An effective approach to 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. Firstly, a preprocessing step aiming to spot plausible defective areas is discussed. Lastly, the classification of these proposed regions is made. Hence, a proper pixel-wide segmentation scheme is described.**

***Index Terms*—Computer Vision, Image Processing, Defect Detection.**

I. Introduction

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

There is a large variety of previous work on defect detection on steel surfaces, even though many ideas [1–3] are as interesting as they are poorly tested and very dependent on assumptions about the uniformity of the background surface. When different motifs of steel surfaces are analyzed, this approaches easily fail. Moreover, simple thresholding tecniques poorly perform on more complex defect shapes and different light exposures. There are solution using deep learning architectures [4, 5], although with ridicolously small datasets, which provide an attempt to a robust defect segmentation. Other more refined approaches rely on wavelets to detect abrupt changes in the surfaces, and this kind of previous work [6–9] has been widely used in the textile industry. The core of this paper is based on wavelet analysis and on deep learning, due to two main reasons.

Firstly, multi-resolution analysis (MRA) based on wavelets have been proven effective in facing localization both in spatial and in frequency domains [10–12]. This because of their mathematical properties, compared to Fourier's transform.

Secondly, deep learning [13, 14] has outperformed in the last years any human-designed classificator, indeed computer vision and image processing are increasing in popularity and they are being used ever more in many fields, from autonomous driving vehicles to retail and retail security. Hence, there has been an appreciable improvement in the effectiveness of defect detection based on visual systems and a lot of work has been done since the rise of deep learning applications [15]. Three main computer vision tasks have been 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), from handwritten characters [16, 17] and house numbers [18] to 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 intrinsic difficulties of designing tailored features extractors. Morover, they are also more likely to be shift and scale invariant [20], and they have been proved to have enviable classification accuracies.

A classification task in defect detection field is accounted when objects, e.g. steel surfaces, need to be binary classified as defective or flawless. When visual systems are considered and pictures are taken to classify a particular object, this would be negligently in practical applications. Indeed, monitoring locally the product concerned would be overly expensive, whereas a single global visual system is patently 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.

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. In computer vision, in particular in 2D images, the position is described by a bounding box.

CNNs have been used along with sliding window and multiscale approaches for object detection [21–23], and there is a lot of work aiming to improve performances and bounding boxes accuracies, both by designing different neural network architectures [21] or by tailoring existing one [19]. Regarding to scale-dependence in the object detection task, a solution is given by either brute-force learning (and CNN oversizing) or image pyramids [21], whereas the bounding boxes accuracies can be optimized by combining different scale sliding windows results, taking into account activation confidences in a particular area of the image and applying thresholding tecniques.

In this paper a further refined system is presented, since the purpose of the defect detection algorithm is not only to

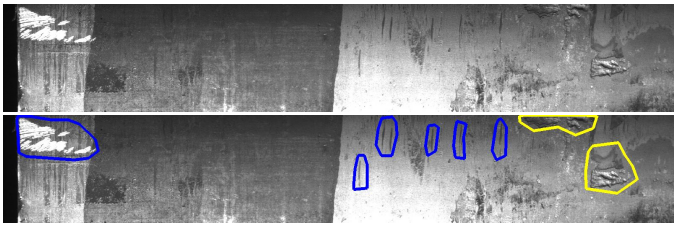

Fig. 1. Detection process input and output.

globally mark an image as picturing a flawless or defective steel surface, but both to highlight flawed regions inside the image and to label it as belonging to a particular defect class.

Pixel-wide classification is known in literature as image segmentation task, and there are three main families of tecniques: hysteresis thresholding, edge-based and region-based [24]. Thresholding exploits a previously known function from the pixels space and classifies pixels through comparison with some discrete values (thresholds) [25], but it is tipically used within other tecniques rather then alone. Region-based approaches use graph algorithms [26, 27] or watersheds analogies [28]. Edge-based tecniques, instead, use an edge detection filter [29–31], along with denoising and thresholding considerations, to solve the boundary detection 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 subdomain of a border. There are also more advanced tecniques [32] boundary-related 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 [24] or supervised [33].

