

Development of a System and Deep Learning Method for Metal Surface Corrosion Detection and Evaluation in Industrial Equipment

Moh. Rizanto Juliarsyah^{1*}, Irwanda Yuni Pungkiarto¹,
Faradilla Fauziyah Risnawati¹, Khoirul Anwar¹, Dhia Fairuz Shabrina²

¹Department of Mechanical Engineering, Politeknik Negeri Malang, Indonesia

²Department of Mechanical Engineering, ITS, Sukolilo Surabaya 60111, Indonesia

Received: 5 September 2025, Revised: 25 September 2025, Accepted: 2 October 2025

Abstract

Corrosion inspection of industrial assets is still dominated by subjective and inconsistent visual inspections. This study develops and validates a deep learning-based corrosion area detection system on metal surfaces in the context of heavy equipment through a binary segmentation task (corrosion vs. non-corrosion). Three architectures were compared: UNet, VGG16–Random Forest, and VGG16–UNet, using 600 annotated images measuring 512×512 pixels taken under lighting conditions of 50–150 lux. The workflow included preprocessing, augmentation, training for 30, 50, and 100 epochs, and evaluation of accuracy, precision, recall, IoU/Jaccard, Dice, and confusion matrix per pixel (positive = corrosion). The results show that VGG16–UNet provides the best performance; in the 150 lux test, it achieved 98.96% accuracy, 0.9934 precision, and 0.994 recall, with good consistency across lighting variations and data scales. These findings confirm the effectiveness of a pre-trained encoder combined with skip connections to recover fine corrosion boundaries and produce reliable corrosion maps. The proposed approach has the potential to standardize the inspection process and accelerate decision-making in reliability-based maintenance practices.

Keywords: corrosion, deep learning, accuracy, VGG16–UNet, automatic inspection

1. Introduction

Corrosion is chemical and electrochemical process that damages metals, which is the main cause of asset damage, safety incidents, and increased life cycle costs in industrial sectors, such as heavy equipment operations and civil infrastructure, including pipelines, bridges, and telecommunications towers [1–3]. Visual inspection during normal routines remains the most commonly used inspection method due to its simplicity and ease of use, but this method is labor-intensive, operator-dependent, and increasingly performed under less than ideal or even hazardous conditions, making consistency and repeatability difficult. These limitations have driven the development of camera-based automated inspection methods that can be applied to large assets and produce auditable output, supporting current maintenance and compliance processes [4–8].

However, eye-based visual assessment is inefficient, experience-dependent, and can even be deadly when the circumstances of the structures are hazardous [9, 10]. Hence, automatic visual inspection is highly desirable for addressing the limitations of the human-based visual

method. Various approaches for efficiently assessing infrastructure have been studied [11, 12]. Most research on damage detection focuses on image processing techniques [13, 14]. This computer vision-based approach can detect specific types of damage, such as cracks, concrete spalling, and steel faults [4, 9]. Furthermore, automatic damage detection algorithms have been effectively used in autonomous moving platforms [7, 8].

Machine Learning (ML) offers an alternative to manual corrosion detection, which primarily relies on visual inspection for maintenance and assessment. Ensuring the dependability and effectiveness of inspection systems is crucial for public safety and economic efficiency. ML introduces new methods to understand and quantify data-intensive processes, especially in agriculture. When combined with big data and high-performance computing, ML is described as a scientific discipline enabling machines to learn without explicit programming [15]. A machine model can learn spatial characteristics of corrosion, such as color or texture, which reduces the time necessary for detection. Recent research classifies machine learning into three categories: supervised, unsupervised, and reinforcement learning [16].

*Corresponding author. Email: Mohammad.rizanto@polinema.ac.id,
© 2025. The Authors. Published by LPPM ITS.

Supervised learning methods rely on labeled data samples to model the characteristics of behavioral size distribution in various applications. These methods are categorized into classification and regression tasks. Classification involves categorical output variables, such as color or health status. This study focuses on supervised learning using Convolutional Neural Networks (CNNs), a type of Deep Learning (DL) method that allows machines to analyze raw data and automatically determine the necessary representations for classification or detection. Deep learning architectures can be single, using one approach, or double, combining two methods [17–20].

This work is positioned in a growing literature on corrosion segmentation that spans from conventional to DL-based methods and consistently encounters sensitivity to dataset organization, illumination, and annotation density. Corpus-specific research, such as RustSEG, demonstrates the capability of automatic corrosion segmentation, highlighting that raw performance is a function of label quality and scene diversity [21]. The robust performance of UNet on its dataset is illustrated by the "Application of Deep-Learning Architecture for Image Analysis-based Corrosion Detection" research, which supports the value of encoder–decoder architectures and highlights the effects of data scale and training depth [22]. Direct quantitative comparison across studies remains limited by various tasks, acquisition processes, and definitions of ground truth; hence, in this paper, deployability and field realism are introduced as the emphasis to provoke conclusions translatable into practice.

This paper presents three results under this specified context. Firstly, it provides a pixel-annotated corrosion image dataset normalized to 512×512 -pixel, acquired under inhomogeneous illumination, and provides a realistic benchmark for segmentation under field-like conditions. Second, it gives a direct comparison of UNet, VGG16-RF, and VGG16-UNet in terms of accuracy–efficiency trade-offs suitable for running on modest hardware, and separates the contribution of pretrained encoders and skip-connected decoders. Third, it constructs and employs a validation scheme involving pixel-level confusion-matrix analysis and overlap measures under illumination variation that yields interpretable evidence of robustness for operational deployment. The remainder of the paper explains the dataset, annotation scheme, and model parameters; presents quantitative and qualitative results on data sizes and lighting; connects the findings to previous work and discusses applications; and concludes with limitations, particularly the current binary scope. The main objective of this study is to develop and validate a deep learning-based corrosion area detection system on metal surfaces through binary segmentation, and to compare UNet, VGG16-RF, and VGG16-UNet at various data scales and illuminations.

2. Experimental/theoretical method

Details materials, equipment, and experimental procedure, or details theoretical or calculation procedure. The experimental section shall provide the necessary information for reproducing the results.

2.1. Types of Corrosion

It is critical to understand the type of corrosion since it is required to locate testing materials for studies. According to [23], some examples of corrosion types that can occur on iron plates include:

- Uniform Corrosion

Uniform corrosion refers to corrosion that happens uniformly across a material's whole surface. It produces a constant and equitable reduction in thickness. This type of corrosion is more predictable and easier to handle than localized corrosion since the material deterioration rate can be anticipated more precisely. Uniform corrosion happens when a corrosive agent is evenly dispersed and affects the full surface area of the material exposed to it.

