An effective approach to steel defect detection

Antonio Terpin
Electronic Engineering, Scuola Superiore
Università degli Studi di Udine
Udine, Italia
Email: terpin.antonio@spes.uniud.it

Claudio Verardo
Electronic Engineering, Scuola Superiore
Università degli Studi di Udine
Udine, Italia
Email: verardo.claudio@spes.uniud.it

***Abstract*—Quality control is a main issue in any industry. The need of assuring a human-like evaluation during products quality control has resulted in an active research aiming to develop an automatic defect detection scheme. In this paper, an effective solution to defect detection on steel surfaces from images is presented. First, a preprocessing step aiming to spot plausible defective areas is discussed. Then, the classification of these proposed regions is made. Hence, a proper segmentation scheme is described. Incidentally, a novel usage of the dilation factor in convolutional layers is introduced, providing an effective approach to classifying images of different sizes.**

***Index Terms*—Defect Detection, Computer Vision, Deep Learning, Wavelet.**

I. Introduction

A fundamental aim in any industry is to design highly efficient and effective quality control processes. Therefore, the research area focusing on automatic defect detection systems has become even more fecund in the last decades.

There is a large variety of previous work on defect detection on steel surfaces [1–5]. Many ideas [1–3] bank on the uniformity of the background surface, while other solutions use deep learning architectures [4, 5], which provide an attempt to a robust defects segmentation. More refined approaches rely on wavelets to detect abrupt changes on the surfaces [6–9].

The core of the architecture proposed in this paper is based on wavelet analysis and deep learning, due to two main reasons. Firstly, multi-resolution analysis (MRA) based on wavelets was proven effective in facing localization in both spatial and frequency domains. [10–12]. This because of wavelet transform mathematical properties, compared to Fourier's transform.

Secondly, in the last years deep learning [13, 14] has outperformed any human-designed classifier. Indeed, computer vision and image processing are increasing in popularity in many fields, from autonomous driving vehicles to retail security. Hence, since the rise of deep learning applications [15] there has been an appreciable improvement in the effectiveness of defect detection based on visual systems.

Three main computer vision tasks can be outlined: classification, object localization and object detection.

The classification task faces the supervised learning problem of identifying to which of a set of categories a given object belongs to. In computer vision this means assigning one of the available labels to an image. This is the simplest of the three tasks, and recognizing the category of the principal object in a picture is the standard application of Convolutional Neural Network (CNN). Examples of usage are identifying handwritten characters [16, 17], house numbers [18] and traffic signs [17].

The main reason why CNNs have become so popular since LeCun originally introduced them [15, 16, 19, 20] is that they represent a black box from raw pixels to categories labels, therefore they overcome the difficulties intrinsic to designing tailored features extractors. Morover, they are also more likely to be shift and scale invariant [20], and they was proven to have enviable classification accuracies.

In defect detection field, a classification task is accounted when objects, e.g. steel surfaces, need to be binarily classified as defective or flawless. In monitoring applications, classifying pictures as a whole would be expensive, since local screening hardware would be needed. Patently, a global visual system is far more appetible.

Moreover, a local analysis may miss some global features of a particular defect; this is the case of burst defects, such as zipper cracks [21].

Object localization sights to find a given number of items in a given context, predicting both their position and their class. Object detection removes the constraint on the number of items, allowing either zero or any finite number of objects, which is not fixed *a priori*. In computer vision, in particular in 2D images, the position is described with a bounding box. CNNs were used along with sliding window and multiscale approaches for object detection [22–24].

In this paper, a further refined system is presented, since the purpose of the defect detection algorithm is not only to globally mark a steel surface as flawless or defective from its picture, but to highlight flawed regions within the image and to label them as belonging to a particular defect class.

Pixel-wide classification is known in literature as image segmentation, and there are three main families of tecniques: hysteretic thresholding, edge-based and region-based [25]. Thresholding exploits a previously known function defined over the pixels space and classifies pixels through comparison with some discrete values (thresholds) [26]. However, it is tipically used within other tecniques rather then alone. Region-based approaches use either graph algorithms [27, 28] or watersheds analogies [29]. Edge-based tecniques use an edge detection filter [30–32] instead, along with denoising and thresholding considerations, to solve the boundary detection

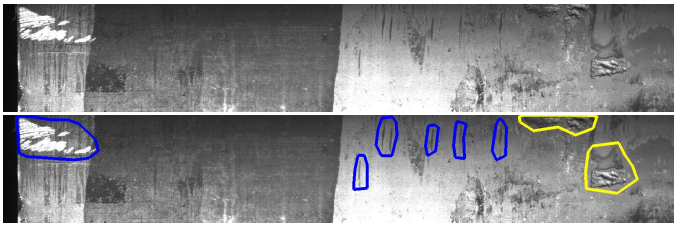

Figure 1. Detection process input and output.

problem. Remark that, although similar, boundary detection aims to describe changes in pixel ownerwhip from one object or surface to another, whereas an edge is an abrupt change, which can be a sub-domain of a border. There are also more advanced boundary-related tecniques [33] which rely on energy minimization and are embedded on region-based approaches. Indeed, all these tecniques can be mixed both together and with learning algorithms, either unsupervised [25] or supervised [34].

The proposed approach merges the more effective and efficient ideas of previously described work, balancing the drawbacks of different tecniques. Since segmentation is needed, an edge-based contour detector is presented, to reach high speed segmentation. Wavelets are used along with image preprocessing and alpha-shape [35] to identify proposals, i.e. regions of interest for the classifier, which may contain a defective area. To overcome the bias introduced from handcrafting the edge-detection filter, the hyperparameters of the algorithm are tuned with Bayesian Optimization (Bopt) [36–38]. A multi-column CNN (MC-CNN) [17] is then used to combine the segmentation information with a well-known classifier architecture, exploiting both local information and global information. The preliminary implementation of the proposed architecture has shown good performances on the *Severstal: Steel Defect Detection* Kaggle competition dataset [39].

II. Steel surfaces defect detection

A. Problem statement

Given a set of steel surfaces images with the description of their defective areas, learn to detect defective pixels in new pictures.

The surfaces may have more disjunct defective areas, and there are four defective classes, described in *Section II-B*.

For each of this areas, a thorough characterization of the member pixels must be provided, along with the class of the defect pictured in the considered region.

Defective pixels are described using a Run Length Encoding (RLE) approach. The rationale is that an efficient way to store pixel-wide information is needed, and it is reasonable to believe that many defective pixels will be adjacent.

To do so, the binary matrix describing interesting pixels is firstly vectorized column-wide, i.e. each column vector is appended to the previous.

Secondly, pixels are enumerated in this vectorized map.

Finally, the RLE algorithm is used on the indices of the cosidered pixels.

```
Example:

    Suppose the ones in the below
    matrix need to be encoded:

     _         _
    | 1  0  1  1 |
    | 1  1  1  0 |
    | 0  1  1  0 |
     -         -

    The interested cells, expressed
    as (x, y) coordinates, are:

    [(1, 1) (1, 2) (2, 2) (2, 3)
        (3, 1) (3, 2) (3, 3) (4, 1)]

    The vectorized matrix is:

    [1 1 0 0 1 1 1 1 1 1 0 0]

    Which can be encoded as:

    "1 2 5 6"
```

An optimized implementation in `MATLAB` of both RLE encoding and decoding schemes described is proposed in [40].

A visual description of the end-to-end process is given in *Figure* 1, where defective areas are highlighted with different colors, depending on the defect class.

Mathematically, the task can be described as:

Given a *training set* $\left(\underline{\underline{\mathbf{X}}}_{train}, \underline{\mathbf{y}}_{train}\right)$ and a *test set* $\left(\underline{\underline{\mathbf{X}}}_{test}, \underline{\mathbf{y}}_{test}\right)$, the goal is to build a *trainer* system $\mathcal{T}$ and a *predictor* function $\mathcal{P}$ such that:

$$\underline{\underline{\mathbf{\Theta}}} = \mathcal{T}\left(\underline{\underline{\mathbf{X}}}_{train}, \underline{\mathbf{y}}_{train}\right); \quad |\mathcal{P}\left(\underline{\underline{\mathbf{X}}}_{test}; \underline{\underline{\mathbf{\Theta}}}\right) - \underline{\mathbf{y}}_{test}| \to 0.$$

Both the trainer and the predictor are implemented through deep learning tecniques and are described in *Section III*.

B. Defect analysis

The production process of flat steel sheet is especially delicate. In literature [21, 41] numerous defects are classified. Hence, many of the traditional defect types may be grouped together into the four classes given. In this section, a plausible explanation of each defect class is provided.

However, the fundamental observation, is that some defects have a global origin, i.e. they come from a flawed machinery, and thus it is reasonable that a local classifier would miss some important details.

1) Defect class No.1: the first type of defects has not been classified into one of the classes found in literature, yet.

However, it is a glaring example of burst defects. In fact, it is reasonable to expect such defect to be repeated multiple

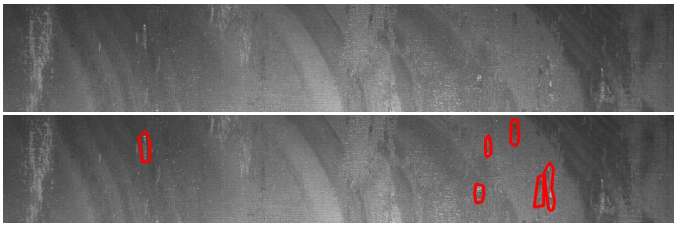

Figure 2. Steel surface with defect class No.1.

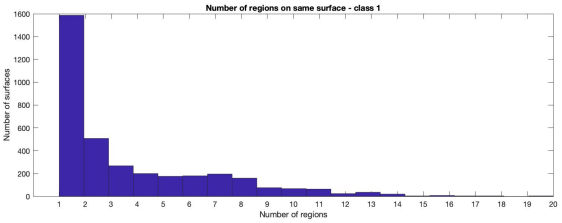

Figure 3. Number of defects of class No.1 per defective surface.

times on the same surface, as deducible from the histogram in *Figure 3*.

An example of surface with a burst of class No.1 defects is visible in *Figure 2*. The shape distribution of a single defect is illustrated in *Figure 4*. This is drawn by super-position of all the defects of the same type, centered in the middle of the figure, and counting the relative frequencies of each pixel.

Defect No.1 dimensions distribution is shown in *Figure 5*. In *Table I* the results of some distribution hypothesis tests are presented. The null hypotheses are data following a particular distribution (e.g. Lognormal, Normal, Weibull), with their parameters tuned using their best sample estimators. It is visible that it is not possible to infer the distribution, and it is eventually better to use a kernel distribution.

Therefore, no information for the tuning of the architecture hyperparameters was extrapolated from data distribution.

2) Defect class No.2: defects of class No.2 usually appear near the transversal edge. Hence, they are probably edge laminations, since they are also visually similar.

These defects are local, since they are usually near the edge. Indeed, from the histogram in *Figure 7* is visible that they occur only a small limited number of times on the same surface.

In *Figure 6*, an example of surface with a defect of this type is shown. The shape distribution of the defect is illustrated in *Figure 8*.

Defect No.2 dimensions distribution is shown in *Figure 9*. In *Table II* the results of some distribution hypothesis tests are

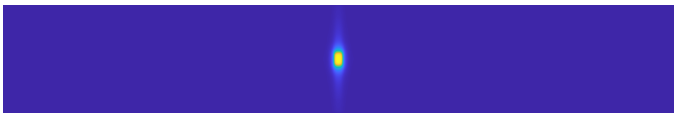

Figure 4. Shape distribution for defect class No.1.

Length distribution	p-value	H_0 rejected
BirnbaumSaunders	1.45633e-138	True
Exponential	0.00000e+00	True
ExtremeValue	0.00000e+00	True
Gamma	2.51653e-184	True
GeneralizedExtremeValue	5.31383e-20	True
GeneralizedPareto	0.00000e+00	True
HalfNormal	2.03112e-267	True
InverseGaussian	2.88195e-125	True
Kernel	1.30866e-03	False
Logistic	0.00000e+00	True
Loglogistic	8.99770e-75	True
Lognormal	1.15810e-117	True
Nakagami	4.23182e-274	True
NegativeBinomial	1.77777e-163	True
Normal	0.00000e+00	True
Poisson	0.00000e+00	True
Rayleigh	2.35767e-68	True
Rician	1.73950e-222	True
Stable	1.20349e-59	True
tLocationScale	1.18919e-200	True
Weibull	1.28151e-120	True
Height distribution	p-value	H_0 rejected
BirnbaumSaunders	0.00000e+00	True
Exponential	0.00000e+00	True
ExtremeValue	0.00000e+00	True
Gamma	0.00000e+00	True
GeneralizedExtremeValue	2.12450e-31	True
GeneralizedPareto	0.00000e+00	True
HalfNormal	0.00000e+00	True
InverseGaussian	0.00000e+00	True
Kernel	4.72288e-01	False
Logistic	0.00000e+00	True
Loglogistic	3.26019e-277	True
Lognormal	0.00000e+00	True
Nakagami	0.00000e+00	True
NegativeBinomial	0.00000e+00	True
Normal	0.00000e+00	True
Poisson	0.00000e+00	True
Rayleigh	0.00000e+00	True
Rician	0.00000e+00	True
Stable	5.27607e-14	True
tLocationScale	1.30243e-315	True
Weibull	0.00000e+00	True

Table I. Hypotheses test results on class No.1.

