www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 16(4), Sep-Dec 2025, 441-458. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2025.v16i4.1037

Binary Classification of Corrosion in Images through the LibAUC Library

Georgina Castillo-Valdez[1], Manuel Paz-Robles[2], Ricardo Arath Díaz-Parra[3], David Lerma-Ledezma[1], Fausto Balderas-Jaramillo[2], Claudia Gomez-Santillan[2]
[1] Universidad Politécnica de Altamira, México
[2] Tecnológico Nacional de México, Instituto Tecnológico de Ciudad Madero, México
[3] Benemérita Universidad Autónoma de Puebla, México
georgina.castillo@upalt.edu.mx, g97071322@cdmadero.tecnm.mx, ricardo.diazp@alumno.buap.mx, david.lerma@upalt.edu.mx, fausto.bj@cdmadero.tecnm.mx, claudia.gs@cdmadero.tecnm.mx

Abstract. Corrosion causes severe damage to metal surfaces in various environments, creating significant economic and safety risks. This study introduces a methodology for developing optimal binary image classifiers by leveraging LibAUC, a library designed to maximize the AUC metric. Three advanced pretrained models, namely DenseNet121, ResNet50, and EfficientNetV2s, were fine-tuned and evaluated on a corrosion image dataset. Data augmentation techniques were employed to enhance model generalization. Performance was assessed using AUC, accuracy, precision, recall, specificity, and F1 score. DenseNet121 achieved an exceptional AUC value of 0.996. More importantly, its strong generalization capability was confirmed by a Wilcoxon test p value of 0.431. This work demonstrates that the LibAUC library effectively improves the performance of pre-trained models for corrosion detection.

Keywords: AUC, binary classification, corrosion, deep learning, image analysis, LibAUC.

Article Info
Received February 25, 2025
Accepted July 6, 2025

1 Introduction

Corrosion gradually deteriorates metallic structures through chemical-physical reactions between the material and its environment (Soares et al., 2023). This deterioration widely impacts key sectors such as infrastructure, transportation, manufacturing, and energy production. According to the National Association of Corrosion Engineers (NACE), these impacts create an annual economic loss of $2.5 trillion (Fondevik et al., 2020). The consequences, however, extend beyond economic losses to include catastrophic damage to humans and the environment. Corrosion can severely weaken structures like bridges, pipelines, tanks, poles, towers, and machinery (Casas et al., 2024). In fact, 20% of all manufactured steel is used to replace corroded equipment, structures, or installations (Soares et al., 2023).

Traditionally, inspectors have performed corrosion inspections manually in the field, but this approach is labor-intensive and time-consuming. These manual techniques are often inefficient, and inspectors cannot access some locations. Now, the development of deep learning has revolutionized image analysis, providing unparalleled possibilities for identifying corrosion accurately and automatically (Suresh Kumar et al., 2023).

While Convolutional Neural Networks (CNNs) excel at classifying patterns in images, most studies evaluate their performance using threshold-dependent metrics such as accuracy, F1-score, precision, and recall. These metrics require converting the predicted probabilities of a model into discrete class labels based on a fixed decision threshold, which means their values are highly sensitive to the specific threshold chosen. This inherent dependency represents a significant limitation for their use in robust performance evaluation. In contrast, the Area Under the ROC Curve (AUC) offers a threshold-independent alternative. Rather than measuring performance at a single arbitrary point, the AUC assesses the fundamental ability of a classifier to discriminate between classes across all possible decision thresholds (Fawcett, 2006).

This work proposes using the LibAUC library, described in (Yuan et al., 2023). The authors report that their approach outperformed other similar tools in experiments involving tasks such as classifying unbalanced data, learning to rank, and contrastive representation learning. The decision to use LibAUC was not motivated by the dataset; the dataset is balanced. Instead, the motivation was the robustness of the AUC metric itself. The AUC metric reliably evaluates the discriminatory power of a classifier, regardless of the decision threshold. Rather than optimizing for a different metric like accuracy or precision, the goal was to train models that achieve the best possible separation between two classes. This was done using a loss function that directly maximizes the AUC value.

Applying the LibAUC library for binary corrosion classification represents a novel approach. Existing methods typically use generic loss functions like cross-entropy, but LibAUC uses a specialized loss function and optimizer to maximize the AUC value directly. Although this library has succeeded in several other tasks, researchers have not yet explored its use for fine-tuning pretrained models.

