Methods

Samim Ahmad Multaheb, Bernd Zimmering* and Oliver Niggemann

Expressing uncertainty in neural networks for production systems

Beschreibung von Unsicherheit Neuronaler Netze für Produktionssysteme

https://doi.org/10.1515/auto-2020-0122
Received July 15, 2020; accepted November 16, 2020

Abstract: The application of machine learning, especially of trained neural networks, requires a high level of trust in their results. A key to this trust is the network's ability to assess the uncertainty of the computed results. This is a prerequisite for the use of such networks in closed-control loops and in automation systems. This paper describes approaches for enabling neural networks to automatically learn the uncertainties of their results.

Keywords: machine learning, anomaly detection, neural networks, uncertainty

Zusammenfassung: Die Anwendung des maschinellen Lernens, insbesondere von Neuronalen Netzen, erfordert ein hohes Maß an Vertrauen in deren Ergebnisse. Ein Schlüssel zu diesem Vertrauen ist, dass solche Netze in der Lage sind, die Unsicherheit der berechneten Ergebnisse abzuschätzen. Dies ist eine Voraussetzung für den Einsatz solcher Netze in geschlossenen Regelkreisen und in Automatisierungssystemen. In diesem Beitrag werden Ansätze beschrieben, die es Neuronalen Netzen ermöglichen, die Unsicherheiten ihrer Ergebnisse automatisch zu lernen.

Schlagwörter: Maschinelles Lernen, Anomalieerkennung, Neuronale Netze, Unsicherheit

1 Introduction

Machine Learning (ML) has become very popular in recent years and arouses high expectations from the automation perspective [1]. But while superficially much data seem to be generated by Cyber Physical Production Systems (CPPSs), a different picture can be seen when ML is applied: Data intensive ML approaches such as neural networks or deep learning often fail, or deliver results not usable in a production setting, while in other domains they show a break-through performance. This is, among others, caused by the following reason:

Using ML in a CPPS, e. g., for compensation of anomalies in context of quality assurance, means using it in a closed control loop [2]. I. e. the ML models must predict output values such as resource consumptions, positions, etc. for all input values. From this, actions of the CPPS are triggered automatically. Since no expert is involved in the interpretation of the learning results, an additional requirement is the quantification of the level of uncertainty. This problem can be stated as "How can we develop a metric which quantifies the level of uncertainty of a ML result?" Only if this challenge is solved, modern data-intensive algorithms such as neural networks can be applied successfully to production plants.

In this paper we will describe methods for expressing uncertainty for two different anomaly detection use cases: (1) a static analysis where temporal dependencies are neglected (i. e., point anomalies) and (2) a dynamic analysis where errors become only visible in the system behavior over time (i. e., contextual anomalies).

Static analysis

For tasks such as condition-monitoring or anomaly detection, only the signal values $\mathbf{x} \in \mathbf{R}^n$ at some point in time t are used, i. e., the analysis uses a static feature vector only [3]. Such anomalies are also called point anomalies. Here, the assumption is that no information is coded in the sequence of values and all necessary information is contained in the current signal values.

In this paper we will use neural network-based autoencoders (AE) for this task. Figure 1 shows such a network structure.

***Corresponding author: Bernd Zimmering,** Institut für Automatisierungstechnik, Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg, Holstenhofweg 85, 22043 Hamburg, Germany, e-mail: bernd.zimmering@hsu-hh.de
Samim Ahmad Multaheb, Oliver Niggemann, Institut für Automatisierungstechnik, Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg, Holstenhofweg 85, 22043 Hamburg, Germany, e-mails: samim.multaheb@hsu-hh.de, oliver.niggemann@hsu-hh.de

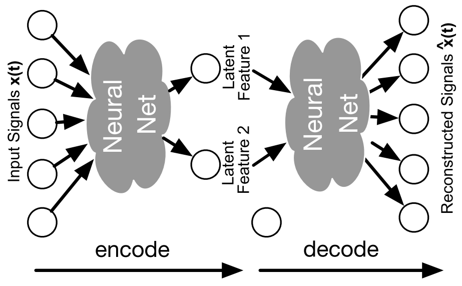


Figure 1: The general solution architecture for the static anomaly detection.

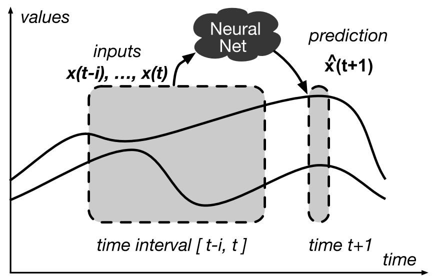


Figure 2: The general solution architecture for the dynamic detection.

First, the static input feature vector $\mathbf{x}$ is transferred into a latent representation, normally corresponding to a low dimensional space—this is called the encoding phase. From this reduced representation, the original signals are reconstructed as $\hat{\mathbf{x}}$—this is called the decoding phase. During the learning phase, the neural network is trained in a way that minimizes a so-called loss function, a function which quantifies how close the true value $\mathbf{x}$ and the estimate $\hat{\mathbf{x}}$ are. During operation, this can also be a measure for the normality of the new data point.

Dynamic analysis

A totally different situation arises when important information is coded in the sequence of signal values over time. Such anomalies are also called contextual anomalies [4]. Given historical data $X = \{x(t-i), \ldots x(t)\}$ and a new data vector $x(t+1)$, the probability is computed that $x(t+1)$ is the logical continuation of the series.

Figure 2 shows the usual solution approach for such a dynamic analysis.

