



HAL
open science

Explainable Deep Learning for Fault Prognostics in Complex Systems: A Particle Accelerator Use-Case

Lukas Felsberger, Andrea Apollonio, Thomas Cartier-Michaud, Andreas Müller, Benjamin Todd, Dieter Kranzlmüller

► **To cite this version:**

Lukas Felsberger, Andrea Apollonio, Thomas Cartier-Michaud, Andreas Müller, Benjamin Todd, et al.. Explainable Deep Learning for Fault Prognostics in Complex Systems: A Particle Accelerator Use-Case. 4th International Cross-Domain Conference for Machine Learning and Knowledge Extraction (CD-MAKE), Aug 2020, Dublin, Ireland. pp.139-158, 10.1007/978-3-030-57321-8_8 . hal-03414728

HAL Id: hal-03414728

<https://inria.hal.science/hal-03414728v1>

Submitted on 4 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Explainable Deep Learning for Fault Prognostics in Complex Systems: A Particle Accelerator Use-Case^{*}

Lukas Felsberger^{1,3}, Andrea Apollonio³, Thomas Cartier-Michaud³, Andreas Müller², Benjamin Todd³, and Dieter Kranzlmüller¹

¹ MNM-Team, Ludwig-Maximilians-Universität Muenchen, Oettingenstr. 67,
D-80538 München, Germany

² Hochschule Darmstadt, Haardtring 100, 64295 Darmstadt, Germany

³ CERN, Route de Meyrin, 1211 Genève, Switzerland

Abstract. Sophisticated infrastructures often exhibit misbehaviour and failures resulting from complex interactions of their constituent subsystems. Such infrastructures use alarms, event and fault information, which is recorded to help diagnose and repair failure conditions by operations experts. This data can be analysed using explainable artificial intelligence to attempt to reveal precursors and eventual root causes. The proposed method is first applied to synthetic data in order to prove functionality. With synthetic data the framework makes extremely precise predictions and root causes can be identified correctly. Subsequently, the method is applied to real data from a complex particle accelerator system. In the real data setting, deep learning models produce accurate predictive models from less than ten error examples when precursors are captured. The approach described herein is a potentially valuable tool for operations experts to identify precursors in complex infrastructures.

Keywords: Prognostics and Diagnostics · Explainable AI · Deep Learning · Multivariate Time Series.

1 Introduction

Technical infrastructures are becoming increasingly complex while demands on availability are constantly rising. Failures of simple systems can often be manually analyzed in a relatively straight-forward manner by experts, whereas systems with increasing complexity and dependencies between infrastructures render manual failure analysis largely infeasible. This is primarily due to the sheer amount of potentially relevant failure precursors that must be considered.

In this study, modern particle accelerators are taken as example of complex infrastructures. Complete analytical models of failures and abnormal behaviors of particle accelerators are usually impossible; accelerators have numerous inter-operating subsystems, recording large amounts of diverse data, which

^{*} This work has been sponsored by the German Federal Ministry of Education and Research (grant no. 05E12CHA).

is difficult to analyse. In addition, the operational modes of particle accelerators can change over time, influencing the reliability and operating margins of sub-systems (e.g. an accelerator may operate as both an ion mode or a proton mode, with significant operating margins in one mode, and limited margins in the other). Moreover, an accelerator is a continuously evolving infrastructure, maintenance is carried out, sub-systems are upgraded and evolved, modes of operation are tuned, and adjusted. All combined, this makes traditional modelling approaches inadequate.

In general, operation data, such as system alarms, events, faults, physical measurements, etc., are abundant and are logged at rates and dimensionalities which are impossible to analyze in real time by a human operator. Hence, automated data driven analyses are required to assist human operators in diagnosis of system operation.

For application in particle accelerator environments, data driven methods need to handle heterogeneous data sources (e.g. binary alarms, discrete logged settings, continuous monitoring values), function with raw data and extract features automatically, learn from few observed failures and many observed non-failures (imbalanced data), and scale to several hundreds of input signals.

A data driven prognostics and diagnostics framework should reveal both (a) prediction of failures and alarms in advance and (b) clear insights for the interpretation of failure predictions. This would allow operators to increase infrastructure availability by preventive and pro-active actions to mitigate or remove failure conditions.

Related work Methods to predict faults have been reviewed extensively in the fields of prognostics and diagnostics [14, 34, 28], system health management [18] and predictive maintenance [26]. They are commonly classified as model driven, when a-priori modeling of the system behaviour is employed, or data driven, when the system behaviour is inferred from data. As model driven approaches are becoming infeasible when prior knowledge on the infrastructure behaviour is limited, only data driven methods are considered here.

Within data driven approaches, a distinction can be made between classical Machine Learning (ML) (e.g. support vector machine, k-nearest neighbour, decision tree), deep learning (e.g. deep belief networks, convolutional neural networks, recurrent neural networks) and probabilistic reasoning (e.g. Hidden Markov models, Gaussian Processes, Bayesian Graphical Networks) methods.

Further possible classifications are based on the field of application (e.g. mechanical systems, electronics, software), the complexity of the studied system (e.g. component, unit, system, system of system, human interaction), the type of learning (supervised, semi-supervised, unsupervised) and the kind of data used (univariate, multivariate, binary, numeric, discrete, text, raw data, features). Methods are usually not classified by the interpretability and explainability of their predictions, which is an important criterion for the considered use case and for many other applications [1]. In the following, related work will be grouped by choice of modeling approach.

Support-Vector Machines (SVMs) play a large role within traditional ML approaches. Zhu et al [35] and Fulp et al [13] developed SVM based methods to predict the failures of hard drives based on features indicating system health. Leahy et al [19] predicted wind turbine failures based on features of data from Supervisory Control and Data Acquisition (SCADA) systems in wind turbines. Fronza et al [12] extended the approach to systems of systems by predicting failures in large software systems. SVMs would allow interpretation of trained models, especially with linear Kernel functions. However, none of the authors considered investigating the structure of the learned models in order to gain insights into the failure mechanisms of the systems.

L1 regularized Granger causality was developed by Qiu et al [25] to detect root causes of anomalies in industrial processes. They reported interpretable results, scalability and robust performance. However, the method was not used to predict faults.

Association rule mining based methods were used by Vilalta et al [31] and Serio et al [30] to identify failure mechanisms in complex infrastructures using interpretable models. Vilalta et al detected anomalies in computer networks. The class imbalance problem is overcome by learning only from the minority class representing anomalies. They reported good accuracy but limited applicability of the method. The work by Serio et al represents the only relevant application in the particle accelerator domain. The authors successfully extracted expert verified fault dependencies between subsystems from logging data. However, time dependence between events were not considered and fault predictions are therefore not possible.

As an example of probabilistic reasoning methods, Mori et al [22] proposed a Bayesian graphical model approach to perform root cause diagnosis in industrial processes. The method is interpretable and accurate. At the overlap of probabilistic modeling and deep learning is the work of Liu et al [20]. The framework combines ideas from state space modeling with Restricted Boltzmann Machines or Deep Neural Networks to identify root causes of anomalies in industrial processes. High accuracy and scalability were reported.

