Detection of corrosion on steel structures using an Artificial Neural Network

Mojtaba Khayatazad[1,2], Matthias Honhon[1], Wim De Waele[1].

[1] SOETE LABORATORY, Department of Electromechanical, Systems and Metal Engineering, Faculty of Engineering and Architecture, Ghent University, 9052, Zwijnaarde, Belgium

[2] SIM VZW, Technologiepark 48, 9052 Zwijnaarde, Belgium

emails: Mojtaba.Khayatazad@ugent.be Wim.DeWaele@ugent.be

Corresponding author: Mojtaba Khayatazad

Abstract

Image based-corrosion detection has become a widespread practice for steel structures, but fine-tuning their model parameters is time-consuming. Alternatively, convolutional neural networks (CNNs) can also be trained fast and automatically, but they demand a huge training dataset. In this paper, a corrosion detection approach based on an artificial neural network (ANN) whose training dataset size is less than 0.1% of that of typical CNNs is introduced. The input layer of the proposed ANN consists of textural and color properties. In the present work, different color spaces and textural properties are examined for their impact on the robustness of the ANN. Results reveal that the best color channels can be achieved by combining *CIE L*u*v** and *YUV* color spaces. Moreover, *energy* is selected as the best texture feature with respect to the ANN robustness. The proposed ANN outperforms an available image processing algorithm from the perspective of both speed and accuracy. In conclusion, this ANN can be used for actual applications after a fast and straightforward training step.

Keywords: Steel structures, Image-based corrosion detection, Artificial neural network, Deep hidden layers, Color space, Texture metric.

1. Introduction

NACE reports that the global annual cost of corrosion amounted approximately to US$2.5 trillion in 2016 (Koch, Varney, Thopson, Moghissi, Gould, & Payer, 2016). Corrosion is also reported to be responsible for 42% of engineering components' failures (Petrovic, 2016). In these circumstances, timely corrosion detection can mitigate further damage, and prevent heavy casualties or environmental pollutions.

Industry uses different methods for corrosion detection, including visual inspection (See, Drury, Speed, Williams, & Khalandi, 2017), electromechanical impedance method (Zhu, Luo, Ai, & Wang, 2016), ultrasonic inspection (Sharma & Mukherjee, 2015), thermography (Doshvarpassand, Wu, & Wang, 2019), eddy current technique (He, Tian, Zhang, Alamin, Simm, Jackson, 2012), radiography (McCrea, Chamberlain, & Navon, 2002) and acoustic emission technique (Cole & Watson, 2006).

Visual inspection is more popular than the others because it is contactless and allows easy interpretation. However, in some specific cases it is executed at a huge cost, both from economic and time terms. For instance, corrosion monitoring of a giant crude carrier with a typical steel area of 600,000 m^2 is time-consuming and demands scaffolding and/or presenting at hazardous zones (Ortiz, Bonnin-Pascual, Garcia-Fidalgo, & Company, 2016). Researchers proposed to use uncrewed aerial or underwater vehicles to solve this problem by capturing hundreds of images from infrastructures in a short time and at zero life risk conditions (Jahanshahi & Masri, 2013), (Khan, Ali, Anwer, Adil, & Meriaudeau, 2018).

To facilitate the procedure even more, image processing algorithms like the adaptive ellipse approach (Chen, Yang, & Chang, 2009), rust defect recognition method (Shen, Chen, & Chang,

2013), image restoration and enhancement algorithm (Khan et al., 2018) and weak classifier color based corrosion detector (Bonnin-Pascual & Ortiz, 2014) came into the picture to segment corrosion-like regions. However, fine-tuning their parameters is done by trial-and-error and thus time-consuming (Ahuja & Shukla, 2018).

On the other hand, machine learning algorithms can be trained fast and automatically for pattern recognition. Hence, they are considered excellent alternatives for image-based corrosion detection algorithms. Several researchers developed a convolutional neural network (CNN) as a deep learner for analyzing imagery (Cha, Choi, & Büyüköztürk, 2017). Successful implementation of a CNN for corrosion detection has for example been reported in (Cha, Choi, Suh, Mahmoudkhani, & Büyüköztürk, 2018), (Bastian, N, Ranjith, & Jiji, 2019). As a drawback, it should be mentioned that a CNN requires a relatively large training dataset. For instance, Bastian et al. (2019) mentioned that 140,000 optical images of pipelines with different corrosion levels have been used for training their CNN. Although converting video records to thousands of individual shots as a handy alternative alleviates the situation, they are not always available.

In these circumstances, an artificial neural network (ANN) is helpful since it requires less training data than a CNN. Moreover, an ANN can be applied at the pixel level; as such an image of m by n pixels provides $m \times n$ samples. Application of an ANN for corrosion detection dates back to 1955 when Furuta, Deguchi and Kushida used an ANN to determine threshold levels for the *HSV* (Hue-Saturation-Value) color space. Although it seems an attractive application, an image processing algorithm was used for the main corrosion detection. Livens et al. (1995) utilized a specific ANN case, i.e. a Learning Vector Quantization network, for corrosion detection. They classified corrosion based on textural properties only. Jahanshahi and Masri (2013) used color wavelet-based texture analysis and built an ANN for corrosion detection. They concluded that both color and textural properties should be used.

Color images contain more data than greyscale ones, and therefore these are used primarily for corrosion detection. Naik, Sajid, Kiran and Chen (2020) constructed a multi-layer perceptron working based on the color properties only. They evaluated four different color spaces and concluded that the 'rgb' color space outperforms the others. They also mentioned that the lack of textural properties would lead to some limitations. Ortiz et al. (2016) optimized an ANN with one hidden layer for coating breakdown/corrosion detection by considering both color and textural properties. Although achieving good results, they neglected to mention the textural property used in their paper.

As mentioned earlier, CNNs are powerful for analyzing visual imagery but require a vast training dataset. Having multiple hidden layers is the common feature of CNNs. To combine this deep learning feature with the pixel-based nature of ANNs, this paper evaluates the potential of building a robust ANN with different numbers of hidden layers for corrosion detection. The entire study is established using open source software (Python 3.7, OpenCV and Pytorch). The proposed ANNs are trained using a small dataset that is only a fraction of what convolutional neural networks, commonly used in image-based classification tasks, would require.

First, the pixel-level results of an ANN with one hidden layer are compared with those of an image processing-based algorithm described in (Khayatazad, De Pue & De Waele, 2020). This algorithm, presented in Figure 1, uses two weak classifiers based on color and texture features. It starts with reducing the smallest side of a given image to 256 pixels whilst conserving the aspect ratio. Then the roughness analysis is performed for each patch of the resized image. If the roughness of a patch is less than a threshold value, that patch - including all of its pixels - is considered non-corroded. If not, the color analysis is next performed on all pixels and pixels with a color belonging to the color spectrum of corrosion are eventually defined as corroded pixels and the rest as non-corroded. Obviously, several threshold parameters must be fine-tuned

for this algorithm and consequently this procedure is time-consuming. Moreover, the optimized parameter values are case dependent and cannot be used for general purposes. Another limitation is that roughness and color analyses can only be performed sequentially and the performance of the algorithm depends for a large amount on the accuracy of the first step. On the contrary, the proposed ANN using identical color and texture features, considers both color and texture simultaneously and can be trained fast and automatically for any new case.