A neural network f is trained during a training phase on a time series $\{\mathbf{x_1}, \ldots \mathbf{x_p}\}$ so that $\hat{\mathbf{x}}_t = f(\mathbf{x_{t-i}}, \ldots \mathbf{x_t}), i \in \mathbf{N}$ is a good prediction for $\mathbf{x}(\mathbf{t}+\mathbf{1})$, i. e., the distance $||\mathbf{x}_{t+1}, \hat{\mathbf{x}}_{t+1}||$ is minimized. During operation, this is a measure for the normality of the new data point $\mathbf{x}_{t+1}$.

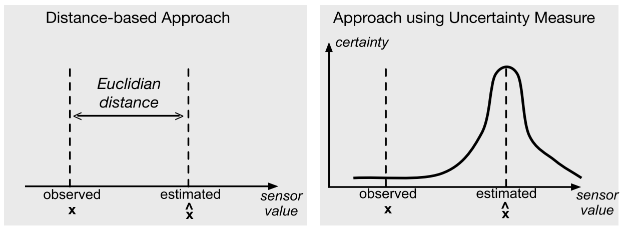


Figure 3: The role of uncertainty in anomaly detection.

In both cases, the learned models must estimate the uncertainty of the prediction. Figure 3 visualized this: In several ML-approaches (Figure 3, *left*), a Euclidean distance $||\hat{\mathbf{x}}, \mathbf{x}||_2$ is used to identify unusual data points. Please note that this distance incorporates a measure of similarity, not a measure of uncertainty about the expected similarity.

On the right hand side of Figure 3, the estimate $\hat{\mathbf{x}}$ comes with a measure of uncertainty, so the degree of similarity between $\hat{\mathbf{x}}$ and $\mathbf{x}$ can be quantified. In this work, we model the uncertainty as a variance of a normally distributed random variable $\hat{\mathbf{x}}$.

This paper deals with two main research questions (RQ):

Research Question 1. How can we, for a dynamic anomaly detection, encode the uncertainty of the prediction $\hat{\mathbf{x}}_{t+1}$ into the metric $||\mathbf{x}_{t+1}, \hat{\mathbf{x}}_{t+1}||$?

Research Question 2. How can we, for a static anomaly detection, encode the uncertainty of the prediction $\hat{\mathbf{x}}_t$ into the metric $||\mathbf{x}_t, \hat{\mathbf{x}}_t||$?

2 State of the art

To estimate prediction uncertainty in CPPSs, several methods exist which we group into three categories: (1) statistical methods, (2) machine learning approaches based on the reconstruction error, and (3) energy-based machine learning approaches.

(1) Uncertainty in and itself is a statistical value, bound to several factors along the path of observation, as was described in Section 1. To express an uncertainty estimate, given an observation of data points, there are several information to derive from the distribution of data points itself, including the mean and spatial distribution. While the mean expresses the average value of all observed data points, another measure is ought to be known: the standard deviation. The standard deviation expresses the difference of each data point from said mean. By knowing the

probability distribution, it is possible to conclude an uncertainty measure of an observed data point based on the difference of mean and standard deviation. Capturing the probability distribution of technical systems however, 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 statistical model fits the observation. This however requires extensive knowledge of the system's probability behavior.

For tasks with simple probability distributions where the distribution is known, or the a-priori likelihood estimation is near to the actual probability distribution, a reliable measure of confidence can be expected with uncertainty and sensitivity analyses, UA and SA, respectively. With growing complexity and lack of knowledge of the probability distribution, logically concluded, the uncertainty of the model rises.

In literature, measuring uncertainty within the family of traditional statistical methods often leads to sampling-based techniques. With sampling, i. e., the selection of representative data points to make statistical inference about the whole model, the likelihood function is estimated significantly more precise.

One way to do so is UA, usually with Monte Carlo sampling—in this case repeated random selection of input parameters. By selecting random variations of input parameters, the model outputs a distribution of samples.

SA uses gradual variations of input parameters to identify the model's response, and thus find out which parameters are most sensitive to the outcome. Input parameters that result in smaller output changes are more robust, whereas those that yield a larger change in the output indicate that the uncertainty for that particular parameter is high. Both methods however need a large amount of model runs. Furthermore, finding out the underlying likelihood distribution becomes difficult with complex data distributions comprising more parameters where interdependancy and non-linear interactions come into play [5, 6].

However, the limitation of the described statistical methods is bound to the complexity of the data's distribution. Since the likelihood has to be estimated before-hand, this task seems highly challenging for non-trivial likelihood functions. Generally speaking, 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 more complex data distributions, this is not always possible with statistical methods. To accomplish this, models with higher complexity, i. e., with a higher degree of freedom, e. g., neural networks, are needed.

(2) Reconstruction-based machine learning methods as the autoencoder in Figure 1 measure the uncertainty by calculating the distance between ground truth and the reconstructed output based on the latent representation [7]. Data samples coming from a similar data distribution like the learnt representation, will result in a smaller distance, i. e., smaller reconstruction error, whereas anomalies would result in a higher reconstruction error. This is also being referred to as traditional or deterministic autoencoder [8].

This approach is widely used for unsupervised analysis of static problems, referring to Section 1, where data distributions are time-invariant, i. e., not changing or dependent of time, and the learning is not human-assisted via labels. The problem with this approach is that uncertainty in this context means the imparity of known and observed data points. Referring to (1) above, uncertainty is a measure of two variables: mean and standard deviation. Reconstruction-based methods provide an estimate to the former, i. e., mean, only. This has the logical implication that such methods cannot draw an inference about the standard deviation of an observed data point, without further adaptation to the algorithm to capture variance, e. g., with sampling techniques.

(3) Energy-based machine learning 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 (1), where the likelihood has to be estimated a-priori, 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. Logically concluded, the more data exist to train the model, the more accurate the fitted likelihood represents the system.

