

---

## **Development of Corrosion Segmentation Using Deep Learning Double Architecture Method to Assist the Analysis and Evaluation Process of Corrosion Inspection**

**Mohammad Rizanto Juliarsyah<sup>1</sup>, Alief Wikarta<sup>2</sup>**

<sup>1</sup>[mrizantoj@gmail.com](mailto:mrizantoj@gmail.com), <sup>2</sup>[wikarta@me.its.ac.id](mailto:wikarta@me.its.ac.id)

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

---

### **Article Information**

Submitted : 23 Dec 2023

Reviewed: 12 Apr 2024

Accepted : 24 Apr 2024

---

### **Keywords**

Pump unit, coal mines, corrosion inspection, corrosion detection, computer vision, VGG16-UNET.

---

### **Abstract**

Corrosion of pump unit components often occurs in coal mines and can lead to frequent failures of some components. As a result, a corrosion inspection needs to be performed on each component to minimize the possibility of damage. Currently, manual inspection methods are used for corrosion testing but there are still metal defects in the form of corrosion that are uninspected. Therefore, this study aimed to develop corrosion segmentation using computer vision with deep learning double architecture method for detection and evaluation of metal corrosion in order to reduce the loss due to manual inspections. To produce a faster and more accurate analysis method, deep learning double architecture algorithm, namely VGG16-UNET, can be applied with the help of computer vision technology. Consequently, the use of VGG16-UNET method achieved an accuracy of 98.42%. This is in contrast with the single UNET architecture, which produced an accuracy of 92.6%. Based on these findings, it was concluded that the development of this recommended inspection made the analysis and evaluation of corrosion inspection to be quick and easy.

---

## A. Introduction

Corrosion is chemical or electrochemical deterioration of materials exposed to the environment such as metals, semiconductors, insulators, and polymers. In general, corrosion can damage industrial buildings including roofs, pipes, poles, bridges, and telecommunications towers [1] [2]. The deformation of metal in offshore industrial assets is a common problem in the industry. Therefore, corrosion detection is very important to ensure maintenance and safety. Instead of manually identifying the corrosion features, Machine Learning (ML) method can be used.

ML offers new ways to comprehend and quantify data-intensive processes in agricultural operations, particularly when combined with big data and high-performance computing. Among other fields, ML is characterized as a scientific discipline that allows machines to learn without being explicitly programmed [3]. Machine model can learn spatial aspects of corrosion, such as color or texture, thereby reducing the time required for detection. According to recent research, machine learning can be classified into three categories, namely supervised, unsupervised, and reinforcement learning [4]. In general, supervised learning methods are based on labeled data samples. This set of samples is used to describe the characteristics of the behavioral size distribution in each type of application, thereby producing a model from the data [5]. Furthermore, supervised learning can be categorized into classification and regression tasks. Classification issues arise when the output variable is categorical, such as red or blue, sick or not sick [6].

This research focuses on supervised learning, specifically using CNN as the selected approach. Typically, CNN approach uses Deep Learning (DL) methods, which allow machines to examine raw data and automatically determine the representation required for classification or detection [7]. According to research conducted by [8], there are 2 types of architecture in deep learning, namely single architecture and double architecture. Specifically, single architecture uses only 1 method, while double architecture combines 2 deep learning methods.

A comparative analysis of corrosion segmentation techniques on critical assets within the Oil and Gas Offshore industry is presented. Even though the proposed method can adequately segment defects, there are still some errors in classifying corrosion during testing [9]. Another research titled “RustSEG – Automated Segmentation of Corrosion using DL”, uses Machine Learning methods, specifically CNN approach, to classify corrosion. However, the experiments conducted achieved an accuracy rate below 90%. Additionally, research performed by [11] titled “Application of Deep-Learning Architecture for Image Analysis-based Corrosion Detection” is among the investigations that use ML methods, specifically DLCNN. The experiment was a single UNET architecture, but due to the difference in the number of layers, an accuracy of more than 90% was achieved.

Previous research explores corrosion inspection approach, using Deep Convolutional Neural Network Encoder-Decoder Architecture for segmentation tasks. In order to develop an accurate segmentation system, there is a need to perform validation. The success rate of visualization can be measured using intersection-over-union (IoU), which shows the amount of overlap per class. The system created is expected to effectively segment defects across various workspaces, including those with different patterns. Consequently, research

proposes combining several architectures to enhance accuracy and ensure precision in inspecting corrosion defects in workspaces.

