



Science Arts & Métiers (SAM)

is an open access repository that collects the work of Arts et Métiers Institute of Technology researchers and makes it freely available over the web where possible.

This is an author-deposited version published in: <https://sam.ensam.eu>
Handle ID: <http://hdl.handle.net/10985/23026>

To cite this version :

Kenza AMZIL, Esma YAHIA, Nathalie KLEMENT, Lionel ROUCOULES - Automatic neural networks construction and causality ranking for faster and more consistent decision making - International Journal of Computer Integrated Manufacturing p.1-21 - 2022

Any correspondence concerning this service should be sent to the repository

Administrator : scienceouverte@ensam.eu



Automatic neural networks construction and causality ranking for faster and more consistent decision making

Kenza Amzil^a, Esma Yahia^a, Nathalie Klement^b and Lionel Roucoules^a

^aArts et Metiers Institute of Technology, LISPEN, HESAM Université, F-13617 Aix-en-Provence, France; ^bArts et Metiers Institute of Technology, LISPEN, HESAM Université, 59046 Lille, France.

ARTICLE HISTORY

Compiled November 30, 2022

ABSTRACT

The growth of Information Technologies in industrial contexts have resulted in data proliferation. These data often underlines useful information which can be of great benefit when it comes to decision making. Key Performance Indicators (KPIs) act simultaneously as triggers and drivers for decision making. When they deviate from their targets, decisions must be rapidly and well made. Therefore, experts need to understand the underlying relationships between KPIs deviations and operating conditions. However, they often interpret deviations empirically, or by following methods that may be time consuming, or not exhaustive. This article proposes a generic neural network based approach for improving decision making, by ensuring that decisions are consistent and made as early as possible. On the one hand, the proposal relies on improving KPIs deviations prediction, which is made possible thanks to the automatic construction of neural networks using neuro-evolution. On the other hand, the decision making consistency is improved by identifying, among the operating conditions, contextual variables that most influence a given KPI of interest. This identification, that guide the decision making process, is based on the analysis of the final weights of the neural network used for the KPI deviation prediction, given the contextual variables.

KEYWORDS

Decision making; Neural networks; Neuro-evolution; Predictors' prioritization; Causality analysis.

1. Introduction

Nowadays, industry, as many other sectors, is facing an increasing competitiveness due to changing market and globalization (Tsai-chi 2021). This leads to increasing demands on process performance and products quality (Aydiner et al. 2019). With the aim of fulfilling these demands, decision-makers are constrained to determine the desired industrial objectives and continuously monitor them. To do so, Key Performance Indicators (KPIs) are widely used. KPIs serve as a strong monitoring tool that quantifies efficiency and effectiveness of processes and engaged actions, as well as product's performance, quality, or business goals achieving (Tambare et al. 2022; Yin, Zhu and Kaynak 2015). Therefore, monitoring by KPI facilitates the detection of deviations and unexpected evolutions of the entity being monitored (Laudon and Laudon

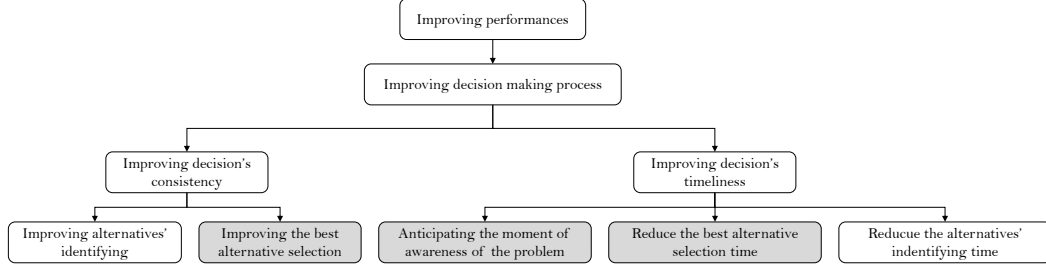


Figure 1. Decision making improvement aspects treated in the present paper.

2019). In this context, when deviation is observed, a decision has to be made in order to re-establish the situation. Nevertheless, simply reporting performance information is not enough to improve it and to take wiser decisions (Pérez-Álvarez et al. 2018).

Decision making consists in considering, when confronted with a problem, several alternatives, and then making a choice after reflection (Keding and Meissner 2021). Improving the decision making process involves two aspects : (i) improving the timeliness of the decision-making process, and (ii) improving the decision's consistency. The first aspect is about reducing the time required for the decision to be made. Indeed, decision making can be time and resource consuming, especially when dealing with new issues. The second improvement aspect requires understanding the original causes of the problem in order to consider feasible alternatives and select the best one (Keding and Meissner 2021). Dealing with KPIs implies identifying, among the contextual variables, the ones involving changes in the values or states of the addressed KPI, and establishing the hierarchy of their respective impacts, in order to conclude on the most effective action plan. By "contextual variables" we mean manipulable and measurable variables retrieved from the physical or digital world.

Concerning the first aspect, when decisions are driven by KPIs monitoring, KPIs deviations' forecasting can be of great help in anticipating the awareness of the need to take a decision ; which allows anticipating actions before deviation occurs. For the second aspect, identification of causal relationships that drive the behaviour of the addressed KPI can be achieved using several approaches, being either empirical, descriptive, or analytic. Among these methods, Bayesian networks are a widely used tool for representing independences and reporting causal information for decision making if they are built for that purpose (Misirli and Bener 2014). Bayesian Belief Networks (BBN), also called causal Bayesian networks, are Bayesian networks where links represent causal impacts (Hänninen and Kujala 2012). The graphical causal structure provided by BBN makes them easy to understand, and gives an intuitive representing of causal relationships which can be of a great help to the experts. In this paper, we assume that the causal Bayesian structure describing causal relationships between a given KPI of interest and the contextual data is already known.

The aim of this paper is to provide a proactive decision support for two purposes: **(i) automatic construction of forecasting models in order to enable forecasting for as many KPIs as available**, these models will then be used in order to trigger the decision making process before deviation occurs ; and **(ii) providing the hierarchy of the influencing factors** in accordance with their respective

involvement in the KPI variations. The effectiveness of our approach is based on two strong assumptions. The first assumption is about the availability of a causal graph faithful to reality. This can be achieved by combining prior knowledge and automatic learning in order to provide a causal Bayesian network structure that captures causal relationships between contextual variables and the addressed KPI (Amzil et al. 2020). This ensures that hierarchy of influencing variables would only concern causes, and not correlated variables. The items treated herein are based on this resulting causal structure, in the sense that only the variables causally related to the monitored KPI are taken into account to make the prediction of feared events and to guide decision making. The second one is about assuming that historical values or states of KPIs of interest, and historical data related to the contextual variables, are available, and that is possible to retrieve as much contextual data as possible, including machine, processes, environmental, and human factor data, by using Internet of Things (IoT). The purpose of this assumption is to enable the construction of the forecasting model on the one hand, and to expand the research scope on the other hand. This assumption is related to the first one, in the sense that with the availability of these data, the causal graph would be more likely to overcome cognitive biases, and to include factors that weren't suspected to be involved in the treated KPI deviation. It is also related to the first one in the sense that historical data are required for building the causal structures. In this way, data explosion problem resulting from the growth of Information Technologies in the context of Industry 4.0 (Klingenberg, Borges and Antunes Jr. 2019) can be turned into a real opportunity. Hence, the executives' inability to cope with large amount of data and to find useful information among it will be reduced (Laudon and Laudon 2019). This second assumption is supported by the increasing accessibility and use of IoT in the industry, as well as its ongoing fall in costs (Kamiński et al. 2019).

The rest of this paper is organized as follows. First, the problem is described and an overview of related work is provided. The approach is then presented, followed by a benchmark. A consistency analysis of results is then performed on a second benchmark. Finally, the results are presented and discussed, before concluding and highlighting future work.

2. Problem description and related work

2.1. *Construction of models for events forecasting*

When a given KPI is being monitored and a deviation from the target is discovered, action must be taken on the most influential element. In order to be proactive and to reduce problem solving time by taking timely decisions, it is important to predict the future evolution of the monitored indicator (Huang et al. 2019). The availability of historical data makes supervised learning possible. The objective is therefore to build a prediction model that is able to predict the KPI state or value, given an observation of contextual variables. Since we consider that the causal structure is available, variables related in a direct or indirect way to the addressed KPI would be enough for serving as predictors. In this article, when we talk about predicting KPI evolution we both mean prediction of KPI value, *i.e.* regression with a continuous variable to predict; and KPI state, *e.g.* binary classification to predict whether the indicator will meet the target or not.

Several predictive data mining techniques have been developed and used in many domains in the aim of driving prediction. In industrial contexts, different KPIs must be monitored and predicted on the basis of available data. These indicators can sometimes be numerous and may change over time and according to the desired objectives. The prediction model must therefore be able to be easily built and modified for all the indicators of interest. Depending on the KPI of interest, the contextual variables that are causally related to it, the data types of both the KPI of interest and its causal contextual variables, and their historical values, a prediction model must be built individually, and its accuracy will depend on the model's parameters. In order for the decision making to be triggered when needed, the model built for the prediction has to be as accurate as possible. Also, regarding the industrial contexts, both qualitative and quantitative data have to be manipulated by the prediction model, since the causal structure may include contextual variables with mixed types of data.