This study simplifies the complex nature of corrosion into a binary classification problem (corrosion or no corrosion), even though the condition manifests in various types and degrees. This approach is practical for two main reasons. First, it serves as the initial step to validate the performance of the LibAUC library in this specific area. Second, early corrosion detection, a simple yes-or-no result, is sufficient to trigger a technical response. This work provides three main contributions. First, we apply the LibAUC library to the task of corrosion classification. Second, we comparatively evaluate three pretrained CNNs (DenseNet121, ResNet50, and EfficientNetV2s), each optimized for AUC. Finally, we analyze the performance of these models using five metrics: AUC, accuracy, precision, recall, and specificity.

This document is structured as follows. Section 1 establishes the context and motivation for this study. Section 2 reviews related work in the field. Section 3 provides the necessary theoretical background. Section 4 then outlines the proposed methodology. The experimental setup and a discussion of the corresponding results are presented in Section 5. Finally, Section 6 concludes the paper by summarizing the key findings and their implications.

2 Related work

Corrosion inspection is an intensive and time-consuming task, often hampered by inefficient techniques and environments inaccessible to technicians. However, the development of deep learning has revolutionized image analysis, enabling the automatic and accurate identification of corrosion. Consequently, several studies have investigated the application of CNNs for image-based corrosion detection. This section reviews six notable examples of this approach.

R. Zhao et al. (2017) employed a CNN for the binary classification of material images to detect corrosion. Their approach utilized transfer learning, specifically through the fine-tuning of multiple EfficientNetV2 model variants. Performance was evaluated based on accuracy, precision, recall, F1-score, and AUC. The top-performing configurations were the EfficientNetV2B0 and EfficientNetV2s models, with fine-tuning rates of 20% and 15%, respectively. These models achieved a maximum testing accuracy of 0.9176, an AUC of 0.97, and precision, recall, and F1-score values all exceeding 0.9.

Vriesman et al. (2019) developed a Texture Convolutional Neural Network (TCNN) for the automated visual inspection of corrosion on thermoelectric metallic pipelines. Their experimental results demonstrated that the proposed TCNN achieved a high accuracy of 99.20% in classifying three distinct levels of corrosion severity: non-defective, medium, and aggravated. In a related study, Bastian et al. (2019) proposed a custom computer vision approach based on a CNN to detect corrosion in images of water, oil, and gas pipelines. Their model classifies corrosion into four severity levels (high, medium, low, or none) and achieved a reported accuracy of 98.8%.

Holm et al. (2020) conducted a study to compare the performance of four different convolutional neural networks: AlexNet, GoogLeNet, ResNet-50, and VGG-16. Their goal was to find the best model for automatically classifying corrosion-related damage on bridge coatings from images. To measure performance, they used recall, precision, accuracy, and the F1-score. They reported that AlexNet achieved the highest accuracy for damage detection at 99.14%.

In one study, Srivastava et al. (2021) used a deep learning UNET architecture with eight layers to analyze and classify corroded regions in images, achieving an accuracy rate of 96.87%. In a different approach, Nie (2023) focused on classifying six specific types of corrosion: Crazing, Inclusion, Patches, Pitted, Rolled-in, and Scratches. The author built a small, conventional CNN and compared it with five other architectures (VGG16, ResNet50, MobileNetV2, InceptionResNetV2, and Xception). The experiments showed that the custom CNN, InceptionResNetV2, and Xception models obtained an accuracy rate of more than 90%.

In one comparative study, Tariber et al. (2023) evaluated four models for classifying submarine pipelines into three categories: corroded, cracked, and undamaged. The models evaluated were EfficientNetB0, MobileNetV2, ResNet50, and VGG-16. Of these, VGG-16 achieved the highest accuracy at 99.04%. Finally, in the most recent study considered, Farooqui et al. (2024) proposed a custom convolutional neural network (CNN) for binary corrosion classification and employed two pretrained models, YOLOv8 and EfficientNetB0. They reported that their custom CNN and EfficientNetB0 both achieved perfect scores of 100% across accuracy, precision, recall, and F1 metrics. In contrast, YOLOv8 achieved scores of 95% for accuracy, 100% for precision, 90% for recall, and 94.74% for the F1 score.