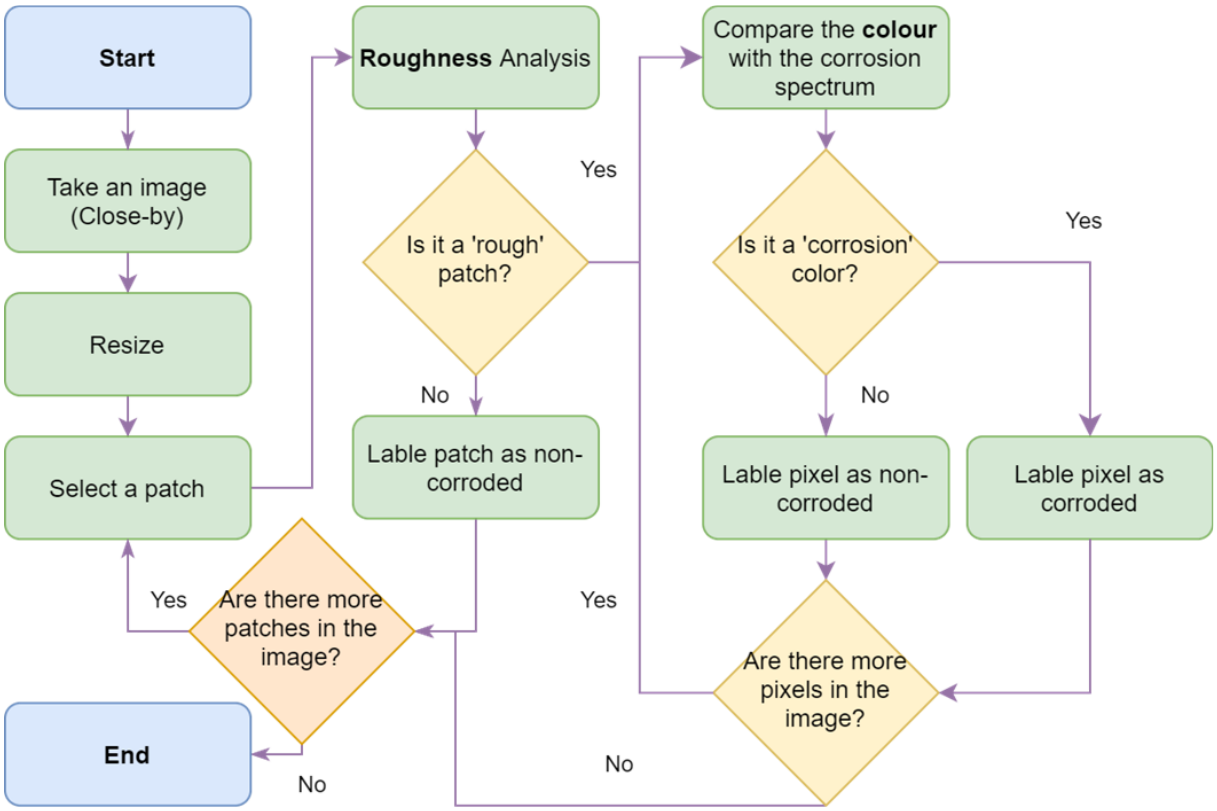


Figure 1: *Corrosion detection flowchart of the benchmark image processing algorithm (Khayatazad et al., 2020).*

Next, the ANN hyperparameters (number of neurons and hidden layers) are optimized. The features contrast, dissimilarity, homogeneity, energy and correlation are evaluated with respect to texture. Regarding color, seven color spaces are investigated. A grouping of two color spaces for the input feature is also investigated.

This paper is organized as follows. Section 2 describes the input features, color and texture. Next, the details of the applied artificial neural network are explained in section 3. Section 4 and 5 present step-by-step the evolution of the proposed ANN for the sake of robust corrosion detection. Eventually, the main conclusions of this paper are summarized in section 6.

2. Input Features of the Artificial Neural Network

In this paper, both color and textural features will be used as input to the proposed ANN. Details of these features are given in the following subsections. Hereunder some general notes are presented.

Before extracting relevant features from the image, the Python `cv2.resize()` function resizes images to smaller dimensions to ensure efficient processing time. The smallest dimension of the image is resized to 256 pixels while keeping its aspect ratio. Color features are pixel-specific, whereas textural properties are attributed to a patch of pixels. Therefore, the proposed ANN takes as input the pixel's color components and the texture feature of the patch to which the pixel belongs. A patch size of 15 by 15 pixels is used to accurately represent the textural features (Khayatazad et al., 2020). Sola and Sevilla (1997) recommended normalizing input features before introducing them to an ANN for faster calculations and better results. Since color and textural features have different scales and consequently a different variation range, they are normalized using:

$$x = \frac{x_{ini} - x_{min}}{x_{max} - x_{min}} \tag{1}$$

Where x_{ini}, x_{min} and x_{max} are the initial, minimum and maximum values of a specific input feature. It should be mentioned that these extreme values are determined in the training step of an ANN and are stored to be used for the test step.

2.1. Texture

When comparing the state of a piece of steel before and after corrosion, the surface texture of the corroded component appears rougher. Therefore this visual feature can be used for

corrosion detection if it is quantified. In the present work, so-called Haralick textural features are used for roughness quantification (Haralick, Shanmugam, & Dinstein, 2007). Since these textural features demand greyscale images with a predefined number of grey levels N_g, the `cv2.COLOR_BGR2GRAY` function is used to convert color images to greyscale ones. The grey level indicates the brightness of a pixel. For instance, a grey level of zero means that the pixel absorbs all the light, and therefore it appears in black. Opposite, a grey level of $N_g - 1$ means that the pixel is white and reflects all the light. Figure 2-a and b present a greyscale image and its corresponding grey matrix respectively, when the number of grey levels N_g is 8.

In the next step, the Grey-Level Co-occurrence Matrix (GLCM) is built using the `greycomatrix` function, Figure 2-c. The GLCM with a size equal to $N_g \times N_g$ determines the spatial relationship between greyscale values in an image. The matrix component $P(i,j)$ indicates how often a pixel with a grey value i has a specific spatial relationship with a pixel with a grey value j, Figure 2-d. This spatial relationship is determined by distance and angle, Figure 2-e. The distance is expressed as the number of pixels and its minimum value is 1 (for adjacent pixels). The angle can be either 0°, 45°, 90° or 135°. After the initial calculation, the values $P(i,j)$ of the GLCM can be normalized to obtain $\sum P(i,j) = 1$.

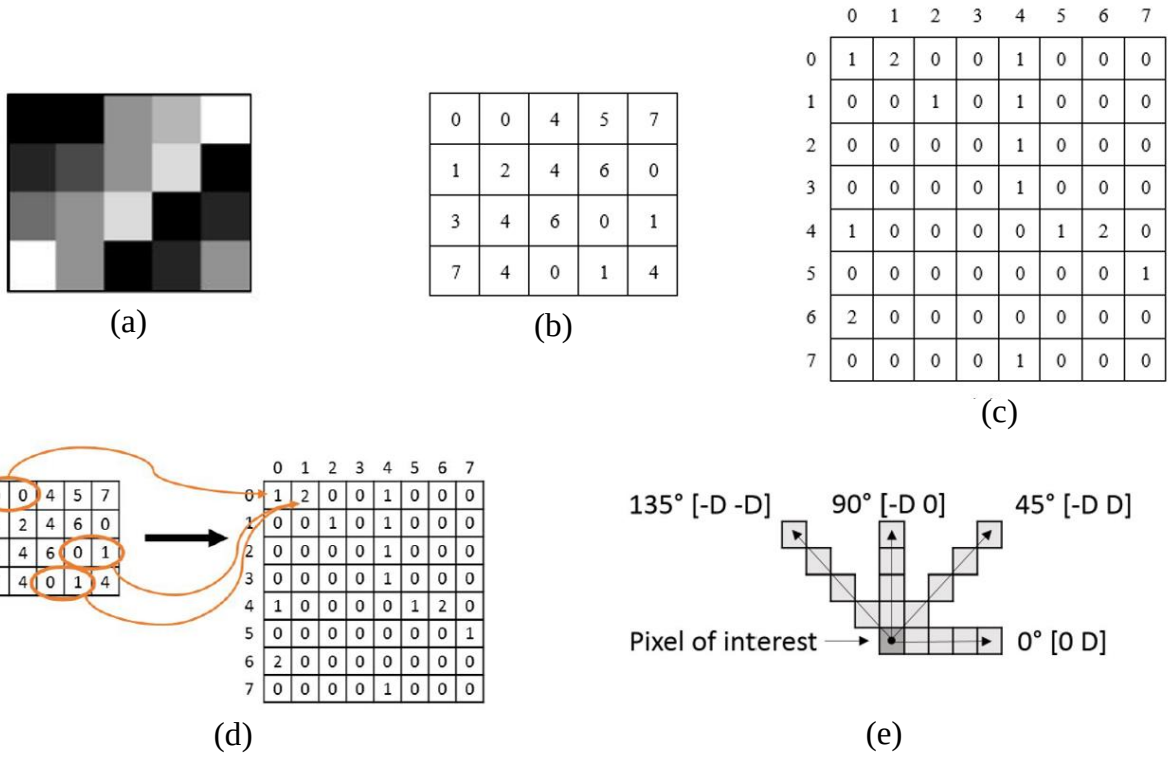


Figure 2: A greyscale patch (a), its corresponding grey levels (b) and grey level co-occurrence matrix (c). (d) illustrates how the GLCM is constructed. (e) defines distance and direction around a pixel.

Haralick extracted some features from the GLCM and argued that they can be linked to textural characteristics (Haralick et al., 2007). For instance, one way to evaluate the grey level distribution of a patch of pixels, is to calculate its *uniformity*:

$$Uniformity = \sum_{i,j=0}^{N_g-1} P(i,j)^2 \quad (2)$$

An utterly uniform patch leads to a *uniformity* of 1 while the rest gets a value between 0 and 1. *Contrast* is another feature measuring the intensity contrast between a pixel and its neighbor over the whole patch:

$$Contrast = \sum_{i,j=0}^{N_g-1} (i-j)^2 P(i,j) \quad (3)$$

Utterly uniform patches get a *contrast* of zero. *Homogeneity* measures the similarity of pixels in a patch:

$$Homogeneity = \sum_{i,j=0}^{N_g-1} \frac{P(i,j)}{1+(i-j)^2} \tag{4}$$

Homogeneity approaches one, its maximum value, when the local changes are minimum in a patch. *Correlation* is a measure for the grey-tone linear-dependencies in a patch:

$$Correlation = \sum_{i,j=0}^{N_g-1} \frac{(i-\mu_i)(j-\mu_j)}{\sqrt{\sigma_i^2 \sigma_j^2}} P(i,j) \tag{5}$$

where μ and σ represent the mean and standard deviation, respectively. *Dissimilarity* is akin to *contrast* with measuring local variations, but has a linear dependency on off-diagonal entries of the GLCM:

$$Dissimilarity = \sum_{i,j=0}^{N_g-1} |i-j| P(i,j) \tag{6}$$

In the current work, Equation (2) to Equation (6) are evaluated as input features of the proposed ANN.

