Analysis of applicability of deep learning methods in compressor fault diagnosis

Anna Sztyber[1], **Łukasz Chechliński**[2], **Michał Syfert**[3], **Paweł Wnuk**[4], **Piotr Lipnicki**[5] and **Daniel Lewandowski**[6]
[1234]Warsaw University of Technology,
e-mail: [1]a.sztyber@mchtr.pw.edu.pl, [2]lukasz.chechlinski@gmail.com, [3]m.syfert@mchtr.pw.edu.pl, [4]p.wnuk@mchtr.pw.edu.pl
[56]ABB Corporate Research Center, Kraków, Poland
e-mail: [5]piotr.lipnicki@pl.abb.com

Abstract

The paper presents the results of work carried out on the applicability of deep learning techniques for the purpose of diagnostics of industrial rotary compressors. The paper focuses on the possibility of using the library TensorFlow by Google to build classifiers typical for this library, e.g. convolutional neural networks, as well as classical ones used in diagnostics, e.g. multilayer perceptron (MLP) or support vector machine (SVM). To provide a complete diagnostic tool was not the aim of the paper. Thus, only selected examplary faults were considered - dips of the power supply voltage and surge. At the beginning, a description of test stand, from which the test data were collected, is given. The main part of the work contains a description of the implementation of classifiers, and the results of their tests conducted on the actual measurement data. The data, registered during the experiments on site, represented both, the fault free state, as well the state with selected faults. Finally, the concept of the software (functionality and structure) dedicated for using considered techniques for both off-line tasks of building classifiers, as well as on-line monitoring in the cloud is discussed.

1 Introduction

Timely and accurate fault diagnosis is important for the performance of an industrial plant. There are many well developed model-based diagnostic techniques from the DX and FDI communities. On the other hand, in recent years, one observes rapid development of data driven and deep learning techniques, with successful applications in computer vision, machine translation and natural language processing [1]. The progress of deep learning algorithms is accompanied by the development of dedicated software like Tensorflow by Google.

The interesting question is: if and how can one apply some of these new techniques in an industrial fault diagnosis. Similar ideas were shown for example in [2; 3; 4].

The main aim of the described project was to find out if Tensorflow can be applied as a computational tool for industrial rotary compressors diagnosis. The paper is structured as follows: in Section 2 test stand is described and the considered faults are introduced in Section 3. Implementation of models for fault detection in Tensorflow and obtained results are presented respectively in Sections 4 and 5. Section 6 shows concept of a dedicated software for compressor diagnosis.

2 Test stand - industrial rotary compressors

The PLCRC Compressor Rig Test stand enables the testing and verification of different control, monitoring and protection algorithms in a closed environment [5]. In contrast to many standard experimental systems seen in research and academic laboratories using simple blow-off valves, the piping system of the PLCRC compressor system incorporates a hot recycle system. Its important property is the ability to truly reflect physical phenomena which are observed in industrial applications. The open loop installation receives air from ambient conditions, compresses and pumps the air to the discharge tank which models the volume of the connected pipeline. After that, the air can be redirected through the recycle valve back to the inlet of the installation which represents the hot recycle valve often encountered in practice. Potentially, there is also the possibility to add an additional blow-off valve to the test stand in order to investigate a more complex system. The P& ID diagram of the setup is presented in Figure 1 and the complete installation is shown in Figure 2.

The piping layout was designed such that the experiment may run in different operating conditions by opening and closing the inlet and outlet valves and switching between parallel and series operation of two compressors. Each of the compressors is equipped with fast recycle pneumatic valves (in case of surge occurrence). In addition, a selection of induction and switched reluctance motors fed by ABB variable-speed drives gave opportunity to control torque and speed of the machines. This design increased the degrees of freedom for the control system and was also able to work with a recycle line, in an arrangement more closely aligned with those seen in industrial systems.

The data acquisition was foreseen to be based on the AC500 controller with its dedicated I/O and communication modules. The control of the whole stand may be realized on AC800 PEC platform or AC500 High Performance PLC covering conventional anti-surge control, process or performance control and load sharing control.