Length distribution	p-value	H_0 rejected
BirnbaumSaunders	4.94697e-08	True
Exponential	4.95301e-242	True
ExtremeValue	2.56607e-118	True
Gamma	2.49894e-12	True
GeneralizedExtremeValue	7.91295e-12	True
GeneralizedPareto	7.64274e-140	True
HalfNormal	4.99423e-142	True
InverseGaussian	3.96727e-08	True
Kernel	7.72253e-02	False
Logistic	3.97960e-41	True
Loglogistic	2.42820e-18	True
Lognormal	2.24200e-09	True
Nakagami	2.27381e-21	True
NegativeBinomial	2.03218e-14	True
Normal	2.14626e-43	True
Poisson	0.00000e+00	True
Rayleigh	7.33838e-40	True
Rician	6.99744e-34	True
Stable	5.01488e-24	True
tLocationScale	7.57445e-43	True
Weibull	1.51970e-27	True
Height distribution	p-value	H_0 rejected
BirnbaumSaunders	1.57970e-18	True
Exponential	9.06864e-198	True
ExtremeValue	2.14881e-61	True
Gamma	2.09598e-18	True
GeneralizedExtremeValue	3.48901e-134	True
GeneralizedPareto	2.06040e-81	True
HalfNormal	5.82180e-126	True
InverseGaussian	3.95275e-19	True
Kernel	1.55970e-08	True
Logistic	5.33847e-35	True
Loglogistic	4.76467e-15	True
Lognormal	7.77929e-18	True
Nakagami	6.40566e-22	True
NegativeBinomial	4.77487e-18	True
Normal	6.42093e-32	True
Poisson	0.00000e+00	True
Rayleigh	6.76946e-36	True
Rician	1.63582e-28	True
Stable	1.70156e-33	True
tLocationScale	1.01581e-32	True
Weibull	7.30485e-29	True

Table II. Hypotheses test results on class No.2.

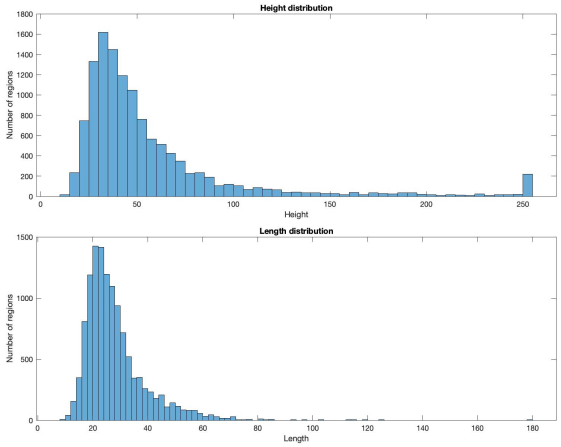

Figure 5. Defect class No.1 dimensions distribution.

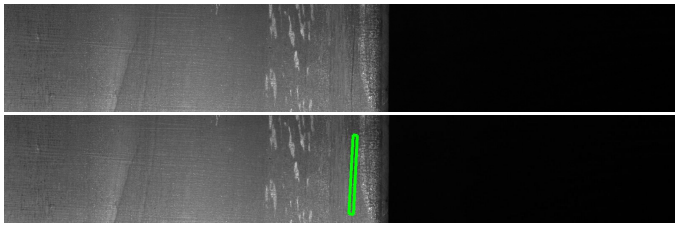

Figure 6. Steel surface with defect class No.2.

reported. It is clear that it is not possible to model the height with any distribution, whereas the length may be modeled with the kernel distribution.

3) Defect class No.3: the mill rolls should be perfectly parallel to correctly flatten the steel. When this is not the case, a stress pattern arises, with tension along the centreline.

Defects of class No.3 are probably rolling defects, in particular their repetitive pattern along the centreline is a symptom of *zipper cracks*, i.e. centre line cracking. This is another glaring example of burst defect.

The hypothesis of transversality of the presence of a defect

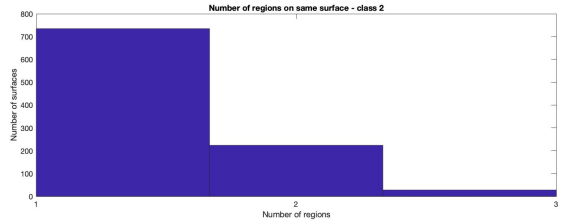

Figure 7. Number of defects of class No.2 per defective surface.

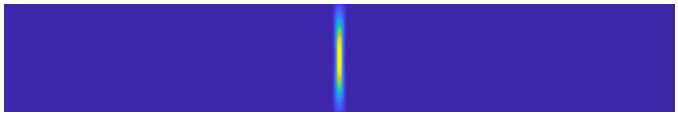

Figure 8. Shape distribution for defect class No.2.

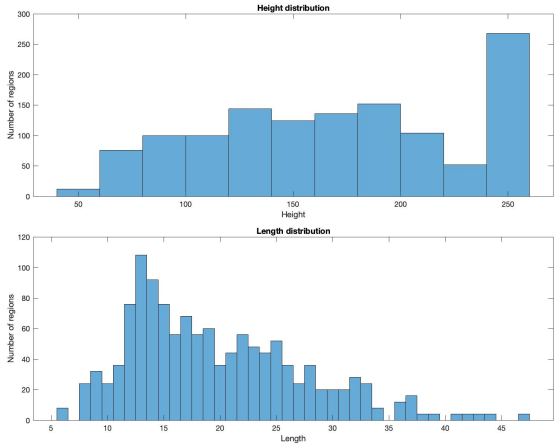

Figure 9. Defect class No.2 dimensions distribution.

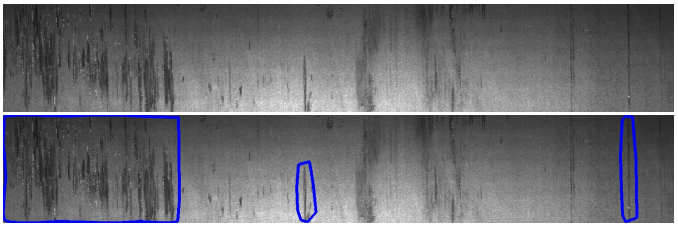

Figure 10. Steel surface with defect class No.3.

of class No.3 over the surface is supported by the histogram in *Figure 11*. However, as extrapolated from a visual analysis (as an example, consider *Figure 10*), many defects of class No.3 are encoded together, since they are very near one another. Hence, many of these are considered as an atomic region.

An example of surface affected by defects of class No.3 is shown in *Figure 10*. A single defect of class No.3 (i.e. a single crack) has usually the shape described in *Figure 12*.

Defect No.3 dimensions distribution is shown in *Figure 13*. In *Table III* the results of some distribution hypothesis tests are reported. Conclusions are the same discussed in *Section*

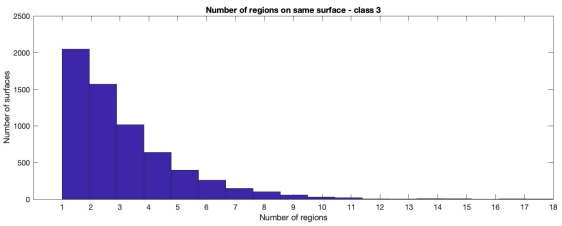

Figure 11. Number of defects of class No.3 per defective surface.

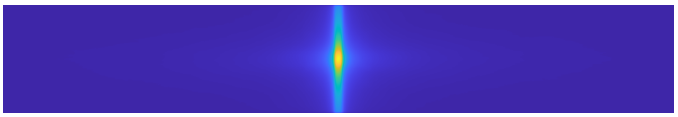

Figure 12. Shape distribution for defect class No.3.

Length distribution	p-value	H_0 rejected
BirnbaumSaunders	0.00000e+00	True
Exponential	0.00000e+00	True
ExtremeValue	0.00000e+00	True
Gamma	0.00000e+00	True
GeneralizedExtremeValue	4.02649e-52	True
GeneralizedPareto	4.83663e-85	True
HalfNormal	0.00000e+00	True
InverseGaussian	0.00000e+00	True
Kernel	1.00000e+00	False
Logistic	0.00000e+00	True
Loglogistic	0.00000e+00	True
Lognormal	0.00000e+00	True
Nakagami	0.00000e+00	True
NegativeBinomial	0.00000e+00	True
Normal	0.00000e+00	True
Poisson	0.00000e+00	True
Rayleigh	0.00000e+00	True
Rician	0.00000e+00	True
Stable	6.99084e-44	True
tLocationScale	0.00000e+00	True
Weibull	0.00000e+00	True
Height distribution	p-value	H_0 rejected
BirnbaumSaunders	0.00000e+00	True
Exponential	0.00000e+00	True
ExtremeValue	0.00000e+00	True
Gamma	0.00000e+00	True
GeneralizedExtremeValue	0.00000e+00	True
GeneralizedPareto	0.00000e+00	True
HalfNormal	0.00000e+00	True
InverseGaussian	0.00000e+00	True
Kernel	1.26501e-112	True
Logistic	0.00000e+00	True
Loglogistic	0.00000e+00	True
Lognormal	0.00000e+00	True
Nakagami	0.00000e+00	True
NegativeBinomial	0.00000e+00	True
Normal	0.00000e+00	True
Poisson	0.00000e+00	True
Rayleigh	0.00000e+00	True
Rician	0.00000e+00	True
Stable	0.00000e+00	True
tLocationScale	0.00000e+00	True
Weibull	0.00000e+00	True

Table III. Hypotheses test results on class No.3.

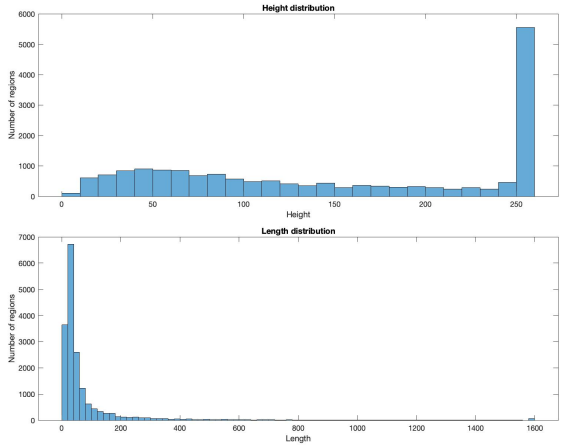

Figure 13. Defect class No.3 dimensions distribution.

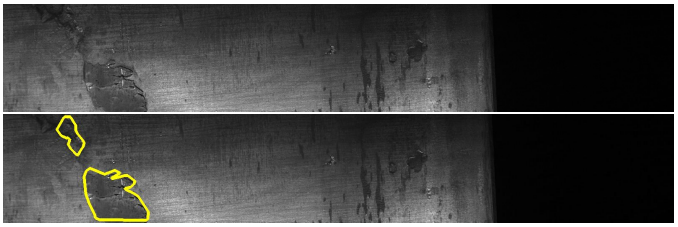

Figure 14. Steel surface with defect class No.4.

II-B2.

4) Defect class No.4: this defective class seems to be concerned with protrusions on the steel surface. The main typologies of protuberances in the given dataset are *scabs* and *blisters*. Scabs are flattened protrusions, and they tend to be round or oval shaped and concentrated to only certain blooms or billets. Blisters, or gas porosities, are small bulges on the surface of the components and their dimension can vary. Some gasses may remain trapped inside the steel sheet. The high pressure due to the rolling process produces then protrusions on the surface.

An example of surface affected by defects of class No.4 is illustrated in *Figure 14*. A single defect of class No.4 (i.e. a single crack) has usually the shape described in *Figure 15*.

Defect No.4 dimensions distribution is shown in *Figure 16*. In *Table IV* the results of some distribution hypothesis tests are reported. Conclusions are the same discussed in *Section II-B1*.

Even this defect class has global characteristics, as confirmed by the histogram in *Figure 17*.

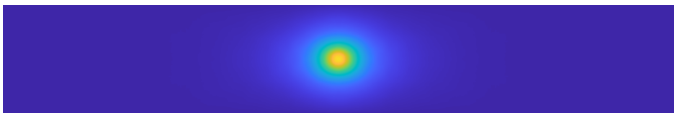

Figure 15. Shape distribution for defect class No.4.

Length distribution	p-value	H_0 rejected
BirnbaumSaunders	2.05128e-29	True
Exponential	3.56392e-59	True
ExtremeValue	0.00000e+00	True
Gamma	6.23653e-96	True
GeneralizedExtremeValue	1.40571e-16	True
GeneralizedPareto	2.63219e-64	True
HalfNormal	1.75462e-284	True
InverseGaussian	4.94670e-35	True
Kernel	9.11631e-01	False
Logistic	0.00000e+00	True
Loglogistic	2.07045e-18	True
Lognormal	2.77764e-15	True
Nakagami	3.75484e-287	True
NegativeBinomial	4.04830e-97	True
Normal	0.00000e+00	True
Poisson	0.00000e+00	True
Rayleigh	0.00000e+00	True
Rician	0.00000e+00	True
Stable	1.18318e-41	True
tLocationScale	3.43097e-180	True
Weibull	1.99339e-106	True
Height distribution	p-value	H_0 rejected
BirnbaumSaunders	2.46342e-53	True
Exponential	0.00000e+00	True
ExtremeValue	0.00000e+00	True
Gamma	1.08592e-22	True
GeneralizedExtremeValue	7.60394e-51	True
GeneralizedPareto	6.51578e-254	True
HalfNormal	3.46716e-166	True
InverseGaussian	1.86974e-71	True
Kernel	6.66962e-03	False
Logistic	5.88596e-308	True
Loglogistic	2.23825e-38	True
Lognormal	1.21032e-34	True
Nakagami	6.73712e-75	True
NegativeBinomial	1.83090e-24	True
Normal	3.26083e-322	True
Poisson	0.00000e+00	True
Rayleigh	4.86382e-150	True
Rician	7.47629e-151	True
Stable	8.61147e-58	True
tLocationScale	1.97626e-323	True
Weibull	5.44922e-59	True

Table IV. Hypotheses test results on class No.4.

