Machine Learning for Cyber-Physical Systems

Oliver Niggemann, Bernd Zimmering, Henrik Steude,
Jan Lukas Augustin, Alexander Windmann and Samim Multaheb

Abstract

Machine Learning plays a crucial role for many innovations for Cyber-Physical Systems such as production systems. On the one hand, this is due to the availability of more and more data in ever better quality. On the other hand, the demands on the systems are also increasing: Production systems have to support more and more product variants, saving resources is increasingly in focus and international competition is forcing companies to innovate faster. Machine Learning leverages data to solve these issues. The goal is to have self-learning systems which improve over time. There are various algorithms and methods for this, for which an overview is given here. Furthermore, this article discusses special requirements of Cyber-Physical Systems for Machine Learning processes.

O. Niggemann (✉) · B. Zimmering · H. Steude · J. L. Augustin · A. Windmann · S. Multaheb
Institute of Automation Technology, Helmut Schmidt University, Hamburg, Germany
e-mail: oliver.niggemann@hsu-hh.de
URL: http://www.hsu-hh.de/imb

B. Zimmering
e-mail: bernd.zimmering@hsu-hh.de

H. Steude
e-mail: henrik.steude@hsu-hh.de

J. L. Augustin
e-mail: lukas.augustin@hsu-hh.de

A. Windmann
e-mail: alexander.windmann@hsu-hh.de

S. Multaheb
e-mail: samim.multaheb@hsu-hh.de

B. Vogel-Heuser and M. Wimmer (eds.), *Digital Transformation*,
https://doi.org/10.1007/978-3-662-65004-2_17

Keywords
Machine Learning • Cyber-Physical Systems

1 Introduction

Cyber-Physical Systems (CPS) are becoming a major field for Machine Learning (ML). Challenges go far beyond a mere application of existing algorithms. What is needed are specialized algorithms which meet domain requirements such as reliability, real-time capability and maintainability. This paper gives an introduction to the opportunities and challenges of ML for CPS.

For this, potential CPS application scenarios for ML are described in Sect. 6. Sect. 3 outlines the state of the art for ML and maps features of algorithms to the uses cases from Sect. 6. From this, Sect. 4 derives specific requirements of CPS to ML. The next sections describe these requirements and the corresponding solution approaches in detail. Sect. 9 summarizes the paper content.

2 Application Scenarios

Currently ML is applied to several industrial use cases:

2.1 Condition Monitoring and Predictive Maintenance

For every plant operator, it is desirable that certain components are replaced at exactly the right time, not too early out of caution, but also not so late that the risk of failure becomes significant. The method of choice is predictive maintenance. It is made possible by condition monitoring, i.e. the continuous monitoring of the system. Nowadays, condition monitoring is based on data and observations [77, 83].

The basic idea is that various component data, such as vibration, speed and energy consumption, are continuously collected and evaluated [95]. In many cases, such data are already available anyway, but remain unused. It is only necessary in exceptional cases to install new sensors in order to generate additional data.

For condition monitoring, ML is mainly used to learn a model of the normal system behavior, including thresholds to non-normal behavior. Once this threshold is crossed, a warning is given. Predictive maintenance requires that these models can be extrapolated in time, e.g. these models predict when a threshold will be crossed.

2.2 Resource Optimization

The consumption of resources in production is becoming more and more important for companies. At the same time, attention is increasingly being paid to saving wastewater and emissions. On the one hand, these goals have financial reasons, on the other hand, increasing environmental awareness and corresponding legislation are also having an effect.

There are two variants for implementing resource optimization [99, 126]: The simple one is limited to the fact that software stores and analyzes the consumption data in detail. The employees can derive change options from this and implement them. The much more complex variant: The software not only makes the consumption data available, but also independently optimizes the control of the systems.

Here a prediction model of the resource consumption is learned. In the end, these models must have a sufficient quality to be integrated into closed-loop control loops—a very demanding requirement.

2.3 Quality Assurance of Products

ML can be used to monitor the quality of a product during manufacturing and to detect irregularities at an early stage [131].

ML systems can generate and maintain a Digital Twin [94], i.e. a virtual image of the real products and intermediate products. These Digital Twins collect all information from the engineering phase and can be used to improve the learning process by providing a-priori information [116]. Digital Twins are then enriched during the operation phase by evaluating sensor data using Artificial Intelligence (AI) and ML methods, combined with information about raw materials and production processes. This Digital Twin allows the prediction of product properties that are difficult or impossible to determine in reality. Thus, virtual measurements that are difficult to implement in reality can be carried out on a Digital Twin. Although these predictions are often less certain than real measurements, they allow an early warning in the event of quality problems.

Such Digital Twins can be used in many areas in which complex end-of-line tests or laboratory tests are currently commonly used, for example after the end of production to analyze the properties of food.

2.4 Diagnosis

AI or ML can help to identify the causes of errors in a system [15, 31, 83]. If many sensors are built into a system, as it is increasingly common today, problems are detected early on by condition monitoring and anomaly detection systems and reported as an alarm. However, the connection between a symptom and the cause of the error is often difficult to determine—this is due to the increasing complexity of modern (production) systems: The systems are getting bigger and bigger, consist of many sub-modules and are characterized by an increasingly high degree of networking and automation. An error often causes subsequent errors early in the production process and only leads to symptoms and alarms much later—a cause of error can propagate through the entire system and lead to symptoms in a wide variety of places, sometimes with a significant delay. The more complex and networked the system, the longer it takes to manually identify the cause. This means that it can take a long time before repairs can be carried out. Today this is seen as an important cost driver.

An AI/ML-based diagnostic system determines the most likely causes of failure based on the symptoms and does so within a short period of time. The user then no longer only sees the symptoms in the form of alarms and warnings—rather, possible causes of errors are displayed immediately, and repair instructions are often supplied directly.

In this use case, mainly two kinds of models are used: First a model of the normal behavior is learned and used to compute symptoms, i.e. warnings. Then a model comprising system causalities is used to identify root causes [96]. The latter is only partially learned.

3 Machine Learning

The number of ML algorithms is legion—and so are the taxonomies used to describe them [10]. Here, we will introduce two dimensions of algorithms' feature to describe algorithm: First, we will use recurrency, i.e. the ability of algorithms to handle dynamic, time-variant data. Second, we will use supervision, i.e. the degree of supervision needed by the algorithm.

Normally, ML algorithms compute a model. Just like manually created models, models can be used to predict specific system features. We start with describing the dimension "recurrency".

Static Analysis For tasks such as condition-monitoring or anomaly detection, only the signal values $\mathbf{x}_t \in \mathbf{R}^n$ at some point in time t are used. This is shown in Fig. 1 on the left hand side. In other words, for the analysis a static feature vector [35] is taken into account. Thereby, the assumption is that no information is coded in the sequence of values and all necessary information is contained in the current signal values. This assumption is true for many CPS, even for systems which have a dynamic nature. For a new data point $\mathbf{x} \in \mathbf{R}^n$, its probability $p(\mathbf{x}|X)$ given some historical data X is (at least approximately) computed. If the data is improbable, an anomaly has been identified.

Dynamic Analysis A totally different situation arises when important information is coded in the sequence of signal values over time: For a time window of data $X = \{\mathbf{x}_{t-k}, \dots, \mathbf{x}_t\}$, its probability is computed. This is shown in Fig. 1 in the center. Again, if the time window is improbable, an anomaly has been identified.

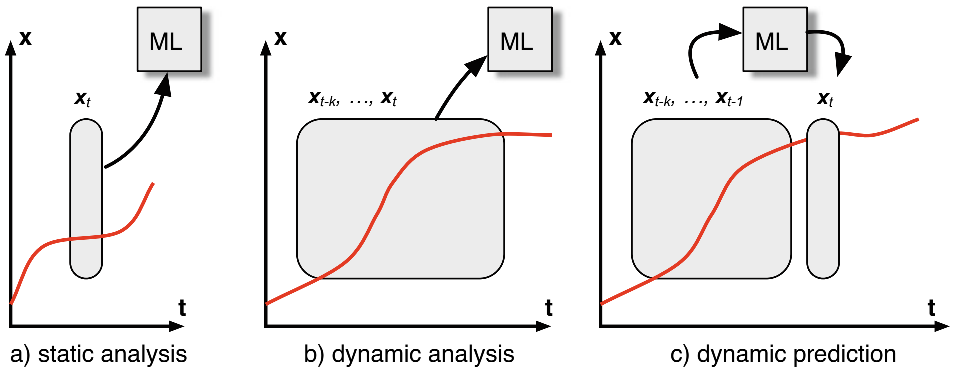


Fig. 1 Comparison of static ML (**a**), dynamic ML (**b**) and the special case of (**c**) ML by means of prediction. Signal values **x** are depicted over time

Often, especially for dynamic analyses, prediction is used to analyze the data: Let X be again a time window of data, then a prediction for the next value $\hat{\mathbf{x}}_{t+1}$ is computed. Once a real measurement for $\mathbf{x}_{t+1}$ is available, it can be compared to the prediction $\hat{\mathbf{x}}_{t+1}$—if they are significantly different, an anomalous situation has occurred. This is shown in Fig. 1 on the right hand side.

Next, the dimension supervision is described.

Supervised Machine Learning Supervised ML algorithms work on labeled data, this is shown in Fig. 2: Each data point $\mathbf{x}$ comes with a label $\mathbf{y}$. The ML algorithm will learn a model which is able to compute suitable labels for a given input $\mathbf{x}$. If $\mathbf{y}$ is nominal, e.g. its values are discrete classes such as "OK" or "KO", this task is called classification. If $\mathbf{y}$ is cardinal, e.g. its values are numerical values such as temperatures, this task is also called regression.

Often, these ML models are trained by using a feedback or residual signal. The algorithms start with an initial, suboptimal model and compute a prediction of $\hat{\mathbf{y}}$. The matching or fitting between the estimation $\hat{\mathbf{y}}$ and the real, wanted label $\mathbf{y}$ is assessed and provides a feedback which is used to improve the model—often the difference between $\hat{\mathbf{y}}$ and $\mathbf{y}$ is used, i.e. a residual.