With deep learning methods Saeki et al [27] classified wind turbine generator anomalies from spectral data. On a test data set, a visual explanation technique attributed importance to the same failure precursors as human experts. They noted that the data set used was not representative for real world scenarios. Amarsinghe et al [2] detected Denial of Service attacks in computer networks using a Deep Neural Network and a so called Layer-wise Relevance Propagation (LRP) introduced by Bach et al [4] to highlight relevant inputs for classification decisions. High classification accuracy was reported and explanations were intuitive to interpret. However, the method used features generated from raw data. Bach-Andersen et al [5] performed an extensive study of early fault precursor detection for ball bearings in wind turbines using raw spectral data. They compared logistic regression, fully connected neural networks and deep convolutional neural networks across three classification tasks. The deep network was found to perform the best. Using a visualization technique of higher layers of the trained

deep network, the strong performance of the network could be explained and insights about the failure behaviour were derived. The method yielded accurate results, handled class imbalance and scaled well.

Demonstrating performance advantage and universality, explainable deep learning frameworks seem promising for fault prediction in the accelerator domain. The work of Bach-Andersen et al provides a strong baseline but it was optimized for a different application domain and used a data structure which is not compatible with the particle accelerator use case. Moreover, the LRP mechanism by Bach et al was more intuitive to the authors of this study than the high level feature visualization used in Andersen et al. Furthermore, LRP is based on a more firmly established theory [21, 29]. A recent extensive review by Fawaz et al [10] on deep learning architectures for time series problems reveals that convolutional neural network structures outperform traditional methods across a variety of multivariate time series classification tasks. Similar findings were previously reported by Wang et al [32] for univariate time series. Therefore, we chose neural network architectures as suggested in Fawaz et al for failure prediction and LRP by Bach et al as the explanation mechanism for the framework presented in this study. The goal is to evaluate whether such a framework can successfully be applied as fault prognostics and diagnostics tool in the accelerator domain and whether it provides significant advantages over classical ML methods. The main contribution of this work is the first application of state-of-the-art deep learning for multivariate time series combined with explainable AI methods in the particle accelerator domain.




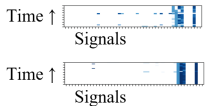


Data		Data Driven Model Prediction		Explanation
Input (image of animal) 	Label (species) cock hammerhead	Input 	Prediction Cock	
Input (past monitoring signals) Time ↑  Signals	Label (leading to alarm in future?) No Yes	Input 	Prediction Yes	

Fig. 1. Upper Row: Machine learning algorithms are able to identify animal species based on labeled images. Explanation techniques help to understand which pixels contribute the most to assign a certain species to an input image. [4] Lower Row: In the same way, a machine learning algorithm can learn a model to predict infrastructure failures based on time series data of monitoring signals. Here explanation techniques provide information about the most relevant signals leading to the failure.

The idea of the proposed framework is illustrated in Fig. 1. Time series of data are obtained during the operation of complex infrastructures, such as particle accelerators. Operational alarms or anomalies lead to specific fault events in the data. A sliding window approach can extract snapshots of the machine behaviour before the occurrence of fault events and snapshots during normal operation. Thereby, a supervised training data set is generated without manual labeling effort. From such data, a discriminator, such as a deep neural network, can learn general rules to predict when certain system faults occur. If such a fault is predicted, LRP highlights the most relevant fault precursors. This helps system experts understand the expected fault mechanism. With this knowledge they can take preventive measures to avoid the fault before it happens. For complex infrastructures effective use of such a method can help to reduce unexpected downtime and increase system availability.

For example, the algorithm could predict a critical failure leading to interruption of operations, such as power converter preventive shut down in a particle accelerator. Power converters supply precise electrical currents to magnets to focus and bend the particle beam. Without knowing the relevant precursors, system experts cannot identify how to prevent the shut down due to the sheer amount of potentially relevant monitoring signals. However, if the LRP highlights that the relevant signal is a noisy measurement of the particle beam position, the experts can simply replace the faulty beam position monitor. If successful, the power converter will not shut down due to erroneous beam position measurements and operations will not be interrupted.

The framework is introduced in Section 2. In Section 3 the effectiveness and the suitability of the framework are evaluated in experiments with synthetically generated time series data. Then, the method is applied to real world logging data sets of a particle accelerator to verify its suitability for the accelerator domain. Throughout, the deep learning frameworks are compared to classical machine learning methods, such as support-vector machines, random forests, and k-nearest-neighbor classifiers. Summary, Conclusions and Outlook are given in Section 4.

2 Methodology

Definitions and Overview The subject of study is an infrastructure \mathbf{I} which can be composed of multiple sub-systems. A range of N observable signals, monitors the behaviour of the infrastructure and its environment over time, forming a multivariate time series, $\mathbf{S} = \{\mathbf{S}_{i,t} : i \in [1 : N] \text{ and } t \in \mathbb{N}\}$. These can include logged continuous and discrete parameters, event- and alarm-logs, input- and output-signals, etc. The infrastructure has a range of failure modes which indicate certain malfunctions. These failure modes are observable in a subset of signals.

An autoregressive model predicts the future behaviour of a system based on its current and past states. For a complex infrastructure, it can be expected that failures may appear without announcing themselves in advance by precursors.

Even for situations with advance precursors, not all relevant processes might be monitored. Therefore, an autoregressive model can only approximate future failures based on time-discrete monitoring signals of complex infrastructures,

$$\mathbf{S}_{N,[t+t_p\delta t : t+(t_p+n_o)\delta t]}^F \approx \Phi(\mathbf{S}_{[1:N],[t-n_i\delta t : t]}^P)$$

with

- $\mathbf{S}_{N,[t+t_p\delta t : t+(t_p+n_o)\delta t]}^F = 1$, if a failure occurs between time $t + t_p\delta t$ and time $t + (t_p + n_o)\delta t$, and zero otherwise,
- $\mathbf{S}_{[1:N],[t-n_i\delta t : t]}^P$ being finite histories of observed signals covering the time stamps $t - n_i\delta t$ to t and being considered as possible precursors,
- δt being the discretization time,
- t_p the prediction- or lead-time,
- n_o the number of time steps chosen to capture the future failure behaviour,
- n_i the number of discrete time steps chosen to capture the history of the observed signals and
- Φ an auto-regressive model,

as illustrated in Fig. 2.