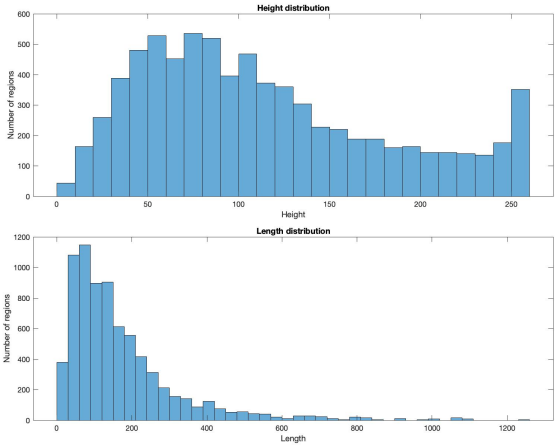

Figure 16. Defect class No.4 dimensions distribution.

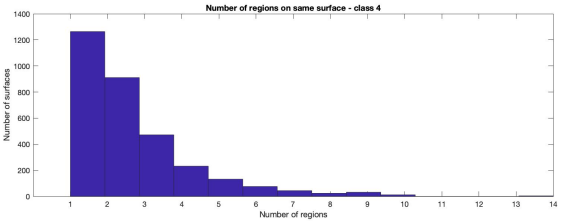

Figure 17. Number of defects of class No.4 per defective surface.

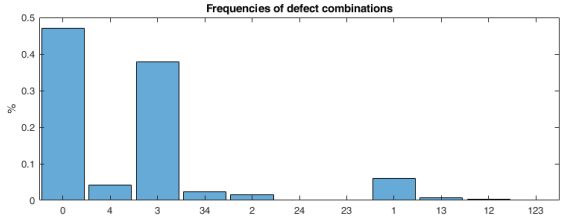

Figure 19. Frequencies of defects combinations.

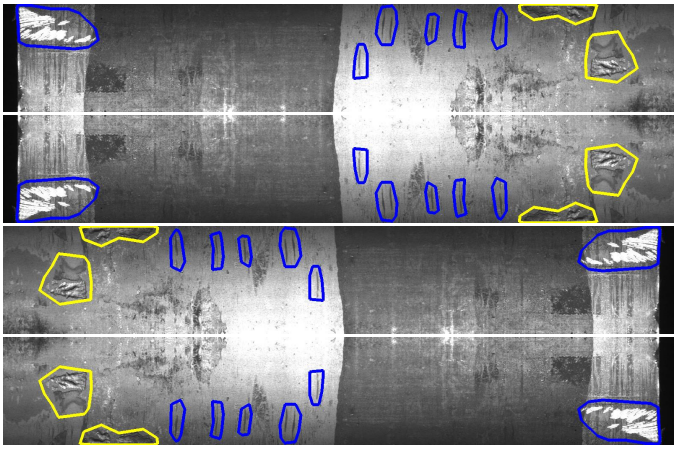

Figure 20. Data augmentation. The image is flipped horizontally, vertically and both. The defective area is moved coherently.

C. Dataset considerations

The dataset is composed by images linked to up to four strings of RLE encoded pixels, one per defective class.

In *Figure 18* the relative representation of the classes in the original dataset is reported. It is patent that class No.3 is far more represented then the other defective classes, and the number of surfaces with at least a defect of this class is nearly tantamount the number of flawless surfaces.

In *Figure 19* images were grouped in mutually exclusive subsets, based on the combination of defects in the pictured surface.

Since skewed dataset lessen the effectiveness of machine learning algorithms, especially in predicting minority classes examples, data augmentation was performed only on those classes or combinations of classes with fewer elements.

In order to keep proper proportions and spatial information, replicas of surfaces were built only using simmetries. Therefore, from a single image other three were generated. An example of such operation is shown in *Figure 20*.

The encoded defective pixels must be mapped onto the new image. This can be easily done considering the binary matrix corresponding to the encoded pixels, flipping it and re-encoding the resulting map.

The resulting dataset is slightly more balanced. The bar chart in *Figure 21* shows the new representation of the different classes after data augmentation.

III. Architecture overview

The defect detection system architecture proposed in this paper is described in *Figure 22*.

Steel surfaces pictures of 1600×256 pixels are taken at the input of the process. Since they may be took under different

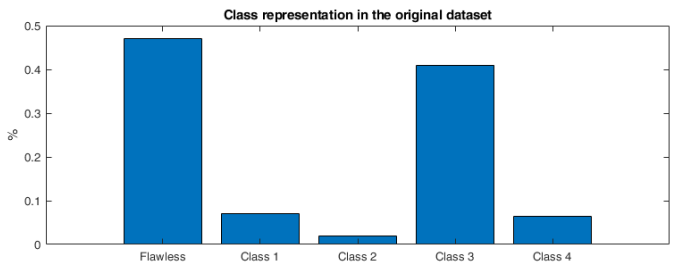

Figure 18. Class representation in the original dataset.

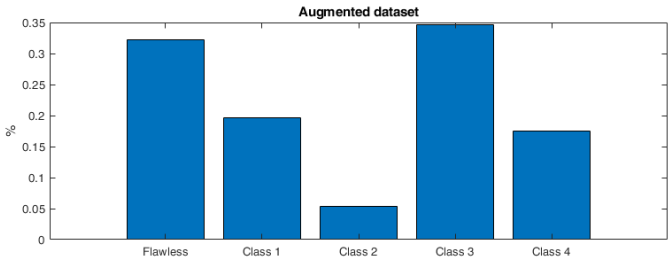

Figure 21. Class representation in the augmented dataset.

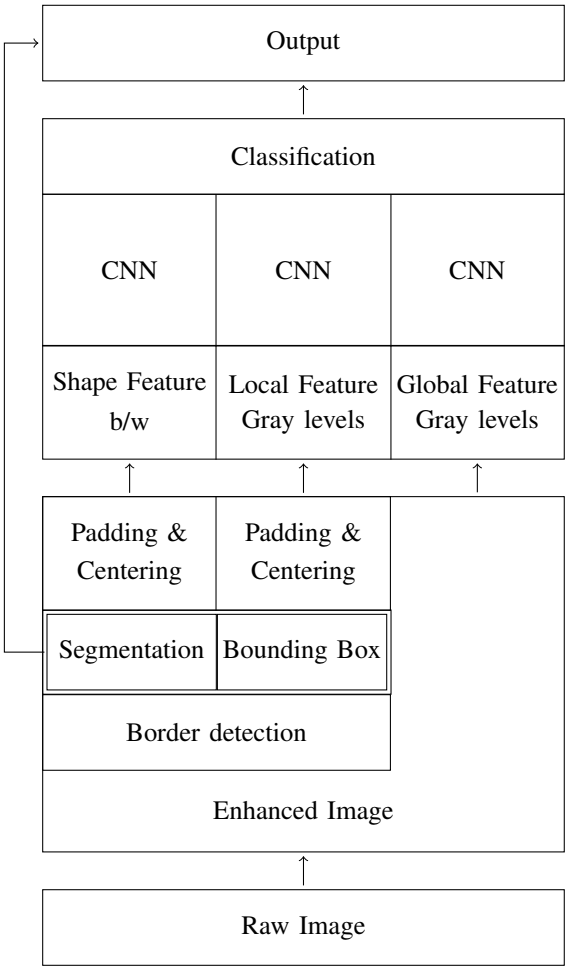

Figure 22. Proposed defect detection system architecture.

light exposure conditions, some preprocessing is made to enhance the quality of the image (e.g., histogram equalization or linear scaling). Moreover, the images considered have three equal colours levels, therefore they are converted into gray levels, to save space. This first step is further described in *Section IV*.

The aim of the architecture proposed in this paper is both to detect pixels representing steel imperfections and to classify those regions. Therefore, image segmentation is either obtained as an output of the system or it is needed in some step during the process. The latter situation can be achieved through a brute-force multi-scale sliding window, but to improve performances without reducing accuracy a particular implementation of a Region based CNN (R-CNN) [22, 23] is proposed in *Section VI*. This R-CNN uses a MC-CNN to combine and to consider separately information regarding interesting regions, which are further described in *Section V*. This reduces the computational cost of the system, compared with a naive sliding window.

Moreover, both local and global information are combined to improve classification accuracy. This approach avoids the complexity of combining different scale information and handling windows with different classes of defects. A further description of this approach is provided in *Section VI*, whereas in *Section VIII* a challenger architecture is described, to compare the results of the proposed system with.

One column of the MC-CNN is concerned with global information, and it is fed with the full enchanced image. Conceptually, this CNN learns to evaluate the probability of presence of the different types of defects in the whole surface picture. The other two columns consider local information instead. This local information is obtained from a further processing step, described in *Section V*. First, a contour detection algorithm (*Section V-A*) is used to spot proposals. Second, image segmentation (*Section V-B*) is done, to feed the MC-CNN only with some interesting regions. This segmentation results in a black and white (b/w) map describing the shape of the plausible defects. One column of the MC-CNN is fed with this map, therefore it learns to classify regions only observing their border. The other column is fed with the portion of original image enveloped in the bounding box (*Section V-C*) of the map, hence, it is trained to consider luminance levels inside, outside and on the border of the considered proposals.

Since defects may have different dimensions, the local information are centered in a 1600×256 pixels black image.

All the MC-CNN columns end with a softmax layer. However, they have different output size, as described in *Section VI*. Their results are then combined in order to properly classify the local regions.

Finally, if the classification outcome labels the region as defective, segmentation coordinates are kept. When all the proposals of the considered image are processed, defective pixels of the same class are encoded together with RLE algorithm. The surface is considered flawless, whenever all this encodings are empty.

IV. Image preprocessing

The first stage of the architecture is concerned with the enhancement of the raw image, in order to improve the learning effectiveness. In *Section VII* the contribution of preprocessing is evaluated.

Firstly, since given images have three equal colours levels, they can be considered gray-levels. Therefore, it is possible to shrink the space occupied on disk by discarding hue and saturation information and using only luminance.

Rec.ITU-R BT.601-7 calculates luminance ($E[y]$) as:

$$E[y] = 0.299 * R + 0.587 * G + 0.114 * B$$

where R, G, B are the three image channels. Since $R = G = B$, also $E[y] = R = G = B$, which justifies the assumption that discarding hue and saturation does not affect effectiveness of the system, whereas improving space and computational efficiency. Luminance is denoted by $E[y]$ since brightness is named y in literature, therefore the luminance, i.e. the physical intensity expected, is labeled in this way.

Secondly, since pictures may be took under different light exposure conditions, and since learning has heuristically be

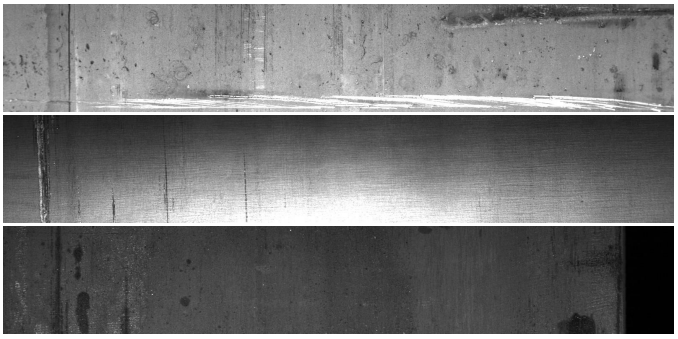

Figure 23. Sample images before preprocessing.

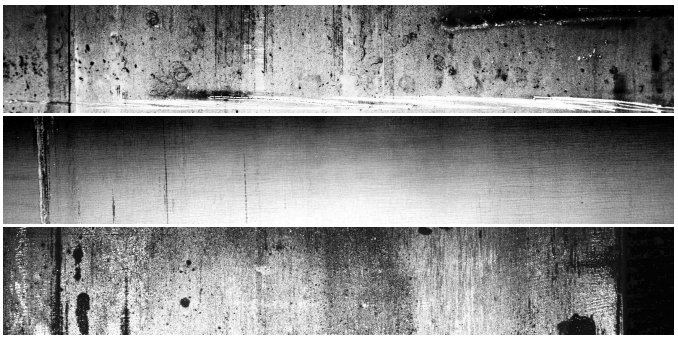

Figure 24. Sample images after preprocessing.

proven to be more effective if input assumptions are constant, the luminance histogram of the image is normalized.

Linear scaling ensures that all pixels luminances spread over all the range of possible values. $\mathcal{I}(x, y)$ refers to the luminance level of pixel (x, y) of image $\mathcal{I}$. Therefore, denoting with G_{max} the greater luminance level (tipically $2^k - 1$ for some k, e.g. $k = 8,\ G_{max} = 255$), the luminance scaled image is obtained as:

$$\mathcal{I}_{new}(x, y) = G_{max} \frac{\mathcal{I}(x, y) - \mathcal{I}_{min}}{\mathcal{I}_{max} - \mathcal{I}_{min}}$$

$$\mathcal{I}_{max} = \max_{x,y} \mathcal{I}(x, y)\ ; \quad \mathcal{I}_{min} = \min_{x,y} \mathcal{I}(x, y)$$

However, histogram equalization is preferred over linear scaling. Indeed, although both linear scaling and histogram equalization are effective in spreading over all the spectrum the luminance levels of an image, the former only ensures that all the intensities are used whereas the latter is also concerned about the shape of the resulting histogram, which ideally should be flat.

In fact, histogram equalization aims to transform a scalar image $\mathcal{I}$ such that all grey levels appear equally often in the transformed image $\mathcal{I}_{new}$, i.e.:

$$H_{\mathcal{I}_{new}}(u) = \text{const} = \frac{N_{cols} N_{rows}}{G_{max} + 1} \quad 0 \leq u \leq G_{max},$$

where N_{cols} and N_{rows} are, respectively, the number of columns and rows of the image. $H_{\mathcal{I}}(u)$ is the absolute frequency of luminance level u.

