An Artificial Neural Network for Automated Fault Detection

Julian Bitterwolf
FZI Research Center for
Information Technology (FZI)
Karlsruhe, Germany
e-mail: bitterwo@fzi.de

Evgenia Rusak
FZI Research Center for
Information Technology (FZI)
Karlsruhe, Germany
e-mail: rusak@fzi.de

Sebastian Reiter
FZI Research Center for
Information Technology (FZI)
Karlsruhe, Germany
e-mail: sreiter@fzi.de

Alexander Viehl
FZI Research Center for
Information Technology (FZI)
Karlsruhe, Germany
e-mail: viehl@fzi.de

Oliver Bringmann
University of Tübingen
Tübingen, Germany
e-mail: bringman@informatik.uni-tuebingen.de

Abstract

Intelligent and interconnected cyber physical systems are a key enabler for future cost-efficient, automated and flexible industrial production systems. Predictive maintenance and condition monitoring are important techniques in order to reduce costs associated with unnecessary maintenance or premature breakdowns. In this paper, we propose techniques from supervised learning for automated malfunctioning detection. For that purpose, we train an artificial neural network on time series data representing the internal system behavior. We present experimental results from an industrial motor control system. We use a digital twin of the electronic component that models the relevant features of the physical system. The obtained information can be used during the runtime of technical systems and installations for a criticality analysis and the subsequent selection of measures.

1. Introduction

The prevalence of software-controlled functionalities in industrial systems is continuously increasing. The high degree of connections between system parts leads to a strong interconnection of the system. This means that in the worst case scenario, small faults of system parts can propagate to cause failures of the entire system. Condition monitoring as part of predictive maintenance serves to determine the condition of the overall system, system parts or components to notice subtle changes in relevant parameters in order to predict when maintenance or other measures should be performed to prevent failures. [1].

Proceedings of the 6th All-Russian Scientific Conference "Information Technologies for Intelligent Decision Making Support", May 28-31, Ufa - Stavropol, Russia, 2018

Predictive maintenance aims to ensure the maximum deployment efficiency of production systems and thereby save high costs of unnecessary maintenance or in the opposite case, costs associated with breakdowns

However, automated fault detection is a challenging task, as the system can react to different faults in different ways. Fault detection means the tracking of unexpected behavior of relevant parameters and the discovery of patterns in data that might indicate malfunctioning. Machine learning techniques are a common "working horse" for the task of pattern recognition. Several machine learning algorithms have been used for the task of predictive maintenance [2, 3].

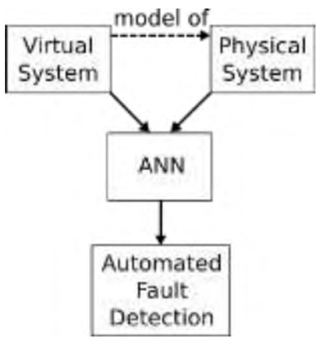

Fig. 1: Schematic of the proposed automated fault detection pipeline

Virtual prototyping (VP) has gained popularity as it allows for a validation of a new design during the development stage prior to building a physical prototype [4, 5]. In addition, VP is also used for maintenance purposes [6]. In this scenario, possible faults or degradation effects are simulated virtually to predict when maintenance should be provided to the physical system.

In this paper, we present a supervised machine learning algorithm for the task of automated fault detection. A schematic of the simulation and the automated fault detection pipeline is presented in Fig.1. The general idea is to train an artificial neural network (ANN) with labeled data that is obtained from a virtual system modeled according to a physical system and therefore being its digital twin. For the task of condition monitoring, different labels indicate different system states. Some system states contain faults and thus, the ANN learns to recognize when faults are present. After the training stage, the ANN can accept data from the physical system in order to track potentially dangerous faults. Our main conclusion is that the ANN is very successful in separating faulty from faultless cases, since we achieve an accuracy of 100% on the validation set after only training for 4 epochs.

The main contributions of this paper are:

- We demonstrate a methodology to generate an automated condition monitoring system from VP.
- We show that techniques from supervised machine learning serve as a valid method to distinguish between faulty and faultless cases and thus are suitable for the task of fault detection.
- We illustrate our methodology using an industry-relevant use-case.