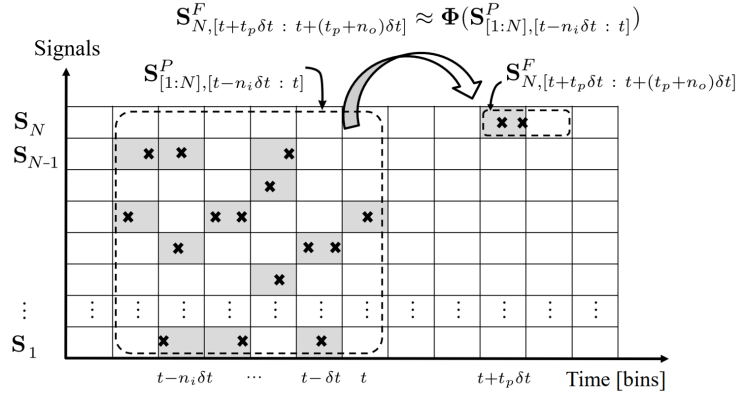


Fig. 2. Time discrete model formulation. The x-axis represents discrete time and the y-axis monitoring signals of the investigated infrastructure. Crosses mark events which could be faults, alarms, changes in monitoring values, etc. Events of the signal \mathbf{S}_N represent infrastructure faults that the model $\Phi(\cdot)$ predicts.

In a few simple cases, the model $\Phi(\cdot)$ can be obtained from first principles. In the case of complex infrastructure, the model needs to be determined in a data driven fashion using machine learning or time series analysis techniques. Models can learn the system behaviour from time series data in a supervised fashion by supplying pairs of input-data, $\mathbf{S}_{[1:N],[t-n_i\delta t : t]}^P$, and output- or target-data, $\mathbf{S}_{N,[t+t_p\delta t : t+(t_p+n_o)\delta t]}^F$, of the observed history of the studied infrastructure.

Once such a model has been trained, it can predict failures when supplied with observed data of the infrastructure. However, acting as black box it is not able to provide operators of the infrastructure any further information concerning the predicted failures. Such information could be used for actions to prevent the reoccurrence of the failure observed.

To address this, the framework provides a relevance measure, $\rho(\mathbf{S}^P) \in \mathbb{R}^{(n_i, N)}$, of each of the observed signals at each time step, which signifies the most relevant input signals for model predictions. This helps infrastructure operators to focus their attention on a small range of potential precursors. The framework does not search for causality but temporal precedence of correlated failure precursors [9]. However, it helps experts to verify causal dependencies and generate model based descriptions of causal failure behaviour [11].

The framework can be used as a real-time analysis tool, acquiring data continuously from the infrastructure as an input, predict imminent failures as an output. In this manner it could advise operators to take preventive measures. These actions require a certain lead-time, $t_p > 0$, to allow the system operators time to preemptively act on predicted failures. As a post-mortem tool, the framework could be used without lead-time constraints, in the analysis and explanation of complex failure mechanisms, which could then be investigated by system experts to mitigate reoccurrence. All of these considerations are particularly relevant for future generations of high-energy particle accelerators, for which the increasing size (order of 100 km), are expected to set unprecedented challenges in terms of maintainability of the infrastructures.

Machine Learning Pipeline This Subsection describes in more detail how predictive models $\Phi(\cdot)$ are learned from observed monitoring data.

Data Collection The observable signals \mathbf{S} of the studied infrastructure are stored in time series format in a data-set \mathcal{D} . Based on a-priori expert knowledge, a pre-selection of potentially relevant signals can reduce the required storage capacities. Further details on the data collection will be given for the use cases 3.

Model Selection and Evaluation The problem formulation contains a range of hyperparameters to be optimized, e.g. $[\delta T, n_i, n_o, t_p]$, some of which are introduced later in this Section. To do so, a K-fold validation strategy is adopted for the studied use-cases [7]. It is performed by splitting the overall data-set, \mathcal{D} , in a training set, \mathcal{D}_{train} up to time t_{split} , and a final test set, \mathcal{D}_{test} after time t_{split} . The K-folds for hyperparameter selection are obtained by a further splitting of the training set into K sub-training sets at splitting times $t_{sub-split, k}, k = 1, \dots, K$.

Subsampling As failures in the considered infrastructures are rare, the target data \mathbf{S}_N^F will contain few failures and many more '0's (no failures) than '1's (failures). The resulting class imbalance depends on the number of faults in the target variable, the discretization time δt , the number of time steps to capture

failure behaviour n_o , and the length of the time series available for training. Depending on the parameters, imbalances up to $1 : 10^4$ are obtained. Such a data-set requires balancing of the classes to ensure good performance of classifiers.

This is primarily achieved by randomly subsampling the majority class until a certain target ratio $p_{0,targ} = freq(cl_0)/freq(cl_1)$, is reached. Here, $freq(cl_x)$ denotes the number of class 'x' instances in the data set.⁴

More advanced sampling strategies were not tested in this initial study. They are often based on transformations of the data. Due to the complexity of the data structure suitable transformations could not be identified a priori and would have added optimization hyperparameters. Nevertheless, advanced subsampling strategies should be subject of future investigations.

After subsampling, data n_{cov} time-steps before and after each class '1' instance are added to the training set as it leads to better performance of the learned models. This can be seen as an oversampling method of class '0' to increase 'contrast' in the class '1' neighbourhood.

Choosing the output representation window length $n_o > 1$ also leads to an artificial oversampling of class '1' because a discretized fault can be captured n_o times within the representation window. As can be seen in the results Section, this can lead to improved classification performance at reduced time resolution.

Input Filtering and Normalization Input signals having values non-equal to zero less than α_{min} ⁵ times or having a variance smaller or equal to σ_{min} ⁶ are automatically removed. After the filtering, the input signals of the training data are normalized to the range $[0, 1]$.

Model Learning Algorithms To learn a model $\Phi(\cdot)$, from pairs of historical input- and output-data of the infrastructure, deep learning algorithms and classical machine learning algorithms were used. The target variables take values '0' and '1'. Hence, the failure forecasting problem is formulated as binary classification.

Based on recent studies of deep learning for multivariate time series classification, fully-convolutional neural networks are chosen as main classification architecture. They reach state-of-the-art performance while being faster to train than recurrent neural networks [10, 32]. These deep networks were compared against SVM, Random-Forest, and K-Nearest-Neighbour classifiers, representing classical ML techniques. Overall, the following algorithms were used:

⁴ Both the training and test data set are subsampled for computational performance reasons. Keeping the full data set would lead to slow out-of-memory-computations due to the size of the input data. Subsampling the test set is normally avoided as it could induce a bias in the generalization performance estimation [15]. However, tests with different subsampling ratios showed no influence on the generalization performance estimation for the studied scenarios. Hence, the subsampling of the test set does not bias the performance estimation but allows faster computation here.

⁵ Unless stated otherwise, $\alpha_{min}=4$, as it was the minimal number of examples from which the selected algorithms could learn from. [3]

⁶ Unless stated otherwise, $\sigma_{min}=0$, to only remove constant signals which do not contain any discriminatory information.