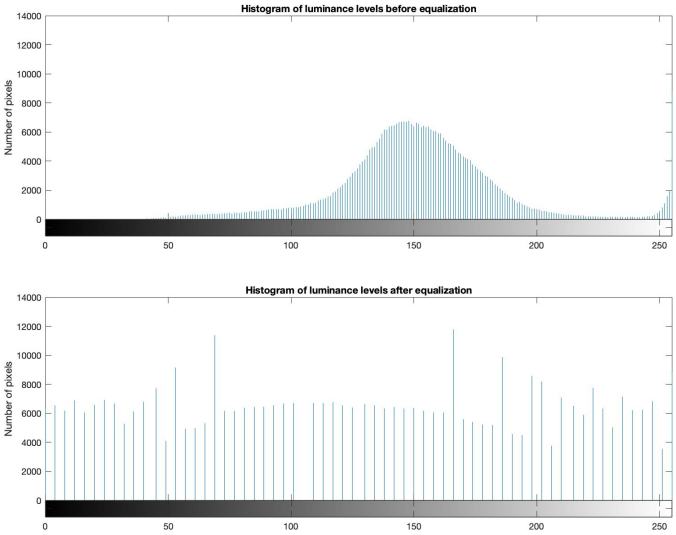

Figure 25. Histogram of luminance distribution on sample image before and after equalization.

However, this is not practically feasible, since identical value in $\mathcal{I}$ must be mapped on the same value of $\mathcal{I}_{new}$. Therefore, the transform is just an approximate solution. Intensities u in $\mathcal{I}$ are mapped onto new intensities $v = g(u)$ by the gradation function g:

$$g(u) = c_{\mathcal{I}}(u) \cdot G_{max},$$

where $c_{\mathcal{I}}$ is the relative cumulative frequency function.

In *Figure 25* the effects of equalization on the histogram of luminance of a sample image are shown. *Figures 23* and *24* show the preprocessing output on some samples images. It is clear that the different light exposures of the three images are compensated through the histogram equalization.

V. Detector

The second stage of the proposed architecture is the *detector* of region proposals, which are then fed into the *classifier* for classification. The latter is described in *Section* VI.

The whole system relies on the *detector* to spot one or more region of interest (ROI), which are the plausible candidates to be defective regions.

To extract them, firstly an edge detector is used (*Section V-A*). Its output is a binary matrix describing the edges found. The alpha shape of this matrix is then used to segmentate ROIs (*Section V-B*). Finally, the bounding box of the regions can be easily calculated (*Section V-C*).

A. Edge detection

The first step towards segmentation consists of the detection of edges within the image. The usage of Kovesi algorithm (KA) [31, 42] together with a hysteretic edge follower [30] is proposed.

KA is a *phase congruency*-based edge detector. The main idea is that wherever the local frequency components of an

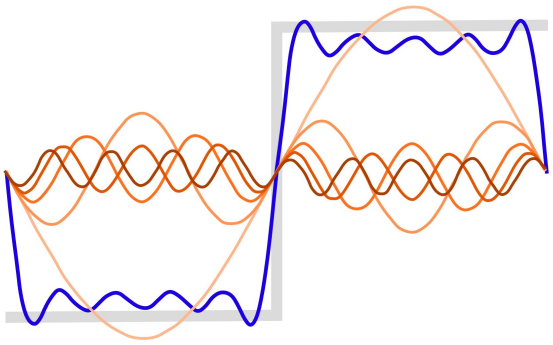

Figure 26. Visualization of the phase congruency concept for a 1D signal. The sum (blue curve) of the frequency components (red and orange curves) gradually approaches a discontinuity wherever the components are in phase.

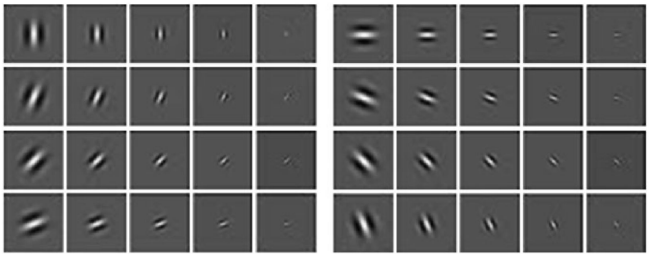

Figure 27. Examples of log-Gabor wavelets with different directions and scales.

image are in phase, there is an edge [43]. This concept is illustrated in *Figure 26*.

Some phase congruency metrics are based on Fourier analysis and on measuring the local energy [32], whereas KA exploits the log-Gabor wavelets. These are shown in *Figure 27*.

Given a image and fixed a direction u, for each pixel q a squared local window $W(q)$ is considered. $W(q)$ is an image crop centered in q. Then, it is convolved with the log-Gabor wavelets at different scales $n = 0 \dots N-1$, obtaining the complex number:

$$z_n(q) = G_n * W(q) = r_n(q) e^{i\alpha_n(q)}.$$

Then, the phase congruency measure at q is:

$$\mathcal{P}(q) = \frac{\max(0, |\sum_n z_n(q)| - T)}{\sum_n r_n(q) + \varepsilon} \in [0,1]\,,$$

where T is a constant that soothes the effects of noise and ε is a small value, used to avoid numerical problems concerning negligible values of r_n.

One of the advantages of this measure is that it is dimensionless. The concept behind this metric is illustrated in *Figure 28*. $W(q)$ sizes are $(2k+1) \times (2k+1)$, with $2k+1 \geq 3\lambda$.

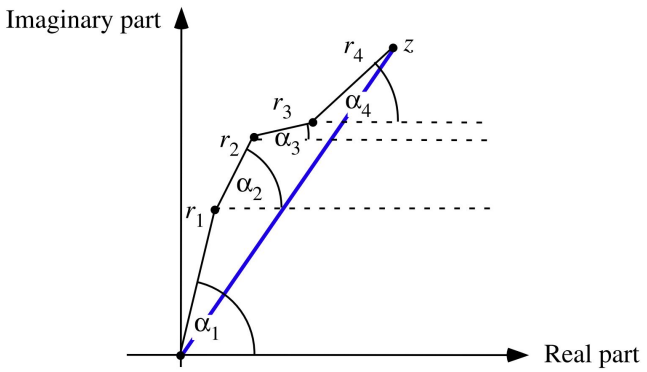

Figure 28. Phase congruency metric concept.

As discussed in *Section VII*, the *min-max normalization*:

$$\hat{\mathcal{P}}(q) = 255 \cdot \frac{\mathcal{P}(q) - \min_q \mathcal{P}(q)}{\max_q \mathcal{P}(q) - \min_q \mathcal{P}(q)}$$

on KA output improves the performance of the detector stage.

KA is based on several parameters that need to be tuned to obtain an effective edge detector. In this paper, the *number of scales* N, the *number of orientations* U, the *minimum wavelength* λ and the *scaling factor* s of the wavelets are considered. Their tuning was performed through Bopt [36, 37, 44], as described in section V-B.

It is not recommend to use directly the output of KA for image segmentation, since it is tipically susceptible to isolated edge-like patterns that arise in sparse locations within the image. Hence, a hysteretic edge follower was used to discard such patterns.

Let $\mathcal{I}$ be the image, $\mathcal{Q} = \left\{q \in \mathcal{I} \colon \hat{\mathcal{P}}(q) > T_{high}\right\}$, $\Omega(q)$ the set of pixels adjacent to q and $\mathcal{E} =$ the set of edge pixels.

Then, the hysteretic edge follower calculates edge pixels as:

1) $q_i \in \mathcal{Q}$; $\mathcal{Q} = \mathcal{Q} \setminus \{q_i\}$; $\mathcal{E} = \mathcal{E} \cup \{q_i\}$
2) $\forall q_j \in \Omega(q_i)$ if $\hat{\mathcal{P}}(q_j) > T_{low}$ then $\mathcal{E} = \mathcal{E} \cup \{q_j\}$ and repeat (2) with $q_i = q_j$
3) if $\mathcal{Q} \neq \emptyset$ then repeat (1)

The final output is a binary matrix marking the positions of edge pixels. The hysteretic parameters *low-threshold* T_{low} and *high-threshold* T_{high} need to be properly tuned, in order to improve the effectiveness of edge detection, as illustrated in *Figure 29*. Therefore, they were free parameters during the Bopt, together with KA ones.

The described techniques `MATLAB` implementation is freely available at [45].

B. Image Segmentation

Given the plausible edges found by the previous step, image segmentation is performed banking on alpha shapes [35].

The Delaunay triangulation $\mathcal{D}$ od a set of point $\mathcal{S}$ is the subset of all triangles $T = \left\{(a,b,c) \subset \mathcal{S}^3, a \neq b, b \neq c, a \neq c\right\}$ such that $t \in T, x \in \mathcal{S} \Rightarrow x \notin \mathcal{C}(t)$, where $\mathcal{C}(t)$ is the circumcircle of t.

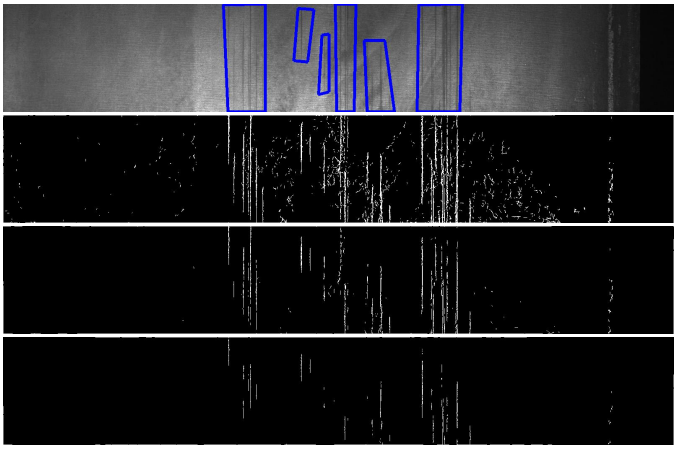

Figure 29. Output of the edge detector on a sample image (first from the top) using different thresholds (T_{low}, T_{high})*. From top to bottom:* $(30, 50)$*,* $(60, 100)$ *and* $(100, 150)$*. Default parameters of the Kovesi algorithm were used.*

The union of cells $c \in \mathcal{D}$ is called polytope of $\mathcal{D}$.

$$\mathcal{D}_\alpha = \{t \in T : r(\mathcal{C}(t)) < \alpha\},$$

where $r(\mathcal{C}(t))$ is the radius of the circumcircle of t. The polytope of $\mathcal{D}_\alpha$ is called *alpha-shape*.

As an example, in *Figure 30* the alpha-shapes with different α values of a set of points are compared. In *Figure 31* the convex-hull of the same set of points is shown. The choice of alpha shapes for the given task is clear.

However, in order to obtain topologically correct image segmentation, three parameters need to be properly chosen [35]. These are the *alpha-radius* α, the *hole-thresholding* T_{hole} and the *region-thresholding* T_{region} [46].

The first is the value of α, the second is the maximum area of interior holes that is filled and the third is the largest area that is suppressed (ignored). Observe that the latter is useful to soothe the effects of noisy edge detections.

However, tuning them manually is quixotic, hence they were choosen through Bopt.

The segmentation step outputs a set of pixels, which can be compared with the ideal segmentation, i.e. only the pixels within some true defective regions.

Therefore, the goodness of fit of the set of proposed pixels (X) to the ideal pixels (Y) can be measured as:

$$\chi(X, Y) = \frac{2|X \cap Y|}{|X| + |Y|} \in [0, 1].$$

Hence, the loss function is:

$$\mathcal{L}(X, Y) = 1 - \chi(X, Y) \in [0, 1].$$

The acquisition function proposed is *expected-improvement-plus* [47].

As a final consideration, using alpha shape lessen the detrimental effect of the noise in edge detection. Indeed, outliers

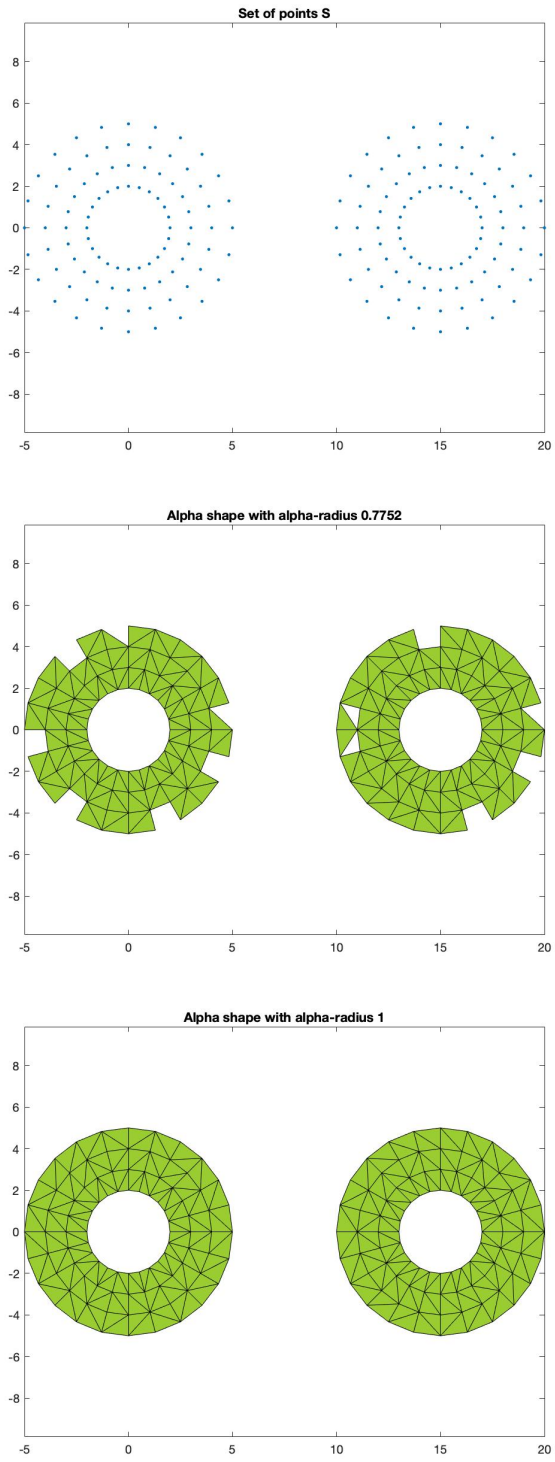

Figure 30. Alpha shape with different alpha values of a set of points.