## **B. Experimental Procedures Details**

Materials, equipment, and experimental procedures, as well as theoretical or calculation procedures, are detailed in this section. The experimental section provides the necessary information for reproducing the results.

### **1. Various Types of Corrosion**

Knowing the type of corrosion is very important because it is needed to find test materials to carry out experiments. Examples of corrosion types that can occur on iron plates include [12]:

- **Uniform Corrosion**

This type of corrosion causes the gradual weakening or softening of metallic material across its entire surface. It leads to uniform thinning or reduction in material thickness.

- **Pitting Corrosion**

Pitting corrosion is a localized form of corrosion characterized by the formation of microscopic pits or depressions on the metal surface. Typically, this type of corrosion is more severe than uniform corrosion as it affects only a small area of the metal surface.

- **Stress Corrosion Cracking**

Stress corrosion cracking (SCC) occurs in metallic materials due to mechanical stress and corrosive conditions. In addition, SCC can create fissures in metallic materials without significantly altering the surface appearance.

- **Erosion Corrosion**

Erosion corrosion results from a combination of mechanical erosion and corrosive chemical reactions on metallic surfaces. The process frequently occurs when metallic materials are exposed to fluid streams containing corrosive particles or chemicals.

### **2. Data Collection and Preparation**

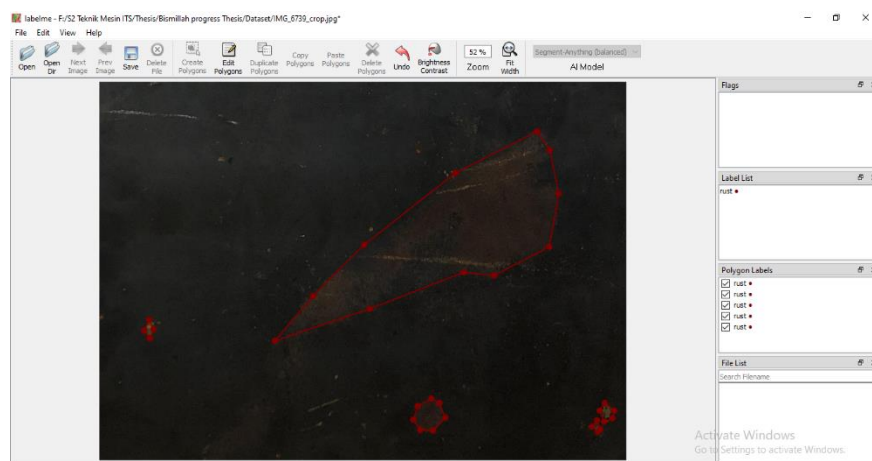
The dataset development of corrosion images was carried out by labeling each pixel. Initially, the dataset used was split into each folder, totaling 600 images. In particular, the data collected were images of iron affected by corrosion. Some samples of iron corrosion datasets were labeled as seen in **Figure 1**, with a size of 512 x 512, captured using a camera under different lighting intensities. To construct the defect detection system, the image dataset was divided into 60% and 40% training and testing data, respectively. Subsequently, the collected dataset was carefully selected to ensure optimal detection results. Images with similar patterns, sourced from multiple samples of varying sizes, went through traditional preprocessing, which includes resizing to standardize input according to the designated architecture. The required image input size was set at 512 x 512 pixels and was obtained by photographing and downloading metal images with corrosion

on the internet, So the dataset created by the researcher is not available to the public and the dataset used is purely the development of the researcher.



**Figure 1.** Dataset File for Input Image Before Labeling.

To distinguish between rust and background, area segmentation was performed by using labelme, a tool contained in Anaconda software. Subsequently, the Images that had been adapted to the input architecture were labeled for each image pixel according to the label class. Typically, the system uses 2 labels, namely rust and background, to determine the label class for each pixel. The background class is colored black while the rust is red as shown in **Figure 2**. In order to make segmentation easier at this stage, polygon tools can be used to follow the shape of the corrosion.