This approach includes Restricted Boltzmann Machines (RBM), the use of ensembles targeting the mean and variance of data samples, and dropout as a statistical means to estimate uncertainty, which are further described below [9, 10, 11].

RBMs are a type of bayesian neural networks, consisting of a visible layer, i. e., the input layer, and a hidden layer of neurons, i. e., a layer of neurons where the calculation takes place to output the intended result, in the following example. They are used to learn probability distributions of an unknown data distribution [12, 13]. In the energy equation (1), v_i, h_i represent the binary states of the visible neurons i and hidden neurons j. a_i and b_j represent the respective bias, i. e., a factor to shift the activation. w_{ij} represents the weights, a factor to transform the input, between the neurons. Both variables are first set randomly

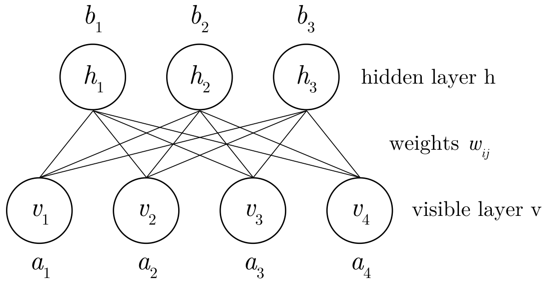


Figure 4: The Restricted Boltzmann Machine is built of a visible layer, and a hidden layer of neurons v_i, and h_i, respectively, with its respective biases a_i, and b_i. Circles in this regard represent the neurons, with lines in between representing the weighted connections w_{ij} in between. The term "restricted" refers to the omission of intra-layer connections between neurons [12, 9].

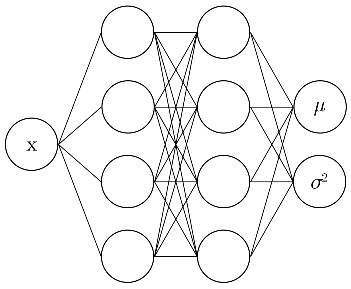


Figure 5: Architecture of a feed-forward neural network with an input layer and its input **x**, two hidden layers and an output layer. Circles represent neurons. Lines represent the neurons' inter-layer connections. The neurons of the output y are distributional parameters: mean μ and variance σ^2.

and then learned in accordance to the data the system is trained on. The value associated with each state of the network is thus referred to as the energy E of the network.

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i \in visible} a_i v_i - \sum_{i \in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij}, \qquad (1)$$

$$P(\mathbf{v}) = \frac{e^{-E(\mathbf{v})}}{\sum_{\mathbf{v}^*} e^{-E(\mathbf{v}^*)}}. \qquad (2)$$

The probability of a configuration **v** is estimated 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.

$$\mathcal{L}(\theta,\mathcal{D}) = \frac{1}{N} \sum_{\mathbf{v}^{(i)} \in \mathcal{D}} log\ P(\mathbf{v}^i). \qquad (3)$$

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 **v**. The loss function **L**, i. e., the optimization function, being the negative log-likelihood as shown in Equation (4) is minimized through learning, and thus the likelihood is maximized.

$$\mathbf{L}(\theta,\mathcal{D}) = -\mathcal{L}(\theta,\mathcal{D}). \qquad (4)$$

Another method uses targets of mean and variance to directly learn uncertainty from the dataset [10, 4]. The network outputs two values, the predicted mean $\mu(\mathbf{x})$ and variance $\sigma^2(\mathbf{x})$. The maximum likelihood gets calculated using the mean negative log-likelihood as the loss function.

$$\mathbf{L}(\mathbf{x},y) = -log\ p_\theta(y_n|\mathbf{x}_n) = \frac{log\ \sigma_\theta^2(\mathbf{x})}{2} + \frac{(y-\mu_\theta(\mathbf{x}))^2}{2\sigma_\theta^2(\mathbf{x})} + constant. \qquad (5)$$

Using ensembles, i. e., several networks, of the same trained model, but with varying initialization of the neural network, the variance of data points can be measured. Using variation in the model's initializations and architecture with the same unaltered data input, orthogonal to the UA approach, the learnt data distribution is induced with variance in every single data point. This leads to statistical inference. Testing unknown data points outside of the learnt data distribution therefore will result in an inherently different outcome. Through different initialization, anomalies cannot reach the same outcome, and thus receive a higher uncertainty value. However, the training of ensembles can be computationally expensive in comparison to single neural network methods.

Another method to make statistic inference is using dropout. Dropout, which is the stochastic omission of neurons, can be used as a simple regularization method, i. e., to enhance the generalization ability of the network [14]. If dropout is applied on every layer in both training and testing phase, it can moreover be used for uncertainty estimation [11]. Through the stochastic omission of neurons, every training procedure uses a slightly different neural network. These slightly different networks can be seen as sub-networks. Training on the same data, every sub-network learns to output the same result, while for data outside of the known data distribution, i. e., anomalies, the result of each sub-network will vary significantly. This approach can also be interpreted as ensemble model combination [14].

In addition to the requirements due to the observed dataset, irrespective of the quality of results, choosing an approach is also bound to the problem intended to solve. Generally speaking, for classification and regression tasks for static analyses, simple feed-forward network architectures are used. Additionally, Convolutional Neural Networks can be mentioned at this point [15]. Regression tasks for static cases are handled similarly. For dynamic analyses, a recurrent architecture, such as Long Short-Term Memory, Gated Recurrent Units, or Sequence-to-Sequence architectures such as Transformers are mentioned in [16, 17], and [18].

