

# A league-winner algorithm for defect classification in an industrial web inspection system

Angel Gaspar Gonzalez-Rodriguez<sup>a,\*</sup>, Antonio Gonzalez-Rodriguez<sup>b</sup>, Fernando Jose Castillo-Garcia<sup>b</sup>

<sup>a</sup>*Department of Electric Engineering and Automation, University of Jaen, Campus Las Lagunillas s/n, 23071 Jaen, Spain +34 953 212457*

<sup>b</sup>*School of Industrial Engineering, University of Castilla-La Mancha. Av. Carlos III s/n, 45071, Ciudad Real, Spain*

---

\*Corresponding author

*Email addresses:* `agaspar@ujaen.es` (Angel Gaspar Gonzalez-Rodriguez),  
`antonio.gonzalez@uclm.es` (Antonio Gonzalez-Rodriguez),  
`fernando.castillo@uclm.es` (Fernando Jose Castillo-Garcia)

---

## Abstract

This paper presents a modification to be added to multiclass classifiers, that improves their performance when classifying, in this case, defects appearing in polyethylene films. It aims to classify a new defect by confronting every defect type against each of the other types. In a simplified way, the type that results winner in more matches is the type that the defect belongs to. Different ways of implementing neural networks have been tested, using Gradient Descent and Levenberg-Marquardt techniques for backpropagation. These techniques have been formally and understandably explained. In addition, a method based on decision trees has been included for comparison. Different issues related to the practical implementation of the detection and identification system within an installed production chain are addressed. The resulting system has been incorporated as a real inspection automatism in a polyethylene manufacturing line, and trained with defects previously obtained from the same line.

*Keywords:* Pattern recognition, pairwise comparison, artificial neural networks, decision tree, supervised training, gradient descent, Levenberg-Marquardt

---

## 1. Introduction

### 1.1. *Necessity for an automated inspection system*

The need for quality control in the industrial processes ensures that the features or properties of the final products meet the metrics offered by the manufacturer, and that customers receive products according to their requirements. In certain cases, such control is performed outside the production process, for example when the impact resistance or the operation as a whole of an assembled part must be tested. In others, the inspection system must detect and identify the defects on-line, and act according to the severity or frequency of the detected defects or deviations.

Another concern to take into account is the tolerance to defect existence, their severity and origin. If a certain amount of defective elements is admitted, the defect severity is not critical, or the source of the defect may be found in a deviation of the characteristics of the raw material or in the operation of some  
15 machine, then statistical tools can be used for the quality assurance. Two main statistical approaches are used, with different objectives and procedures (Juran & Godfrey, 1998):

- Statistical process control (SPC) that monitors, collects and plots continuous variables of interest or certain production metrics, in which a deviation  
20 from the rated values usually leads to a final product with characteristics different from the desired ones. Including the quality control as an integral part of the production process reduces the response time in case of finding a critical problem; makes possible to supervise the process operation; and allows performing corrective actions to guarantee acceptable quality levels  
25 (Paladini, 2000)(Zhang, 1996). It usually fits in process industry.
- Acceptance sampling, that inspects a portion of a produced lot and deduces whether to accept or reject it (Carolino & Barão, 2013), and/or modify the process to avoid future rejections. It naturally fits in the manufacturing industry.

30 There are, however, circumstances in which statistical methods are not feasible, or at least not the only one to be applied. There are several reasons to include other inspection techniques:

- Quality in the manufactured product is determined, not only by the fulfilment of production metrics (composition, width, colour...) but also by  
35 the absence of defects. This is the case of the appearance of defects due to the presence of occasional foreign bodies (e.g. insects) during the manufacturing process.
- In certain industrial processes it is not possible to extract random samples from the manufactured product, like in a continuous roll of fixed-width

40 film/paper. There are also situations in which a product with zero defects, or at least zero critical defects, must be guaranteed, and statistical sampling is not dependable in these cases.

The application presented in this paper is a Web Inspection System (WIS) for the supervision of PolyEthylene Terephthalate (PET) rolls used in the manufacturing of trays for the food sector. This inspection system combines these  
45 two conditions mentioned above:

- The quality control must be integrated as a part of the production stage, because it is necessary for the material to be inspected before being rolled. It is not feasible to extract samples from the material being rolled in order  
50 to inspect them off-line.
- A SPC is not possible, or at least not as the only supervision system. On the contrary, the complete run must be inspected in order to detect in time the presence of foreign bodies, especially insects and holes, and to remove/mark the stretch involved.

55 Under these constraints, a continuous complete inspection is required to assure that all imperfections or defects must be, if not avoided, at least detected, classified and quantified. This way, the location of certain imperfections can be indicated to the customer, or in case of critical situations, the roll can be discarded.

60 Despite the importance of including an automated inspection system to ensure the quality of the final product, the practical aspects of this concern have not been addressed in the literature on visual inspection in the profusion that one would expect. Among the investigations carried out, most refer to the automobile industry and electronic components. In any case they correspond to  
65 the manufacturing industry where matrix cameras are usually the preferred option. The defect detection, rather than classification, was the inspection system objective.

A review of different defect identification methods can be found in (Karimi

& Asemani, 2014). The simplest approach compares the final product to a  
70 defect-free reference image, named as golden template.

Inclusion of the automated inspection system in the process industry is less  
often, with some exceptions as those installed at paper, textile (Cho et al.,  
2005) or plastic film industry Kuo et al. (2018) receiving the general name of  
Web Inspection Systems (WIS). In this case, the linear cameras are the most  
75 suitable option for a continuous production regime (Cho et al., 2005). Presence  
of some defects is often unavoidable, and the inspection system task mainly  
lies in classifying, counting and advising in case of critical situation (excessive  
number or appearance of bugs/holes). With the exception of several patents, no  
bibliography has been found related to WIS implemented and working in real  
80 process industry.

### *1.2. State of art of defect classifiers*

Different methods have been used to classify the defects, although the main  
important ones are: decision trees (DT), artificial neural networks (ANN) and  
support vector machines (SVM).

85 A decision tree is a dichotomous flowchart-like structure composed of mul-  
tiple decisions or rules, which are solved as a function of one of the features (or  
combination of them) calculated for all of the training data. Both outcomes of  
each decision element are named as branches, that in turn can be divided (after  
corresponding decisions) in successive branches. A branch will be considered  
90 a final one, and will be named as leaf, when all the instances that satisfy the  
sequence of decisions that lead to a branch belong to the same class.

A survey of usual techniques for constructing Top-down induction of decision  
tree classifiers is presented in (Rokach & Maimon, 2005). Authors collected the  
idea from most researcher that neither the impurity-based splitting criteria nor  
95 the pruning method is a determinant factor in the classifier performance.

Performance of DTs are compared to other classifiers in (Ooi et al., 2013),  
finding that more important than the technique used, is the selection of ap-  
propriate geometrical features. This can be applied to, not only DTs, but any

machine learning for image classification.

100 Among the DT classifiers, the RIPPER algorithm (Cohen, 1995), and those based on it, is composed of simple and intuitive rules. Assuming  $nc$  sorted classes (typically in ascending order according to size/probability), these rules are in the following form

```

105 IF conditions1 THEN class is 1
ELSE IF conditions2 THEN class is class(2)
...
ELSE IF conditionsr THEN class is class(r)
... nr rules
ELSE IF conditionsnr THEN class is class(nr) = nc - 1
110 ELSE class is nc
with class(r + 1) ∈ {class(r), class(r) + 1} ∀ r < nr and 1 ≤ class(r) < nc ∀ r ≤ nr

```

115 An extension of Ripper to tackle with multiclass classifications problems is presented in (Hühn & Hüllermeier, 2009), using fuzzy rules instead of conventional ones.

Another extension to multiclass problems is found in (Asadi & Shahrabi, 2016), that among other improvements, eludes the necessity of learning all the rules for a class until it is exhausted, and so a parallel learning is carried out that jumps over unrestricted classes. This feature prevents learning from being

120 biased toward a certain class or classes. The structure was the following one:

```

IF conditions1 THEN class is c1
ELSE IF conditions2 THEN class is c2
ELSE IF conditions3 THEN class is c3
... (nr rules)
125 ELSE class is cnc

```

with  $c_i \in \{1, 2, \dots, nc\} \forall i \in \{1, \dots, nr\}$ .

Another example of DT based classifier is found in (Ravikumar et al., 2011) that used DTs to distinguish between three grades (severity) of defects, with a mis-classification of 9.6%.

130 However, the technique that is usually preferred for the task of pattern recognition or image classification uses Artificial neural networks (ANNs) (Bishop, 1995)(Karabatak & Ince, 2009).

As an example, (Ferreira & Giraldi, 2017) uses Convolutional Neural Networks (CNNs), with filters to highlight some patterns such as edges or textures, this way generating a *feature map*. CNNs were also used in (Affonso et al., 2017) 135 to classify wood boards, and listed several works related to image classification together with the technique used.

Different methods for backpropagation are compared in (Mukherjee & Routroy, 2012), with the background of finding the empirical relationship between five response characteristics, two input variables and seven in-process variables in 140 order to model a multiple response grinding process. The methods are: Levenberg-Marquardt (LM), quasi-Newton (QN) and adaptive learning (AL) finding that: LM and QN algorithm-based ANNs are equally efficient as AL ones to predict the behaviour of multiple response grinding process; and LM algorithm 145 has fastest network convergence rate, followed by QN and AL algorithm.

Another method is the Support Vector Machine (SVM), which aims to maximize the margin between defects of two classes to find a separating hyperplane (Xiang et al., 2018),(Alpaydin, 2010) that confidently classify future defects. Many modification exists to tackle with the multi-class problem.

150 The performance of ANNs, SVMs and nearest-neighbour algorithm are compared in (Valavanis & Kosmopoulos, 2010) for the classification of weld defects, showing that first two methods are superior in terms of accuracy (85% for both classifiers against 60% of the third one). The experiments also stated the convenience of increasing the number of features to be used as classifier inputs.

### 155 1.3. Pairwise comparison

A DT algorithm, and up to three ANN ones have been programmed and compared. Two ANNs from the Neural Network Matlab Toolbox ('trainlm' N outputs, and single output) have been also used and compared. However, except for one of the toolboxes, a significant percentage of the test defects (items that

160 were not used to train the classifier) were not adequately classified.