- FCN: The classifier is based on an architecture proposed by Wang et al [32]. It consists of three convolutional blocks with three operations in each: a convolution is followed by a batch normalization [17] and fed into a ReLU activation. The output of the last convolutional block is averaged over the whole time-dimension in a Global Average Pooling layer (GAP). Lastly, the GAP’s output is fully connected to a traditional softmax classifier. The convolutions are characterised by having a stride of 1 with zero padding to conserve the shape of the input data. The three convolution layers contain 128, 256, and 128 filters with a filter length of 8, 5, and 3, respectively. It was selected as it achieved the highest accuracy across 13 multivariate time series datasets in the review by Fawaz et al [10]. The implementation of the review paper is used with minor modifications. The number of training epochs is set to 2000. An early stop criterion ensures that training is terminated if the validation loss is not decreasing by more than 0.001 after 200 epochs, with the loss function set to categorical cross-entropy.
- FCN2drop: This is the same classifier as FCN, except with dropout applied to two layers in the network. Dropout is applied on the second convolution and on the GAP layer with a dropout probability of $p_{drop} = 0.5$. Due to the scarcity of class '1' items in the learning data, dropout regularization is expected to prevent overfitting of the network.
- FCN3drop: This is the same classifier as FCN, except with dropout applied to three layers in the network. Dropout is applied on the second and third convolution and on the GAP layer with $p_{drop} = 0.7$.
- tCNN: As proposed by Zhao et al [33], the network consists of two convolutional layers with 6 and 12 filters, respectively. The final layer is a traditional fully-connected layer with sigmoid activation function. It uses the mean-squared error instead of cross-entropy as loss function. The implementation from Fawaz et al is taken using the same early stopping criterion as for the FCN models.
- SVM: A support vector machine with linear kernel functions. The implementation from [24] is used with default parameters.
- RF: A random forest classifier is a meta classifier of many decision trees. The implementation from [24] is used with default parameters except for the number of features to consider for optimal splitting set to the square-root of the number of features.
- kNN: A k-Nearest-Neighbour classifier based on the implementation from [24] using default parameters except for $n = 7$ neighbours.

The classifier parameters were manually pre-selected based on recommendations in the Scikit-learn user guide [24] and preliminary experiments on data-sets as described in Section 3. SVM, RF and kNN classifiers require the input to be one-dimensional. Therefore, the 2D multivariate time series input data is flattened to one dimension when fed into the classifier for training and prediction.

To evaluate the performance of the classifiers, the accuracy and the F1 score on the test-set is examined. Due to the imbalance of classes, the accuracy might be misleading. Hence, additionally the fraction of the majority class in the test-set is reported for reference.

Explaining Predictions In order to help operation experts interpreting predictions of the framework and discovering failure mechanisms, the relevance of each input at discrete time steps in the observed history, $\rho(\mathbf{S}^P) \in \mathbb{R}^{(n_i, N)}$, is quantified and reported.

For deep learning methods, several such relevance reporting techniques have been developed. The so-called LRP was chosen for implementation as it provides best-in-class explanations [29]. It is based on a backward pass within the neural network which is layer-wise relevance conserving. Neurons contributing the most to the following layer receive most relevance from it during its backwards pass.

Testing the Gradient x Input [21], LRP-0, and LRP- ϵ rules [4], the LRP-0 rule showed a better omission of irrelevant failure precursors with synthetic test data.

For the classical machine learning methods, the input relevance for the SVM classifier was calculated by evaluating the input feature weight vector [24]. Input activations for kNN and RF are not calculated as these classifiers were solely used as classification performance benchmarks. The input relevance is depicted as 2D heatmaps with more relevant inputs being represented in darker colours.

The quality and usefulness of explanations for its users should be assessed as proposed in the literature (e.g. Holzinger et al [16]). Since the proposed method is currently a proof of concept, the quality of explanations could only be assessed by assuming the conditions of actual usage scenarios. This is carried out by qualitatively evaluating three criteria (inspired by [16]) for the use cases in Section 3: The completeness of the provided explanation factors, the ease of understanding, and the degree of causality within the studied processes that can be derived from the explanation. The timeliness of explanations is not discussed as it can always be ensured, depending on the choice of the prediction lead time t_p and the intended usage as either predictive or post-mortem explanation.

3 Numerical Experiments

The method described above is applied to several use-cases, which are briefly introduced. Implementations are available on github⁷.

Synthetic Data Experiments The framework is tested with synthetically generated data. In comparison to the real-world data-sets it has the advantage of a known ground truth. This allows to verify if the framework predicts faults accurately and identifies the correct failure precursors.⁸

Noise Robustness In this experiment, an infrastructure is modeled by n_{rand} systems firing precursors randomly and one system firing two consequent failure precursors which are always followed by an infrastructure failure S_s^F . The timing

⁷ <https://github.com/lfelsber/alarmsMining>

⁸ It has to be pointed out that the performance of the methods on synthetic data can not directly be generalized to real world data due to differences in data distributions.

of the alarms is illustrated in Fig. 3. The goal is to study the ability of the framework to filter and explain the deterministic pattern at increasing numbers n_{rand} of randomly firing systems despite being provided only less than ten failures to learn from.

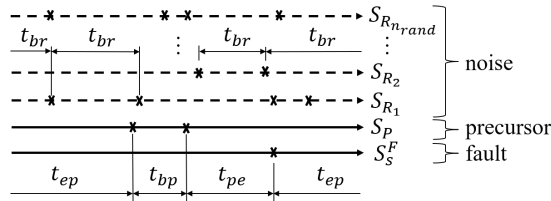


Fig. 3. Parameters of synthetic pattern.

The synthetic data is generated with a time $t_{br} \sim \mathcal{N}(\mu = 14.61d, \sigma = 14.61d)$ between randomly firing precursors, S_{R_l} , $l = 1, 2, \dots, n_{rand}$, a time $t_{bp} \sim \mathcal{N}(\mu = 1d, \sigma = 1d/24)$ between deterministic precursors S_p , a time $t_{pe} \sim \mathcal{N}(\mu = 10d/24, \sigma = 1d/24)$ between deterministic precursors S_p and infrastructure failures S_s^F , and a time $t_{ep} \sim \mathcal{N}(\mu = 36.525d, \sigma = 36.525d)$ between infrastructure failure S_s^F and deterministic precursors S_p with d being a day of 24 hours. It covers a time range of 2.7 years and n_{rand} being $[2^0, 2^1, \dots, 2^9]$.

The method is applied with sampling times $\delta t = [2h, 3h]$ (h for hours), an input range $n_i = 40$, a lead-time $t_p = 0$, an output range of $n_o = [1, 2, 3, 4]$, a sub-sampling target ratio $p_{0,targ} = 0.8$, and a class '1' neighbourhood coverage $n_{cov} = 2$. The data is split at times t_{split} chosen so that 80 percent of the data-set are used for training and model-selection and 20 percent for final testing. Training and model selection is performed by a 7-fold validation for sub-splitting times $t_{sub-split}$ chosen so that $[50, 55, 60, 65, 70, 75, 80]$ percent of the training data set are used for training and $[50, 45, 40, 35, 30, 25, 20]$ percent for validation. On average 7 (13) infrastructure failures were in the training data of the validation folds (the whole data-set). Hence, a small data scenario is investigated.

Of the 4480 trained models, the results for $\delta t = 3h$ and $n_o = 2$ led to good results for all classifiers and are presented. In Fig. 4a and 4b, the performance metrics are plotted as a function of the number of randomly firing signals, n_{rand} . Evidently, for higher numbers of random signals the performance is decreasing. Nevertheless, the correct patterns can be identified out of up to 100 random signals from as little as 7 examples in the training sets with the chosen parametrization. This suggests that in a real-data scenario, less than 10 training examples can be sufficient to detect failure patterns.