The approach here described 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 here, to reach high speed segmentation. Wavelet are used along with image preprocessing and alpha-shape [34] to identify proposals, i.e. regions of interest for the classificator, which may contain a defective area. To overcome the bias introduced from hand-crafting the edge-detection filter, the hyperparameters of the algorithm are tuned with Bayesian Optimization [35–37]. A multi-column CNN (MC-CNN) [17] is then used to combine the segmentation information with a well-known classificator architecture, exploiting both local information and global information. The proposed architecture is shown to have optimal performances on the *Severstal: Steel Defect Detection* Kaggle competition dataset.

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 II-B.

For each of this classes, a thorough characterization of the pixels in the defective areas is given.

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 4 6"
```

An optimized implementation in `MATLAB` of both the rle encoding and decoding scheme described is proposed in [38].

A visual description of the end to end process is given in figure 1, where defective areas have been highlighted with different colors, depending on the defect class.

A mathematical description of the task is:

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 they are described in III.

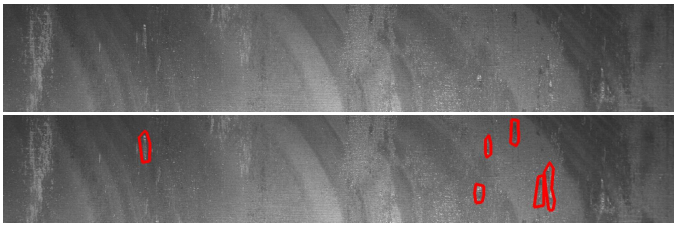

Fig. 2. Steel surface with defect class #1.

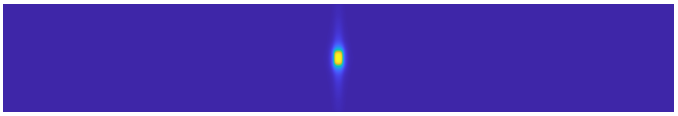

Fig. 3. Shape distribution for defect class #1.

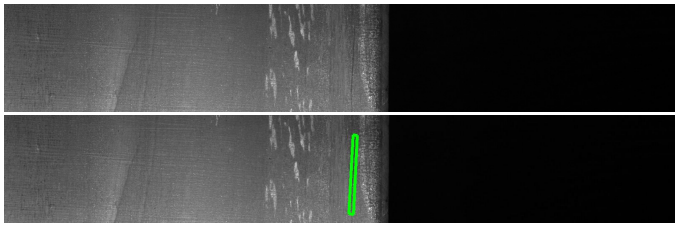

Fig. 5. Steel surface with defect class #2.

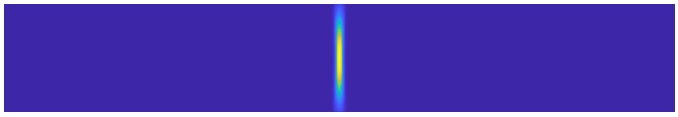

Fig. 6. Shape distribution for defect class #2.

B. Defect analysis

The dataset is concerned with flat steel sheet, which production process is especially delicate and structured in many phases.

Therefore, there are numerous defects classified in literature [39, 40]. Hence, many of the traditional types may have been grouped together in the four class given. However, in this section a plausible explanation of each defect class is provided.

The fundamental observation, however, is that some defects have a global origin, i.e. they are due to a flawed machinery, therefore is reasonable that a local classifier would miss important details.

One of the main stage of the production process is rolling [41, 42], which is the procedure of plastically deforming steel by passing it between rolls. Therefore, the stell is subjected to high compressive stresses as a result of the friction between the rolls and the metal surface.

The semi-finished products of rolling are named *boom*, *billet* and *slab*. The former is the product of the first breakdown of the ingot, the second is obtained from a further reduction by hot rolling, and the latter is the hot rolled ingot.