This way, a novel modification to these classifiers was introduced, consisting in confronting each type of defect with the remaining ones, as in a competition league. For  $nc$  types of defects, the training phase requires obtaining  $nc(nc - 1)/2$  classifiers, one per each possible couple of matches. Assuming, e.g. a  
165 hole-type defect, it is likely that the class  $< hole >$  will be the winner at its  $nc - 1$  matches, and will obtain the maximum score. This situation is typical in cases where a grading of a subjective feature must be made, as explained in a very understandable fashion in (Thurstone, 1927). This technique receives the name of *pairwise comparison* (PC), and is often used in the scientific study of  
170 preferences, attitudes, voting systems, or social choice, with the name of *paired comparison*. As an example, (Dalitz et al., 2018) obtained continuous sentiment scores for words from comparisons made by test persons. It has been also used in physiology studies, as in (Amlani & Schafer, 2009) that recommended procedures for implementing the method of PC in the fitting of sophisticated  
175 hearing aids.

It is worth clarifying the following terms that can lead to confusion:

**Attribute, feature, descriptor and parameter.** An attribute is a characteristic extracted from the defect (length, or area of dark pixels); a feature is a value obtained from the defect after some operations (standard deviation of intensity) or attribute combination. These are the two sort of  
180 descriptors that will be used. A descriptor implies the nuisance of attribute/feature quantification. Parameters are the numerical value of the descriptors.

**Instance, defect.** Instance is a single object used for training, or a single object that will be eventually classified. In this work, instances are defects  
185 appearing in a PET roll.

**Class, type.** Class is a category or set to what an instance can belong. In this work, classes correspond to the defect types.

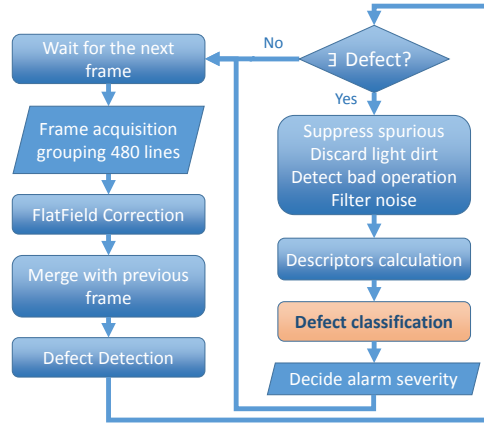


Figure 1: Tasks to be accomplished by the inspection system

## 2. Description of the process and the visual inspection system

As aforementioned, the application is a visual inspection system to detect and classify defects appearing in a production line of PET (see Fig. 1). The inspection system (see Fig. 2 and 3) basically consists of the following items:

- A transmission lighting system with high-power leds, in which light source and camera are in opposite sides of the inspected object.
- A line-scan camera ( $1 \times 16384$  px), preferred to matrix cameras for web inspection.
- Camera Link cables with two repeaters to overcome a distance of 30 m.
- The processing unit made up of an industrial PC, a frame-grabber and an I/O card.
- Additional components as power-electronic cards or visualization columns.

The system is able to detect and identify eight types of defects: gels, holes, black specks, insects, wrinkles, rubber, fish-eyes and oil drops (see Fig. 4). The system must also disregards slight dirt or imperfections as well as acquisition/transmission errors. In overall, the system must classify the potential defects into **ten classes**.

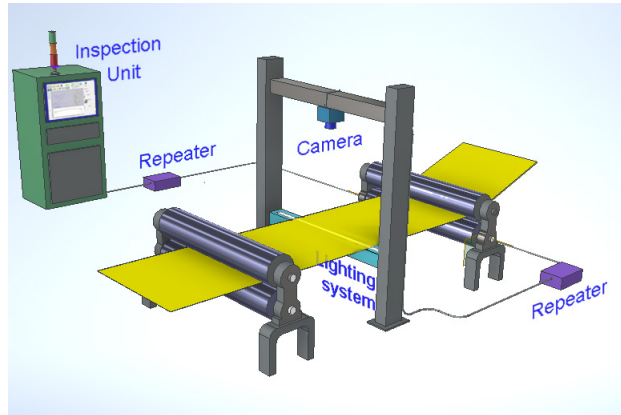


Figure 2: Scheme of the Web Inspection System

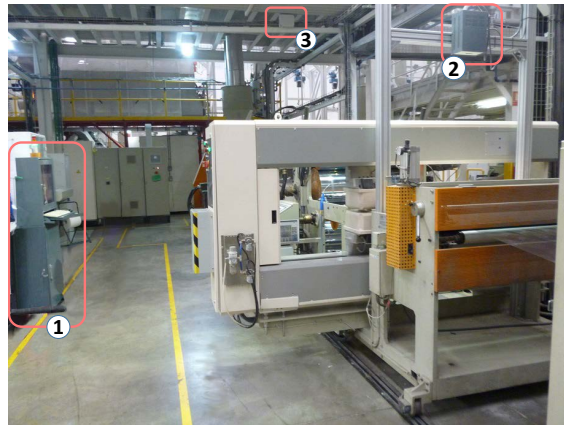


Figure 3: Location of the main components in the vision system: ① Processing unit, ② Camera, ③ Repeater 1 (repeater 2 is not shown in the picture)

Table 1: Quantity of defects used during the training and test procedure.

| Size (mm <sup>2</sup> ) | [1.4, 4) | [4, 9) | [9, 25) | >25 | Total |
|-------------------------|----------|--------|---------|-----|-------|
| Gel                     | 228      | 129    | 15      | 13  | 385   |
| Hole                    | 34       | 221    | 52      | 22  | 329   |
| Black sp                | 167      | 140    | 37      | 15  | 359   |
| Insect                  | 6        | 74     | 65      | 50  | 195   |
| Wrinkle                 | 35       | 66     | 78      | 50  | 229   |
| Rubber                  | 104      | 232    | 121     | 52  | 509   |
| Fish eye                | 15       | 60     | 16      | 2   | 93    |
| Oil Drop                | 0        | 0      | 0       | 7   | 7     |
| Dirt                    | 181      | 58     | 48      | 11  | 298   |
| Error                   | 57       | 112    | 115     | 78  | 362   |
| Total                   | 827      | 1092   | 547     | 300 | 2766  |
| Train                   | 703      | 928    | 464     | 255 | 2351  |
| Test                    | 124      | 164    | 82      | 45  | 415   |

Required by the customer, defects must be grouped according to their size. In general, the smaller defects, the more frequent ones. Table 1 lists the area range for each one as well as the number of defects used in the experiments. Last columns refers to the total number of defects, regardless of their size. From  
210 the set of available images, 85% has been used to train the algorithm, and the remaining 15% is used to test its performance.

### 3. Inspection system

Classification of images, in our case defects, is a challenging task in computer vision, with an increasing presence in the quality management of industrial processes. Figure 5 represents the traditional sequence of operations to  
215 be performed in order to design and tune the image classifier. There are two major components in this sequence: descriptors definition and classifier training (Xiang et al., 2018).

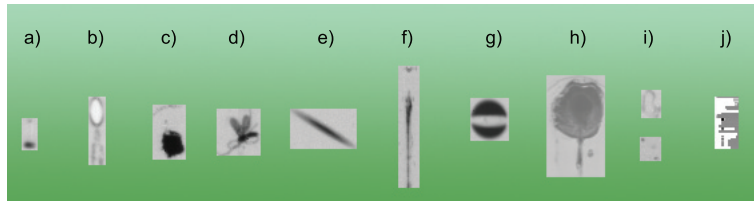


Figure 4: Defects occurring along the PET roll: a) gel, b) hole, c) black speck, d) insect, e) wrinkle, f) rubber, g) fish-eye, h) oil drop. Although detected and identified, i) slight dirt and j) acquisition/transmission error are disregarded.

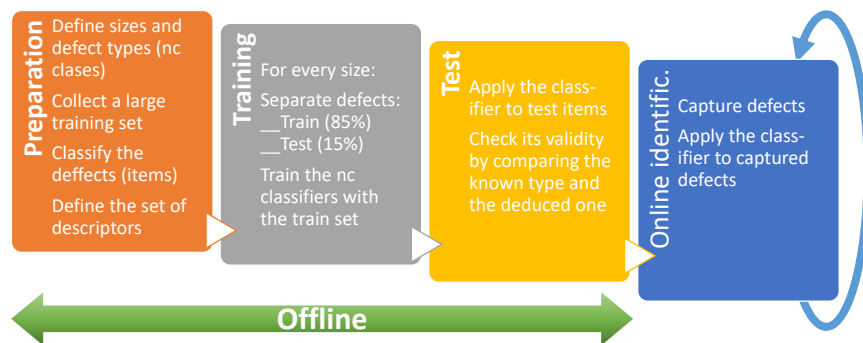


Figure 5: Sequence of operations when defining and using a classifier

### 3.1. Descriptors definition

220 A classification of images can only be successful if, as a previous task, it is possible to select a set of attributes, or define a set of features, presenting a certain correlation with the defect types, i.e. if they allow discrimination between types of defect.

225 An adequate attribute selection or feature definition is the key for pattern recognition, to the extent that a selection/definition of descriptors that are not correlated to the classes will deteriorate the classifier performance (Karabatak & Ince, 2009).

230 As an example, overlighting is a good feature for the discrimination of holes, notwithstanding that this situation also appears with insect wings or transmission errors.

This way, up to fifty four descriptors have been selected or defined. When possible, parameters have been adimensionalized to disregard the size influence. These inputs basically are:

- 235 • Enlargement of the minimum rectangle containing the defect, density of light-gray pixels (LGP), density of dark-gray pixels (DGP) and bright pixels (BP).
- Intensity average for those LGP, DGP and BP.
- Several descriptors accounting for the number of connected zones in a row with LGP, DGP or BGP.
- 240 • Standard deviation (SD) of the intensity, SD of the intensity along rows, SD of the intensity along columns, and ratio between these two last values.
- Several descriptors obtained from the intensity histogram, like dispersion, gap widths, intensity frontier of the 10% darkest pixels, and intensity frontier of the 10% brightest pixels.
- 245 • Several descriptors related to the defect edge and the second derivative, computed with Sobel filters and Laplacians as well as unidirectional gradients, including the edge momentum.