As mentioned, textural properties are attributed to a patch of pixels, so the first calculation step is to form patches. The implemented algorithm for this purpose is presented as follows. After downsampling a given image so that its smallest dimension is 256 pixels, a patch with a size of 15×15 pixels is assigned to the top left corner of the image. The next patch is located exactly at the right side of the previous one. In other words, pixels from the 16th to the 30th columns and the 1st to the 15th rows form the second patch. This approach is repeated for the next patches. In case the image width is not a multiplication of 15, a final patch of size 15×15 pixels is formed at the top right of the image, although it overlaps with the previously formed patch. The second row of patches is formed in an identical way, now starting from the 16th row of pixels. That procedure is continued for the entire image. If the image height is not a multiplication of 15 pixels, the last row of patches is situated at the bottom of the image and overlaps with the previous row of patches. After patch forming, the GLCM matrix of each patch

is composed, and the texture feature is calculated. It is assumed that all pixels in a patch inherit the texture feature of the patch. Some pixels near the right and or bottom sides of the image might be attributed to more than one patch. The texture feature of the last attributed patch is then assigned to such pixel.

2.2. Color

While texture analysis techniques do not use the color information present in images, color can still be considered beneficial for corrosion detection. Although corrosion encompasses a characteristic color range, an immediate challenge for color detection is that it heavily depends on the illumination of the image. When looking at a structure that is partially in the shade, Figure 3, it is clear that shadow causes the corrosion colors to look darker than they actually are, while direct sunlight makes them look brighter. Chen et al. (2019) mentioned that choosing an adequate color space, i.e. having a channel that describes the illumination, could help alleviate this problem. In this work, seven different color spaces (six of them having a channel for illumination) are studied. The seventh color space, *BGR*, is included in this study because it is the default color space of OpenCV. The different color spaces are briefly described hereunder.

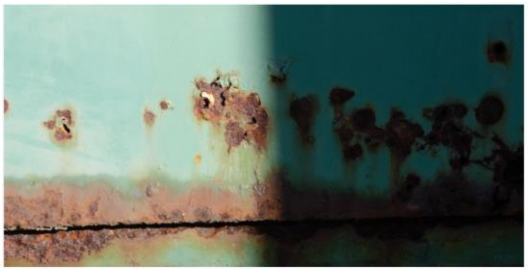


Figure 3: Different levels of illumination, a challenge for corrosion detection.

Color spaces *HSL* (hue, saturation, lightness) and *HSV* (hue, saturation, value) have been developed to simulate how humans perceive color. In contrast to *BGR* (blue, green, red), where blue, green and red coordinates can represent the full spectrum in a cube, both *HSL* and *HSV*

values can be represented by coordinates in a cylinder, Figure 4. The Hue is the primary channel and takes a value between 0 and 360, representing the angle of rotation in the base plane of the cylinder. Saturation varies between 0 and 1 and indicates how intense the color is, with a low value for Saturation corresponding to grey color. Lastly, Value and Lightness indicate the relative lightness of the color (Brewer, 1999). While they both vary between 0 and 1, Lightness goes from black to white and has fully saturated colors at half the height of the outer ring of the cylinder, while Value goes from black to fully saturated colors at the outer ring.

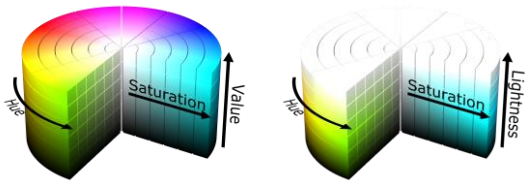


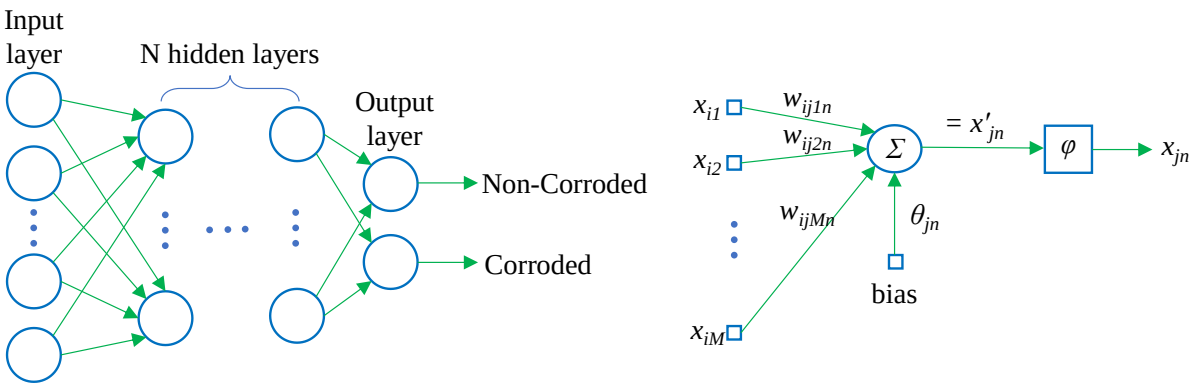
Figure 4: HSV and HSL color spaces.

Both *CIE L*a*b** and *CIE L*u*v** are *CIE* uniform color spaces and have been derived from the *CIE XYZ* color space. While *CIE L*u*v** (lightness, first chromaticity coordinate, second chromaticity coordinate) is used to characterize color displays, *CIE L*a*b** (lightness, green/magenta chromatic axis, blue/yellow chromatic axis) is used for dyes and colored surfaces to emulate human vision. Both color spaces try to represent color so that a proportional change in visual importance occurs when a change in color values occurs. In these color spaces, *L** indicates perceptual lightness (Alessi, Carter, & Fairchild, 2004).

YUV (luminance, bandwidth, chrominance) is a color space that was used in analog televisions. The *Y* channel is the luminance and contains the greyscale information, while the *U* and *V* channels are added to contain the color information. In the same way, *YUV* is used for analog encoding, *YCbCr* (luminance, chroma: blue, chroma: red) is used for digital encoding (Podpora, Korbaś, & Kawala-Janik, 2014).

3. Details of the Proposed Artificial Neural Network

An ANN comprises different layers, clustering a certain amount of artificial neurons, Figure 5-a. Neurons of subsequent layers are connected with each other by the use of weighted links, as shown in Figure 5-b. In a feed-forward neural network, the current layer only influences the next layer.



(a) (b)

Figure 5: (a) Typical architecture of a feed-forward ANN, (b) an artificial neuron.

In this paper, the input layer contains the textural and color properties of the pixels of the inspection photographs. Since the goal is to detect corrosion regions in an image, the output layer classifies the individual pixels as either non-corroded or corroded.

Referring to Figure 5-b, the value x_{jn} of the n^{th} neuron located in the j^{th} layer, is determined as follows:

$$x'_{jn} = \sum_{m=1}^{M} w_{ijmn} x_{im} + \theta_{jn} \tag{7}$$

$$x_{jn} = \varphi(x'_{jn}) \tag{8}$$

Where x_{im} is the value of the m^{th} neuron situated in the i^{th} layer with $m = 1, \dots, M$ neurons, w_{ijmn} is the inter-layer weight, θ_{jn} is the bias and φ is the activation function that is applied on x'_{jn}. In this paper, the activation function for the hidden layers is the rectified linear unit $(ReLU)$:

$$x = ReLU(x') = \max(0, x') \tag{9}$$

This function allows a fast convergence while it is not computationally expensive (Szandała, 2021). The `softmax` function, $Softmax(x_n) = e^{x_n}/(e^{x_1} + e^{x_2})$ with n = 1,2, is applied to both neurons of the output layer. The first and the second components of the output layer correspond to non-corroded and corroded probabilities that indicate how confident the network is that a pixel belongs to a class. A pixel can be classified based on the largest probability of the two. For example, values [0.19; 0.81] would result in the categorization of the pixel as corroded.