While some networks and algorithms are inherently designed and used for uncertainty estimation, e. g., RBM, other techniques are independent of the network's architecture, i. e., can be built on top of existing architectures, e. g., stochastic dropout and ensemble model combination. Also, the techniques used vary in their tasks to be solved. While reconstruction-based machine learning techniques are used for static analyses, and are suited for unsupervised learning in settings where anomalous data is scarce, energy-based machine learning techniques can also be used for dynamic analyses and when training data exist abundantly. There are also other approaches such as Variational Autoencoder and Generative Adversarial Networks [8, 19], which however would be out of the scope of this paper.

For the contribution part of this paper, we exemplarily show two approaches, one for each case, i. e., static and dynamic analysis, respectively. In addition to the research questions RQ1 and RQ2, we ascertain if for both static and dynamic analyses the standard algorithms, without further adjustment, i. e., out-of-the-box, achieve satisfactory results in anomaly detection. We see the research gap in whether adjustments to the algorithms are specific to the challenges of CPPSs.

3 Solutions

For the dynamic analysis part, we use an LSTM comparing a reconstruction based metric and a metric incorporating the variance of the prediction. Similarly, for the static analysis part we compare the ability of an Autoencoder (AE) to detect anomalies with that of a modified version, i. e., Denoising Autoencoder (DAE). We then compare the results with an RBM as an energy-based approach.

We use artificially generated datasets, and in addition a real-world dataset for the dynamic analysis, to analyze the influence of uncertainties on the performance of the different solutions.

3.1 Dynamic analysis

Time-series data extracted from CPPSs in most cases contain a certain amount of noise, either caused by the mea-

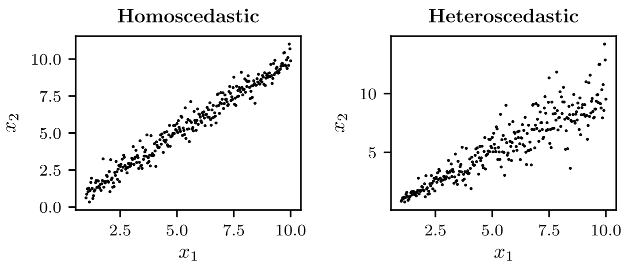


Figure 6: Example for homo- and heteroscedastic uncertainty.

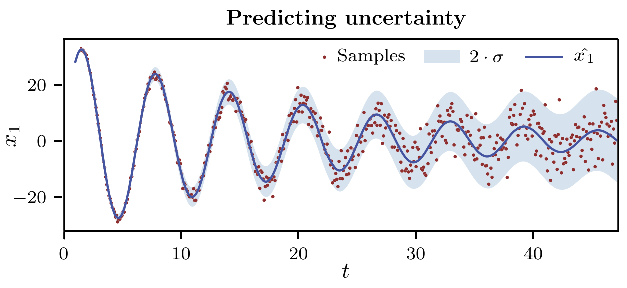


Figure 7: Example for prediction in context of uncertainty. For a signal containing a normal distributed heteroscedastic noise, the mean $\hat{x}_1$ as well as the standard deviation σ is predicted.

surement itself, or variances in underlying processes. In cases where this kind of uncertainty is not constant, i. e., heteroscedastic, a prediction is difficult, as it will likely show similar noise. It is hardly possible to distinguish between certain and uncertain predictions $\hat{x}_i$ by monitoring a fixed distance to the sensor value x_i, as this distance can be shifting with $\mathbf{x}$. Referring to Figure 3 in the introduction of this paper, a prediction with a large Euclidean distance to the ground truth, can be certain in case the normal behavior of the CPPS shows a high variance in this specific operational mode.

Figure 6 shows the same signal added with homoscedastic and heteroscedastic uncertainty. The heteroscedastic case shows Gaussian noise where σ^2 is a linear function of x. This dependency can be seen in similarity to the prediction of $\hat{x}$: It is another function that has to be learned from the dataset and thus be predicted.

In case of comparing new samples to the predictions made, the learned confidence interval helps to classify whether the new samples were expected as shown in Figure 7. The shaded area shows a 2σ confidence interval. This will later be used for anomaly detection.

To predict data points from a time-series of multiple sensors of a CPPS, a recurrent neural net (RNN) architecture can be applied. An RNN uses internal states to encode temporal information between the data points. Similarly to an autoencoder (Figure 1), these internal, i. e., hidden,

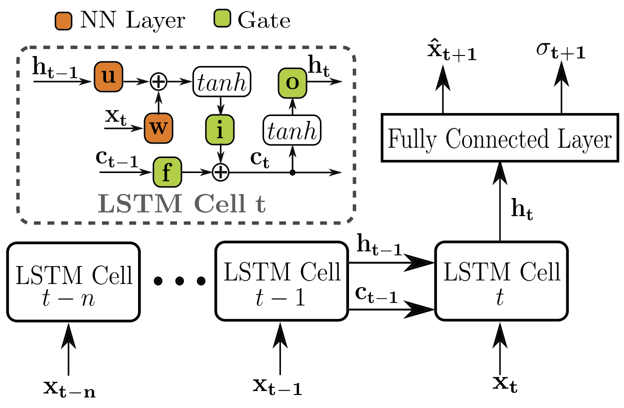


Figure 8: Neural network architecture for dynamic data using an LSTM layer as an RNN.

states of the RNN have to be decoded into a prediction of $\hat{\mathbf{x}}_{t+1}$. For this task, a feed-forward architecture containing one or more layers is used. This architecture implies that the sensor signals are generated from a latent space, i. e., hidden states, inside the CPPS.