The billet is characterized by a cross sectional area larger then $40 \times 40\,\mathrm{mm}^2$, whereas the slab has one larger then $100\,\mathrm{mm}^2$ and a width two times greater then its thickness.

Rolling mill products are called *plate*, *sheet*, *strip* based on their size.

The plate has a thickness greater then $6\,\mathrm{mm}$, whereas both sheet and strip are smaller. However, the latters are distinguished by their width. Sheets width is larger then $600\,\mathrm{mm}$, whereas strips width is not.

1) Defect class #1: The first type of defect has not been classified yet into one of the classes found in literature.

However, it is a glaring example of burst defects, indeed it is reasonable to expect such defects to repeat multiple times on the same surface.

Fig. 4. Shape size bivariate lognormal distribution for defect class #1

An example of surface with a burst of class #1 defects is visible in illustration 2. The shape distribution of the defect is illustrated in picture 3.

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.

The bivariate lognormal distribution of the shape size is shown in 4.

2) Defect class #2: Defects of class #2 usually appears near the transversal edge, hence, they are probably edge laminations, since they are also visually similar.

This defects are due to the overcooling of the slab off of the caster. The coil mill edges looks like a continuous or semi-continuous line of slivers.

In figure 5 an example of surface with a defect of this type is shown. The shape distribution of the defect is illustrated in picture 6. The bivariate lognormal distribution of the shape size is shown in 7.

3) Defect class #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 #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 patent example of burst defect, indeed it is reasonable to expect such defects to repeat multiple times on the same surface.

Fig. 7. Shape size bivariate lognormal distribution for defect class #2

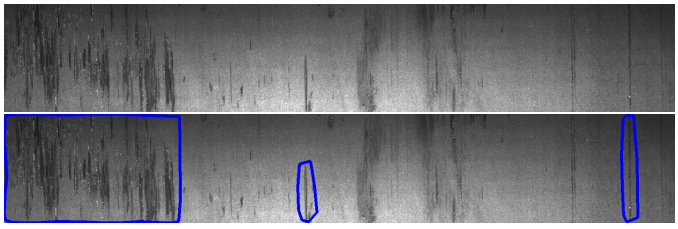

Fig. 8. Steel surface with defect class #3.

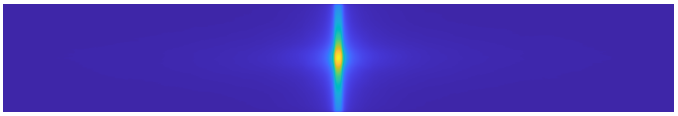

Fig. 9. Shape distribution for defect class #3.

Fig. 10. Shape size bivariate lognormal distribution for defect class #3

An example of surface affected by defects of class #3 is shown in figure 8. A single defect of class #3 (i.e. a single crack) has usually a shape as the one shown in picture 9. The bivariate lognormal distribution of the shape size is shown in 10.

4) Defect class #4: This defect class seems to be concerned with protrusions on the steel surface. The main typologies of protuberances in the 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 constrained 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 #4 is shown in figure 11. A single defect of class #4 (i.e. a single crack) has usually a shape as the one shown in picture 12. The bivariate lognormal distribution of the shape size is shown in 13. These statistics have been used, along with the visual inspection, to infer the origin of this defect class.