Most of the reviewed studies assessed model performance using threshold-dependent metrics like accuracy, precision, recall, and the F1-score. Calculating these metrics requires converting the predicted probabilities of a model into discrete class labels based on a fixed decision threshold. As a result, the reported values can vary depending on the threshold chosen. Of the works discussed, only R. Zhao et al. (2017) reported AUC. This metric is threshold-independent and evaluates the fundamental power of a classifier to discriminate between classes.

This work proposes to evaluate model performance using the AUC metric, which offers a more robust measure of the discriminatory power of a model. To maximize AUC values during training, we utilized the LibAUC library, which is specifically designed for optimizing this metric. In addition to AUC, we employed a suite of complementary threshold-dependent metrics, such as accuracy, precision, recall, F1 score, and specificity, to provide a comprehensive performance evaluation.

3 Background

This section provides the theoretical background for the key concepts and tools used in this work. It begins with an overview of CNNs, and the transfer learning technique used to adapt them for specialized tasks. The discussion then covers the LibAUC library, the specific framework employed for model optimization. Finally, the section details the metrics for performance evaluation, including the ROC curve, the Area Under the ROC Curve (AUC), and the confusion matrix with its derived metrics.

3.1 Convolutional Neural Network

Artificial Neural Networks (ANNs) are computational systems inspired by the biological structure and function of the human brain. These networks consist of numerous interconnected processing units, or neurons, which work in parallel to recognize complex patterns and relationships within data through iterative learning processes (Agatonovic-Kustrin & Beresford, 2000; O'Shea & Nash, 2015). A specialized and highly effective type of ANN for processing visual information is the CNN (Anzures-Garcia et al., 2024). The standard CNN architecture is composed of a sequence of distinct layers, each performing a specific operation on the input data. These include convolutional layers for feature extraction, pooling layers for dimensionality reduction, activation functions for introducing non-linearity, and fully connected layers for final classification (Indolia et al., 2018). Mathematical formulations of CNN are explained in (Koushik, 2016) . The starting point is the input signal x, and the next layer x_j is computed as

$$x_j = \rho W_j x_{j-1} \tag{1}$$

In Equation (1), W_j represents a linear operator, and ρ is a nonlinearity. In CNNs, W_j is almost always a convolution, and ρ is a non-linear function such as max $(x, 0)$ or $\frac{1}{1+exp^{-x}}$. To understand what W_j does, we can interpret it as a set of convolutional filters. A convolutional layer produces filter maps, and each of these maps can be represented as a sum of convolutions of the previous layer, as is written in Equation (2).

$$x_j(u, k_j) = \rho\left(\sum_k \left(x_{j-1}(.,k) * W_{j,k_j}(.,k)\right)(u)\right) \tag{2}$$

Where $*$ represents the convolution operator:

$$(f * g)(x) = \sum_{u=-\infty}^{\infty} f(u)g(x-u) \tag{3}$$

3.2 Transfer learning

Deep learning models, including CNNs, are typically trained using one of two primary methodologies: complete training or transfer learning (Das et al., 2024). Complete training involves initializing all model weights randomly and training the entire architecture from scratch on a specific dataset. In contrast, transfer learning leverages the feature extraction capabilities of a model previously trained on a large, general-purpose dataset, such as ImageNet. This work employs a transfer learning approach, utilizing a technique known as fine-tuning. The fine-tuning process consists of two distinct phases. First, the feature extraction layers of the pretrained model are frozen, and only the classifier layers are trained on the target dataset for a set number of epochs. Subsequently, the feature extraction layers are unfrozen, and the entire model undergoes further training. This two-stage process allows the model to adapt its generalized features to the specific nuances of the new task while mitigating the risk of overfitting.

Among the architectures selected for this study is DenseNet121, a member of the Densely Connected Convolutional Networks family introduced by Huang et al. (2016). This model, pretrained on the ImageNet dataset (Deng et al., 2009), is characterized by its dense connectivity pattern. Its core innovation lies in its dense blocks, where each layer receives the feature maps of all preceding layers as input and passes its own feature maps to all subsequent layers, as illustrated in Fig. 1. This connectivity pattern encourages feature reuse, reduces the vanishing gradient problem, and improves parameter efficiency. The key components of a dense block are described in Table 1, while the standard architecture of DenseNet121 for ImageNet multiclass classification is detailed in Table 2. For this investigation, the final classification layer of the DenseNet121 model was modified to perform binary classification.