- Pitting Corrosion

Pitting corrosion is a type of corrosion that causes microscopic, often undetectable pits or holes on a material's surface. This sort of corrosion is very harmful since it can cause extensive structural damage while only impacting a small region. Pitting corrosion is commonly seen in metals, particularly stainless steel and aluminum, and is frequently induced by exposure to chloride ions or other aggressive chemicals. Pits can penetrate deep into the material, resulting in probable failure without significant material loss.

- Stress Corrosion Cracking

Stress Corrosion Cracking (SCC) is a type of localized corrosion caused by the combination of tensile stress and a corrosive environment. It causes the creation of cracks in materials, particularly metals, which can spread quickly, resulting in unexpected and catastrophic failure. SCC is sometimes difficult to identify because it can develop beneath surface coatings and in locations that are inaccessible for inspection. The presence of residual or applied tensile strains, the kind of material, and the unique chemical environment are all factors that contribute to SCC. Chlorides, caustic solutions, and hydrogen sulfide are common environmental causes of SCC.

- Erosion Corrosion

Erosion corrosion is a type of corrosion that results from the combined effects of mechanical wear and chemical attack on a material's surface. This process is most common in areas with high fluid flow rates, such as pipelines, pumps, and turbines. The fast-moving fluid dissolves the material's protective oxide coating, exposing the naked metal to the corrosive atmosphere. This leads to accelerated material loss, which can cause substantial damage over time. Some example images of the above corrosion are shown in Figure 1 below.

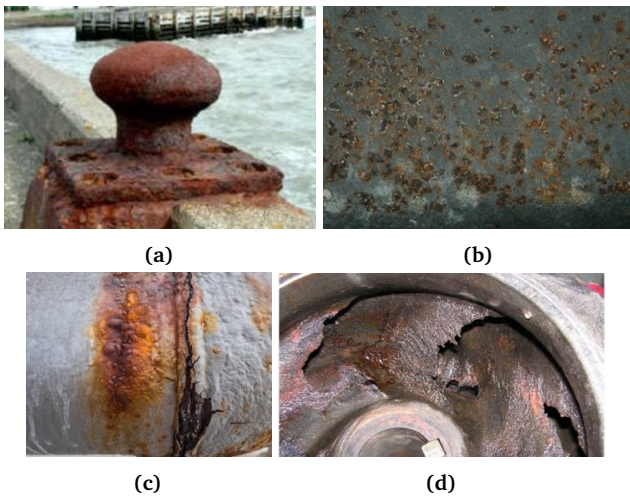


Figure 1. (a) Uniform Corrosion, (b) Pitting Corrosion, (c) Stress Corrosion Cracking, (d) Erosion Corrosion.

The images utilized are 600 images, 512×512 -pixel images labeled inside the corrosion area. They were selected to represent four main types of corrosion, i.e., uniform corrosion (240 images; 40%), pitting corrosion (180 images; 30%), stress corrosion cracking (SCC) (90 images; 15%), and erosion corrosion (90 images; 15%). This work depicts the kind of prevalence most likely in industrial equipment, where pitting corrosion and uniform corrosion are most common compared to SCC and erosion corrosion. In every type, we have included changes in material, roughness of the surface, condition of coating/paint, light, and image takeout distance.

2.2. Preparing the Dataset



Figure 2. Dataset File for Input Image Before Labeling

The dataset of corrosion photos was created by annotating each pixel. Initially, the dataset was divided into folders, with a total of 600 photos. The acquired data included photos of corroded iron. Some iron corrosion datasets were labeled as shown in Figure 2, with a size of 512×512 -pixel and taken using a camera under various lighting conditions. To build the defect identification system, the image dataset was separated into 200, 400, and 600 training datasets, with 20% of each being used for testing. The acquired dataset was then carefully selected to achieve the best detection results. Images having identical patterns, supplied from many samples of varied sizes, underwent traditional preprocessing, which included resizing to standardized input in accordance with the specified

architecture. The required image input size was 512×512 pixels, which was collected by capturing and downloading metal photographs with corrosion on the internet. As a result, the dataset developed by the researcher is not publicly available, and the dataset utilized is solely the researcher's creation.

To discriminate between rust and background, area segmentation was performed with LabelMe, an Anaconda software tool. The images that had been modified to the input architecture were then labeled for each pixel based on the label class. The technique typically employs two labels, rust and background, to establish the label class for each pixel. Figure 3 shows that the backdrop class is black, and the rust is red. To make segmentation easier at this step, polygon tools might be employed to follow the contour of the corrosion.

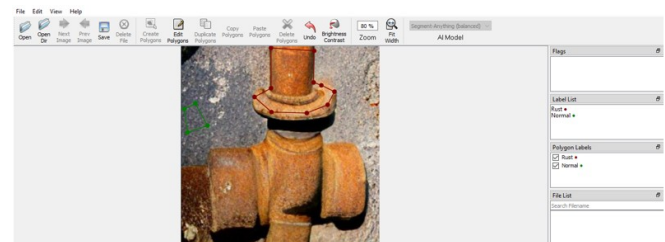


Figure 3. Labeling Input Dataset Using LabelMe

2.3. Selection of Specific Architecture

• UNet

UNet is a commonly used semantic segmentation architecture, selected in this study for its simplicity and popularity. The UNet design is a fully connected network with encoders and decoders. Convolutional layers use filters to extract low and high-dimensional characteristics through iterative training. This architecture primarily encodes images using a CNN for down-sampling. UNet concatenates the down-sampling and up-sampling parts to decode a segmentation mask [24]. Figure 4 below, adapted from previous research [2] by the same researcher, illustrates the VGG16 scheme.

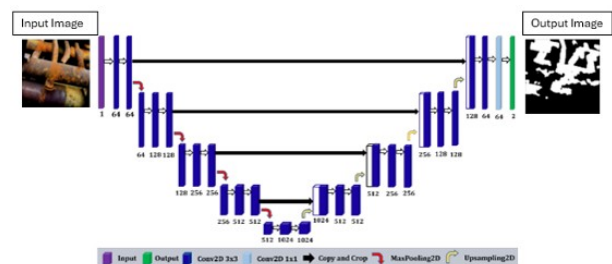


Figure 4. Unet Architecture Model

• VGG16-RandomForest

VGG is a pre-trained model and has 138 million parameters. VGG trains on over 14 million images with up to 1000 classes and learns to detect common features from the images.