Differently from the defects of class #3, these flaws are local, i.e. it is not unusual to find only a single region with this issues in the whole picture.

C. Dataset considerations

In illustration 14 the relative representation of the classes in the original dataset is shown. It is patent that class 3 is far more represented then the other defect classes, and the number of

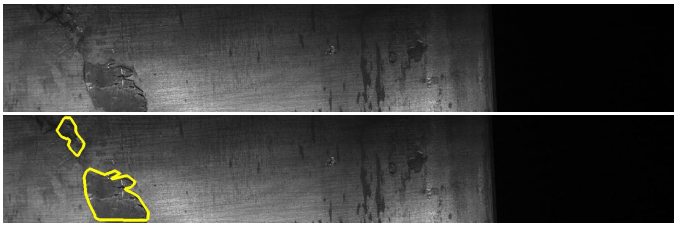

Fig. 11. Steel surface with defect class #4.

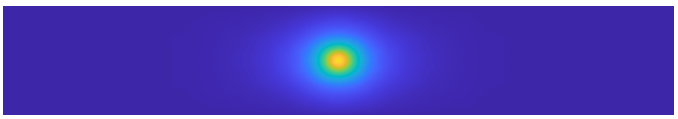

Fig. 12. Shape distribution for defect class #4.

Fig. 13. Shape size bivariate lognormal distribution for defect class #4

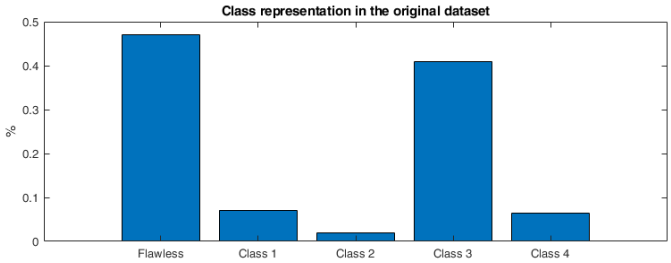

Fig. 14. Class representation in the original dataset.

surfaces with at least a defect of this class is nearly tantamount the number of flawless surfaces.

In figure 15 images have been 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 class examples, data augmentation is done only on those classes with fewer elements.

In order to keep proper proportions and spatial information, replicas of surfaces are built only using simmetries. Therefore, from a single image other three are created. An example of such operation is shown in 16.

The encoded pixels must of course be mapped onto the new image. This is 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 17 shows the new representation of the different classes after data augmentation.

III. Architecture overview

The defect detection system architecture proposed in this paper is shown in figure 18.

Steel surfaces pictures of 1600×256 pixels are taken at the input of the process. Since they may be taken one with a different light exposure condition then the others, some preprocessing is made to enhance the quality of the image, e.g. hystogram 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 IV.

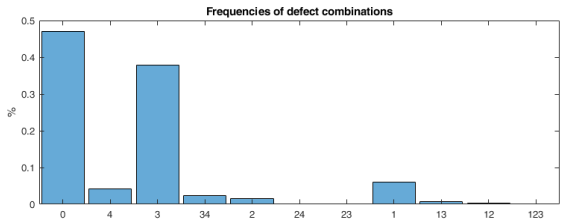

Fig. 15. Frequencies of defects combinations.

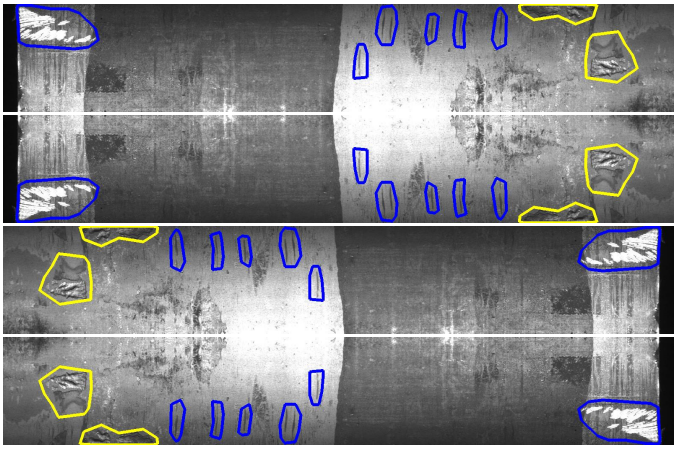

Fig. 16. Data augmentation. The image is flipped horizontally, vertically and both. The defective area is moved coherently.

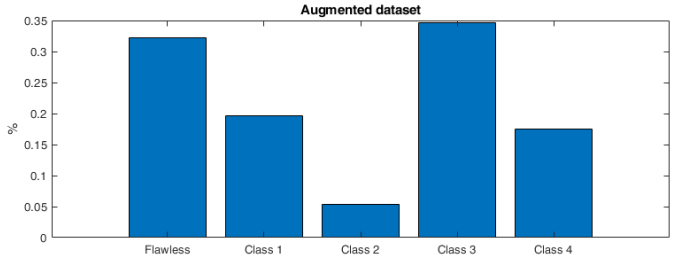

Fig. 17. Class representation in the augmented dataset.

The aim of 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. To achieve this, a brute-force multi-scale sliding window on the picture could be used, but to improve performances without reducing accuracy a particular implementation of a Region based CNN (R-CNN) [21, 22] is proposed in VI. This R-CNN uses a MC-CNN to combine and consider separately interesting regions, which are called proposals and which are described in V, to reduce the number of evaluations. 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 with relevant comparisons and results is provided in VI.