A literature review allowed us to identify many existing methods of machine learning. Our need for handling both quantitative and qualitative types of data forced us to discard automatic classification methods such as decision trees, k-means, k-medoids and its variants since they do not enable continuous variables prediction. Moreover, learning the optimal decision trees is NP-complete according to many aspects, and it may lead to very complex decision trees with poor generalization capability (Bramer 2007). Logistic and linear regression are also widely used to predict, however their predictive power is limited only to the case of linear relationships between the predictors and the variable to be explained (Tu 1996). Artificial neural networks (ANN) have widely been used to solve complex problems. They enable learning and modelling non-linear and complex relationships between predictors and outputs in order to predict them, whether qualitative or quantitative. They also owe their success to their high performance, computing efficiency, and ease of use (Kalainathan 2019). Hence, in this paper, focus will be placed on prediction of KPIs deviations, regarding their values or states using ANN.

Although neural networks are a very efficient forecasting technique that has been used in many industrial applications (Dalzochio et al 2020), their building is still subject to discussion. Indeed, the classical way of defining an ANN topology and hyper-parameters is an iterative and blind process conducted by the ANN designer, by adjusting them and launching the ANN learning and testing, until the learned outputs meet the expected outputs with high accuracy. The designer must therefore follow a recurrent process of editing, learning and testing until finding structure and parameters that maximize accuracy. Indeed, manual construction of ANNs with good predictive and generalization power results from an experimental approach composed of a series of successive refinements of its hyper-parameters and structure, and this must be conducted for each single indicator we aim to monitor, which can be time and human resources consuming. By structure we mean the number of hidden layers, and the number of neurons per hidden layer. By hyper-parameters we mean initial learning rate and its set-up, activation function, the number of epochs, and the learning algorithm. Manual construction of ANNs goes against the overall timeliness of processes, because the KPIs to predict may change over time by depending to other contextual variables. In this case, ANNs that used to predict KPIs have to be updated. The KPIs to predict may also be replaced by others, according to the

enterprise’s new objectives. In this case, new ANNs for predicting the new KPIs have to be built.

In the literature, many algorithms are suggested for designing ANNs, the most used ones are the constructive, pruning or destructive algorithms (Shih-Hung and Yon-Ping 2012). Constructive algorithms, which begin with a minimal architecture, then add neurons or connections during the training phase, suffer from the drawback that it is difficult to decide when to add neurons or hidden connections and when to stop the upgrading (Shih-Hung and Yon-Ping 2012). Pruning algorithms perform in the opposite way, they begin from a large ANN then prune units using one of the existing pruning criteria (Han, Zhang and Qiao 2017). Oppositely, these algorithms suffer from the constraint of the starting network, usually defined by the ANN designer. The problem of underestimating the problem may then arise, and consequently, the starting network would be insufficient to get high accuracy.

The proposed approach aims to automatically define ANN parameters and structure in order to easily and rapidly build or update a model for forecasting future events related to KPIs of interest. It suggests to use an evolutionary ANN designing method, also called neuro-evolution, based on Genetic Algorithms (GA), and which have shown large potential in designing ANN (Mantzaris, Anastassopoulos, and Adamopoulos 2011). Although many researchers have used GA to design ANN (Stanley et al. 2019; Ahmadizar et al. 2015; Shih-Hung and Yon-Ping 2012), to our best knowledge, the choice of the appropriate activation function has not been addressed so far by these evolutionary approaches.

2.2. *Hierarchy of identified causes*

When deviation is predicted, the issue is to rapidly pinpoint the factors on which it would be judicious to act. Let us consider that the KPI of interest is the Overall Operations Effectiveness (OEE) for which the experts have already set a threshold. Consider also that a good performing ANN has been built for forecasting the state of the OEE (*i.e.* if it will exceed the threshold or not), given the actual states and values of the contextual variables. The formula for calculating the OEE is given by 1 :

$$OEE = \frac{\text{Theoretical Production Time}}{\text{Actual Production time}} \quad (1)$$

Let us assume that the ANN used to predict the OEE state is expecting the OEE to exceed the threshold. This deviation prediction will therefore trigger the experts’ need to make a decision, which is good for anticipating the decision making and probably avoiding the deviation. However, the prediction of the OEE deviation is not enough to guide the decision making. Also, the basic definition of the OEE, given by 1, is not enough to identify the elements on which direct action can be made in order to efficiently restore the situation. Since it is not possible to act on production time in order to re-establish the situation, this formula will not allow experts to engage an action : they need to understand the underlying causes behind the OEE deviation. Furthermore, when causes are known, experts can identify alternatives in order to prevent deviation, but they yet need to select the best alternative.

When the causal structure is known, as it is one of our assumptions, the hierarchy of causes related to the OEE can be given by the experts who analyse the situation and pinpoint elements that have most contributed to the OEE deviation. However, this process can be time and resource consuming, especially when dealing with new issues. Besides, in the context of manufacturing, many factors may be present in the causal structure related to the OEE, and it may be difficult to pinpoint the ones that most influence the OEE states (Laudon and Laudon 2019). Also, the analysis of the identified causes of a given KPI's deviations is sometimes impacted by cognitive biases due the decision-maker's experience, which prevents them from increasing their actions' scope and discovering factors that may be more influencing than they think (Ballard 2019; Moeuf et al. 2017).

Also, when a BN describing relationships between an addressed KPI and contextual data is given, marginal and conditional probabilities tables associated to the causal BN structure give the probabilities attached to nodes states, given the states of their parents (Weber et al. 2012). In industrial contexts with complex systems, the addressed KPI is a part of many other available variables that may explain the KPI deviation. This results in a complex Bayesian network in which each node may take different states and may have parents, which also may take multiple states. This leads to very large conditional probabilities tables that may be very difficult to interpret (Marcot 2017), especially since most people cannot interpret information beyond four dimensions (Pollino and Henderson 2010). This hinders the consideration of relative causal influences of the different influencing variables (Marcot 2017). Influence levels can still be captured by conducting sensitivity analysis on the different combinations of implicated nodes states, but this is often time and effort consuming (Kjærulff and Van Der Gaag 2013), and therefore not practical in a context where timely and accurate decisions must be taken to avoid deviations. Focusing more closely on ANNs, we realize that in addition to being able to forecast future events, the ANN final weights can be recovered and analysed with the intention of identifying the most influential factors. The connection weights can be considered as the equivalent of the coefficients in regression models and contain the "knowledge" acquired by a neural network after it has been trained (Tu 1996).

Therefore, in our context, and thanks to automatic construction of ANNs, it would be more suitable to identify factor influences by interpreting final weights of a good performing ANN. Indeed, since ANNs that have been built can provide high precision, and since they can handle different data types, their final connection weights can then be exploited in order to provide the hierarchy of respective influences of their entries on their output. Hence, the hierarchy of the respective influences of contextual variables causally linked to KPI on this latter. The resulting hierarchy, which highlights the relationships' strengths between contextual variables and the KPI of interest can thus be used to drive action on manipulable elements. The problem to which this article tries to answer is therefore to be able to reduce decision making time and experts' efforts by giving the ability (i) to easily build a model to forecast future events related to a KPI ; and (ii) to prioritize the elements on which action must be taken by ranking the influencing factors used in the model. In this way, only one model (*i.e.* one ANN) will be used for both forecasting future events in order **to trigger the decision making process** before KPI deviation occurs on the one hand, and for providing hierarchy of contextual variables in order **to guide the decision making and to accelerate the best alternative se-**

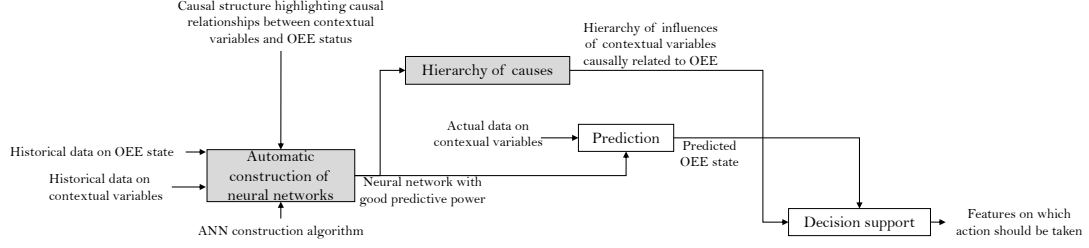


Figure 2. SADT diagram describing the global process of the proposal.

lection on the other hand.

3. Proposal: Architecture of the method

Figure 2 presents the SADT diagram (stands for Structured Analysis and Design Technique), describing the global process of our approach. On this figure, the KPI of interest is represented by the OEE, and the contextual variables represent all the measurable variables retrieved from physical and digital worlds. As previously mentioned, causal structure linking the contextual variables to the OEE is supposed to be known. First, an ANN with high accuracy is defined using an evolutionary construction algorithm. The ANN definition is constrained by the causal structure, in the sense that only contextual variables causally linked to the OEE can be included in the entries of the ANN. Then, the hierarchy of the predictors, which are also the contextual variables linked causally to the OEE, is performed. These two steps (with grey color in the figure), represent configuration steps. That is to say that they need to be performed beforehand so that the proposed system can be used in real time. It means that they are performed at least once for each KPI of interest in order to get its corresponding ANN and hierarchy of causes. These steps can then be renewed punctually in order to update the predictive ANN and the hierarchy of causes. Once these two elements are available, prediction of the OEE state can be launched in real time, according to a predefined frequency, by using the ANN that have been constructed in the first step, and based on the actual values or states of the contextual variables. Finally, if OEE deviation is predicted, decision support is triggered, and it is based on the hierarchy of influences of contextual variables causally linked to the OEE state. The decision support provides, as output, the features on which action should be taken, with a ranking in accordance to the influences hierarchy, in order to prevent the predicted OEE deviation.

