Schedae Informaticae Vol. 26 (2017): 9–21
doi: 10.4467/20838476SI.17.002.7246

Failures Prediction Based on Performance Monitoring of a Gas Turbine: a Binary Classification Approach

Bartłomiej Mulewicz[1], Mateusz Marzec[1,2],
Paweł Morkisz[1,2], Piotr Oprocha[1,2]
[1]Reliability Solutions, ul. Lublańska 34, 31-476 Kraków, Poland
[2]AGH University of Science and Technology, Faculty of Applied Mathematics,
al. Mickiewicza 30, 30-059 Kraków, Poland
e-mail: *bartlomiej.mulewicz@relia-sol.pl*, *mateusz.marzec@relia-sol.pl*,
morkiszp@agh.edu.pl, *oprocha@agh.edu.pl*

Abstract. This paper is dedicated to employ novel technique of deep learning for machines failures prediction. General idea of how to transform sensor data into suitable data set for prediction is presented. Then, neural network architecture that is very successful in solving such problems is derived. Finally, we present a case study for real industrial data of a gas turbine, including results of the experiments.

Keywords: Predictive Maintenance, Deep Learning, binary classification, convolutional neural network

1. Introduction

One of key goals in production management is to maintain production line in service. Failures of machines in production lines can result in huge loss caused by not sufficient total production, waste of semi-products etc. It is one of main reasons why companies are interested in continuous improvements in their maintenance managements. The key feature is that sometimes it may cost less to make occasionally an unnecessary change of parts or perform some additional tests, rather than stop production for a longer time. In general, there are four main approaches to maintenance:

Received: June 26, 2017 / Accepted: September 7, 2017

- *Run-to-Failure (R2F):* in this approach any actions are performed after failure of some machine occurs. This is clearly the simplest possible approach. It is not obvious; however, that it is the cheapest one. First of all, it may generate high costs of production stop. Second thing is that unexpected failure can destroy some other parts of the machine, which in the process of well directed maintenance could be saved. On the other hand, there are types of problems where not much can be done before failure occurs;

- *Preventive Maintenance (PvM)*: in this approach maintenance actions are conducted according to some general schedule, based on expected lifetime of parts. This approach is based much more on statistical knowledge about machine and its parts, than everyday observation. Of course, when one part is replaced following the schedule, some additional parts might be replaced, if they are noticeably worn out;

- *Condition-based monitoring (CBM)*: in this approach, condition of the machinery is estimated based on sensor data (c.f. [1]). This is not limited to the sensors already available on the machine, but also includes complementary sensors, e.g. vibration, acoustic or oil analyses [2]. In this approach it is possible to quickly detect the failure in its early stage but it does not give any time to prepare for it.

- *Predictive Maintenance (PdM)*: in this approach, sensor measurements and knowledge about historical behavior of the machine are used in real time to predict necessary replacement of parts. In other words, we try to predict a failure before it occurs. For that purpose the holistic analysis of the data from the whole installation also adds significant value as some of the failures have its germs in other parts of the process and cannot be observed only by analyzing sensor data from the particular machine.

In recent years high development in electronics which resulted in huge increase of data related to performance and control of machines could be observed. That includes both the steering type data but also sensor data from the machinery built-in sensors or from complementary condition-based monitoring sensors. In PdM one of the goals is to successfully employ this data for prediction of oncoming failure. This data can be also used for assessment of degradation state of parts and better estimation of failure probability in PvM approach [3].

Motivation for the present research comes from partnership with a large chemical company which indicated a problem of a gas turbine repeatedly reaching high vibrations. Such vibrations succeeded in multiple emergency stops done by the vibro-diagnostics condition monitoring systems, as otherwise the turbine can be damaged. As the production process is strictly linear, each turbine downtime caused outage of the whole production, hence causing huge financial losses. A possible solution is to predict incoming problem of unusual vibrations. Such preventive alert enables operator to slightly decrease the turbine load in prior to the oncoming failure, stabilizing the vibrations in advance. Then, after short time period, turbine can safely reach its regular efficiency. In this paper we present a possible solution to obtain this goal. We employ some modern tools to test this approach in practice and compare its efficiency with some other standard tools that can be used.