The remainder of the paper is structured as follows: In Section 2, we discuss previous publications on the classification of time series data and the use of supervised machine learning algorithms for condition monitoring. In Section 3, we present our analysis approach for the employed methodology and time series classification. In Section 4, we describe our industrial use-case in more detail. Experimental results are provided in Section 5 and finally, the paper ends with a conclusion in Section 6. References are provided in Section 7.

2. Related Work

The data that is used for the fault detection algorithm has the nature of time series. For the tasks of condition monitoring and fault detection, time series classification has risen in popularity in the last decade. Time series analysis is a well-studied field [7]. In particular, time series classification has been studied extensively. Bagnall *et al.* presented a comprehensive review of different classification algorithms and their evaluation on publicly available data sets from the University of California Riverside time series classification archive (UCR) [8]. They have found that 1-NN DTW and Rotation Forest classifiers offer the best results in most cases. Despite these findings, we have used a shallow ANN as a classifier, because it was very successful for our use-case. In a recent publication, Wang *et al.* showed that deep learning can be used for time series classification [9]. The authors implement both a fully convolutional network and a very deep residual network and argue that with deep learning, heavy pre-processing or feature crafting is no longer necessary to achieve premium performance. They demonstrate their findings on the UCR data set.

Several authors have applied methods of time series classification for the task of condition monitoring and automated fault detection. Zhong *et al.* studied fault diagnosis for a gearbox based on Support Vector Machines for condition monitoring [10]. The data was obtained from an experimental test rig. Relevant features were extracted from the vibration signals of the gearbox with the wavelet packet transform as well as using time-statistical features. For an optimal set of features, the authors reported an accuracy of 100% on the test set. Shulian *et al.* presented an artificial neural network as a classifier for the task of fault diagnosis of a gearbox [11]. The data was obtained from a physical gearbox and faults were classified according to their severity such as e.g. 'gentle fault' or 'bad fault'. Aydin *et al.* presented a fuzzy c-means algorithm that was used to distinguish between broken rotor bar faults and the healthy condition of an induction motor at four different operation speeds [12]. Recently, Yang *et al.* showed a time-efficient clustering-k-nearest-neighbors algorithm for the purpose of fault detection in gas sensor arrays [13]. The authors verified the performance of the algorithm with a real gas sensor array experimental system with different kinds of faults. Campbell *et al.* examined the suitability of artificial neural networks to be used for condition monitoring of electric power transformers [14]. Sreejith *et al.* demonstrated an algorithm for the task of fault diagnosis of rolling element bearings using time-domain features and feedforward neural networks [15]. After the training stage, an accuracy of 100% to distinguish different states of the bearing was reported.

The references mentioned above were all evaluated with data obtained from physical experimental setups and made no use of VP. In general, most evaluations were performed on measured data. Li *et al.* used both VP as well as experimental studies for the task of gear multi-fault diagnosis [16]. The employed methods include the wavelet transform technique, Autoregressive models and Principal Component Analysis. With our work, we complement the existing research for automated fault detection in industrial systems.

3. Analysis Approach

3.1. Methodology

The automated fault detection framework was briefly introduced in the Introduction and Fig.1 and is explained here in more detail. An extended schematic of the automated fault detection pipeline is displayed in Fig. 2.

We differentiate between the Virtual Domain, the Analysis Domain and the Physical Domain. The Virtual Domain contains the virtual prototype and data obtained from it, while the Physical Domain contains the physical counterpart that is modelled by the VP. Data output from both the Virtual and the Physical Domains is analyzed in the Analysis Domain. The arrows indicate the directions of information flow between the various stages within the whole system. The domains and their interconnections will be explained in the following, starting with the Virtual Domain (left part of Fig.2).

The System under Test (SuT) and test bench configurations are defined for the stimulation of the VP within the test bench. Upon stimulation, the VP produces time series as output. Depending on the parameters defined in the SuT and test bench configurations, the resulting time series have different shapes. In half of the cases, a fault is injected during the simulation. The time series are then used as training data to train an artificial neural network (ANN) in the Analysis Domain. The ANN learns patterns in the time series and determines class labels to group similar time series into the same classes. The classes can either represent different motor states or, for the task of automated fault detection, different faults. Thus, after classification, a criticality analysis is performed to determine critical classes that indicate critical faults. The information obtained from the criticality analysis is used for decision making to conclude whether and which (safety) measures should be invoked to prevent system failures. These measures are then applied to the VP.