Unsupervised Machine Learning Unsupervised ML uses unlabeled data (see also Fig. 3). Since again a feedback or residual signal is needed to learn a model, a generic, external criterion is used to assess the quality of a model prediction e.g. for clustering, i.e. the identification of clusters of similar data such as plant phases ("ramp-up", "normal operation", etc.). A good clustering has a high similarity between data within clusters and a low similarity between clusters.

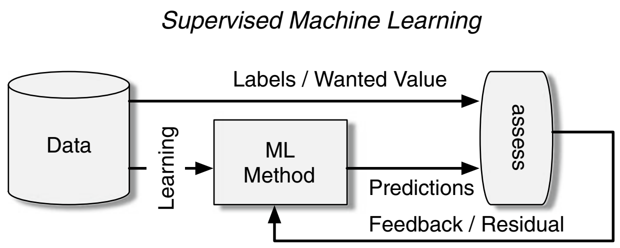


Fig. 2 Principle idea of supervised ML

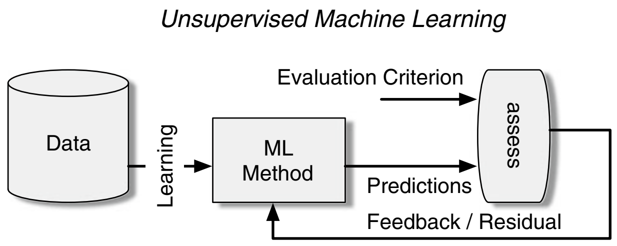


Fig. 3 Principle idea of unsupervised ML

If the external criterion is given dynamically from an environment, e.g. a camera signal for an autonomous vehicle, we speak of reinforcement learning.

We can now use these two dimensions (recurrency, supervision) to describe some common ML algorithms and describe their suitability for the use cases from Sect. 6. This is also visualized in Fig. 4.

Static, Supervised Machine Learning (bottom left quadrant in Figure 4)

Feedforward Neural Networks: Neural networks [91] approximate complex functions by parameterizing a network of simple, generic functions. The architecture of the network, i.e. the so-called topology, and the chosen generic functions, i.e. the so-called neurons, decide about the class of functions which can be approximated. The connections between neurons comprise parameters, i.e. the so-called weights, which are used to fit the neural network to a given data set—normally by means of optimization algorithms.

In most cases the topology of feedforward networks comprises a number of layers where only neighboring layers are connected. The most bottom layer is fed with the input x_i where the top most layer models the labels y_i. The network then learns the mapping from typical inputs to corresponding labels.

Decision Trees and Random Forrests: Decision trees [105] learn a tree of decision rules to map from a data vector x_i to labels y_i. Each decision rule splits the set of data by an inequality on the elements in the vector x_i.

Random forests [12] extend this idea by learning a set of decision trees, decisions are made by a majority vote.

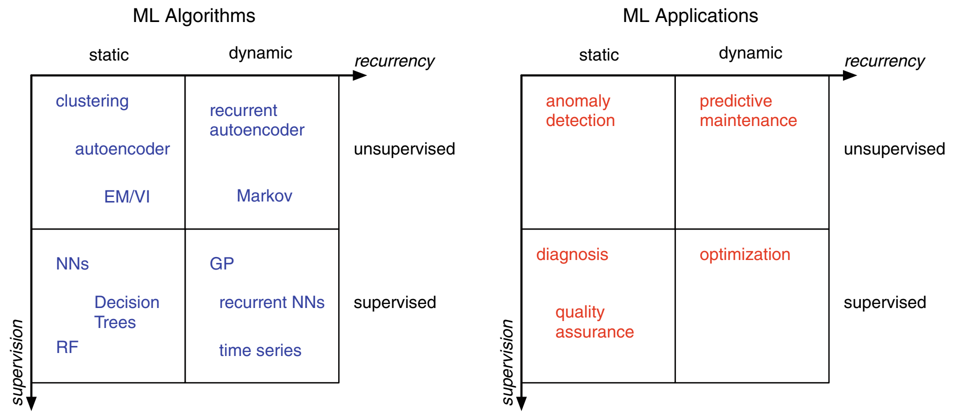


Fig. 4 Mapping of ML features to algorithms and applications

Static, Unsupervised Machine Learning (top left quadrant in Figure 4)

Clustering: Clustering groups given observations $x_i \in X$ in clusters $X_1, \dots, X_p$, $p \in \mathbf{N}$ with $\bigcup_i X_i = X$. The similarity between elements within clusters is maximized while the similarity between elements in different clusters is minimized. Algorithms range from simple shaped clusters (with a given number of clusters p) such as k-means clustering [92] to approaches which are able to identify complex shapes and also identify p, e.g. DBSCAN [107]. The similarity criterion is defined externally. Clustering methods often suffer from the problem that different cluster shapes require different algorithms or algorithm configurations.
Autoencoder: Autoencoders [40] are neural networks which remember learned data vectors x_i. The key idea is that instead of mapping from x_i to y_i, autoencoders map from x_i to x_i. Thus, autoencoders remember already observed situations and can check how similar new observations are to this memory. By checking whether a new data vector is remembered, e.g. an anomaly detection method can be implemented. A variational autoencoder (VAE) extends the idea of the classical autoencoder (AE) with concepts from probabilistic modeling and was originally introduced in [64].
Expectation Maximization (EM) / Variational Inference (VI): EM [91] and VI [114] are algorithms which learn probability distributions from data. For this, the structure of the probability distributions must be given. The learned distribution can be used to compute the probability for new data. This solution of course requires that the underlying distribution is already known.

Dynamic, Supervised Machine Learning (bottom right quadrant in Figure 4)

Time Series: In statistics, time series analysis [91] is a well-established field. Typical solutions such as ARMA [91] learn a given autoregressive function which expresses x_i as a linear functions of x_j, $j < i$ and of stochastic terms. The drawback is the simplicity of the used functions.

Gaussian Processes: Gaussian processes [125] model a time series as a stochastic process. The underlying probability function for a range of time steps is a multivariate normal distribution. Gaussian processes are well-suited to capture uncertainties about the learned model but often suffer from runtime challenges.

Recurrent Neural Networks, Gated Networks, Attention-based Networks: Neural networks can also be used to learn a model of time series. The simplest solution is the use of computed values of neurons at time step t as an input for the next time step $t+1$, i.e. so-called recurrent neural networks [100]. Since such networks have problems using values from several time steps in the past, gated networks such as Long Short Term Memory (LSTM) have been used [54]. Such networks try to generate a memory of important CPS information. Gated networks must learn when to update the memory which leads to corresponding demands on the data quantity and quality. Attention-based networks [122] simplify things by only learning which past data element is relevant.

Dynamic, Unsupervised Machine Learning (top right quadrant in Figure 4)

Recurrent Autoencoder: Recurrent neural networks can also be used in an unsupervised fashion, i.e. as autoencoders [129].

Markov Models: Markov models [51] capture the time series $x_i, i = 1, \ldots, n$ as a path through a given graph. The graph comprises a set of predefined states where transitions model the probabilistic movement from one state to another state. Hidden Markov models [91] assume that the current state is not directly observable.

4 Challenges to ML for Cyber-Physical Systems

ML methods are currently a central component of many research and business activities. Despite many advances, these processes are currently mainly used in non-technical areas and are usually difficult to transfer to technical applications such as production or vehicles [120]. The reason: AI and ML methods were often developed for completely different data, such as economic data. Current ML challenges at the interface between AI / ML and engineering focus on special requirements by technical systems [95, 97].

The results of non-technical ML applications such as business data are usually interpreted, checked and used by a human [117]. The use of ML in a CPS, on the other hand, often means the application in a closed control loop: the results are interpreted by software and then automatically used for optimization. A person is usually no longer involved. This different usage scenario creates challenges [8–10, 83] which distinguish CPS from non-technical ML applications such as business data and image processing—and therefore require adapted ML solutions.

Challenge 1: Time and State: The main characteristic of all physical systems is that their behavior must be considered over time, not just at a specific point in time—therefore, for example, all ML results must also predict system behavior over time. E.g. the behavior of a chemical plant can only be understood if the history of the last hours is known or many problems of transport system arise from incorrect accelerations. The behavior over time thus includes current states, aggregating changes from the past, and information about state changes. Basic requirements for this are common, uniform time models both for the physical and software parts of a production system [74].

Challenge 2: Uncertainty and Noise: In order to use ML procedures and learned models in CPS, it is imperative to evaluate the uncertainty of the predictions of the ML systems [90]. If, for example, an ML system predicts a system failure, the degree of certainty of this forecast is decisive for the correct procedure. Uncertainties mostly arise from noise on the sensor data or from values that cannot be observed.

Challenge 3: Usage of A-Priori Knowledge: There is a lot of prior knowledge for CPS from the design phase, based on physical laws and engineering knowledge. This individual knowledge should be used to improve AI and ML practices. For ML solutions in particular, prior knowledge can alleviate the need for big amounts of data.

Challenge 4: Representations and Concepts: ML results and generated models must be explained to human operators: Why is a maintenance action necessary? Why are new parameters better than old ones? What happens if repairs are not done? For this, symbolic concepts must be learned from the (numerical) models, e.g. the concept "ramp-up phase" for some activations of a neural network. Based on these concepts, explanations and reasons are generated.

In general, it can be said that ML is in principle an interdisciplinary topic between engineering and computer science and must be approached using appropriately adapted methods. In the following, the points from above are discussed in detail.

5 Challenge 1: Time and State

In engineering or more generally speaking physical modeling, time is an essential concept, as systems change their behavior dynamically. Looking at a single point in time is insufficient, as key information about the context would be lost. A common way to encode these temporal dependencies is to introduce the latent state $\mathbf{z}(t) \in \mathbf{R}^m$, which describes the system at a given time t. The state evolves over time depending on new observations and can thus store information about the system, which can then be used for tasks like forecasting, classification or anomaly detection. Examples are automatons, where $\mathbf{z}(t) \in \{m_0, \ldots, m_i\}$ models discrete modes which are switched by events e, or ODEs, where the state $\mathbf{z}$ changes continuously over time, e.g. according to $\dot{\mathbf{z}}(t) = f(\mathbf{z}, t)$.