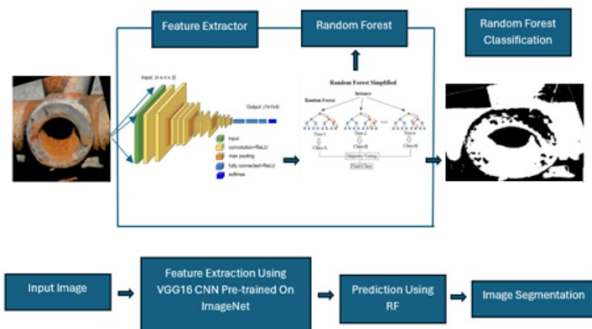


Figure 5. VGG16-RandomForest Architecture Model

There are 16 and 19 weight layers in the network for VGG-16 and VGG-19, respectively. This research uses VGG-16 as a base model and modifies it to create different networks. When VGG16 reaches a certain test accuracy on ImageNet, and therefore high performance, the pre-trained weights are retained, and only three Fully Connected Layers or Dense Layers are modified to fine-tune the neural network. In this work, the features extracted from VGG16 are provided as input to RF to reduce the training time and improve classification accuracy. Figure 5 explains the VGG16 schematic. All resized images in this model are 512x512-pixel.

• VGG16-Unet

VGG16-Unet is a variation of the UNet architecture that uses the VGG16 network as an encoder. The VGG16 network, notable for its deep convolutional layers, extracts high-level information from input images. In the VGG16-Unet design, the encoder (VGG16) is followed by a decoder network that reconstructs the segmented image using these high-level features. This combination takes advantage of VGG16's robust feature extraction capabilities and UNet's efficient segmentation skills, making it ideal for jobs such as biomedical picture segmentation. Figure 6 below, adapted from previous research [25] by the same researcher, illustrates the VGG16 scheme.

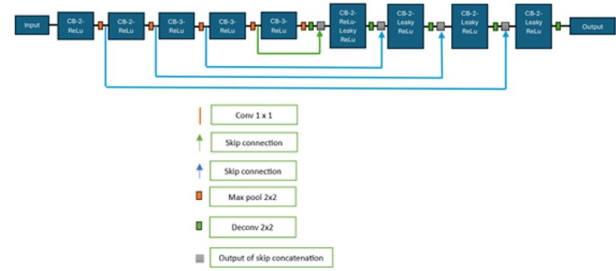


Figure 6. VGG16-UNet Architecture Model

2.4. Hyperparameter Tuning

Hyperparameter tuning is the process of tweaking the hyperparameters of a machine learning model to increase its performance. Unlike model parameters, which are learned during training, hyperparameters are selected before training and have a substantial impact on the model's capacity to learn effectively. Common hyperparameters include learning rate, batch size, epoch count, and architecture-specific settings like the number of layers or units in neural networks.

Based on Table 1, Adam is an optimization algorithm that combines the advantages of two other optimization algorithms: AdaGrad and RMSProp. Adam maintains momentum calculations and adaptive learning rates for each parameter, which helps the model converge faster and more efficiently. The choice of a learning rate of 0.001 is used for the stability of the training process, and with a value that is not too large, the risk of overshooting around the minimum is reduced, allowing the model to approach the minimum loss more stably. Increasing the number of epochs during deep learning model training can bring significant benefits in terms of convergence, model training, generalization ability, and validation metric monitoring. However, we must be cautious when increasing the number of epochs, as the model may begin to overfit if training continues for too long. This can be monitored through techniques such as early stopping, regularization, and cross-validation, which ensure that increasing the number of epochs yields optimal results without the risk of overfitting on the training data.

Table 1. Hyperparameter Tuning

No	Hyperparameter	Parameter Type
1.	Optimizer	Adam Optimizer (Adaptive Moment Estimation)
2.	Learning Rate	0.001
3.	Epoch	30, 50, 100
4.	Batch Size	14
5.	Loss Function Parameters Smooth	100
6.	Early Stopping Parameters Patience	3

Batch size is a critically important parameter that influences a wide range of aspects in the deep learning model training process. There is no one-size-fits-all solution; the optimal batch size will depend on several factors, including dataset size, hardware memory capacity, and the model's specific objectives. In this study, a batch size of 14 was required to obtain a good model for the segmentation process.

2.5. Model Training and Validation

To train the network, input images and segmentation masks or ground truth are fed into deep learning-based implementations of the UNet, VGG16-RandomForest, and VGG16-UNet architectures. A total of 200, 400, and 600 photos, representing 20% of the training dataset, were set aside for testing and validation, using segmented masks developed on Jupiter Notebook. The number of epochs was 30, 50, and 100. A batch separates each parameter, determining the number of points that must be processed before the model's internal parameters are updated. Epochs are used to determine how many times the learning algorithm will run throughout the training. The model was trained on data for approximately 14 hours, achieving consistent accuracy across prediction and validation. The model was successfully trained, with 92.46% binary training accuracy, 7.68% training loss, 98.96% binary validation accuracy, and 1.04% validation loss. The trained model was evaluated on a 10% validation dataset, and the prediction covered the corroded portion of the picture data; when it comes to model performance, the loss function is key. A universal loss function cannot be used for complex objectives such as segmentation. Thus, binary cross-entropy is the optimal technique for pixel-level classification issues [26, 27]. Data testing and validation are shown in Figure 7, as performed in previous studies [25] by the same researcher.

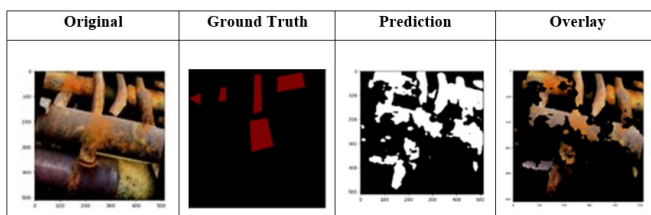


Figure 7. Examples of Validation Data Results

2.6. Confusion Matrix

The confusion matrix calculated in segmentation differs slightly from that calculated in object classification segmentation, in that the confusion matrix is calculated for pixels that have been detected. Corrosion will be included in the calculation of true positive corrosion, as shown in Figure 8 below is adapted from previous research [25] by the same researchers.