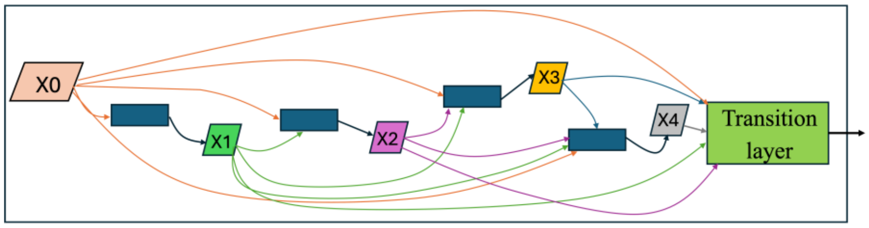


Fig. 1. Example of a dense block with five layers. Each layer receives information from all previous layers, as was explained in (Huang et al., 2016).

ResNet50 is a deep convolutional architecture from the Residual Networks (ResNets) family, introduced by He et al. (2015). Its defining innovation is the use of Deep Residual Learning (DRL), a framework designed to overcome the vanishing gradient problem in very deep networks. The core principle of DRL is the use of shortcut connections, also known as skip connections, which bypass one or more layers of the network. These connections allow gradients to propagate more effectively during training and enable the network to learn identity functions with greater ease. Crucially, if a group of intermediate layers does not enhance network performance, the residual connection permits the input to be passed forward unchanged. This mechanism prevents performance degradation that often occurs with increasing network depth. Like other modern architectures, ResNet50 is pretrained on the ImageNet dataset. The specific configuration of the layers and residual blocks of this model is detailed in Table 3.

EfficientNetV2s is a convolutional neural network model belonging to the EfficientNetV2 family. It was specifically designed to be smaller in size and faster in operation compared to earlier models in the EfficientNet lineage. This model family employs a combination of neural architecture search and scaling techniques to achieve an optimal balance between accuracy, training speed, and parameter efficiency (Tan & Le, 2021). A distinguishing characteristic of EfficientNetV2s is its implementation of both MBConv and Fused MBConv blocks, with emphasis on the latter in initial layers to enhance computational efficiency during training. Fig. 2 provides a visual representation of these architectural components. Like the previously discussed architectures, EfficientNetV2s was pretrained using the ImageNet dataset.

3.3 LibAUC library

The LibAUC library implements algorithms from scientific literature that are designed to optimize a family of statistical risk functions known as X risks. The Area Under the ROC Curve, or AUC, is a member of this X risks function family, (Yuan et al., 2023). The specific loss function implemented in LibAUC is based on a mathematical formulation introduced by Yuan et al. (2020). This formulation is called the AUCM loss. The library optimizes this loss function using a specialized technique known as the Proximal Epoch Stochastic Gradient, or PESG, method. We now provide a detailed description of both the AUCM loss function and its corresponding PESG optimizer.

Loss function. LibAUC implements a function that directly optimizes AUC. This metric is widely used to evaluate classifier performance, especially for imbalanced datasets standard in fields like medicine, where negative samples often outnumber positive ones. To achieve this optimization, Yuan et al. formulated a specialized loss function, which they referred to as the AUC margin loss (4). For a detailed explanation of this loss formulation, readers can refer to section 3.3 of the paper by Yuan et al. (2020).

$$A_M(\mathrm{w}) = A_1(\mathrm{w}) + A_2(\mathrm{w}) + \max_{\alpha \geq 0} 2\alpha\big(m - a(\mathrm{w}) + b(\mathrm{w})\big) - \alpha^2 . \tag{4}$$

Yuan et al. showed that the above objective corresponds to the min-max objective.

Theorem 1 Minimizing the AUC margin loss (1) is equivalent to the following min-max optimization (5):

Table 1. Description of the elements in a dense block

Element	Description
🟧	Input layer
🟩 🟪 🟨 ⬜	4 layers for feature extraction
🟦	Transition layer

Table 2. DenseNet121 architecture for the dataset Multiclass ImageNet (Huang et al., 2016)

Block	Details
Input Block	7×7 Conv, Stride 2
	3×3 Max Pooling, Stride 2
Dense Block 1	6 densely connected layers
	Each layer: 1×1 Conv → 3×3 Conv
Transition Layer 1	1×1 Conv
	2×2 Avg Pooling, Stride 2
Dense Block 2	12 densely connected layers
	Each layer: 1×1 Conv → 3×3 Conv
Transition Layer 2	1×1 Conv
	2×2 Avg Pooling, Stride 2
Dense Block 3	24 densely connected layers
	Each layer: 1×1 Conv → 3×3 Conv
Transition Layer 3	1×1 Conv
	2×2 Avg Pooling, Stride 2
Dense Block 4	16 densely connected layers
	Each layer: 1×1 Conv → 3×3 Conv
Classification Layer	7 × 7 global average pool
	1000D fully connected, sigmoid