The column of the MC-CNN concerned with global information 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 V. Firstly, a contour detection algorithm (V-A) is used to spot proposals. Secondly, image segmentation (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

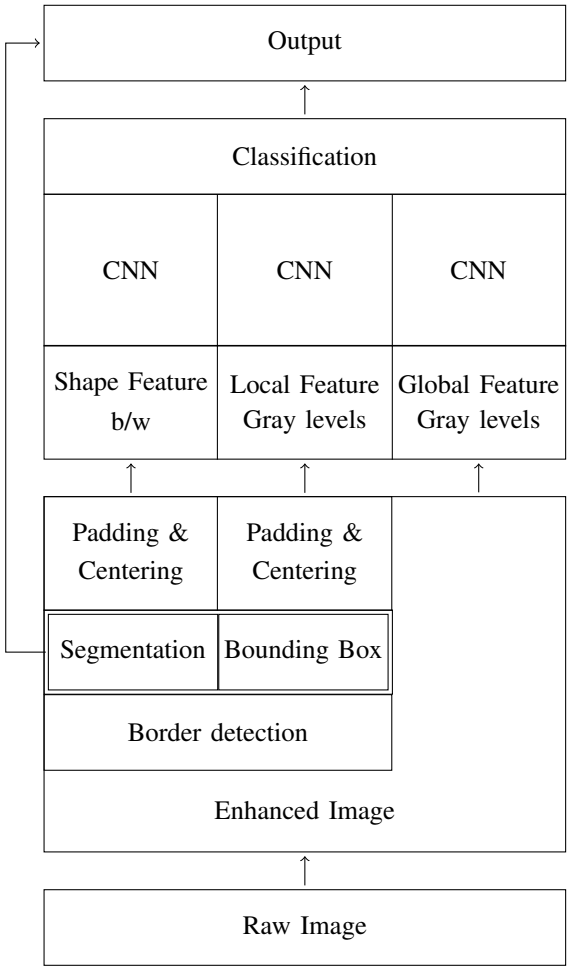

Fig. 18. Proposed defect detection system architecture

of the MC-CNN is fed with this map, therefore it learns to classify regions only observing their borders. The other column is fed with the portion of original image enveloped in the bounding box (V-C) of the map, therefore it is trained to consider luminance levels inside, outside and on the border of the considered proposal.

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

The two MC-CNN columns concerning local information ends with a soft-max layer, because they focus on a region which is proposed to represent a single class defect. Instead, the one dealing with global information has a different output layer. Their results are then combined in order to properly classify the local regions, and the approach is described in VI.

Finally, if the classification outcome labels the region as defective, segmentation coordinates are kept. When all the proposals of the considered image have been processed, defective pixels are encoded with RLE algorithm. The output given is the RLE-encoding of all defective pixels grouped by class. If the surface is flawless, all this encodings are empty.

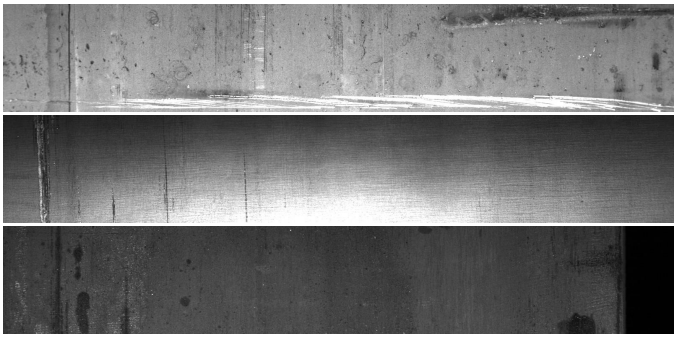

Fig. 19. Sample images before preprocessing.

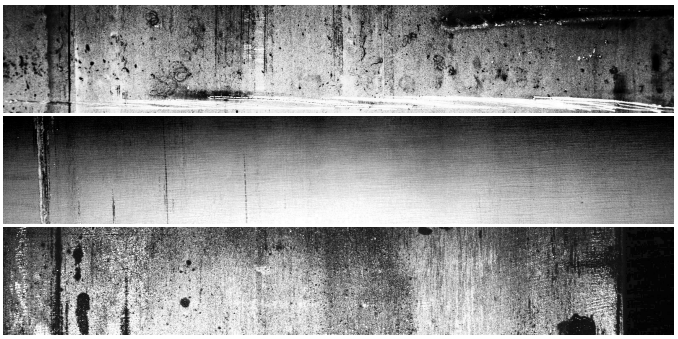

Fig. 20. Sample images after preprocessing.

IV. Image preprocessing

In this section image preprocessing is introduced, and the raw image is enhanced to improve learning quality. In VIII 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. Observe that 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 taken under different light exposure conditions, and since learning has heuristically be proven to be more effective if input assumptions are always the same, the luminance histogram of the image is normalized.

Linear scaling ensure that all images gray levels 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

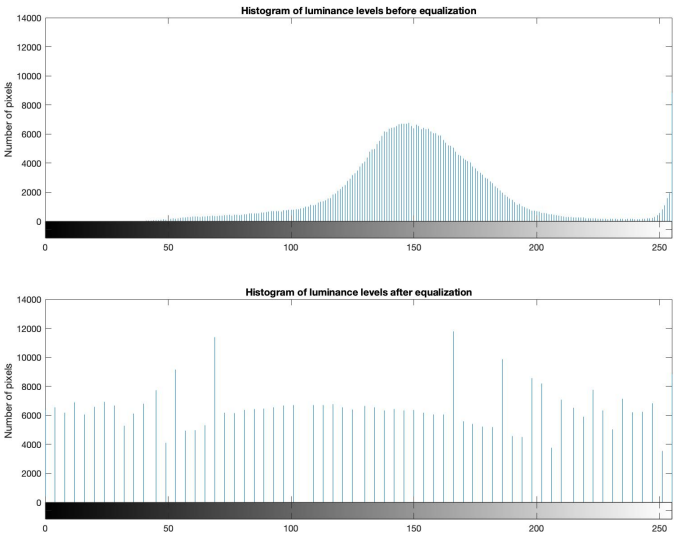

Fig. 21. Hystogram of luminance distribution on sample image before and after equalization.

G_{max} the greater luminance level (tipically $2^k - 1$ for some k), 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 ensure 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.

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 21 the effects of equalization a sample image hystogram of luminance distribution are shown.

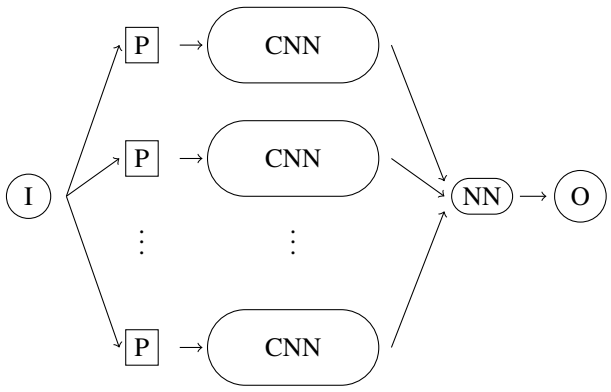

Fig. 22. General structure of a MC-CNN

Pictures 19 and 20 show the preprocessing output on some samples images. It is patently visible that the different light exposures of the three images are compensated through the histogram equalization.