**Figure 2.** Labelling Input Dataset Using labelme.

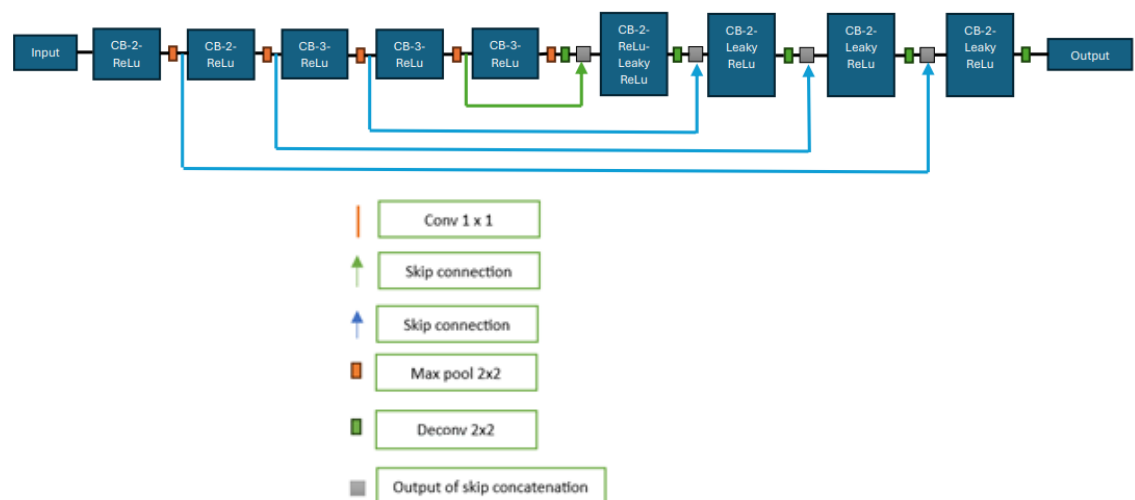
After all the data is labeled according to their respective classes, it can be saved. However, the saved file is in .json format and can be converted into a folder containing images in PNG format via the command prompt. Following the conversion process, a folder appears bearing the designated name specified in the

command prompt. The original image is contained in the folder along with their labels and visualized labels.

### 3. Selection of specific architecture

- VGG-16-UNET Architecture

VGG16 network has 13 convolutions, 5 unions, and 3 fully connected layers at the end of the network. The network has a homogeneous architecture that only performs  $3 \times 3$  convolutions and a maximum of  $2 \times 2$  union throughout its structure. For the improved VGG16-UNet shown in **Figure 3**, the last 3 fully connected layers of VGG16 are substituted with an architecture resembling the decoding segment of U-Net. In general, the segment forms an extension path with a convolution layer and an upsampling layer. Meanwhile, VGG16 without the last 3 fully connected layers is retained as a contraction path. Furthermore, this research introduces 3 additional modifications, and the original rectified linear unit (ReLU) functions in the 7 convolution layers (the last four convolution blocks in **Figure 3**) were replaced with Leaky ReLU ( $\alpha = 0.1$ ). A total of 4 jump connections namely 3 merges and 1 sum, were used to combine feature maps from different modules in both the contraction and expansion paths.

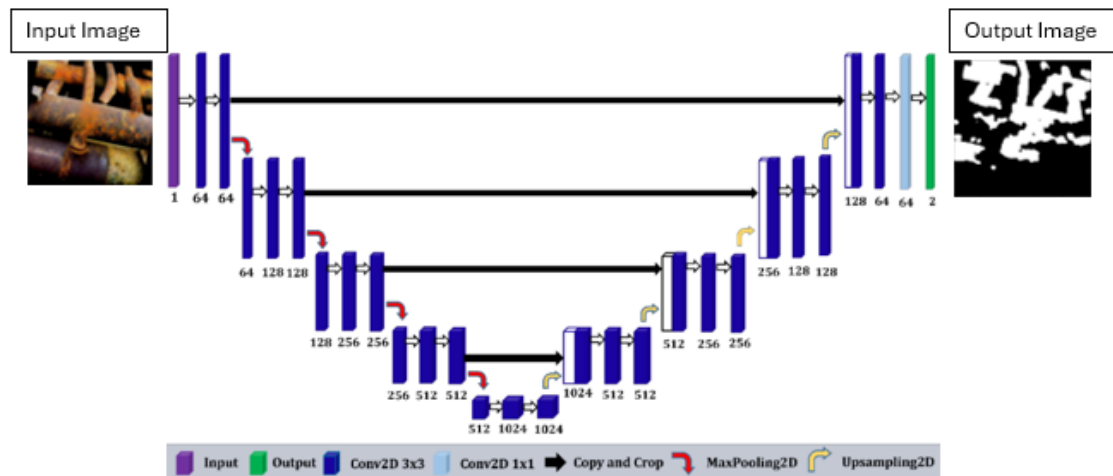


**Figure 3.** VGG16-UNET Architecture Model.

- UNET Architecture

Image segmentation is a series of image processing operations performed by labeling each pixel of the image. In this context, the image is categorized into several parts such that the pixels in one part have the same characteristics. In this corrosion research, the image is divided into two parts, namely the metal affected by corrosion and the background. Specifically, the metal corrosion segmentation process is carried out by applying U-Net architecture, which also has two paths, namely the encoder and decoder. The encoder path is used to capture feature information from the input image and reduce the dimensions of the input image size. Meanwhile, the decoder path is used to capture feature information from the encoder results and as an output for the