Figure 3 represents an generic explanatory scheme of the process of the proposal, which is made up of two phases : configuration phase, which corresponds to the two bricks of figure 2 with grey background in figure ; and utilization phase, which corresponds to the two bricks of figure 2 with white background.

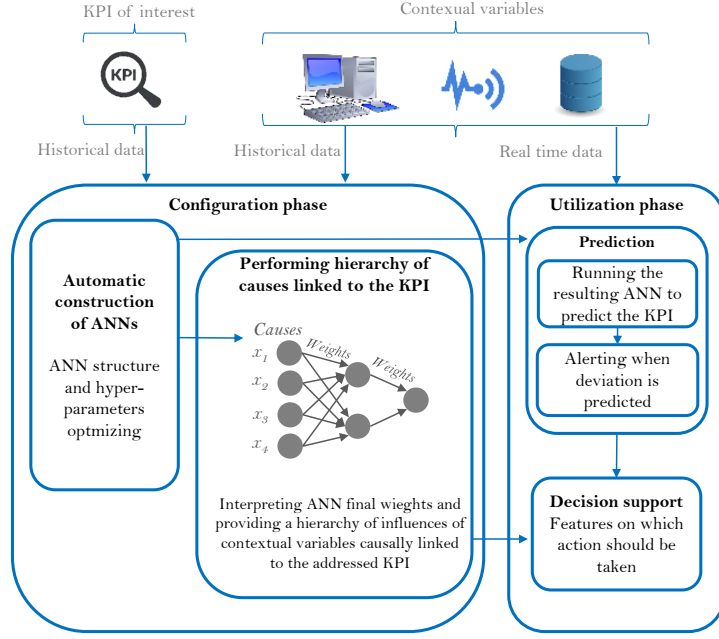


Figure 3. Explanatory scheme of the process of the proposal: Configuration and utilization phases for guiding decision making processes.

4. Proposal: Building ANNs and ranking the predictors

4.1. 4.1. Neuro-evolution for neural network construction

In this article, focus is placed on multilayer perceptron (MLP) neural networks, which have been shown to be efficient for many industrial applications such as manufacturing process control, process and machine diagnosis, machine maintenance analysis, and planning (Hagan et al 2014). As shown in figure 4, multilayer perceptron neural networks are composed of a series of layered neurons that transform inputs into outputs through hidden layers, by means of an activation function and connection weights between neurons. Input neurons are represented by $x_1; \dots; x_n$, where n is the number of the inputs, hidden neurons are represented by h_{ij} where i is the layer number and j is the hidden neuron index in the i^{th} layer. Y represents the output. Neurons of the different layers are interconnected and a weight is assigned to each connection, the weights are represented by w_{kl} where k is the connection departure neuron and l is the arrival neuron.

Figure 4. Multilayer artificial neural network with single output.

Our approach suggests to use an evolutionary ANN designing method, also called neuro-evolution. For each KPI of interest, the aim is to find an ANN maximizing the prediction accuracy. Figure 5 illustrates the best ANN searching process for the prediction of the OEE state, given the contextual variables : different ANNs with different structures and hyper-paramters have to be trained and tested ; the one with the best accuracy is then selected for the prediction of future OEE states.

Figure 5. Best ANN searching process for the prediction of the OEE state.

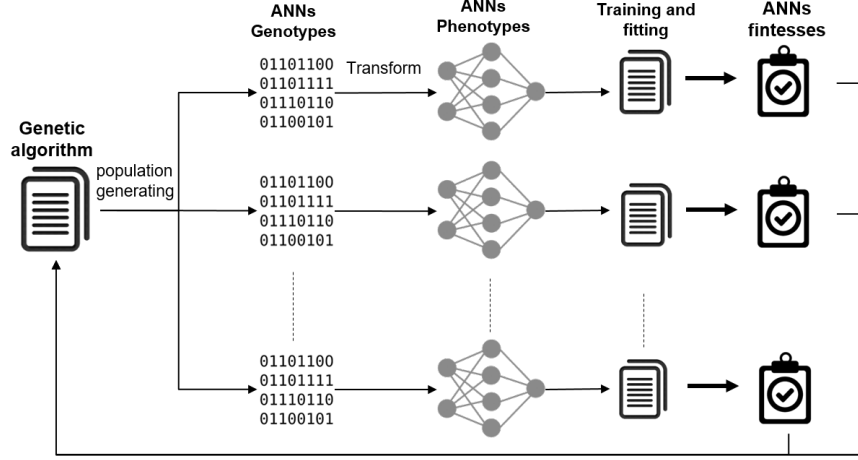


Figure 6. Artificial neural networks evolution process.

The developed GA aims to automate this processes. It is launched with the aim of constructing an ANN by optimizing, through generations, the number of hidden layers, the number of neurons per hidden layer, the learning algorithm, the choice of the activation function, and the initial learning rate as well as and its set-up. Figure 6 shows an overview of the proposed ANNs automatic creation and evolution process. The principal GA file aims to create a population of ANNs genotypes, with a specific coding that enables the evolution operations application (*i.e.* crossover and mutation). The ANNs genotypes of the population are then transformed to real ANNs in order to be trained and tested. The testing results of the created ANNs (*i.e.* their accuracies) are then sent to the GA file which performs the selection of best ANNs, as well as crossover and mutation operations, in order to generate the ANNs genotypes population of the next generation. The ANNs genotypes of the next generation follow the same process, until reaching the maximum generations number. The number of generations and the population size are pre-set by the user. It should be noted that whatever the KPI of interest, the procedure remains the same.

4.1.1. Initial population generating

The initial population is made up of a set of randomly generated and fully connected ANNs, encoded in genotypes, each genotype is composed of binary chromosomes. The generated ANNs are composed of one to five hidden layers. The choice of this interval is based on the recommendations available in the literature (Yu and Seltzer 2011). In general, ANNs with two hidden layers are enough for approaching functions with any kind of shape (Panchal et al. 2011). However, the optimal number of hidden layers depends on the number of input and output units, the number of training samples, the amount of noise in the sample data set, and the training algorithm (Sheela and Deepa 2013). In our proposal, to avoid having too many starting networks with more than two hidden layers, and to avoid falling into over-learning, the number of hidden layers per generated neural network is randomly defined according to the following probability distribution: a probability of 0.3 for the generated ANNs to have one hidden layer, 0.4 to have two hidden layers, 0.2 to have three hidden layers, 0.05 to

Table 1. Artificial neural network genotype description.

| Chromosome | Hyper-parameter | Transformation rule |
|------------------|--|--|
| 0000000000000000 | Hyper-parameters beginning | Binary to decimal conversion. Binary to decimal conversion, each activation function corresponds to an integer code number going from 0 to 6. |
| 0001000000000010 | Hidden layers number | |
| 0010000000000011 | Activation function | |
| 0011000000110010 | Number of epochs | Binary to decimal conversion, the decimal number is then multiplied by 10 and rounded off to the nearest multiple of 50. |
| 0100000011010110 | Learning rate | Binary to decimal conversion, the decimal number is then divided by 10^4 . |
| 0101000000000001 | Learning algorithm | Binary to decimal conversion, each learning algorithm corresponds to an integer code number going from 1 to 3. |
| 0110000000000000 | Learning rate set-up | Binary to decimal conversion, 0 corresponds to constant learning rate and 1 to adaptive learning rate. |
| 1000000000000000 | Hidden layers beginning | Binary to decimal conversion. |
| 1001000000000110 | 1 st hidden layer number of units | |
| 1010000000000100 | 2 nd hidden layer number of units | |

have four hidden layers, and 0.05 to have ve hidden layers. These probabilities were fixed given the fact that ANNs with up to two hidden layers are able to approximate most of non linear complex problems (Karsoliya 2012). Activation functions for each ANN are obtained randomly from a uniform discrete distribution on the following: identity, binary step, sigmoid, hyperbolic tangent, rectified linear unit, softplus, and bent identity. The training algorithm is selected in the same way among the classical Stochastic Gradient Descent (SGD), the quasi-Newton Limited memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS), and Adaptive Moment estimation (Adam algorithm). A learning rate between 10^{-4} and 0.4 (Smith 2018) is applied for ANNs using a stochastic gradient descent learning algorithm, *i.e.* SGD or Adam. It may be either constant, or adaptive by remaining constant as long as the loss is decreasing, and by decreasing as soon as the loss does not decrease for two consecutive epochs. The number of epochs is chosen between 200 and 1200 from a uniform discrete distribution on multiples of 50. The number of hidden units per hidden layer is chosen between one and 30 using a uniform discrete distribution. For each ANN, all of these hyper-parameters are encoded in a binary genotype of binary chromosomes as explained in table 1. A mask is applied on the chromosomes in order to prevent them from taking unexpected values and to ensure that they stay in the defined intervals. The first four bytes starting from the left encode the hyper-parameter encoded by the chromosome, e.g 0001 for the chromosome that specifies the hidden layers number, 0010 for the chromosome that specifies the activation function, etc. The remaining twelve bytes define the value of the hyper-parameter. The genotype is split into two groups, the first group that begins with 0 starting from the left encodes hyper-parameters applied to the ANN, and the second group that begins with 1 starting from the left encodes parameters applied to the each hidden layer. Table 1 describes the coding of a network composed of two hidden layers with respectively six and four hidden units, using a sigmoid activation function, a constant learning rate of 500 epochs, and SGD optimizer.