The Compressor Rig test stand is treated as multipurpose experimental rig. The user can verify the developed control methods for each of the compressors running independently, having a parallel or series operation of both compres-

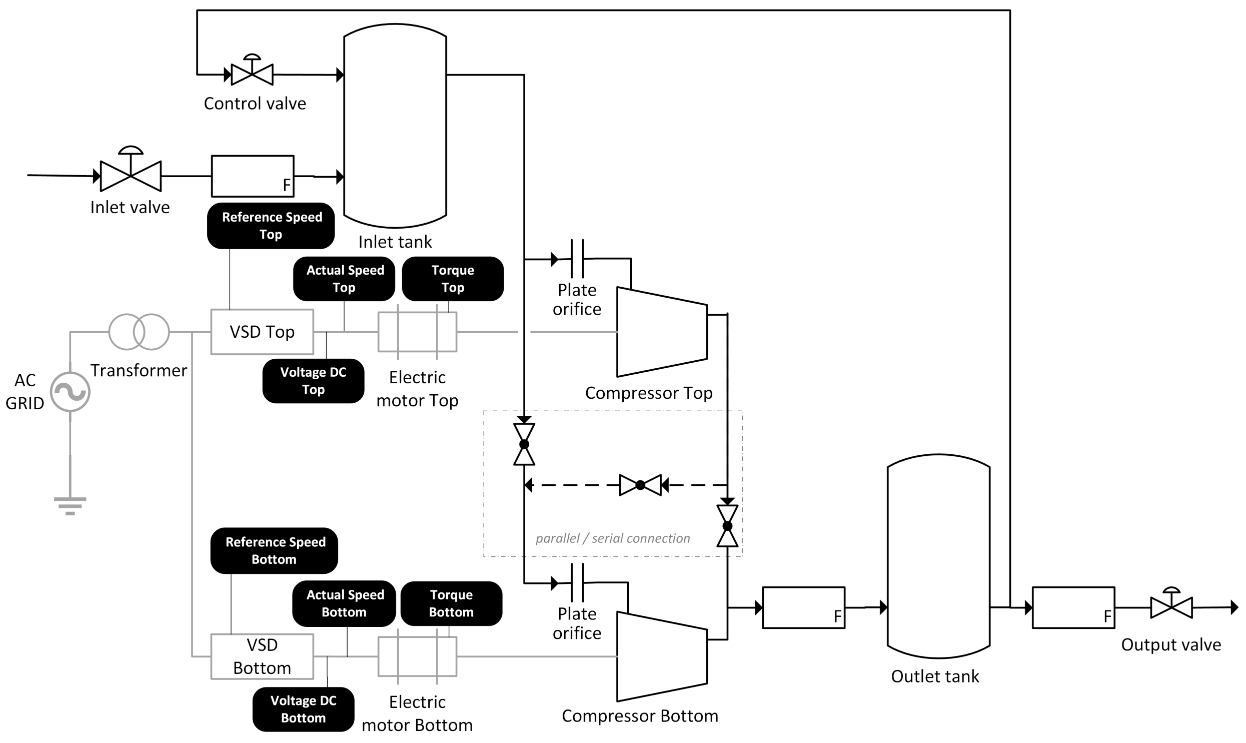


Figure 1: Compressor Rig P& ID diagram

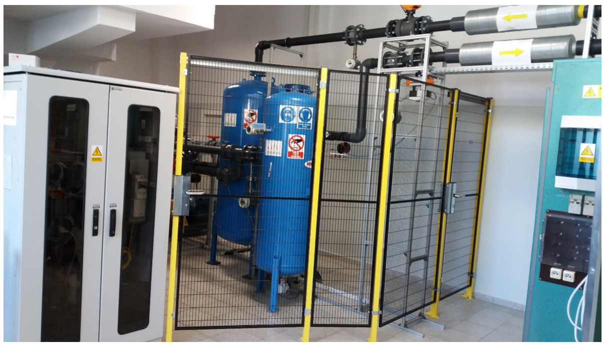


Figure 2: Compressor Rig test stand

sors running at the same time. The control may be realized on AC500 High Performance or AC800 PEC. The process measurements are collected and recorded by a communication hub, the AC500 PLC with I/O modules. Note, that all signals are integrated, i.e. electrical signals, process signals and mechanical signals can be recorded on the same hardware. This enables the analysis and online use of all data, leading to drive and compressor control integration. From the hub actual and reference values for the ACS880 and ACS850 drives can be read and send. The supervisory control of the stand is realized by java application, which allows, as well, data streaming and logging.

3 Faults description

The stand test enables the operation of a compressor system under various operating conditions and during the simulation of various types of faults, including faults related to power supply, process components, rotating machinery and control system.