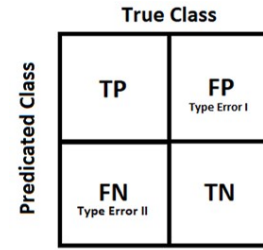


Figure 8. Confusion Matrix

3. Results and Discussion

3.1. Training Methods VGG16-UNet, VGG16-RandomForest, UNet with Different Hyperparameters

Table 2 below shows the training results for hyperparameter variations to determine the best parameters to use and to enable use on laptops/PCs with limited specifications, and reduce overfitting in a model generated from the training that has been carried out. Table 2 below shows the results of hyperparameter tuning with a comparison of batch sizes 12 and 14 using 30 epochs and the same dataset size of 200 images and 600 images. For the 200-image dataset, 30 epochs, and batch size 14, the VGG16-UNet method achieved an accuracy of 92%, the VGG16-RandomForest achieved an accuracy of 90%, and UNet achieved an accuracy of 89.81%. Then, on the 200-image dataset, 30 epochs, and batch size 12 for the VGG16-UNet method, the accuracy was 89.59%, VGG16-RandomForest achieved an accuracy of 89.03%, and UNet achieved an accuracy of 88.13%. In addition to accuracy, Table 22 below also concludes that hyperparameter tuning can be used to optimize training so that it can be performed on laptops with limited specifications and reduce model training time, so that it does not take too long.

Table 2 below aggregates the performance of three architectures, VGG16-UNet, VGG16-RF, and UNet, trained on 30 epochs with Adam on some data size and batch size cases; on 200 images and batch size 14, VGG16-UNet performed 92% accuracy (loss 8%, Val_{Acc} 85.66%, Val_{loss} 14.29%) and outperformed VGG16-RF (90%, 10%, 82.67%, 17%) and UNet (89.81%, 10.19%, 77.72%, 22.48%), a pattern that persisted when the batch size was reduced to 12 (VGG16-UNet 89.59% against VGG16-RF 89.03% vs UNet 88.13%); when scaled to data of 600 images (batch 12), the VGG16-UNet accuracy increased to 93.67–95.50% with loss 6.33–4.50% and Val_{Acc} 82.52–84.66% (Val_{loss} 17.36–15.22%), demonstrating the benefit of using more data to generalization, while computational cost also increased from around 3 hours 21–28 minutes (200 images) to ± 11 hours 15–21 minutes (600 images) on an Intel Core i5 gen-8 CPU with 63–88% utilization; Generally, VGG16-UNet performed best throughout, VGG16-RF was worse but relatively light, UNet was worst, and batch size change only slightly affected compared to data scale effect.

Table 2. Training Table for VGG16-UNet, VGG6-RandomForest, and UNet Methods with Different Hyperparameters

Metode	VGG16-UNet	VGG16-RF	UNet	VGG16-UNet	VGG16-RF	UNet	VGG16-UNet	VGG16-UNet
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Accuracy	92%	90%	89.81%	89.59%	89.03%	88.13%	93.67%	95.5%
Loss	8%	10%	10.19%	10.41%	10.98%	11.76%	6.33%	4.5%
Val_Acc	85.66%	82.67%	77.72%	84.16%	84.10%	76.71%	82.52%	84.66%
Val_loss	14.29%	17%	22.48%	15%	15.56%	23.38%	17.36%	15.22%
Batch Size	14	14	14	12	12	12	12	14
Epoch	30	30	30	30	30	30	30	30
Image Dataset	200	200	200	200	200	200	600	600
Total training time	03:31:44	03:27:06	03:25:31	03:28:29	03:24:23	03:21:31	11:15:42	11:20:55
Performance CPU (Intel(R) core (TM) i5 8th)	73%-80%	73%-80%	73%-80%	63%-70%	63%-70%	63%-70%	68-78%	76-88%

Quantitatively, VGG16-UNet outperformed UNet and VGG16-RF in all cases, with test accuracy being 93–95% when the data size was increased to 600 images, while increasing the number of epochs reduced loss and enhanced boundary clearness without a huge increase in training time. This is consistent with early research that pretrained encoders combined with skip connections have a pattern of improving boundary precision in corrosion images [24]. Relative to the RustSEG, which achieved <90% performance because of scene diversity and label quality [21, 28], our performance is better on this dataset, but cross-study comparison has to be read with caution because tasks, data curation, and evaluation protocols

are different. One other UNet paper on STCR also listed >90% on its outcome [22], our finding using VGG16-UNet reinforces that using a pretrained encoder does, in fact, provide a real benefit on small-medium sized datasets. Our strongest setup (VGG16-UNet) beats an unadorned UNet baseline on our dataset, as found previously that pre-trained encoders combined with skip connections enhance boundary restoration [29, 30].

3.2. Results of Learning & Testing using the UNet, VGG16-RandomForest, and VGG16-UNet methods to obtain the best method

Table 3. Overall results of learning & testing using the UNet, VGG16-RandomForest, and VGG16-UNet methods with a dataset of 200 images

Result Parameter	VGG16-UNet 200 images 30 Epoch	VGG16-RF 200 images 30 Epoch	UNet 200 images 30 Epoch
Train Accuracy	0.9246	0.900	0.8915
Train Loss	0.0768	0.19	0.1091
Train Accuracy Validation	0.8461	0.81	0.8299
Train Loss Validation	0.1555	0.1963	0.1709
K-Fold Cross Validation Acc	0.87	0.8514	0.83
K-Fold Cross Validation Loss	0.47961	0.4768	0.4513
Test Accuracy	0.87	0.8603	0.83
Test Loss	0.42	0.3606	0.48

Table 4. Segmentation Result

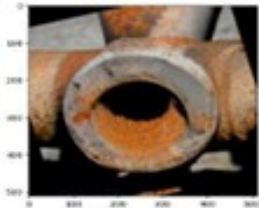
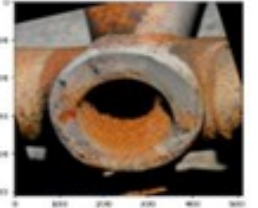
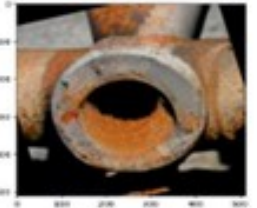






Description	VGG16-UNet 200 Images 30 Epoch	VGG16-RF 200 Images 30 Epoch	UNet 200 Images 30 Epoch
Input Image			
Prediction			
Overlay			

Table 3 above shows the overall results of learning and testing that have been carried out using the UNet, VGG16-RandomForest, and VGG16-UNet methods using 200 images for training and 25 images for testing, totaling 225 different images. However, during training and testing, each method must use the same images during the process to determine which method is the best and most accurate for use as a corrosion detection method. From the quantitative results shown in Table 3 above, it can be concluded that VGG16-UNet outperforms the other methods, with VGG16-UNet achieving the highest training accuracy of 92.46%. After cross-validation using the K-Fold cross-validation, VGG16-UNet also achieved the best accuracy of 87%. The purpose of cross-validation is to evaluate the performance of machine learning models more accurately. After the cross-validation process, a testing process was also conducted, and the VGG16-UNet method was found to be more accurate than other methods, with a testing accuracy of 87%. It can be concluded that the best method is the double architecture using the VGG16-UNet method. In addition to the quantitative results, the segmentation results using the three methods will also be shown in Table 4 below.