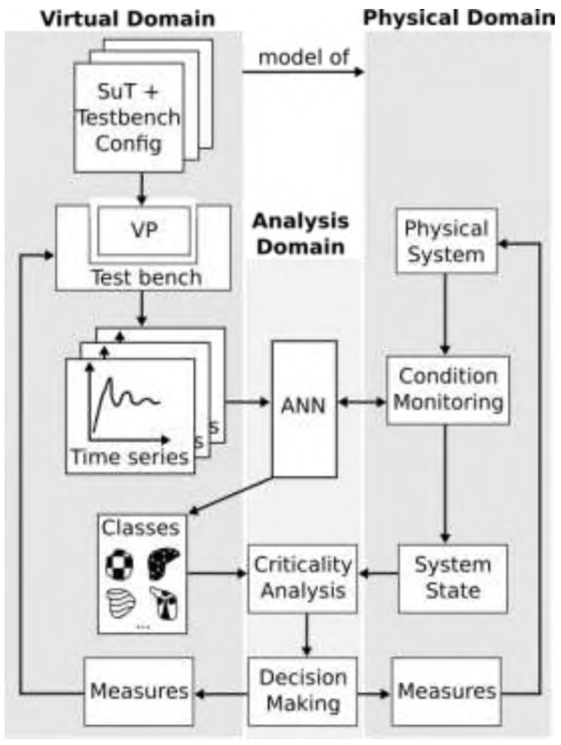

Fig. 2: Flowchart of the Virtual, Analysis and Physical Domains showing information flow and interconnections between the domains.

The proposed fault detection algorithm is evaluated on data obtained by the VP. The VP is a model of a physical system and the proposed algorithm can be used for condition monitoring in the Physical Domain (right part of Fig.2). The trained ANN can be used for the evaluation of time series data produced by the physical system. Analogously to the different classes in the Virtual Domain, a criticality analysis can be performed for the system state of the physical system. Based on the results of the criticality analysis, the subsequent decision making and safety measure stages are analogous to the Virtual Domain as described above.

The employment of VP for the simulation of different faults has several benefits. First, running a simulation is usually much faster than tracking the faults physically and for example, in case of degradation effects, gaining a sufficient amount of data in the physical world can take very long. Additionally, with the employment of VP, it is

```
def paa(ts, n):
  f = np.zeros(n)     #initialize s
  d = len(ts)//n      #slice length
  for i in range(0,n):
    slice = ts[i*d:(i+1)*d]
    f[i] = np.mean(slice)
  return f
```

possible to simulate many different faults and observe their propagation through the system. Some faults are very rare, so the simulation of all the possible fault manifestations allows for their discovery and prediction in the physical world before they actually appear for the first time. In the best case scenario, the simulation of faults makes it unnecessary to obtain a data base of possible faults in the physical world in the first place. In that case, unexpected behavior of physical machinery actually becomes ‘expected’ and certain measures can be invoked depending on the fault.

In this work, we focus on the first five stages in the Virtual and Analysis Domains. Our use-case is a SystemC-based VP of an industrial motor control. The employed SystemC framework is described in Ref. [17]. The motor control model and its parameters are described in Section 4 in more detail. The output of the simulation is the engine speed over time which is a sequence of discrete time data, and can therefore be treated as a time series. We focus on the easiest case of fault detection with only two classes: ‘Pass’ if no fault is injected and ‘Fail’ if a fault is injected. The resulting labeled engine speed data is then fed as input in the ANN that then distinguishes faulty and faultless classes. In the future, we intend to implement the next steps as displayed in Fig.2 such as the stages of criticality analysis, decision making and measures.