When an ANN is first initialized, all inter-layer weights and biases get random values resulting in a non-sensical output that necessitates training of the ANN. An overview of the training step can be seen in Figure 6. This step takes a pixel as input together with its corresponding mask. A mask is a representation of the image, with "1" (or black) representing a non-corroded pixel and "2" (or white) representing a corroded pixel. This mask serves as the ground truth of what is considered corrosion or not and can be manually obtained using any photo editing software. Color and texture properties are extracted for each pixel. Then they are put in an array together with the value provided by the mask. How wrong the network is, or how much the predicted output differs from the desired one, can be determined using a loss function. Selecting the appropriate loss function depends on the neural network's task, i.e., classification or regression. Since the goal of the proposed ANN is to predict if a pixel is corroded or not, which is a classification task, the appropriate loss function is `CrossEntropyLoss`. This loss function minimizes the distance between the predicted probability and actual distributions (PyTorch, 2020):

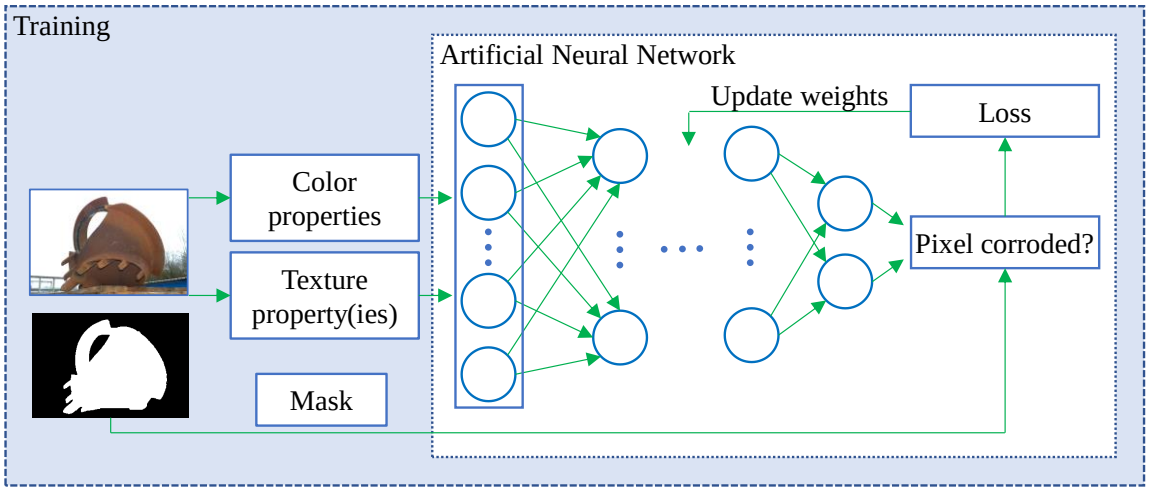


Figure 6: The training step of the proposed ANN.

$$loss(\boldsymbol{x}_{output}, mask) = W[mask]\left(-\boldsymbol{x}_{output}[mask] + log\left(\sum_{k=1}^{2} e^{\boldsymbol{x}_{output}[k]}\right)\right) \quad (10)$$

It is recommended to have a balanced training set, where each class occurs the same number of times. Since this is not practically possible in this work, a class weight W argument is used as mentioned in Equation (10).

The main part of the training step is to update the network parameters (biases and interlayer weights) so that the final loss becomes minimum. The most popular approach for parameter updating is called mini-batch gradient descent, in which subsets of the whole dataset are selected randomly and fed to the network one by one (Dey, Borra, Ashour, & Shi, 2019). Then the average loss function, $loss_{avg}$, is calculated for all members of each subset:

$$loss_{avg} = \sum_{k=1}^{N} loss(i, class[i]) \Big/ \sum_{k=1}^{N} weight[class[i]] \quad (11)$$

Where N is the number of samples in each subset. The next step is to select an optimization method for parameter updating. In this work, the momentum method is used to efficiently deal with the mini-batch gradient descent (Sutskever, Martens, Dahl, & Hinton, 2013). The momentum method updates the network's parameters using:

$$v_{t+1} = \mu v_t + \nabla_t \quad (12)$$

$$p_{t+1} = p_t - l_r v_{t+1} \tag{13}$$

Where v is called velocity, t denotes the t^{th} iteration, p is the inter-layer weight or bias, μ is the momentum factor that is equal to 0.9 (Dey et al., 2019), ∇ is the gradient with respect to parameter p and l_r is the learning rate. Velocity is zero in the first iteration. The initial value of l_r is set to 0.05 and by proceeding the training step, the `StepLR` function decays the learning rate l_r in a step-wise fashion by a factor of 0.1 after 10 epochs have passed. All data that are used for training an ANN comprise a package called an epoch. This epoch should be passed through the ANN several times in order to train it. How many times is enough to avoid over or under fitting depends on the diversity of the problem. This value proved to be 100 epochs for this work due to the combination of mini-batch gradient descent, momentum, and a decaying learning rate.

After training, a test dataset and its corresponding masks are introduced to the ANN. The ANN predicts the corroded spots in the test dataset. Then a post-processing step is performed to compare the result of the ANN with that of a mask. Eventually, an indicator is used to evaluate the performance of the ANN. In this work, the *F_1-score* is used as the performance indicator; it is calculated as follows:

$$F_1 - score = \frac{2TP}{2TP + FP + FN} \tag{14}$$

Where TP, FP and FN are true positive, false positive and false negative, respectively.

4. Evaluation of the Proposed ANNs

4.1. ANN With One Hidden Layer

This section investigates whether an ANN merely has a benefit because of the automatic training or if it outperforms image processing algorithms for corrosion detection as well. For this purpose, an ANN with only one hidden layer is compared with a previously reported image processing algorithm (Khayatazad et al., 2020). This algorithm uses two weak classifiers based on color and texture features (see Table 1) for corrosion detection.

It should be mentioned that the ANN proposed in this subsection is also based on the parameters mentioned in Table 1. The parameters introduced in Table 1 for composing the GLCM matrix have been shown to ensure the best result from both computational cost and acceptable accuracy points of view when the image's smallest dimension is 256 pixels (Khayatazad et al., 2020).

Table 1: Identical conditions used for both the proposed ANN and the benchmark image processing algorithm.

Color features			
HSV color channels			
Texture feature			
Type: *Uniformity*	Grey levels: 32	Distance: 7px	Angle: 0°

Though the image processing algorithm parameters are already fine-tuned, the ANN should be trained. The selected initial architecture of the ANN has one hidden layer consisting of 2^4=16 neurons. The input layer contains 4 neurons, 3 of which are related to color features (Hue, Saturation and Value) and 1 is related to the texture feature *uniformity*. The network classifies a pixel as non-corroded or corroded in the output layer consisting of 2 neurons. Therefore this architecture is called the *4[HSV+uniformity]-2^4-2* architecture. The training dataset consisted of 42 images containing varying degrees of corrosion, see Appendix A. While this might seem like a meager amount of data, it must be noted that the training occurs on the pixel level, not on the image or patch level, which means that 42 images, after resizing, lead to 3,921,920 training pixels. The training dataset consists of miscellaneous images, covering:

- different corrosion colors spanning from red to brown,
- different corrosion textures from low to high roughness,
- different backgrounds from the metal itself to the ambient environment including water channels, trees, sky, etc.,

- gradual and sudden change in the illumination level,
- and different misleading objects like rust stain, handwriting in corrosion color on the metal surface.

To compare the size of the required training database for the proposed ANN with that of a typical CNN, a conservative estimate is that the ANN needs 100 images instead of 42 for the training step. On the other hand, assume that a typical CNN optimistically requires 100,000 images instead of 140,000 images mentioned in (Bastian et al., 2019) for the same purpose. A simple calculation shows that the proposed ANN only needs 0.1% of the CNN's required training database.

After training the ANN, both methods are used to detect corrosion in not only simple images but also in images having non-uniform illumination and misleading colors. For this purpose, three different datasets are considered. The first one, dataset A, has 14 images in which the illumination is uniform and there is no misleading object, e.g. Figure7(a). Dataset B has 8 images with non-uniform illumination, e.g. Figure 7(b). Finally, there are 8 images in dataset C having misleading colors, e.g. Figure 7(c). These 30 new images consisting of 2,762,752 pixels cover different features already described for the training dataset. From a pixel point of view, the training and test datasets own 58.7% and 41.3% of the whole used dataset. Images of the test dataset are presented in Appendix A.

Figure 7 presents the post-processing of the original images a-c, by the image processing algorithm (Khayatazad et al., 2020) (d-f) and the ANN (g-i). As can be visually observed, the ANN offers better results than the image processing algorithm for all images.

Table 2: Different test image datasets.

Type	Number	Explanation
Dataset A	14	Uniform illumination and no misleading colors, e.g. Figure 7-a.

Dataset B	8	Non-uniform illumination, e.g. Figure 7-b.
Dataset C	8	Misleading colors, e.g. Figure 7-c.

