Machine Learning Application in Predictive Maintenance

Karolis Liulys
Department of Electrical Engineering
Vilnius Gediminas Technical University
Naugarduko str. 41-413, LT-03227 Vilnius, Lithuania
karolis.liulys@gmail.com

***Abstract*—Industrial organizations worldwide cannot ignore Industry 4.0 and its impact to their businesses. The biggest struggle is to find the way how to adopt all the possibilities for each plants unique use cases. In those situations where it is hard to find unified solutions internet is playing major part. Inseparable part of Industry 4.0 is Internet of Things (IoT) paradigm, where it is possible to connect all devices into united system. While robust Distributed Control Systems (DCS) are preferred for their safety, Industrial IoT (IIoT) allows next level prospects: big data performance analyzation, control patterns identification and predictive preventative maintenance by using machine learning algorithms. The study shows how implementing open-source software enables engineers to develop predictive maintenance applications with basic programming knowledge. These type of applications can be widely used in industrial field to inform about possible equipment malfunction helping reduce possible damages.**

Keywords—Industry 4.0, IoT, preventive maintenance, machine learning

I. Introduction

The vast amount of products for global list of customers created large amounts of data which consists of people habits, provided services and processes information [1]. In order to increase equipment efficiency data gathering and analyzing became inevitable. A great need for companies to gather and store, previously seemly useless, information was the root cause of cloud computing transforming into the fourth industrial revolution [2]. Popularly referred as Industry 4.0.

Finally, internet has reached manufacturing processes and is challenging standard control model – hand written PLC code and process feedback signals interpretation by operators. Internet applications are changing this paradigm and IoT and is a key component of Industry 4.0. It not only does improve operations, but using the cloud environment, equipment and operations can be optimized by leveraging the insights of others. Research and development of processes can be done using less resources by relying on open-source applications, software and neural network models.

New developments in certain domains like mathematics and computer science (e.g. statistical learning) and availability of easy-to-use, often freely available (software) tools offer great potential to transform the manufacturing domain and their grasp on the increased manufacturing data repositories sustainably. One of the most exciting developments is in the area of machine learning. However, the field of machine learning is very diverse and many different algorithms, theories, and methods are available. For many manufacturing practitioners, this represents a barrier regarding the adoption of these powerful tools and thus may hinder the utilization of the vast amounts of data increasingly [3], but machine learning and neural networks applications in manufacturing shouldn't be overlooked.

Machine learning models can be implemented by using wide range open-source libraries like Tensorflow [4] and PyTorch [5] or by using problem directly targeted ensemble algorithms.

One of the most popular machine learning algorithms for big datasets analysis is gradient boosting (GB). GB on decision trees is a form of machine learning that works by progressively training more complex models to maximize the accuracy of predictions. Gradient boosting is particularly useful for predictive models that analyze ordered (continuous) data and categorical data [6].

Example of data analysis where trained machine learning models will recognize nonstandard variations in datasets will be showed further in this paper. Received results will be interpreted and forwarded to computerized maintenance management system (CMMS) through application program interface (API).

II. Materials And Preliminaties

Presented datasets were collected with Siemens DCS system SIMATIC WinCC v6.1. Main motor is controlled by SIEMENS G120 control unit (6SL3244-0BB13-1PA1). Speed feedback is received via PROFIBUS field connection protocol. System is closed-loop with encoder as feedback device. Data logging rate was set at 500 ms. Data results from Siemens controller CPU 412-5H are taken using NODE-RED programming tool via npm package - node-red-contrib-s7 [7]; machine learning library - tensorflow.js [8] is used for patterns recognition. RESTful API requests are made to IBM "MAXIMO" CMMS system.

Even though there are examples of machine lifelong working encoders which manage to reach 10-15 years [9] and manufacturers declare warranties up to 300 years [10], but in reality encoder's life span is reduced by external impacts. Main problem is bearings wear out caused by physical damages and hazardous environment (high temperatures and humidity). For encoder failure detection frequency inverters [11] have integrated encoder fault detection when speed difference has exceeded the preset value over several sampling cycles. This type of error is mostly used for internal troubleshooting.

978-1-7281-2499-5/19/$31.00 ©2019 IEEE

In Figure 1 closed-loop system behavior with declining encoder is presented.