3.2. Framework for time series classification

The data that is used for the task of condition monitoring has the nature of time series. A time series is a sequence of observations that are arranged according to the time of their outcome [18]. In this work, we stick to univariate time series, which have a real numbered value at each point in time. A time series may be interesting as a whole, if it is the result of the observation of a finite-time process. Another category of time series are single series of ongoing or online data such as temperature curves or stock value charts. The questions that pose themselves for those different categories of time series are somewhat differing, and in this paper we focus on the first kind. Thus, in the remainder of this paper, by 'time series' we specifically mean a univariate whole time series. We further assume that for a given problem, the data points are time series which all have the same length.

Since a time series consists of one numeric value per time step, the dimension of a space of possible time series is its length, which equals to the total duration multiplied by the temporal resolution. This number can easily become very large and make pointwise comparisons between time series very time-consuming. Furthermore, many classification algorithms require or benefit from each sample being a point in a relatively low dimensional space. Thus, while it is in principle possible to treat a time series as an element of a very high dimensional space, for many algorithms that try to 'understand' the nature of time series, it is necessary to reduce this dimension drastically. This can be done by extracting certain features from time series. Piecewise Aggregate Approximation (PAA) is a simple feature extraction method [19]. Given a time series `ts` as a `numpy` array, we extract `n` features with the following Python 3 implementation of PAA:

The features are the means of `n` disjoint slices that cover the original time series. The calculated features are returned in an array `f`.

After the feature extraction step, the data is fed into a shallow artificial neural network with one hidden layer. The architecture of the ANN is depicted in Fig. 3. The ANN has the structure *affine → ReLU → affine → softmax*. The softmax layer turns the score from the last affine layer into the probability $p(\text{'Fail'})$. Eventually, 'Fail' is predicted if $p(\text{'Fail'})$is larger than 50% and 'Pass' is predicted if $p(\text{'Fail'})$ is below 50%.

Our current analysis stops at this stage after the separation of faulty from faultless time series. To follow the flowchart presented in Fig.2, a criticality analysis would be the next step and we intend to implement it in the future and thereby extend our automated fault detection framework. In the current scenario, we only consider two classes of 'Pass' or 'Fail' and do not distinguish different kinds of faults. For the task of a criticality analysis, such a simple 'Pass'/'Fail' threshold does not suffice. Instead, it is necessary to sort different faults in different fault classes. Different faults exhibit different behavior and thus, affect the system differently. The purpose of the criticality analysis is to determine which faults are the most critical ones and therefore, which measures must be invoked as a reaction.

4. Use-Case

The use-case for the proposed time series classification approach is a motor control model. This model is implemented in SystemC. A schematic of the simulated control process and the fault injection is displayed in Fig. 4. The test bench orders a programmable logic controller (PLC) when and how fast to run the motor. Given this information, the PLC transmits timed instructions to a transaction-level modelling queue, which provides them the motor control unit. The motor control unit now determines the duty cycle for the motor, i.e. what percentage of a fixed time period the motor should be powered. It thus conveys the number of 'on' and the number of 'off' time steps within a period to a pulse width modulator (PWM). The PWM applies power to the motor for the number of 'on' time steps and then deprives the motor of current for the number of 'off' steps. The PWM repeats this process until a new signal is received from motor control.

The output of the simulation is the motor's rotational speed over time, with subsequent time steps of 10^{-5}s between $t = 0$s and $t = 0.5$s. In Fig.5, an example for such a time series produced by specific engine parameters is shown. After 68ms, the motor is started and quickly overshoots to a speed of 305rpm at 74ms. Afterwards, the system oscillates until it levels out to a running speed of 197rpm. At $t = 267$ms, a load of 7Nm is attached to the motor. Subsequent to a short period of disturbance, the motor speed falls to its final plateau of 186rpm. For this example, we disabled the feedback to the motor controller and therefore the closed-loop control.

Fig.6 shows the same motor with identical parameters, but this time a fault is induced into the process: at $t = 208$ms, a transient bit error inside the PWM is simulated. This causes the PWM to operate on the basis of an overlong

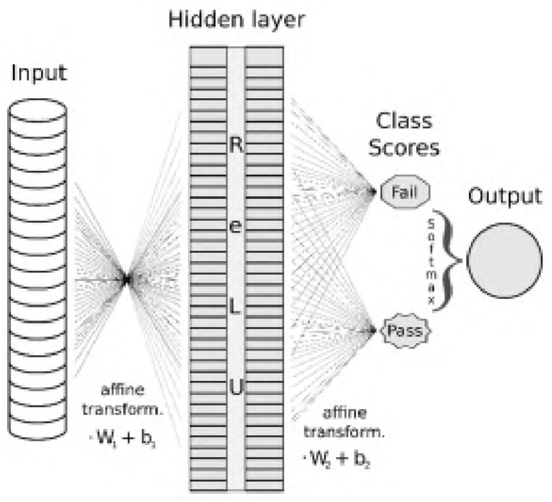

Fig. 3: Architecture of the considered artificial neural network.