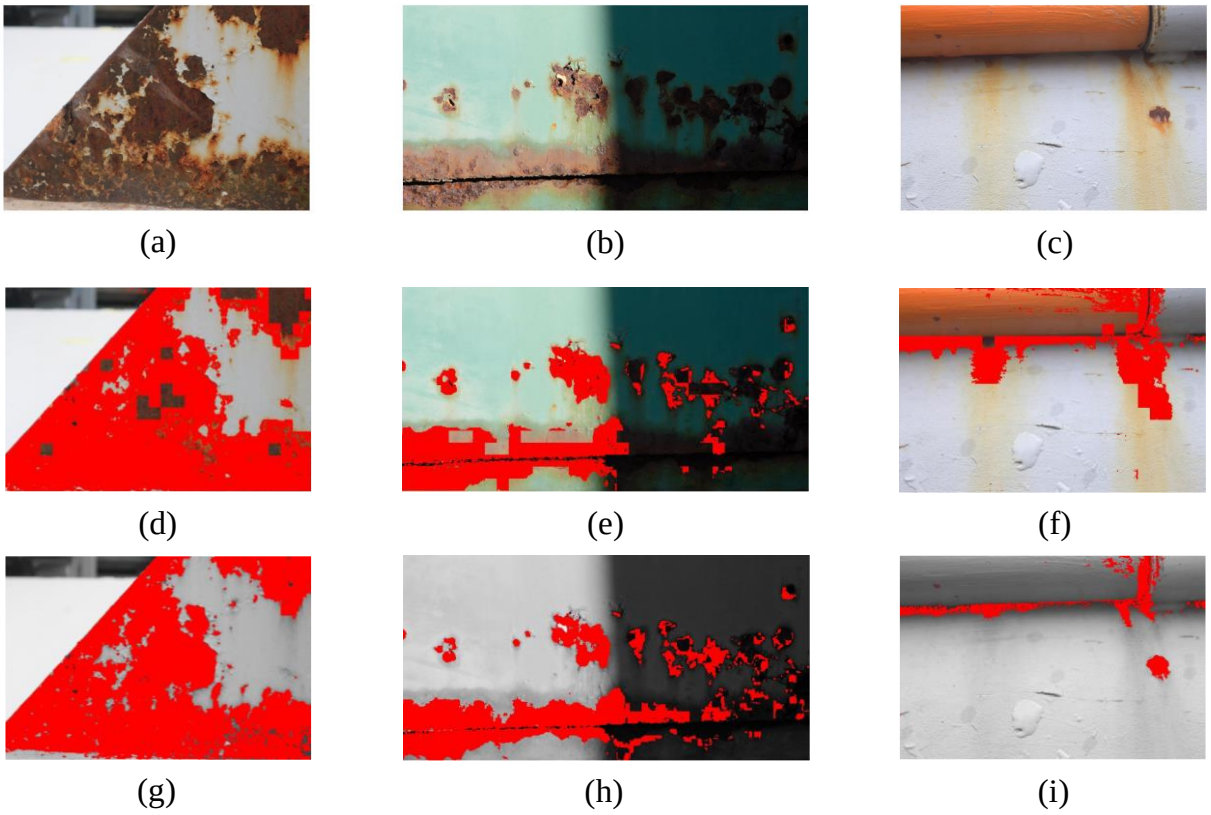


Figure 7: Images from different datasets: (a) uniform illumination and no misleading colors; (b) non-uniform illumination; (c) misleading colors. The red regions on images (d) to (f) and on images (g) to (i) show the corroded regions detected by the image processing algorithm and by the ANN with one hidden layer respectively.

In the remainder of this paper, the performance of the ANNs and the image processing technique shall refer to the overall performance of each method, i.e. for the entire dataset, unless stated otherwise. Table 3 reports the *F1-score* and processing time of both algorithms. It can be seen that the performance of the *4[HSV+uniformity]-2^4-2* architecture surpasses that of the image processing algorithm while using the same input parameters. The most significant improvement can be seen for images in the presence of misleading colors (dataset C), but a considerably better result is also obtained in the presence of uneven illumination (dataset B).

When comparing processing time per image, the ANN can analyze images 32% faster than the image processing algorithm. Both observations indicate that the use of an ANN can indeed form an improvement for corrosion detection. Figure 8 shows the loss function curve as a function of training epochs. The other ANNs that have been trained in this study showed similar loss function curves.

Table 3: Comparing the F1-score and processing time of the 4[HSV+uniformity]-2^4-2 ANN architecture and the benchmark image processing algorithm (Khayatazad et al., 2020).

Images of	Image processing algorithm	*4[HSV+uniformity]-2^4-2*	
	Absolute value	Absolute value	Relative value[1]
All datasets	0.6462	0.7080	+9.6%
Dataset A	0.7739	0.8110	+4.8%
Dataset B	0.6495	0.7037	+8.3%
Dataset C	0.4196	0.5322	+26.8%
Time/image [s]	0.33	0.25	-32.0%

[1] *Relative value is calculated based on the corresponding absolute value of the image processing algorithm.*

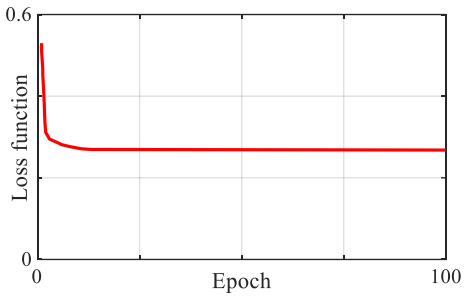


Figure 8: *Loss function curve of ANN up to 100 training epochs.*

The reported time in Table 3 is the average required processing time for each image. This time accounts for reading and resizing an image, extracting the input features from it and eventually the binary classification of its pixels. A personal computer with the specifications mentioned in Table 4 has been used for running both algorithms.

Table 4: PC specification.

O.S.	Microsoft Windows 10 Home
Processor	AMD Ryzen 7 2700X 8 cores @ 3.70 GHz base speed
GPU	NVIDIA RTX 2060 Cores: 384 Clock speed: 1365 MHz GPU RAM: 6GB
RAM	16 GB @ 2999 MHz

4.2. ANN With More Than One Hidden Layer

In general, an ANN with one hidden layer performs better in memorizing the training data, while an ANN with two or more hidden layers can learn the hidden characteristics of the data and is, therefore, better in generalization. Hence, this sub-section evaluates the effect of the number of hidden layers on the performance of an ANN. Moreover, the influence of different input features is also discussed to further improve the ANN's performance.

It is worth mentioning that both the number of hidden layers and the number of input features are considered as the ANN's hyperparameters. Generally, a hyperparameter is set beforehand and cannot be learned during training. On the other hand, color and texture features are the ANN parameters and are learned during training (Makwe & Rathore, 2021).

Some ANN properties, e.g., inter-layer weights, take their initial values based on generated random numbers. These random numbers make a specific ANN to present slightly different results after each training session regardless of experiencing the same training data set. This behavior is not desirable when particular items, e.g. different color spaces, are evaluated. In the present work, a seed is assigned to avoid this random behavior. A seed is a number used to

initialize a pseudorandom number generator that generates all random numbers demanded by an ANN.

As mentioned earlier, this study is established on the Python platform. The scikit-image package of Python for GLCM calculation, uses a texture property called *energy*, which relates to the property *uniformity* as follows:

$$Energy = \sqrt{uniformity} \tag{15}$$

Since the investigated image processing algorithm had been established based on *uniformity*, the same texture feature was deployed for the ANN proposed in section 4.1. Nevertheless for the rest of this paper, the *energy* texture feature is used to be consistent with the scikit-image package. To ensure that this change does not significantly affect the results an ANN with *4[HSV+energy]-2^4-2* architecture was trained and tested. The difference between the overall performance of *4[HSV+uniformity]-2^4-2* and *4[HSV+energy]-2^4-2* architectures is negligible (0.4%). The ANN with *4[HSV+energy]-2^4-2* architecture is considered as the baseline for the remainder of this paper.

How well a network performs on a test dataset through generalization, is mainly dependent on its architecture. Therefore, in this sub-section, several networks with different numbers of layers and neurons are evaluated as follows:

- The number of hidden layers varies between 2 and 6.
- The number of neurons is equal in each hidden layer.
- The number of neurons in a hidden layer can be 2^n, with $n = 0, 1, 2, \ldots, 8$.
- The performance of each architecture is calculated for 3 seeds (46, 200, 740), and their average value is reported as performance.

After training all plausible architectures, the overall performance of each architecture has been calculated and is shown in Figure 9. As can be seen, a simultaneous increase of both hidden layers and neurons does not necessarily lead to better performance. The best performing

architecture is the *4[HSV+energy]-5×2^4-2* architecture, consisting of 5 hidden layers containing 2^4 neurons in each. A comparison between the best multi-hidden-layer architecture with the baseline architecture, *4[HSV+energy]-2^4-2*, can be found in Table 5. It can be seen that architecture optimization leads to an increase in overall performance (+3.9%). This is mainly attributed to its ability to better handle misleading colors; more than 16% improvement is reached for dataset C. A slight decrease (-1.7%) is observed in its ability to deal with uneven illumination, dataset B, but this can be compensated for as will be discussed later. There is no significant impact on processing time, which can be explained by the relatively low number of neurons in the hidden layers, while feature extraction accounts for most of the time.