Table 3. Architecture of ResNet50 (He et al., 2015)

Layer Name	Output Size	ResNet50 Configuration
conv1	112 × 112	7 × 7, 64, stride 2
conv2_x	56 × 56	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28 × 28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4_x	14 × 14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$
conv5_x	7 × 7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
FLOPs		3.8 × 10^9

$$\min_{\substack{\mathbf{w}\in\mathbb{R}^d \\ (a,b)\mathbb{R}^2}} \max_{\alpha\geq 0} \mathbb{E}_z[F_M(\mathbf{w},a,b,\alpha;\mathbf{z})], where \tag{5}$$

$$\begin{aligned} F_M(\mathbf{w},a,b,\alpha;\mathbf{z}) = (1-p)(h_w(\mathbf{x})-a)^2\mathbb{I}_{[y=1]} \\ + p(h_{\mathbf{w}}(\mathbf{x})-b)^2\mathbb{I}_{[y=-1]} - p(1-p)\alpha^2 \\ +2\alpha\left(p(1-p)m + ph_{\mathbf{w}}(\mathbf{x})\mathbb{I}_{[y=-1]} - (1-p)h_{\mathbf{w}}(\mathbf{x})\mathbb{I}_{[y=1]}\right). \end{aligned} \tag{6}$$

Where:
$h_{\mathbf{w}}(\mathbf{x})$ is the prediction scoring function, e.g., deep neural network
p is the ratio of positive samples to all samples
a is a statistic of positive predictions
b is a statistic of negative predictions
α is an auxiliary variable
m is a margin term that defines the desirable separation between classes

Optimizer. To optimize AUCM loss Yuan et al., employed the proximal epoch stochastic method (PESG). The algorithm of the PESG optimizer, which is also described in (Yuan et al., 2020) and the main steps are presented in Algorithm 1. As explained in their work, the algorithm utilizes several key parameters: λ represents a standard regularization parameter, often referred to as weight decay; $\gamma > 0$ is a positive parameter included to enhance model generalization; and η denotes the learning rate.

Hyperparameters. The hyperparameters for the loss function are:
p = (positive samples in train dataset) / (all samples in train dataset)
And
m with a value of 1.0
The hyperparameters and their values for the optimizer were:
η= 0.1
λ = 1e-4

```
Algorithm 1. PESG for optimizing the AUC margin loss
Inputs: $\boldsymbol{\eta},\boldsymbol{\gamma},\boldsymbol{\lambda},\boldsymbol{T}$
1: Initialize $\mathbf{v_1},\boldsymbol{\alpha_1} \geq \mathbf{0}$
2: for $\boldsymbol{t}=\mathbf{1},\dots,$ do
3:     Compute $\nabla_{\mathrm{v}}\boldsymbol{F}_{\mathbf{M}}(\mathrm{v_t},\boldsymbol{\alpha_t};\mathbf{z_t})$ and $\nabla_{\boldsymbol{\alpha}}\boldsymbol{F}_{\mathbf{M}}(\mathrm{v_t},\boldsymbol{\alpha_t};\mathbf{z_t})$.
4:     Update primal variables
           $\mathbf{v}_{t+1} = \mathbf{v}_t - \boldsymbol{\eta}\left(\nabla_{\mathrm{v}}\boldsymbol{F}_{\mathbf{M}}(\mathrm{v_t},\boldsymbol{\alpha_t};\mathbf{z_t}) + \boldsymbol{\gamma}(\mathbf{v}_t - \mathbf{v}_{ref})\right) - \boldsymbol{\lambda\eta}\mathbf{v}_t$
5: Update $\boldsymbol{\alpha}_{t+1} = [\boldsymbol{\alpha}_t + \boldsymbol{\eta}\nabla_{\boldsymbol{\alpha}}\boldsymbol{F}_{\mathbf{M}}(\mathrm{v_t},\boldsymbol{\alpha_t};\mathbf{z_t})]_+$.
6: Decrease $\boldsymbol{\eta}$ by a factor and update $\mathbf{v}_{ref}$ periodically
7: end for
```

3.4 ROC curve

The ROC curve is a graph that illustrates the performance of a diagnostic test. The Receiver Operating Characteristic (ROC) concept was initially developed during the Second World War to evaluate the ability of radar operators to distinguish real objects from noise on their screens (Fan et al., 2006). An ROC curve is created by plotting the true positive rate (sensitivity) on the vertical axis against the false positive rate (1 minus specificity) on the horizontal axis. Fig. 3 shows three examples of ROC curves.