- Several descriptors related to the  $x$  and  $y$  values of the Center of gravity (CoG) and momenta for LGP, DGP and BP.
- 250 • Number of unconnected zones with LGP, DGP and BP, together with descriptors about the difference in  $x$  ( $\Delta x$ ), in  $y$  ( $\Delta y$ ) of their CoG, as well as the angle  $\tan^{-1} \Delta y / \Delta x$  (in case of two unconnected zones).
- Several descriptors related to the largest connected zone (in case of several unconnected zones).
- 255 • Correlation factors, indicating how similar is the defect to a line or an arc.
- Several heuristic descriptors aiming to discriminate the existence of insect wings, small holes and small gels.

### 3.2. Classifier training

Let's assume that we have:

- 260 • A set  $\mathbf{D}$  of available training defects; its cardinality being  $NTD = n(\mathbf{D})$ ;
- $np$  parameters previously calculated for every defect (fifty four in our case). This way, each defect can be expressed as a tuple of  $np$  parameters values  $\mathbf{d} = \{d_1, d_2, \dots, d_{np}\}$ .
- $nc$  classes or types of defect ( $nc = 10$  in our case).

265 The first step, performed by an expert operator in the case of supervised training, consists in labelling the available input defects as belonging to one of the  $nc$  possible classes or types of defects. Therefore, the total number of defects  $NTD$  is classified into  $nc$  sets of defects  $\mathbf{D}_c$ , of size  $nd_c$  (number of defects of class  $c$ ).

$$\begin{aligned}
 \mathbf{D}_c &\in \mathbf{D} \subset \mathbb{R}^{np}, \quad \text{with } \mathbf{d} \in D_c \Leftrightarrow \mathbf{d} \text{ is a defect of class } c \\
 nd_c &= n(\mathbf{D}_c) \\
 \sum_{c=1}^{nc} nd_c &= NTD
 \end{aligned} \tag{1}$$

270 Starting from this parameters definition and the list of labelled defects, it  
is now task of the classifier to obtain the rules or conditions to classify future  
instances. The traditional way of training and testing a classifier is shown in  
Fig. 6 and is performed off-line. Then, during the on-line identification task,  
the test procedure is repeated for the incoming defects. Two basic approaches  
275 haven been tested: decision tree-based classifier (DTBC) and artificial neural  
network (ANN). In the former, only one classifier is used for all of the classes.  
For the ANNs, one classifier for each class will be preferentially used in this  
work.

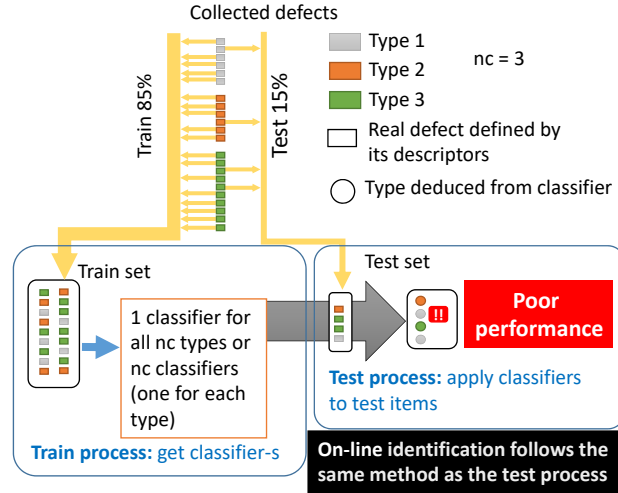


Figure 6: Casting/traditional method: train and test processes to obtain the classifier-s to be used during the on-line defect identification. This last task is essentially the same as the test process.

280 As will be seen, this traditional procedure leads to many misclassifications,  
and hence, poor performance. Following sections describe the proposed method  
to improve it. Section 4 explains one of the aforementioned methods, the DTBC,  
together with traditional improvements to it, with unsatisfactory results. Sec-  
tion 5 presents the pairwise comparison as a method to improve the performance  
of DTBC or in general any classifier. Section 6 presents a comprehensive but

285 formal explanation of ANNs as a previous step before presenting in section 7  
the experiments that validate the proposed pairwise comparison method.

#### 4. Decision tree based classifier

A decision tree is a nonparametric method that divides the input space into successive local regions. This hierarchical division into regions are finally  
290 converted into a set of simple and intuitive rules, in which a distance measure is generally used as a selection criteria to divide the input space into local regions (Alpaydin, 2010). It deals well with non-separable cases in which is not assured to find an hyperplane dividing a class from the others.

The presented method is an extension of the RIPPER algorithm, whose  
295 rules are described in (Hühn & Hüllermeier, 2009) as  $r = \langle r_A | r_C \rangle$ , where  $r_A$  is the premise and  $r_C$  is the decision understood to mean the assigned class (see Fig. 7). The premise part  $r_A$  is a conjunction of predicates (selectors) which are of the form  $(d_i \theta v_i)$  for numerical attributes, where  $\theta \in \{\leq, =, \geq\}$  and  $v_i$  is a real value acting as a limit. A rule  $r = \langle r_A | r_C \rangle$  is said to cover an instance  
300  $\mathbf{d} = (d_1, \dots, d_{np})$  if the descriptor values  $d_i$  satisfy all the predicates in  $r_A$ . In this case, instance  $\mathbf{d}$  is assigned the class  $r_C$ .

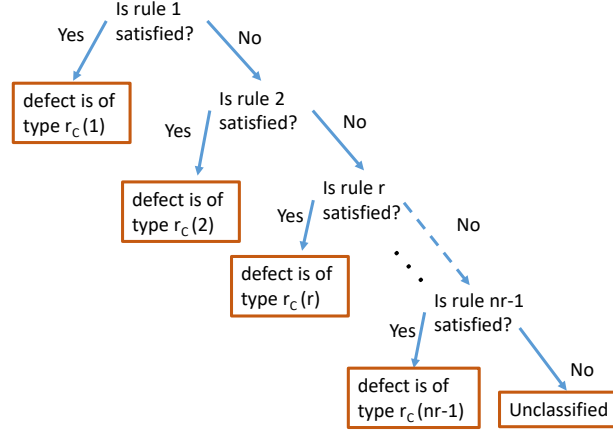


Figure 7: Decision tree based classifier.

Rather than using a distance measure, the underlying idea of the programmed DT is finding the largest set of instances from the same type that can be differentiated according to certain parameters. In a first approach, only one predicate will be considered, and the condition will be in the form  $(d_i \geq v_i)$ . Therefore, the aim is finding for which parameter the largest number of items from the same type  $c$  are higher than all of the items of the remaining classes (see Figure 8 for an example with four classes and three parameters). This way, the instance can be confidently assigned as belonging to class  $c$ . In this method the impurity is zero for the training data.

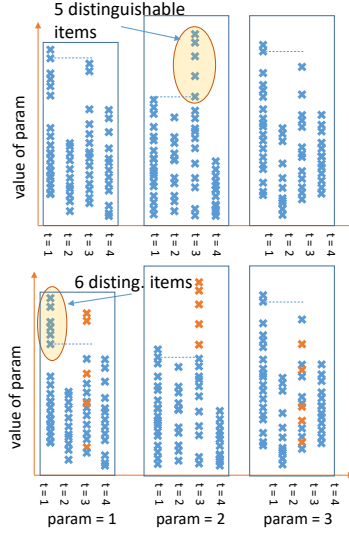


Figure 8: At the first step (top), the class  $c = 3$  is the one that can distinguish a highest number of items (5) from the value of a single parameter ( $param = 2$ ). After extracting these 5 items (in orange in figure at the bottom), the class  $c = 1$  is the one for which a highest number of items can be distinguished.

In order to carry out this algorithm, the first task is to calculate the maxima for each parameter  $p$  and each class  $c$ .

$$mx_{p,c} = \max_{\mathbf{d} \in D_c} \{d_p\} \quad p \leq np \quad c \leq nc \quad (2)$$

and also the maximum maximorum

$$MAX_p = \max_{c \leq nc} \{mx_{p,c}\} \quad p \leq np \quad (3)$$

Simultaneously, for each parameter  $p$ , we can define the *class with the maximum* for this parameter ( $cMx_p$ ) as

$$\begin{aligned} cMx_p &= \arg \max_c \{MAX_p\} \quad \text{meaning that} \\ cMx_p = c &\Leftrightarrow MAX_p = mx_{p,c} \end{aligned} \quad (4)$$

It can be proven that, if two classes share the same  $MAX_p$ , we can choose interchangeably any of them. Analogously, the *next maximum*  $NxMx_p$  can be defined as

$$NxMx_p = \max_{c \neq cMx_p} \{mx_{p,c}\} \quad (5)$$

Then, first rule  $r = 1$  consists in deducing the most significant parameter ( $\hat{p}$ ) for which a largest number of distinguishable defects can be extracted (see Fig. 8).

$$\begin{aligned} \hat{p} &= \arg \max_p \{n(\check{D}_p)\} \quad \text{with} \\ \check{D}_p &= \{\mathbf{d} \in D : d_p > NxMx_p\} \end{aligned} \quad (6)$$

Logically,  $\check{D}_p \subseteq D_{cMx_p}$ . This way, by setting the threshold  $v = NxMx_{\hat{p}}$ , and naming  $\hat{c} = cMx_{\hat{p}}$ , the first rule states that:

A defect  $\mathbf{d}$  belongs to class  $\hat{c}$  if  $d_p > v$ .

In fact, this rule can only extract some of the training set, and presumably, few of the future incoming defects. Therefore, the same reasoning must be repeated to the remaining items of the training set, leading to the following algorithm:

1. Assign  $r = 1$  (first rule) and classify the total number of classes into  $nc$  sets  $D_c^r$
2. Obtain the maxima for each parameter  $p$  and class  $c$ , and the maximum

maximorum and next maximum for every parameter  $p$

$$mx_{p,c}^r = \max_d \{\mathbf{d}_p\} \quad \forall \mathbf{d} \in D_c^r \quad (7)$$

$$MAX_p^r = \max_c \{mx_{p,c}^r\} \quad p \leq np \quad (8)$$

$$cMx_p^r = \arg \max_c \{MAX_p^r\} \quad (9)$$

$$NxMx_p^r = \max_{c \neq cMx_p^r} \{mx_{p,c}^r\} \quad (10)$$

3. Obtain the most significant parameter ( $\hat{p}$ ) for which a largest number of distinguishable defects can be extracted.

$$\hat{p}^r = \arg \max_p \{n(\check{D}_p^r)\} \text{ with} \quad (11)$$

$$\check{D}_p^r = \{\mathbf{d} \in D : d_p > NxMx_p^r\} \quad (12)$$

4. Defects satisfying (12) belong to the *distinguishable class* ( $\hat{c}^r$ )

$$\hat{c}^r = cMx_{\hat{p}^r} \quad (13)$$

5. Extract these distinguishable defects from the set of training items.

$$\begin{aligned} D &= D - \check{D}_{\hat{p}^r} \\ D_{\hat{c}^r} &= D_{\hat{c}^r} - \check{D}_{\hat{p}^r} \end{aligned} \quad (14)$$

6. If  $D \neq \{\emptyset\}$  repeat the process from 2.

- 325 7. The number of iterations is assigned to the number of rules  $nr$ .