In order to maintain the similarity trait essential to the good functioning of the

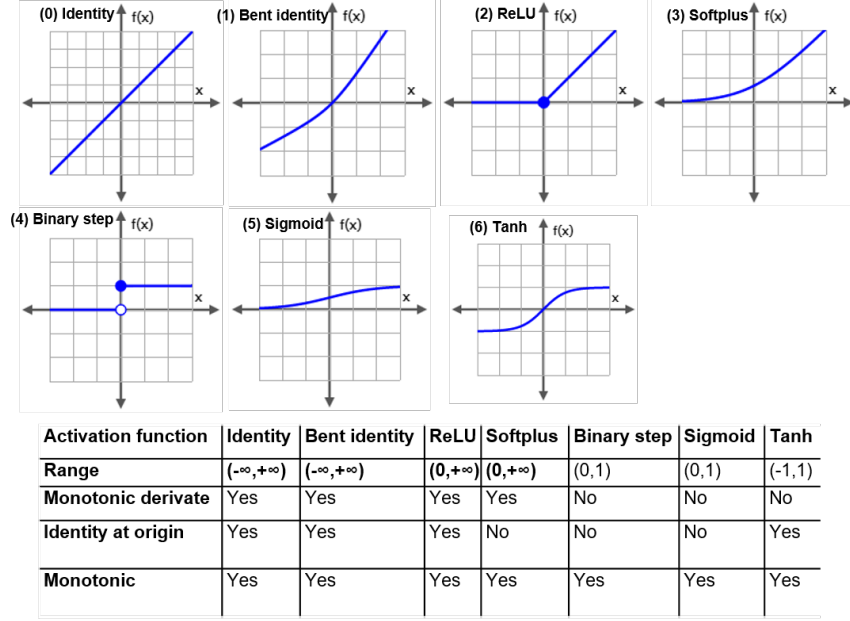


Figure 7. Activation functions codes and characteristics.

genetic algorithms, the activation function, which is considered as qualitative, should be encoded in a way that the offspring activation function should partially inherit some aspects from the parents. For this purpose, the assignment of codes for each activation function should be done in such a way that the crossover generates offspring whose activation function is coherent with ones of its parents. Therefore, their codes were sorted as shown in figure 7 according to their characteristics. Each activation function have similarities with its neighbours. Usefulness of this sorting will be further explained when introducing the crossover step.

The resulting ANN genotypes are then transferred one by one into another file that transforms them into phenotypes (*i.e.* real ANNs) by decrypting the codes according to the initial definitions. Learning and predicting phases are then launched for each of them.

4.1.2. Evaluation and selection

Individuals are selected to generate the offspring based on their performances. The fitness to optimize is simply the mean accuracy, given by 2, in case of classification problems.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

where TP , TN , FP , FN stand respectively for true positives, true negatives, false positives and false negatives; and the coefficient of determination R^2 , given by 3, in case of predicting continuous values.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (3)$$

where \hat{y}_i is the predicted value, y_i is the correct value, and \bar{y} is the mean of all of the correct values. Elitist selection is applied and elements maximizing the fitness are chosen for breeding.

4.1.3. Crossover and mutation

The next generation will be composed of the selected individuals of the previous generation, in addition to offspring individuals resulting from the crossover between the selected individuals of the previous generation. Every two parents generate one offspring according to the following rules:

- Hidden layers and units numbers: 0.5 probability of performing a single point crossover, 0.5 probability of taking one of the two parents whole chromosome with the same probability;
- Epochs number and learning rate: single point crossover. A mask is mapped to avoid applying learning rate with an incompatible algorithm;
- Learning algorithm and learning rate set-up: taking one of the two parents whole chromosome with the same probability, a mask is applied to avoid setting the learning rate if there is none;
- Activation function: an arithmetic crossover is performed by converting to decimal and then randomly selecting an integer in the range $[c1 - 0.5(c2 - c1); c2 + 0.5(c2 - c1)]$; where $c1$ and $c2$ are the smallest and largest codes of the parents' activation functions. The offspring has 0.5 probability of having the activation function corresponding to the selected code, 0.5 of having the activation function of one parent. This interval enables getting a code between the two parents' codes, but also a neighbour code beyond the interval of the two parents' codes, *i.e.* on the right of the largest code and on the left of the smallest one. The selection interval is reduced to the range of possible values if it is wider;
- Learning rate: this step is performed after the chromosomes of the training algorithm have crossed. If the algorithm resulting from this crossing is the LBFGS, the crossing of the chromosomes of the learning rate does not take place, and this chromosome takes a string of bits all at 0, translating that no rate is applied. If the algorithm resulting from the crossover is SGD or Adam, the crossover of the learning rate chromosomes takes place, and the resulting chromosome has a probability of 0.5 to take over the entire chromosome from one of the two parents, with equal chances, and a probability of 0.5 of carrying out an arithmetic crossing making it possible to obtain a learning rate corresponding to the average of the learning rates of the two parents in the case where both parents apply a learning rate. In case only one parent has a learning rate, the descendant takes the same rate;
- Configuration of the learning rate: this step is performed after the crossing of the chromosomes of the learning rate. If the chromosome resulting from this crossover is different from a string of 0, the crossover of the chromosomes of the learning rate configuration takes place, and the resulting chromosome has a probability of 0.5 to take the learning rate configuration of the first parent, and 0.5 to take that of the second parent. We note that when we speak of a string

Figure 8. Output of a single neuron.

of 0, we are referring to the modifiable string of the chromosome (i.e. the entire chromosome, except the first four bits starting from the left).

In order to enrich the exploration, a mutation is randomly applied, with a rate of 0.005 to a chromosome gene among those allowed to mutate. At the end of these operations, a new population is thus generated, it is composed of the best performing ANNs of the previous population and their offspring. The process is repeated until maximizing the fitness.

4.2. Predictors prioritization

4.2.1. Methodology

Thanks to a neural network capable of correctly predicting the addressed KPI, experts are capable of anticipating deviations and must therefore take action in advance in order to prevent them. To do so, they need to identify factors that most influence the KPI so that they can act as effectively as possible. As previously mentioned, data used to forecast may contain some that have more significant influence than the decision-maker thinks. For this issue, an action prioritization can be provided by ranking the predictors, based on the final weights of the ANN that has been used to predict, as we suggest in this section. It should be noted that whatever the KPI of interest, the procedure remains the same, as long as a good predictive ANN is available for the KPI of interest.

The ANN weights represent the strength of connections between units of the ANN, and highlight the degrees of importance of the inputs values (Han et al. 2015). Figure 8 shows how inputs are transformed to provide the output in a single neuron ANN. A weighted sum is first performed, then the activation function is applied on it and summed with the bias value to give the output Y as:

$$Y = f\left(\sum_{i=1}^n (w_i x_i + b)\right) \quad (4)$$

where x_i is the i^{th} input value, w_i is the connection weight between x_i and the output, and b is the bias.

When dealing with a multilayer perceptron ANN, the same formula is applied for computing each hidden layer unit value, which then behave as inputs for the next layer, and so on. In our ranking method, we only exploit inputs and final weights values. Bias values are not used, since biases act in our case like additional neurones, one per layer. We therefore consider that their influence as equally distributed on the hidden neurons of the same layer. Their considering in our prioritization proposition will hence make the method more complex without changing the final ranking. Moreover or two neurons with the same value, the one with the biggest weight value will automatically most increase the weighted sum and hence the output, if the used activation function is increasing. Therefore, the equivalences given by 5 and 6 can be established.

$$\sum_{i=1}^n w_i x_i - \sum_{j=1}^n w_j x_j > \sum_{k=1}^n w_k x_k - \sum_{l=1}^n w_l x_l \iff f\left(\sum_{i=1}^n w_i x_i\right) > f\left(\sum_{j=1}^n w_j x_j\right) \quad (5)$$

$$\begin{aligned} \sum_{j=1}^n w_j x_j - \sum_{i=1}^n w_i x_i > \sum_{k=1}^n w_k x_k - \sum_{i=1}^n w_i x_i &\iff \\ f\left(\sum_{j=1}^n w_j x_j\right) - f\left(\sum_{i=1}^n w_i x_i\right) > f\left(\sum_{k=1}^n w_k x_k\right) - f\left(\sum_{j=1}^n w_j x_j\right) \end{aligned} \quad (6)$$

In the case of a neural network without a hidden layer, if the output Y given by 4 increases, this means that $f(\sum_{i=1}^n w_i x_i)$ has increased, since an increasing function is used. In this work, only increasing activation functions are supported. As seen in the previous section, all neural networks that can be built use a strictly increasing activation function. Therefore, we can omit the activation function in the evaluation of the influence of inputs on the output. Besides, the objective of our study is to evaluate the influence degree of the inputs, without specifying the direction of the relationship. Therefore, we consider the absolute values of the weights rather than their signed values. Moreover, the updating of weights in each epoch is based on the error value, and the objective of the learning algorithm is to minimize this error by modifying the weights. For each estimation, the quadratic error is given by 7.

$$e(W) = (\hat{y}_i - y)^2 \quad (7)$$

The cost function after the training with a given W weights matrix is:

$$E(W) = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y)^2 \quad (8)$$

where N is the training sample size, which is constant. The objective being to minimize $(\hat{y}_i - y)^2$, where $\hat{y}_i = f(\sum_{j=1}^n w_j x_{ji})$ if we neglect the intercept, with n the number of inputs, and x_{ji} the j^{th} input value of the observation i . Thus, the error function depends on both weights and inputs values. For the standard gradient descent learning algorithm, the weights are updated as following:

$$W_t = W_{t-1} - \lambda \nabla E(W_{t-1}) \quad (9)$$

where W_{t-1} is the weights matrix of the previous epoch. For the stochastic gradient descent learning, the weights are optimized after each observation as following:

$$W_t = W_{t-1} - \lambda \nabla e(W_{t-1}) \quad (10)$$

For the Quasi-Newton optimization, e.g BFGS algorithm, weights are updated as following:

$$W_t = W_{t-1} - \mu H_{t-1} \nabla E(W_{t-1}) \quad (11)$$

where H_{t-1} is an approximation of the Hessian matrix of the cost function.