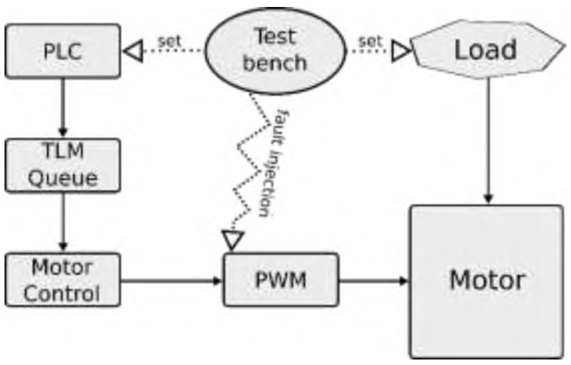

Fig. 4: Schematic of the simulated motor and the fault injection process

duty cycle. Aggregated over time, this means that it transmits an excessive amount of power to the motor. After 200ms, at $t = 407$ms, the bit error is corrected and the PWM is controlled with the correct value. The fault injection process results in an immediate disturbance of the speed curve and eventually the speed being increased by a value of 12rpm when running with no load and by a value of 16rpm while the load is present. Additionally, the oscillation process that occurs when the load is attached changes slightly in form.

5. Experimental Results

The full data set consists of 8000 time series similar to those displayed in Fig.5 and Fig.6. The motor parameters that are modified to get this number are:

- The supply voltage (UDC): steps of 1.4 V between 126 V and 154 V
- The motor's flux constant between 0.5 Vs and 0.6 Vs, step size of 0.005 Vs
- An amount of inertia to start the motor from 0.5 gm^2 to 1.5 gm^2 with steps of 0.1 gm^2

These parameter variations give a total of 4000 time series. For each parameter configuration, the simulation was run once without fault injection and once with an injected fault. For the cases where a fault was injected, it was done at a random time. This onset time for fault injection was drawn from a uniform distribution in the interval between 200ms and 300ms.

The structure of the data set thus contains 4000 tuples of time series with and without injected faults and is therefore very regular. This regularity could lead to problems of the learned classification heuristics of bad generalization to random and independent data. Similar concerns hold for the data generation part, where parameter spaces were discretized in regular grids. To break these symmetries, for each run, we only used a fraction of the data (500 time series for each) both for our training and validation sets. As in applications with real data there might be relatively few samples where a fault occurs, we chose a fault percentage of 20% for training. That means that from 500 time series used for training, 100 contain faults and 400 do not contain faults.

To reduce the samples' dimension of $l = 50000$ to a more manageable number, we define and extract a desired

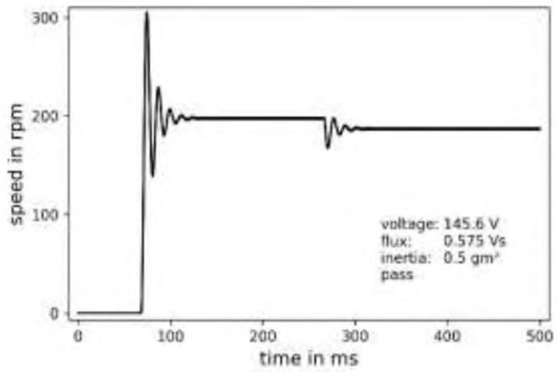

Fig. 5: Motor speed depending on the time without a fault being inserted ('Pass').