V. Detector

** .... The first part of the architecture is a *detector* for region proposals.... Explain how to pick region of interests (ROI).... **

A. Contour detection

B. Image Segmentation

Introduction to alpha shapes and cite article describing proper segmentation usign alpha shapes.... describe parameteres and present limitations of such an approach in this practical application.... Therefore, bayesian optimization is proposed for alpha value.... Present table comparing different alpha values.... Explain evaluation scheme for alpha value optimization....

C. Bounding box

VI. Classificator

The ROIs are then fed into the second part of the proposed architecture, the *classificator*, and properly classified as flawless or flawed; in the latter, they are assigned a defect class.

The *classificator* is structured as a MC-CNN, which in general has a structure similar to the one in figure 22.

Firstly, the input image I is eventually preprocessed to extract n column input P.

Secondly, these P are fed into different CNNs in parallel, therefore independently.

Finally, the output of the MC-CNN columns is combined through a final classificator, e.g. a neural network (NN), to produce the appropriate 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 concerning 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 area. 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.

However, in IX 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 characterize the effectiveness.

Observe that the output of global column must be calculated only once per each image, since it is constant throughtout the single surface, hence 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. This considerations are reported in VIII.

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

A final consideration about the architecture training has to be done in order to comprehend the upper bound reported in VIII. Indeed, the *classificator* has been implemented before the *detector*. In fact, although the former relies on the latter for the ROIs and their relative features (described in VI-A, VI-B and VI-C), it has been preempting supposed to have an ideal *detector*, i.e. one which proposes the optimal ROIs.

This effort-outcome oriented approach has been done for two main reasons.

Firstly, this approach highlight the upper bound reachable with the whole architecture.

Secondly, dividing the *detector* outcome from the *classificator* input during the training allows to export the trained MC-CNN and to use it within the challenger architecture, explained in VII. This specifies the effectiveness of the *detector* proposed.

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 23. 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

Fig. 23. Shape column input.

Fig. 24. Shape column CNN layers