It is difficult to decide on best solution to the classification problem as a whole. A huge number of different methods were developed during last decade, and which method works best depends largely on the nature of the data (e.g. see [4, 5, 6, 7]). It seems however, that for data with any internal structure, the best option available presently are convolutional neural networks (CNN for short). Their development can be depicted on the example of the classification of images and the Imagenet LSVRC competition. Since 2012, when AlexNet [8] was introduced, CNNs win year by year [9]. Results are significantly better each time, achieving superhuman performance in 2015 [10] and top 5 error below 3% in 2016 (cf. LSVRC results published annually in the internet). The main feature of CNNs is that they use relationships between neighboring variables (adjacent pixels in case of image analysis) and it turned out that features learned by most recent CNN architectures are better than all other proposed in the computer vision. The lowest layers usually detect edges and some other simple motifs, more and more complex features are detected by consecutive layers. In the case of time series, usually the relationship between variables is more complex and there is no method to stack them against each other. However we aim to show that the use of two-dimensional convolutional filters still can give very satisfactory result. In our study we have decided to use one-dimensional filters that work on the values of a given variable at the moment and at its closest history. This approach is based on the assumption that similar features should be relevant to many variables, but instead of looking for them by statistical tools, we use convolutional networks to detect these features for us in the learning process.

2. Model concept and considered architectures

There are several different concepts that can be used for dealing with PdM problems (e.g. see [11]). In our work we will focus on a particular one and solve it using a few different tools for classification of data. By standard approach to machine learning we assume that we have some number of observations, which compose a data set $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^n$ where n should be large. For our problem we may assume that machine is working continuously until it breaks, then is stopped and repaired. But it is not visible directly in the time series because during period of stopping the sensors measurements are not recorded. We only have a time-stamp that at some moment failure occurred. Each X_i represents a response of sensors which are real numbers in some range, that is each $X_i \in \mathbb{R}^p$. Clearly p is in practice much larger than the number of sensors, since each sensor can measure more than one parameter (e.g. temperature and humidity). Output parameter Y_i relates measurement of sensors with present state of the machine. In our approach we consider two possible states *faulty:* F and *not faulty:* NF. State F means that there were some problems with the machine. In some cases it was stopped and repaired, but other possibility is that some preventive control took place, and while the machine was not stopped itself, production was halted for a while. Since we want to prevent bad behavior of the machine, we adopt suggestion of [11] and mark as F not only the last moment when the failure took place, but also last k records preceding the failure. Size k depends on the time horizon which we want to take into account in prediction.

As we see, we transformed prediction problem to a binary classification problem. We have to assign X_i to proper class $Y_i \in \{F, NF\}$, possibly using some "history window" $X_{i-m}, X_{i-m+1}, \ldots, X_i$.

2.1. Two benchmark classifiers

In the paper we want to judge efficiency of deep neural networks in our problem, so we need some benchmark methods as a check of performance. We focus on the following two methodologies, which were quite successful in previous years.

Random forest is one of the basic ensemble methods. It is using pre-specified number of decision trees of the same maximal depth. Trees are build on randomly selected variables and records from original data set. New examples are classified by vote counting from all of them. Random forest is ensemble of weak classifiers which is proven to be fast to learn and evaluate and is a good classifier if trees are sufficiently diverse (see [12] for more details).

Extreme Gradient Boosting (XGBoost) is a model consisted of a lot of weak classifiers, also decision tress. Difference between XGBoost and random forest lies in the way of construction of new trees. They are created consecutively using special cost function. It puts more weights on previously misclassified examples and also punish more complicated trees. Additionally implementation of learning algorithm is optimized so it can run very fast on large data sets. Since its release in 2015 it was highly competitive and won a lot of machine learning competitions (e.g. see [13]).