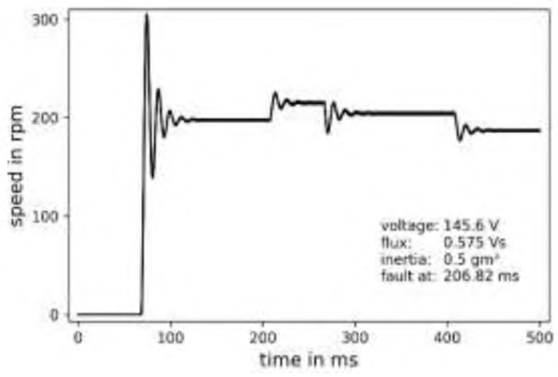

Fig. 6: Motor speed depending on the time with an inserted fault ('Fail').

number of features with the method of PAA described earlier. We use $f = 20$ as the number of features. Fig. 7 shows features 10 and 15 for 25 samples with an injected fault and for 119 samples without a fault. While for those time series without fault injection, there is a clear linear correlation between the two regarded features, the positions of the samples with an injected fault are more scattered.

The previously described ANN architecture was implemented using the publicly available software package PyTorch. The width of the hidden layer was set to 40. For backpropagation, the Adam optimizer that is implemented in PyTorch as `torch.optim.Adam` was used [20]. The learning rate was set to 0.005. In Fig. 8, the training loss is shown. The results on the validation set are flawless: after 4 training epochs, we reach an accuracy of 100% on the validation set. The accuracy on the validation set is displayed in Fig.9. The training time for 4 epochs takes less than one second, which can be attributed to the heavily reduced size of the training set due to the feature extraction. We have thus shown that a shallow ANN can learn meaningful patterns from features extracted with PAA and distinguish between faulty and faultless time series.

It is important to discuss the limitations of our approach. We considered the task of automated fault detection in the context of supervised learning. The disadvantage of this approach is that we have to rely on the availability of labeled data. As a next step, it would be interesting to tackle the automated fault detection problem with techniques from unsupervised learning. This way, a solution for cases when there is no labeled data could be offered.

6. Conclusion

We have demonstrated a methodology supporting automated fault detection using virtual prototyping. We have shown that methods from supervised machine learning are suitable for the task of automated fault detection. Based on the results from the automated fault detection, a criticality analysis can be performed to judge which measures must be triggered. We have evaluated our methods on data obtained from VP and, since the virtual prototype is a digital twin of a physical system, the trained network can be applied for condition monitoring of the physical machine.