3.5 Area under ROC curve

The Area Under the ROC Curve (AUC) is a widely used metric that measures the discriminatory power of a test (Fan et al., 2006). While common in the medical field, it is also used to evaluate the performance of classification models in engineering (Namdar et al., 2021). The AUC value ranges from 0 to 1. An ideal test, which correctly classifies all positive and negative samples, achieves a perfect score of 1.0. A score of 0.5 indicates the test has no discriminatory power, which is equivalent to random guessing. Therefore, any useful classifier must have an AUC greater than 0.5 (Majnik & Bosnić, 2013).

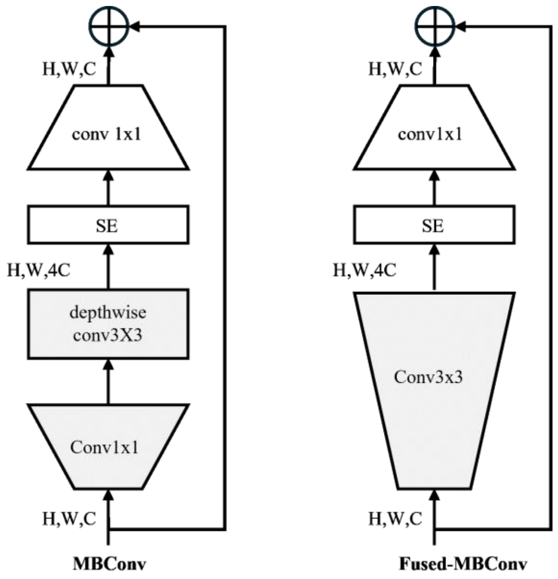


Fig. 2. Structure of MBConv and Fused-MBConv (Tan & Le, 2021).

3.6 Confusion matrix

According to Majnik and Bosnić (2013), a binary classifier must determine whether a target belongs to one of two possible classes. This work, for example, aims to determine if a metal structure has corrosion based on a set of images. The classification process yields four possible outcomes. A true positive (*TP*) occurs when an image of a corroded structure is correctly classified as corroded. A false negative (*FN*) occurs when an image of a corroded structure is incorrectly classified as not corroded. A true negative (*TN*) occurs when an image of an uncorroded structure is correctly classified as not corroded. A false positive (*FP*) occurs when an image of an uncorroded structure is incorrectly classified as corroded.

In this context, it is essential to note that the term positive refers to the class of interest, which is corrosion. Conversely, negative refers to the absence of that condition, such as no corrosion. The terms; *true* and *false* describe correct and incorrect classifications, respectively. As Majnik & Bosnić indicated, these four outcomes can be represented in a matrix known as a confusion matrix, which is illustrated in Fig. 4.

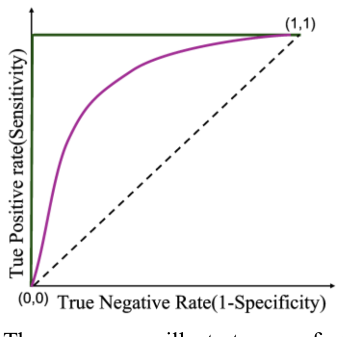


Fig. 3. Three examples of ROC curves. The green curve illustrates a perfect classifier, the purple curve represents a classifier with a high classification power, and the straight black line corresponds to a classifier with a random classification power.

4 Proposed methodology

This work proposes a methodology for classifying images of corroded surfaces using three pre-trained Convolutional Neural Network (CNN) models: DenseNet121, ResNet50, and EfficientNetV2s. To develop this methodology, we adapted an example from the LibAUC library that was designed initially for classifying skin cancer (melanoma) (Yuan, n.d.). The pre-trained network used in that example was DenseNet121.

		Predicted class	
		Positive	Negative
Actual class	Positive	TP	FN
	Negative	FP	TN

Fig. 4. Graphical representation of the confusion matrix.