Further characteristics of the framework have been studied in [3]. It was found that increasing n_o by one or two can lead to accuracy improvements especially when the duration between precursors and failures has a high variance. The input relevance highlights the precursors with the lowest timing variance, and patterns are identified from as few as four failure examples.

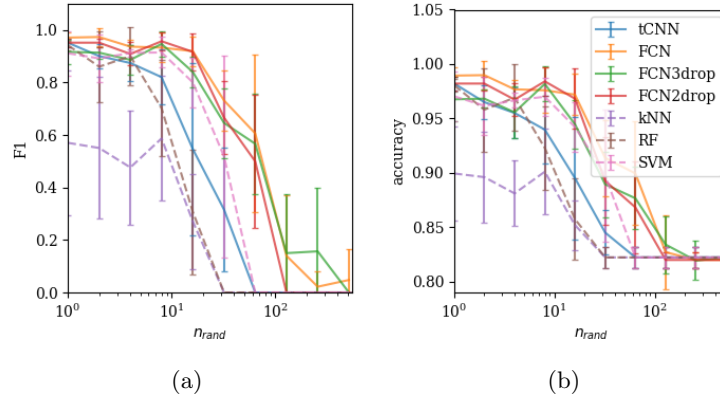


Fig. 4. Dependency of the predictive performance on the number of randomly firing signals. The line depicts the mean and the error bar plus and minus one standard deviation calculated over the 7 validation sets. Solid lines represent deep models which perform on average better than the classical models (dashed). (a) The F1 score. Note that the predictors level out at 0.0 for large n_{rand} , which is the binary F1 score when always predicting the majority class ('0'). (b) The accuracy. The predictors level out at 0.82 for large n_{rand} , which is the accuracy when always predicting the majority class ('0').

Recovering Fault Tree Structure Often faults in infrastructures are due to the interaction of multiple sub-systems. This experiment tests if the framework is able to identify failures due to interaction of sub-systems and if it can be explained by a system operator.

To do so, synthetic data is generated by simulating multiple sub-system interactions leading to infrastructure failures as illustrated in Fig. 5. An infrastructure failure, S_b^F , occurs after two precursor signals, S_{P_1} and S_{P_2} , fulfill either a Boolean AND, OR, or XOR condition. Four additional noise signals, $S_{R_{1-4}}$, are added to simulate non-interacting parts of the infrastructure. Time delays between signals are chosen to represent realistic scenarios.

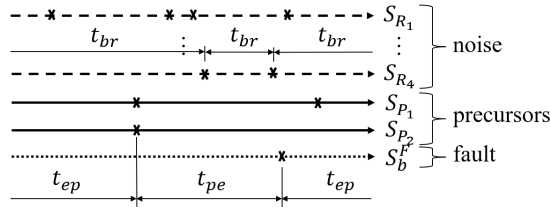


Fig. 5. Parameters of synthetic pattern.

The results for data generation parameters $t_{br} \sim \mathcal{N}(\mu = 23min, \sigma = 24min)$, $t_{pe} = 71min$, $t_{ep} = 120min$ are shown in Fig. 6. The framework is applied with sampling time $\delta t = 12min$ (*min* for minutes), an input range $n_i = 5$, a lead-time $t_p = 5$, an output range of $n_o = 1$, a sub-sampling target ratio $p_{0,targ} = 0.8$, and a class '1' neighbourhood coverage $n_{cov} = 2$. The data set was split in an equally sized training and testing set without additional K-fold validation as no hyperparameter selection was performed. The input data contain less than 20 examples of infrastructure failures. Deep networks and traditional ML algorithms perform well, consistently reaching $F1 > 0.97$. The results for the FCN2drop network are detailed below.

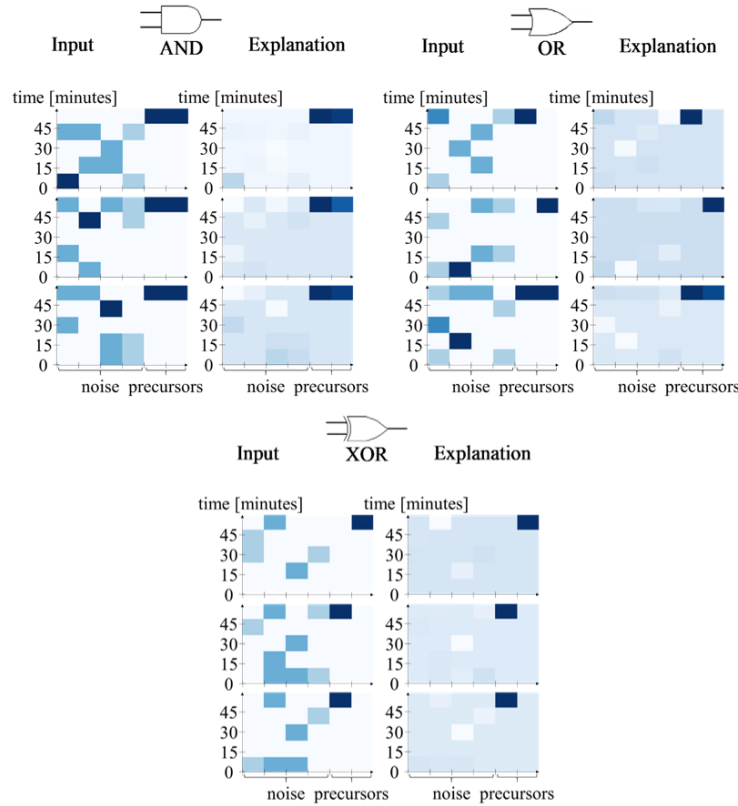


Fig. 6. Illustration of AND, OR and XOR fault logic extraction. Left columns show three randomly selected input windows before failure occurrence. Right columns show the relevant precursors obtained with the FCN2drop network (darker colours indicate higher relevance). Comparing the relevant precursors (right columns), allows to distinguish different Boolean rules and recover the fault logic of the system.

Fig. 6 shows three randomly selected input windows from the test data set with subsequent infrastructure failures for the AND, OR, and XOR scenario. Left columns show the unfiltered input window and right columns show the explanations with relevant signals as obtained from the FCN2drop network. Clearly, the correct precursors, S_{P_1} and S_{P_2} , could be identified by the framework and the noise, $S_{R_{1-4}}$, filtered. Comparing the groups of three input windows, the Boolean logic can be reconstructed. However, this relies on comparing different situations and is not possible from a single image. Still, the results confirm that the framework can identify failures due to interaction of multiple subsystems and allows system operators to explain the interactions. The inputs (left columns) would not expose the fault logic without the filtering by the explanation framework (right columns). Hence the explanation can be considered complete if different system behaviours are presented. The ease of understanding largely depends on the complexity of the studied system. For this simple case it is straightforward to disentangle different system behaviour. The causality cannot be assessed for this synthetic example.