Table 4 below shows the corrosion segmentation results obtained using the UNet, VGG16-RandomForest, and VGG16-UNet methods, using 200 images for training and 25 images. The segmentation results above indicate

that the UNet method does not effectively segment all the corrosion effectively. The VGG16-RandomForest method can perform segmentation effectively, but still has segmentation errors. The VGG16-UNet method produces segmentation that is nearly perfect.







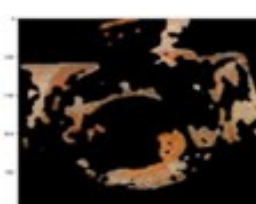
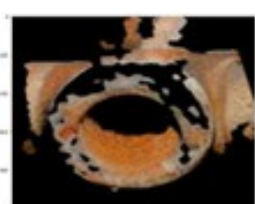
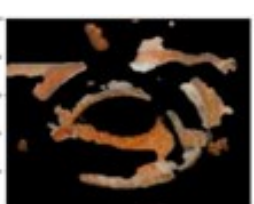
3.3. Overall Results of Learning & Testing using the VGG16-UNet method 400-Images Dataset

In this step, the training image dataset used consisted of 400 images, and the testing image dataset used consisted of 50 images. The dataset was trained using a double architecture method, namely VGG16-UNet, with the epoch increased to 50, and 100. The training results can be seen in Table 5, which shows the highest accuracy value of 95% and a loss of 4.95%. In this case, the accuracy value obtained is already over 90%. To test the success of the training, evaluate the trained network on the test set. After increasing the epoch by 50, the training results can be seen in Table 5, with the highest accuracy value of 96.4% and a loss of 3.62%. In this case, the accuracy value obtained is more than 90%. To test the success of the training, test the neural network on the held-out split. Then, after increasing the epoch by 100, the training results can be seen in Table 5, with the highest accuracy value of 98.03% and a loss of 1.97%. In this case, the accuracy value obtained is more than 90%. To test the success of the training, evaluate the trained network on the test set.

Table 5. Overall Results of Training and Testing VGG16-UNet 400-Image Dataset

Result Parameter	VGG16-UNet 400 Images 30 Epoch	VGG16-UNet 400 Images 50 Epoch	VGG16-UNet 400 Images 100 Epoch
Train Accuracy	0.95	0.964	0.9803
Train Loss	0.0495	0.0364	0.0197
Train Accuracy Validation	0.82	0.83	0.83
Train Loss Validation	0.18	0.17	0.16
K-Fold Cross Validation Acc	0.93	0.93	0.94
K-Fold Cross Validation Loss	0.31	0.362	0.35
Test Accuracy	0.93	0.9348	0.94
Test Loss	0.31	0.3331	0.37

Table 6. Segmentation Result

Description	VGG16-UNet 200 Images 30 Epoch	VGG16-RF 200 Images 30 Epoch	UNet 200 Images 30 Epoch
Input Image			
Prediction			
Overlay			

From the quantitative results obtained with the same dataset but with an increased number of epochs, it was found that 100 epochs yielded the best results, with a training accuracy of 98.03% and a testing accuracy of 94%. Table 6 below shows that VGG16-UNet using the 400-image dataset and 100 epochs obtained better results because the training accuracy was the highest at 98.03%, which is in line with the theory that the better the training accuracy, the better the segmentation results. The table

above shows the effect of the number of epochs on the segmentation output: the first row is the input image, the second row is the binary prediction mask (white = rust), and the third row is the overlay mask on the original image. At 30 epochs, the prediction is still fragmented with a significant false positive in the dark background area, and the rust contours on the pipe lip are not yet merged; this can be seen from the white islands that do not follow the geometry of the rust. At 50 epochs, the continuity of the

rusted area increases, leakage to the background begins to decrease, and the intermaterial boundaries are more consistent, although some thin parts at the edge of the rust are still disconnected (local under-segmentation). At 100 epochs, the rust contours become most complete and the thin texture on the inner surface of the pipe is captured more (increased recall); however, there is a slight expansion into clean areas in some parts (minor false positives), indicating the onset of overfitting, so model selection must


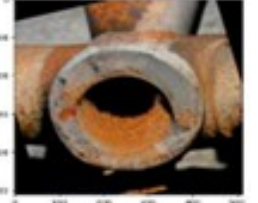

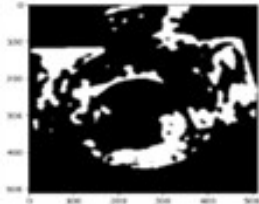
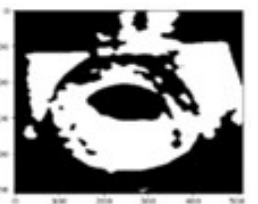


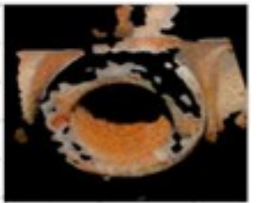
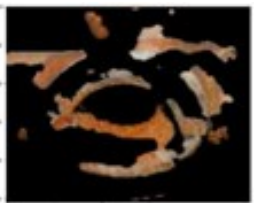
still refer to validation metrics (IoU/Dice/precision-recall) and not just visuals. Overall, this visual trend supports the interpretation that adding epochs improves convergence and boundary accuracy, with a saturation point that needs to be controlled through early stopping and validation-based threshold selection.