defect spanning over the entire surface. The shape is centered to ensure that the classificator is translation independent.

In order to provide also negative examples, i.e. flawless areas proposed by the *detector*, while training the *classificator* with ideal input, it is needed to generate those.

*** TODO DESCRIBE FAKE ELEMENTS GENERATION ***

The layers of this CNN are shown in 24. *** ... explanation, dimensioning, type of layers .... Stats on defects shapes.... ***

The output layer is a $n+1$ vector, where n is the number of defect classes. Each entry of this vector describe the confidence of the network in considering the input shape as flawless or related to one of the n defect classes.

B. Local column

The *local column* is thought to consider luminance levels around the border of the defect, to learn from the local context wheter a region proposal is defective or not, and eventually its class.

Therefore, the column is fed with the grey scale portion of the image in which the considered region is. As an example, in figure 25 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 VI-A.

The layers of this CNN are shown in 26. *** ... explanation .... Stats on defects shapes.... ***

The output layer is analogous to the one in VI-A.

C. Global column

The *global column* is concerned in detecting global patterns, like zipper cracks. Hence, it is fed with the whole image. An example of input is shown in figure 27. The importance of this column is clear from the observations made in II-B. Indeed, when two different classes are locally equal, the global column is determinant to correctly classify the defects.

Fig. 25. Local column input.

Fig. 26. Local column CNN layers

Fig. 27. Global column input.

Fig. 28. Global column CNN layers

*** structure, explanation, output ***

*** MAYBE OUTPUT AS MULTIPLE COMBINATIONS CLASSES? ***

*** DESCRIBE STRUCTURE, RATIONALE BEHIND DIMENSIONING, USE DEFECT STATS ***

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 combination would be ineffective.

In figure 29 the structure of the last layer of the architecture is shown.

The first layer is constrained to the output size of the three columns, i.e. *** TODO COMPLETARE... *** Then there is an hidden layers with 7 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.

Observe that bias units have not been included in any layer.

** DISPLAY WEIGHTS, infer what matters most to determine output **

The results of this architecture are shown in VIII.

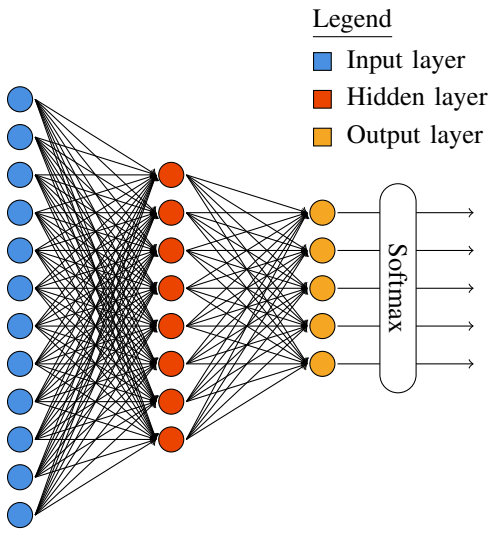

Fig. 29. Final classifier structure.

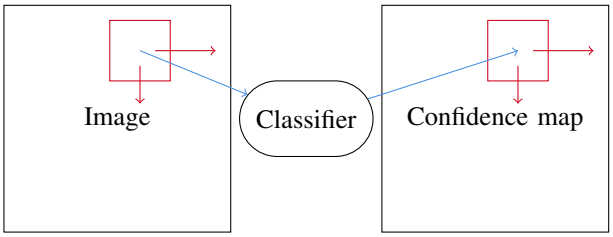

Fig. 30. Sliding window architecture

VII. Challenger

To measure the effectiveness of the R-CNN approach relying on the edge-based image segmentation involving wavelet analysis, the results are compared with another well-known architecture, i.e. a *sliding window* classificator. Results are reported in VIII.

A. Sliding window architecture