Therefore, the gradient of the error is always involved in the weights updating, which clearly means that the values of the inputs are also involved, since the error function depends on \hat{y}_i , which depends on the inputs values. It is thus intuitive to conclude that for two predictors x_1 and x_2 with exactly the same influence on the output, if x_1 varies within a range of very large values, and x_2 within a range of very small values compared to x_1 , the weights assigned to x_1 would have an absolute value much smaller than the ones assigned to x_2 . The variation ranges of the two final weights matrices W_1 and W_2 will be inversely correlated with the variation ranges of the inputs x_1 and x_2 . Therefore, the formula we use in order to allocate an influence degree to each input i is given by 12, which corresponds to the contribution of an input i in the output variations.

$$R_i = \left| \frac{\mu_i}{\sigma_i} \left(\sum_{j=1}^m \frac{w_{ij}}{\sum_{i=1}^n |w_{ij}|} * \prod_{k=1}^m \frac{w_{jk}}{\sum_{j=1}^m w_{jk}} \right) \right| \quad (12)$$

where i is the input for which we are evaluating the strength of its association to the output, n is the number of the ANN inputs, m denotes the number of hidden neurons connected to the input i , w_{xy} denotes a connection weight between a neuron x and a neuron y , μ_i and σ_i denote the mean and the standard deviation of the values of input i . The standard deviation is used to take into account the dispersion of the values of the input, indeed, an input with relatively small values and which takes by moment very large values may have a large mean value which is not representative of the sample. The inverse of the coefficient of variation $\frac{1}{CV} = \frac{\mu}{\sigma}$ is then used, the bigger its value is, the smallest is the dispersion.

An influence degree is hence attributed to each input, and the ones that have the biggest R_i values are the ones that influence the most the KPI being predicted. But yet, the causal structure has to be taken into account in order to balance this ranking, as we will see in section 6. Thanks to this ranking, and after balancing it according to the causal structure ; when a KPI deviation is predicted, an action on the entity that corresponds to the variable with the highest influencing degree would be more likely to allow the deviation avoidance.

4.3. Validation process of the prioritization degrees

In order to ensure that the prioritization is correct for a given KPI, it should be compared on different ANNs that all perform good predictions of the KPI, and which obviously provide different final weights matrices since they have different structures and hyper-parameters. The suggested neural network construction method can be used to build several ANNs giving high prediction performances for the same KPI. The genetic algorithm is therefore run several times if needed, in order to provide ANNs with different structures and hyper-parameters, but with approximately the same performance. The resulting ANNs will then certainly have different numbers and values of weights. The ranking method can then be applied on each of these ANNs' final weights. Normally, the same order should be returned for all of these well performing ANNs. This process is shown in figure 9. Moreover, inputs can be successively omitted

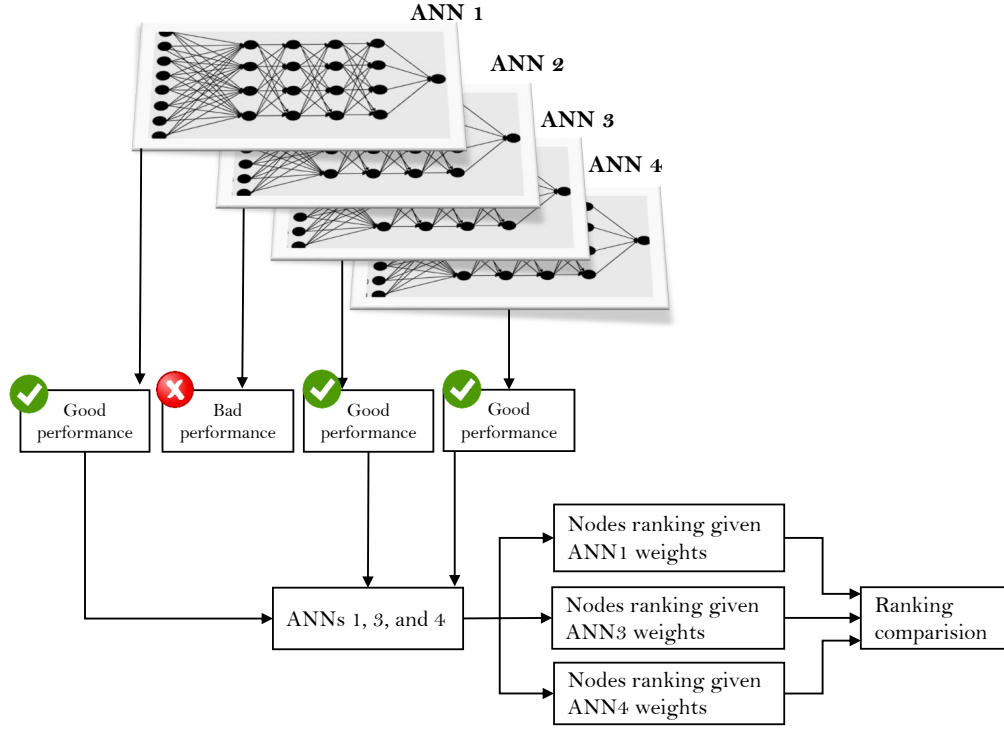


Figure 9. Validation process of the prioritization degrees.

one by one, and new ANNs can then be built for each case. The prediction performance should significantly decrease for the ANNs built without the top ranked inputs, and should not be much altered when omitting the last ranked inputs from the model.

5. Proposal validation

An academic industrial case study dealing with the OEE related to an assembly operation has been built in order to implement the proposed methodology and to evaluate the experiments' correctness. It is based on a representative summary data sample. Since our approach only deals with variables causally related to the KPI (*i.e.* to the OEE in this case study), and since we consider that a causal structure is available, the studied data sample has been obtained from a data simulation based on a causal Bayesian network in order to deal with a meaningful and realistic dataset. First, a causal structure has been defined, as well as its corresponding probability tables in order to generate a causal Bayesian network using `pgmpy` and `bnlearn` python libraries. Then, data have been sampled, based on the causal links and on the probability tables previously set, using the same libraries. In this use case, the authors consider that OEE deviation is causally related to the following discrete variables: waste, slow down, tool unavailability, the ambient temperature in the shop-floor, the used assembly guide, and the chosen scheduling of the assembly tasks. The expected output is a classification of whether the OEE will deviate or not. Figure 10 illustrates the causal Bayesian network structure describing causal links between

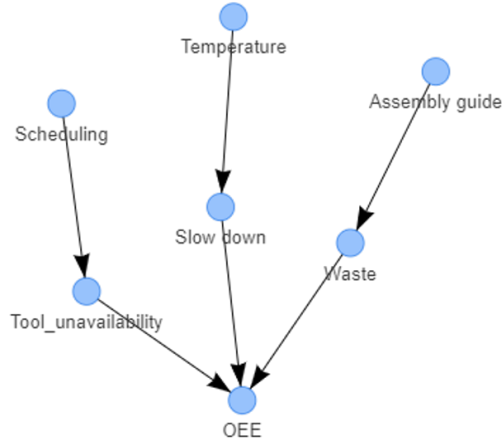


Figure 10. Causal Bayesian network structure describing causality links between contextual variables and OEE state.

| | | | | | | | | |
|-------------|--------------------|--------------------|-----------------|----------------------|--------------------|--------------------|--------------------|-----|
| | Assembly guide (1) | Assembly guide (2) | | Temperature (normal) | Temperature (high) | | Assembly guide (1) | 0.5 |
| Waste (no) | 0.1 | 0.9 | Slow down (no) | 0.1 | 0.9 | Assembly guide (2) | 0.5 | |
| Waste (yes) | 0.9 | 0.1 | Slow down (yes) | 0.9 | 0.1 | | | |

| | | | | | | | | |
|---------------------------|----------------|----------------|----------------------|-------------------|-----|----------------|----------------|-----|
| | Scheduling (1) | Scheduling (2) | | Temperatre (high) | 0.5 | | Scheduling (1) | 0.5 |
| Tool unavailability (no) | 0.1 | 0.9 | Temperature (normal) | 0.5 | | Scheduling (2) | 0.5 | |
| Tool unavailability (yes) | 0.9 | 0.1 | | | | | | |

| | | | | | | | | |
|-----------------|--------------------------|--------------------------|--------------------------|--------------------------|---------------------------|---------------------------|---------------------------|---------------------------|
| | Tool unavailability (no) | Tool unavailability (no) | Tool unavailability (no) | Tool unavailability (no) | Tool unavailability (yes) | Tool unavailability (yes) | Tool unavailability (yes) | Tool unavailability (yes) |
| Waste (no) | Waste (no) | Waste (yes) | Waste (yes) | Waste (no) | Waste (no) | Waste (yes) | Waste (yes) | |
| Slow down (no) | Slow down (yes) | Slow down (no) | Slow down (yes) | Slow down (no) | Slow down (yes) | Slow down (no) | Slow down (yes) | |
| OEE (normal) | 1 | 0.86 | 0.9 | 0.2 | 0.7 | 0.15 | 0.1 | 0 |
| OEE (Deviation) | 0 | 0.14 | 0.1 | 0.8 | 0.3 | 0.85 | 0.9 | 1 |

Figure 11. Probability tables associated with the OEE causal graph.

contextual variables and the OEE state. The probability tables associated to this causal structure are given by figure 11.