For our approach, we use a Long Short-Term Memory network (LSTM), which is a common type of RNN [16]. As depicted in Figure 8, the LSTM layer takes the time-series $\{\mathbf{x_t}, \cdots, \mathbf{x_{t-n}}\}$ as input and encodes a hidden representation $\mathbf{h_t}$ of it. To incorporate a measure of uncertainty, the fully connected layer decodes the hidden states into a prediction for the sensor values $\hat{\mathbf{x}}_{t+1}$ and their standard deviation $\boldsymbol{\sigma}_{t+1}$. The following part will first explain how the LSTM decodes a time series into a hidden representation followed by a derivation of a loss function that incorporates the $\boldsymbol{\sigma}_{t+1}$ output to enable the presented neural network to be trained.

The zoomed representation in Figure 8 (dashed lines) shows an LSTM cell architecture as used in [20]. The central element that characterizes the family of RNNs is the use of a memory cell [21]. In case of an LSTM it is realized as a cell state $\mathbf{c}$ which models internal states of the CPPS. The interconnection of the cells through $\mathbf{c}$ and $\mathbf{h}$ enables information to flow forward from past time steps to the current time step t. The LSTM cell at time step t takes the input samples $\mathbf{x_t}$, the previous hidden representation $\mathbf{h_{t-1}}$, and the previous cell state $\mathbf{c_{t-1}}$ into account to build the cell state $\mathbf{c_t}$.

Gates control the information flow inside the LSTM cell and enable the network to take important information from a certain point in time into account, e. g., the beginning of a transient phase, and neglect other points in time, e. g., stationary phases. The gates are realized as neural network layers according to Equation (6).

$$\begin{aligned}\mathbf{f_t} &= \boldsymbol{\sigma}(\mathbf{W_f x_t} + \mathbf{U_f h_{t-1}} + \mathbf{V_f c_{t-1}}),\\ \mathbf{i_t} &= \boldsymbol{\sigma}(\mathbf{W_i x_t} + \mathbf{U_i h_{t-1}} + \mathbf{V_i c_{t-1}}),\\ \mathbf{o_t} &= \boldsymbol{\sigma}(\mathbf{W_o x_t} + \mathbf{U_o h_{t-1}} + \mathbf{V_o c_t}).\end{aligned} \tag{6}$$

The matrices $\mathbf{W}, \mathbf{U}, \mathbf{V}$ of every gate are weights of the NN layer and thus learned during the training phase of the network. The sigmoid function limits the gates values to an interval of $[0, 1]$. Depending on their particular value, information is neglected or accepted. In case of the forget gate $\mathbf{f_t} = \mathbf{1}$ and the input gate $\mathbf{i_t} = \mathbf{0}$, the cell state $\mathbf{c_t}$ is calculated only from the previous cell state $\mathbf{c_{t-1}}$. In the opposite case where $\mathbf{f_t} = \mathbf{0}$, $\mathbf{i_t} = \mathbf{1}$, only the hidden representation $\mathbf{h_{t-1}}$ and the input samples $\mathbf{x_t}$ are taken into account. Before they pass the input gate $\mathbf{i_t}$, they are weighted by $\mathbf{u}$, $\mathbf{w}$ which are learned during the training process.

For cases where the gates take values $0 < \alpha < 1$, $\mathbf{c_t}$ is calculated from new inputs as well as the cell state of the previous cell. The tanh multiplication acts as a regularization as it restricts the cell states within an interval of $[-1, 1]$. Which features of $\mathbf{c_t}$ contribute to the hidden representation vector $\mathbf{h_t}$ is controlled through the output gate $\mathbf{o_t}$.

The dimensions of $\mathbf{h_t}$ and $\mathbf{c_t}$ determine the number of internal states. If it is chosen less than the number of sensors ($dim(\mathbf{x_t})$), the LSTM is forced to learn a reduced latent representation, similar to the decoder part of the autoencoder in Figure 1. To train the shown network by reducing the prediction loss, the loss function has to incorporate $\boldsymbol{\sigma}_{t+1}$. This can be achieved by expressing the loss function as a maximum likelihood approach considering three underling assumptions:

(1) All relevant hidden states $\mathbf{h_t}$ 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 probability for $\mathbf{x}_{t+1}$ is given by the multivariate Gaussian distribution in Equation (7).

$$p_\theta(\mathbf{x}_{t+1}|\mathbf{h_t}, \mathbf{x_t}) = \prod_i \frac{1}{\sqrt{2\pi}\sigma_{t+1}^i} \exp\left[-\frac{1}{2}\left(\frac{x_{t+1}^i - \hat{x}_{t+1}^i}{\sigma_{t+1}^i}\right)^2\right]. \tag{7}$$

We enforce the NN to compute x_{t+1}^i and σ_{t+1}^i, with regard to the maximum likelihood, to describe the data by expressing the loss function as the negative log maximum likelihood of Equation (7). This results in the integration of the standard deviation into the mean square error loss

Table 1: Comparison of a distance-based metric (MSE) to the energy-based likelihood metric (MLE) for two artificial datasets (Art. DS) and an Ion Mill Etching System (IMES DS). The upper rows show the F1 measures of MLE and MSE. The lower columns the metric that was used to distinguish normal from anomalous data.