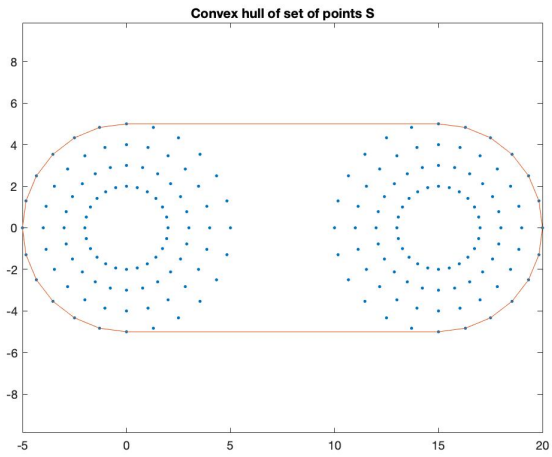

Figure 31. Convex hull of a set of points.

are suppressed by the hole threshold and region threshold parameters.

The regions of this alpha shape are the ROIs, which are fed into the *classifier*.

C. Bounding box

The bounding box of a ROI can be easily calculated considering the extremal coordinates of its pixels.

Let `pixels` be the $M \times 2$ matrix of the coordinates of the pixels within the ROI, with M the number of pixels. Then the bounding box is:

$$\texttt{bounding\_box} = \begin{bmatrix} \texttt{minX} & \texttt{minY} \\ \texttt{maxX} & \texttt{maxY} \end{bmatrix}$$

Where:

```
minX = min(pixels(:,1))
maxX = max(pixels(:,1))
minY = min(pixels(:,2))
maxY = max(pixels(:,2))
```

As a first approximation, the implementation of the *detector* was done to output the bounding boxes instead of the alpha shape. Therefore, the X set in *Section V-B* was characterized by all the pixels inside the bounding box of the alpha shape regions.

In *Figure 32* examples of ROIs extracted from a steel surface picture are illustrated. In the picture, also the ROIs bounding boxes are shown.

VI. Classifier

The ROIs are then fed into the third part of the proposed architecture, the *classifier*, and properly classified as flawless or flawed; in the latter case, they are assigned a defect class. The *classifier* is structured as a MC-CNN, which in general has the structure illustrated in *Figure 33*.

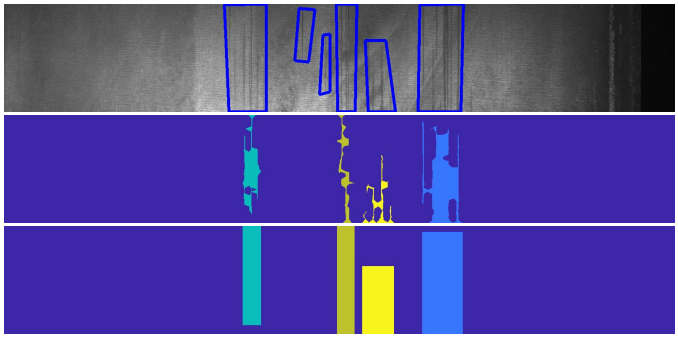

Figure 32. ROIs (middle) and relative bounding boxes (bottom) obtained on an example surface (top).

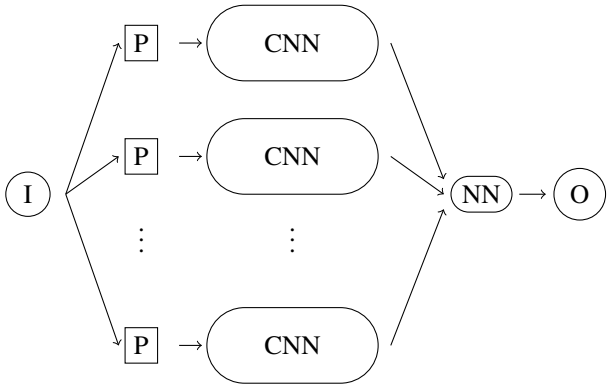

Figure 33. General structure of a MC-CNN.

First, the input image I is preprocessed to extract n column input P. Second, these P are fed into different CNNs in parallel, therefore all the columns are independent one another. Finally, the output of the MC-CNN columns is combined through a classifier, e.g. a neural network (NN), to produce the final output.

The choice of using a MC-CNN is due to several reasons, beyond the proved effectiveness described in [17]. Primarily, the training of a MC-CNN is highly parallelizable, indeed the different columns can learn separately one from another, once their correspective input is prepared.

Moreover, it is possible to merge both local and global information in a far easier way then using a traditional, single-column, CNN. Indeed, instead of focusing only on a rectangular area, which brings only the local information about the plausible defect, with a MC-CNN it is immediate to add another column concerned with the whole image.

This is the main point in favour of MC-CNN, since the class of a defect may be inferred using global patterns, such as the number of similar areas on the same surface. Imagine, for example, an error burst on the surface. Although traditional single-column CNNs may consider directly the input to the global column, this would face problems regarding the presence of multiple defects classes on the same surface. Therefore, a MC-CNN approach with some columns concerning local information and other focusing on global patterns is

heuristically better.

In *Section VIII* another approach to combine such information through a CNN is described. Although possible, is patently more convoluted then the MC-CNN approach. It would still be interesting to compare them in order to better evaluate the effectiveness of the proposed architecture.

Observe that the output of the *global column* must be calculated only once per each image, since it is constant throughout the single surface, and, thus, both the training and the predicting process can be lighten.

Finally, as an incidental outcome, it is possible to further study the amount of contribution of the different columns in accurately determining the defective class, if any, of the considered area. These considerations are reported in *Section VII*.

The proposed MC-CNN has three columns, namely a *shape*, a *local* and a *global* column.

The *classifier* was implemented before the *detector*. In fact, although the former relies on the latter for the ROIs and their relative features (described in *Section VI-A*, *VI-B* and *VI-C*), it was preempting supposed to have an ideal *detector*, i.e. one which proposes the optimal ROIs.

This effort-outcome oriented approach was done for two main reasons. Firstly, it highlights the upper bound reachable with the whole architecture. Secondly, dividing the *detector* outcome from the *classifier* input during the training allows to export the trained MC-CNN and to use it within the challenger architecture, proposed in *Section VIII*.

The three columns share the common ideas used in edge-cutting work [48] in their structure, beside some their idiosyncrasies. These are explained in *Section VI-A*, *VI-B* and *VI-C*.

The basic principles used in structuring the CNNs are mainly three:

1) The input layer should be a multiple of a high power of 2. Common dimensions are 32 (e.g. CIFAR-10), 64, 96 (e.g. STL-10), or 224 (e.g. common ImageNet ConvNets), 384, and 512. However, the approach described in this paper deals with input images of various size.
2) The convolutional layers should involve small filters (e.g. 3×3 or 5×5), using a unitary stride, and it should not alter the spatial dimensions of the input. Hence, proper padding (e.g., $[1, 1, 1, 1]$ for 3×3 filters) should be added.
3) The reduction of the size of the input should be due to the pooling layers, tipically a 2×2 downsampling with a 2×2 stride.

A. Shape column

The *shape column* is concerned to learn from the shape of the proposed region. This is fed into the CNN as a binary matrix, in which ones represent points in the border of the area.

An example of this binary images is given in *Figure 34* (top). The shape is centered in a 1600×256 black image. Indeed, the size of the input is set to the largest area that could be found, i.e. a defect spanning over the entire surface.

Figure 34. Shape column input (top) and filled input (bottom).

Figure 35. Local column input.

The shape is centered to ensure that the classifier is translation independent.

The *detector*, since is not ideal, provides to the *classifier* also false positive ROIs, i.e. regions that are not defective. Therefore, this stage of the architecture need to be able to discard flawless proposed regions.

However, training the *shape column* to classify some shapes as not defective is not feasible. Indeed, there is not a flawless surface shape, and manually generating it would not be safe.

Hence, the *shape column* should only be trained to classify a region into one of the four defect classes, and to mark flawless proposals will be duty of the final classifier (*Section VI-D*).

Therefore, the output layer is a n-dimensional vector, where n is the number of defect classes. Each entry of this vector describes the confidence of the network in considering the input shape as related to one of the n defect classes.

However, from data analysis was observed that, sometimes, defective input defects of the same class are clustered together, since very near one another. This is usually true for defects of class No.3, which are burst defects and might be nearly adjacent. Therefore, the shape that would be extracted is the shape of the region and not the one of the defects. Hence, the network would not be properly trained.

For this reason, the *shape column* is left as a further development in *Section VIII*.

The *shape column* input could also be filled before to be fed into the CNN, as shown in *Figure 34* (bottom).

B. Local column

The *local column* is thought to consider luminance levels around the border of the defect, to learn from the local context. Similarly to the *shape column* (*Section VI-A*), this column is concerned only in classifying proposals into one of the four defective classes.

Therefore, the column is fed with the grey scale portion of the image inside the bounding box of the considered region. As an example, in *Figure 35* is illustrated a plausible input to the *local column*.

This grey scale portion is generated from the shape of the region and the original image.

Firstly, the bounding box of the shape is calculated. Secondly, the original image outside the bounding box is discarded. Finally, the cropped image is centered in a black 1600×256 picture. The reasons behind the centering and the dimensioning of the input are the same described in *Section VI-A*.

The layers of this CNN are shown in *Table V*. The output layer is analogous to the one in *Section VI-A*.

The peculiarity of this CNN lies in its first convolutional layer, named *spreader*. It is a novel application of the dilation factor.

Recall that given a $n \times n$ filter, a dilation of $[a\ b]$ applied to the given filter produces a $[(n-1)*a+1] \times [(n-1)*b+1]$ filter. The original entries are equally spreaded in the new filter, whereas the additional values are null.

As an example, the below transformation applies a dilation factor of $[3\ 5]$ to the given 2×2 filter.

```
 _   _                          _           _
| a b |    [3 5]              | a 0 0 0 0 b |
| c d |  ------>              | 0 0 0 0 0 0 |
 -   -                        | 0 0 0 0 0 0 |
                              | c 0 0 0 0 d |
                               -           -
```

An example on a 3×3 filter is proposed below:

```
 _     _                       _           _
| a b c |    [2 2]            | a 0 b 0 c |
| d e f |  ------>            | 0 0 0 0 0 |
| g h i |                     | d 0 e 0 f |
 -     -                      | 0 0 0 0 0 |
                              | g 0 h 0 i |
                               -           -
```

The dilation factor is usually used in CNNs to implement a wider receptive field, using the same number of weights. Indeed, a $K_1 \times K_2$ filter convolves pixels in an area of $K_1 \times K_2$. Using a dilation factor of $[a\ b]$ it would convolve the same number of pixels, but on an area of $[(K_1-1)*a+1] \times [(K_2-1)*b+1]$. Using a proper stride, this would be the same of a downsampling followed by a convolution.

However, in the proposed architecture it is used in a novel way, to address the problem of different-sized bounding boxes. In fact, the usage of the *spreader* layer aims to reproduce in different locations of the layer activation the original informative region, through some learned linear functions. Indeed, with a proper sizing, the convolution in the first layer results in the sum of the bias factor and a single informative pixel multiplied by a weight of the filter.

In *Figure 36* the learned features of four 3×3 filters of a traditional convolutional layer with padding $[673\ 1]$ are juxtaposed to the ones learned by four 3×3 filters with a dilation factor of $[400\ 400]$ and proper padding $[672\ 0]$.

The two pictures are scaled to be paired. However, the former is 1600×1600, the latter is 800×800. Still it is patent that the proposed approach greatly exceed the traditional one in terms of information kept.

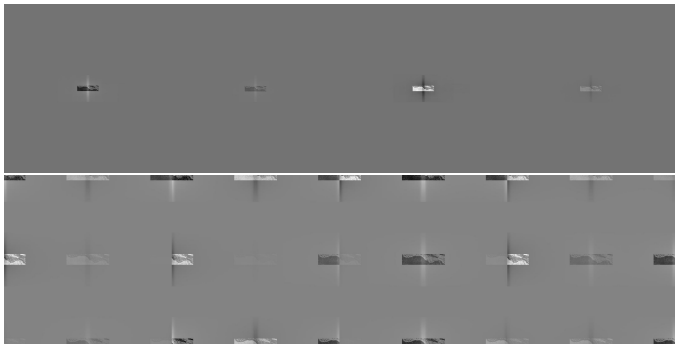

Figure 36. Learned features with a traditional convolutional layer (top) and with the spreader *(bottom).* 4 *tiled features are illustrated in both cases.*

Indeed, the problem with convolutional layer with massive padding (in alternative to cropping) is that large empty regions have null activations. Then, pooling layers keep the proportions between *death* area and the informative, central, region. This leads to massive memory waste.

A 3×3 filter with a dilation factor of half the output activation size and proper padding enhances the effectiveness of the following layers, recycling empty areas. In fact, they are convolved together with the central informative region.