2.2. Deep Neural Networks

The most important element of ANN approach is selection of correct network architecture. In our approach we decided to use convolutional layers, since networks with such structure were successful previously in data classification problems [8, 10]. This approach took inspiration from study of visual cortex of brain presented by Hubel and Wiesel [14]. Application of this type of networks in the analysis of multidimensional time series was presented before [15, 16], but is not that much common in literature because of the popularity of recurrent neural networks for time-series predictions. The main difference between chosen approach and basic standard layers (e.g. in standard MLP network) is that neurons in convolutional layer are not connected with all neurons from previous layer, but only to small block of neurons. In some sense we may view such a layer as a set of filters, which depends only on a small neighborhood of a neuron when passing data to next layer. It relies on the assumption that if filtering recognizes a pattern properly at some part of the space, then it should work the same well in other peaces (neighborhoods of other neurons).

Layer No.	Type	Input size	Filter size	No. filters
1	Convolutional	$1 \times m \times n$	$1 \times 3 \times 1$	4
2	Convolutional	$4 \times m \times n$	$4 \times 3 \times 1$	4
3	Convolutional	$4 \times m \times n$	$4 \times 3 \times 1$	4
4	Convolutional	$4 \times m \times n$	$4 \times 3 \times 1$	8
5	Convolutional	$8 \times m \times n$	$8 \times 3 \times 1$	8
6	Convolutional	$8 \times m \times n$	$8 \times 3 \times 1$	8
7	Convolutional	$8 \times m \times n$	$8 \times 3 \times 1$	16
8	Convolutional	$16 \times m \times n$	$16 \times 3 \times 1$	16
9	Convolutional	$16 \times m \times n$	$16 \times 3 \times 1$	16
10	Dense	256		
11	Out	2		

Table 1. General architecture of our networks.

Convolutional neural networks are broadly used because of their ability to learn increasingly complex features in consecutive layers [17].

Assume that our data (one time step) has n 1-dimensional sensors and that the time interval of prediction is at least m time steps (e.g. if we collect data every minute and want to feed the network with history of 5 minutes we set $m = 5$). Then the input data of our network has dimension $m \times n$, consisting of m consecutive (in time) recordings of n sensors. This input can also be viewed as $1 \times m \times n$ cube (of depth 1), which will simplify out notation.

Now we have to decide the size and number of filters, which will influence the architecture of all further layers. Since ordering of sensors in our data set is in a sense random, we believe that filter should not compare records of neighboring sensors. However, it should definitely take into account neighboring (in time) measurements. Therefore filter applied in first layer will have dimension $1 \times 3 \times 1$. We will use at this point 4 filters with stride $1 \times 1 \times 1$ and padding $0 \times 1 \times 0$, therefore input to the second layer will have dimension $4 \times m \times n$. In other words, we do not reduce the size of "data matrix" $m \times n$ but test a few different filters (this will apply to all convolutional layers we build). Filter applied in the second layer will have the same dimension on data 3×1, but will take into account all 4 values returned by filtering of first layer. Therefore filter window for second layer will have dimension $4 \times 3 \times 1$ and the same stride and padding as before, which will result in layer three of size $4 \times m \times n$. In total we will consider 9 convolutional layers, with dimensions presented in Table 1. As we can see, first 9 layers are divided into 3 groups with $4, 8$ and 16 acting filters, respectively.

Every filter use ReLU (or PReLU) activation function. Additionally, in each of three blocks we have a residual connection between first and last layer (e.g. layer 1&3 or 4&6) which is added to respective signals of third layer before activation function ReLU (resp. PReLU) is applied. After last (ninth) convolutional layer we include fully connected layer with 256 neurons and again ReLU (resp. PReLU) activation function. Last layer is consisted of two neurons with softmax activation function for proper interpretation as probabilities. In other words, after filtration (convolution) we have a dense drop off layer and finally on the output we have two signals, which provide probability of F and NF events (at the end we choose the most probable class).