3.4. Overall Results of Learning & Testing using the VGG16-UNet method 600 Images Dataset

Table 7. Overall Results of Training and Testing VGG16-UNet 600 Image Dataset

Result Parameter	VGG16-UNet 600 Images 30 Epoch	VGG16-UNet 600 Images 50 Epoch	VGG16-UNet 600 Images 100 Epoch
Train Accuracy	0.955	0.9614	0.9890
Train Loss	0.045	0.0386	0.0111
Train Accuracy Validation	0.8172	0.82	0.8244
Train Loss Validation	0.1812	0.18	0.1739
K-Fold Cross Validation Acc	0.9308	0.9456	0.95
K-Fold Cross Validation Loss	0.3374	0.1344	0.1
Test Accuracy	0.9268	0.94	0.95
Test Loss	0.3037	0.16	0.1

Table 8. Segmentation Result

Description	VGG16-UNet 200 Images 30 Epoch	VGG16-RF 200 Images 30 Epoch	UNet 200 Images 30 Epoch
Input Image			
Prediction			
Overlay			

During this stage, the model was trained on 600 images and evaluated on 75 test images using the VGG16-UNet architecture. The overall results are shown in Table 7. Results of training grew more accurate with rising epochs: 95.55% (30 epochs), 96.14% (50 epochs), up to 98.90% (100 epochs). Testing showed maximum accuracy at 95% when using 100 epochs. Unlike other studies, where 20% of the test data was used, this one used only 12.5%. The results indicate that using 100 epochs provides the best results, as hypothesized by the idea that high training accuracy will yield better segmentation, as shown in Table 8.

The following figure shows the result of VGG16-UNet trained on 600 images at 30, 50, and 100 epochs; the first row in each column is the input image, the second row is the binary prediction mask (white = rust), and the third row is the overlay mask on the original image. At 30 epochs, segmentation is still disjoint, and there is leakage to the background so that the rust ring on the pipe lip has not yet appeared (lots of local false positives and false negatives). At 50 epochs, the rust area more reliably joins together to form an inner ring, leakage into the background becomes less, and the border is neater, indicating a superior precision–recall trade-off compared to 30 epochs. At 100 epochs, the rust coverage is maximally thorough, and fine details on the inner surface become more visible (recall improves).

3.5. Validating the Model Obtained After Learning

Testing of the VGG16-UNet model architecture that has been trained on metal corrosion data, with visualizations illustrated under various surface conditions and materials. The batch size used was only 14. To measure the success rate of visualizations from semantic segmentation on small objects like thin “rust,” which is not as accurate as the “background” that dominates the frame. The number of overlaps per class can be measured using the intersection-over-union (IoU) metric, also known as the Jaccard index. The Jaccard function is used to calculate image intersections, which can be binary images, labeled images, or categorized images. The Jaccard similarity coefficient is returned as a numerical scalar or numerical vector with values in the range [0, 1]. A similarity of 1 means that the segmentation in the two images is a perfect match. If the input array is: image labels, the similarity is a vector, where the first coefficient is the Jaccard index for label 1, the second coefficient is the Jaccard index for label 2, and so on. Test results for the pre-trained VGG16-UNet model. As shown in Figure 9, the image with a brightness of 150 lux will be tested.

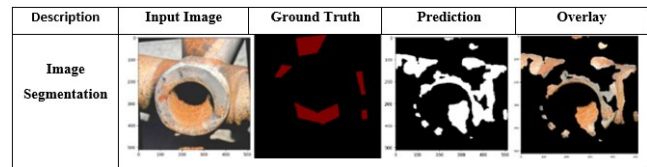


Figure 9. Comparison between Ground Truth and Prediction at brightness 150 lux

This shows the comparison between the ground truth and the prediction randomly selected by the system, which is able to segment the location of corrosion defects by segmenting the corrosion defect area using white color. This indicates that the model has been able to learn the defect patterns according to the data provided. As can be seen in Figure 10, this image will be tested with a brightness of 100 lux.

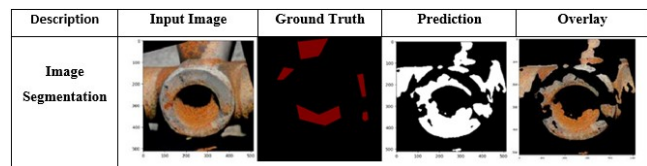


Figure 10. Comparison between Ground Truth and Prediction at brightness 100 lux

This shows the comparison between the ground truth and the prediction randomly selected by the system, which is able to segment the location of corrosion defects by segmenting the corrosion defect area using white color. This indicates that the model has been able to learn the defect patterns according to the data provided. As can be seen in Figure 11, this image will be tested with a brightness of 50 lux.

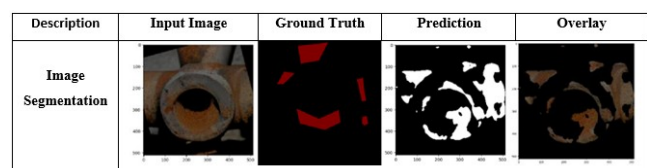


Figure 11. Comparison between Ground Truth and Prediction at brightness 50 lux

This shows that the comparison between ground truth and predictions randomly selected by the system is capable of segmenting corrosion defect locations by segmenting the corrosion defect area using white color. This indicates that the model has been able to learn the defect patterns according to the data provided. From the three simulation experiments above in various lighting conditions, it can be seen that lighting also affects the corrosion segmentation process.

3.6. System validation to detect 4 types of corrosion (Uniform Corrosion, Pitting Corrosion, Stress Corrosion Cracking, Erosion Corrosion)

This subsection also presents the results of segmentation from model testing on various types of corrosion found in materials. The types of corrosion that will be tested are pitting corrosion, uniform corrosion, stress corrosion cracking, and erosion corrosion. Figure 12 below shows the segmentation results for pitting corrosion.

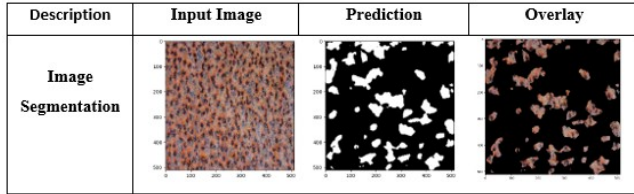


Figure 12. shows segmentation results to evaluate pitting corrosion

Figure 12 shows that the model trained using the VGG16-UNet method can perform segmentation well on pitting corrosion occurring on iron plates. Figure 13 shows the segmentation results for evaluating Uniform Corrosion.

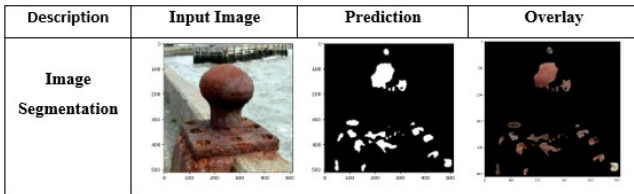


Figure 13. shows segmentation results to evaluate uniform corrosion

Figure 13 shows that the model trained using the VGG16-UNet method can perform segmentation well on stress corrosion cracking that occurs on iron plates. Figure 14 shows the segmentation results for evaluating erosion corrosion.