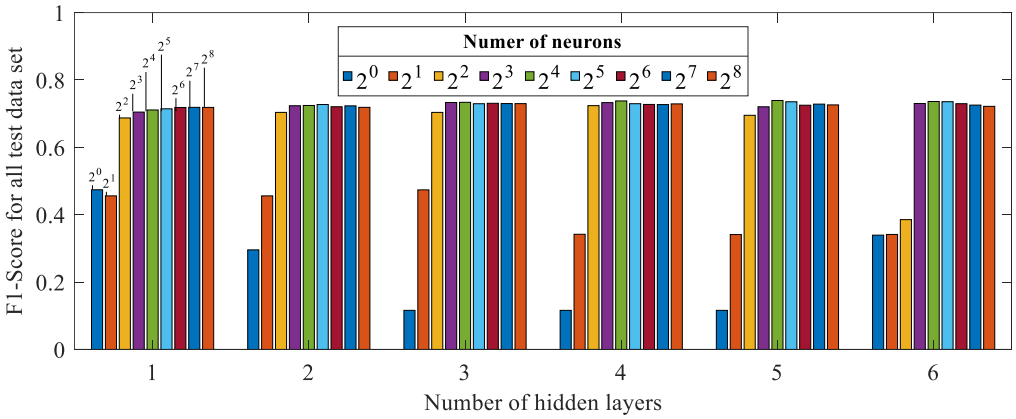


Figure 9: F1-score of all test data set for all combinations of hidden layers and neurons when the input feature consists of HSV color space channels and energy.

Table 5: Comparing the F1-score and processing time of the 4[HSV+energy]-2^4-2 and 4[HSV+energy]-5×2^4-2 architectures.

Images of	*4[HSV+energy]-2^4-2*	*4[HSV+energy]- 5×2^4-2*	
	Absolute value	Absolute value	Relative value[1]
All datasets	0.7110	0.7391	+3.9%
Dataset A	0.8129	0.8304	+2.1%
Dataset B	0.7033	0.6913	-1.7%
Dataset C	0.5404	0.6273	+16.1%

Time/image [s]	0.25	0.25	0.00%

[1] *Relative value is calculated based on the corresponding absolute value of the 4[HSV+energy]-2^4-2 architecture.*

5. Optimisation of the proposed ANN architecture

5.1. Color Space

The seven color spaces introduced in section 2.2 are evaluated in this subsection, while *energy* is consistently used as the texture feature. Table 6 presents the *F1-score* of the different ANNs with an architecture of *4[color space +energy]-5×2^4-2*.

Table 6: F1-score of different ANNs with an architecture of 4[color space +energy]-5×2^4-2 and seven different color spaces. Numbers in parentheses show the rank of the algorithm for each dataset.

Images of	*4[color space +energy]-5×2^4-2*			
	CIE $L^*u^*v^*$	*BGR*	*HSV*	*YUV*
All datasets	0.7426(1)	0.7410(2)	0.7391(3)	0.7368(4)
Dataset A	0.8253(2)	0.8190(5)	0.8304(1)	0.8192(4)
Dataset B	0.7026(2)	0.7066(1)	0.6913(6)	0.6932(4)
Dataset C	0.6377(2)	0.6388(1)	0.6273(6)	0.6360(3)
Time/image [s]	0.25(1)	0.25(1)	0.25(1)	0.25(1)

	YCrCb	*CIE $L^*a^*b^*$*	*HSL*
All datasets	0.7354(5)	0.7351(6)	0.7103(7)
Dataset A	0.8128(7)	0.8183(6)	0.8211(3)
Dataset B	0.7006(3)	0.6921(5)	0.6578(7)
Dataset C	0.6346(4)	0.6326(5)	0.5688(7)
Time/image [s]	0.26(2)	0.25(1)	0.25(1)

It is clear that whilst using the *HSV* color space for corrosion detection leads to relatively good results, it is not the best performing one overall. The only dataset for which it yields optimal results is dataset A, where no real difficulties encountered in real life have been introduced. Comparing all color spaces, *CIE L*u*v** has the best overall rank. Although the *BGR* color space has a better result than *CIE L*u*v** for datasets B and C, it does not show a good result for dataset A (rank 5 out of 7). Therefore, *CIE L*u*v** color space is considered the best choice for corrosion detection of the current test dataset. Changing the color space has no real influence on the processing time, except for a slight increase for the *YCrCb* color space.

When comparing *4[CIE L*u*v*+energy]-5×2^4-2* with the values for the baseline architecture *4[HSV+energy]-2^4-2*, Table 7, there is a clear improvement in overall performance (+4.4%) and a significant improvement for dataset C (+18.0%). The performance of the previous architecture, *4[HSV+energy]-5×2^4-2*, over dataset B was -1.7% of the corresponding performance of the baseline architecture, Table 5. The current architecture, *4[CIE L*u*v*+energy]-5×2^4-2,* alleviates the situation and has a negligible performance reduction for dataset B (-0.1%).

Table 7: Comparing the F1-score and processing time of the 4[HSV+energy]-2^4-2 and 4[CIE L*u*v*+energy]-5×2^4-2 architectures.

Images of	*4[HSV+energy]-2^4-2*	*4[CIE L*u*v*+energy]- 5×2^4-2*	
	Absolute value	Absolute value	Relative value[1]
All datasets	0.7110	0.7426	+4.4%
Dataset A	0.8129	0.8253	+1.5%
Dataset B	0.7033	0.7026	-0.1%
Dataset C	0.5404	0.6377	+18.0%
Time/image [s]	0.25	0.25	0.00%

[1] *Relative value is calculated based on the corresponding absolute value of the 4[HSV+energy]-2^4-2 architecture.*

Whilst only one color space is used in image processing techniques, nothing stands in the way of adding more color spaces to the input layer of the proposed ANN. This can be helpful since using multiple color spaces simultaneously can lead to a better color interpretation while it is not accompanied with a significant penalty in the processing time. To evaluate the potential of this idea, two color spaces are used simultaneously. This leads to 7 input features and thus an ANN with the format *7[color space-1 + color space-2 +energy]-5×2^4-2*.

After evaluating all possible combinations, the *7[CIE L*u*v*+YUV+energy]-5×2^4-2* architecture presents the best overall performance. Table 8 shows the result of this optimal architecture as well as the baseline architecture *4[HSV+energy]-2^4-2*. As can be seen, the ANN architecture *7[CIE L*u*v*+YUV+energy]-5×2^4-2* hits a slightly better overall performance compared to the architecture based on the *CIE L*u*v** color space that was reported in Table 7. The former and latter are respectively 4.8% and 4.4% better than the reference architecture. The architecture with 7 input features shows not only a better result for dataset B compared to architecture *4[CIE L*u*v*+energy]-5×2^4-2*, Table 7, but also outperforms the baseline architecture. ANN architecture *7[CIE L*u*v*+YUV+energy]-5×2^4-2* furthermore delivers significantly better results for those images containing misleading objects (+19.8%), database C, compared to the reference architecture. Lastly, using two color spaces causes a mere 4% increase in processing time, which can be considered an insignificant disadvantage compared to the significant performance improvement.

Table 8: Comparing the F1-score and processing time of the 4[HSV+energy]-2^4-2 and 7[CIE L*u*v*+YUV+energy]-5×2^4-2 architectures.

Images of	*4[HSV+energy]-2^4-2*	*7[CIE L*u*v*+YUV+energy]- 5×2^4-2*	
	Absolute value	Absolute value	Relative value[1]
All datasets	0.7110	0.7450	+4.8%

Dataset A	0.8129	0.8241	+1.4%
Dataset B	0.7033	0.7042	+0.1%
Dataset C	0.5404	0.6472	+19.8%
Time/image [s]	0.25	0.26	+4.0%

[1] *Relative value is calculated based on the corresponding absolute value of the 4[HSV+energy]-2^4-2 architecture.*

5.2. Texture Features

Considering the ANN architecture *7[CIE L*u*v*+YUV + texture feature]-5×2^4-2*, the texture features of Equations (3) to (6) are evaluated in order to determine the most appropriate one. Table 9 presents the *F1-score* of these architectures. The *dissimilarity* and *homogeneity* texture features present the best performance regarding the datasets B and C, respectively. However, the architecture with the *energy* texture feature has the best overall performance. Moreover, the results of this architecture show a minimum deviation in rank for different datasets, indicating the robustness of this texture feature for corrosion detection.

Table 9: Performance evaluation of 7[CIE L*u*v*+YUV + texture feature]-5×2^4-2 architecture considering different texture features. Numbers in parentheses show the rank of the algorithm for each dataset.

