manufacturing industry: an overview of recent applications, Appl. Soft Comput. 5 (3) (2005) 281–299.
[65] H. Pierreval, L. Tautou, Using evolutionary algorithms and simulation for the optimization of manufacturing systems, IIE Trans. 29 (3) (1997) 181–189.
[66] H. Pohlheim, Genetic and Evolutionary Algorithm Toolbox for use with MATLAB. Dept. Comput. Sci., Univ. Ilmenau, Ilmenau, Germany, 1998. Retrieved from http://www.geatbx.com/download/GEATbx_Tutorial_v33c.pdf.
[67] I. Rechenberg, Evolutionsstrategien. Simulationsmethoden in der Medizin und Biologie, Springer, 1978, pp. 83–114.
[68] P.J. Rousseeuw, Silhouettes - a graphical aid to the interpretation and validation of cluster-analysis, J. Comput. Appl. Math. 20 (1987) 53–65.
[69] A.L. Samuel, Some studies in machine learning using the game of checkers, IBM J. Res. Dev. 3 (3) (1959) 210–229.
[70] H. Sawai, et al., Parallelism, hierarchy, scaling in time-delay neural networks for spotting Japanese phonemes/CV-syllables, in: Proc. Intl. Joint Conf. on Neural Networks, 1989.
[71] H.-P. Schwefel, Numerical Optimization of Computer Models, John Wiley & Sons Inc, 1981.
[72] R. Setiono, Feedforward neural network construction using cross validation, Neural Comput. 13 (12) (2001) 2865–2877.
[73] R.R. Sokal, C. Michener, A statistical method for evaluating systematic relationships, Univ. Kansas, Sci. Bull. 38 (1958) 1409–1438.
[74] B. Steiner, Optimierungsverfahren zum Entwurf von Neuronalen Netzen. Institute of Engineering Design and Product Development. Research Group Mechanical Engineering Informatics and Virtual Product Development, TU Wien. B.Sc, 2020.
[75] W. Sukthomya, J. Tannock, The optimisation of neural network parameters using Taguchi's design of experiments approach: an application in manufacturing process modelling, Neural Comput. Appl. 14 (4) (2005) 337–344.
[76] A.J. Thomas, et al., Identifying the UK's manufacturing challenges as a benchmark for future growth, J. Manuf. Technol. Manage. 23 (2) (2012) 142–156.
[77] J.T. Tsai, et al., Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm, IEEE Trans. Neural Netw. 17 (1) (2006) 69–80.
[78] X. Wang, et al., Experimental comparison of representation methods and distance measures for time series data, Data Min. Knowledge Discov. 26 (2) (2012) 275–309.
[79] T. Warren Liao, Clustering of time series data—a survey, Pattern Recogn. 38 (11) (2005) 1857–1874.
[80] R. Wirth, J. Hipp, CRISP-DM: Towards a standard process model for data mining. Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining, Citeseer, 2000.
[81] S.-J. Wu, et al., A neural network integrated decision support system for condition-based optimal predictive maintenance policy, IEEE Trans. Syst. Man Cybernet.-Part A: Syst. Hum. 37 (2) (2007) 226–236.
[82] T. Wuest, et al., An approach to monitoring quality in manufacturing using supervised machine learning on product state data, J. Intell. Manuf. 25 (5) (2014) 1167–1180.
[83] L.D. Xu, et al., Industry 4.0: state of the art and future trends, Int. J. Prod. Res. 56 (8) (2018) 2941–2962.
[84] C.-C. M. Yeh, et al., Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View that Includes Motifs, Discords and Shapelets. IEEE ICDM, 2016.
[85] S.Y. Yim, et al., Using process topology in plant-wide control loop performance assessment, Comput. Chem. Eng. 31 (2) (2006) 86–99.