5.1 Approaches

A time series consists of observations over multiple time steps. In practice, there is no obvious beginning. Rather than working with all observations that are available, often a rolling window of k time steps is applied. Let $\mathbf{x}_t \in \mathbf{R}^n$ denote a sample at t. Then, a window consists of the data $X = \{\mathbf{x}_{t-k}, \ldots, \mathbf{x}_t\}$.

A general approach to describe the state of a system is the state space model [91]:

$$\begin{aligned} \mathbf{z}_t &= g\left(\mathbf{u}_t, \mathbf{z}_{t-1}, \boldsymbol{\epsilon}_t\right) \\ \mathbf{x}_t &= h\left(\mathbf{z}_t, \mathbf{u}_t, \boldsymbol{\delta}_t\right) \end{aligned}$$

where $\mathbf{u}_t \in \mathbf{R}^l$ is an optional input or control signal, g is the transition model, h is the observation model and $\boldsymbol{\epsilon}_t \in \mathbf{R}^p$ and $\boldsymbol{\delta}_t \in \mathbf{R}^q$ describe noise, which is modeled as a random variable. Popular examples include ARIMA models [11, 34] and exponential smoothing [59]. For simple time series, these models work well, are interpretable and data efficient and thus widely used. Unfortunately, state space models often fail to detect complex patterns in time series and have to be tuned to every system seperately, which makes applying them labor-intensive [102].

ML methods are suited to tackle these issues. They can detect complex patterns across multiple time series and require very little engineering by hand [7, 72]. However, they lack interpretability and generally require a lot of data to work well. A model architecture that is built around that idea of learning the state of a time series is the Recurrent Neural Network (RNN). In contrast to state-space models, it does not rely on stochastic variables, but tries to model the sample distribution directly. One of the simplest forms of an RNN can be expressed as Delay Differential Equation (DDE) ([110] gives a detailed introduction). For $i \in \{t-k, \ldots, t\}$, with a randomly initialized $\mathbf{z}_{t-k-1} \in \mathbf{R}^m$, a DDE can be expressed as

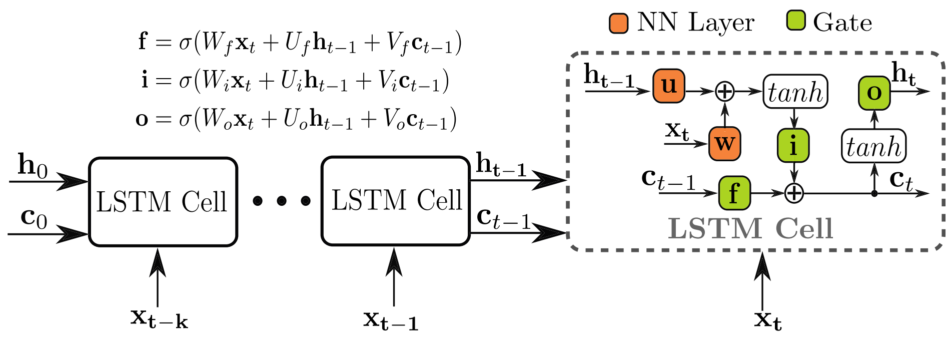
$$\mathbf{z}_i = W_z \mathbf{z}_{i-1} + W_r \mathbf{r}_{i-1} + W_x \mathbf{x}_i + \theta_z, \tag{1}$$

$$\mathbf{r}_i = G(\mathbf{z}_i), \tag{2}$$

where $G(\cdot)$ is a nonlinear function (e.g. tanh) and the matrices $W_z, W_r \in \mathbf{R}^{m\times m}$, $W_x \in \mathbf{R}^{m\times n}$ and the bias $\theta_z \in \mathbf{R}^m$ are trainable parameters of the network. If this RNN is applied to time series data X by shifting it from $t-k$ to t for fixed Δt between observations, the first two terms of Eq. (1) allow the network to propagate information from the past to the current time step t.

5.2 State of the Art

Gating Standard RNNs are hard to train, especially for long time series. RNNs suffer from the vanishing gradient problem [55]: The error gradient, which is needed to train neural



$\mathbf{f} = \sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + V_f\mathbf{c}_{t-1})$
$\mathbf{i} = \sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + V_i\mathbf{c}_{t-1})$
$\mathbf{o} = \sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + V_o\mathbf{c}_{t-1})$
NN Layer
Gate
LSTM Cell
LSTM Cell
LSTM Cell
$\mathbf{h}_0$
$\mathbf{c}_0$
$\mathbf{x}_{t-k}$
$\mathbf{x}_{t-1}$
$\mathbf{x}_t$
$\mathbf{h}_{t-1}$
$\mathbf{c}_{t-1}$
$\mathbf{h}_t$
$\mathbf{c}_t$
u
w
f
i
o
tanh
tanh

Fig. 5 Representation of an LSTM Network unfolded in time. In addition to the cell state $\mathbf{c}_t$, a hidden state vector $\mathbf{h}_t$ is also formed and propagated along the time axis. The information flow along the time axis is controlled by the forget gate $\mathbf{f}$, the input gate $\mathbf{i}$ and the output gate $\mathbf{o}$. The matrices W, U and V of the respective gates are the learnable parameters of the LSTM cell that is shifted iteratively from $t-k$ to t. The LSTM is initialized with $\mathbf{h}_0$ and $\mathbf{c}_0$

networks, often vanishes when flowing back along the time axis for too long. This leads to a short term behavior, as information from the past is forgotten. To overcome this problem, the Long Short-Term Memory (LSTM) [54] uses a gating mechanism as shown in Fig. 5. Its gates can decide to take new information into account or to neglect it. The gating concept is also used as a basis for further improvements of the LSTM, e.g. Gated-Recurrent Units (GRU) [25]. LSTMs are broadly used in applications like anomaly detection in CPS [47], optimizing productivity perfomance [23], smart grids [2] as well as for artificial generated sensor data [135].

Attention & Transformer Although the introduction of gates has limited the vanishing gradient problem, some core problems remain. When processing long sequences, even gated RNNs often fail to capture information from the start of the sequence properly, as all the information is crammed into one or two hidden vectors of limited size.

To mitigate this issue, the concept of attention has been introduced and modified to self-attention [4, 79]. The main idea of self-attention is to save all of the hidden states in a matrix $Z = (\mathbf{z}_{t-k}, \ldots, \mathbf{z}_t) \in \mathbf{R}^{n\times(k+1)}$ and to calculate a weighted average context vector $\tilde{\mathbf{z}} \in \mathbf{R}^n$ to work with. The original self-attention mechansim learns on what hidden states to focus on:

$$\mathbf{a} = \text{softmax}\left(\mathbf{w}^T \tanh(WZ)\right),$$

where $W \in \mathbf{R}^{l\times n}$ and $\mathbf{w} \in \mathbf{R}^l$ are learnable weights with adjustable dimension l. The attention vector determines how much a hidden state contributes to the context vector

$$\tilde{\mathbf{z}} = Z\mathbf{a}^T.$$

More recently, the hidden state vector has been divided into dedicated parts in order to dynamically identify where to pay attention with key-value attention [28] or multi-head self-attention [122].

The Transformer [122], which gets rid of the recurrent structure of RNNs altogether while solely focusing on attention, has shown spectacular results in many domains [14, 30, 32]. Transformers do not require to process samples time step by time step, as RNNs do, which accelerates training. However, this parallelized structure comes with a drawback: the attention mechanism cannot differentiate between time steps, which has to be taught to the Transformer seperately. Furthermore, the original Transformer has a quadratic runtime with respect to the input length, which makes it infeasible for long sequences. There have been various attempts to tackle these issues and to design a more efficient Transformer architecture for time series [78, 134]. While the Transformer has not seen wide adaption to CPS yet, research on the application to multivariate time series in general has been promising [80, 130].

Neural ODEs Another shortcoming of RNNs is rooted in their origin in a DDE. Standard RNNs are not suited for irregularly sampled time series, as $\mathbf{z}$ is updated once an observation $\mathbf{x}$ occurs. Defining the evolution of $\mathbf{z}_t$ continuously in form of an Ordinary Differential Equation (ODE) $\dot{\mathbf{z}}(t) = f(\mathbf{z}, t)$, where f is realized as a neural net, enables to output values for $\mathbf{z}$ whenever an observation $\mathbf{x}_t$ occurs. [20] presents an approach to efficiently learn Neural ODEs and demonstrates the advantages over RNNs on toy examples. [104] combines the idea of a continuously defined state $\mathbf{z}$ that is updated at the observation times by combining Neural ODEs and RNN, which shows impressive extrapolation capabilities. Often, physical systems show discontinuous behavior (e.g. a moving ball bouncing at the ground), therefore [18] introduces Neural Event Functions for ODEs, which are able to learn ODEs together with points in time where $\mathbf{z}(t)$ suddenly changes.

Neural ODEs have not seen wide adaption to CPS data yet, as current research focuses on architectures [33], fundamentals [86] and general properties (e.g. robustness) [3, 128].

5.3 Conclusion

Time is a fundamental concept for CPS data, yet until now there is no ideal solution that incorporates its characteristics. While RNNs and DDEs seem to be a natural fit, in practice the approach fails to produce good results. Gates, attention and the Transformer architecture try to mitigate issues of RNNs, but they introduce new challenges, as training an LSTM can be hard and Transformers are ill-suited for long sequences. Neural ODEs are another promising approach, but research is still in its early stages. Teaching models to handle time will be crucial in developing models that can work with CPS data effectively. How to do this is still largely unresolved.

6 Challenge 2: Uncertainty and Noise

Often a model is an approximation rather than a comprehensive description of every effect that takes place in the underlying system. This raises the question of how certain a prediction can be. In the literature, uncertainty is subdivided into epistemic uncertainty (model uncertainty), as well as aleatory uncertainty (data uncertainty). While epistemic uncertainty in ML models is caused by insufficiencies in model structure or training of the model, aleatory uncertainty is caused by the loss of information during the data collection of a real world system, e.g. due to noise within the path of measurement or faulty label information. Epistemic uncertainty can be reduced by improving the model or the training process, but it cannot be completely eliminated in practice. Removing aleatory uncertainty from the data is only possible to a very limited extent without further knowledge of the real world system [44, 67].