segmentation results. The U-Net architecture for corrosion segmentation proposed in this study can be seen in **Figure 4**.



**Figure 4.** UNET Architecture Model

**Figure 4** shows the U-Net architecture, consisting of two paths, namely the encoder on the left side and the decoder on the right side. The encoder path starts with a double  $3 \times 3$  convolution layer process, followed by the ReLU activation function, resulting in 64 feature maps. Subsequently, a  $2 \times 2$  max-pooling process was performed. In the encoder path, this research uses four convolution blocks, where the number of feature maps increases in each convolution block by 2. Subsequently, the process moved to the fifth block, which represents the connection between the encoder path and the decoder path. The stages are the same as the first block, except that the max-pooling process does not need to be followed. Furthermore, the decoder path begins with a  $2 \times 2$  upsampling process and then continues with the same process as the first block without the max-pooling process. In the decoder path, this research uses four convolution blocks thereby reducing the number of feature maps for each block. The final 2 stages in this path are the  $1 \times 1$  convolution layer process and the sigmoid activation function, which is multiplied until the number of feature maps matches the original number used to produce a segmented image.

#### 4. Hyperparameter tuning

An important stage in deep learning is hyperparameter tweaking. Recent research showed that instead of introducing new learning paradigms, cutting-edge image classification benchmarks can be improved through appropriate tuning of existing methods. Generally, hyperparameters can be categorized into three types, and the first is model hyperparameters, which define the essential construct of a neural network architecture. This type includes parameters such as filter size, pooling, stride, and padding. For example, in the UNET architecture, filter size ranges from 16 to 128 for each layer, with one stride and the same padding. Typically, Max-pooling is a technique used for determining the feature map patch with the highest pixel value. The second is optimizer hyperparameters, a technique or procedure that changes neural network features, such as learning rate and

weights in order to minimize losses. It is crucial to be aware that optimizers contained gradient descent, stochastic gradient descent (SGD), Adam, RMSprop, and Adadelta. The Adaptive Moment Estimation (Adam) algorithm is a popular deep neural network training strategy in several machine learning systems [13]. Moreover, the third category is the data hyperparameter which is generally used when there is insufficient or variance in the data. Consequently, techniques such as cropping, resizing, binarization, and data augmentation can be used to improve the amount of data available.


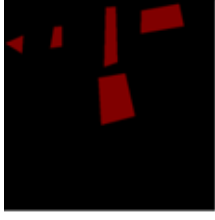
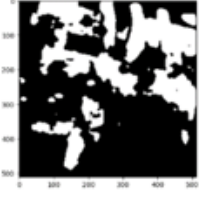
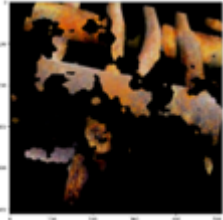