	Art. DS1	Art. DS 2	IMES DS
F1 MSE	0.13	0.62	0.52
F1 MLE	0.7	0.62	0.7
Metric MSE	$0,4 \cdot MAX$	$1,6 \cdot MEAN$	$1,6 \cdot MEAN$
Metric MLE	2σ	1σ	2σ

function (MSE) as proposed by [4]. By dividing through σ^i_{t+1}, and the log term as a penalty term in Equation (8), the maximum-likelihood-error loss (MLE) is formed.

$$L_{t+1} = \sum_i \left[\left(\frac{x^i_{t+1} - \hat{x}^i_{t+1}}{\sigma^i_{t+1}} \right)^2 + 2 \log \sigma^i_{t+1} \right]. \tag{8}$$

While using the standard MSE loss function for the combination of an LSTM network followed by a fully connected layer cannot properly model noise in a signal, a leftover loss at the end of the learning phase will be present. The MLE loss forces the NN to reduce this leftover loss by learning σ^i_{t+1} that fits best to the noisy sensor data. Mapping the hidden representation of the LSTM into σ^i_{t+1} and $\hat{x}^i_{t+1}$ by a fully connected layer can be interpreted as mapping to a specific internal state of the CPPS into prediction and standard deviation. In case of heteroscedastic uncertainty as shown in Figure 6, the linear function changing the variance, as well as $x_2 = f(x_1)$ itself is learned.

As the introduced NN architecture is able to predict the standard deviation $\boldsymbol{\sigma}_{\mathbf{t+1}}$ together with the sensor values $\hat{\mathbf{x}}_{\mathbf{t+1}}$, a confidence interval as shown in Figure 7 can be applied to the prediction. It acts as a measure of uncertainty for predictions made, and can be used to make further decisions, e. g., classifying measured data points to detect anomalies. Furthermore, in control loops a high standard deviation in certain operational modes can be taken as mistrust in the NN model, leading to using a backup model with lower performance [22].

In [23] a NN similar to Figure 8 is used to detect anomalies on two artificial and one real world dataset. With reference to Figure 7, a fixed decision border together with an MSE loss function is compared to the MLE loss function in Equation (8) using an automatically generated uncertainty interval $|x^i_{t+1} - \hat{x}^i_{t+1}| > k \cdot \sigma^i_{t+1}$.

Table 1 shows the F1 scores together with the metric used. Artificial dataset 1 contains a sine and a saw-tooth signal with Gaussian noise of different intensity levels added to each signal. Six anomalies are added to the sine signal as shown in Figure 9.

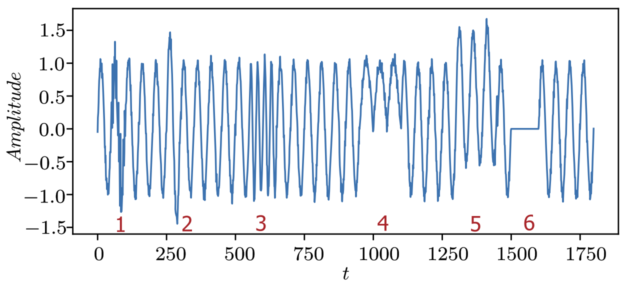


Figure 9: Anomalies in sine signal of DS1: (1) Higher Noise, (2) Higher Amplitude, (3) Higher Frequency, (4) Lower Amplitude, (5) Offset, (6) No Signal.

Artificial dataset 2 contains ten signals that are generated from a latent random-walk on surface in a three dimensional room which is mapped into 10-D space through a sinusoidal function. After a phase of normal behavior, an error of two intensities is added to the latent space. The results shown refer to the high intensity error, as this case can be properly identified as an anomaly. For low intensity errors both metrics show poor results. The real world dataset consists of 231 days of operation of an Ion Mill Etching System. As several recipes for products are operated, the most frequent one (recipe No. 67 with 51 % of the time) is used. The degeneration processes of a pressure signal is identified and marked as anomalous **before** this leads to non-acceptable product quality.

For all datasets it is shown that the learned uncertainty interval of the MLE Approach is at least equal or better than a fixed border for every sensor. In DS 1 the anomalies, added to one of two signals, are detected very well. DS 2 shows similar results for MSE and MLE. The manipulation of the latent space for the anomalous case results in a well detectable anomalous system behavior as all sensor signals show anomalous behavior. Similar to DS 1 the IMES DS shows good results for the MLE metric. This can be explained as many sensors behave normal and only one sensor shows anomalous behavior. The choice of metric also shows that 2σ is a reliable hyperparameter for anomaly detection. This reduces the amount of setting manual thresholds, and makes the method suitable for practical CPPS integration.

3.2 Static analysis

For the static analysis case, we use a Denoising Autoencoder [8]. We compare this result with a standard Autoencoder, and a Restricted Boltzmann Machine to include an energy-based approach.

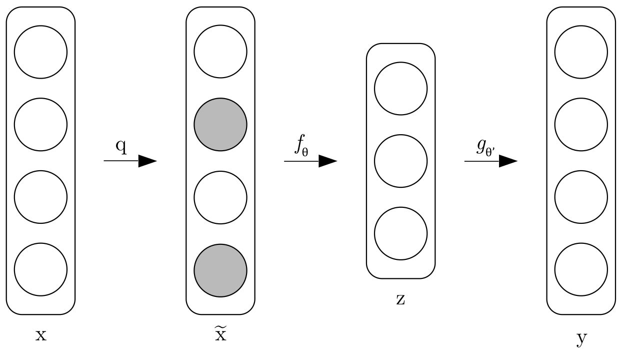


Figure 10: Representation of the basic principle of a Denoising Autoencoder.

As already shown in Fig. 1, the autoencoder is a stack of layers of neurons, with at least one hidden layer, consisting of an encoder and a decoder architecture. We used an altered approach, i. e., a Denoising Autoencoder (DAE). The difference to the vanilla autoencoder depicted in Section 1 is limited to the input. We use $\tilde{\mathbf{x}}_t \in \mathbf{R}^n$ which represent damaged data points. The objective of the DAE is to reconstruct the unaltered data distribution $\mathbf{x}$ for the training and validation phase. Through a stochastic function $\mathbf{q}$, i. e., Gaussian noise function, the initial data input $\mathbf{x}$ is transformed to a manipulated version $\tilde{\mathbf{x}}$ which serves as input to the encoder-decoder architecture. With the encoder network f, the altered input $\tilde{\mathbf{x}}$ gets encoded into a latent representation $\mathbf{z}$, before decoded to the output $\mathbf{y}$ with the decoder network g'_θ.