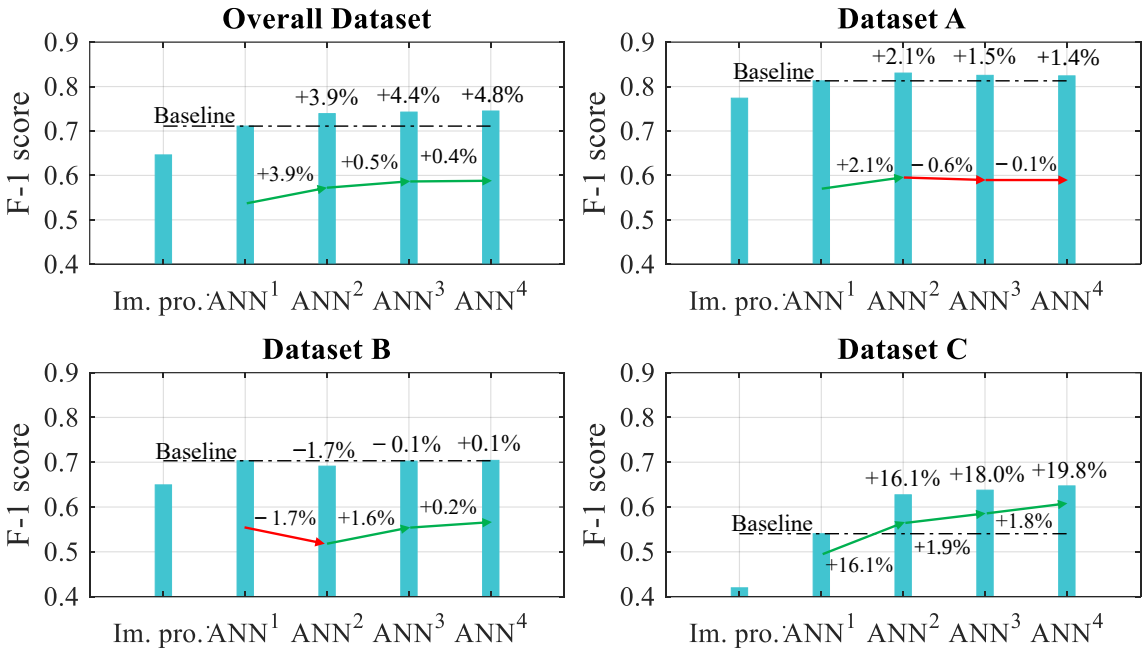
Images of	*7[CIE L*u*v*+YUV + Texture feature]-5×2^4-2*				
	Energy	*Contrast*	*Homogeneity*	*Correlation*	*Dissimilarity*
All datasets	0.7450(1)	0.7287(4)	0.7400(2)	0.7206(5)	0.7371(3)
Dataset A	0.8241(1)	0.8213(3)	0.8171(4)	0.7926(5)	0.8235(2)
Dataset B	0.7042(2)	0.6891(5)	0.6976(4)	0.7017(3)	0.7061(1)
Dataset C	0.6472(2)	0.6063(5)	0.6475(1)	0.6137(4)	0.6168(3)
Time/image [s]	0.26(1)	0.26(1)	0.26(1)	0.26(1)	0.26(1)

6. Discussion

Figure 10 presents a summary of the best architecture in each step of this study. As shown, all evaluated ANNs outperformed the image processing algorithm for all investigated datasets (i.e., uniform illumination, non-uniform illumination, misleading objects).

The capability of architectures with multiple hidden layers for better corrosion detection has been investigated. Results reveal that such architectures can yield better results for the overall dataset as compared to an architecture with a single hidden layer (increase of 3.9%)As shown on Figure 10, the improvement is most pronounced for the subset of images with misleading objects (dataset C), but negligible or slightly worse for the subset containing images with different levels of illumination (dataset B).

Based on an evaluation of different color spaces, it was shown that using the *CIE L*u*v** color space instead of *HSV* led to a 1.6% increase in the relative performance for the subset of images with different levels of illumination (dataset B), Figure 10. This proves that using a color space with a channel that better describes illumination, can help alleviate the challenges encountered with images containing different illumination levels. Using two color spaces, and thus almost doubling the number of input features, increased computing time with not more than 4%. In return, the architecture's performance is slightly improved when *CIE L*u*v** and *YUV* color spaces are used in combination, Figure 10. Different texture features have also been evaluated. Results show that when the *energy* texture feature is incorporated in the architecture, the best overall result can be achieved. Figure 10 shows a deterioration for the subset containing non-challenging images (dataset A). This slight deterioration is considered insignificant since dataset A typically gets a very high hit rate, ensuring a desirable margin of safety.



Im. pro. = Image processing algorithm, ANN[1]: *4[HSV+ energy]-2⁴-2*,

ANN[2]: *4[HSV + energy]-5×2⁴-2*, ANN[3]: *4[CIE L*u*v + energy]-5×2⁴-2*,

ANN[4]: *4[CIE L*u*v + YUV + energy]-5×2⁴-2*

Figure 10: F1-score of the different ANNs studied in this paper.

7. Conclusions

This paper has proven the outstanding merit of artificial neural networks compared to a previously developed image processing algorithm for corrosion detection. As stated, the required size of the proposed ANN's training dataset is a few orders of magnitude of that of a typical CNN, demonstrating its practical relevance. The paper's findings prove that combining the deep learning feature of CNNs with the pixel-based nature of ANNs builds a robust ANN for corrosion detection.

The presence of different levels of illumination and misleading objects in an image are considered as two major challenges in image-based corrosion detection techniques. Different inputs to the ANN architecture, including color and textural features, have been investigated to address these issues. First, different color spaces were investigated. Results showed *CIE L*u*v**

color space works better than the others for corrosion detection. Then, the idea of having two color spaces was examined. Although a slight increase occurred in the computing time, combining *CIE L*u*v** and *YUV* color spaces yielded better results than an architecture with one color space. Different texture features were also studied, but the preselected texture feature, *energy*, worked better than the others.

Eventually, a feed-forward neural network with an architecture of *7[CIE L*u*v*+YUV+ energy]-5×2^4-2* having 7 input features and 5 hidden layers containing 16 neurons in each is considered as the best ANN for image-based corrosion detection, based on the tested architectures and datasets used in this work.

Appendix A

All of the investigated architectures have been trained by the training dataset shown in Figure A.1. The test dataset is shown in Figure A.2.

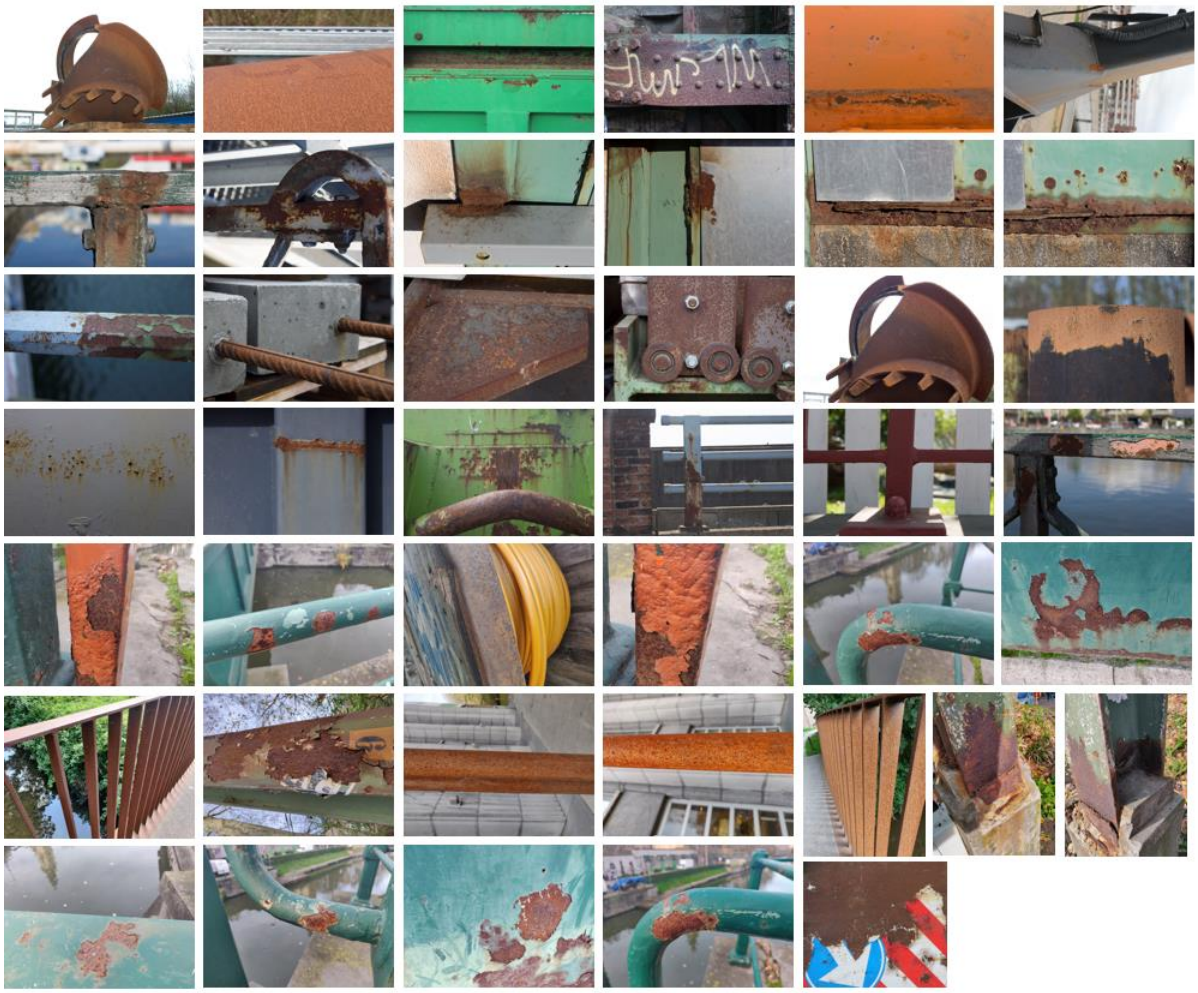


Figure A.1: Training dataset.