After completing the process, the classifier will adopt the following form:

Assuming a new defect  $\mathbf{d}$

If  $d_{\hat{p}^1} > v^1$  then defect is of class  $\hat{c}^1$ .

Else, if  $d_{\hat{p}^2} > v^2$  then defect is of class  $\hat{c}^2$ .

330 ...

As a consequence, all of the items included in the training set will be classified as belonging to one of the classes after being extracted by one of the obtained rules. Formerly, when detecting a defect during the on-line inspection, its descriptors parameters must be calculated, and the algorithm should check  
335 all of the rules until one condition is satisfied, thus assigning the corresponding class. The procedure is like in a *casting* in which one class is selected among the set of defect types.

#### 4.1. Traditional modifications to the procedure

This basic procedure limits the premise  $r_A$  to only one predicate, and also  
 340 limits the condition to only be in the form  $d_i \geq v_i$ . The performance improves  
 if these constraints are relieved.

- The number of false positives is reduced if two parameters ( $p_A$  and  $p_B$ ) are checked at every rule. In this case, expressions (11,12) go from having an unique parameter to having a duple  $\mathbf{p}$  of them ( $\mathbf{p} = \{p_A, p_B\}$ )

$$\hat{\mathbf{p}}^r = \arg \max_{\mathbf{p}} \{n(\check{D}_{\mathbf{p}}^r)\} \text{ with} \quad (15)$$

$$\check{D}_{\mathbf{p}}^r = \{\mathbf{d} \in D : d_{p_A} > NxMx_{p_1}^r, d_{p_B} > NxMx_{p_2}^r\} \quad (16)$$

345 It is easy to demonstrate that  $cMx_{p_A} = cMx_{p_B}$ . By naming this value as  $\hat{c}^r$ , the rules, then, become into the form

If  $d_{p_1} > NxMx_{p_1}^r$  AND  $d_{p_2} > NxMx_{p_2}^r$  then defect is of class  $\hat{c}^r$

Nevertheless, in the situations in which a feature is discriminant enough of the defect type, an only feature is used instead of two, although intro-  
 350 ducing a penalty factor in order to promote a double comparison.

- The defect descriptors have been quantified so that they attain high values for the defects in which the descriptor can be used as a good discriminant factor (e.g. the number of clear pixels is used to discriminate holes, and this descriptor value is usually low for other types of defect). However,  
 355 it has been checked that in occasions, a low value of certain descriptors can be used to discriminate defects. Therefore, it has resulted useful to include comparisons to the minimum values of the descriptors, not only the maxima. This way, the rules can adopt one (exclusively) of the following forms:

360 a) If  $d_{p_1} > NxMx_{p_1}^r$  AND  $d_{p_2} > NxMx_{p_2}^r$  then defect is of type  $\hat{c}^r$ ,

b) If  $d_{p_1} < NxMn_{p_1}^r$  AND  $d_{p_2} < NxMn_{p_2}^r$  then defect is of type  $\hat{c}^r$ ,

c) If  $d_{p_1} > NxMx_{p_1}^r$  AND  $d_{p_2} < NxMn_{p_2}^r$  then defect is of type

365  $\hat{c}^r$ ,

d) If  $d_{p_1} < NxMn_{p_1}^r$  AND  $d_{p_2} > NxMx_{p_2}^r$  then defect is of type

$\hat{c}^r$ ,

where  $NxMn_p^r$  is obtained in the same way that  $NxMx_p^r$ , changing "max" into "min" and ">" into "<". The training procedure should select which  
 370 of the four conditions, which of the defect types, and which of the parameter duple yield a largest number of distinguishable defects from the training set.

After testing the traditional/casting procedure, even with the exposed improvements, it was checked that the results were not satisfactory, with failure  
 375 (misclassification) ratio of around 20%, reaching 50% for some experiments.

## 5. Pairwise comparison

During the validation phase, there were many situations in which a defect of type A is confused as belonging to type B, but if the classifier is retrained taking into account only these two types, the result is generally the right one.

380 Under the hypothesis that the classifier performance improves when only two defect types are considered, a different strategy is then proposed in which the procedure is more like in a *competition league*. Assuming that there are  $n$  defect classes, when a new defect is detected, every class competes against each of the remaining ones to be the one to which the new defect better fits.

385 Previously, in order to prepare these  $nm = nc(nc-1)/2$  possible confrontations, the training phase consists in obtaining  $nm$  classifiers as the one of Fig. 9. This way, for the classes  $i$  and  $j$ , the corresponding classifier  $classif_{i,j}$  is deduced taking into account only the  $nd_i$  and  $nd_j$  (see (1)) training defects belonging to these classes. Figure 10 represents the train and test procedures, which is  
 390 applied in the same way that the online defect identification.

Figure 11) shows the training phase and an example of on-line operation in which, after a new defect is detected, the  $nm$  classifiers are launched, yielding

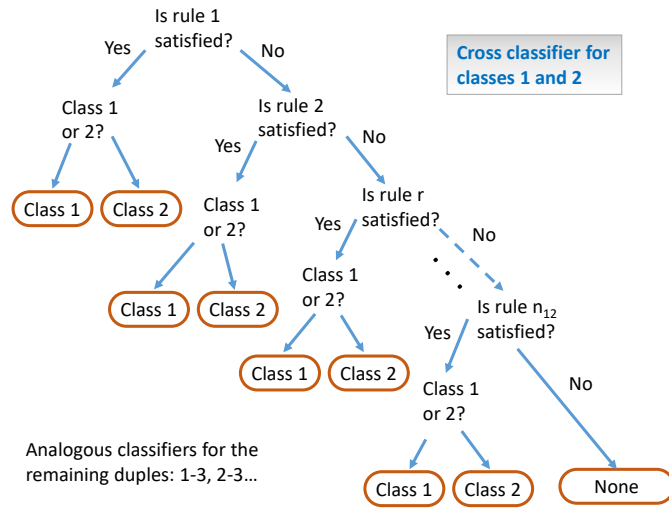


Figure 9: Decision tree based cross classifier. Only the one for the classes 1 and 2 is represented, although a total of  $nc(nc-1)/2$  must be deduced to cover all possible confrontations.

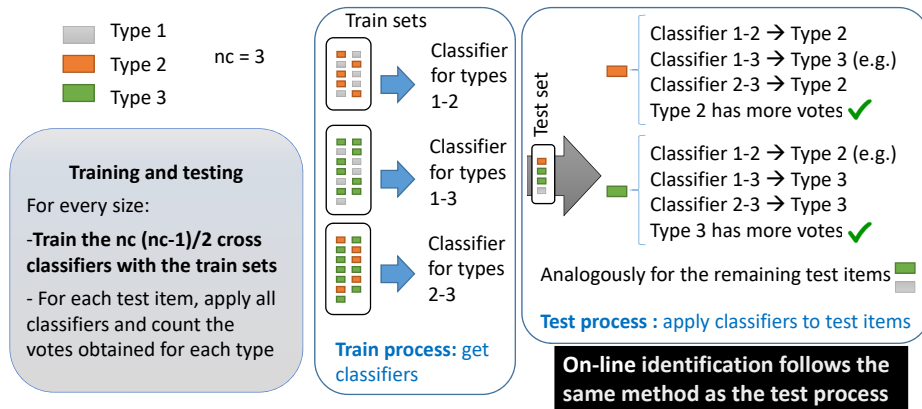


Figure 10: League winner method: train and test processes to obtain the cross classifiers to be used during the on-line defect identification. This last task is essentially the same as the test process.

the following results:  $r_{G,H} = r_{G,I} = r_{R,G} = r_{G,D} = r_{I,H} = r_{R,H} = r_{D,H} = r_{R,I} = r_{I,D} = r_{R,D} = 1$  and the remaining ones equal to zero.

#### Training phase.

A classifier must be obtained for every possible confrontation.

Classifier<sub>i,j</sub> is trained only with defects of type  $i$  and  $j$

|                                                 |                              |                           |
|-------------------------------------------------|------------------------------|---------------------------|
| Universe <sub>Gel,Hole</sub> = $D_G \cup D_H$   | nc (nc -1) /2<br>trains<br>→ | Classifier <sub>G,H</sub> |
| Universe <sub>Gel,Insect</sub> = $D_G \cup D_I$ |                              | Classifier <sub>G,I</sub> |
| ⋮                                               |                              |                           |
| Universe <sub>i,j</sub> = $D_i \cup D_j$        |                              | Classifier <sub>i,j</sub> |
| ⋮                                               |                              | ⋮                         |
| Universe <sub>Rubb,Dirt</sub> = $D_R \cup D_D$  |                              | Classifier <sub>R,D</sub> |

#### Operation (or test) phase.

A new defect appears and must be classified

| vs            | Gel | Hole                             | Insect                              | Rubber                              | Dirt                                | Victories<br>/ votes |
|---------------|-----|----------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|----------------------|
| <b>Gel</b>    |     | Classifier <sub>H,G</sub><br>Gel | Classifier <sub>I,G</sub><br>Gel    | Classifier <sub>R,G</sub><br>Rubber | Classifier <sub>D,G</sub><br>Gel    | Gel: 3               |
| <b>Hole</b>   |     |                                  | Classifier <sub>I,H</sub><br>Insect | Classifier <sub>R,H</sub><br>Rubber | Classifier <sub>D,H</sub><br>Dirt   | Hole: 0              |
| <b>Insect</b> |     |                                  |                                     | Classifier <sub>R,I</sub><br>Rubber | Classifier <sub>D,I</sub><br>Insect | Insect: 2            |
| <b>Rubber</b> |     |                                  |                                     |                                     | Classifier <sub>D,R</sub><br>Rubber | Rubber: 4            |
| <b>Dirt</b>   |     |                                  |                                     |                                     |                                     | Dirt: 1              |

Figure 11: Training and operation phase using pairwise comparison (for only five classes).