The research presented in this paper focuses on selected two following faults:

- Voltage Dip - sudden, short-term (several dozen of ms, that means a few or a dozen samples) drops (a fall of several dozen or so percent) of three-phase power supply (nominal voltage is 400V). This type of voltage drop can not occur in normal operation, it is a result of electrical disturbances at the input to the system. Therefore, it is a fault from the group of power supply related faults. Changes in the input voltage also occur during the changes of the control signals. In such a case, the voltage drops should not be treated as faults. The important thing is, that the voltage drop is not measured directly by any of the available process signals (measurements);
- Surge - the occurrence of pressure oscillation (pulsation) in the process part, with a frequency of several Hz, often occurring together with the reverse flow. It is characterized by loud operation of the device and it is a highly destructive phenomenon. It occurs under specific operating conditions (flow, pressure before and after the compressor, rotation speed), therefore, it is necessary to analyze several process signals to detect it. It is a fault from the group of process faults. It is important, that the detection algorithms should distinguish oscillations of specific process signals resulting from the surge phenomenon, from the temporary oscillations associated with the change of working point and / or operating parameters that occur during transient states in order to avoid generating false alarms.

3.1 Signals selection

There are 46 measured signals available in the test stand. All of them are sent with a fixed sampling period equal to 2 ms. Those are both, fast-changing signals (power parameters and rotating systems), as well as slow-changing signals (e.g. temperature).

Based on the preliminary analysis and the expert's knowledge a subsets of signals were selected for the purpose of particular fault detection. Some signals, from the whole set of available ones, were excluded to reduce the dimensionality of the training data space. Some other signals coulnd not be used, because they were used to trigger a fault, e.g. surge was caused by the voltage reduction or flow strangling, so the usage of voltage signals for surge detection would cause in task simplification. Signals used for each fault are presented in Table 1.

Table 1: Signals used for detection of each fault

Signal name	VD	Surge
ActSpeedCompressorTop	✓	✓
ActSpeedCompressorBottom	✓	✓
ActTorqueCompressorTop	✓	✓
ActTorqueCompressorBottom	✓	✓
RefSpeedCompressorTop	✓	✓
RefSpeedCompressorBottom	✓	✓
ActVoltageDCLinkCompressorBottom	✓	
ActVoltageDCLinkCompressorTop	✓	

4 Implementation of classifiers using Tensorflow

The TensorFlow library [6] is an interface for designing machine learning algorithms, and an implementation for executing such algorithms. A computation expressed with the use of TensorFlow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems of hundreds of machines and thousands of computational devices such as GPU cards. It is focused on novel machine learning techniques known as deep learning [7]. It is not the only one in the domain. Examples of other frameworks are Caffe [8] or Theano [9]. An overview of deep learning frameworks can be found in [10].

Classifiers used in this work learns directly from data, no prior knowledge of the system model is needed. This effects in necessity to use the labelled training data, containing both normal process state and faults examples. For a given test stand this resulted in manual data labelling, noting when each fault begins and ends. Signals are saved in experimental data files, while each experiment is assigned to one of the fault categories (including no fault, i.e. fault free state). This means, that for each experiment only a single fault/no-fault label is needed, and the fault category is determined based on the experiment name. The example of the labelled surge fault is shown in Figure 3.

Detection of faults not present in the training data (in a sufficient quantity) cannot be robustly performed with the models presented below. However, futher work may adress this problem, like in [11], where triplet loss is used to classify face of the human not present in the training dataset.

Fault detection must be performed at the time when the fault occurs, so each sampling step is a new detection task. The border between normal and faulty state may be wider than a single sampling step, so some states should not be considered during training and testing in the future work.

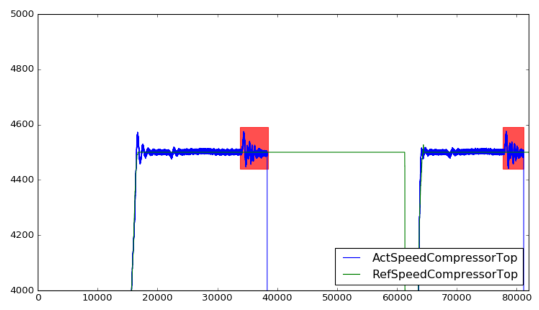


Figure 3: An example of the labelled surge fault, marked with a red background

For example, voltage dip start is rapid, but the surge end cannot be precisely determined.

Fault detection can be performed on the basis of signals values:

- from the current time step,
- from the current time step with some memory of previous steps,
- from last N time steps (explicitly, without internal model memory).

In this work cases above were implemented appropriately by the following models:

- Multi Layer Perceptron (MLP) and Support Vector Machine (SVM),
- Recurrent Neural Network with Long-Short Term Memory (LSTM),
- Convolutional Neural Network (CNN), MLP, SVM.

The larger signal period is considered by a classifier the more complex process model can it handle, but, or the other hand, more data is necessary for its training.

Results obtained for LSTM Network were not rewarding, so this case will not be described in details. All other models are described below.

4.1 SVM

Algorithm Support Vector Machine (SVM) serves for division of linearly-separable data by a hiperplane with maximal margin between classes. It can be applied to nonlinear problems using kernel trick. We selected this algorithm to test applicability of Tensorflow to non-neural classifiers.