First, the genetic algorithm was developed and launched in order to provide an ANN with high predictive performance. Table 2 shows the accuracy evolution between ANNs constructed in the initial population, and ANNs constructed in the last population. The best performing network gives an accuracy of 0.8786% and its normalized confusion matrix is given in figure 12. The final ANNs built by the genetic algorithm provide good accuracies. Indeed, given the probabilities set for the OEE's status, the assertion that the OEE will or will not deviate can be certain at an average probability of 0.87625.

The general ranking, when predicting with all of the 6 causes is given by table 3. In order to ensure that this ranking is correct, the method described in figure 9 has been performed. The second best ANN of the last population was thus selected and the resulting ranking is described in table 4. Obviously, this ANN has a different

Table 2. ANNs accuracy evolution between the first and the last population.

| First population accuracies | Last population accuracies |
|-----------------------------|----------------------------|
| 0.4883666666666667 | 0.8785666666666665 |
| 0.5108666666666667 | 0.8686666666666667 |
| 0.4902666666666667 | 0.8745333333333334 |
| 0.4867333333333335 | 0.87648 |
| 0.7764666666666666 | 0.8757333333333334 |
| 0.6852 | 0.8738666666666667 |
| 0.7964666666666666 | 0.8782333333333333 |

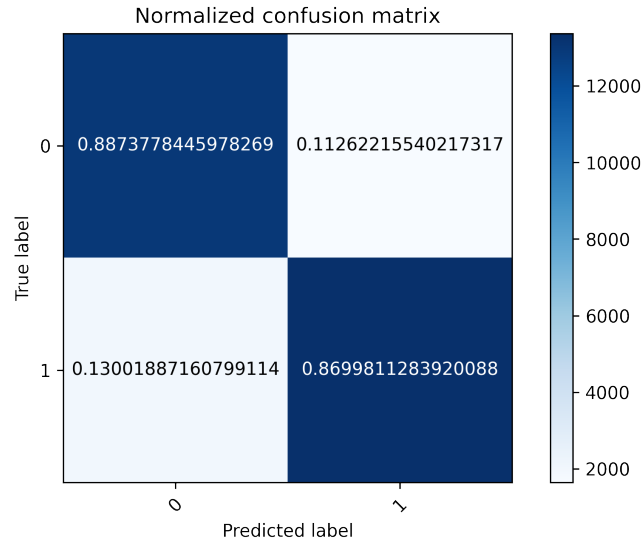


Figure 12. Prediction normalized confusion matrix of ANN resulting from the genetic evolution.

Table 3. Inputs general ranking.

| Ranking | Ranking score | Input |
|---------|--------------------|---------------------|
| #1 | 16.945206257137187 | Tool unavailability |
| #2 | 10.060149902028803 | Waste |
| #3 | 9.808118368056034 | Slow down |
| #4 | 9.500956633913196 | Temperature |
| #5 | 9.232680220598699 | Assembly guide |
| #6 | 9.14470575536555 | Scheduling |

Table 4. Inputs general ranking using the second best ANN.

| Ranking | Ranking score | Input |
|---------|--------------------|---------------------|
| #1 | 14.600708117842089 | Tool unavailability |
| #2 | 12.892838321870572 | Waste |
| #3 | 11.623951703243291 | Slow down |
| #4 | 8.885434822003866 | Temperature |
| #5 | 8.862161240203546 | Assembly guide |
| #6 | 8.805144359822698 | Scheduling |

structure, hyper-parameters, and weights from the first one. The resulting ranking gives the same prioritization. In this case, if a deviation of the OEE is predicted during the utilization phase, focus should preferably be placed on the analysis of the tool unavailability, if we only consider direct causes.

The ranking analysis has to take into account the level of a cause among other causes. The scores of direct causes can directly be interpreted, however, the analysis of the indirect causes ranking has to take into account the ranking of its related direct causes and its related descending indirect causes. In this case study, tool unavailability, waste and slow down are the direct causes, and assembly guide, scheduling, and temperature are at the first indirect causes level.

In order to illustrate the direct causes influences, a series of four ANNs has been built and its ANNs have been compared using the areas under the Receiver Operating Characteristic (ROC) curves. ROC curve is a technique for visualizing and selecting classifiers based on their performance (Hoo, Candlish and Teare 2017). The performance is evaluated by calculating the area under the curve (AUC). AUC is a very widely used measure of performance for classification and diagnostic rules (Airola et al. 2011). A first ANN has been built using all direct causes. The three remaining ANNs were built by omitting, one by one, and successively, one of the three direct causes initially used inputs. Then, the authors compared the performances of the predictions that have been run, by superimposing their ROC curves, and the performances of each prediction have been evaluated by calculating the AUCs. From the ROC curves in figure 13, one can clearly see that when altering the most influencing direct factor, *i.e.* Tool unavailability, the prediction is more altered than when altering Slow down or Waste.

Figure 13. ROC curves when predicting with all direct causes, and when omitting one input at a time.

Table 5. Direct causes ranking.

| Ranking | Ranking score | Input |
|---------|--------------------|---------------------|
| #1 | 14.196932965129683 | Tool unavailability |
| #2 | 12.500569730345381 | Waste |
| #3 | 11.050863041452459 | Slow down |

Concerning the comparison of the indirect causes between each other, their descending nodes scores have to be taken into account. In fact, considering the analysis of the scores given by table 3, and taking only direct scores into account, one can conclude that Tool unavailability has a contribution of 46% on the OEE state prediction, Waste contributes to 27,3%, and Slow down to 26.7%. The scores of the first level indirect causes should take into account these contributions. Hence, they should be weighted by their descending causes contributions, which gives the following scores when comparing the first level indirect causes : $9.14 \times 0.46 = 4.2044$ for Scheduling, since it is the parent of Tool unavailability, $9.23 \times 0.27 = 2.49$ for Assembly guide, since it is the parent of Waste, and $9.5 \times 0.26 = 2.47$ for Temperature, since it is the parent of Slow down. This brings the Scheduling to the top of the indirect causes ranking, and Temperature to its bottom.

6. Comparison with Bayesian sensitivity analysis

Remaining with the study case presented above, the objective of this section is to compare the resulting influences ranking with contribution percentages resulting from Bayesian sensitivity analysis. Indeed, the authors aim to evaluate the consistency of the proposal results, that are based on final learned ANN weights, with sensitivity analysis results, that are based on conditional probabilities tables associated to the original Bayesian network. This section also highlights difficulties and mistakes related to the interpretation of the conditional probabilities tables associated to a BN.

The general ranking has already been given by table 3. Two more ANNs have been built : a first one with the direct causes only, and a second with the indirect causes only. Tables 5 and 6 give the rankings of respectively the direct and indirect causes taken separately, given one built ANN. As shown in these tables, the rankings are coherent with the analysis the authors did in the previous section.

6.1. Predictors contributions based on a Bayesian sensitivity analysis

In order to correctly conclude on the predictors contributions ranking based on probabilities tables, without falling into misinterpretations, a sensitivity analysis has to be conducted. The resulting contributions will serve as a reference to verify the consis-

Table 6. Indirect causes ranking.

| Ranking | Ranking score | Input |
|---------|--------------------|----------------|
| #1 | 12.290120535676731 | Scheduling |
| #2 | 9.69051603768232 | Assembly guide |
| #3 | 9.6223850678669 | Temperature |

| Direct Effects on Target OEE | | | | | | |
|------------------------------|------------------|----------------------------|---------------|------------|--------------------|--------------|
| Node | Prior Value/Mean | Standardized Direct Effect | Direct Effect | Elasticity | Overall Elasticity | Contribution |
| Tool unavailability | 0.4991 | 0.5026 | 0.5025 | 50.2469% | 50.2469% | 40.0834% |
| Waste | 0.4994 | 0.3781 | 0.3780 | 37.8045% | 37.8045% | 30.1578% |
| Slow down | 0.4983 | 0.3731 | 0.3730 | 37.3047% | 37.3047% | 29.7589% |

Figure 14. OEE state direct contributions according to Bayesialab.

tency of our resulting rankings. Bayesialab ¹ software contains the sensitivity analysis functionality and can provide us with a complete analysis that responds to the problem of ranking the influencing factors. Since our objective is to be able to engage actions on root cause if possible, we first start by performing direct contributions of the OEE, in order to further investigate the contributions of causes of the most influencing direct cause. Direct contributions on OEE state, according to Bayesialab sensitivity analysis are presented in figure 14. The last column of figure 14 shows that Tool unavailability has more impact on OEE (40.08%) than Waste and Slow down (respectively 30,16% and 29.75%).