395

More formally, when a new defect is detected, a result matrix  $R$  is obtained after completing the  $n_p$  matches

$$\mathbf{R} = \begin{bmatrix} 0 & \dots & r_{1i} & \dots & r_{1j} & \dots & r_{1n} \\ \vdots & \ddots & & & & & \\ r_{i1} & & 0 & & r_{ij} & & r_{in} \\ \vdots & & & \ddots & & & \\ r_{j1} & & r_{ji} & & 0 & & r_{jn} \\ \vdots & & & & & \ddots & \\ r_{n1} & & r_{ni} & & r_{nj} & & 0 \end{bmatrix} \quad \begin{aligned} &\text{with } r_{ij} + r_{ji} = 1 \\ &\forall i, j, i \neq j \\ &r_{ii} = 0 \end{aligned} \quad (17)$$

from which a score array  $SC$  is calculated. The class with higher score is the

one to which the new set of defect parameters better fits.

$$SC_j = \sum_i r_{ij} \quad i \leq n \quad (18)$$

$$wc = \arg \max_{j \leq n} \{SC_j\}. \quad (19)$$

and thus, we can assume that the detected defect belongs to type  $wc$ .

### 5.1. Variants depending on the output mapping

In the case of DTs, the results  $r_{ij} \in \{0, 1\}$ . For other classifiers, e.g. ANNs,  
 400 the output for a new instance is given by a continuous saturation function, that maps a weighted combination of inputs into the interval  $[0, 1]$ . This continuous range for the output (we assume scalar output) allows considering three possibilities for the results  $r_{ij}$ , represented in Fig. 12.

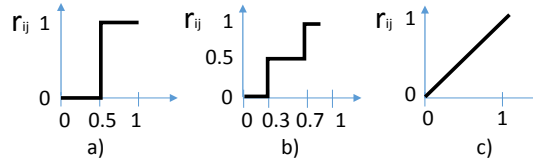


Figure 12: Different types of output: a) Binary, b) With draw zone, c) Continuous.

As will be presented in Section 7, the selection of the output shape is not  
 405 a determinant factor, although the binary output usually gives rise to slightly worse results.

## 6. Artificial neural network

Three different structures based on ANN have been programmed and tested:  
 one of them is a one-layer perceptron and the others are multi-layer perceptrons  
 410 (MLP) with two hidden layers  $H = 2$ .

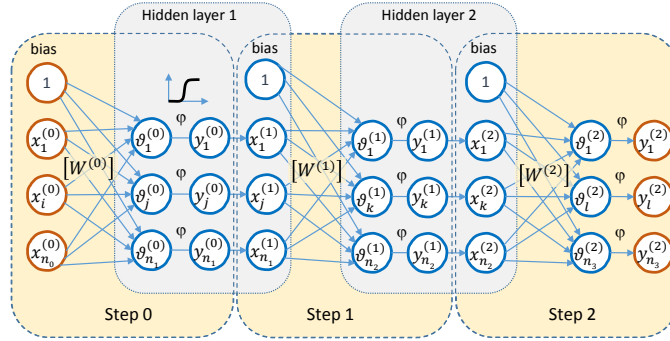


Figure 13: Propagation in an MLP with two hidden layers

### 6.1. Forward Propagation

The general propagation structure for the MLP - in our case particularized to two hidden layers - is shown in Fig. 13.

It can be assumed that each instance  $\mathbf{X}$  consists of  $np$  parameters  $x_i$ , and  
 415 is assigned a known target vector  $\mathbf{T}$  consisting of  $nt$  targets  $t_j$ . As can be seen, the remaining terms of the ANN forward propagation consists of the following components and relationships:

**Inputs at step  $s$**   $x_i^{(s)}$

$x_i^{(s)} \in \mathbf{X}^{(s)} \subset \mathbb{R}^{n_s \times 1}$ , being  $n_s$  the number of inputs at step  $s$ , and  $0 \leq$   
 420  $s \leq H$ .

The following considerations apply:

- For sake of coherence with the remaining formulation, a layer 0 has been included whose inputs are the instance parameters  $\mathbf{X}^{(0)} = \mathbf{X}$
- For the remaining steps, inputs are a copy of the outputs at previous step.

$$\mathbf{X}^{(s)} = \mathbf{Y}^{(s-1)} \quad (20)$$

- In general, inputs can be added an independent term that acts as a bias, and hence  $n_s$  must be increased by one. The formulation is simpler if simultaneously to the weight calculation to be explained

425

next, this value is calculated as another weight  $w_0^{(s)}$  (weights will be explained next). In this case  $x_0^{(s)} = 1$ .

**Weights at step  $s$**   $w_{ji}^{(s)}$   
 $w_{ji}^{(s)} \in \mathbf{W}^{(s)}, \mathbf{W}^{(s)} \subset \mathbb{R}^{(n_{s+1}) \times (n_s + 1)}$

Weights are responsible for connecting the inputs to the outputs along the different layers. For a better handling of the weight matrix, and a better comprehension of the geometrical analogy, it will be treated as a composition of row-vectors:

$$\mathbf{W}^{(s)} = \begin{bmatrix} W_1^{(s)} \\ \vdots \\ W_j^{(s)} \\ \vdots \\ W_{n_{s+1}}^{(s)} \end{bmatrix} \quad \dim(W_j^{(s)}) = n_s + 1 \quad (21)$$

**Induced local fields at step  $s$  (or at hidden layer  $s+1$ )**  $v_j^{(s)}$   
 $v_j^{(s)} \in \Upsilon^{(s)} \subset \mathbb{R}^{n_{s+1} \times 1}$ . It is a combination of inputs and, in general, an independent term that acts as a bias.

$$\Upsilon^{(s)} = \mathbf{W}^{(s)} X^{(s)} \Leftrightarrow v_j^{(s)} = \sum_{i=0}^{n_s} w_{ji}^{(s)} x_i^{(s)} \quad j \in \{1 \dots n_{s+1}\} \quad (22)$$

**Activation function**  $\varphi$

430 It is a mapping function, usually a sigmoid or a hyperbolic tangent, that saturates the induced local field.

**Outputs at step  $s$  (or hidden layer  $s+1$ )**  $y_j^{(s)}$   
 $y_j^{(s)} \in Y^{(s)} \subset \mathbb{R}^{n_{s+1} \times 1}$ . It is a mapping of  $v_j^{(s)}$  through the activation function  $\varphi$ .

$$y_j^{(s)} = \varphi(v_j^{(s)}) \quad \forall j \in \{1 \dots n_{s+1}\} \quad (23)$$

We can name as *network outputs* the values given by the ANN, i.e. the outputs at the last step. The remaining outputs can be named as *hidden outputs*.

For each pattern given by a set of inputs  $\mathbf{X}$ , the obtained network outputs must be compared to the targets  $t_j(\mathbf{X})$  or simply  $t_j$ , which are the known output values previously assigned to the studied instance  $\mathbf{X}$ . Their differences are the residuals that the training procedure should minimize.

## 6.2. Back-propagation

The aim of the training method is deducing the weights at each step, so that the outputs at each hidden layer and eventually at the final layer are obtained. The weights have to be calculated in order to minimize the errors, i.e. the differences between calculated network outputs and targets.

$$\varepsilon(\mathbf{X}) = \frac{1}{2} \sum_{j=1}^{n_H} e_j^2(\mathbf{X}) \quad \text{with} \quad e_j(\mathbf{X}) = t_j(\mathbf{X}) - y_j^{(H)} \quad (24)$$

440 where  $n_H$  is the number of outputs at the final layer, i.e. the number of network outputs.

Different methods exist to optimize the weights by propagating backwards the errors through the network, from which the ones used in this paper are Gradient Descent (GD) and Levenberg-Marquardt .

### 445 6.2.1. Gradient Descent

At step  $s$ , the weights are modified a short distance in the direction of the greatest rate of decrease of the error, i.e. in the direction of the negative gradient of the error vector evaluated with respect to the row-vectors of weights at step  $s$  (Bishop, 1995).

$$\Delta W_j^{(s)} = -\eta \nabla \varepsilon|_{W_j^{(s)}} \Rightarrow \Delta w_{ji}^{(s)} = -\eta \frac{\partial \varepsilon}{\partial w_{ji}^{(s)}} \quad j \in \{1 \dots n_{s+1}\}, \forall i \in \{0 \dots n_s\} \quad (25)$$

where  $\eta$  is called the *learning rate*. For the last step, the derivatives can be easily obtained by means of the chain rule:

$$\frac{\partial \varepsilon}{\partial w_{ji}^{(H)}} = \sum_{k=1}^{n_H} \frac{\partial \varepsilon}{\partial e_k} \frac{de_k}{dy_k^{(H)}} \frac{dy_k^{(H)}}{dv_k^{(H)}} \frac{\partial v_k^{(H)}}{\partial w_{ji}^{(H)}} \quad (26)$$

By using (24), (23) and (22), respectively, it yields that at the last step

$$\frac{\partial \varepsilon}{\partial w_{ji}^{(H)}} = \sum_{k=1}^{n_H} e_k(-1) \varphi'(v_l^{(H)}) \frac{\partial v_k^{(H)}}{\partial w_{ji}^{(H)}} = -e_j \varphi'(v_j^{(H)}) x_i^{(H)} \quad (27)$$

since

$$\frac{\partial v_k^{(s)}}{\partial w_{ji}^{(s)}} = \begin{cases} 0 & \forall k \neq j \\ x_i^{(s)} & k = j \end{cases} \quad (28)$$

In a similar way,

$$\frac{\partial \varepsilon}{\partial x_i^{(H)}} = \sum_{k=1}^{n_H} e_k(-1) \varphi'(v_k^{(H)}) \frac{\partial v_k^{(H)}}{\partial x_i^{(H)}} = - \sum_{k=1}^{n_H} e_k \varphi'(v_k^{(H)}) w_{ki}^{(H)} \quad (29)$$