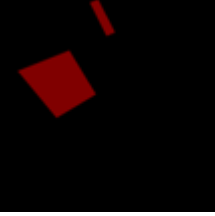
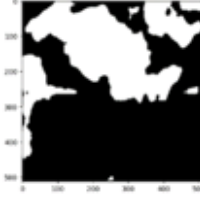
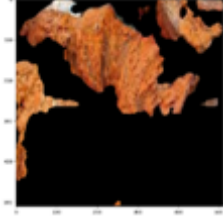
(None, 224, 224, 3) (None, 224, 224, 64) Model: "VGG16-Unet"			
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 224, 224, 3, 0)		input_1[0][0]
block1_conv1 (Conv2D)	(None, 224, 224, 64, 1792)		input_1[0][0]
block1_conv2 (Conv2D)	(None, 224, 224, 64, 36928)		block1_conv1[0][0]
block1_pool1 (MaxPooling2D)	(None, 112, 112, 64, 0)		block1_conv2[0][0]
block2_conv1 (Conv2D)	(None, 112, 112, 12, 73856)		block1_pool1[0][0]
block2_conv2 (Conv2D)	(None, 112, 112, 12, 147584)		block2_conv1[0][0]
block2_pool1 (MaxPooling2D)	(None, 56, 56, 128, 0)		block2_conv2[0][0]
block3_conv1 (Conv2D)	(None, 56, 56, 256, 295168)		block2_pool1[0][0]
block3_conv2 (Conv2D)	(None, 56, 56, 256, 590080)		block3_conv1[0][0]
block3_conv3 (Conv2D)	(None, 56, 56, 256, 590080)		block3_conv2[0][0]
block3_pool1 (MaxPooling2D)	(None, 28, 28, 256, 0)		block3_conv3[0][0]
block4_conv1 (Conv2D)	(None, 28, 28, 512, 1180160)		block3_pool1[0][0]
block4_conv2 (Conv2D)	(None, 28, 28, 512, 2350080)		block4_conv1[0][0]
block4_conv3 (Conv2D)	(None, 28, 28, 512, 2350080)		block4_conv2[0][0]
block4_pool1 (MaxPooling2D)	(None, 14, 14, 512, 0)		block4_conv3[0][0]
block5_conv1 (Conv2D)	(None, 14, 14, 512, 2350080)		block4_pool1[0][0]
block5_conv2 (Conv2D)	(None, 14, 14, 512, 2350080)		block5_conv1[0][0]
block5_conv3 (Conv2D)	(None, 14, 14, 512, 2350080)		block5_conv2[0][0]
conv2d_transpose (Conv2DTranspose)	(None, 28, 28, 512, 1840080)		block5_conv3[0][0]
concatenate (Concatenate)	(None, 28, 28, 1024, 0)		conv2d_transpose[0][0], block4_conv3[0][0]
conv2d_transpose_3 (Conv2DTranspose)	(None, 224, 224, 64, 32832)		activation_5[0][0]
concatenate_3 (Concatenate)	(None, 224, 224, 12, 0)		conv2d_transpose_3[0][0], block1_conv2[0][0]
conv2d_6 (Conv2D)	(None, 224, 224, 64, 73792)		concatenate_3[0][0]
batch_normalization_6 (Batch Normalization)	(None, 224, 224, 64, 256)		conv2d_6[0][0]
activation_6 (Activation)	(None, 224, 224, 64, 0)		batch_normalization_6[0][0]
conv2d_7 (Conv2D)	(None, 224, 224, 64, 36928)		activation_6[0][0]
batch_normalization_7 (Batch Normalization)	(None, 224, 224, 64, 256)		conv2d_7[0][0]
activation_7 (Activation)	(None, 224, 224, 64, 0)		batch_normalization_7[0][0]
conv2d_8 (Conv2D)	(None, 224, 224, 1, 65)		activation_7[0][0]
Total params: 25,862,337 Trainable params: 25,858,497 Non-trainable params: 3,840			

**Figure 5.** Model Summary with the Number of Parameters at Each Layer

## 5. Model training and validation

The input images and their associated segmentation masks or ground truth, are fed into a deep learning-based VGG16-UNet implementation in order to train the network. Specifically, Jupiter Notebook was used to train 600 photos with segmented masks, and a batch size of 14 and 100 epochs was specified. A batch is a hyperparameter that determines the number of data points processed before updating the model's internal parameters. Meanwhile, the epochs specify the number of times the learning algorithm runs through the training data [14]. During the training phase, the model takes approximately 9 hours to achieve satisfactory results in terms of prediction accuracy and validation. The model demonstrates effective training as it achieved 98.42 % accuracy, with a loss of 0.0158 %, as well as 98.34 % validation accuracy, with a 0.0165 % loss. It should be acknowledged that the training and validation data showed similar accuracy, thereby indicating excellent performance. **Figure 6** showed the process of data testing and validation.



No.	Original	Ground Truth	Prediction	Overlay
1				
2				

**Figure 6.** Examples of Validation Data Results.

#### 6. Confusion matrix

The confusion matrix calculated in segmentation is slightly different from object classification segmentation, where in segmentation the confusion matrix is calculated per pixel, where later per pixel that has detected corrosion will be included in the calculation of true positive corrosion, the TP, TN, FP, FN table will be shown in **Figure 7** below this.