2.3. Combined approach

Instead of using single ANN, we can also use some hybrid approach which relies on outputs of a few different ANNs which we find most effective. We consider m classifiers, together with score

$$sc_F = \frac{1}{m} \sum_{\{1 \leq j \leq m\}} C_j^F(X),$$

where $C_j^F(X_i)$ is response (probability of the classification to class F) of j-th classifier on data X. Hence, we obtain weighted probability of all classifiers. If $sc_F > 0.5$ then we classify X as F and as NF otherwise. As we will see, it can have positive effect on decrease of classification mismatch. Such approach was reported previously as successful in most of competitions (see [10]) and as we will see, it is also the case here.

3. Experimental results

3.1. Data set description

Whole data set $\mathcal{D}$ was divided into three parts: *training* $\mathcal{D}_t$, *validation* $\mathcal{D}_v$ and *test set* $\mathcal{D}_f$. In order to avoid the problem of *"predicting the past with the future"*, set $\mathcal{D}_t$ consisted of the data from the first half of the period collected in $\mathcal{D}$, records of $\mathcal{D}_v$ came from the third quarter and records in the last 25% of records of $\mathcal{D}$ were included in $\mathcal{D}_f$. More precisely, 204000 of records in $\mathcal{D}$ were divided into 102500 records in $\mathcal{D}_t$, 51000 records in $\mathcal{D}_v$ and 50500 records in $\mathcal{D}_f$.

As we can see in Table 2, collected data was extremely unbalanced, when classes F and NF were considered. Concrete numbers are presented in Table 2. Such a situation is very common in maintenance problems and its explanation is easy to guess. Simply, operators of the machine associate avoiding fault the highest priority, together with the fact that all parts of the machine are of very high quality so failure by solely machine part failure is very rare as well (it is rather a combination of unexpected behavior of the machine and unexpected, above average part worn out). This causes many practical problems, since the proportion of records between classes is around 1/1000.

Then we cannot simply consider number of bad classifications as the main indicator in training process, because even if all classifications of F event were wrong, it can

Data set	Faulty (F)	Not faulty (NF)
$\mathcal{D}$	83	102417
$\mathcal{D}_v$	21	50979
$\mathcal{D}_f$	54	50446

Table 2. Structure of records distribution between classes F and NF.

be very small number compared to good classifications (ca. 1/1000 in the worst scenario). On the other hand, not detected F event can have much more negative consequences than numerous warnings of F event where there was NF case. To deal with this problem, in training process we use the following cost function for gradient descent method algorithm:

$$H(y, \hat{y}) = -\alpha_F y \log(\hat{y}) - \alpha_{NF}(1-y)\log(1-\hat{y}),$$

where $y \in \{0,1\}$ is classification assigned to sample (1 denotes class F), $\hat{y} \in [0,1]$ is value returned by ANN for that sample understood as the probability of a sample belonging to F, and α_F, α_{NF} are weights representing cost of misclassification of each class. In our approach, we define them as diverted proportions of number of examples from class, that is $\alpha_F = \#NF/\#\mathcal{D}_t$ and $\alpha_{NF} = 1 - \alpha_F$.

3.2. Parameters and algorithms

Random forest was calculated using scikit-learn 0.18. Model which achieved the best result has 200 trees with maximal depth equal to 3. For XGBoost we used xgboost package for python. Model was composed of 200 trees of maximal depth equal to 2. We set learning rate to 0.009 and λ for L2 regularization equal to 0.4. Both models also assigned high weights to class F, equal to number of results in NF class divided by number of results in F class.