In order to compare in a meaningful way the Bayesialab results to the ones provided by the approach proposed in this paper, the rankings must be expressed in percentages. Direct causes contributions percentages according to the ranking approach proposed in this paper can be derived from table 5, by dividing each cause's score by the sum of scores of direct causes. Therefore, contributions of direct causes, according to our approach are : Tool unavailability contributes to 37.61%, Waste contributes to 33.11%, and Slow down 29.38%. These results are consistent with the result provided by the Bayesialab analysis, and contributions percentages provided by our approach are close to ones provided by the Bayesialab analysis, even though they are not the same. This gaps are due to the fact that final weights on which the ranking is based are issued from an ANN which still has a margin of error. Indeed, the proposed method relies on final connection weights of ANN, but one has to keep in mind that this ANN is not 100% accurate, it hence has learned some errors in the training phase where ANN did not see all of the dataset samples, but only the training set samples. This leads to very representative weights matrix when accuracy is high enough, but nearly never faithful to 100% of reality interactions.

In the same way, indirect causes influences can be taken separately in order to compare their influences. Indirect contributions on OEE state, according to Bayesialab sensitivity analysis are presented in figure 15. The last column of figure 15 shows that Scheduling has more impact on OEE (39.99%) than Assembly guide and Temperature (respectively 30.19% and 29.81%). These contributions are in accordance with the

¹ www.bayesia.com

| Direct Effects on Target OEE | | | | | | |
|------------------------------|------------------|----------------------------|---------------|------------|--------------------|--------------|
| Node | Prior Value/Mean | Standardized Direct Effect | Direct Effect | Elasticity | Overall Elasticity | Contribution |
| Scheduling | 0.5010 | -0.4015 | -0.4014 | -40.1439% | -40.1439% | 39.9973% |
| Assembly guide | 0.4999 | -0.3031 | -0.3030 | -30.3037% | -30.3037% | 30.1930% |
| Temperature | 0.5019 | -0.2993 | -0.2992 | -29.9192% | -29.9192% | 29.8097% |

Figure 15. OEE state indirect contributions according to Bayesialab.

| Total Effects on Target OEE | | | | | | | | | |
|-----------------------------|------------------|----------------------------|---------------|-------------|----|---------|---------------|-----------|----------------|
| Node | Prior Value/Mean | Standardized Total Effects | Total Effects | G-test | df | p-value | G-test (Data) | df (Data) | p-value (Data) |
| Tool unavailability | 0.4991 | 0.5026 | 0.5025 | 79,353.4513 | 1 | 0.0000% | 79,552.5125 | 1 | 0.0000% |
| Scheduling | 0.5010 | -0.4017 | -0.4016 | 49,795.2717 | 1 | 0.0000% | 49,777.0789 | 1 | 0.0000% |
| Waste | 0.4994 | 0.3781 | 0.3780 | 43,984.1295 | 1 | 0.0000% | 43,901.9330 | 1 | 0.0000% |
| Slow down | 0.4983 | 0.3731 | 0.3730 | 42,799.9576 | 1 | 0.0000% | 42,882.2319 | 1 | 0.0000% |
| Assembly guide | 0.4999 | -0.3020 | -0.3020 | 27,801.9960 | 1 | 0.0000% | 27,810.6592 | 1 | 0.0000% |
| Temperature | 0.5019 | -0.2983 | -0.2982 | 27,106.6241 | 1 | 0.0000% | 27,043.3632 | 1 | 0.0000% |

Figure 16. Effects of all causes on the OEE according to Bayesialab.

ranking given by table 6, which gives contribution to 38.9% for Scheduling, 30.65% for Assembly guide, and 30.45% for Temperature. The Baysialab ranking results are also consistent with the analysis driven in the last paragraph of section 5 based on the general ranking given by table 3.

So far in this section, the authors have deal with direct causes and indirect causes separately. Let us now consider the whole ranking of the effects of all causes on the OEE state. The analysis of the general ranking is important for the alternative selection phase of the decision making process. In fact, decision makers may want to compare causes with different levels, especially when they judge that acting on a given root cause would be effort or resource consuming, and that it would be interesting to compare its descendent(s) with another root cause(s). Figure 16 shows, in the fourth column, the respective effects of each cause on the OEE. When comparing with the authors' methodology, one must consider the absolute value of these effects, since the authors do not deal in the present scope with the influencing directions of the causes on the addressed KPI. In order to compare these contributions with our ranking analysis, we first bring the effects values of the sensitivity into percentages, which gives the following contributions : 22.28% for Tool unavailability, 17% for Scheduling, 16.76% for Waste, 16.53% for Slow down, 13.4% for Assembly guide, and 13.22% for Temperature.

Since the general ranking given by table 3 involves direct and indirect causes together, the causal structure plays a key role in its analysis. In fact, weights are attributed in order to minimise the error in the output, but given the causal structure, it is obvious that weak weights would be attributed to connections related to a causal parent of another input which has already been given a high weight. For example, since Tool unavailability highly contributes to the OEE state change, its parent Scheduling is no more important for the prediction, if and only if Tool unavailability is present in the dataset ; but this does not mean that Scheduling is not highly related to the OEE in a causal way. Therefore, when dealing with prioritization between causes of different levels, the causal structure is indispensable for analysing the ranking given by the proposed methodology. In our case study, this means that

in the general ranking, we shall first calculate the contributions of causes of different levels separately given the scores of the general ranking given by table 3. These contributions will then allow us to scale all of the scores, in order to provide the final contributions of all causes of mixed levels. Concerning the direct causes, we already got, in the previous section the following contributions percentages : Tool unavailability contributes to 37.61%, Waste contributes to 33.11%, and Slow down 29.38%. The ranking of indirect scores given by table 3 has to be weighted with the contributions of direct causes related to each indirect cause, before comparing the indirect causes contributions between each other, as previously explained. Hence, the new weighted scores of indirect causes are: 4.2044 for Scheduling, 2.49 for Assembly guide, and 2.47 for Temperature. The reverse operation must then be done in order to take into account the contributions of indirect causes of level one. The ranking of direct scores, given by table 3, has to be weighted with the contributions of indirect causes related to each direct cause. First, the contributions of indirect causes should be compared to each other, independently from the direct causes. These contributions can be retrieved from table 3 by dividing each indirect cause score by the sum of indirect cause scores, which gives : $9.5 \div (9.5 + 9.23 + 9.14) = 34.09\%$ for Temperature, $9.23 \div (9.5 + 9.23 + 9.14) = 33.12\%$ for Assembly guide, and $9.14 \div (9.5 + 9.23 + 9.14) = 32.8\%$ for Scheduling. The new weighted scores of indirect causes are : $16.94 \times 0.328 = 5.554$ for Tool unavailability since it is the descendent of Scheduling, $10.06 \times 0.3312 = 3.332$ for Waste since it is the descendent of Assembly guide, and $9.81 \times 0.3409 = 3.344$ for Slow down since it is the descendent of Temperature.

Once the general ranking scores are weighted given the causal structure, their contribution percentages can be retrieved by dividing each weighted score by the sum of all other weighted scores, which gives the following final contributions: 25.96% for Tool unavailability, 19.65% for Scheduling, 15.63% for Slow down, 15.57% for Waste, 11.64% for Assembly guide, and 11.54% for Temperature. This ranking and contribution percentages are very close to the ones given by the effects analysis provided by Bayesialab. Positions of Waste and Slow down in the proposed general ranking analysis and Bayseialab Analysis are reversed. However the values of Waste and Slow down contributions are very close to each other in both analyses (15.63% for Slow down and 15.57% for Waste in the proposed general ranking analysis, and 16.76% for Waste, 16.53% for Slow down, in the Bayesialab analysis). This reversing is therefore not critical, since it held at less than 0.03%.

Given this final ranking, causes of different levels involved in the OEE state can be compared to each other. For example, if we compare Scheduling, Waste, and Slow down, Scheduling proves to be more influential than the Waste or Slow down, even though it is an indirect cause. In order to make sure of this assertion, an ANN taking as predictors only Scheduling, Waste, and Slow down has been built using the ANN construction GA. Three more ANNs have also been generated by omitting one of these three predictors one by one. Figure 17 illustrates the impact of omitting each of Scheduling, Waste, and Slow down from the predictors. The ROC curves and the AUCs shown in this figure show that omitting scheduling has larger impact than omitting Waste or Slow down.

Figure 17. Comparison of the predictive power contribution of Scheduling, Waste, and, Slow down.

6.2. Discussion

Despite the little changes in percentages resulting from our proposal, the ranking values give the exact order with coherent tendencies. Our initial objective being to inform decision makers about the most relevant causes to act on, we consider that the proposal gives results that are able to fulfil this function, and that it offers genericity and easy applicability on different KPIs. We hence fulfilled genericity criteria, indispensable in industrial contexts where KPIs are likely to evolve and to be numerous. Furthermore, we believe that the proposed method can be considered as being convenient since it addresses the ease and rapidity of use, that are also important in industrial contexts where decision makers do not have time or the required background to interpret probabilities tables. Also, the decision maker would not need to spend much time for getting used to the tool handling, since he will only need to inject historical data of the variables present in the causal structure in order to learn a good forecasting ANN and to get the scores ranking of the causes and root causes. We also showed that decision makers can either decide by choosing an alternative related to one cause among other causes of the same level, or to one cause among others of several levels. Also, usability is fulfilled since the prediction using the learned ANN will enable a proactive decision making in order to handle a potential deviation before it occurs, by acting on the most influencing cause(s). Furthermore, ANNs can give high prediction accuracy and can handle both quantitative and qualitative data. Besides, for future internal developments, there will be no need for purchasing software that provide causal contributions.