To include time variability of process signals, classifier inputs can include values from previous time steps.

Two SVM classifier variants were tested:

(a) only current samples,

(b) current samples and three previous values for each signal: $x(k)$, $x(k-1)$, $x(k-2)$, $x(k-3)$. It should be noted, that many variants are possible (other time delays, signal decimation, etc.).

Due to the functionality of available estimator class only the linear version of the classifier was tested.

4.2 Multi Layer Perceptron

Multi Layer Perceptron (MLP) is the most popular type of artificial neural network. It contains input, output and several hidden layers. This type of network does not have recurrent connections. Each neuron of a given layer is connected with all neurons of the next layer and each connection has its individual weight. This network can model nonlinear functions, for more complex functions one needs more hidden layers.

Two variants were tested:

(a) only current samples (8 inputs),

(b) current and previous values of signals (32 inputs).

The network contains two hidden layers with respectively 100 and 50 neurons with nonlinear activation function $f(x) = \max(0, x)$ and one output neuron with sigmoidal activation. The model was implemented with high level Keras interface (`https://keras.io/`) and Tensorflow backend.

4.3 Convolutional Neural Network

Convolutional Neural Network is commonly used in the domain of image classification, where spatial relations between pixels must be considered. For signals with time relation Recurrent Neural Networks are used, as they are computationaly more efficient. However, signals used for fault detection are often analised by engineers as charts, so human can percept values from some period of time at once. This observation turned us to try CNN in fault detection domain based on time series analysis.

CNN takes as an input a tensor of sample length S_L x number of channels N_C. N_C depend on the number of signals N_S used for the fault diagnosis ($N_C = N_S$ or $N_C = 2 \cdot N_S$, explained below). S_L is a hyperparameter, which equals 12 (unit: probing steps) for voltage dips and 1200 for surge.

Moreover, human percept both signal value and its changes. Signal differending can be learned by the network from data. However, providing a simple preprocessing can speed up (less time and less train data) the training process, because the meaningfull variations in signal values are much smaller than the signal mean. Three cases were tested:

- $N_C = N_S$, only the raw signal is considered (later refered as *CNN V*),
- $N_C = 2 \cdot N_S$, where each raw signal is assited with its preprocessed changes signal (later refered as *CNN V+D*),
- $N_C = N_S$, only the preprocessed change signal (later refered as *CNN D*).

The change signal is calculated as follows: value from the first probe of the sample is subtracted from every probe value, and then, the magnitude is multiplied by a factor of F_C. It is a hyperparameter, which equals 20 for both fault types, which suit with observing the changes with magnitude of 5% of the signal range.

CNNs used for detection of both fault types have the same parametric structure, and differ only in values of those parameters. This parameters are:

- sample length S_L,
- Convolutional Channels Factor CCF,
- Layers Grouping Factor LGF

The CNN structure is described below:

- Network input goes through convolutional layers convX_Y (e.g. conv1_1, conv1_2, conv_2_1, conv2_2 etc.). X is the layer group index, while Y is the layer local index.
- Each convolutional layer convX_Y has $X \cdot CCF$ output channels and the output length equal to the input length.
- Number of layers in each layer group equals LGF. In other words, if each convolutional layer is noted as convX_Y, the following layers exists: convX_1, convX_2, ... convX_LGF.
- Each convolutional layer kernel size equals 3, and the convolution is followed with bias add and ReLU nonlinearity.
- After each layers group a max pooling operation is performed, reducing the output size by a factor of 2 – despite the layer, which output length is smaller then 5. This is the last convolutional layers group.
- Convolutional layers are followed with three fully connected layers, containing respectively 64, 16 and 2 neurons.
- Finally the softmax is calculated for the last fully conneted layer. Its values match model beliefs for fault and correct work.

The usage of the dropout regularization was tested, but it usually descreased results for only few percent.

Parameter values, for each fault type, are presented in Table 2.

Table 2: CNN parameters for each fault type

Fault type	S_L	CCF	LGF
Volatage Dips	12	12	2
Surge	1200	12	3

5 Results

The models described in Section 4 were trained and tested on the same subset of labelled files (experimental data files). The samples from the normal process state were randomly selected, so that the training set contained about 30% of faulty data (representing state with fault). The results are described in Table 3. The test data contains mostly non-faulty states, therefore, the accuracy is not a best metric. We use the following metrics to evaluate the models quality:

$$precision = \frac{TP}{TP + FP}, \tag{1}$$

$$recall = \frac{TP}{TP + FN}, \tag{2}$$

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}, \tag{3}$$

where: TP - number of true positives (correct detections), FP - number of false positives detections (false alarms), FN - number of false negatives (missed detections).