Once obtained the derivatives of (27) and updated the weights at the last step  $s = H$  using (25), the weight at all of the previous steps can be obtained in an iterative way. This way, at the penultimate step and the previous ones, and from Fig. 13,

$$\frac{\partial \varepsilon}{\partial v_k^{(s)}} = \frac{\partial \varepsilon}{\partial x_k^{(s+1)}} \frac{dx_k^{(s+1)}}{dv_k^{(s)}} = \frac{\partial \varepsilon}{\partial x_k^{(s+1)}} \varphi'(v_k^{(s)}) \quad (30)$$

$$\frac{\partial \varepsilon}{\partial x_j^{(s)}} = \sum_{k=1}^{n_s} \frac{\partial \varepsilon}{\partial v_k^{(s)}} \frac{\partial v_k^{(s)}}{\partial x_j^{(s)}} = \sum_{k=1}^{n_s} \frac{\partial \varepsilon}{\partial v_k^{(s)}} w_{kj} \quad (31)$$

and by using (28)

$$\frac{\partial \varepsilon}{\partial w_{kj}^{(s)}} = \sum_{p=1}^{n_s} \frac{\partial \varepsilon}{\partial v_p^{(s)}} \frac{\partial v_p^{(s)}}{\partial w_{kj}^{(s)}} = \frac{\partial \varepsilon}{\partial v_k^{(s)}} x_j^{(s)} \quad (32)$$

450 The aim of the training method starts from the following data: inputs  $\mathbf{X}$  for each of the instances composing the training universe; targets or known outputs for each of these instances  $\mathbf{T}(\mathbf{X})$ ; and random initial values for weights at each step  $s$ ,  $\mathbf{W}^{(s)}$ , of the network. The algorithm must update the weights  $\mathbf{W}^{(s)}$  along a large enough number of iterations or *epochs* until the errors between  
455  $\mathbf{T}(\mathbf{X})$  and the outputs given by the network is below a certain value. This way, every epoch must perform the following tasks:

**Forward propagation:** for each instance given by its parameters  $\mathbf{X}$  and known outputs  $\mathbf{T}(\mathbf{X})$  (see Fig. 13):

1. Use the Weights to calculate induced local vectors at each step  $\Upsilon^{(s)}$  for  $0 \leq s \leq H$  from (22).
2. Use the activation function to obtain inputs at the remaining steps  $\mathbf{X}^{(s)}$  for  $0 < s \leq H$  from (23,20).
3. Deduce the *network output*  $\mathbf{Y}^{(H)}$ , and errors from (24).

**Back propagation:** starting from  $s = H$

1. Obtain  $\nabla \varepsilon|_{\mathbf{W}_j^{(H)}}$ , with  $j \in \{1 \dots n_{H+1}\}$  from (27).
2. Obtain  $\nabla \varepsilon|_{\mathbf{X}^{(H)}}$  from (29).
3. Decrement  $s$  and use  $\nabla \varepsilon|_{\mathbf{X}^{(s+1)}}$  in (30) to calculate  $\nabla \varepsilon|_{\mathbf{v}^{(s)}}$ .
4. Calculate  $\nabla \varepsilon|_{\mathbf{W}_j^{(s)}}$  from (32)  $\forall j \in \{1 \dots n_{s+1}\}$ .
5. Modify  $w_{ij}^{(s)}$ ,  $\forall j \in \{1 \dots n_{s+1}\}$ ,  $\forall i \in \{0 \dots n_s\}$  by using (25).
6. If  $s > 0$ , calculate  $\nabla \varepsilon|_{\mathbf{X}^{(s)}}$  from (31), and return to 3.
7. Evaluate the exit conditions (number of epochs and errors) to deduce whether exiting or starting a new epoch.

In addition to adding a bias as a new input at each step, (Plaut et al., 1986) included a *momentum* (or inertia) term to the gradient descent formula,

$$\Delta W_j^{(s)} = -\eta \nabla \varepsilon|_{W_j^{(s)}} + \alpha \Delta W_j^{(s)'} \quad (33)$$

where  $\alpha$  is an exponential decay factor between 0 and 1, and  $\Delta W_i^{(s)'}$  is the weight modification at step  $s$  for the previous pattern. This term smooths out the oscillations in  $\Delta W_j^{(s)}$  and improves the convergence. This is of particular convenience when large differences exist between the direction of successive  $\|\nabla \varepsilon|_{W_j^{(s)}}\|$ , resulting in stiff differential equations which are difficult to solve by a fixed-step numerical integration method (Bishop, 1995).

### 6.2.2. Levenberg-Marquardt

A different strategy to steer the solution towards the minimum is the Newton's method (NM) (Bishop, 1995) which uses a Taylor's series expansion to deduce the stationary points  $\mathbf{X}^m$  at which  $\nabla \varepsilon(\mathbf{X}^m) = 0$ . If  $\mathbf{X}_a$  is the actual

state, then this method aims to point directly towards the minimum of the error function.

$$0 = \nabla \varepsilon(\mathbf{X}_m) = \nabla \varepsilon(\mathbf{X}_a) + \mathbf{H}(\mathbf{X}_a)(\mathbf{X}_m - \mathbf{X}_a) \Rightarrow \Delta \mathbf{X} = -\mathbf{H}^{-1}(\mathbf{X}_a) \nabla \varepsilon(\mathbf{X}_a) \quad (34)$$

A local quadratic approximation to NM is the Gauss-Newton iteration, in which the elements of the Hessian matrix  $\mathbf{H}$  can be approximated as the outer (Kronecker) product of the Jacobian  $\mathbf{J}$  and its transposed (Nocedal & Wright, 1999)

$$\frac{\partial^2 \varepsilon}{\partial x_i \partial x_j} \approx \frac{\partial \varepsilon}{\partial x_i} \frac{\partial \varepsilon}{\partial x_j} \rightarrow \mathbf{H} = \mathbf{J}^T \mathbf{J} \quad \text{with } \mathbf{J} = \nabla \varepsilon \quad (35)$$

480 This approximation is not valid when residuals are long, i.e. far from the minimum. This way, the convergence is based on heuristic rules, rather than having a mathematical support. To avoid this uncertainty, mainly at the first iterations, Levenverg added a diagonal matrix  $\lambda \mathbf{I}$  to  $\mathbf{H}$ , modification which was improved by Marquardt resulting in the Levenberg-Marquadt (LM) update rule.

485 For the case of row vectors of the weight matrix, it yields

$$\Delta \mathbf{W}_j^{(s)} = -(\mathbf{J}^T \mathbf{J} + \lambda \text{diag}[\mathbf{J}^T \mathbf{J}])^{-1} \mathbf{J}, \quad \text{with } \mathbf{J} = \nabla \varepsilon|_{\mathbf{W}_j^{(s)}} \quad (36)$$

This is an alternative to GD for updating the weights via back propagation; hence, the LM algorithm should substitute this expression for (25) in Step 5 of the back propagation. Although it is an heuristic method it has become as one of the most widely used optimization algorithms. In essence, it is a blend of GD and Gauss-Newton iteration.

490 In practice, a common approach is to start with, e.g.  $\lambda = 1$  and vary it at each iteration/pattern, according to the obtained error. If the error increases after updating the weights, the old ones are restored and increase  $\lambda$  by a factor of 10, in order to blend more towards simple GD, so that the solution approaches to the minimum. If, on the contrary, the error decreases, then accept the step (i.e. keep the weights at their new values) and decrease  $\lambda$  by a factor of 10. In this case, we are moving close to the minimum in which the quadratic approximation is valid, and the Gauss-Newton iteration is dependable for updating the weights according to (34).

## 500 7. Experimental results and comparison of methods

A battery of twenty experiments has been launched to compare the performance of different methods for classifying defects starting from a supervised train. At each experiment, the distribution of instances to be used for training or testing has been changed, although maintaining the ratio 85/15. The first tested method is a DT-algorithm explained at subsection 4. In addition, 505 five different structures based on ANN have been tested: first one is a one-layer perceptron; second one is a multi-layer perceptrons (MLP) obtained from (Acquatella, 2012); following two ones are a MLP obtained from (Chen, 2018), and a biased modification of it which includes an independent term summed to inputs; for the last one, the Matlab command 'trainlm' (lm for Levenberg-Marquadt) from Neural Network Toolbox was used, with 'mse' as performance 510 function and 10 hidden layers. First four ones use GD and the last one uses LM strategy. The battery of experiments have been repeated for the four sizes, and also for the complete universe of defects.

Table 2: Properties of the used ANNs.

|            | 1 Layer    | GD1        | GD2u       | GD2b       | Lev-Mq   |
|------------|------------|------------|------------|------------|----------|
| Layers     | 1          | 1+2 hidden | 1+2 hidden | 1+2 hidden | 1+1 hid. |
| Neurons    | 54×1       | 54×10×4×1  | 54×10×4×1  | 54×10×4×1  | 54×10×1  |
| Backpro    | Grad. Desc | Grad. Desc | Grad. Desc | Grad. Desc | Lev-Mq   |
| Bias       | Yes        | Yes        | No         | Yes        | Yes      |
| Momentum   | No         | Yes        | No         | No         | -        |
| Saturation | sigmoid    | tansig     | sigmoid    | sigmoid    | tansig*  |
| Weight upd | pattern    | pattern    | epoch      | epoch      | -        |
| Validation | Yes        | No         | No         | No         | Yes      |

\* in Layer 1

515 Table 2 summarizes the characteristics of the ANN-based methods. The row *Bias* refers to the addition of a independent term for weights, which was introduced in subsection 6.1. The row *Moment* refers to the term expressed in (33). The row *Weight upd* refers to whether the weights are updated after each train

pattern evaluation or after each epoch iteration. Finally, the row *Validation*,  
520 when affirmative, indicates that a percentage of train instances are extracted  
and used to validate the robustness of the trained network for instances differ-  
ent to the strictly training ones; this way, a robuster network is more likely to  
perform better for new patterns.

The average of fails (in percentage) for the twenty experiments is shown in  
525 Fig. 14 for every method. The bars represents the minimum and maximum  
percentage of fails occurring for a certain experiment. Based on these methods,  
different variants have been tested for the ANNs. First three ones correspond to  
the league strategy and are represented as  $\circ$ ,  $\star$  and  $\times$ , specifically to the three  
types of outputs of Fig. 12. As can be seen, the output choice (binary, with  
530 draw-zone and continuous) is not a determinant factor, but in any of the three  
cases, the results are quite better than with the traditional *casting* selection,  
represented by  $*$ , in which  $n$  networks are applied to each instance. The figure  
also represents as  $\square$  the satisfactory results of a single N-output Matlab Toolbox  
using L-M backpropagation. The classifier performance, measured as the ratio  
535 of true positives, is similar between the DT-algorithm and the MLPs.