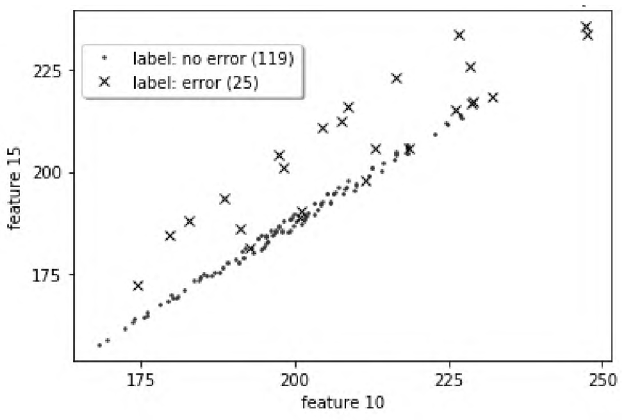

Fig. 7: Distribution of features 10 and 15 for 144 samples

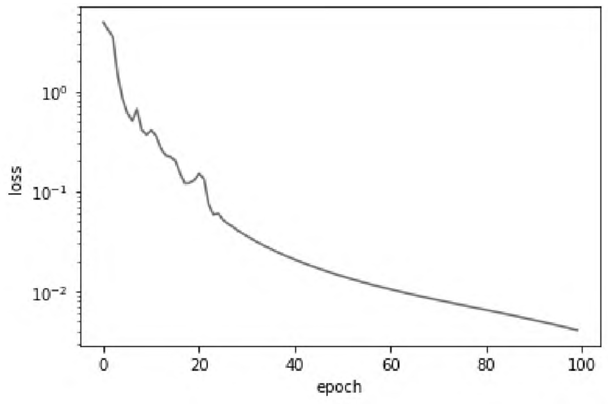

Fig. 8: Training loss for the training of the ANN

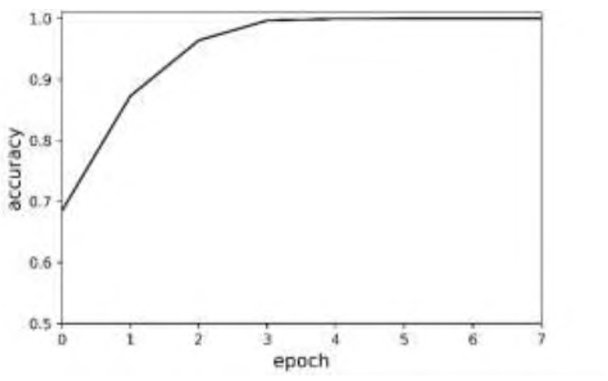

Fig. 9: Validation accuracy during the first training epochs of the ANN

8. Acknowledgements

This work has partially been supported by the German Ministry of Science and Education (BMBF) in the project SAFE4I under grant 01IS17032C.

9. References

1. An introduction to predictive maintenance. / Mobley, R. K. Butterworth-Heinemann, 2002.
2. Susto G.A., et al. Machine learning for predictive maintenance: A multiple classifier approach. // IEEE Trans. Ind. Informat. 2015. Vol 11 № 3 (2015)
3. Li H. et al. Improving rail network velocity: A machine learning approach to predictive maintenance. // Transp. Res. Part C Emerg. Technol. 2014. Vol 45.
4. Schaaf, J. C. et al. Systems Concept Development with Virtual Prototyping // Proc. of the 29th conference on Winter simulation 1997. P. 941–947.
5. Oetjens J-H., et al. Safety evaluation of automotive electronics using virtual prototypes: State of the art and research challenges. // Proc. of the 51st Annual Design Automation Conference. ACM, 2014.
6. De Sa, Gomes A., Zachmann G. Virtual reality as a tool for verification of assembly and maintenance processes. // CG 1999. Vol. 23 № 3 P. 389-403.
7. Fu T.C. A review on time series data mining // Eng. Appl. Artif. Intell. 2011. Vol. 24 P. 164-181
8. Bagnall, A. et al. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. // Data Min. Knowl. Discov. 2017. Vol. 31 № 3 P. 606-660.
9. Wang, Z., Yan W., Oates T. Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline. arXiv:1611.06455

10. Zhong J., Zhixin Y., Wong S.F. Machine condition monitoring and fault diagnosis based on support vector machine.// Industrial Engineering and Engineering Management (IEEM), 2010 .
11. Shulian Y. et al. Intelligent condition monitoring and fault diagnosis of a gearbox based on Artificial Neural Network. // Proc of the ICEMI'07, IEEE, 2007.
12. Aydin I.et.al. A simple and efficient method for fault diagnosis using time series data mining. // Proc. to IEMDC 2007, IEEE International
13. Yang J., Sun Z., Chen Y. Fault Detection Using the Clustering-kNN Rule for Gas Sensor Arrays. // Sensors (Basel) 2016. Vol. 12. P. 2069.
14. Campbell B., McDonald J.R. The use of artificial neural networks for condition monitoring of electrical power transformers // Neurocomputing 1998. Vol 23 № 1-3 (1998) P.97-109.
15. Sreejith B., Verma A. K., Srividya A. Fault diagnosis of rolling element bearing using time-domain features and neural networks // Proc.of the ICIIS'2008
16. Li, Z. et al. Virtual prototype and experimental research on gear multi-fault diagnosis using wavelet-autoregressive model and principal component analysis method // MSPP 2011. Vol 25 № 7
17. Reiter S. et al. Fault injection ecosystem for assisted safety validation of automotive systems. // High Level Design Validation and Test Workshop (HLDVT), 2016 IEEE International.
18. http://statistik.mathematik.uni-wuerzburg.de/fileadmin/10040800/user_upload/time_series/the_book/2012-August-01-times.pdf (дата обращения: 31.03.2018).
19. Yaodong Z., Glass J. A piecewise aggregate approximation lower-bound estimate for posteriorgram-based dynamic time warping. // Proc. of the 12th Annual Conference of the International Speech Communication Association. 2011.
20. Kingma D. P., Ba J. Adam: A method for Stochastic Optimization. https://arxiv.org/abs/1412.6980