Best models for each type of fault are marked bold (Table 3).

Examples of models performance are shown in Figures 4-7. Legend for each figure is shown in Figure 4.

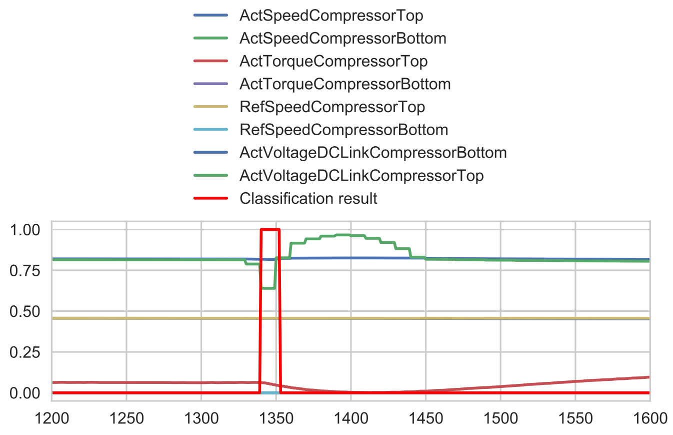


Figure 4: Voltage dips, SVM, good classification results for a simple case

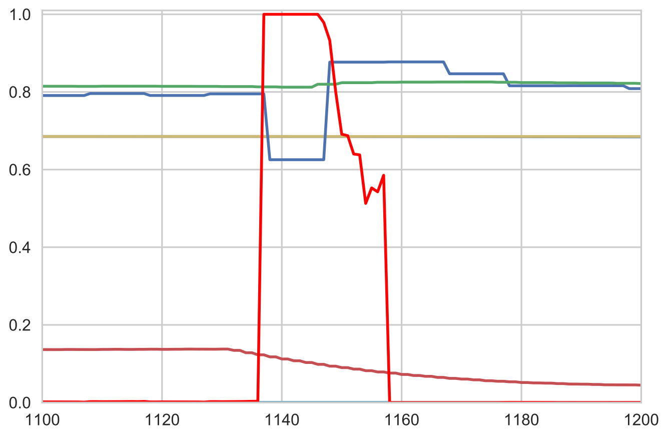


Figure 5: Voltage dips, CNN, delayed recovery from the faulty state

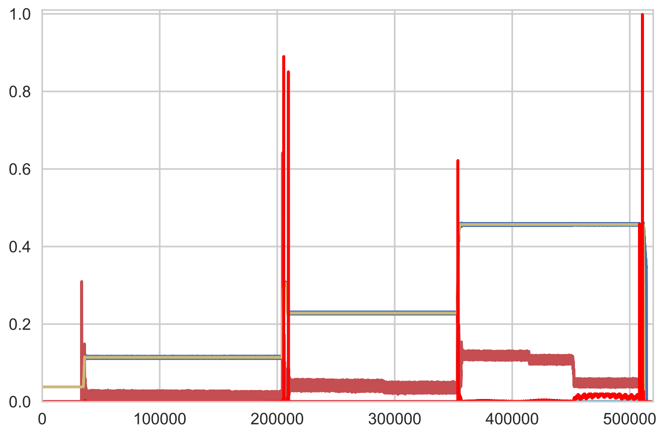


Figure 6: Voltage dips, CNN, false alarms in dynamic transient states

Table 3: Fault detection results

Model	*precision*	*recall*	F_1
VD, MLPa	0.843	0.522	0.645
VD, MLPb	0.537	0.384	0.448
VD, SVMa	0.101	0.700	0.176
VD, SVMb	0.239	0.752	0.363
VD, CNN V	0.168	0.446	0.244
VD, CNN V+D	0.246	0.829	0.379
VD, CNN D	0.028	0.350	0.052
Surge, CNN V	0.816	0.638	0.716
Surge, CNN V+D	0.927	0.630	0.750
Surge, CNN D	0.832	0.638	0.723
Surge, MLPa	0.682	0.546	0.606
Surge, MLPb	0.792	0.475	0.594
Surge, SVMa	0.200	1	0.333
Surge, SVMb	0.201	1	0.333