Particle Accelerator Data Experiments Logging data from a particle accelerator infrastructure operated at CERN is used. The so-called Proton Synchrotron Booster (PSB) has a radius of 25 m and is composed of four superimposed rings [6]. It continuously logs failure modes, alarm data, operational settings data, physical monitoring and condition data, among others. The goal is to learn predictive models of failure modes and their mechanisms from data stored between January 2015 and December 2017. The learning task is difficult as the infrastructure is continuously worked on and modified, the data logging mechanisms have not been designed for historical data analysis, and failure data are rare [23]. As for any real-world infrastructure, only a subset of all relevant processes leading to failures can be observed. Furthermore, only system operators and experts can verify if the framework highlights the correct relevant precursors.

The goal is to forecast failures of eight power converters used in the PSB. The following signals are used for the analysis:

- Alarm logging data from the LASER alarm system [8]. The alarms are characterised by a system name, fault code and priority. The priority can be either 2 or 3 for the selected data. Priority 2 leads to a warning, whereas, priority 3 leads to a shutdown of the system. The ten most frequent priority 3 fault types of the 8 investigated power converters are selected as the target failure signals $\mathbf{S}_{O,F}$. The alarms are logged with a rising and falling flag indicating the beginning and the end of the alarm state, respectively. Only the rising flag is used as data. Data is grouped by system name for the input. This leads to eight input signals.
- Interlock signals register external and internal disturbances which potentially lead to the shut-down of the infrastructure. Based on operations experts' recommendations 27 signals were taken.

- The beam destination variable is an indicator of the operational mode of the PSB. The eight different beam destinations are hot encoded and added to the input data.

Time periods in which the PSB is switched off for maintenance were removed, as the alarm data is not valid during these periods of time.

Mixing Synthetic and Real Data In a first attempt, the goal is to test if a known pattern can be isolated from real-world data. Therefore, the noise robustness experiment was repeated with the randomly firing systems being replaced by real-world data containing 8 LASER alarm signals, 27 interlock signals, and 8 beam destination signals from the PSB.

The framework was applied to the data with sampling times $\delta t = [2h, 3h]$ (h for hours), an input range $n_i = 40$, a lead-time $t_p = 0$, an output range of $n_o = [1, 2, 3, 4]$, a sub-sampling target ratio $p_{0,targ} = 0.8$, and a class '1' neighbourhood coverage $n_{cov} = 2$. The same model validation strategy is chosen as for the synthetic data experiment. All classifiers were trained and evaluated.

Table 1. Performance metrics for mixing synthetic and real data experiments. **frac_{maj}** stands for the fraction of the majority class and is shown as reference for the accuracy of a trivial predictor always predicting the majority class. v and σ_v stand for the mean and standard deviation over the 7 validation folds, respectively, and t for results on the test set.

	FCN			FCN3drop			FCN2drop			tCNN			kNN			RF			SVM			frac maj		
	v	σ_v	t	v	σ_v	t	v	σ_v	t	v	σ_v	t	v	σ_v	t	v	σ_v	t	v	σ_v	t	v	σ_v	t
acc	0.97	0.03	0.98	0.93	0.06	0.95	0.97	0.02	0.98	0.83	0.03	0.95	0.84	0.03	0.87	0.83	0.03	0.89	0.91	0.05	0.94	0.83	0.03	0.89
F1	0.89	0.11	0.93	0.79	0.22	0.80	0.92	0.05	0.93	0.00	0.00	0.80	0.12	0.16	0.20	0.00	0.00	0.00	0.73	0.15	0.71			

Of the 448 trained models, the results for $\delta t = 3h$ and $n_o = 3$ achieved high F1 and accuracy and are presented in Table 1. The *FCN* networks show a strong performance in this experiment reaching F1 close to 1 based on only 7 training examples. This indicates that artificial patterns within the real-world data can be detected from less than ten examples. Fig. 7a shows the input activation for the FCN and SVM, respectively. Both correctly identify the relevant precursor out of 43 signals. As the explanation highlights a single precursor signal, it can be considered easy to understand. Completeness and causality cannot be assessed for this synthetic example.

Real Data The framework is tested to determine whether it can predict and explain critical priority 3 failures in the PSB accelerator power converters using the data introduced above. All signals are chosen as input data and the ten most active failure signals within the data set ($[\mathbf{S}_{F_0}, \dots, \mathbf{S}_{F_9}]$) are predicted.

The framework is applied with sampling times $\delta t = [10min, 30min, 2h]$ (h for hours, *min* for minutes), input ranges $n_i = [16, 32, 64]$, lead-times $t_p = [0, 1]$, output ranges of $n_o = [1, 2, 4, 16]$, a sub-sampling target ratio $p_{0,targ} = 0.8$, and

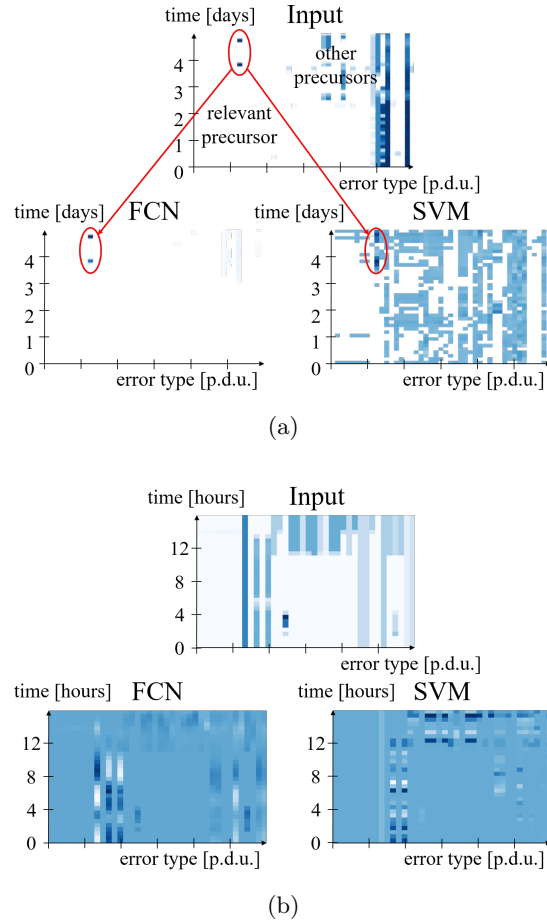


Fig. 7. (a) Upper: Input data for a single example of class '1' in the test set (not shown, occurring shortly after day 5). Lower left: Correctly identified relevant failure precursors by FCN network. Lower right: Correctly identified failure precursors by SVM network across all class '1' examples in the test set. Note that FCN and SVM evaluate non-relevant error messages differently in their models (darker colours in the heatmap signify higher relevance). (b) Input relevance for real data snapshot with $\delta t = 30min$, $n_i = 32$, $t_p = 0$, $n_o = 4$ and \mathbf{S}_{F_0} . The relevance in the upper region is higher for SVM. System experts could identify that certain combinations of external interlock signals and operational modes leading to infrastructure failures.

a class '1' neighbourhood coverage $n_{cov} = 2$. The same model validation strategy is chosen as for the synthetic data experiment.