6.1 Approaches

Methods that allow an estimation of uncertainty for their predictions can be grouped into three categories: (i) statistical methods, (ii) ML approaches based on reconstruction error, and (iii) energy-based ML approaches [90].

(i) Given a set of data points X, statistical information like their distribution can be interfered. Statistical moments of e.g. first (expected value) or second order (variance) can be calculated (in case of their existence) as high level properties that describe the distribution. In case the data can be represented by a Gaussian distribution, the mean μ expresses the average value of all observed data points and the spatial distribution can be described by the standard deviation σ. By **knowing** the probability distribution, it is possible to conclude an uncertainty measure of predicted data points based on the comparison of mean and standard deviation to the training dataset.

However, capturing the probability distribution of technical systems, due to the underlying physical properties and complex interdependencies, is a difficult task. By observing the system, it is possible to estimate the likelihood, i.e. the measure how well the a-priori defined statistical model (the prior) fits the observation. This however requires extensive knowledge of the system's behavior. For tasks with simple probability distributions where an accurate prior can be found, likelihood estimation is near to the actual probability distribution, a reliable measure of confidence can be expected with uncertainty analyses (UA) and sensitivity analyses (SA) [62, 121].

However, the limitation of the described statistical methods is bound to the complexity of the data's distribution [81]. For non-trivial likelihood functions finding accurate priors is challenging: A model's complexity, i.e. the number of degrees of freedom of a model to build a function, has to match the data representation of the problem. For real-world problems with complex data distributions, this means that the epistemic uncertainty is rising as “simple" priors like Gaussian models cannot represent the data. To accomplish this, models with higher complexity, i.e. with a higher degree of freedom, e.g. neural networks, are needed.

Without knowing priors, frequentist approaches like sampling-based techniques, i.e. the selection of representative data points, can help to make statistical inference about the whole model. They are often computationally expensive, but can also be applied to NNs.

(ii) Reconstruction-based ML methods like autoencoders (see Sect. 3) measure the uncertainty by calculating the distance e.g. Mean-Squared-Error $MSE = \|\mathbf{x} - \hat{\mathbf{x}}\|_2$ between ground truth $\mathbf{x}$ and the reconstructed output $\hat{\mathbf{x}}$ [85]. Through learning a latent representation $\mathbf{z}$, consisting of fewer dimensions than the input sample, they are forced to learn the key features of the training data. Therefore, samples coming from a similar distribution to the learned representation will result in a smaller distance, i.e. smaller reconstruction error, whereas data far from the learned distribution would result in a higher reconstruction error.

But using reconstruction error as a measure of uncertainty is difficult as it cannot be clearly interpreted like σ of (i) . Therefore, the distance between the reconstruction and the grounded truth can act as an indicator for uncertainty, but not as a clear measurement of uncertainty. As it measures similarity of the training and predicted data set's distributions, it can be poorly understood as estimate for data uncertainty (e.g. noise). As the similarity of two data distributions are well suited to distinguish normal from anomalous data sets, this method is often used in anomaly detection. Here, a large reconstruction error indicates that an incoming sample is anomalous [44, 75].

(iii) Energy-based ML approaches use likelihood estimation instead of the reconstruction error, i.e. in addition to the distance to the mean, the variance of the data is considered for uncertainty estimation. In comparison to the statistical methods depicted in (i) , where a suitable statistical model has to be defined a-priori (e.g. Gaussian distribution), energy-based methods learn to fit the corresponding likelihood function to the probability of the data's occurrence. The objective is to train the model to maximize said likelihood function. Logically concluded, the more data exists to train the model, the more accurately the fitted likelihood represents the system.

6.2 State of the Art

As distributions of data sets are usually high dimensional and complex, the applicability of (i) statistical methods for modeling data is limited. While (ii) reconstruction based methods measure similarity to the training data set, (iii) energy based methods are considered as state of the art for estimation of uncertainty for predictions. In the following, specific techniques for the individual high-level approaches are presented.

Loss function based While some methods incorporate statistical values by default, some methods use the idea of [98]: They try an energy-based approach to learn statistical concepts by adding σ as a second output to the architecture and by modifying the loss function. For dynamical analysis a loss function that includes a measure of uncertainty is derived in [36] using maximum likelihood approach considering three underling assumptions for an LSTM:

(1) All relevant hidden states $\mathbf{h}$ can be learned from the data.
(2) Gaussian distribution for covariance of the error $\hat{\mathbf{x}}_{t+1} - \mathbf{x}_{t+1}$ (e.g. white noise).
(3) Knowing the hidden states $\mathbf{h}_t$, the remaining noise on each sensor x_t^i is independent (e.g. white noise).

With these assumptions, the conditional probability for $\mathbf{x}_{t+1}$ is given by the multivariate Gaussian distribution in Eq. (3).

$$p_\theta(\mathbf{x}_{t+1}|\mathbf{h}_t, \mathbf{x}_t) = \prod_{i \in \mathbf{N}}^{n} \frac{1}{\sqrt{2\pi}\sigma_{t+1}^i} \exp\left[-\frac{1}{2} \left(\frac{x_{t+1}^i - \hat{x}_{t+1}^i}{\sigma_{t+1}^i} \right)^2 \right] \tag{3}$$

With regard to the maximum likelihood, Eq. (3) can be formulated as maximum-likelihood-error loss function (Eq. (4)) which can be used for training of neural nets that compute x_{t+1}^i and σ_{t+1}^i.

$$L_{t+1} = \sum_{i \in \mathbf{N}}^{n} \left[\left(\frac{x_{t+1}^i - \hat{x}_{t+1}^i}{\sigma_{t+1}^i} \right)^2 + 2 \log \sigma_{t+1}^i \right] \tag{4}$$

Equation (4) can be interpreted as an extension of the MSE loss function that further reduces the distance $\|\mathbf{x} - \hat{\mathbf{x}}\|_2$ through σ_{t+1} with respect to the log term as a penalty.

Sampling based While energy based methods explicitly learn uncertainty, frequentist approaches like sampling allow us to gather information about uncertainty. A universal approach for many applications and network types is introduced in [43]. E.g. dropout (randomly switching off neurons during the training on a NN), which is usually used to avoid overfitting, can also be used to observe uncertainty during the prediction phase. By performing predictions with dropout, the mean and standard deviation can be evaluated empirically. Another approach is training an ensemble of models [70] to estimate σ. Here several models are initialized differently and the samples of the training data that are presented during a training epoch are shuffled. During training, adversarial samples (samples that are slightly different from the original samples) are generated. Furthermore, a loss function similar to Eq. (4) is used to estimate the overall uncertainty.

Bayesian Networks Restricted Boltzmann Machines (RBM) are a type of Bayesian neural networks which is considered as energy based method. It consists of a visible layer of neurons v_i and a hidden layer of neurons h_j. These two layers are used to learn probability distributions of an unknown data distribution by taking binary states [41, 112]. Each layer uses a bias (a_i for the input layer and b_j for the hidden layer). The neurons are connected via their weights w_{i_j}. The bias as well as the weights are first set randomly and then learned in accordance to the data the system is trained on. The value associated with each state of the network is referred to as the energy E of the network (Equation (5)).

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i \in visible} a_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{5}$$

Equation (6) estimates the probability of a configuration $\mathbf{v}$ by the exponential energy term of the observed state divided by the sum of the exponential energy terms of all possible observations $\mathbf{v}^*$. Thus, samples going outside of the learned distribution result in a higher energy level.

$$P(\mathbf{v}) = \frac{e^{-E(\mathbf{v})}}{\sum_{\mathbf{v}^*} e^{-E(\mathbf{v}^*)}} \tag{6}$$

For a given set of parameters and data, θ and $\mathcal{D}$ respectively, the likelihood is the weighted sum of the log-probability of observed states $\mathbf{v}$.

$$\mathcal{L}(\theta, \mathcal{D}) = \frac{1}{N} \sum_{\mathbf{v}^{(i)} \in \mathcal{D}} log\ P(\mathbf{v}^i) \tag{7}$$

The loss function $\mathbf{L}$, i.e. the optimization function, being the negative log-likelihood as shown in Eq. (8), is minimized through learning, and thus the likelihood is maximized.

$$\mathbf{L}(\theta, \mathcal{D}) = -\mathcal{L}(\theta, \mathcal{D}) \tag{8}$$

6.3 Conclusion

Sources of uncertainty are of various types. While epistemic uncertainty can often be reduced by more data or a more detailed model, the occurrence of aleatory uncertainty implies an estimation of uncertainty when predictions are made with NNs. As (*i*) statistical methods are limited to non complex data distributions (*ii*) reconstruction-based methods can act as an indicator for uncertainty but do not quantify it. (*iii*) Energy-based methods such as RBMs or the modification of the loss function allow to learn a measure of uncertainty and are thus able to quantify uncertainty also for complex datasets. Furthermore, sampling based methods offer an empirical opportunity to quantify uncertainty for predictions.

7 Challenge 3: Usage of A-Priori Knowledge

The performance of a trained neural network is measured based on the expected performance on new data samples drawn from an underlying, normally unknown, distribution. While classic signal processing is typically done in up to three dimensions, the situation for high-dimensional problems dealt with in ML is substantially different. Interpolation cannot be done by techniques allowing for accurate estimation of errors. Instead, neural networks are prone to over- or underfitting and therefore limited regarding their capability to generalize to data unseen during training. A function (trained neural network) should be locally smooth with slight differences of the input resulting in similar outputs. However, if this was to be ensured solely through a sufficient amount of samples, the required amount increases exponentially as the dimensionality of the input increases. Therefore, effective priors that capture the expected regularities and complexities of the high-dimensional real-world prediction tasks need to be found and the amount and quality of training samples need to be maximized.