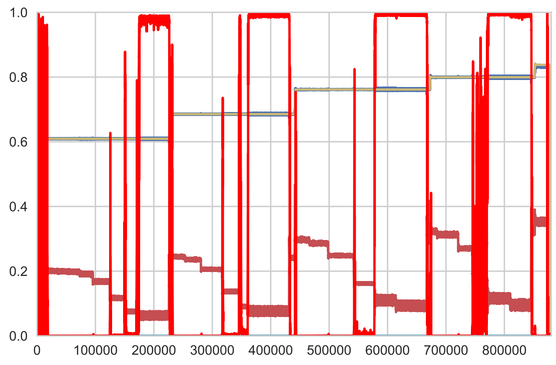


Figure 7: Surge, CNN, correct (but a little noisy) detection

5.1 Results summary

It should be noted, that the main aim of this project was not to prepare ready-to-use classifier, but to analyse the applicability of Tensorflow for such a task. Therefore, different methods were tested, but without fine tuning. The results could be further improved by:

- tuning networks structures (number of neurons, number of layers, number of input signals),
- meta-parameters tuning (learning rate, type of activation functions, type and parameters of optimization method),
- data preprocessing,
- post-processing - filtration of detection signals and thresholds selection.

Summarising, conducted tests show, that the Tensorflow can be used to build classifiers for the purpose od industrial fault diagnosis.

The results of tests of different classifiers can be summarised as follows:

- SVM classifier - Tensorflow libraries contain only linear version of this classifier, which has limited ability to represent complicated problems. We were able to build working classifier for voltage dips, but it is worse than neural networks. SVM classifier is only available in tf.contrib.estimator library, which is not a core library of Tensorflow. Therefore, for non-neural classifiers we recommend tests using other libraries and eventually final implementation with the use of low level Tensorflow functionalities.

- MLP network - for voltage dips this structure gives surprisingly good results even with only static data (without past values of signals). The speed of computations is also an advantage of this network. In the case of surge we need more past values, therefore this structure loses its advantages.
- CNN networks - 1-D convolution (filtration in a time domain) is a natural way of time series processing. Convolutional networks are recommended for processes with larger time spans, like surge. It is potentially possible to speed up computations by memorizing results from previous samples (similar idea for computer vision was presented in [12]).

The carried out tests show that the problems, i.e. false alarms, mainly occur in the following situations:

- when finding the exact moment when voltage dip ends - practically this is not a crucial issue,
- during startup and shutdown of the process some signals decrease rapidly causing false alarms - this can be filtered out by an additional logic in a diagnostic system,
- during dynamic state transients (caused short false alarms) - these can be partially filtered out, another solution is to increase the amount of data from dynamic states in the training set.

6 Concept of a dedicated software

This chapter presents the concept of dedicated software to implement algorithms for on-line diagnostics of compressors using the analyzed algorithms. The software will consist of the following two parts:

- an off-line part responsible for the synthesis of the detection algorithms and conducting learning phase,
- an on-line diagnostic part responsible for carrying out the current process state monitoring.

6.1 Diagnostic algorithm synthesis module

Whole detection algorithm synthesis module is designed as a typical off-line software dedicated to work on a classic PC's.

This part, as an input, analyzes sample sets of measurement data from the experiments, and, at the output, delivers: (a) labeled learning and test sets, (b) diagnostic classifiers.

The general structure of the module with marked general data flow is shown in Figure 8.

The individual components are responsible for:

- acquisition of experimental data from various operating states, including possible states with faults. New unmarked process data from experiments are saved by the standalone Learning Data Bridge in the Learning and Testing Database. This task can be also performed manually, by a diagnostics engineer or an ordinary operator. Even in this case, it is useful to develop a tool that simplifies saving the data file into the database.
- preparation of training data, including proper data labeling (marking a "presence" of fault). This operation is performed under the supervision of the diagnostic engineer with the use of the Labeler component. After entering the necessary information, the module creates the signals that describes the presence of fault in the learning data and stores it in the Learning and Testing Database. This tool should also enable simple manipulations on data sets such as partitioning, deletion of data or simple operations on signals. This module can also use pre-built classifiers stored in the Classifiers Database to perform an automatic pre-labeling test.
- conducting the procedure of selecting training and testing data as well as teaching models. With the help of the Modeling Module, the user carries out preparation of the training data (selection of training cases, data limitation, additional processing, etc.) and performs the appropriate process of identifying the classifier parameters (construction of the classifier). The obtained classifiers are saved in the Classifiers Database. The module must be able to use the TensorFlow library and Learning Algorithms provided by it. The learning procedure is usually supervised by the diagnostic engineer, however, the proper identification procedurte is conducted automatically.