Table 2. Performance metrics for real data experiments. **frac_{maj}** stands for the fraction of the majority class and is shown as reference for the accuracy of a trivial predictor always predicting the majority class. v and σ_v stand for the mean and standard deviation over the 7 validation folds, respectively, and t for results on the test set.

S_{Fi} n_i t_p	FCN			FCN3drop			FCN2drop			tCNN			kNN			RF			SVM			frac maj		
	v	σ_v	t	v	σ_v	t	v	σ_v	t	v	σ_v	t	v	σ_v	t	v	σ_v	t	v	σ_v	t	v	σ_v	t
acc 4 16 0	0.95	0.03	1.00	0.95	0.03	1.00	0.95	0.03	1.00	0.91	0.06	0.92	0.90	0.02	0.92	0.90	0.04	0.92	0.89	0.04	0.92	0.87	0.04	0.92
F1 4 16 0	0.81	0.12	1.00	0.84	0.09	1.00	0.84	0.09	1.00	0.36	0.50	0.00	0.68	0.10	0.67	0.35	0.33	0.00	0.13	0.30	0.67			
acc 7 16 0	0.95	0.03	1.00	0.95	0.03	0.83	0.92	0.06	0.96	0.87	0.03	0.92	0.91	0.03	0.92	0.87	0.03	0.96	0.89	0.04	0.96	0.86	0.03	0.92
F1 7 16 0	0.84	0.09	1.00	0.84	0.09	0.33	0.71	0.27	0.80	0.00	0.00	0.00	0.74	0.03	0.67	0.00	0.00	0.67	0.17	0.38	0.80			
acc 4 32 0	0.98	0.03	1.00	0.97	0.02	0.92	0.98	0.03	1.00	0.98	0.03	0.92	0.89	0.01	0.92	0.88	0.03	0.92	0.85	0.04	0.96	0.88	0.04	0.92
F1 4 32 0	0.94	0.09	1.00	0.90	0.05	0.00	0.92	0.11	1.00	0.93	0.10	0.00	0.57	0.09	0.67	0.18	0.25	0.00	0.33	0.10	0.80			
acc 7 32 0	0.97	0.03	1.00	0.95	0.03	0.92	0.97	0.02	0.92	0.93	0.05	0.92	0.87	0.04	0.96	0.88	0.03	0.92	0.85	0.04	1.00	0.86	0.03	0.92
F1 7 32 0	0.89	0.07	1.00	0.84	0.09	0.00	0.89	0.07	0.00	0.68	0.39	0.00	0.53	0.14	0.80	0.46	0.29	0.00	0.38	0.21	1.00			
acc 4 16 1	0.94	0.06	1.00	0.92	0.03	1.00	0.97	0.03	1.00	0.88	0.05	0.92	0.91	0.03	0.92	0.88	0.03	1.00	0.94	0.02	0.96	0.86	0.04	0.92
F1 4 16 1	0.75	0.26	1.00	0.74	0.06	1.00	0.89	0.10	1.00	0.16	0.36	0.00	0.74	0.06	0.67	0.08	0.18	1.00	0.66	0.24	0.80			
acc 7 16 1	0.95	0.01	1.00	0.91	0.03	1.00	0.96	0.01	1.00	0.86	0.03	0.91	0.91	0.03	0.96	0.86	0.03	0.91	0.94	0.02	1.00	0.83	0.00	0.91
F1 7 16 1	0.85	0.06	1.00	0.77	0.06	1.00	0.88	0.01	1.00	0.00	0.00	0.00	0.76	0.04	0.80	0.00	0.00	0.67	0.67	0.25	1.00			
acc 4 32 1	0.92	0.03	1.00	0.93	0.02	0.92	0.96	0.01	0.96	0.88	0.04	0.92	0.88	0.02	0.96	0.91	0.05	0.92	0.87	0.03	0.96	0.86	0.03	0.92
F1 4 32 1	0.62	0.18	1.00	0.68	0.19	0.00	0.84	0.06	0.80	0.16	0.36	0.00	0.55	0.12	0.80	0.72	0.18	0.00	0.40	0.15	0.80			
acc 7 32 1	0.92	0.03	1.00	0.96	0.01	0.91	0.97	0.02	0.96	0.88	0.04	0.91	0.86	0.05	0.96	0.89	0.05	0.91	0.84	0.04	1.00	0.84	0.02	0.91
F1 7 32 1	0.68	0.23	1.00	0.86	0.04	0.00	0.90	0.05	0.80	0.17	0.38	0.00	0.53	0.14	0.80	0.33	0.45	0.00	0.38	0.21	1.00			

Of the 11520 trained models, the results for $\delta t = 30min$, $n_i = [16, 32]$, $t_p = [0, 1]$, and $n_o = 4$ when predicting fault code S_{F_4} (malfunction of a power converter controller) and S_{F_7} (failure of a current measurement device) are presented in Table 2. It shows high accuracy for both on-line ($t_p = 1$) and post-mortem ($t_p = 0$) use.

The FCN networks show F1 close to 1. Note that the models were trained with as few as 17 class '1' examples on average for both the validation folds and the final training on the whole data-set. The F1 for the tCNN networks ranges from zero to close to one for different problem parameters. Its performance might be improved by additional tuning of the network parameters. The F1 of classical models is mostly smaller than 0.5 which is evidence that for infrastructures with discrete data deep networks outperform traditional machine learning classifiers. Applying dropout does not significantly improve the performance.

Overall, the framework demonstrates good performance in predicting system failures. However, only a subset of failures are predictable. Most likely this is due to insufficient observability of relevant processes within the logged data, which was neither conceived nor stored with the goal of using it for failure predictions. Furthermore, in all systems there are randomly occurring errors that do not necessarily have precursors.

An input relevance example is shown in Fig. 7b. The FCN filters less inputs as relevant than the SVM. Still both have comparable predictive performance. Analyzing the input relevance plots with system experts, the system behaviour could be recovered and non-trivial insights obtained. In terms of quality of explanations, it was concluded that the explanations are partially complete as

additional sources needed to be consulted. The ease of understanding dropped quickly with the number of highlighted signals in the input activation plots. The explanation helped to postulate causal chains but a degree of uncertainty remained. However, system experts confirmed that the input relevance provides useful insights for failure analysis, which gives confidence in the method and approach for future analyses.

4 Summary, Conclusions and Outlook

Our data driven framework identifies failure mechanisms in complex infrastructures, such as particle accelerators. Using multivariate time series data from infrastructure monitoring signals, a predictive model is learned with deep convolutional neural networks and classical machine learning algorithms. Explainable AI methods, such as layer wise relevance propagation, identify the most relevant failure precursors in the monitoring data. This has the potential to allow a more focused trouble-shooting of operational incidents.