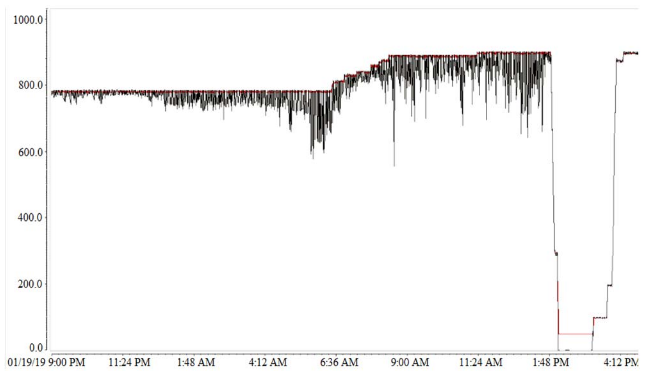

Fig. 1. Speed deviations caused by declining encoder performance.

Data statistics calculated in WinCC during different time ranges are given in Table I.

- Range 1 - 11:00 PM - 01:30 AM, where encoder damages are starting to show.
- Range 2 - 11:20 AM - 01:50 PM, where encoder's breakpoint is approaching.
- Range 3 - 04:40 PM - 06:40 PM, where the broken encoder is already replaced with a new one.

Standard deviation is calculated by formula [17]:

$$\delta = \sqrt{\frac{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2}{N(N-1)}}. \qquad (1)$$

Average value is calculated by formula:

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}. \qquad (2)$$

TABLE I. COLLECTED STATISTICS THROUGH DIFFERENT TIME RANGES

Range	*No. of data points*	*Standard deviation*	*Average value*
11:00 PM - 01:30 AM	18000	3.2	798,7
11:20 AM - 01:50 PM	18000	10.4	916.2
04:40 PM - 06:40 PM	14400	0.8	899,8

From Table I data we can see that speed decrease spikes in first time range don't significantly influence statistics. Standard deviation is small and average values are close to set point. Standard deviation in second time range is significant, but fast fall of performance doesn't create required time window for planned encoder replacement.

III. PROPOSED SYSTEM DESIGN

A. Traditional PLC approach

Traditional error detection models are implemented by monitoring difference between setpoint (SP) and actual process value (PV) in each sample time. In Fig.1 presented system behavior it is easy to calculate error between two coupled drives. To prevent "fake" alarm detection logic filters should be developed where alarm is set only when error threshold counter is overflowing specified value in predefined range. Example of function block, developed for Siemens PLC is presented in Fig 2.

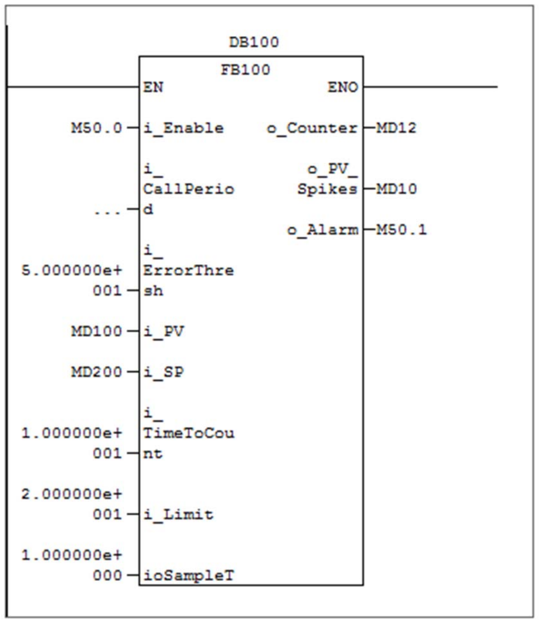

Fig. 2. Traditional PLC logic block with implemented error detection logic.

In Figure 2 function block is shown predetermined parameters are based on failure defined knowledge. Explanations of function block parameters:

- i_ErrorThreshold – after exceeding this value, error "spike" is taken into counter.
- i_PV – process value which is monitored. Main drive speed.
- i_SP – setpoint value. Speed value of the coupled motor.
- i_TimeToCount – time range to count error "spikes". When internal block timer reaches this value it is reset.
- i_Limit – error "spikes" counter value to generate alarm.

Presented function block design is widely used because of its simplicity.This type of logic can be helpful to prevent from failures repeating twice. That said its main flaw – parameters adjustment can be done only having failure describing information. This negates primary principle of preventative maintenance: detect faults before failure.

B. Machine learning models training

Neural networks (NN) have created possibilities to better analyze performance of systems which are hard to define by mathematical models. NN need for retraining after new information is given in predictive maintenance can be counted as advantage. After initial model is created with "high" and "low" performance inputs it can be held as epitome which can be modified with a few tweaks when new critical information is received.

For predictive models that analyze ordered and categorical data gradient boosting is especially useful technique. Gradient boosting is a technique in which the predictors are made sequentially. Using this technique subsequent predictors learn from the mistakes of the previous predictors. So the observations are not chosen based on the bootstrap process, but based on the error. Because new predictors are learning from mistakes committed by previous predictors, it takes less time/iterations to reach close to actual predictions.