7. Conclusion and future work

In this article, we introduced neural a network based decision making approach for supervision in the context of the Industry 4.0. In this context, the availability of data can be profitable in the sense that we exploit it to take decisions beyond the beliefs arising from the experience. The main idea was to collect as much data as possible in order to predict a given KPI. The prediction is based on an ANN that also serves, thanks to its final weights, for prioritizing the influencing factors. This ANN is built using an evolution process based on genetic algorithms, which provided genericity to our proposal since we can easily define ANN structure and hyper-parameters for predicting any KPI. A ranking method based on ANN final weights was then presented in order to provide decision aid that pinpoints causes on which actions should preferably be engaged. The contributions results are coherent with Bayesian Sensitivity analysis, and present the advantage of being easily retrieved without manipulating the algorithm. This prioritization can also serve to better exploit data sources, by using the last ranked inputs' sensors for other aims rather than supervising a KPI that it impacts a little or not at all. Future work should be centred on utilization phase in real time, by exploiting the ANNs and the prioritization of contributors of a given KPI, in order to raise alarms and suggest actions in real time and before deviation occurs. This will hence need a real industrial environment and its corresponding real historical data. Furthermore, the proposal should also be enriched by adding a cause ranking enrichment component which consists of exploiting the feedback after each decision made

and action taken. This can be done by evaluating the effect of an action on a given cause, on the process, and then balancing the cause ranking. This balancing will then either support the ranking and increase its level of confidence, or disapprove it and decrease its level of confidence. Finally, the results computing in order to rank causes from different levels shall be automated and integrated to the ranking algorithm.

References

- Ahmadizar, Fardin, Soltanian Khabat, Akhlaghiantab Fardin and Tsoulos Ioannis. 2015. "Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm." *Engineering Applications of Artificial Intelligence* 39: 1–13.
- Airola, Antti, Pahikkala Tapio, Waegema Willem, De Baets Bernard and Salakoski Tapio. 2015. "An experimental comparison of cross-validation techniques for estimating the area under the ROC curve". *Computational Statistics & Data Analysis* 55 (4) : 1828–1844.
- Amzil, Kenza, Yahia Esma, Klement Nathalie and Roucoules Lionel. 2020. "Causality learning approach for supervision in the context of Industry 4.0". Paper presented at tenth Joint Conference on Mechanics.
- Aydiner, Arafat Salih, Tatoglu Ekrem, Bayraktar Erkan, Zaim Selim and Delen Dursun. 2021. "Business analytics and firm performance: The mediating role of business process performance". *Journal of business research*. 96: 228–237.
- Ballard, Andrew. 2019. "Framing bias in the interpretation of quality improvement data: evidence from an experiment". *International Journal of Health Policy and Management*. 8: 307.
- Bramer, Max. 2007. *Principles of Data Mining*.
- Dalzochio, Jovani, Kunst Rafael, Pignaton Edison, Binotto Alecio, Sanyal Srijnan, Favilla Jose and Barbosa Jorge. 2020. "Machine learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges" *Computers in Industry* 123: 103298.
- Hagan, Martin T., Demuth Howard B., Hudson Beale Mark and De Jesús Orlando. 2014. *Neural Network Design, second edition*.
- Han, Hong-Gui, Zhang Shuo and Qiao Jun-Fei. 2017. "An adaptive growing and pruning algorithm for designing recurrent neural network". *Neurocomputing* 242: 51–62.
- Han, Song and Pool Jeff, Tran John and Dally William. 2015. "Learning both weights and connections for efficient neural network". *Advances in neural information processing systems* : 1135–1143.
- Hänninen, Maria and Kujala Pentti. 2012. "Influences of variables on ship collision probability in a Bayesian belief network model." *Reliability Engineering & System Safety* 102: 27–40.
- Hoo, Zhe Hui, Candlish Jane and Teare Dawn. 2017. "What is an ROC curve?"
- Huang, Binbin, Wang Wenbo, Ren Shan, Zhong Ray Y and Jiang Jingchao. 2019. "A proactive task dispatching method based on future bottleneck prediction for the smart factory." *International Journal of Computer Integrated Manufacturing*. 32 (3):278-293.
- Kalainathan, Diviyan. 2019. "Generative Neural networks to infer Causal Mechanisms : Algorithms and applications." PhD diss. Université Paris-Saclay.
- Kamienski, Carlos, Soininen Juha-Pekka, Taumberger Markus, Dantas Ramide, Toscano Attilio, Salmon Cinotti Tullio, Filev Maia Rodrigo and Torre Neto André. 2019. "Smart water management platform: Iot-based precision irrigation for agriculture." *International journal of communication systems* 19 (2): 276.
- Karsoliya, Saurabh. 2012. "Approximating number of hidden layer neurons in multiple hidden layer BPNN architecture". *International Journal of Engineering Trends and Technology* 3 (6): 714–717.
- Keding, Christoph and Meissner Philip. 2021. "Managerial overreliance on AI-augmented decision-making processes: How the use of AI-based advisory systems shapes choice behavior in R&D investment decisions". *Technological Forecasting and Social Change* 171 :

- 120970.
- Kjærulff, Uffe and Van Der Gaag Linda C. 2013. "Making Sensitivity Analysis Computationally Efficient". *UNCERTAINTY IN ARTIFICIAL INTELLIGENCE PROCEEDINGS*: 317–325.
- Klingenberg, Cristina Orsolin, Borges Marco Antônio Viana and Antunes Jr José Antônio Valle. 2019. "Industry 4.0 as a data-driven paradigm: a systematic literature review on technologies". *Journal of Manufacturing Technology Management*.
- Laudon, Jane P. and Laudon Kenneth C. 2019. "Management Information Systems: Managing the Digital Firm."
- Mantzaris, Dimitrios, Anastassopoulos George and Adamopoulos Adam. 2011. "Genetic algorithm pruning of probabilistic neural networks in medical disease estimation". *Neural Networks* 24 (8): 831–835.
- Marcot, Bruce G. 2017. "Common quandaries and their practical solutions in Bayesian network modeling". *Ecological Modelling* 358: 1–9.
- Misirli, Ayse Tosun and Bener Ayse Basar. 2014. "Bayesian networks for evidence-based decision-making in software engineering." *IEEE Transactions on Software Engineering* 40 (6):533-554.
- Moeuf, Alexandre, Pellerin Robert, Lamouri Samir, Tamayo-Giraldo Simon and Barbaray Rodolphe. 2017. "The industrial management of SMEs in the era of Industry 4.0." *International Journal of Production Research* 56 (3): 1–19.
- Panchal, Gaurang, Ganatra Amit, Kosta Y P and Panchal Devyani. 2011. "Behaviour Analysis of Multilayer Perceptrons with Multiple Hidden Neurons and Hidden Layers". *International Journal of Computer Theory and Engineering* 3 (2): 332–337.
- Pérez-Álvarez, José Miguel, Maté Alejandro, Gómez-López María Teresa and Trujillo Juan. 2018. "Tactical Business-Process-Decision Support based on KPIs Monitoring and Validation". *Computers in Industry* 102 : 23–39.
- Pollino, Carmel A. and Henderson Christian. 2010. "Bayesian networks: A guide for their application in natural resource management and policy". *Ecological Modelling* 14.
- Sheela, K. Gnana and Deepa Subramaniam N. 2013. "Review on methods to fix number of hidden neurons in neural networks". *Mathematical Problems in Engineering* 2013.
- Shih-Hung, Yang and Yon-Ping Chen. 2012. "An evolutionary constructive and pruning algorithm for artificial neural networks and its prediction applications". *Neurocomputing* 86: 140–149.
- Smith, Leslie N. 2018. "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay".
- Stanley, Kenneth O, Clune Jeff, Lehman Joel and Miikkulainen Risto. 2019. "Designing neural networks through neuroevolution". *Nature Machine Intelligence* 1 (1): 24-35.
- Tambare, Parkash, Meshram Chandrashekhar, Lee Cheng-Chi, Ramteke Rakesh Jagdish and Imoize Agbotiname Lucky. 2022. "Performance Measurement System and Quality Management in Data-Driven Industry 4.0: A Review." *Sensors* 22 (1): 1424-8220.
- Tsai-Chi, Kuo, Hsu Ni-Ying, Yi Li Tzu and Chao, Chin-Jung. 2021. "Industry 4.0 enabling manufacturing competitiveness: Delivery performance improvement based on theory of constraints." *Journal of Manufacturing Systems* 60:152-161.
- Tu, Jack V. 1996. "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes." *Journal of Clinical Epidemiology* 49 (11):1225–1231.
- Weber, Philippe, Medina-Oliva Gabriela, Simon Christophe and Iung Benoît. 2012. 'Overview on Bayesian networks applications for dependability, risk analysis and maintenance areas". *Engineering Applications of Artificial Intelligence* 25 (4): 671–682.
- Yin, Shen, Zhu Xiangping and Kaynak Okyay. 2015. "Improved PLS focused on key-performance-indicator-related fault diagnosis." *IEEE Transactions on Industrial Electronics* 62 (3) : 1651–1658.
- Yu, Dong and Seltzer Michael L. 2011. "Improved bottleneck features using pretrained deep neural networks". Paper presented at the twelfth annual conference of the international

speech communication association.