The framework is applied to synthetic and real world data-sets. With synthetic data, the framework correctly isolates relevant failure precursors from up to hundred time series with as few as ten examples of failures to learn from and is able to recover interactions between multiple sub-systems. With real-world data, deep neural networks predict failures with F1 scores close to 1 for particle accelerator problems at a bare minimum of data pre-processing and problem-adaptation. Non-trivial system behaviour could be identified from the explanation mechanism. Fully Convolutional Neural Networks outperform classical ML methods in our experiments. Explainable deep learning proves to be a promising tool for future fault prognostics applications in particle accelerators.

For future research, the quality of explanations should be further surveyed in actual usage settings with system operators and experts. The experimental verification should be extended within and outside the particle accelerator domain. Since the framework does not rely upon specific insights from particle accelerators, it can be assumed that it performs well in other fields of application. To solve the limitations due to the lack of failure data, transfer learning and few-shot learning could be investigated. Changes in the infrastructure causing concept drifts could be tackled by re-learning approaches.

References

1. Abdul, A., Vermeulen, J., Wang, D., Lim, B.Y., Kankanhalli, M.: Trends and trajectories for explainable, accountable and intelligible systems: An hci research agenda. In: Proceedings of the 2018 CHI conference on human factors in computing systems. pp. 1–18 (2018)
2. Amarasinghe, K., Kenney, K., Manic, M.: Toward explainable deep neural network based anomaly detection. In: 2018 11th International Conference on Human System Interaction (HSI). pp. 311–317. IEEE (2018)

3. Apollonio, A., Cartier-Michaud, T., Felsberger, L., Müller, A., Todd, B.: Machine learning for early fault detection in accelerator systems (Jan 2020), <http://cds.cern.ch/record/2706483>
4. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS one* **10**(7) (2015)
5. Bach-Andersen, M., Rømer-Odgaard, B., Winther, O.: Deep learning for automated drivetrain fault detection. *Wind Energy* **21**(1), 29–41 (2018)
6. Benedikt, M., Blas, A., Borburgh, J.: The ps complex as proton pre-injector for the lhc-design and implementation report. Tech. rep., European Organization for Nuclear Research (2000)
7. Bergmeir, C., Benítez, J.M.: On the use of cross-validation for time series predictor evaluation. *Information Sciences* **191**, 192–213 (2012)
8. Calderini, F., Stapley, N., Tyrell, M., Pawlowski, B.: Moving towards a common alarm service for the lhc era. Tech. rep. (2003)
9. Eichler, M.: Causal inference with multiple time series: principles and problems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **371**(1997), 20110613 (2013)
10. Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery* **33**(4), 917–963 (2019)
11. Felsberger, L., Todd, B., Kranzlmüller, D.: Power converter maintenance optimization using a model-based digital reliability twin paradigm. In: 2019 4th International Conference on System Reliability and Safety (ICSRS). pp. 213–217. IEEE (2019)
12. Fronza, I., Sillitti, A., Succi, G., Terho, M., Vlasenko, J.: Failure prediction based on log files using random indexing and support vector machines. *Journal of Systems and Software* **86**(1), 2–11 (2013)
13. Fulp, E.W., Fink, G.A., Haack, J.N.: Predicting computer system failures using support vector machines. *WASL* **8**, 5–5 (2008)
14. Guo, J., Li, Z., Li, M.: A review on prognostics methods for engineering systems. *IEEE Transactions on Reliability* (2019)
15. Hastie, T., Tibshirani, R., Friedman, J.: *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media (2009)
16. Holzinger, A., Carrington, A., Müller, H.: Measuring the quality of explanations: the system causability scale (scs). *KI-Künstliche Intelligenz* pp. 1–6 (2020)
17. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015)
18. Khan, S., Yairi, T.: A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing* **107**, 241–265 (2018)
19. Leahy, K., Hu, R.L., Konstantakopoulos, I.C., Spanos, C.J., Agogino, A.M., O’Sullivan, D.T.: Diagnosing and predicting wind turbine faults from scada data using support vector machines. *International Journal of Prognostics and Health Management* **9**(1), 1–11 (2018)
20. Liu, C., Lore, K.G., Sarkar, S.: Data-driven root-cause analysis for distributed system anomalies. In: 2017 IEEE 56th Annual Conference on Decision and Control (CDC). pp. 5745–5750. IEEE (2017)
21. Montavon, G.: Gradient-based vs. propagation-based explanations: an axiomatic comparison. In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pp. 253–265. Springer (2019)

22. Mori, J., Mahalec, V., Yu, J.: Identification of probabilistic graphical network model for root-cause diagnosis in industrial processes. *Computers & chemical engineering* **71**, 171–209 (2014)
23. Niemi, A., Apollonio, A., Ponce, L., Todd, B., Walsh, D.J.: CERN Injector Complex Availability 2018 (Feb 2019), <https://cds.cern.ch/record/2655447>
24. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011)
25. Qiu, H., Liu, Y., Subrahmanya, N.A., Li, W.: Granger causality for time-series anomaly detection. In: 2012 IEEE 12th international conference on data mining. pp. 1074–1079. IEEE (2012)
26. Ran, Y., Zhou, X., Lin, P., Wen, Y., Deng, R.: A survey of predictive maintenance: Systems, purposes and approaches. arXiv preprint arXiv:1912.07383 (2019)
27. Saeki, M., Ogata, J., Murakawa, M., Ogawa, T.: Visual explanation of neural network based rotation machinery anomaly detection system. In: 2019 IEEE International Conference on Prognostics and Health Management (ICPHM). pp. 1–4. IEEE (2019)
28. Salfner, F., Lenk, M., Malek, M.: A survey of online failure prediction methods. *ACM Computing Surveys (CSUR)* **42**(3), 1–42 (2010)
29. Samek, W., Binder, A., Montavon, G., Lapuschkin, S., Müller, K.R.: Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems* **28**(11), 2660–2673 (2016)
30. Serio, L., Antonello, F., Baraldi, P., Castellano, A., Gentile, U., Zio, E.: A smart framework for the availability and reliability assessment and management of accelerators technical facilities. In: *Journal of Physics: Conference Series*. vol. 1067, p. 072029. IOP Publishing (2018)
31. Vilalta, R., Ma, S.: Predicting rare events in temporal domains. In: 2002 IEEE International Conference on Data Mining, 2002. Proceedings. pp. 474–481. IEEE (2002)
32. Wang, Z., Yan, W., Oates, T.: Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International joint conference on neural networks (IJCNN). pp. 1578–1585. IEEE (2017)
33. Zhao, B., Lu, H., Chen, S., Liu, J., Wu, D.: Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics* **28**(1), 162–169 (2017)
34. Zhao, G., Zhang, G., Ge, Q., Liu, X.: Research advances in fault diagnosis and prognostic based on deep learning. In: 2016 Prognostics and System Health Management Conference (PHM-Chengdu). pp. 1–6. IEEE (2016)
35. Zhu, B., Wang, G., Liu, X., Hu, D., Lin, S., Ma, J.: Proactive drive failure prediction for large scale storage systems. In: 2013 IEEE 29th symposium on mass storage systems and technologies (MSST). pp. 1–5. IEEE (2013)