Gradient boosting algorithm:

1. Assign every observation, x_i, an initial weight value $w_i = \frac{1}{n}$, where n is the total number of observations in the data.
2. Train a "weak" model on the data.
3. For each observation: if weak model predicts x_i correctly, w_i is increased. If weak model predicts x_i incorrectly, w_i is decreased.
4. Train a new weak model where observations with greater wi are given greater priority.
5. Repeat steps 4 and 5 until the data is perfectly predicted or a preset number of weak models has been trained.

For gradient boosting Catboost open-source model will be used.

Results after passing random data sets from different Table I ranges for prediction evaluation are presented in Table II. Model was trained with following parameters:

- Iterations = 10000;
- Learning rate = 0.1;
- Depth = 2;
- Loss function = MultiClass

TABLE II. PROBABILITIES FOR DATASET PREDICTION

Range	***Probability For range***
Range I	0.97191751
Range II	0.9120562
Range III	0.99875508

From Table II we can see that gradient boosting model can recognize precisely which range of dataset random sample of data represents.

As NN training data arrays of elements from WinCC data log were set as inputs with outputs "good", "warning" and "alarm" performance.

Results of NN training with activation function of sigmoid and leaky-reLU are given in Fig 3.

In practical model we are using model created by using leaky-reLU activation function. Primary leaky-reLU advantage inst sigmoid is stronger gradients, because data is centered around 0 - the derivatives are higher. Also leaky-reLU avoids bias in the gradients.

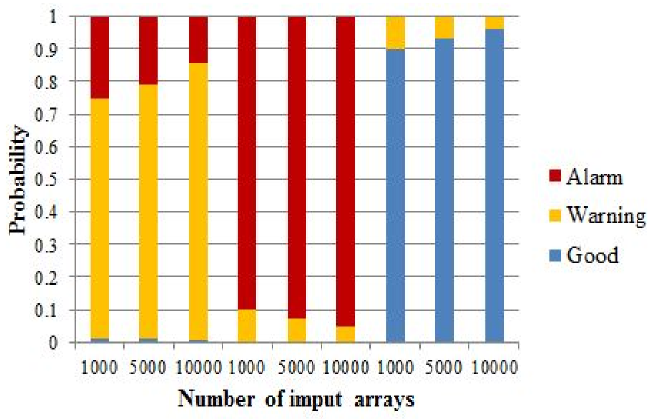

Fig. 3. Change of training error value during NN training cycle, when error threshold was set 0.0005.

Results, where test data is passed into created NN when model is trained with different number of input arrays, are given in Figure 4.

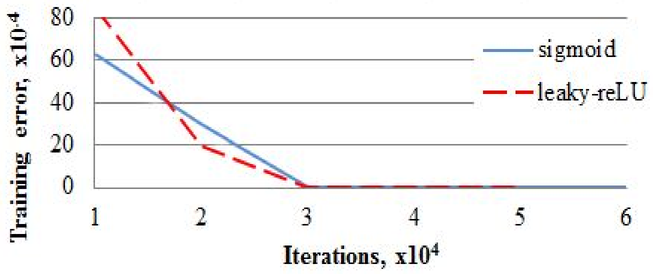

Fig. 4. Prediction possibilities in described data ranges depending on input numbers.

As seen from Figure 3 depending on inputs number passed into model, neural network with high possibility predicts correct states of the system. To increase predictions - more inputs should be inserted and model recalculated for lower error threshold. Learning rate can also be changed if system has several extremum points.

C. Designing API

We use REST API for direct data transfer to CMMS system IBM "Maximo" to create immediate service request for engineers to take action. Flow diagram how REST API handles requests in CMMS systems presented in Figure 5.

The REST API call flows through the authentication phase, API routes, authorization, and then it interacts with the systems artifacts: such as workflows and automation scripts. With native authentication the REST API expects the HTTP request with an AUTH request header and required asset id. Received JSON data, containing asset attributes, is used to create HTTP request with POST method. This request creates new record in CMMS system.

To create new service request after alarm appearance "gl acc read" node generates HTTP request with URL: "*http://USERID:PSW@SERVER_IP/maxrest/rest/mbo/asset?&_compact=true&_format=json&assetnum==XX&siteid=YY*", where USERID, PSW, SERVER_IP, XX,YY are corresponding user parameters. Message method *"GET"* is used.