Indeed, this approach ensures that at least 9 areas will convolve with the central region. The learned weights of the filter will then decide whether to copy, along with some multiplicative factor (which can be thought as varying the exposure of the image portion), the defect in these new regions.

Therefore, without any manual tuning, the network would eventually learn to handle large original pictures. As an example, if the dataset is composed by a cornucopia of large defects, the *spreader* may copy only vertically the defect, using only three of the nine available weights.

In the proposed architecture, four of this spreading filters are used in the first layer, conceptually in order to tune them one per defective class.

The *spreader* layer represents an effective alternative to either lossy crop or padding when dealing with either images of different dimensions or non-squared pictures.

By cropping the image, massive memory saving could be allowed. As an example, the 512×512 cropping proposed in *Table IX* would save $800 \times 800 - 512 \times 512 \approx 1\,\mathrm{MB}$ per image, about 59.04%.

However, the proposed spreader introduces $\times 9$ information compared to cropping, which is followed by activations similar, although with less memory consumption, to the ones shown in *Figure 36* (top).

There is patently a tradeoff between training time, memory (hence both batch and mini-bath size), number of weights and information loss.

Using a classical approach with these different-sized bounding boxes, with either cropping or padding, would conceptu-

Layer	Type	Activations	Learnables
Input $[256 \times 1600 \times 1]$ image "zerocenter" normalization	Image input	$256 \times 1600 \times 1 = 409.600$	
Spreader Filter 4 $[3 \times 3 \times 1]$ Stride $[1\ 1]$ Dilation factor $[400\ 400]$ Padding $[672\ 0]$	Convolution	$800 \times 800 \times 4 = 2.560.000$	Weights $3 \times 3 \times 1 \times 4 = 36$ Bias $1 \times 1 \times 4 = 4$ Total: 40
Conv1 Filter 8 $[3 \times 3 \times 4]$ Stride $[1\ 1]$ Padding "same"	Convolution	$800 \times 800 \times 8 = 5.120.000$	Weights $3 \times 3 \times 4 \times 8 = 288$ Bias $1 \times 1 \times 8 = 8$ Total: 296
MaxPool1, $[4 \times 4 \times 1]$ Stride $[4\ 4]$ Padding "same"	Max pooling	$200 \times 200 \times 8 = 320.000$	
Conv2 Filter 16 $[3 \times 3 \times 8]$ Stride $[1\ 1]$ Padding "same"	Convolution	$200 \times 200 \times 16 = 640.000$	Weights $3 \times 3 \times 8 \times 16 = 1152$ Bias $1 \times 1 \times 16 = 16$ Total: 1168
MaxPool2, $[4 \times 4 \times 1]$ Stride $[4\ 4]$ Padding "same"	Max pooling	$50 \times 50 \times 16 = 40.000$	
Conv3 Filter 32 $[3 \times 3 \times 16]$ Stride $[1\ 1]$ Padding "same"	Convolution	$50 \times 50 \times 32 = 640.000$	Weights $3 \times 3 \times 16 \times 32 = 4.608$ Bias $1 \times 1 \times 32 = 32$ Total: 4640
MaxPool3, $[2 \times 2 \times 1]$ Stride $[2\ 2]$ Padding "same"	Max pooling	$25 \times 25 \times 32 = 20.000$	
Conv4 Filter 64 $[3 \times 3 \times 32]$ Stride $[1\ 1]$ Padding "same"	Convolution	$25 \times 25 \times 64 = 40.000$	Weights $3 \times 3 \times 32 \times 64 = 18.432$ Bias $1 \times 1 \times 64 = 64$ Total: 18496
MaxPool4, $[2 \times 2 \times 1]$ Stride $[2\ 2]$ Padding "same"	Max pooling	$13 \times 13 \times 64 = 10.816$	
Conv5 Filter 128 $[3 \times 3 \times 64]$ Stride $[1\ 1]$ Padding "same"	Convolution	$13 \times 13 \times 128 = 21.632$	Weights $3 \times 3 \times 64 \times 128 = 73.728$ Bias $1 \times 1 \times 128 = 128$ Total: 73856
MaxPool5, $[2 \times 2 \times 1]$ Stride $[2\ 2]$ Padding "same"	Max pooling	$7 \times 7 \times 128 = 6.272$	
FullConn1	Fully connected	$1 \times 1 \times 16 = 16$	Weights $16 \times 6272 = 100.352$ Bias $16 \times 1 = 16$ Total: 100.368
FullConn2	Fully connected	$1 \times 1 \times 16 = 16$	Weights $16 \times 16 = 256$ Bias $16 \times 1 = 16$ Total: 272
FullConn3	Fully connected	$1 \times 1 \times 4 = 4$	Weights $4 \times 16 = 64$ Bias $4 \times 1 = 4$ Total: 68
Output	Softmax	$1 \times 1 \times 4 = 4$	

Table V. Local column CNN layers.

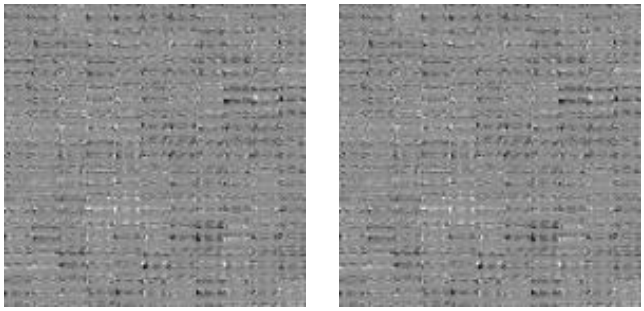

*Figure 37. Final layer features with classical architecture (*512 13×13 *features) (a) and with the spreader (*128 13×13 *features) (b).*

Figure 38. Global column input.

ally lead at the end to suppress informative areas, due both to pooling and convolving empty regions.

Instead, the proposed initial layer spreads over the squared activation matrix the defect, allowing to have more informative features in the final layer, as visible in the comparison in *Figure 37*. Indeed, the visual effect of equivalence between the two suggests that the initial *spreader* reduces the number of successive filters required, hence the number of parameters. A smaller number of parameters requires less data to prevent overfitting. If the outcomes are the same, it means that the model is better.

What can be heuristically inferred is that what is lost as memory efficiency, is far more gained as information. Hence, the memory can be saved in successive layers of the architecture. In the proposed architecture this is done through the first two max pooling layers, each of them introducing a shrinking factor of 4.

Moreover, performing a massive crop operation one accepts to misclassify a given percentage of the dataset. From an intuitive point of view, the *spreader* could learn to repeat the defect only vertically. For example, the filter:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

would simply replicate three times the image vertically. The takeaway is that this filter removes the manual tuning of the crop operation, while it improves the efficiency of the feature extraction stage of the CNN.

However, further investigating the effectiveness of the *spreader*, compared to the crop operation, is proposed in *Section VIII*.

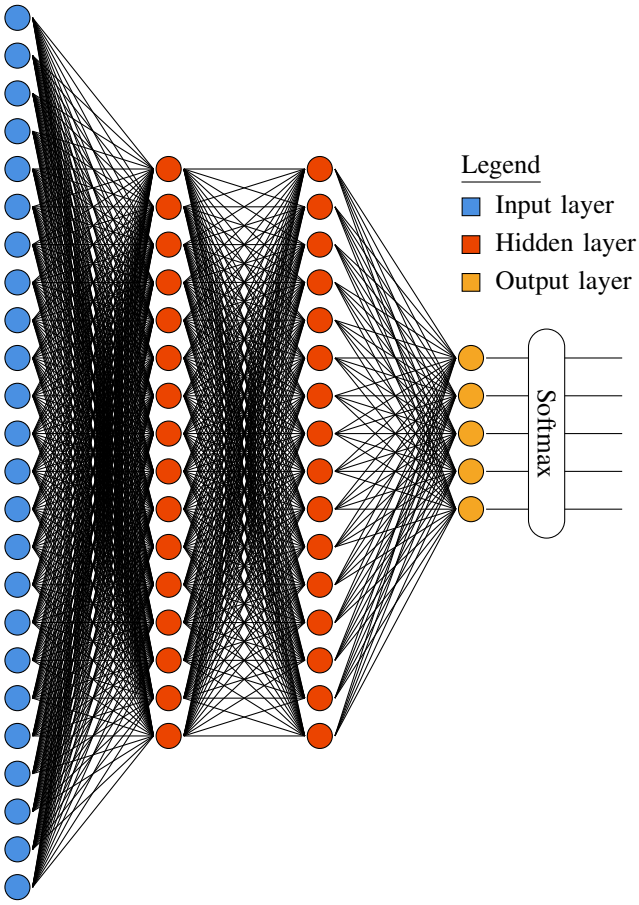

Figure 39. Final classifier structure.

C. Global column

The *global column* is concerned in detecting global patterns, such as zipper cracks. Hence, it is fed with the whole image. An example of input is shown in *Figure 38*. The importance of this column is clear from the observations made in *Section II-B*. Indeed, when two different classes are locally equal, the *global column* is determinant to correctly classify the defective region.

Ideally, the *global column* should output a 16-dimensional vector, with one entry per combination of classes. However, the dataset is very skewed when accounting for different classes combinations, as illustrated in *Figure 19*. Hence, the output layer has to be structured in a different way. Moreover, the inner structure of the CNN should be deeper then the other two, and, thus, requires a larger amount of data. Since longer training times and more hardware resources are required, the *global column* is left as a further development in *Section VIII*.

D. Final classifier

The outcomes of the different columns of a MC-CNN are then combined to obtain the final output. In the proposed architecture, this combination is done through a neural network, since a manual tuning of such data fusion would be ineffective.

In *Figure 39* the structure of the last layer of the architecture is described. The first layer is constrained to the output size of the three columns, i.e. $4+4+16=24$. Then there are two hidden layers with 16 activation units, and finally the output layer with 5 neurons. The output is passed through a softmax layer, hence the final output describe the confidence per each class and for the flawless regiono condition.

Observe that bias units have not been included in the picture.

The *final classifier* is left as further work, together with both *shape* and *global columns*, in *Section VIII*.

VII. Results

Although the proposed architecture was not fully implemented, both the *detector* and the *classifier* were proven effective. Indeed, the concept proof of the described defect detection system on steel surfaces was proven by the results reported in *Section VII-A* and *Section VII-B*.

From results in *Section VII-B* one can infer that the proposed novel usage of the dilatation factor in the first convolutional layer represents an effective alternative to cropping when dealing with images of different sizes.

A. Detector

The Bayesian optimization was performed using batches of 128 images and 50 iterations. During each iteration, the average loss function was computed over the batch and used to predict the next point in the hyperparameters domain.

After the optimization procedure, the trained *detectors* were tested using batches of 1024 images (720 for the class 3) and computing on them the accuracy measure:

$$\mathcal{A} = \frac{|X \cap Y|}{|Y|} \in [0,1]\,.$$

In *Table VI* are reported the average loss and accuracy obtained by the trained *detectors*, either using or not the equalization and the MinMax normalization on the images. Recall that the defects were localized with bounding boxes. Hence, overhead pixels were intrinsic to the implementation and, thus, the loss function had a non-null lower bound.

It is clear that in both the equalized and non-equalized settings, the MinMax normalization improves the performance of each *detector*. Indeed, although the accuracy increases using normalization, the loss value increases too. This means that the localization of more defects is not due to an increase in the *detector* precision, but only to larger bounding boxes and, thus, more overhead pixels.

Without equalization, accuracies 0.55, 0.77, 0.62 and 0.75 were reached for classes No.1, No.2, No.3 and No.4, respectively. Classes No.3 and No.4 achieved the lowest values of loss function. Hence, the detection of these defects is generally more precise and with a lower amount of overhead pixels.

In *Table VII* the optimal parameters values are reported. It is visible that tuned parameters reflect the shape of defects studied in *Section II-B*. In *Figures 40, 41, 42* and *43* are shown some examples of localization of defects from classes

No.	Equal.	MinMax	Batch	Loss	Accuracy
1	no	no	1024	0.8773	0.6234
	no	yes	1024	**0.8378**	**0.5477**
	yes	no	1024	0.9205	0.6752
	yes	yes	1024	0.7977	0.4925
2	no	no	760	0.9103	0.7318
	no	yes	760	**0.9061**	**0.7738**
	yes	no	760	0.9676	0.9587
	yes	yes	760	0.9172	0.3333
3	no	no	1024	0.7382	0.6710
	no	yes	1024	**0.6852**	**0.6168**
	yes	no	1024	0.8161	0.9145
	yes	yes	1024	0.6995	0.4600
4	no	no	1024	0.6261	0.6190
	no	yes	1024	**0.6455**	**0.7528**
	yes	no	1024	0.6694	0.7334
	yes	yes	1024	0.6050	0.5550

Table VI. Comparison between the performances obtained by detectors trained on different classes (No.), either using or not the equalization and the MinMax normalization. The best results obtained, according to the study, are bolded.

Parameters	Class 1	Class 2	Class 3	Class 4
N	4	3	3	3
U	11	7	16	15
λ	4.2905	4.0612	4.5523	3.9422
s	1.7205	2.3772	2.9674	2.8169
T_{low}	43	54	49	53
T_{high}	242	164	126	106
α	12	13	12	19
T_{hole}	14168	14536	10651	10594
T_{region}	541	1158	682	1204

Table VII. Optimal parameters obtained for each class with Bayesian optimization (non-equalized images).