7.1 Approaches

Structure of the respective domain presents a source of regularity which can be utilized by making use of the corresponding symmetries, i.e. transformations leaving certain properties unchanged or invariant. Symmetries of the underlying data impose structure and are powerful priors improving learning efficiency by reducing the space of possible functions to be learned [13]. Arguably, the most illustrative examples can be found in Convolutional Neural Networks (CNNs)[73] applied to images. Convolutional filters with shared weights shifted across a grid combined with pooling layers are characteristic for the CNN network topologies exploiting translational symmetry [48]. In image classification, the image class is unchanged by shifts of the object within the image. Similarly, in time series often encountered in CPS, an anomaly is to be detected as such regardless of the point in time, so shifts are also symmetries in the problem of anomaly detection in CPS. However, whereas flipped images are often considered as equally valid samples, in the case of time series only orientation-preserving transformations may be appropriate choices. Since RNNs introduced in Sect. 5 make use of network topologies allowing to dynamically aggregate information in a way that respects the temporal progression of inputs while also allowing for online arrival of novel data-points, they are a natural choice when dealing with sequential, temporal data. One reason why shifted versions of the sequence can be treated equally is that the RNN input vectors can be seen as points on a temporal grid—a very useful prior.

Whereas in the case of images and sequences data is already recorded with inherent structure in the form of 1D or 2D grids in Euclidean space, no such structure is provided for static analysis of single time steps of multivariate CPS sensor data (see Fig. 6(a)). Typically heterogeneous sensors provide information about numerous subsystems in a non-Euclidean space. Therefore, inputs $x_t^{(a\ldots f)}$ from sensors a through to f are typically concatenated in some arbitrary but fixed order to generate a feature vector $\mathbf{x}_t$ (see Fig. 6(b)) serving as the input for a neural network. However, domain experts such as engineers designing or maintaining such systems are aware of the underlying system structure, namely relationships and

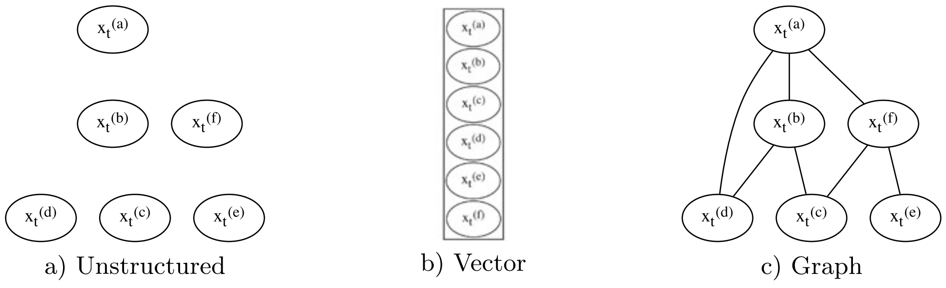


Fig. 6 Sensor inputs $x_t^{(a \ldots f)}$ represented as **a**) an unstructured set, **b**) an input vector $\mathbf{x_t}$ and **c**) a graph G_t with verticals V and edges E adding information in the form of relations between inputs

interdependencies between information from different sensors. This source of knowledge remains to be unlocked by representing the data on a connected graph as shown in Fig. 6(c). Such graph structure can improve learning efficiency by providing additional information that limits the space of functions to be learned and enables the use of modern deep learning techniques able to operate directly on graph-structured data: Graph Neural Networks (GNNs).

Labels created by domain experts represent the most common and direct way of making use a-priori knowledge. However, with labeling being a time-consuming process resulting in quite limited amounts of training samples, supervised ML has recently been outperformed by self-supervised learning algorithms [21, 61], a subclass of unsupervised learning introduced in Sect. 3). Such techniques employ knowledge about the problem to increase the amount of training samples by obtaining labels from the unlabeled data itself. This is done by reconstructing hidden parts of the input from unhidden parts of the input or using data augmentation to learn better representations. Suitable self-supervised ML pipelines cannot be designed without a-priori knowledge allowing for appropriate choices of architecture as well as masking or augmentation techniques.

Simulations of production systems are created during the design phase to model and optimize their expected behaviour. The advantage of such models is twofold: they are interpretable by the domain expert and can be used to transfer knowledge from the expert to the learning algorithm by generating additional training samples. In contrast to real-world training samples typically covering normal system states, simulations can extend this subspace by sampling from the entire distribution of possible system states. This mitigates the issue of deep learning models often not being able to extrapolate to data unseen during training. However, real-world and simulation distributions cannot be expected to be identical without adaptation.

7.2 State of the Art

Compared to Challenges 1 and 2, usage of a-priori knowledge encompasses a set of open and heterogeneous research directions in the context of CPS. Therefore, rather than presenting specific ones in detail, an overview of highly promising methods to be explored by the community is given.

Graph Neural Networks is a collective term for deep learning approaches operating on inputs given in the form of a node feature matrix X, an adjacency matrix A and (optionally) an edge feature matrix X^e. GNNs unlock the potential of deep learning for non-Euclidian data without discarding relational information. Network layers are designed to be permutation equivariant since no canonical ordering of graph nodes is assumed [13]. Modern GNN architectures can be categorized as convolutional [65], attentional [124] or message-passing [5, 45] and are capable of operating on graphs directly processing information in the form of node features as well as edge features. Such models achieve state-of-the-art results for node, link or graph prediction tasks on protein biology [46] or detection of misinformation [89]. Notably, research on GNNs has largely been driven by the increasing availability of graph-structured data [58, 103, 108]. In the field of CPS such structure remains to be leveraged by adding it based on prior knowledge or by learning structure applying latent graph learning approaches [27, 109, 123].

Self-supervised learning has been employed to unlock the potential of the vast amounts of data available today by removing the need for human labeled data. This has led to great success in advancing the field of natural language processing, particularly when combined with transformer architectures [26, 30, 122]. These algorithms make use of knowledge about the language domain by discretizing the feature space to most common words or characters and use the inherent structure of text samples to learn about relations of characters, words or sentences by masking different parts of the input. Other approaches—some of which have already been extended to the graph domain [118]—make use of augmentations [21] or two joint slightly different architectures [50]. Successfully designing suitable self-supervised learning pipelines for CPS will require a-priori knowledge to come up with suitable masking or augmentation techniques.

Simulations model the expected behaviour of CPS even before the system is built. These simulations can be used to uniformly generate high amounts of synthetic samples covering both normal and abnormal system states [17]. Since the synthetic domain is not expected to exactly match the real-world domain of system behaviour, it is necessary to combine both by removing synthetic samples in overlap regions [16] or domain adaptation [29, 119].

7.3 Conclusion

Interdisciplinary cooperation will be a key factor for successfully incorporating a-priori knowledge. Significant parts of the store of knowledge of engineers remain to be utilized and incorporated into ML pipelines. Opportunities range from enhancing inputs by including system structure as graph-structured inputs to enable the use of GNNs, through building CPS-specific self-supervised learning techniques, to making use of often preexisting simulations. Interdisciplinary cooperation will be a key factor for successfully incorporating a-priori knowledge.

8 Challenge 4: Representations and Concepts

Recent years have shown exceptional progress in ML research, particularly deep learning [72]. This method's impact is mainly due to its successes in solving rather specific tasks, such as playing games [111], detecting diseases on medical images [38], or identifying anomalies in CPS sensor data [76]. However, little progress has been made towards general AI. Teaching machines to learn (physical) concepts from observations (e.g. sensor data) is considered a major step in this journey [69]. The emerging research field called Representation Learning (RepL) is dedicated to this objective. Clear, simple and meaningful representations of high-dimensional and complex data can enable the explainability of AI algorithms and thereby also simplify their evaluation. This is particularly relevant in the context of CPS.

8.1 Approaches

In more technical terms, the core motivation of RepL is to build models which are capable of encoding noisy real world observations of (physical) processes, into meaningful representations [6]. These representations are typically vectors of reals numbers, but might also emerge in form of other data formats such as (automate) graphs [57, 101]. Since most RepL models can be trained with unlabeled data, their most common use case is to utilize the typically lower-dimensional representations as input for downstream supervised learning tasks. Based on these representations, less complex ML models with little labeled training data can achieve satisfying performance in many cases [71]. According to [48] a good representation is one that makes it easier to solve subsequent learning tasks, as illustrated in Fig. 7. Among the frequently applied examples are the Word2Vec [88] algorithm for natural language processing and ResNet [21] for computer vision. In its most extreme form, the procedure of simplifying or enabling downstream ML-tasks with representations leads to one-shot [39] and zero-shot learning [113], where only one or even zero training examples are required respectively.

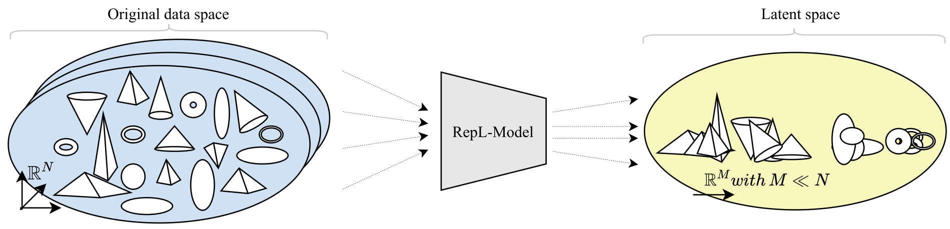


Fig. 7 Principle idea of RepL. RepL models typically encode data points from a high-dimensional space $\mathbf{R}^n$ into a lower-dimensional space $\mathbf{R}^m$. Ideally the representations also encode meaning. In this case objects of the same type are close to each other in the latent space and are clearly separated from objects of another type. Thus, the object types can be separated linearly in the latent space, while a more complex model would be needed in the original data space. This illustrates how downstream ML-tasks can be simplified using RepL models

With regards to CPS, these methods are promising for two main reasons: First, (*i*) good representation of CPS data (in most cases multivariate time series) are an important step to explainable AI. Especially among engineers who are used to working with causal system models, e.g. based on ODEs (see Sect. 5) deep learning methods are often considered as black box solutions which cannot be understood and hence also not trusted. In this context, meaningful and simple representations of the often high-dimensional data are helpful in understanding how AI algorithms work and how corresponding decisions are made. Once a RepL model is trained, downstream ML tasks can be implemented using simpler models such as linear regression with very few and interpretable parameters. This process represents the transition from sub-symbolic to symbolic AI. Engineers and humans in general think in symbols and explain causal relationships, processes and logic in symbols rather than in high-dimensional data spaces. Thus, e.g. mappings from the observation spaces (sensor data) into contexts that are understandable from an engineering point of view, such as automated graphs [57] and potentially even existing ontology models, can be very useful. Second, (*ii*) especially in regard to predictive maintenance use cases such as anomaly detection or failure predictions, the amount of labeled data is usually very limited. Thus, learning good representations from the large amounts of unlabeled data can be highly beneficial for diverse downstream analysis tasks.