Figure A.2: Test dataset.

Statements and Declarations

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the Vlaio under Grant 179P04718W; Strategic Initiative Materials in Flanders; and IBN Offshore Energy.

Data Availability Statement

Some or all data, models, or code generated or used during the study are proprietary or confidential in nature and may only be provided with restrictions.

References

Ahuja, S. K., & Shukla, M. K. (2018). A survey of computer vision based corrosion detection approaches. *Smart Innovation, Systems and Technologies*, *84*, 55–63. https://doi.org/10.1007/978-3-319-63645-0_6

Alessi, P.J.; Carter, E.C.; Fairchild, M. D. (2004). *CIE Colorimetry 15* (3rd ed., Vol. 1). International commision on illumination.

Bastian, B. T., N, J., Ranjith, S. K., & Jiji, C. V. (2019). Visual inspection and characterization of external corrosion in pipelines using deep neural network. *NDT and E International*, *107*, 102134. https://doi.org/10.1016/j.ndteint.2019.102134

Bonnin-Pascual, F., & Ortiz, A. (2014). Corrosion Detection for Automated Visual Inspection. In *Developments in Corrosion Protection* (pp. 619–632). INTECH. https://doi.org/http://dx.doi.org/10.5772/57209

Brewer, C. A. (1999). Color Use Guidelines for Data Representation. In *Proceedings of the Section on Statistical Graphics, American Statistical Association* (pp. 55–60).

Cha, Y. J., Choi, W., & Büyüköztürk, O. (2017). Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks. *Computer-Aided Civil and Infrastructure Engineering*, *32*(5), 361–378. https://doi.org/10.1111/mice.12263

Cha, Y. J., Choi, W., Suh, G., Mahmoudkhani, S., & Büyüköztürk, O. (2018). Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types. *Computer-Aided Civil and Infrastructure Engineering*, *33*(9), 731–747. https://doi.org/10.1111/mice.12334

Chen, P.-H., Yang, Y.-C., & Chang, L.-M. (2009). Automated bridge coating defect recognition using adaptive ellipse approach. *Automation in Construction*, *18*(5), 632–

643. https://doi.org/10.1016/j.autcon.2008.12.007

Cole, P. T., & Watson, J. R. (2006). Acoustic Emission for Corrosion Detection. In *Advanced Materials Research* (Vol. 13–14, pp. 231–236). https://doi.org/10.4028/www.scientific.net/amr.13-14.231

Dey, N., Borra, S., Ashour, A. S., & Shi, F. (Eds.). (2019). *Machine Learning in Bio-Signal Analysis and Diagnostic Imaging*. *Machine Learning in Bio-Signal Analysis and Diagnostic Imaging*. ACADEMIC PRESS, INC. https://doi.org/10.1016/c2017-0-02827-6

Doshvarpassand, S., Wu, C., & Wang, X. (2019). An overview of corrosion defect characterization using active infrared thermography. *Infrared Physics and Technology*, *96*, 366–389. https://doi.org/10.1016/j.infrared.2018.12.006

Furuta, H., Deguchi, T., & Kushida, M. (1995). Neural network analysis of structural damage due to corrosion. *Annual Conference of the North American Fuzzy Information Processing Society - NAFIPS*, 109–114. https://doi.org/10.1109/isuma.1995.527678

Haralick, R. M., Shanmugam, K., & Dinstein, I. (2007). Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-3*(6), 610–621. https://doi.org/10.1109/tsmc.1973.4309314

He, Y., Tian, G., Zhang, H., Alamin, M., Simm, A., Jackson, P. (2012). Steel Corrosion Characterization Using Pulsed Eddy Current Systems. *Ieee Sensors Journal*, *12*(6), 2113–2120.

Jahanshahi, M. R., & Masri, S. F. (2013). Effect of color space, color channels, and sub-image block size on the performance of wavelet-based texture analysis algorithms: An application to corrosion detection on steel structures. In *Computing in Civil Engineering*

- *Proceedings of the 2013 ASCE International Workshop on Computing in Civil Engineering* (pp. 685–692). Reston, VA: American Society of Civil Engineers. https://doi.org/10.1061/9780784413029.086

Khan, A., Ali, S. S. A., Anwer, A., Adil, S. H., & Meriaudeau, F. (2018). Subsea pipeline corrosion estimation by restoring and enhancing degraded underwater images. *IEEE Access*, *6*, 40585–40601. https://doi.org/10.1109/ACCESS.2018.2855725

Khayatazad, M., De Pue, L., & De Waele, W. (2020). Detection of corrosion on steel structures using automated image processing. *Developments in the Built Environment*, *3*, 100022. https://doi.org/10.1016/j.dibe.2020.100022

Koch, G., Varney, J., Thopson, N., Moghissi, O., Gould, M., & Payer, J. (2016). International Measures of Prevention , Application , and Economics of Corrosion Technologies Study. *NACE International*, 1–216. Retrieved from http://impact.nace.org/

Livens, S., Scheunders, P., de Wouwer, G., van Dyck, D., Smets, H., Winkelmans, J., & Bogaerts, W. (1995). Classification of corrosion images by wavelet signatures and LVQ networks. In V. Hlaváč & R. Šára (Eds.), *Computer Analysis of Images and Patterns* (pp. 538–543). Berlin, Heidelberg: Springer Berlin Heidelberg.

Makwe, A., & Rathore, A. S. (2021). An Empirical Study of Neural Network Hyperparameters. *Advances in Intelligent Systems and Computing*, *1176*, 371–383. https://doi.org/10.1007/978-981-15-5788-0_36

McCrea, A., Chamberlain, D., & Navon, R. (2002). Automated inspection and restoration of steel bridges - A critical review of methods and enabling technologies. *Automation in Construction*, *11*(4), 351–373. https://doi.org/10.1016/S0926-5805(01)00079-6

Naik, D. L., Sajid, H. U., Kiran, R., & Chen, G. (2020). Detection of corrosion-indicating

oxidation product colors in steel bridges under varying illuminations, shadows, and wetting conditions. *Metals*, *10*(11), 1–19. https://doi.org/10.3390/met10111439

Ortiz, A., Bonnin-Pascual, F., Garcia-Fidalgo, E., & Company, J. P. (2016). Visual inspection of vessels by means of a micro-aerial vehicle: An artificial neural network approach for corrosion detection. In *Advances in Intelligent Systems and Computing* (Vol. 418, pp. 223–234). Springer Verlag. https://doi.org/10.1007/978-3-319-27149-1_18

Petrovic, Z. (2016). Catastrophes caused by corrosion. *Vojnotehnicki Glasnik*, *64*(4), 1048–1064. https://doi.org/10.5937/vojtehg64-10388

Podpora, M., Korbaś, G. P., & Kawala-Janik, A. (2014). YUV vs RGB—Choosing a Color Space for Human-Machine Interaction. In *Position papers of the 2014 Federated Conference on Computer Science and Information Systems* (Vol. 3, pp. 29–34). PTI. https://doi.org/10.15439/2014f206

PyTorch. (n.d.). Retrieved October 23, 2020, from https://pytorch.org/

See, J. E., Drury, C. G., Speed, A., Williams, A., & Khalandi, N. (2017). The Role of Visual Inspection in the 21st Century. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *61*(1), 262–266. https://doi.org/10.1177/1541931213601548

Sharma, S., & Mukherjee, A. (2015). Ultrasonic guided waves for monitoring corrosion in submerged plates. *Structural Control and Health Monitoring*, *22*(1), 19–35. https://doi.org/https://doi.org/10.1002/stc.1657

Shen, H. K., Chen, P. H., & Chang, L. M. (2013). Automated steel bridge coating rust defect recognition method based on color and texture feature. *Automation in Construction*, *31*, 338–356. https://doi.org/10.1016/j.autcon.2012.11.003

Sola, J., & Sevilla, J. (1997). Importance of input data normalization for the application of

neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, *44*(3 PART 3), 1464–1468. https://doi.org/10.1109/23.589532

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). *On the importance of initialization and momentum in deep learning*. *30th International Conference on Machine Learning, ICML 2013*.

Szandała, T. (2021). Review and comparison of commonly used activation functions for deep neural networks. *Studies in Computational Intelligence*, *903*, 203–224. https://doi.org/10.1007/978-981-15-5495-7_11

Zhu, H., Luo, H., Ai, D., & Wang, C. (2016). Mechanical impedance-based technique for steel structural corrosion damage detection. *Measurement*, *88*, 353–359. https://doi.org/10.1016/J.MEASUREMENT.2016.01.041