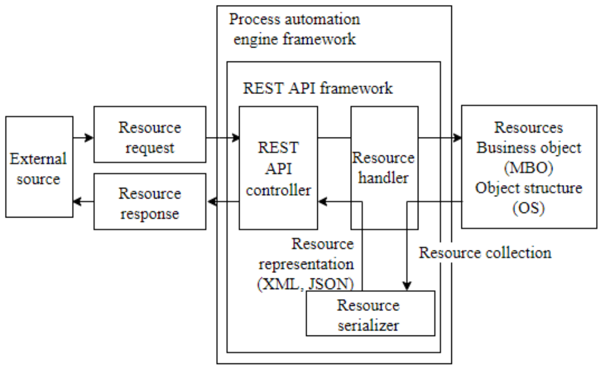

Fig. 5. Flow of REST API request handler.

Implementation of REST API in NODE-RED environment is shown in Figure 6.

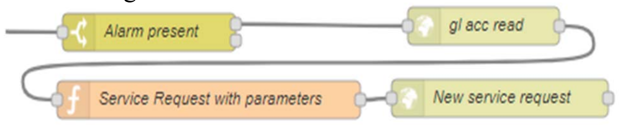

Fig. 6. REST API flow diagram in NODE-RED to create message.

Received response is interpreted into new HTTP URL using POST method. This URL contains all necessary information describing existing problem: asset ID, asset location, problem desription, etc. Code under node "Service Request with parameters":

```
GLACCOUNT=msg.payload.ASSETMboSet.ASSET[0].GLDEBITACCT;
location=msg.payload.ASSETMboSet.ASSET[0].LOCATION;
assetnum=msg.payload.ASSETMboSet.ASSET[0].ASSETNUM;

maximoIP = "SERVER_IP";
maximoUser = "USER";
maximoPassword = "PSW";
description = "Alarm description";
longdesc="Long description";
ORGID="GRIGISKES";

msg.url=encodeURI("http://" + maximoUser + ":" + maximoPassword + ""+
"@" + maximoIP +  "/maxrest/rest/os/mxsr?_action=AddChange" +
"&DESCRIPTION=" + description + "&location=" +location + "&asset"+
"num=" +assetnum +"&DESCRIPTION_LONGDESCRIPTION=" +longdesc
+
"&GLACCOUNT=" + GLACCOUNT + "&ASSETORGID=" + ORGID);
msg.method="POST";
return msg;
```

IV. Conclusions

Machine learning can be used in patterns recognition where it is hard to describe as mathematical model. It is particularly useful in PLCs where a lot of logic is written in "case" and "if...then...else" nature.

For predictive models gradient boosting is very useful technique, when big data model prediction improves while training several smaller ("weaker") models.

Study shows that leaky-reLU is faster activation function than sigmoid.

In neural network training greater set of inputs improve prediction accuracy. For repetitive patterns, saturation can be reached with smaller sets.

IoT technology enables RESTful API communication which helps machines without human interaction communicate with CMMS systems.

References

[1] Search Data Management [online], 2019. Big data : [cited 2 February 2019]. Available from internet: https://searchdatamanagement.techtarget.com/definition/big-dataI.

[2] Reliable Plant [online], 2019. 3 Factors Driving Industry 4.0 : [cited 4 February 2019]. Available from internet: https://www.reliableplant.com/Read/30933/factors-driving-industry

[3] Thorsten Wuest, Daniel Weimer, Christopher Irgens & Klaus-Dieter Thoben, "Machine learning in manufacturing: advantages, challenges, and applications". Production & Manufacturing Research, vol. 4:1, 2016 pp. 23-45, [cited 4 February 2019].

[4] Tensorflow [online], 2019. Getting started : [cited 2 February 2019]. Available from internet: https://www.tensorflow.org.

[5] PyTorch [online], 2019. GET STARTED: [cited 2 February 2019]. Available from internet: https://pytorch.org/get-started/locally/.

[6] Catboostb[online]. 2019. CatBoost Enables Fast Gradient Boosting on Decision Trees Using GPUs : [cited 2 February 2019]. Available from internet: https://catboost.ai/news/catboost-enables-fast-gradient-boosting-on-decision-trees-using-gpus.

[7] GitHub [online], 2019. Node-red-contrib-s7.: [cited 6 February 2019]. Available from internet: https://github.com/netsmarttech/node-red-contrib-s7#readme

[8] GitHub [online], 2019. tensorflow.js.: [cited 6 February 2019]. Available from internet: https://github.com/tensorflow/tfjs-examples

[9] R. Sawada, E. Higurashi, Y. Jin, "Hybrid Microlaser Encoder," J. Lightwave Technol. Vol. 21, 2003. [Online]. Available: https://www.osapublishing.org/jlt/abstract.cfm?URI=jlt-21-3-815

[10] DFS60 Incremental encoders: Online data sheet, Waldkirch, Germany, 2018.: [cited 1 February 2019]. Available from internet: https://www.sick.com/media/pdf/9/09/309/dataSheet_DFS60B-S4MA10000_1056180_en.pdf