However, only little research has been done on RepL for CPS. Apart from a few use cases such as computer vision methods for optical quality control, the majority of CPS related ML use cases rely on sensor data. Thus, in most cases, the training and input data are in the form of Multivariate Time Series (MTS). For this reason, in the following subsection, we will summarize the current state of research related to learning representations with a focus on MTS.

8.2 State of the Art

In line with our approach in Chap. 7, the following section provides an overview of the approaches to Challenge 4 currently discussed in the literature, rather than discussing specific solutions in detail. A very well known and comprehensive (yet slightly outdated) literature review on the field of RepL is given in [6]. According to the authors a good representation captures the underlying factors that generated the data. This definition shows that good representations are anything but unique. In order to disentangle the underlying factors, modern RepL algorithms use so-called clues or priors. In most cases, these clues are implemented in terms of the model architecture or the loss function and aim at enforcing the disentanglement of the learned representations. In a way these clues might also be interpreted as a-priori knowledge (see Sect. 7.1). A list of such clues for unsupervised RepL is provided in [6] and [48].

Time series, unlike images or other typical ML inputs, do not represent explicit features [127]. Thus, mapping MTS to meaningful representations requires particularly strong clues. Some examples of such clues that we consider to be the most important for our context are described below.

Dimensionality reduction and manifolds Learning representations, i.e. interesting and meaningful features, from MTS has a long history. A stream of research that is very closely related to RepL (and might also be considered as such) is dimensionality reduction. Well known and still frequently applied techniques for dimensionality reduction such as PCA [56] and MDS [68] have been developed decades ago. More recently, methods based on manifold learning such as Stochastic Neighbor Embedding [53], t-SNE [82], and UMAP [87] have gained popularity. A very powerful family of algorithms exploiting this clue are AEs (see Sect. 3), which can also be applied to MTS. Different model architectures utilizing RNNs and CNNs in the encoder and decoder network allow the implementation of so-called sequence to sequence models, that encode MTS into lower-dimensional representations [24, 84, 132]. In many cases the behaviour of CPS, which are observed with a large number of sensors, can be described with only a few latent variables. This concept is exploited, for example, in the artificial generation of CPS data [135].

Natural clustering The basic concept of clustering algorithms is described above in Sect. 3. The mapping of objects described in high-dimensional spaces to clusters (some cluster identifier, mostly an integer value) is a kind of representation in the sense of the definition given above. This "clue" mainly assumes, that MTS generated by similar underlying processes (factors) also have similar shapes and patterns according to some distance measure suitable for time series data such as Dynamic Time Warping (DTW) [37]. However, clustering MTS data is not a trivial problem due to the potentially high dimensionality of MTS and the challenge of defining a distance or similarity measure. A review of time series clustering methods is provided in [1]. Some examples of applications of time series clustering for CPS can be found in [42], [106], and [66].

Simple and sparse dependencies between factors The essence of this clue is to assume very simple dependencies between the underlying explanatory factors that created the data. The relationships are assumed to be so simple and general that they can be integrated into the model architecture or the loss functions. This is motivated by the fact that many physical laws can also be described in terms of simple relationships of a few quantities [6]. This "clue" is very popular because it is often used in conjunction with deep generative models, which have achieved very good results in recent years. In the context of RepL, generative models approximate the joint probability $p(\mathbf{x}, \mathbf{z})$, where $\mathbf{x} \in \mathbf{R}^n$ are the observation and $\mathbf{z} \in \mathbf{R}^m$ the latent space variables. A very basic assumption for these simple dependencies is the marginal independence of the latent space variables, such that

$$P(\mathbf{z}) = \prod_{i=1}^{m} P(z_i). \tag{9}$$

This assumption lies at the core of many famous unsupervised RepL algorithms such as Generative Adversarial Networks (GANs) [49] and Variational Autoencoders (VAEs) [64]. Many extensions or versions of VAEs and GANs have been introduced in recent years, some of which introduce other clues in addition to the marginal independence, e.g. mutual information criteria. Examples are the β-VAE [52], FactorVAE [63], β-TCVAE [19], InfoGAN [22] or the InfoVEA [133] just to name a few. Note that the dimensionality reduction and manifold assumption is also explicitly exploited in all of these algorithms.

Others assume simple causal dependencies between the latent variables and the data [115] or even between the individual univariate time series in the MTS [101].

Communicating agents A very new and still experimental approach is to train several small neural networks. These networks act as agents that perform different subtasks. The subtasks are chosen in such a way that different subsets of the underlying explanatory factors are needed to answer them. Together with a loss function that minimizes the amount of information exchanged between agents, meaningful variables can be disentangled in the latent space. These ideas are described in [93] and [60]. The main motivation is to identify physical concepts. The application of this clue in CPS use cases has not been studied yet.

8.3 Conclusion

RepL is the core area of today's deep learning and AI research. For its key challenge, the disentanglement of meaningful latent space variables, many methods have been developed. RepL is particularly relevant for CPS applications because it represents an important step towards explainable AI and because unlabeled sensor data can be used and exploited.

9 Conclusion

This paper explains the role of ML for CPS, provides an overview of the state of the art and discusses challenges that accompany the application of ML algorithms to CPS data.

To highlight the relevance of ML for CPS, Sect. 6 describes several application scenarios. ML algorithms can automatically analyze large amounts of data and can thus be used for tasks like predictive maintenance, resource optimization, the creation of Digital Twins or automated diagnosis.

An overview of ML in general is provided in Sect. 3. The ML algorithms are categorized based on two properties: how they handle dynamic, time-dependent data and how much supervision they need.

Section 4 explains why CPS require specific ML algorithms and gives an overview of the four main challenges identified in this paper: time, uncertainty, a-priori knowledge and meaningful representations.

Section 5 outlines the CPS' main characteristic: time. In order to describe a dynamically changing system, a latent state is introduced. Since traditional approaches often fail to detect complex patterns in the data, the recent surge in computational capacity has motivated an increasing interest in ML algorithms.

Uncertainty, as another important characteristic for CPS, is discussed in Sect. 6. Traditionally, uncertainty can be expressed with statistical methods. However, for high-dimensional and complex CPS data, their applicability is limited. ML algorithms can model more complex distributions and learn uncertainty directly. Furthermore, the usage of sampling approaches allows estimating uncertainty empirically.

How to use existing engineering and physical know-how to improve ML is the topic of Sect. 7. The main challenges include encoding the relationship of the CPS components, minimizing the dependence on labels set by domain experts and leveraging simulated data.

Section 8 discusses how to derive meaningful representations of high-dimensional data. Such representations are key to build explainabe ML models and to transfer knowledge from models built on different datasets.

ML has led to breakthroughs in many domains, such as computer vision, natural language processing or computational biology. The amount of data available rises rapidly, as does the computational capacity that fuels ML models. By applying more and better sensors, a CPS can generate large amounts of data as well. However, there has not been a comparable disruption for CPS yet. Incorporating successful approaches from other domains is a promising start. Yet many application problems arguably exist due to the very nature of CPS data, which are highly interwined with physical processes and have unique challenges that first have to be solved. This paper highlights these challenges and gives an outlook of possible research directions.

In the future, specialized ML algorithms are needed which work on a level of accuracy, reliability and maintainability required in the field of engineering. For this, a corresponding research field of "ML for Engineering" has to be established. The visions are ML algorithms which are fed by engineering knowledge, compute interpretable engineering models and can be deployed in closed-loop control loops and in autonomous systems.

References

1. Aghabozorgi, S., Seyed Shirkhorshidi, A., Ying Wah, T.: Time-series clustering – a decade review. Inf. Syst. **53**, 16–38 (Oct 2015)
2. Alazab, M., Khan, S., Krishnan, S.S.R., Pham, Q.V., Reddy, M.P.K., Gadekallu, T.R.: A multidirectional lstm model for predicting the stability of a smart grid. IEEE Access **8**, 85454–85463 (2020). 10.1109/access.2020.2991067
3. Anumasa, S., Srijith, P.K.: Improving robustness and uncertainty modelling in neural ordinary differential equations. In: 2021 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 4052–4060 (2021). 10.1109/WACV48630.2021.00410
4. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate, https://arxiv.org/pdf/1409.0473
5. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., Pascanu, R.: Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261 (Jun 2018)
6. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1798–1828 (Aug 2013)
7. Bengio, Y., LeCun, Y., Hinton, G.: Deep learning for ai. Communications of the ACM **64**(7), 58–65 (2021). 10.1145/3448250
8. Beyerer, J., Kühnert, C., Niggemann, O.: Machine Learning for Cyber Physical Systems – Selected papers from the International Conference ML4CPS 2018. Springer (2019)
9. Beyerer, J., Maier, A., Niggemann, O.: Machine Learning for Cyber Physical Systems – Selected papers from the International Conference ML4CPS 2017. Springer (2020)
10. Beyerer, J., Maier, A., Niggemann, O.: Machine Learning for Cyber Physical Systems – Selected papers from the International Conference ML4CPS 2020. Springer (2021)
11. Box, G.E.P., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time Series Analysis: Forecasting and Control. John Wiley & Sons, Inc., Hoboken, New Jersey, USA (2015)
12. Breiman, L.: Random forests. In: Machine Learning (2001). 10.1023/A:1010933404324
13. Bronstein, M.M., Bruna, J., Cohen, T., Veličković, P.: Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. arXiv preprint arXiv:2104.13478 (Apr 2021)
14. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. Advances in Neural Information Processing Systems **33**, 1877–1901 (2020)
15. Bunte, A., Stein, Benno an d Niggemann, O.: Model-based diagnosis for cyber-physical production systems based on machine learning and residual-based diagnosis models. Hawaii, USA (2019)
16. Burrows, S., Frochte, J., Völske, M., Torres, A.B.M., Stein, B.: Learning overlap optimization for domain decomposition methods. In: Advances in Knowledge Discovery and Data Mining. pp. 438–449. Springer Berlin Heidelberg (2013)