		True Class	
Predicated Class		TP	FP Type Error I
		FN Type Error II	TN

**Figure 7.** Confusion Matrix



### C. Result and Discussion

A personally collected dataset was assembled based on documented sources. The dataset consists of various metal images with different conditions, types, and shapes of metal. These images are standardized to a size of 512 x 512-pixel size to match the pre-prepared input image. The datasets used include 200, 400, and 600 RGB images, with 60% and 40 % for training and testing, respectively. An implementation of VGG16-UNET, compatible with Anaconda and Jupiter Notebook GPU, was developed using publicly available optimization library functions. To facilitate a better understanding of the system architecture, the write-up is segmented into several parts. Before segmenting corrosion on metal, the dataset was classified based on the training data and the process included.

- Preparation of a dataset in the form of images with various metals subjected to corrosion processes.
- Determination of the number of classes and labels, achieved using Labellme in the Anaconda application.
- Semantic segmentation to assign pixel label classes, using VGG16-UNET.

In the course of the training, most learning methods use SGD to initialize weights appropriately and adjust learning rates accordingly [14]. To maximize GPU utilization and minimize CPU memory transfers, mini-batches were used [15]. Optimization learning was performed using 30, 50, and 100 epochs with 41 iterations per epoch. The encoder-decoder is trained to move the cross-entropy loss label through mini-batches. Some examples of differences in each class in the training set include the background associated with corrosion dataset. This results in different weight derivations based on the actual class, necessitating class balancing. The median offset of frequencies whose weights are sorted by class in the loss function is used for this purpose. To compare quantitative performance, three commonly used performance measures include 1) Global Accuracy, which measures the percentage of correct pixels in the dataset, 2) Average accuracy per pixel, and 3) Average Intersection over Union (mIoU) for all classes [16]. The IoU metric, also known as the Jaccard Index is used to assess interaction, while mIoU metric is stricter than average age as it penalizes false positive predictions [17]. Despite this condition, mIoU may not always be appropriate qualitatively, as it does not restrict boundaries. Measuring the contour score in semantic segmentation is performed by evaluating the F1 measure, which comprises the precision and recall values between the prediction and ground truth boundaries. Subsequently, the average of all tests denoted by F1 size limit (BF) was calculated together with the average F1 size of the image. The implementation, conducted using Jupiter Notebook, uses 512 x 512-pixel image input and NVIDIA GeForce MX110.

#### Training Model VGG16-UNET & UNET

The training process uses Jupiter Notebook and Python, considering 1400 images. Specifically, the dataset consists of 200, 400, and 600 different images for the training set, 150 for the validation set, and 50 for the test set process. The training was carried out for 30, 50, and 100 epochs with the batch size set to 14, and the input image dimension configured to 512 x 512. The outcome of the training process can be seen in **Table 1**.

**Table 1.** Training Result VGG16-UNET

Number of pictures	Epoch	Accuracy	Loss
200	30	87,5%	12,5%
200	50	89,26%	10,66%
200	100	95,34%	4,64%
400	30	88%	12%
400	50	92,4%	7,6%
400	100	96,76%	3,24%
600	30	89,51%	10,42%
600	50	92,71%	7,24%
600	100	98,42%	1,58%

Table 1 shows the accuracy and loss values for training with VGG16-UNET using datasets of 200, 400, and 600 images across different epochs, namely 30, 50, and 100. The accuracy result for each dataset and epoch configuration includes 87.5%, 89.26%, and 95.34% for 200 images at 30, 50, and 100 epochs, respectively. For the 400 image dataset, accuracies were 88%, 92.4%, and 96.76% for 30, 50, and 100 epochs, respectively. Similarly, for the 600 image dataset, the accuracies include 89.51%, 92.71%, and 98.42% for 30, 50 and 100 epochs, respectively. Therefore, VGG16-UNET method produced higher accuracy compared to RustSEG [10] and UNET [9], as shown in **Table 2**.