The data used are generated through a random walk algorithm, in reference to Section 3.1, and normalized in accordance to its mean and variance. In total 50 sensors were simulated, with 50,000 signal values each. The architecture consisted of 3 encoder layers reducing in terms of number of neurons, the latent representation layer, consisting of 10 neurons, and 3 decoder layers with increasing number of neurons back to the same number of outputs as the initial number of input neurons. Before the data go into the encoder, Gaussian noise is being applied. We applied noise only in the training and validation phase. We used the Adam optimizer with a learning rate of 10^{-3} and a weight decay of 10^{-4}, and trained for 100 epochs with a batch size of 50. We minimize the reconstruction error $\mathbf{L}(\mathbf{x}, \mathbf{y})$ as a mean squared error (MSE) through training of the network. Additionally, an equal size of validation dataset, and test dataset, consisting of normal and anomalous data points, respectively, are used for the evaluation.

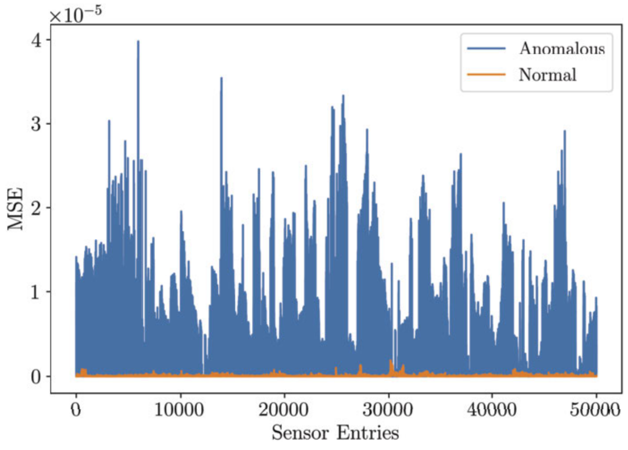


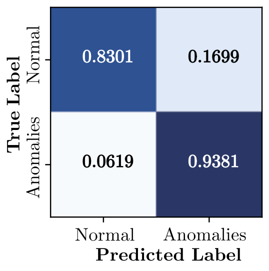
Figure 11: Visual representation of one exemplary sensor with the distances of prediction and ground truth of the normal (orange; validation dataset) and anomalous (blue; test dataset) data points. Prediction after the training of the DAE.

Table 2: Comparison of the Denoising Autoencoder (DAE) with a standard Autoencoder (AE), and a Restricted Boltzmann Machine (RBM) with the same random-walk point anomalies dataset.

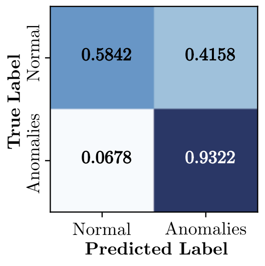
	AE	**DAE**	**RBM**
F1 Score	0.75	0.88	0.93

Categorizing data points as normal or anomalous is estimated through the MSE, serving as the distance between ground truth and predicted values, and a threshold that is set at the standard deviation of 1σ above the mean of the validation error. Predictions going beyond the threshold are viewed as anomalies, with the Euclidean distance serving as an estimate for uncertainty. One of the sensor's distance error is depicted exemplarily in Figure 11.

We found that adding Gaussian noise leads to higher F1 scores. By using the noise-induced input $\tilde{\mathbf{x}}$, we achieve robustness to a partial manipulation of the input data. The manipulation also serves generalization purposes through the augmentation of training data due to the stochastic transformation of the input [8]. We used different levels of gaussian noise, and found 20 % to be a good estimate. The results are based on the top three averaged F1 scores. For the confusion matrices in Figure 12, we used the average of the respective results. Additionally, we show in Table 2, that the RBM on the same data performs better than both autoencoder. This goes back to the inherent ability of energy-based methods to learn probability distributions from an unknown data distribution, referring to Section 2.



(a) Confusion Matrix DAE



(b) Confusion Matrix AE

Figure 12: (a) Evaluating the confusion matrix of the DAE shows a robust detection of anomalies. 93.81 % of all anomalies were detected, whereas only 83.01 % of non-anomalous data points were classified as such. This means around 16.99 % detections of anomalies are in fact wrong and only 6.19 % of detected anomalies are in fact non-anomalous. (b) It is noticeable that the vanilla autoencoder, i. e., without denoising ability, performs significantly worse in detecting normal data, i. e., the false positive rate is significantly higher. The true positive rate shows no significant difference to the DAE.

4 Conclusion

For static, as well as dynamic data, we presented to what extent uncertainty can be expressed in neural networks regarding the task of anomaly detection.

For dynamical data we presented an approach using an LSTM network combined with an energy-based loss function derived from a maximum likelihood expression of the remaining prediction uncertainty. Experimental results on three different datasets, comparing the MLE metric to the MSE metric, corroborate that the approach is able to encode uncertainty and suits the application of anomaly detection (RQ1).

For the static analysis we showed that the Denoising Autoencoder, as well as the Restricted Boltzmann Machine is suitable for anomaly detection. Experiments show that both approaches are well able to detect anomalies in multi-dimensional artificial data (RQ2). While in security-related tasks and environments, e. g., airport, medical industry, the relatively high false positive rate of the standard autoencoder might be acceptable, for CPPSs it can lead to high costs in manual inspections. We showed the advantage of adding noise to the training data to achieve an autoencoder with higher generalization abilities, i. e., a Denoising Autoencoder, by achieving a significantly lower false-positive rate. We further showed the advantage of an energy-based machine learning model, i. e., RBM, over reconstruction-based machine learning approaches in anomaly detection.