17. Burrows, S., Stein, B., Frochte, J., Wiesner, D., Müller, K.: Simulation data mining for supporting bridge design. In: Proceedings of the Ninth Australasian Data Mining Conference-Volume 121. pp. 163–170 (2011)
18. Chen, R.T.Q., Amos, B., Nickel, M.: Learning neural event functions for ordinary differential equations. ICLR https://arxiv.org/pdf/2011.03902
19. Chen, R.T.Q., Li, X., Grosse, R., Duvenaud, D.: Isolating sources of disentanglement in variational autoencoders (Feb 2018)
20. Chen, R.T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations, https://arxiv.org/pdf/1806.07366
21. Chen, T., Kornblith, S., Swersky, K., Norouzi, M., Hinton, G.: Big Self-Supervised models are strong Semi-Supervised learners (Jun 2020)
22. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets (Jun 2016)
23. Chiu, M.C., Tsai, C.D., Li, T.L.: An integrative machine learning method to improve fault detection and productivity performance in a cyber-physical system. Journal of Computing and Information Science in Engineering **20**(2) (2020). 10.1115/1.4045663, https://asmedigitalcollection.asme.org/computingengineering/article/20/2/021009/1071865
24. 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 (Jun 2014)
25. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling http://arxiv.org/pdf/1412.3555v1
26. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116 (Nov 2019)
27. Cosmo, L., Kazi, A., Ahmadi, S.A., Navab, N., Bronstein, M.: Latent-Graph learning for disease prediction. In: Medical Image Computing and Computer Assisted Intervention – MICCAI 2020. pp. 643–653. Springer International Publishing (2020)
28. Daniluk, M., Rocktäschel, T., Welbl, J., Riedel, S.: Frustratingly short attention spans in neural language modeling, https://arxiv.org/pdf/1702.04521
29. Daumé III, H.: Frustratingly easy domain adaptation. arXiv preprint arXiv:0907.1815 (Jul 2009)
30. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (Oct 2018)
31. Diedrich, A., Niggemann, O.: Model-based diagnosis of hybrid systems using satisfiability modulo theory. Hawaii, USA (2019)
32. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale, https://arxiv.org/pdf/2010.11929
33. Dupont, E., Doucet, A., Teh, Y.W.: Augmented neural odes, https://arxiv.org/pdf/1904.01681
34. Durbin, J., Koopman, S.J.: Time series analysis by state space methods, Oxford statistical science series, vol. 38. Oxford Univ. Press, Oxford, 2. ed. edn. (2012). 10.1093/acprof:oso/9780199641178.001.0001
35. Eiteneuer, B., Hranisavljevic, N., Niggemann, O.: Dimensionality reduction and anomaly detection for cpps data using autoencoder. In: 20th IEEE International Conference on Industrial Technology (ICIT). IEEE, Melbourne, Australien (Feb 2019)
36. Eiteneuer, B., Niggemann, O.: Lstm for model-based anomaly detection in cyber-physical systems. In: Proceedings of the 29th International Workshop on Principles of Diagnosis. Warsaw, Poland (Aug 2018)

37. Esling, P., Agon, C.: Time-series data mining. ACM Comput. Surv. **45**(1), 1–34 (Dec 2012)
38. Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S.: Dermatologist-level classification of skin cancer with deep neural networks. Nature **542**(7639), 115–118 (Feb 2017)
39. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Trans. Pattern Anal. Mach. Intell. **28**(4), 594–611 (Apr 2006)
40. Fei-Niu, Y., Lin, Z., Jin-Ting, S., Xue, X., Gang, L.: Theories and applications of auto-encoder neural networks: A literature survey. Chinese Journal of Computers (2019)
41. Fischer, A., Igel, C.: An introduction to restricted Boltzmann machines. In: Lecture Notes in Computer Science. vol. 7441 LNCS, pp. 14–36. Springer, Berlin, Heidelberg (2012)
42. Fontes, C.H., Pereira, O.: Pattern recognition in multivariate time series – a case study applied to fault detection in a gas turbine. Eng. Appl. Artif. Intell. **49**, 10–18 (Mar 2016)
43. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. International Conference on Machine Learning pp. 1050–1059 (2016), http://proceedings.mlr.press/v48/gal16.html
44. Gawlikowski, J., Tassi, C.R.N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., Shahzad, M., Yang, W., Bamler, R., Zhu, X.X.: A survey of uncertainty in deep neural networks https://arxiv.org/pdf/2107.03342
45. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 1263–1272. PMLR (2017)
46. Gligorijevic, V., Renfrew, P.D., Kosciolek, T., Leman, J.K., others: Structure-based function prediction using graph convolutional networks. bioRxiv (2020)
47. Goh, J., Adepu, S., Tan, M., Lee, Z.S.: Anomaly detection in cyber physical systems using recurrent neural networks. In: IEEE 18th International Symposium on High Assurance Systems Engineering. IEEE, Piscataway, NJ (2017). 10.1109/hase.2017.36
48. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning, vol. 1. MIT press Cambridge (2016)
49. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks (Jun 2014)
50. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P.H., Buchatskaya, E., Doersch, C., Pires, B.A., Guo, Z.D., Azar, M.G., et al.: Bootstrap your own latent: A new approach to self-supervised learning. arXiv preprint arXiv:2006.07733 (2020)
51. Guo, G., Lu, Z., Han, Q.L.: Control with markov sensors/actuators assignment. IEEE Transactions on Automatic Control **57**(7), 1799–1804 (2012). 10.1109/TAC.2011.2176393
52. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: beta-vae: Learning basic visual concepts with a constrained variational framework. In: ICLR (2017)
53. Hinton, G., Roweis, S.T.: Stochastic neighbor embedding. In: NIPS. vol. 15, pp. 833–840 (2002)
54. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997). 10.1162/neco.1997.9.8.1735
55. Hochreiter, S., Bengio, Y., Paolo, F., Schmidhuber, J.: Gradient flow in recurrent nets: The difficulty of learning longterm dependencies. In: Kolen, J.F., Kremer, S.C. (eds.) A field guide to dynamical recurrent networks. IEEE Press and IEEE Xplore, New York and Piscataway, New Jersey (2009). 10.1109/9780470544037.ch14
56. Hotelling, H.: Analysis of a complex of statistical variables into principal components. J. Educ. Psychol. **24**(6), 417–441 (Sep 1933)

57. Hranisavljevic, N., Maier, A., Niggemann, O.: Discretization of hybrid CPPS data into timed automaton using restricted boltzmann machines. Eng. Appl. Artif. Intell. **95**, 103826 (Oct 2020)
58. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: Datasets for machine learning on graphs. arXiv preprint arXiv:2005.00687 (May 2020)
59. Hyndman, R.J., Koehler, A.B., Ord, J.K., Snyder, R.D.: Forecasting with Exponential Smoothing: The State Space Approach. Springer Berlin Heidelberg (2008)
60. Iten, R., Metger, T., Wilming, H., Del Rio, L., Renner, R.: Discovering physical concepts with neural networks. Phys. Rev. Lett. **124**(1), 010508 (Jan 2020)
61. Jing, L., Tian, Y.: Self-supervised visual feature learning with deep neural networks: A survey. IEEE transactions on pattern analysis and machine intelligence (2020)
62. Kennedy, M.C., O'Hagan, A.: Bayesian calibration of computer models. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **63**(3), 425–464 (2001). 10.1111/1467-9868.00294
63. Kim, H., Mnih, A.: Disentangling by factorising (Feb 2018)
64. Kingma, D.P., Welling, M.: Auto-Encoding variational bayes (Dec 2013)
65. Kipf, T.N., Welling, M.: Semi-Supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (Sep 2016)
66. Kiss, I., Genge, B., Haller, P., Sebestyén, G.: Data clustering-based anomaly detection in industrial control systems. In: 2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP). pp. 275–281 (Sep 2014)
67. Kiureghian, A.D., Ditlevsen, O.: Aleatory or epistemic? does it matter? Structural Safety **31**(2), 105–112 (2009). 10.1016/j.strusafe.2008.06.020
68. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika **29**(1), 1–27 (Mar 1964)
69. Lake, B.M., Ullman, T.D., Tenenbaum, J.B., Gershman, S.J.: Building machines that learn and think like people (2017)
70. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. undefined (2017), https://www.semanticscholar.org/paper/Simple-and-Scalable-Predictive-Uncertainty-using-Lakshminarayanan-Pritzel/802168a81571dde28f5ddb94d84677bc007afa7b
71. Le Paine, T., Khorrami, P., Han, W., Huang, T.S.: An analysis of unsupervised pre-training in light of recent advances (Dec 2014)
72. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (May 2015)
73. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
74. Lee, E.A.: Cps foundations. In: Proceedings of the 47th Design Automation Conference. DAC '10, ACM, New York, NY, USA (2010)
75. Legrand, A., Trannois, H., Cournier, A.: Use of uncertainty with autoencoder neural networks for anomaly detection. In: IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering. pp. 32–35. Conference Publishing Services, IEEE Computer Society, Los Alamitos, California and Washington and Tokyo (2019). 10.1109/AIKE.2019.00014
76. Li, D., Chen, D., Jin, B., Shi, L., Goh, J., Ng, S.K.: MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In: Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series. pp. 703–716. Springer International Publishing (2019)
77. Li, P., Niggemann, O.: Improving clustering based anomaly detection with concave hull: An application in condition monitoring of wind turbines. In: 14th IEEE International Conference on Industrial Informatics (INDIN 2016). Poltiers (France) (2016)

78. Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.X., Yan, X.: Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. Advances in Neural Information Processing Systems **32** (2019)
79. Lin, Z., Feng, M., Santos, C.N.d., Yu, M., Xiang, B., Zhou, B., Bengio, Y.: A structured self-attentive sentence embedding, https://arxiv.org/pdf/1703.03130
80. Liu, M., Ren, S., Ma, S., Jiao, J., Chen, Y., Wang, Z., Song, W.: Gated transformer networks for multivariate time series classification, https://arxiv.org/pdf/2103.14438
81. Loucks, D.P., van Beek, E., Loucks, D.P., van Beek, E.: System Sensitivity and Uncertainty Analysis. In: Water Resource Systems Planning and Management, pp. 331–374. Springer International Publishing (2017)
82. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(86), 2579–2605 (2008)
83. Maier, A., Schriegel, S., Niggemann, O.: Big data and machine learning for the smart factory - solutions for condition monitoring, diagnosis and optimization. Industrial Internet of Things: Cybermanufacturing Systems (2016)
84. Malhotra, P., Vishnu, T.V., Vig, L., Agarwal, P., Shroff, G.: TimeNet: Pre-trained deep recurrent neural network for time series classification (Jun 2017)
85. Mallidi, S.H., Ogawa, T., Hermansky, H.: Uncertainty estimation of DNN classifiers. In: 2015 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2015 - Proc. pp. 283–288. I. of Electrical and Electronics Engineers Inc. (2 2016). 10.1109/ASRU.2015.7404806
86. Massaroli, S., Poli, M., Park, J., Yamashita, A., Asama, H.: Dissecting neural odes. In: H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 3952–3963. Curran Associates, Inc (2020), https://proceedings.neurips.cc/paper/2020/file/293835c2cc75b585649498ee74b395f5-Paper.pdf
87. McInnes, L., Healy, J., Melville, J.: UMAP: Uniform manifold approximation and projection for dimension reduction (Feb 2018)
88. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (Jan 2013)
89. Monti, F., Frasca, F., Eynard, D., Mannion, D., Bronstein, M.M.: Fake news detection on social media using geometric deep learning. arXiv preprint arXiv:1902.06673 (Feb 2019)
90. Multaheb, S., Zimmering, B., Niggemann, O.: Expressing uncertainty in neural networks for production systems. at - Automatisierungstechnik **63(3)**, 221–230 (2021)
91. Murphy, K.: Machine Learning: A Probabilistic Perspective. MIT Press, Cambridge, Massachusetts, USA (2012)
92. Na, S., Xumin, L., Yong, G.: Research on k-means clustering algorithm: An improved k-means clustering algorithm. In: 2010 Third International Symposium on Intelligent Information Technology and Security Informatics. pp. 63–67 (2010). 10.1109/IITSI.2010.74
93. Nautrup, H.P., Metger, T., Iten, R., Jerbi, S., Trenkwalder, L.M., Wilming, H., Briegel, H.J., Renner, R.: Operationally meaningful representations of physical systems in neural networks (Jan 2020)
94. Niggemann, O., Diedrich, A., Pfannstiel, E., Schraven, J., Kühnert, C.: A generic digitaltwin model for artificial intelligence applications. In: IEEE International Conference on Industrial Cyber-Physical Systems (ICPS) (2021)
95. Niggemann, O., Frey, C.: Data-driven anomaly detection in cyber-physical production systems. at - Automatisierungstechnik(63) **63**, 821–832 (2016)
96. Niggemann, O., Lohweg, V.: On the diagnosis of cyber-physical production systems - state-of-the-art and research agenda. Austin, Texas, USA (2015)
97. Niggemann, O., Schüller, P.: IMPROVE - Innovative Modelling Approaches for Production Systems to Raise Validatable Efficiency. Springer Vieweg (2018)

98. Nix, D.A., Weigend, A.S.: Estimating the mean and variance of the target probability distribution. In: The 1994 IEEE International Conference on Neural Networks. pp. 55–60 vol.1. IEEE Neural Networks Council, New York and Piscataway, NJ (1994). 10.1109/ICNN.1994.374138
99. Otto, J., Vogel-Heuser, B., Niggemann, O.: Automatic parameter estimation for reusable software components of modular and reconfigurable cyber physical production systems in the domain of discrete manufacturing. IEEE Transactions on Industrial Informatics (2018)
100. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: Dasgupta, S., McAllester, D. (eds.) Proceedings of the 30th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 28, pp. 1310–1318. PMLR, Atlanta, Georgia, USA (17–19 Jun 2013), http://proceedings.mlr.press/v28/pascanu13.html
101. Pineau, E., Razakarivony, S., Bonald, T.: Unsupervised ageing detection of mechanical systems on a causality graph. In: ICMLA (2020)
102. Rangapuram, S.S., Seeger, M.W., Gasthaus, J., Stella, L., Wang, Y., Januschowski, T.: Deep state space models for time series forecasting. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018), https://proceedings.neurips.cc/paper/2018/file/5cf68969fb67aa6082363a6d4e6468e2-Paper.pdf
103. Rossi, R., Ahmed, N.: The network data repository with interactive graph analytics and visualization. AAAI **29**(1) (Mar 2015)
104. Rubanova, Y., Chen, R.T.Q., Duvenaud, D.: Latent ordinary differential equations for irregularly-sampled time series (2019), https://openreview.net/forum?id=HygCYNSlLB
105. Safavian, S., Landgrebe, D.: A survey of decision tree classifier methodology. IEEE Transactions on Systems, Man, and Cybernetics **21**(3), 660–674 (1991). 10.1109/21.97458
106. Schmidt, T., Hauer, F., Pretschner, A.: Automated anomaly detection in CPS log files. In: Computer Safety, Reliability, and Security. pp. 179–194. Springer International Publishing (2020)
107. Schubert, E., Sander, J., Ester, M., Kriegel, H.P., Xu, X.: Dbscan revisited, revisited: Why and how you should (still) use dbscan. ACM Trans. Database Syst. **42**(3) (Jul 2017). 10.1145/3068335, https://doi.org/10.1145/3068335
108. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AIMag **29**(3), 93–93 (Sep 2008)
109. Shang, C., Chen, J., Bi, J.: Discrete graph structure learning for forecasting multiple time series. arXiv preprint arXiv:2101.06861 (2021)
110. Sherstinsky, A.: Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. Physica D: Nonlinear Phenomena **404**(8), 132306 (2020). 10.1016/j.physd.2019.132306, https://arxiv.org/pdf/1808.03314
111. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D.: Mastering chess and shogi by Self-Play with a general reinforcement learning algorithm (Dec 2017)
112. Smolensky, P.: Information Processing in Dynamical Systems: Foundations of Harmony Theory, p. 194-281. MIT Press, Cambridge, MA, USA (1986)
113. Socher, R., Ganjoo, M., Sridhar, H., Bastani, O., Manning, C.D., Ng, A.Y.: Zero-Shot learning through Cross-Modal transfer (Jan 2013)
114. Sun, X., Bischl, B.: Tutorial and survey on probabilistic graphical model and variational inference in deep reinforcement learning. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 110–119 (2019). 10.1109/SSCI44817.2019.9003114
115. Suter, R., Miladinovic, D., Schölkopf, B., Bauer, S.: Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness. In: Chaudhuri, K., Salakhutdinov,

R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 6056–6065. PMLR (2019)
116. Talkhestani, B.A., Jung, T., Lindemann, B., Sahlab, N., Jazdi, N., Schloegl, W., Weyrich, M.: An architecture of an intelligent digital twin in a cyber-physical production system:. at - Automatisierungstechnik **67**(9), 762–782 (2019). https://doi.org/10.1515/auto-2019-0039
117. Tan, P.N., Steinbach, M., Karpatne, A., Kumar, V.: Introduction to Data Mining, 2nd Edition. Pearson Education, New York, NY, USA (2019)
118. Thakoor, S., Tallec, C., Azar, M.G., Munos, R., Veličković, P., Valko, M.: Bootstrapped representation learning on graphs. arXiv preprint arXiv:2102.06514 (2021)
119. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7167–7176. openaccess.thecvf.com (2017)
120. University, S.: Artificial Intelligence Index Report 2021. HAI Human-centered Artificial Intelligence (2021)
121. Uusitalo, L., Lehikoinen, A., Helle, I., Myrberg, K.: An overview of methods to evaluate uncertainty of deterministic models in decision support (jan 2015). 10.1016/j.envsoft.2014.09.017
122. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in Neural Information Processing Systems **30** (2017)
123. Veličković, P., Buesing, L., Overlan, M.C., Pascanu, R., Vinyals, O., Blundell, C.: Pointer graph networks. arXiv preprint arXiv:2006.06380 (2020)
124. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (Oct 2017)
125. Wang, J.M., Fleet, D.J., Hertzmann, A.: Gaussian process dynamical models. In: Proceedings of the 18th International Conference on Neural Information Processing Systems. p. 1441-1448. NIPS'05, MIT Press, Cambridge, MA, USA (2005)
126. Windmann, S., Niggemann, O., Stichweh, H.: Energy efficiency optimization by automatic coordination of motor speeds in conveying systems (2015)
127. Xing, Z., Pei, J., Keogh, E.: A brief survey on sequence classification. SIGKDD Explor. Newsl. **12**(1), 40–48 (Nov 2010)
128. Yan, H., Jiawei, D., Tan, V.Y.F., Feng, J.: On robustness of neural ordinary differential equations. In: 2020 International Conference on Learning Representations (2020), https://arxiv.org/pdf/1910.05513
129. Yang, Y., Sautière, G., Ryu, J.J., Cohen, T.S.: Feedback recurrent autoencoder. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3347–3351 (2020). 10.1109/ICASSP40776.2020.9054074
130. Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., Eickhoff, C.: A transformer-based framework for multivariate time series representation learning. In: Zhu, F., Chin Ooi, B., Miao, C. (eds.) Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 2114–2124. ACM, New York, NY, USA (08142021). 10.1145/3447548.3467401
131. Zhang, F., Pinkal, K., Wefing, P., Conradi, F., Schneider, J., Niggemann, O.: Quality control of continuous wort production through production data analysis in latent space (2019)
132. Zhang, J.X., Ling, Z.H., Liu, L.J., Jiang, Y., Dai, L.R.: Sequence-to-Sequence acoustic modeling for voice conversion. IEEE/ACM Transactions on Audio, Speech, and Language Processing **27**(3), 631–644 (Mar 2019)
133. Zhao, S., Song, J., Ermon, S.: InfoVAE: Balancing learning and inference in variational autoencoders. AAAI **33**(01), 5885–5892 (Jul 2019)
134. Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W.: Informer: Beyond efficient transformer for long sequence time-series forecasting, https://arxiv.org/pdf/2012.07436

135. Zimmering, B., Niggemann, O., Hasterok, C., Pfannstiel, E., Ramming, D., Pfrommer, J.: Generating artificial sensor data for the comparison of unsupervised machine learning methods. Sensors **21**(7) (2021). 10.3390/s21072397, https://www.mdpi.com/1424-8220/21/7/2397