**Table 2.** Training Result RustSEG [10] & UNET [9]

Method	Number of pictures	Accuracy	Loss
RustSEG	1200	77.83%	22.17%
UNET	423	67.59%	32.41%

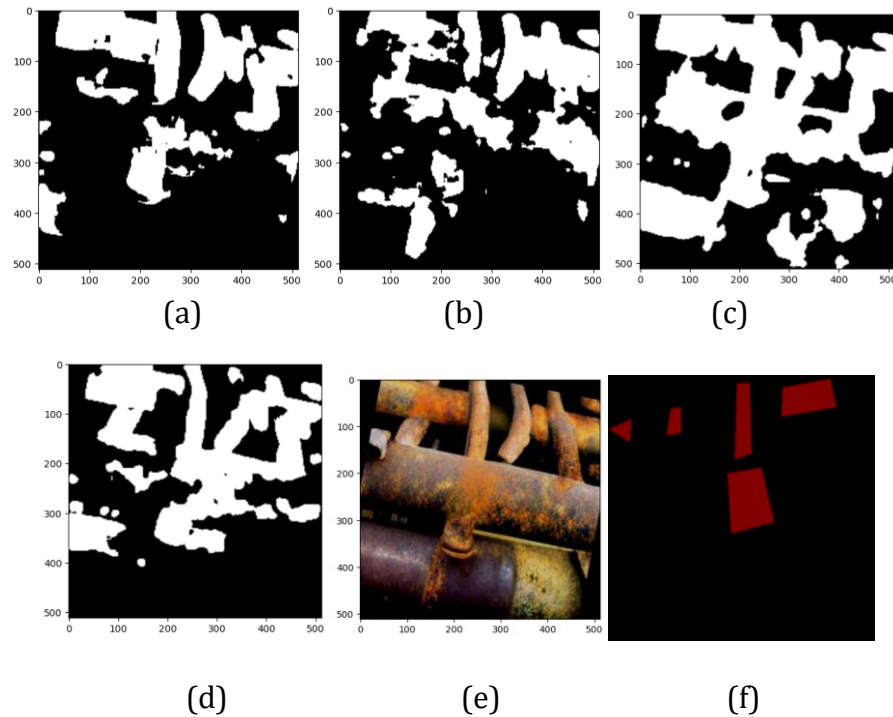
Based on the reference in Table 2, the two methods with a single architecture did not achieve less than 90% accuracy for validating the learning process of single UNET architecture using the image dataset consisting of 600 images in 100 epochs. The results in **Table 3** show that the accuracy of 600 images in 100 epochs is 92.6%.

**Table 3.** Training Result UNET

Method	Number of pictures	Accuracy	Loss
UNET	600	92.6%	7.34%

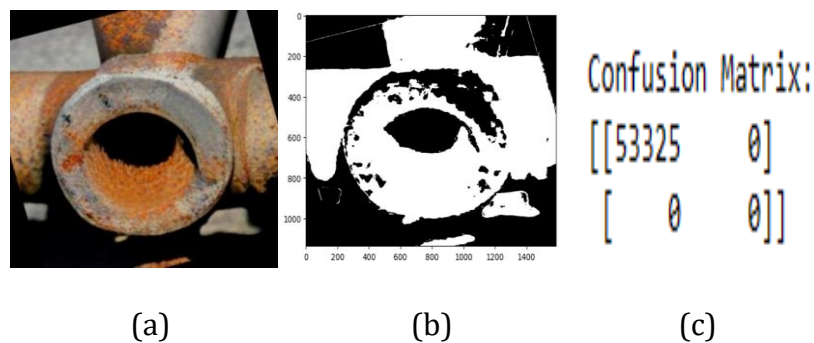
### Testing Image Segmentation Results VGG16-UNET & UNET

**Figure 7** shows the image segmentation results of VGG16-UNET model. The figure shows that VGG16-UNET model used has successfully segmented corroded iron objects.



**Figure 8.** Model Validation Results (a) 200 Image Datasets (b) 400 Image Datasets (c) 600 Image Datasets (d) UNET 600 Image Dataset (e) Input Image (f) Groundtruth.

In addition to the test as shown in Figure 8 above, a confusion matrix will be used to ensure that the test carried out has correctly detected corrosion or not, the results of the confusion matrix test will be shown in Figure 9 below.



**Figure 9.** Model Validation Results (a) Input Image Datasets (b) Segmentation Image (c) Confusion Matrix.

#### D. Conclusion

In conclusion, the research results using VGG16-UNET showed its capability in metal segmentation. Specifically, the post-training results showed an accuracy of 95.34% for 100 epochs with 200 images, 96.76% for 400 images, and 98.42% for

600 images. This showed that the model produced a high level of accuracy in object detection. Comparatively, VGG16-UNET obtained more accurate results compared to previous metal segmentation methods, such as RustSEG and UNET. The validation of UNET single architecture learning process, using a dataset comprising 600 images produced an accuracy of 92.60%. With the implementation of the double-architecture deep learning method, the metal segmentation was better than the single architecture. This enhancement was crucial as high accuracy translated to good segmentation results. Finally, by applying this method, it was expected that manufacturing industry losses resulting from manual inspection errors would decrease. In addition, the method offered an improved metal segmentation system, because accuracy directly influenced segmentation quality.