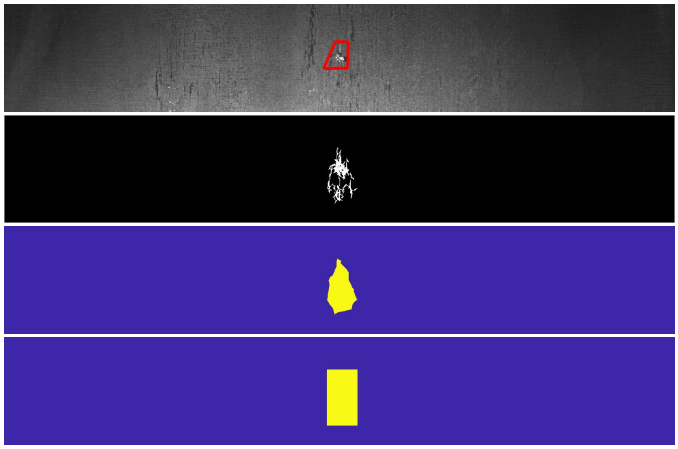

Figure 40. An example of the region proposal procedure performed on an image with a defect of class No.1.

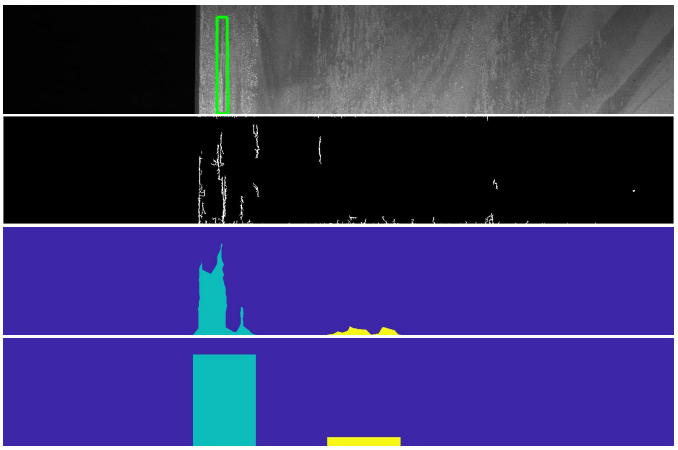

Figure 41. An example of the region proposal procedure performed on an image with a defect of class No.2.

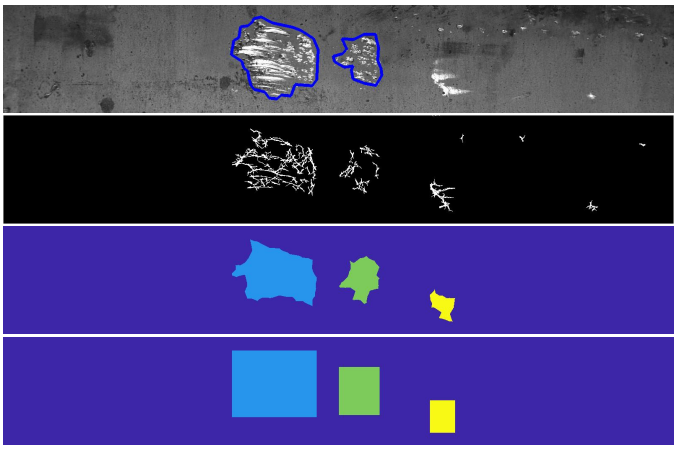

Figure 42. An example of the region proposal procedure performed on an image with a defect of class No.3.

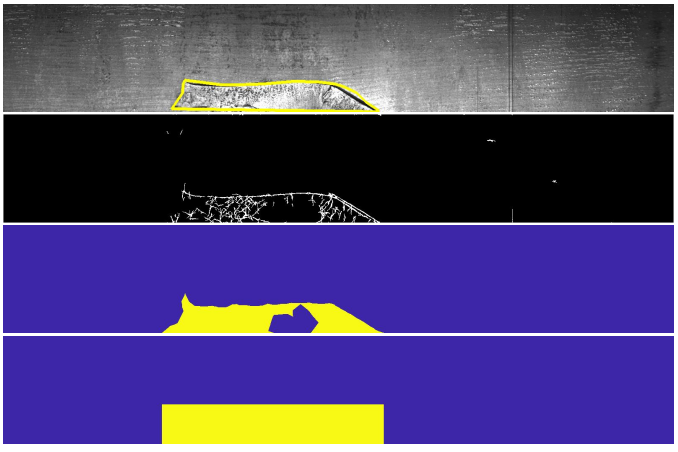

Figure 43. An example of the region proposal procedure performed on an image with a defect of class No.4.

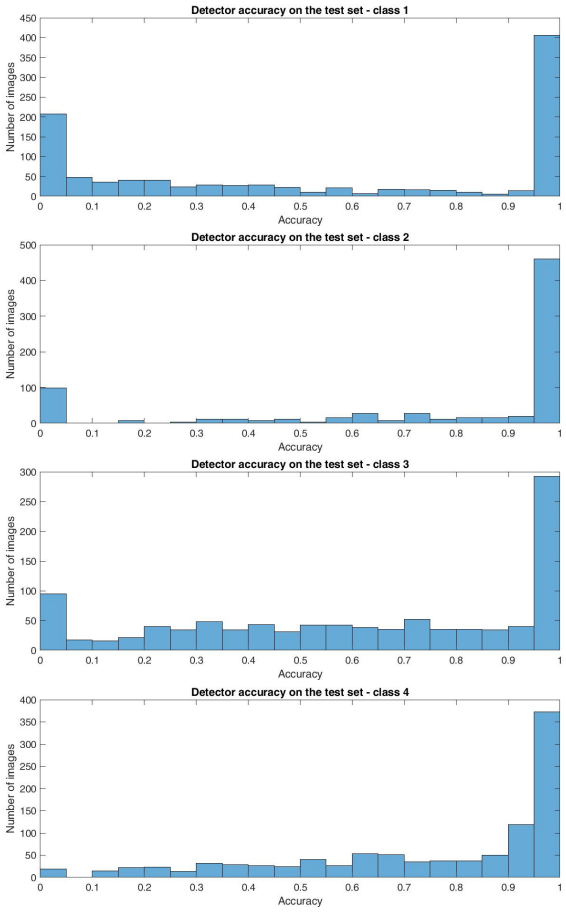

Figure 44. Accuracies distributions of the trained detectors, evaluated on their test sets (non-equalized images).

Parameters	Class 1	Class 2	Class 3	Class 4
N	3	4	5	4
U	15	7	5	15
λ	3.9199	2.3621	4.9810	3.5076
s	2.1161	1.5552	1.5559	1.4307
T_{low}	59	77	60	59
T_{high}	113	212	104	179
α	7	8	6	6
T_{hole}	9593	6460	7179	9986
T_{region}	510	942	727	1134

Table VIII. Optimal parameters obtained for each class with Bayesian optimization (equalized images).

No.1, No.2, No.3 and No.4 respectively. In *Figure 44* the test accuracies distribution for each *detector* are reported.

The distribution of accuracies in the equalized setting (*Figure 45*) differs greatly from the non-equalized one. Indeed, there is a larger amount of images with values of accuracies near zero. Average accuracies of only 0.49, 0.33, 0.46 and 0.55 were reached for classes No.1, No.2, No.3 and No.4, respectively.

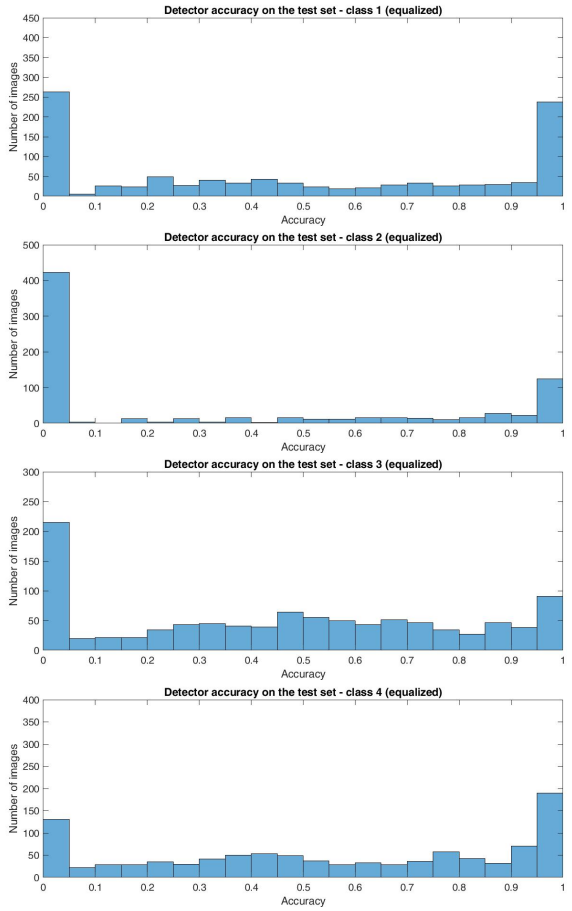

Figure 45. Accuracies distributions of the trained detectors, evaluated on their test sets (equalized images).

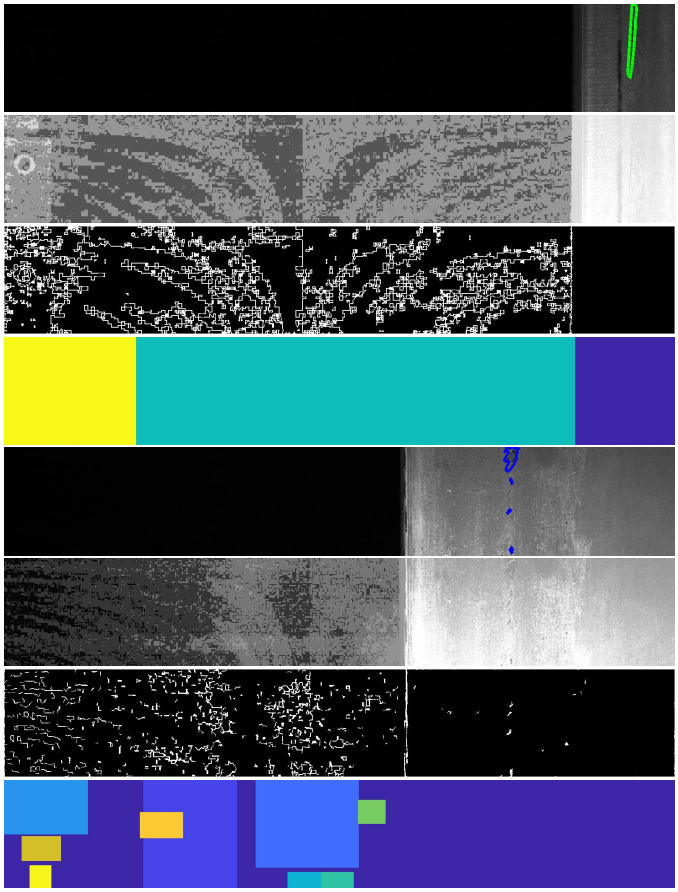

Figure 46. Effect of the equalization on the detector performance with images with large black regions.

The worsening of the results after equalization is probably due to the usage of a global (i.e. over all the image domain) equalization procedure. Indeed, noise is greatly increased by this equalization approach, wherever there are wide and nearly monochromatic regions (e.g., black backgrounds). Examples of these cases are shown in *Figure 45*. This was obviously detrimental to the optimization procedure. However, the optimal values found are reported in *Table VIII*.

Although the accuracies of the *detectors* on equalized images were lower, one can infer from the obtained losses that the proportion of overhead pixels is lower too. Hence, equalized *detectors* seems to be more precise. This is also confirmed by the performance of the *classifier*, which are superior on equalized images. Conceptually, the equalized images have higher contrast, and thus an edge detector is expected to identify better abrupt changes.

These results suggest that higher performances could be obtained by the *detectors* using a proper equalization procedure, dealing finer with the situations with non-negligible noise.

B. Classifier

The *local column* was trained with a batch of 1024 images, and mini-batch size of 256.

Figure 47. Local column confusion matrix on ideal input (with spreader layer).

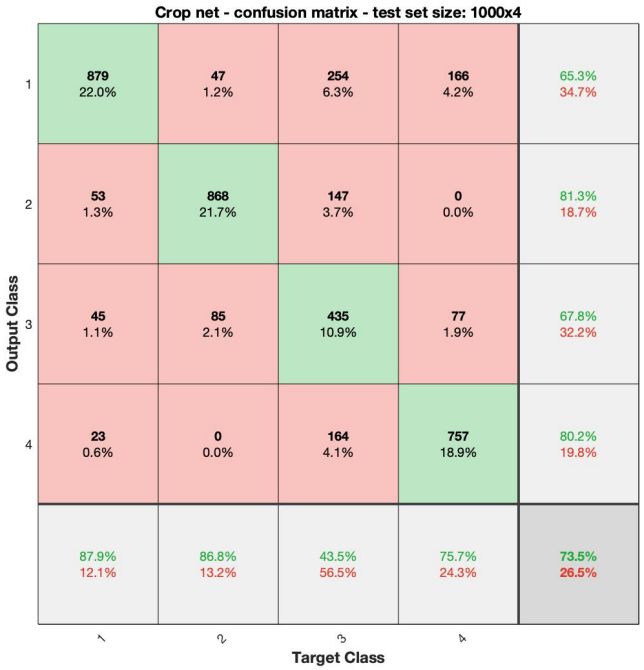

Figure 48. Local column confusion matrix on ideal input (with crop).