The idea behind this architecture is to crop the input image at different locations (eventually all the ones possible, as in this paper) 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 euristic to combine different values of confidences is needed. In this article ** TODO completare... **.

In figure 30 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 VII-B.

Since defects may have different dimensions, more refined tecniques could be used to improve the accuracy of the challenger. However, these are outside the scope of this work, and they are proposed in IX. Moreover, the segmentation tecnique proposed in VII-B soothes this problem, since it combines local information from different areas to build the defective regions.

B. 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 [28] considering all the defective classes as one, and distinguishing them only later. The $n+1$ levels whatershed algorithm is left as a further development in IX.

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.

Fig. 31. Shape column confusion matrix on ideal input.

Fig. 32. Local column confusion matrix on ideal input.

VIII. Results

Review the article, make some considerations on results

*** Confronto con esplicitamente indicati il contributo di ogni step al miglioramento del risultato ***

The whole system implementation can be found in the `GitHub` repository [38].

IX. Further work

... provide suggestions to further work... * use image pyramids in challenger * Eventually, it is possible to consider a multi-scale approach ... * Remove the assumption of non-overlapping regions for the $n+1$ levels whatershed 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.

[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.

Fig. 33. Global column confusion matrix on ideal input.

Fig. 34. Final classifier confusion matrix on ideal input.

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

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

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

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

[31] 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.

[32] 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

[33] 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.

[34] 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.

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

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

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

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

[39] 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

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

[41] Wikipedia. Rolling (metalworking) — wikipedia, the free encyclopedia. [Online]. Available: https://en.wikipedia.org/wiki/Rolling_(metalworking)

[42] (2017, November) Rolling process. [Online]. Available: https://www.slideshare.net/SivaKumar1226/rolling-process-82171623