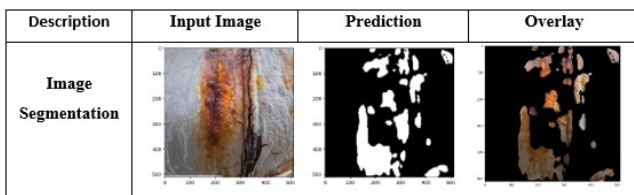


Figure 14. shows segmentation results to evaluate Stress Corrosion Cracking

Figure 14 shows that the model trained using the VGG16-UNet method can perform segmentation well on

stress corrosion cracking that occurs on iron plates. Figure 15 shows the segmentation results for evaluating erosion corrosion.

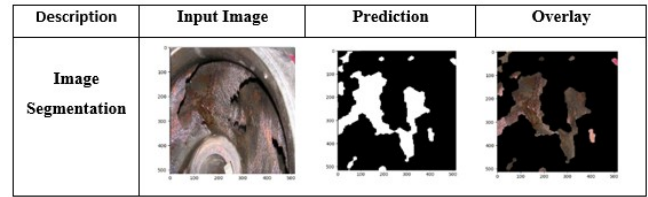


Figure 15. shows the segmentation results for evaluating erosion corrosion

Figure 15 shows that the model trained using the VGG16-UNet method can perform segmentation well on erosion corrosion occurring on iron pipes.


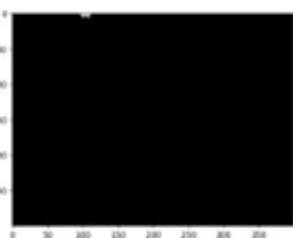

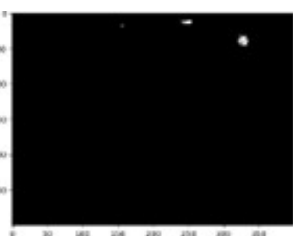

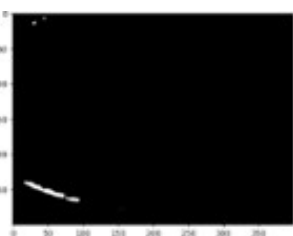

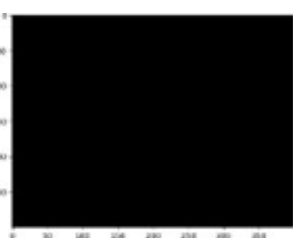
3.7. Confusion Matrix Validation Results

Figure 9 shows the results from training a corrosion detection model using the VGG16-UNet architecture. The results of the Confusion Matrix are shown in Table 9 below. This model combines VGG16 for feature extraction and UNet for precise pixel-level segmentation, making it highly suitable for identifying rust on metal surfaces. The visual outputs show strong agreement between ground truth and predictions, with clear and accurate detection of rust areas. Due to its semantic segmentation problem, it is assessed pixel-wise with corrosion as the positive class; hence, reported accuracy, precision, and recall are the ratio of correctly labeled pixels, and IoU/Jaccard and Dice are spatial overlap measures between the prediction mask and ground truth [21, 24, 26]. At 150 lux, the best-performing model exhibited high recall (the clean metal was not commonly misclassified) and high precision (corrosion was not commonly missed), and the same was true at 100 and 50 lux, demonstrating immunity to the lighting variations typical in field acquisition. After viewing the segmentation results in Figure 9, the system then calculates the confusion matrix. The confusion matrix calculation is shown in Table 9 below.

Table 9. Confusion Matrix Results at 150 lux brightness

	Formula	Result
Accuracy	$\frac{TP + TN}{TP + FP + FN + TN}$	0.9896
Precision	$\frac{TP}{TP + FP}$	0.9934
Recall	$\frac{TP}{TP + FN}$	0.994

Table 10. Results of detection validation conducted at PT STP.

	Input Image	Prediction Image System
1		 detected corrosion
2		 detected corrosion
3		 detected corrosion
4		 No corrosion detected

The model achieved high performance with an accuracy of 98.96%, precision of 99.34%, and recall of 99.4%. This indicates the model is both reliable and sensitive, correctly identifying nearly all rust areas while avoiding false detections. The use of skip connections in UNet helps preserve fine details, allowing the model to detect even small corrosion spots. In practical applications, this model can support real-time rust detection in pipelines, machines, or structures using handheld devices or drones. Its efficiency also allows deployment on lower-powered hardware. Compared to traditional approaches with color thresholding or hand-tuned features sensitive to lighting, background, and surface and requiring retuning at each

location, the DL approach with pretrained encoders + skip connections achieves improved cross-scene portability and more robust recovery of fine-scale structure (as seen in overlays and overlap measures) [19,24,31,32]; results are consistent with modern DL critique reports and standards on industrial images.

The system validation conducted at PT STP, as shown in Table 10, successfully proved that the corrosion detection system is effective in identifying various types of corrosion, including subtle corrosion that is difficult to detect with the naked eye, and can distinguish between corrosion and visual anomalies such as scratches or marks. The system is also highly sensitive to complex damage patterns,

such as fine cracks. In images that were not corroded, the system never reported false positives, which is a measure of the balance between sensitivity and accuracy. Overall, the system proved to be consistent, reliable, and accurate under varying surface conditions, and has great potential for application in automated industrial inspection, aiming to improve the efficiency and speed of damage detection. Lines 1-3 indicate true positives, meaning that corrosion was detected, while line 4 indicates a true negative, which means that the metal was not corroded. Additional testing with larger and more diverse datasets may improve the system's capabilities, including in the classification of corrosion severity. The generated pixel masks can be compressed into area/coverage indicators, linked to work orders, and tracked between visits to facilitate RCM/RBI. The efficient CPU computational profile and stability at 50, 100, and 150 lux open up opportunities for edge deployment (laptop/UAV) in digital inspection workflows [7–9].

4. Conclusions

This research demonstrates the successful development of an automated corrosion detection system using deep learning-based semantic segmentation, specifically leveraging the VGG16-UNet architecture. By training and validating models across datasets of 200, 400, and 600 images with varying epochs, the VGG16-UNet consistently outperformed UNet and VGG16-RandomForest in both training and testing accuracy. The highest performance was achieved with a dataset of 600 images and 100 epochs, reaching a training accuracy of 98.90% and a test accuracy of 95%, as well as a precision and recall exceeding 99%. These results demonstrate the model's strong generalization capability and robustness under varying lighting conditions and surface characteristics. The system was further validated with real-world industrial data, proving effective in detecting various types of corrosion while minimizing false positives. These findings highlight the potential of integrating VGG16-UNet into real-time industrial applications, offering a reliable, efficient, and accurate alternative to manual corrosion inspection processes.