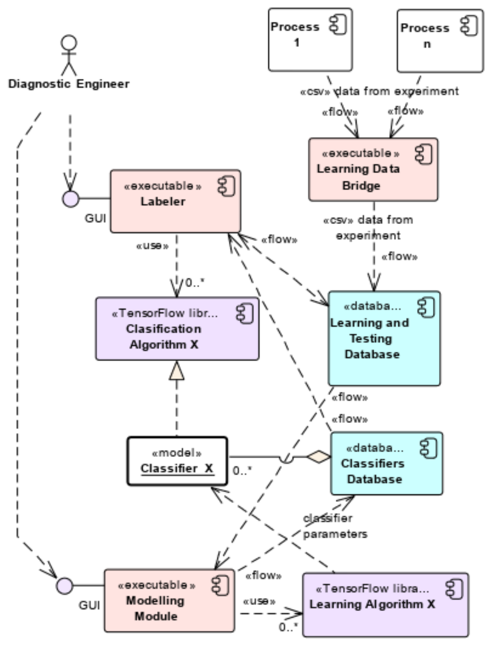
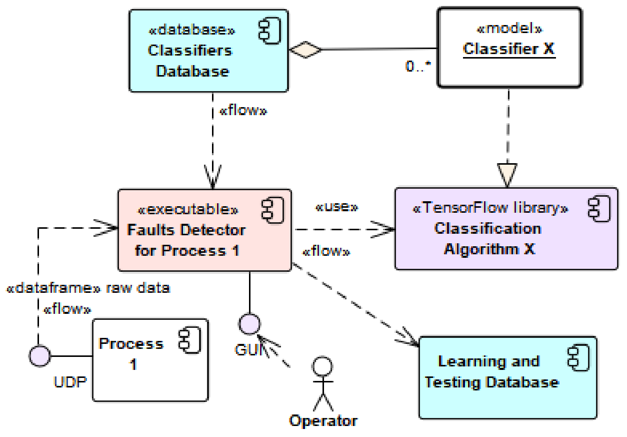

Process 1
Process n
Diagnostic Engineer
«csv» data from experiment
«flow»
«flow»
«executable» Learning Data Bridge
«executable» Labeler
GUI
«use»
0..*
«flow»
«csv» data from experiment
«flow»
«TensorFlow libr... Clasification Algorithm X
«databa... Learning and Testing Database
«flow»
«flow»
«model» Classifier X
0..*
«databa... Classifiers Database
classifier parameters
«flow»
«executable» Modelling Module
GUI
«use»
0..*
«TensorFlow libra... Learning Algorithm X

Figure 8: Components and it's relations - knowledge acquisition phase

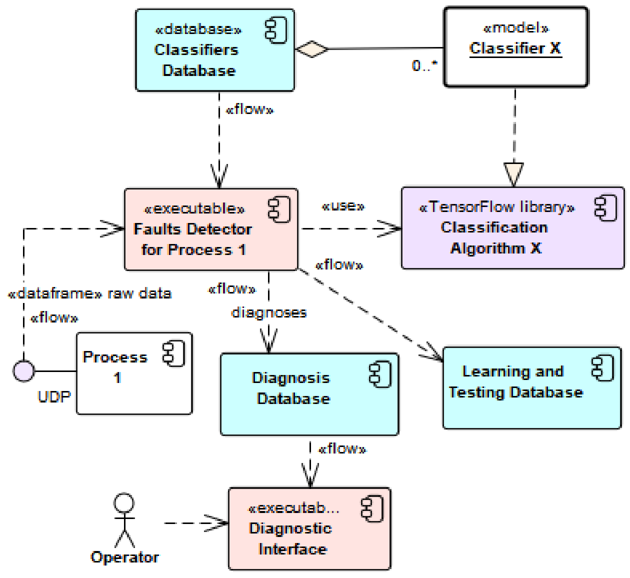
This module is designed to be a tool for diagnostic engineer. It support his work providing a convenient tools and GUI to prepare training data and to supervise the process of building classifiers. In the future, fully automatic operation for this module is foreseen, both, in the scope of data labeling (marking the data with fault labels), and in the learning phase (periodic training of classifiers when new training data becomes available).

6.2 Diagnostic module

The diagnostic part (current monitoring) is a typical software designed to operate on-line. This module analyzes new process dataframes and, at the output, determines the


«database» Classifiers Database
«model» Classifier X
0..*
«flow»
«executable» Faults Detector for Process 1
«use»
«flow»
«TensorFlow library» Classification Algorithm X
«dataframe» raw data
«flow»
Process 1
UDP
GUI
Operator
Learning and Testing Database

(a) with supervisor


«database» Classifiers Database
«model» Classifier X
0..*
«flow»
«executable» Faults Detector for Process 1
«use»
«TensorFlow library» Classification Algorithm X
«flow»
«dataframe» raw data
«flow»
«flow»
diagnoses
Process 1
UDP
Diagnosis Database
Learning and Testing Database
«flow»
Operator
«executab... Diagnostic Interface

(b) autonomous

Figure 9: Components and it's relations - monitoring phase

diagnosis considering system state (presence of faults). In addition, this module is responsible for distribution of elaborated diagnoses.