### E. Acknowledgment

The authors thankfully acknowledge the financial support provided by the Center for Education Financial Service (PUSLAPDIK) and Indonesia Endowment Fund for Education (BPPT) and also the experts for making valuable suggestions to improve the quality and content of this research.

### F. References

- [1] Maurice, V., & Marcus, P. (2018). Progress in corrosion science at atomic and nanometric scales. *Progress in Materials Science*, 95, 132–171. <https://doi.org/10.1016/j.pmatsci.2018.03.001>
- [2] Uhlig, H. H., & Revie, R. W. (1985). Corrosion and corrosion control. An introduction to corrosion science and engineering. Third Edition.
- [3] Samuel, A. L. (2000). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 44(1.2), 206–226. doi:10.1147/rd.441.0206
- [4] Somvanshi, M., & Chavan, P. (2016). A review of machine learning techniques using decision tree and support vector machine. 2016 International Conference on Computing Communication Control and Automation (ICCUBE), 1–7. <https://doi.org/10.1109/ICCUBE.2016.7860040>
- [5] Amei, W., Huailin, D., Qingfeng, W., & Ling, L. (2011). A survey of application-level protocol identification based on machine learning. 2011 International Conference on Information Management, Innovation Management and Industrial Engineering, 3, 201–204.
- [6] Brownlee, J. (2016). Master Machine Learning Algorithms: discover how they work and implement them from scratch. Jason Brownlee.
- [7] Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444. <https://doi.org/10.1038/nature14539>
- [8] Huang, A., Jiang, L., Zhang, J., & Wang, Q. (2022). Attention-VGG16-UNet: a novel deep learning approach for automatic segmentation of the median nerve in ultrasound images. *Quantitative Imaging in Medicine and Surgery*, 12(6), 3138–3150. <https://doi.org/10.21037/qims-21-1074>
- [9] Sookpong, S., Phimsiri, S., Tosawadi, T., Choppradit, P., Suttichaya, V., Utintu, C., & Thamwiwatthana, E. (2023). Comparison of Corrosion Segmentation Techniques on Oil and Gas Offshore Critical Assets. 2023 20th International Conference on Electrical Engineering/Electronics, Computer,

- Telecommunications and Information Technology, ECTI-CON 2023, 1–5.  
<https://doi.org/10.1109/ECTI-CON58255.2023.10153134>
- [10] Burton, B., Nash, W. T., & Birbilis, N. (2022). RustSEG: Automated segmentation of corrosion using deep learning, 1–30.
- [11] Srivastava, A., Ji, G., & Singh, R. K. (2021). Application of Deep-Learning Architecture for Image Analysis based Corrosion Detection. Proceedings - 1st International Conference on Smart Technologies Communication and Robotics, STCR 2021, (October), 1–5.  
<https://doi.org/10.1109/STCR51658.2021.9588887>
- [12] Caridis, P.A, B.Sc, M.Sc, Ph.D MRINA. C. Eng. (1995). Inspection, Repair and Maintenance of Ship Structure, Witherby & CO. LTD, London.
- [13] Bock, S., & Weis, M. (2019). A Proof of Local Convergence for the Adam Optimizer. Proceedings of the International Joint Conference on Neural Networks, 2019-July(July 2019).  
<https://doi.org/10.1109/IJCNN.2019.8852239>
- [14] Brownlee, J. (2018). What is the Difference Between a Batch and an Epoch in a Neural Network? Machine Learning Mastery, July, 3–4.
- [15] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. Proceedings of COMPSTAT 2010 - 19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers, 177–186.  
[https://doi.org/10.1007/978-3-7908-2604-3\\_16](https://doi.org/10.1007/978-3-7908-2604-3_16)
- [16] Behnke, S. (2014). Fast Semantic Segmentation of RGB-D Scenes with GPU-Accelerated Deep. August, 0–6. <https://doi.org/10.1007/978-3-319-11206-0>
- [17] Eigen, D., & Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. Proceedings of the IEEE International Conference on Computer Vision, 2015 International Conference on Computer Vision, ICCV 2015, 2650–2658.  
<https://doi.org/10.1109/ICCV.2015.304>