In our tests, constructed ANNs were always trained on records from set $\mathcal{D}_t$, and the process was interrupted when the loss function calculated on the validation set was not improved for a certain number of epochs (early stopping). Networks were trained with GPUs using stochastic gradient descent (SGD) with nesterov momentum. Weights from network with best result on validation set $\mathcal{D}_v$ were selected for final model (we will comment on this later). Each network was trained for 10,000 epochs, where one epoch is number of iterations necessary to run through whole training set. Learning procedure was stopped when there was no improvement in value of lost function in 100 consecutive epochs. One of training objectives was to accelerate learning process and to make results more stable, because direct application of (SGD) resulted in less than 30% of attempts with good result on records in validation set $\mathcal{D}_v$. It was mainly due to the fact that most of trainings did not converge to the optimal weights.

First method that significantly sped up learning process was batch normalization [18]. This approach enables using higher learning rates and in consequence neural network much faster finds a solution. Unfortunately, even with batch normalization, training time was too long and without any guarantee of good stable results. To resolve this problem, alternative optimization procedures [19] were examined and as was suggested in literature, method "adadelta" from [20] was usually the best choice, since we observed that in practice this method requires much less epochs to converge than the others. Another important element was to stabilize training in such a way that most of results of learning with the same starting architecture will achieve similar results on validation set. First step was to improve initialization of weights. We decided to use approach proposed in [21], to initialize them from Gaussian distribution with mean 0 and standard deviation equal to $\sqrt{2}/\sqrt{n_i}$, where n_i is the number of neurons in *ith* layer. This initialization is more suitable for ReLU activation function. For PReLU activation we took weights from normal distribution with mean 0 and standard deviation equal to $\sqrt{2}/\sqrt{(1+\alpha^2)n_i}$. Finally, to decrease overfitting, L2 regularization was used. By standard, it was obtained by adding norm of vector of all weights in network multiplied by properly selected factor (see [19] for more details).

3.3. Results of considered models on data set

In what follows we will present results of considered classifiers on validation and test data sets $\mathcal{D}_v, \mathcal{D}_f$, by showing misclassification table of the best classifiers in each group. Additionally, we will calculate the following benchmark function which will help us to improve results comparison:

$$S(x_{MF}, x_{MNF}) = (\beta_F x_{MF} + x_{MNF})/(\beta_F + 1),$$

where $\beta_F = \lfloor \#NF/\#F \rfloor$ and x_{MF} is the number of records F classified as NF, and x_{MNF} is the number of records NF classified as F. Motivation of coefficient β_F is to weigh both misclassifications equally (i.e. classifiers returning always 1 or 0 reach the same error level). We use this score also to select best model on $\mathcal{D}_v$. Note that formula for these functions is different for data sets $\mathcal{D}_v$ and $\mathcal{D}_f$, that is:

$$\beta_F^v = \lfloor 50979/21 \rfloor = 2427, \quad \beta_F^f = \lfloor 50446/54 \rfloor = 934.$$

Results for XGBoost and Random Forest are presented in Table 3. Some research suggests that a good choice for time series can be LSTM networks. While they seem good choice for time-series prediction, we were unable to make it work sufficiently well in considered problems. As an example of these issues, we present in Table 4 results obtained for 3-layer GRU network with 256 neurons and L2 regularization. As we can see the results are not any better compared to XGBoost or Random Forest. By this reason we decided to focus on convolutional networks, which were reported as good classifiers.

It is fair to mention here that there are some other functions known from the literature that could be used instead of function S (e.g. see [22] for some possibilities). The advantage of function S is that it highly prefers decrease of F misclassification, while also has some sensitivity on decrease in NF misclassification. Such choice is

	Classification	Data set D_v		Score S_v	Data set D_f		Score S_f
		NF	**F**		**NF**	**F**	
XGBoost	**NF**	50940	39	2.015238	50403	43	5.040641
	F	2	19		5	49	
Random Forest	**NF**	50932	47	2.018533	50425	21	7.014973
	F	2	19		7	47	

Table 3. Results for XGBoost and Random Forest with parameters from Section 3.2.