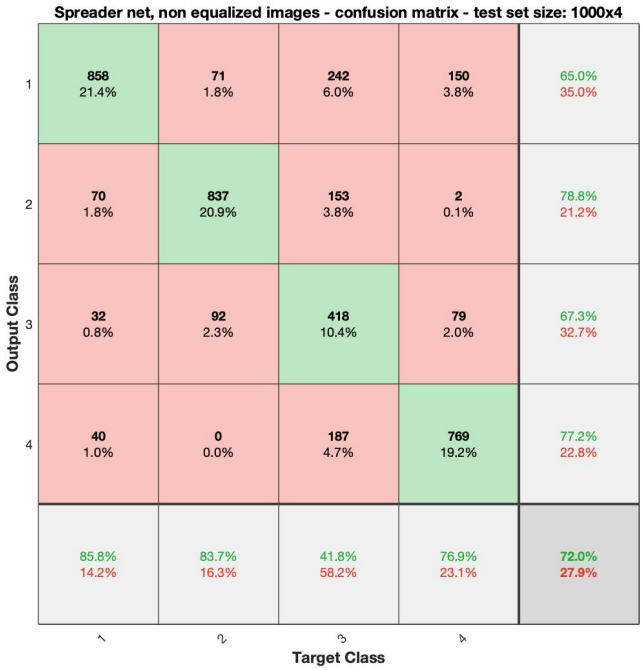

Figure 49. Local column confusion matrix on ideal input (images non-equalized).

The confusion matrix of the proposed architecture for the *local column* on a test set of 1000×4 defects is reported in *Figure 47*. It is visible that the defect class No. 3 is the most misclassified. This situation is probably due to the aggregation of multiple defects into the same defective region. Hence, the variability of the local features hamper the network learning. To evaluate the effectivenes of the spreader layer, a CNN with the same layers but without the novel layer was trained and tested with the same training options, i.e. batch size of 1024, mini-batch size of 256 and 30 epochs. In *Figure 48* its confusion matrix is illustrated.

It is interesting to notice that the network with the spreading layer achieves higher accuracy ($+0.7\%$) then the one which uses cropping instead. Moreover, the former overcome the latter in classifying defects of class No. 3 and No. 4. Indeed, these are the widest defects in the dataset. Hence, one can infer that the spreader layer deals better then cropping with larger images. This is visible also comparing the gained percentage with the outliers percentage, reported in the first row of *Table IX*.

However, neither one of the two CNNs overfits, and both can be trained longer. Moreover, deeper architectures can be used along with either image cropping or the spreading layer. With more powerful hardware, these could easily reach higher accuracies.

Then, the proposed *local column* structure was trained also with non-equalized images, but with the same training options. The resulting confusion matrix is reported in *Figure 49*.

It is visible that there is an actual improvement of 2.2% in the average accuracy when using equalized images. Hence,

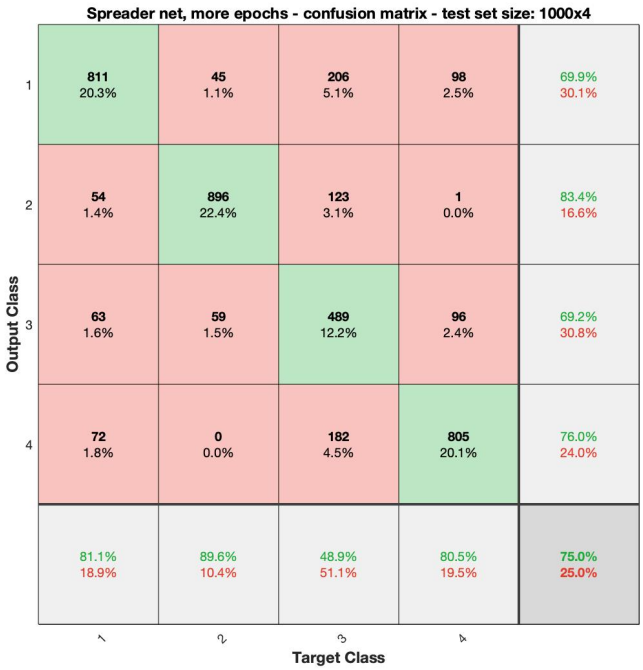

Figure 50. Local column confusion matrix on ideal input (with spreader, more epochs).

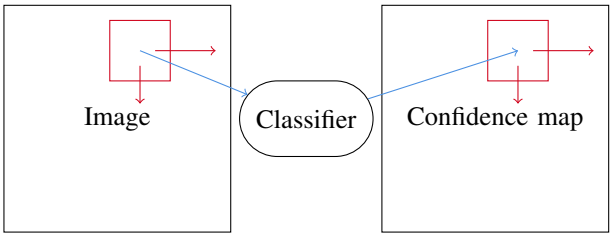

Figure 51. Sliding window architecture.

the architectural choice is clear.

The *local column* proposed was further trained, for other 40 epochs, and reached ans accuracy of 75%. However, it kept the same structure, and thus the same number of parameters. It is predictable that a deeper model with a larger training set will achieve state-of-the-art results. The updated confusion matrix is illustrated in *Figure 50*.

C. Architecture implementation

The whole system implementation can be found in the `GitHub` repository [40].

VIII. Further work

This work can be further developed in many ways.

A. Proposed architecture improvement

Firstly, it would be interesting to implement and train the *shape column*, preeptively discarding misleading data, and the *global column* and measure their contribution. The structure of the *shape column* may be identical to the *local column*, described in *Section VI-B*. Regarding the *local column*, some padding could be considered around defective regions, to fed the CNN also with some more pixels outside the border.

Secondly, the final classifier should be implemented to realize the end-to-end system and evaluate its performance.

Finally, as announced in *Section VI-B*, both the *local* and the *shape columns* could be fed with a cropped image. One could attempt to preemptively discard these outliers and try to optimize the other samples with a smaller input and, hence, more traditional classifiers. Some examples of interesting cropping size are shown in *Table IX*. A further investigation on the spreader layer effectiveness could be done.

B. Challenger

Some comparisons with a challenger architecture are needed to evaluate the effectiveness of the proposed system.

As an example, a well-known *sliding window* classifier could be used.

1) Sliding window architecture: the idea behind this architecture is to crop the input image at different locations (eventually all the ones possible) and use a classifier to assign to the pixels of the considered region an array of confidences.

This cropped regions may overlap. In those circumstances, an heuristic to combine different values of confidences is needed.

In *Figure 51* the sketch of this architecture is shown. In the illustration the resulting map, called *confidence map*, is associated to an array of confidences relative to the possible labels. The *confidence map* is then used along with some image segmentation tecnique to spot defective regions. In particular, a $n+1$ levels whatershed algorithm is proposed in *Section VIII-B2*.

Since defects may have different dimensions, more refined tecniques could be used to improve the accuracy of the challenger. However, the segmentation tecnique proposed in *Section VIII-B2* soothes this problem, since it combines local information from different areas to build the defective regions.

2) Image segmentation: the *confidence map* is used to segmentate the image through a $n+1$ levels whatershed algorithm. However, since the defective regions are always disjunct, the problem can be reduced to a binary whatershed algorithm [29] considering all the defective classes as one, and distinguishing them later.

As a final remark about the challenger, it is patent that even this naive implementation is far more involved then the proposed architecture, and the reason lies on the image segmentation approach.

The challenger could be further refined. As an example, a multi-scale approach could be considered, since the challenger does not take into account the variability of defects dimensions. To solve this flaw, *image pyramids* could be used.

Finally, an interesting development would be to investigate further a segmentation with overlapping (adjacent) regions, when there are more classes then foreground/background only. Towards this multi-level segmentation, it would be interesting to consider a non-binary whatershed-based algorithm.

References

[1] M. Sharifzadeh, S. Alirezaee, R. Amirfattahi, and S. Sadri, "Detection of steel defect using the image processing algorithms," in *2008 IEEE International Multitopic Conference*, December 2008, pp. 125–127.

[2] G. K. Nand, Noopur, and N. Neogi, "Defect detection of steel surface using entropy segmentation," in *2014 Annual IEEE India Conference (INDICON)*, December 2014, pp. 1–6.

[3] L. Chen and J. Deng, "Research on surface defects detection of stainless steel spoon based on machine vision," in *2018 Chinese Automation Congress (CAC)*, November 2018, pp. 1096–1101.

[4] Hongbin Jia, Y. L. Murphey, Jinajun Shi, and Tzyy-Shuh Chang, "An intelligent real-time vision system for surface defect detection," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 3, August 2004, pp. 239–242 Vol.3.

[5] J. Masci, A. Giusti, D. Ciresan, G. Fricout, and J. Schmidhuber, "A fast learning algorithm for image segmentation with max-pooling convolutional networks," in *2013 IEEE International Conference on Image Processing*, September 2013, pp. 2713–2717.

[6] A. Kumar and G. K. H. Pang, "Defect detection in textured materials using gabor filters," *IEEE Transactions on Industry Applications*, vol. 38, no. 2, pp. 425–440, March 2002.

[7] Y. Li and X. Di, "Fabric defect detection using wavelet decomposition," in *2013 3rd International Conference on Consumer Electronics, Communications and Networks*, November 2013, pp. 308–311.

[8] V. V. Karlekar, M. S. Biradar, and K. B. Bhangale, "Fabric defect detection using wavelet filter," in *2015 International Conference on Computing Communication Control and Automation*, February 2015, pp. 712–715.

Cropping size	Class No.1 outliers	Class No.2 outliers	Class No.3 outliers	Class No.4 outliers	Overall outliers
800×800	0%	0%	1.47%	1.06%	0.88%
512×512	0%	0%	2.94%	3.99%	2.11%
256×256	0%	0%	6.61%	17.57%	6.42%

Table IX. Cropping examples.

[9] H. Y. Ngan, G. K. Pang, and N. H. Yung, "Automated fabric defect detection—a review," *Image and Vision Computing*, vol. 29, no. 7, pp. 442 – 458, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885611000230

[10] M. Vetterli and J. Kovačevic, *Wavelets and Subband Coding*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[11] I. Daubechies, *Ten Lectures on Wavelets*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1992.

[12] R. Bernardini, "Wavelets for differential equations and numerical operator calculus [online first]," February 2019. [Online]. Available: https://www.intechopen.com/online-first/wavelets-for-differential-equations-and-numerical-operator-calculus

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

[14] R. Rojas, *Neural Networks: A Systematic Introduction*. Berlin, Heidelberg: Springer-Verlag, 1996.

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, May 2015.

[16] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed. Morgan-Kaufmann, 1990, pp. 396–404. [Online]. Available: http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf

[17] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 3642–3649.

[18] P. Sermanet, S. Chintala, and Y. LeCun, "Convolutional neural networks applied to house numbers digit classification," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, November 2012, pp. 3288–3291.

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[20] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*. London, UK, UK: Springer-Verlag, 1999, pp. 319–. [Online]. Available: http://dl.acm.org/citation.cfm?id=646469.691875

[21] M. M. Inc. Mill surface defects. [Online]. Available: https://mainlinemetals.com/resource-term-category/mill-surface-defects

[22] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, December 2015, pp. 1440–1448.

[23] X. Wang, H. Ma, and X. Chen, "Salient object detection via fast r-cnn and low-level cues," in *2016 IEEE International Conference on Image Processing (ICIP)*, September 2016, pp. 1042–1046.

[24] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv e-prints*, December 2013.

[25] J. Kuruvilla, D. Sukumaran, A. Sankar, and S. P. Joy, "A review on image processing and image segmentation," in *2016 International Conference on Data Mining and Advanced Computing (SAPIENCE)*, March 2016, pp. 198–203.

[26] "Picture thresholding using an iterative selection method," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 8, no. 8, pp. 630–632, August 1978.

[27] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, November 2012.

[28] Jianbo Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, August 2000.

[29] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583–598, June 1991.

[30] R. Klette, *Concise Computer Vision: An Introduction into Theory and Algorithms*. Springer Publishing Company, Incorporated, 2014.

[31] P. Kovesi, "Phase congruency detects corners and edges."

[32] M. Morrone and D. Burr, "Feature detection in human vision: A phase-dependent energy model," *Proceedings of the Royal Society of London. Series B, Containing papers of a Biological character. Royal Society (Great Britain)*, vol. 235, pp. 221–45, Genuary 1989.

[33] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, January 1988. [Online]. Available: https://doi.org/10.1007/BF00133570

[34] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, May 2004.

[35] P. Stelldinger, U. Köthe, and H. Meine, "Topologically correct image segmentation using alpha shapes," in *Discrete Geometry for Computer Imagery*, A. Kuba, L. G. Nyúl, and K. Palágyi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 542–554.

[36] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv e-prints*, p. arXiv:1807.02811, July 2018.

[37] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *arXiv e-prints*, p. arXiv:1206.2944, June 2012.

[38] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[39] Severstal. Severstal: Steel defect detection. [Online]. Available: https://www.kaggle.com/c/severstal-steel-defect-detection

[40] A. Terpin and C. Verardo. (2019) An effective approach to steel defect detection. [Online]. Available: https://github.com/antonioterpin/wavelet_ml

[41] Steeljrv.com. 64 pictures of common defects in strip steel. [Online]. Available: https://www.steeljrv.com/64-pictures-of-common-defects-in-strip-steel.html

[42] P. Kovesi, "Image features from phase congruency," 1995.

[43] M. Morrone and D. Burr, "Feature detection in human vision: A phase-dependent energy model," *Proceedings of the Royal Society of London. Series B, Containing papers of a Biological character. Royal Society (Great Britain)*, vol. 235, pp. 221–45, Genuary 1989.

[44] MATLAB. Bayesian optimization algorithm. [Online]. Available: https://ch.mathworks.com/help/stats/bayesian-optimization-algorithm.html

[45] P. D. Kovesi, "MATLAB and Octave functions for computer vision and image processing." [Online]. Available: http://www.peterkovesi.com/matlabfns/

[46] MATLAB. alphashape. [Online]. Available: https://ch.mathworks.com/help/matlab/ref/alphashape.html

[47] ——. Acquisition function types. [Online]. Available: https://ch.mathworks.com/help/stats/bayesian-optimization-algorithm.html#bvaz8tr-1

[48] Stanford. Cs231n convolutional neural networks for visual recognition. [Online]. Available: http://cs231n.github.io/convolutional-networks