With regard to the convergence speed, it is worth mentioning that DT-  
based algorithm has been programmed in C++, the gradient-descent based  
ANN (GDANN) algorithms have been programmed as Matlab scripts, using  
different epoch limits (500, 2000 and 5000); and finally the LM based ANN  
540 algorithm has been tested using the command 'trainlm from the Matlab Neural  
Network toolbox with mean squared normalized error as performance function.  
Influence of the number of epochs in the classifier performance for GDANN al-  
gorithms (programmed as Matlab scripts) is represented in Fig. 15. As can be  
seen, the training time depends logically on the number of training patterns (see  
545 Tab 1). This training time, determined by the number of maximum epochs im-  
posed to the ANN algorithms, generally leads to a better classifier performance.  
Algorithm GD2b and GD2u (see Tab. 2) are closer to the lower-left corner which  
is optimum. Finally, the medium point in a 3-tuple series ( $max\ epoch = 2000$ )  
is usually the value with a better trade-off between training time and accuracy.

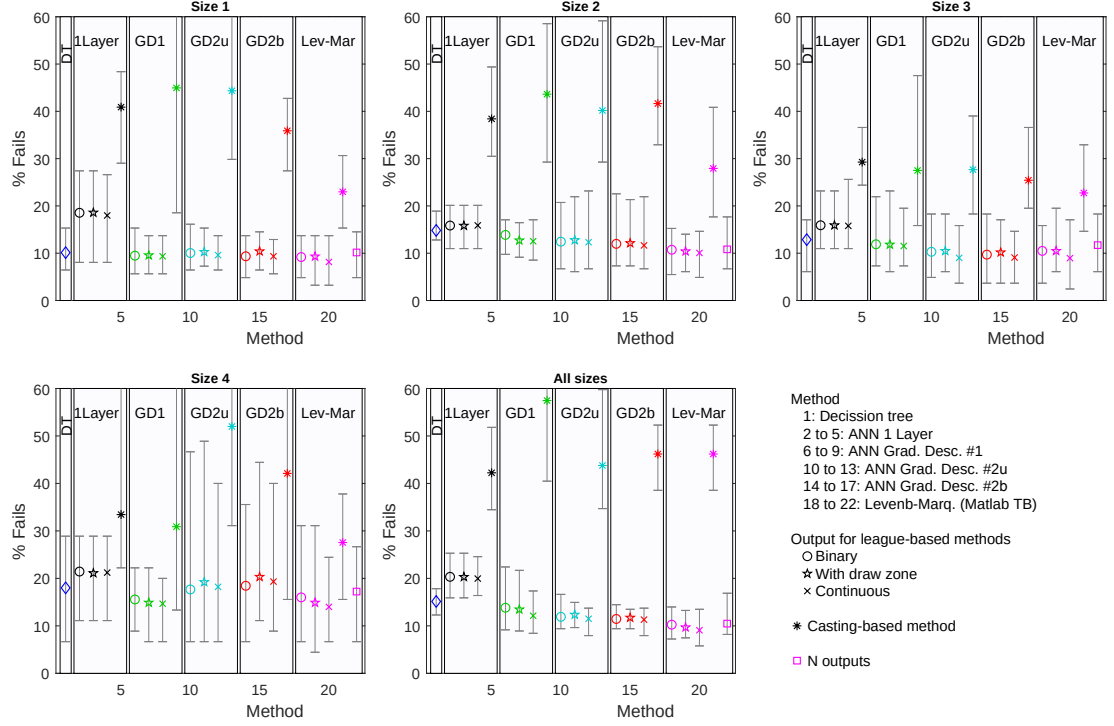


Figure 14: Percentage of failed identification for each group and method.

550 This limit in the number of epoch (2000) is the one chosen for the remaining figures and tables.

Table 3 represents the training times for the GDANN algorithms for an epoch limit of 2000, and highlights the poor convergence speed of first two algorithms. It is worth mentioning that in the traditional *casting* method, an amount of  $n$  training processes had to be launched. With the proposed League-Winner modification, the number is  $(n - 1)/2$  bigger but, on the other hand, the amount of input patterns is reduced, in average, in  $n/2$ ; hence the training time is approximately the same or lower for most of the methods.

### 7.1. Improvement for supervised training

560 During the offline labelling of training defects according to predefined types, most of the assignments are clear and objective. There are, however, certain

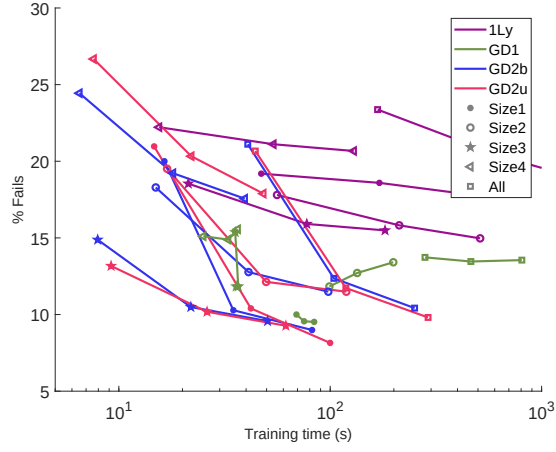


Figure 15: Influence of the number of maximum epochs on the classifier performance (only for league-based ones). Three different epoch limits (500, 2000 and 5000) have been used for every size and algorithm.

Table 3: Training time in seconds spent by the tested methods based in ANNs.

| Time (s)  | League |     |      |      |    | Casting |      |      |      |    | N Outp |
|-----------|--------|-----|------|------|----|---------|------|------|------|----|--------|
|           | 1Lay   | GD1 | GD2b | GD2u | LM | 1Lay    | GD1  | GD2b | GD2u | LM | LM     |
| Size 1    | 171    | 75  | 35   | 42   | 42 | 201     | 226  | 35   | 39   | 12 | 66     |
| Size 2    | 212    | 134 | 41   | 50   | 42 | 262     | 567  | 39   | 44   | 12 | 77     |
| Size 3    | 77     | 37  | 22   | 26   | 24 | 110     | 98   | 17   | 19   | 8  | 33     |
| Size 4    | 54     | 33  | 18   | 22   | 29 | 69      | 48   | 12   | 14   | 6  | 20     |
| All sizes | 647    | 463 | 104  | 119  | 53 | 751     | 2163 | 105  | 140  | 13 | 203    |

defects for which its labelling is not clear (even when asking to experienced workers or engineers), leading to indecision/subjectivity about whether assigning to class A or class B (see Figure 16). During the training, a defect can be included in the class A, but for a new defect with similar attributes, it will not be a clear miss-classification if it is finally assigned of type B. In addition, the subjectivity can lead to inconsistencies in the training sets, which can deteriorate the ANN performance, mainly when there is a high number of neurons (Stańczyk & C., 2015) This situation appeared for some defects of the types *gel*,

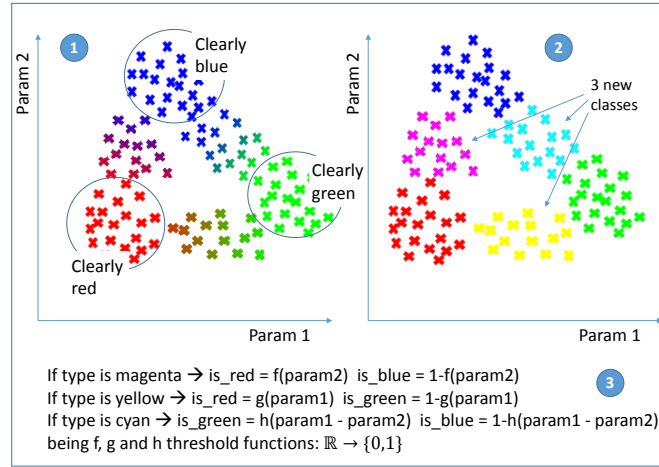


Figure 16: ① Some of the train defects are not clearly labelled; ② three new classes are introduced to include them; ③ heuristic rules are used for new on-line defects classified as one of them.

570 *black speck* and *rubber*.

For defects which are clearly one of these three types, the number of miss-classifications was reduced if three new classes are introduced in the classifier: gel-speck, gel-rubber and speck-rubber. Later, when the algorithm classifies a new defect into one of these additional types, the defect is finally reassigned to one of the original three types (*gel*, *black speck* and *rubber*) as a function of simple heuristic conditions (see an example in Fig. 16).

Table 4 represents in green the situations for which the classifier performance

Table 4: Reduction in fails ratio when adding three new auxiliary classes.

|           | League |       |       |       |       |       | N Outputs |
|-----------|--------|-------|-------|-------|-------|-------|-----------|
|           | DT     | 1Ly   | GD1   | GD2b  | GD2u  | LM    | LM_NO     |
| Size 1    | -0.65  | 1.84  | 1.53  | -0.15 | -0.54 | -0.78 | -0.78     |
| Size 2    | 0.79   | -1.97 | -1.84 | 0.56  | 0.05  | -1.55 | -2.17     |
| Size 3    | -0.46  | -0.5  | 0.04  | -0.79 | -1.36 | -0.79 | 0.43      |
| Size 4    | 2.44   | 3.78  | 1.11  | 3.22  | 3.44  | 0.22  | -1.45     |
| All sizes | 1.81   | -0.86 | 0.87  | -0.18 | -0.35 | -0.16 | 0.65      |

Table 5: Modifications in the project specifications.

|                 | Initial        | Final         |
|-----------------|----------------|---------------|
| Camera length   | < 10m          | > 20m         |
| Web width       | 1500mm         | 920mm         |
| Resolution      | 0.15mm         |               |
| Pixels required | $\geq 10000px$ | $\geq 6100px$ |

improves by introducing this three auxiliary classes. Rather than the performance improvement (except for the biggest defects), the tedious offline task of  
580 labelling each of the training items is significantly faster and relieved.

## 8. Conclusions and learned lessons

### 8.1. Communication protocol

Different changes during the project development arose which have deviated the original setup from the optimum final one. The main one was the separation  
585 between the process unit and the camera, which finally extended to 25 m. On the contrary, requirements in the web width were reduced. Table 5 collects some of the specifications modifications.