It has to be noted that any alteration to the hyperparameters or the training data used will lead to different results and an inferior representation of the internal states of the CPPS. In this case the assumptions made in Section 3.1 are violated.

Author contributions: Samim Ahmad Multaheb and Bernd Zimmering contributed equally.

Acknowledgment: Special thanks go to Carlo Voss and Maurice Thomas for their support in preparing the models and data pipelines.

Funding: S. A. M. is being funded by the Avicenna scholarship foundation.

References

1. R. Mikut, “Maschinelles lernen und künstliche intelligenz – eine revolution in der automatisierungstechnik oder nur ein hype?,” *at – Automatisierungstechnik*, vol. 68, no. 5, pp. 295–300, 2020.
2. B. Lindemann, N. Jazdi and M. Weyrich, “Detektion von anomalien zur qualitätssicherung basierend auf sequence-to-sequence lstm netzen,” *at – Automatisierungstechnik*, vol. 67, no. 12, pp. 1058–1068, 2019.
3. B. Eiteneuer, N. Hranisavljevic and O. Niggemann, “Dimensionality reduction and anomaly detection for cpps data using autoencoder,” in *20th IEEE International Conference on Industrial Technology (ICIT)*, (Melbourne, Australien), IEEE, Feb. 2019.
4. B. Eiteneuer and O. Niggemann, “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.
5. L. Uusitalo, et al., “An overview of methods to evaluate uncertainty of deterministic models in decision support,” Jan. 2015.
6. D. P. Loucks and E. van Beek, “System Sensitivity and Uncertainty Analysis,” in *Water Resource Systems Planning and Management*, pp. 331–374, Springer International Publishing, 2017.
7. S. H. Mallidi, T. Ogawa and H. Hermansky, “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., Feb. 2016.
8. P. Vincent, et al., “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” Dec. 2010.
9. G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, pp. 504–507, July 2006.
10. B. Lakshminarayanan, A. Pritzel and C. Blundell, “Simple and Scalable Predictive Uncertainty Estimation using Deep

Ensembles," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, pp. 6403–6414, Dec. 2016.
11. Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *ICML*, 2016.
12. P. Smolensky, *Information Processing in Dynamical Systems: Foundations of Harmony Theory*, pp. 194–281. Cambridge, MA, USA: MIT Press, 1986.
13. A. Fischer and C. Igel, "An introduction to restricted Boltzmann machines," in *Lecture Notes in Computer Science*, vol. 7441 LNCS, pp. 14–36, Springer, Berlin, Heidelberg, 2012.
14. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.
15. Y. LeCun, P. Haffner, L. Bottou and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 1681, pp. 319–345, Springer Verlag, 1999.
16. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997.
17. K. Cho, et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *EMNLP 2014 – Proceedings of the Conference*, pp. 1724–1734, Association for Computational Linguistics (ACL), June 2014.
18. A. Vaswani, "Attention Is All You Need arXiv:1706.03762v5," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017.
19. V. Edupuganti, M. Mardani, S. Vasanawala and J. Pauly, "Uncertainty Quantification in Deep MRI Reconstruction," Jan. 2019.
20. J. Chung, Ç. Gülçehre, K. Cho and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, 2014.
21. Y. Ma and J. C. Principe, "A taxonomy for neural memory networks," *IEEE transactions on neural networks and learning systems*, 2019.
22. C. Vallon and F. Borrelli, "Data-driven hierarchical predictive learning in unknown environments," 2020.
23. C. Voss, "Comparison of different novelty metrics for lstm networks in the domain of condition monitoring," Master's thesis, Helmut-Schmidt-Universität, Hamburg, 2020.

Bionotes

M. Sc. Samim Ahmad Multaheb
Institut für Automatisierungstechnik, Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg, Holstenhofweg 85, 22043 Hamburg, Germany
samim.multaheb@hsu-hh.de

Samim Ahmad Multaheb studied Mechanical Engineering at the Hamburg University of Applied Sciences. He received his Master's degree in Production Technology and Management from the same university in 2018. Before he has joined the Institute of Automation Technology at the Helmut Schmidt University in Hamburg, he worked in medical AI research. His current research is focused on finding underlying physical mechanisms in cyber-physical systems with deep learning.

M. Sc. Bernd Zimmering
Institut für Automatisierungstechnik, Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg, Holstenhofweg 85, 22043 Hamburg, Germany
bernd.zimmering@hsu-hh.de

Bernd Zimmering studied Automation Technology at the Hamburg University of Applied Sciences. After his graduation in 2017, he was employed in the beverages industry until he has joined the Institute of Automation Technology at the Helmut Schmidt University in Hamburg in 2020. His research fields are machine learning, as well as control & automation of production systems.

Prof. Dr. Oliver Niggemann
Institut für Automatisierungstechnik, Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg, Holstenhofweg 85, 22043 Hamburg, Germany
oliver.niggemann@hsu-hh.de

Prof. Oliver Niggemann did his doctorate in 2001 at the University of Paderborn. He then worked for seven years in leading positions in the industry. From 2008–2019 he had a professorship at the Institute for Industrial Information Technologies (inIT) in Lemgo/Germany. Until 2019 he was also deputy head of the Fraunhofer IOSB-INA which works in industrial automation. On April 1, 2019, he has taken over the university professorship "Computer Science in Mechanical Engineering" at the Helmut Schmidt University in Hamburg/Germany. There, he does research at the Institute for Automation Technology IfA in the field of artificial intelligence and machine learning for cyber-physical systems.