The general structure of the module together with the designed data flow is shown in Figure 9.

Conducted tasks by this module are as follows:

- acquisition of new process data. The task may be also completed with pre-processing of signals, e.g. aggregation or scaling.
- classification, i.e. the calculation of the outputs of diagnostic models (fault signals), and, as a consequence, generating a diagnoses about the state of the process. This task uses a set of available classifiers from the Clasifiers Database. The Fault Detector module uses the TensorFlow library to simulate Classification Algorithms. Due to the load balancing, data security and the use of independent communication channels for different objects, it is planned to create independent detectors for individual objects.
- distribution of diagnoses, i.e. implementation of a simple visualization on the built-in operator interface, the use of dedicated displays, as well as sending alarms to the control or supervision system.

This is a part that works essentially autonomously. In the basic version (Figure 9a), the fault detector will be equipped with a simple user interface used to display diagnoses. It will therefore combine both the detector functions and the operator interface. One can separate these functions by developing (Figure 9b):

- an autonomous module without a user interface responsible for the implementation and execution of diagnostic algorithms. The elaborated diagnoses will be stored in Diagnostic Database;
- an independent GUI for diagnostic module. It can be used for presentation of current process state as well as historical diagnoses.

In the future, it is planned to add an automatic or semi-automatic procedures to create new training and testing datasets. In such case, the module will also use the Training and Test Database.

Proposed architecture should be flexible enough to be implemented and run on different platforms, starting from control computers and ending with cloud systems.

7 Conclusions

The crucial issue is the process of collection and preparation of appropriate training and testing data. The quality of input data is essential to results. In this project, faults were labelled after the experiments. We recommend, when possible, to apply automatic registration of introduced faults during the experiments.

Regarding Tensorflow as a tool for fault detection we consider the following future work possibilities:

- more effective convolutional network implementation to speed up training and on-line calculations,
- selection of training data to include more dynamic transient states,
- experiments with recurrent neural networks,
- low-level implementation of selected non-neural classifiers,
- enlargement, preprocessing and careful labelling of training examples database, including methods of automatic labelling during experiments,
- research on including some compressor model in the classification process (instead of black-box approach).

According to the carried out experiments the deep learning techniques does not improve results in the diagnostic task. The explanation is simple: we can recognize the person on the image without red component of the image, but we cannot detect voltage dips without voltage signal. Simpler models, like MLP network, gives promising results. The simple structures have additional advantage - one can train a model on a CPU in a couple of minutes.

Typical approach to use Tensorflow library is based on raw data. It leads to the following conditions:

- one need a large amount of correctly labelled training data,
- no prior knowledge about the phenomenon is used.

These conclusions are consistent with the researched on deep learning conducted in other application areas. Deep learning techniques gain advantage with increasing amount of data. In case of smaller data sets classical approaches gives similar, or even better, results. Both approaches can be implemented in Tensorflow library.

In all engineering tasks one want to achieve satisfactory results with minimal amount of workload. Therefore, if one do not use simplified models of the process, he needs larger amount of training examples, so the model could learn how the process operate. It may be more efficient to build process model, implemented as Tensorflow graph itself, and use it for model based fault diagnosis.

To summarize, our test show that application of a Tensorflow library to compressor diagnostic can be justified, but the approach cannot be limited to standard deep learning techniques.

Acknowledgments

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[2] Monica Alexandru, Christophe Combastel, and Sylviane Gentil. Diagnostic decision using recurrent neural networks. *IFAC Proceedings Volumes*, 33(11):405–410, 2000.

[3] Belarmino Pulido, Jesus Maria Zamarreno, Alejandro Merino, Anibal Bregon, and Depto Ingenierıa Electromecánica. Using structural decomposition methods to design gray-box models for fault diagnosis of complex industrial systems: a beet sugar factory case study.

[4] Ran Zhang, Zhen Peng, Lifeng Wu, Beibei Yao, and Yong Guan. Fault diagnosis from raw sensor data using deep neural networks considering temporal coherence. *Sensors*, 17(3):549, 2017.

[5] Piotr Lipnicki, Daniel Lewandowski, Michał Kaczmarek, Andrea Cortinovis, and Diego Pareschi. Voltage dips influence on time to surge in compressor application. In Jan M. Kościelny, Michał Syfert, and Anna Sztyber, editors, *Advanced Solutions in Diagnostics and Fault Tolerant Control*, pages 347–356, Cham, 2018. Springer International Publishing.

[6] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[8] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[9] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 472:473, 2016.

[10] Soumith Chintala. An overview of deep learning frameworks and an introduction to pytorch. 2017.

[11] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[12] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.