When starting the project, only few line cameras satisfied the required number of pixels, most of them implementing a *Camera Link* protocol, and this was  
590 the chosen option. Formerly, the distance camera-computer was extended and the limitation of 10 m for the Camera Link cable was saved by including two

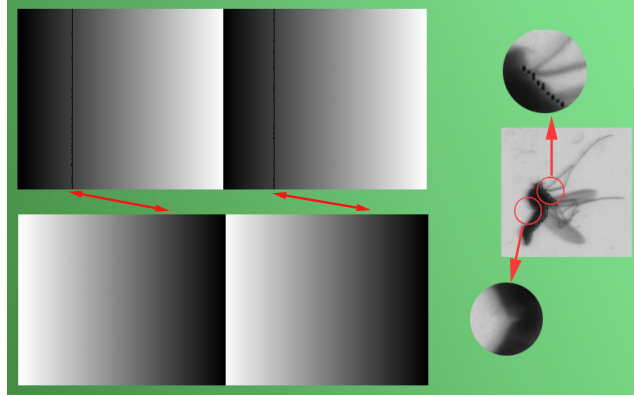


Figure 17: Error in data transmission over the Camera Link setup when bright intensity increases from 64 (0x40) to 66 (0x42). Error does not appear when intensity decreases.

repeaters (see Fig. 2). As previously advised by the repeater supplier, some issues appeared, the most important one being the appearance of spurious dots when the intensity increased from the value 64 to 66. This issue is reflected  
 595 in Fig. 17. The gradings at the left are patterns provided by the camera, with ascending intensity (top) or descending one (bottom). The right-hand image shows the effect on the right contour of a defect capture.

This issue forced including additional coded functions to detect and correct this spurious information, with satisfactory results. Including the repeaters also  
 600 impeded using Camera Link Full Speed, and only Base Speed could be used, with the consequent reduction of transmission rate. Moreover, the hardware cost considerably increased due to the repeaters and three new cables, since the connectors of the original ones were not compatible.

Clearly, in light of the final specifications, a line camera with CoaXpress or  
 605 Gig-E with 8192 pixels would have been a more economic and reliable option.

## 8.2. League winner method

With the exception of a Matlab toolbox performing an ANN classifier with N Outputs based on LM, the number of fails for all of the tested methods were excessive. The performance is drastically improved if the training and classifying

610 phases are organized as a competition league. For the programmed classifiers,  
the training times does not increase.

### 8.3. Machine learning method

Neural networks or deep learning is the preferred method for pattern recog-  
nition and image classifications. Different options exist regarding the activation  
615 function, the number of layers, or the back propagation method, and the de-  
veloper can select one of them according to the capability of programming the  
algorithm or taking advantage of existing ones running at an offline platform.  
Algorithm based on decision trees can also lead to similar classification perfor-  
mances.

620 A single layer perceptron can give rise to good results if the attributes ex-  
tracted from the image are significant enough of the class type, i.e. they allow  
discrimination between types of defect. If it is not the case, their simplicity does  
not compensate a performance quite poorer than the other tested methods.

Levenberg-Marquadt modifications leads to the best convergence speeds, at  
625 the expense of a higher complexity.

### 8.4. Semisupervised training

A tedious task during the supervised training phase is to organize the bulk  
of images according to a predefined classification. Additionally, in an important  
percentage of cases, the images cannot be clearly identified as belonging to one of  
630 the classes and the expert subjectivity can lead to associate two similar defects  
to different classes.

Including a defect in a wrong class during the labelling phase leads to *over-*  
*fitting* which deteriorate the classifier performance, specially for the proposed  
DT algorithm based on the number of distinguishable instances. Although these  
635 lapses, known as teacher noise, are drastically reduced by including additional  
classes as explained in subsection 7.1, a tedious labelling review is necessary to  
minimize them.

A semisupervised training (Mehrizi et al., 2018) may alleviate this task previous to the training, and provide the classifier with higher robustness, since  
 640 it is supposed that the component of subjectivity will be disregarded. It can even teach the expert showing new types of defect if a group is isolated having significant differences with respect to the predefined ones.

To sum up, this work highlights three main factors to improve the classifier performance:

- 645 • A good and numerous selection/description of descriptors, able to discriminate common characteristics of the instances. If not, classes will not be linearly separable, and one or more hidden layers are demanded when using neural networks.
- An objective and coherent classification of the training data made by an  
 650 expert.
- If, even so, the performance is not satisfactory, the explained league-winner method based in pair comparison drastically reduces the misclassification of test or future instances.

## Acknowledgements

655 This research has been possible thanks to the economical and logistical support of Benzaplastic Ltd.

## References

- Acquatella, P. (2012). Backpropagation Tutorial- A training algorithm for neural networks. URL: <https://es.mathworks.com/matlabcentral/fileexchange/35659> Accessed 2 September 2019.  
 660
- Affonso, C., Rossi, A. L. D., Vieira, F. H. A., & de Leon Ferreira de Carvalho, A. C. P. (2017). Deep learning for biological image classification. *Expert Syst Appl*, 85, 114-122. doi:<https://doi.org/10.1016/j.eswa.2017.05.039>.

- 665 Alpaydin, E. (2010). *Introduction to Machine Learning*. (2nd ed.). The MIT Press.
- Amlani, A. M., & Schafer, E. C. (2009). Application of Paired-Comparison Methods to Hearing Aids. *Trends Amplif*, 13, 241–259. doi:10.1177/1084713809352908.
- 670 Asadi, S., & Shahrabi, J. (2016). RipMC: RIPPER for Multiclass Classification. *Neurocomputing*, 191, 19–33. doi:https://doi.org/10.1016/j.neucom.2016.01.010.
- Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc.
- 675 Carolino, E., & Barão, I. (2013). Robust methods in acceptance sampling. *Revstat-Stat J*, 11, 67–82.
- Chen, M. (2018). Deep Multilayer Perceptron Neural Network with Back-propagation. URL: <https://es.mathworks.com/matlabcentral/fileexchange/55946> Accessed 2 September 2019.
- 680 Cho, C.-S., Chung, B.-M., & Park, M.-J. (2005). Development of real-time vision-based fabric inspection system. *Ieee T Ind Electron*, 52, 1073–1079. doi:10.1109/TIE.2005.851648.
- Cohen, W. W. (1995). Fast Effective Rule Induction. In *In Proceedings of the Twelfth Int Conf Mach Learning* (pp. 115–123). Morgan Kaufmann.
- 685 Dalitz, C., Wilberg, J., & Bednarek, K. E. (2018). Paired Comparison Sentiment Scores. doi:https://arxiv.org/abs/1807.03591.
- Ferreira, A., & Giraldi, G. (2017). Convolutional Neural Network approaches to granite tiles classification. *Expert Syst Appl*, 84, 1–11. doi:https://doi.org/10.1016/j.eswa.2017.04.053.

- 690 Hühn, J., & Hüllermeier, E. (2009). FURIA: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, 19, 293–319. doi:10.1007/s10618-009-0131-8.
- Juran, J. M., & Godfrey, A. B. (1998). *Juran's Quality Handbook*. (5th ed.). McGraw-Hill.
- 695 Karabatak, M., & Ince, M. C. (2009). An expert system for detection of breast cancer based on association rules and neural network. *Expert Syst Appl*, 36, 3465–3469. doi:https://doi.org/10.1016/j.eswa.2008.02.064.
- Karimi, M. H., & Asemani, D. (2014). Surface defect detection in tiling Industries using digital image processing methods: Analysis and evaluation. *ISA*  
700 *T*, 53, 834–844. doi:https://doi.org/10.1016/j.isatra.2013.11.015.
- Kuo, C.-F. J., Lai, C.-Y., Kao, C.-H., & Chiu, C.-H. (2018). Integrating image processing and classification technology into automated polarizing film defect inspection. *Opt Laser Eng*, 104, 204–219. doi:https://doi.org/10.1016/j.optlaseng.2017.09.017.  
705
- Mehrizi, A., Yazdi, H. S., & Taherinia, A. H. (2018). Robust Semi-Supervised Growing Self-Organizing Map. *Expert Syst Appl*, 105, 23–33. doi:https://doi.org/10.1016/j.eswa.2018.03.046.
- Mukherjee, I., & Routroy, S. (2012). Comparing the performance of neural networks developed by using Levenberg–Marquardt and Quasi-Newton with the  
710 gradient descent algorithm for modelling a multiple response grinding process. *Expert Syst Appl*, 39, 2397–2407. doi:https://doi.org/10.1016/j.eswa.2011.08.087.
- Nocedal, J., & Wright, S. J. (1999). *Numerical Minimization*. Springer-Verlag.
- 715 Ooi, M. P.-L., Sok, H. K., Kuang, Y. C., Demidenko, S., & Chan, C. (2013). Defect cluster recognition system for fabricated semiconductor wafers.

*Eng Appl Artif Intel*, 26, 1029–1043. doi:<https://doi.org/10.1016/j.engappai.2012.03.016>.

720 Paladini, E. P. (2000). An expert system approach to quality control. *Expert Syst Appl*, 18, 133–151. doi:[https://doi.org/10.1016/S0957-4174\(99\)00059-7](https://doi.org/10.1016/S0957-4174(99)00059-7).

Plaut, D., Nowlan, S., & Hinton, G. (1986). *Experiments on learning back propagation*. Technical Report Carnegie–Mellon University Pittsburgh, PA.

725 Ravikumar, S., Ramachandran, K. I., & Sugumaran, V. (2011). Machine learning approach for automated visual inspection of machine components. *Expert Syst Appl*, 38, 3260–3266. doi:<https://doi.org/10.1016/j.eswa.2010.09.012>.

Rokach, L., & Maimon, O. (2005). Top-down induction of decision trees classifiers - a survey. *IEEE T Syst Man Cy C*, 35, 476–487.

730 Stańczyk, U., & C., J. L. (Eds.) (2015). *Feature Selection for Data and Pattern Recognition*. Springer–Verlag.

Thurstone, L. L. (1927). The Method of Paired Comparisons for Social Values. *J Abnorm Soc Psych*, 21, 384–400.

735 Valavanis, I., & Kosmopoulos, D. (2010). Multiclass defect detection and classification in weld radiographic images using geometric and texture features. *Expert Syst Appl*, 37, 7606–7614. doi:<https://doi.org/10.1016/j.eswa.2010.04.082>.

Xiang, Y., Zhang, G., Gu, S., & Cai, J. (2018). Online multi-layer dictionary pair learning for visual classification. *Expert Syst Appl*, 105, 174–182. doi:<https://doi.org/10.1016/j.eswa.2018.03.048>.  
740

Zhang, J. B. (1996). Computer-aided visual inspection for integrated quality control. *Comput Ind*, 30, 185–192. doi:[https://doi.org/10.1016/0166-3615\(96\)00012-7](https://doi.org/10.1016/0166-3615(96)00012-7).