	Classification	Data set D_v		Score S_v	Data set D_f		Score S_f
		NF	**F**		**NF**	**F**	
GRU	**NF**	50947	32	2.012356	50427	19	8.01176
	F	2	19		8	46	

Table 4. Results for GRU network.

highly connected with out problem, when we have to decrease bad classifications of F at all cost, and at the same time try to decrease bad classifications of NF when possible.

Next we present results for considered architecture of ANN (with convolutional layers) for different values of parameter λ in L2 regularization and ReLU or PReLU activation functions. We consider 6 different approaches to ANN learning as presented in Table 5. We trained 30 different networks. The number of networks with scores on $\mathcal{D}_v$ better than XGBoost is presented in the last column of Table 5. Next, from 10 best networks in each group we constructed ensembles which average answers of these networks (see Section 2.3).

Type	activation function	weights initiation	λ	Better than XGBoost
T1	PReLU	ReLU	0.000 001	6/30 (20%)
T2	ReLU	ReLU	0.000 0005	5/30 (16,67%)
T3	ReLU	ReLU	0.000 001	3/30 (10%)
T4	ReLU	ReLU	0.000 002	3/30 (10%)
T5	PReLU	PReLU	0.000 001	3/30 (10%)
T6	PReLU	PReLU	0.000 002	8/30 (26,67%)

Table 5. Six different methods of training within architecture from Table 1.

	Classification	Data set D_v		Score S_v	Data set D_f		Score S_f
		NF	F		NF	F	
T1	NF	50778	201	1.082372323	50139	307	**0.328342246**
	F	1	20		0	54	
T2	NF	50732	247	1.101317957	42068	8378	8.960427807
	F	1	20		0	54	
T3	NF	50813	166	1.067957166	48576	1870	2
	F	1	20		0	54	
T4	NF	50925	54	1.021828666	49905	541	1.577540107
	F	1	20		1	53	
T5	NF	50647	332	1.136326194	41765	8681	9.284491979
	F	1	20		0	54	
T6	NF	50042	937	0.385914333	50116	330	8.344385027
	F	0	21		8	46	

Table 6. Results for best ANNs within group of 30 networks of given type, with parameters as described in Table 5.

	Classification	Data set D_v		Score S_v	Data set D_f		Score S_f
		NF	F		NF	F	
T1	NF	50830	149	1.060955519	48655	1791	1.915508021
	F	1	20		0	54	
T2	NF	50911	68	1.027594728	49765	681	0.728342246
	F	1	20		0	54	
T3	NF	50869	110	1.044892916	50302	144	1.152941176
	F	1	20		1	53	
T4	NF	50834	145	2.058896211	50291	155	**0.165775401**
	F	2	19		0	54	
T5	NF	50835	144	1.058896211	41466	8980	9.604278075
	F	1	20		0	54	
T6	NF	50863	116	1.047364086	49462	984	1.052406417
	F	1	20		0	54	

Table 7. Results for ensembles of ANNs of type as in Table 5.

4. Conclusions

In this paper we presented a new approach to predictive maintenance problem, employing deep convolutional networks with one dimensional filters and residual connections. Presented single network architecture achieved results better than two standard methods commonly used to deal with this type of problems. The main difficulties in considered problems were highly unbalanced classes and instability of learning.

It is evident in Table 6 that single networks are very sensitive with respect to function S. It may happen (and is visible in the Table) that best function on D_v can perform quite badly on D_f, sometimes even below the benchmark performance of other methods in Table 3. It provides a motivation for considering ensembles, which we found quite effective when dealing with considered problem. We verified that in all considered classes of architectures ensemble of 10 networks achieve very good results of classification classes F as F. Additionally, in some cases they were able to obtain better performance on false alarms (smaller number of such instances).

Obtained results were satisfactory for the industrial partner, ensuring significant economical savings. While it is not a part of present research, we noticed that obtained architectures and learning procedures can achieve very good results on some other data sets, however due to limited space of this paper, we decided to not present these results here.