References

- [1] V. Maurice and P. Marcus, "Progress in corrosion science at atomic and nanometric scales," *Progress in Materials Science*, vol. 95, pp. 132–171, 2018.
- [2] H. H. Uhlig and R. W. Revie, *Corrosion and corrosion control. An introduction to corrosion science and engineering. Third Edition.* John Wiley and Sons, Inc., New York, NY, 12 1985.
- [3] B. Popov, *Corrosion Engineering: Principles and Solved Problems.* Elsevier, 2015.
- [4] S.-H. Chen and D.-B. Perng, "Automatic optical inspection system for ic molding surface," *Journal of Intelligent Manufacturing*, vol. 27, no. 5, pp. 915–926, 2016.
- [5] C.-S. Cho, B.-M. Chung, and M.-J. Park, "Development of real-time vision-based fabric inspection system," *IEEE Transactions on Industrial Electronics*, vol. 52, no. 4, pp. 1073–1079, 2005.
- [6] C. Koch, S. G. Paal, A. Rashidi, Z. Zhu, M. König, and I. Brilakis, "Achievements and challenges in machine vision-based inspection of large concrete structures," *Advances in Structural Engineering*, vol. 17, no. 3, pp. 303–318, 2014.
- [7] Y.-F. Liu, X. Nie, J.-S. Fan, and X.-G. Liu, "Image-based crack assessment of bridge piers using unmanned aerial vehicles and three-dimensional scene reconstruction," *Computer-Aided Civil and Infrastructure Engineering*, vol. 35, no. 5, pp. 511–529, 2020.
- [8] G. Morgenthal and N. Hallermann, "Quality assessment of unmanned aerial vehicle (uav) based visual inspection of structures," *Advances in Structural Engineering*, vol. 17, no. 3, pp. 289–302, 2014.
- [9] H. Adeli and X. Jiang, *Intelligent Infrastructure: Neural Networks, Wavelets, and Chaos Theory for Intelligent Transportation Systems and Smart Structures.* CRC Press, 10 2008.
- [10] S. Agnisarman, S. Lopes, K. Chalil Madathil, K. Piratla, and A. Gramopadhye, "A survey of automation-enabled human-in-the-loop systems for infrastructure visual inspection," *Automation in Construction*, vol. 97, pp. 52–76, 2019.
- [11] I. Abdel-Qader, O. Abudayyeh, and M. E. Kelly, "Analysis of edge-detection techniques for crack identification in bridges," *Journal of Computing in Civil Engineering*, vol. 17, no. 4, pp. 255–263, 2003.
- [12] K. Vaghefi, R. C. Oats, D. K. Harris, T. T. M. Ahlborn, C. N. Brooks, K. A. Endsley, C. Roussi, R. Shuchman, J. W. Burns, and R. Dobson, "Evaluation of commercially available remote sensors for highway bridge condition assessment," *Journal of Bridge Engineering*, vol. 17, no. 6, pp. 886–895, 2012.
- [13] X. Jiang and H. Adeli, "Pseudospectra, music, and dynamic wavelet neural network for damage detection of highrise buildings," *International Journal for Numerical Methods in Engineering*, vol. 71, no. 5, pp. 606–629, 2007.
- [14] J. Uijlings, K. Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, pp. 154–171, 09 2013.
- [15] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 44, no. 1.2, pp. 206–226, 2000.

- [16] M. Somvanshi, P. Chavan, S. Tambade, and S. V. Shinde, "A review of machine learning techniques using decision tree and support vector machine," in *2016 International Conference on Computing Communication Control and automation (ICCUBE)*, pp. 1–7, 2016.
- [17] W. Amei, D. Huailin, W. Qingfeng, and L. Ling, "A survey of application-level protocol identification based on machine learning," in *Proceedings of the 2011 International Conference on Information Management, Innovation Management and Industrial Engineering - Volume 03, ICIII '11, (USA)*, p. 201–204, IEEE Computer Society, 2011.
- [18] W. Amei, D. Huailin, W. Qingfeng, and L. Ling, "A survey of application-level protocol identification based on machine learning," in *Proceedings of the 2011 International Conference on Information Management, Innovation Management and Industrial Engineering - Volume 03, ICIII '11, (USA)*, p. 201–204, IEEE Computer Society, 2011.
- [19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [20] A. Huang, L. Jiang, J. Zhang, and Q. Wang, "Attention-vgg16-unet: a novel deep learning approach for automatic segmentation of the median nerve in ultrasound images," *Quantitative Imaging in Medicine and Surgery*, vol. 12, no. 6, pp. 3138–3150, 2022.
- [21] B. Burton, W. T. Nash, and N. Birbilis, "Rustseg: Automated segmentation of corrosion using deep learning," 2022.
- [22] A. Srivastava, G. Ji, and R. K. Singh, "Application of deep-learning architecture for image analysis based corrosion detection," in *2021 Smart Technologies, Communication and Robotics (STCR)*, pp. 1–5, 2021.
- [23] P. A. Caridis, *Inspection, Repair and Maintenance of Ship Structure*. London: Witherby & Co. Ltd, 1995.
- [24] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.
- [25] M. R. Juliarsyah and A. Wikarta, "Development of corrosion segmentation using deep learning double architecture method to assist the analysis and evaluation process of corrosion inspection," *Indonesian Journal of Computer Science*, vol. 13, no. 2, pp. 2313–2325, 2024.
- [26] I. Zliobaite, "On the relation between accuracy and fairness in binary classification," 2015.
- [27] J. Brownlee, *Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch*. Machine Learning Mastery, 2016.
- [28] P. Roberge, *Corrosion Engineering: Principles and Practice*. McGraw Hill LLC, 2008.
- [29] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), (Cham), pp. 234–241, Springer International Publishing, 2015.
- [30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.
- [31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [32] S. Sookpong, S. Phimsiri, T. Tosawadi, P. Choppradit, V. Suttichaya, C. Utintu, and E. Thamwattana, "Comparison of corrosion segmentation techniques on oil and gas offshore critical assets," in *2023 20th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON 2023*, 2023 20th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON 2023, Institute of Electrical and Electronics Engineers Inc., 2023. Publisher Copyright: © 2023 IEEE.; 20th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON 2023 ; Conference date: 09-05-2023 Through 12-05-2023.