In the future research we would like to find an efficient way to add newly collected observations to already trained network. In predictive maintenance setting, new data is generated every minute and so it is desirable to develop a self-learning model, which does not involve training the whole network. Such algorithms may result in models reaching even better accuracy, being resistant to small changes in the production process.

5. Acknowledgements

Presented work was undertaken within the framework of a research project “Deep learning in distributed system of energetically efficient mobile devices to optimize the machines exploitation process” within the Measure 1.1.1. “Industrial research and development works by companies”, co-financed by the European Union from the European Regional Development Fund under the Smart Growth Operational Programme 2014-2020. Financial support of the above institutions and programs is gratefully acknowledged.

6. References

[1] Rosmaini A., Shahrul K., *An overview of time-based and condition-based maintenance in industrial application.* Computers & Industrial Engineering, 2012, 63 (1), pp. 135–149.

[2] Jardine A.K.S., Lin D., Banjevic D., *A review on machinery diagnostics and prognostics implementing condition-based maintenance.* Mechanical systems and signal processing, 2006, 20 (7), pp. 1483–1510.

[3] Compare M., Zio E., *Predictive maintenance by risk sensitive particle filtering.* IEEE Transactions on Reliability, 2014, 63 (1), pp. 134–143.

[4] Uysal H., *A genetic programming approach to classification problems.* University College Dublin Dublin, Ireland, 2013.

[5] Bishop C.M., *Pattern recognition and machine learning.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[6] Mather P., Tso B., *Classification methods for remotely sensed data.* CRC Press, Boca Raton, 2016.

[7] Aggarwal C.C., *Data classification: algorithms and applications.* Chap-man |& Hall/CRC, 1st edition, 2014.

[8] Krizhevsky A., Sutskever I., Hinton G.E., *Imagenet classification with deep convolutional neural networks.* In *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[9] Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. *Imagenet large scale visual recognition challenge.* International Journal of Computer Vision, 2015, 115 (3), pp. 211–252.

[10] He K., Zhang X., Ren S., Sun J.. *Deep residual learning for image recognition.* In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[11] Susto G.A., Schirru A., Pampuri S., McLoone S., Beghi A., *Machine learning for predictive maintenance: A multiple classifier approach.* IEEE Transactions on Industrial Informatics, 2015, 11 (3), pp. 812–820, 2015.

[12] Breiman L., *Random forests.* Machine learning, 2001, 45 (1), pp. 5–32.

[13] Che T., Guestrin C., *Xgboost: A scalable tree boosting system.* In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[14] Hubel D.H., Wiesel T.N., *Receptive fields and functional architecture of monkey striate cortex.* The Journal of physiology, 1968, 195 (1), pp. 215–243.

[15] LeCun Y., Bengio Y., et al. *Convolutional networks for images, speech, and time series.* The handbook of brain theory and neural networks, 1995, 3361 (10), pp. 1-14.

[16] Yang J., Nguyen M.N., San P.P., Li X., Krishnaswamy S., *Deep convolutional neural networks on multichannel time series for human activity recognition.* IJCAI, 2015, pp. 3995–4001.

[17] Fukushima, K., Miyake S., *Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition.* Competition and cooperation in neural nets, 1982, pp. 267–285.

[18] Ioffe S., Szegedy C., *Batch normalization: Accelerating deep network training by reducing internal covariate shift.* In *International Conference on Machine Learning*, 2015, pp. 448–456.

[19] Courville A., Goodfellow I., Bengio Y., *Deep Learning.* MIT Press, 2016.

[20] Zeiler M.D., *Adadelta: an adaptive learning rate method.* *arXiv preprint arXiv:1212.5701*, 2012.

[21] He K., Zhang X., Ren S., Sun J., *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.* In *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[22] Caruana R., Niculescu-Mizil A., *An empirical comparison of supervised learning algorithms.* In *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 161–168.