For this study, we used a dataset obtained from GitHub (Sun, n.d.) this dataset has also been used in similar studies (Farooqui et al., 2024; Z. Zhao et al., 2024). Unlike the dataset used in the original melanoma classification study (Yuan et al., 2020), the dataset used is balanced, containing 990 images of corroded surfaces (positive samples) and 829 images of non-corroded surfaces (negative samples). We chose the LibAUC library because it is designed to maximize the Area Under the Curve (AUC). The AUC is a helpful metric for evaluating classifier performance (Calders & Jaroszewicz, n.d.; Hoo et al., 2017; Namdar et al., 2021). The overall methodology is represented graphically in Fig. 5.

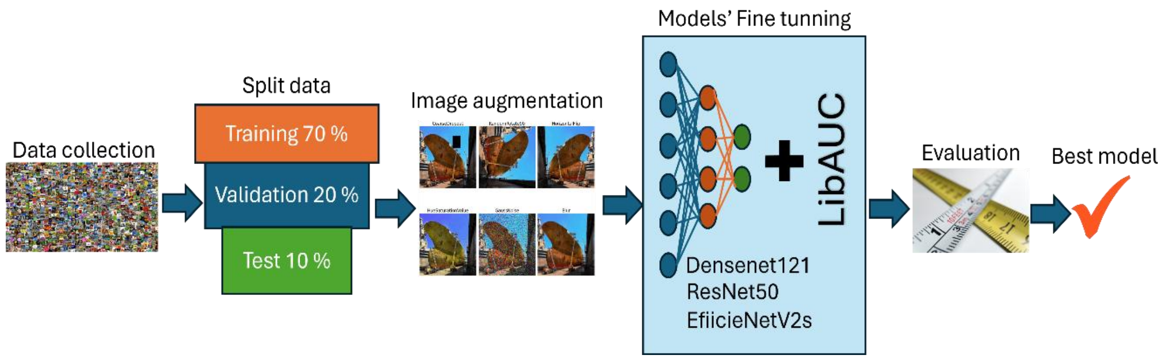


Fig. 5. Workflow for fine-tuning pre-trained CNNs using LibAUC for binary corrosion classification**.**

4.1 Pre-trained models

To leverage the powerful feature extraction capabilities learned from the ImageNet dataset, we employed a transfer learning approach. We fine-tuned three widely recognized pre-trained models for the specific task of corrosion detection: DenseNet121, ResNet50, and EfficientNetV2s. This selection was motivated by their proven efficacy and frequent application in related computer vision research, particularly in domains involving material surface analysis and defect detection.

4.2 Raw data

Composition. The dataset for this study comprises 1,819 images depicting various metal structures, including ships, bridges, cars, pipes, and tanks. Within this collection, 990 images present instances of corrosion, while the remaining 829 images show structures without corrosion. All images possess different dimensions. Representative examples from the dataset include five randomly selected images of corroded structures, shown Fig. 6**Error! Reference source not found.**, and a corresponding set of non-corroded structures, displayed in Fig. 7

Dataset split. The complete dataset was systematically divided into three separate subsets to support the distinct phases of the machine learning workflow. A total of 70% of the images formed the training set, which serves as the primary data for the model to learn feature representations. To enhance the model ability to generalize and to increase the diversity of the training examples artificially, data augmentation techniques were applied exclusively to this subset. Another 20% of the data was allocated to the validation set. This set is essential for model selection and hyperparameter tuning, as performance is evaluated on it after each training epoch using the Area Under the Curve (AUC) metric. The remaining 10% of the images constituted the test set. This subset is withheld entirely until the final stage of the process, providing a rigorous and unbiased assessment of the model performance on completely unseen data.

Data preprocessing. A critical consideration for this study is that the available volume of data is insufficient for robust model training. Training on a limited dataset can produce a model that fails to generalize effectively, a phenomenon known as overfitting.

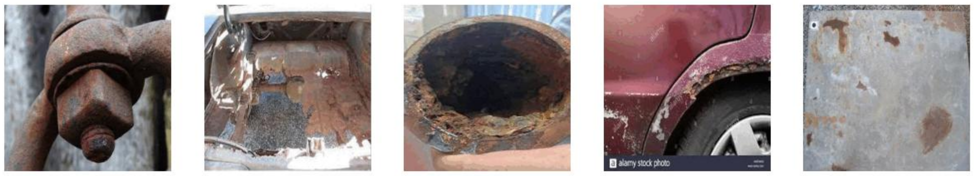


Fig. 6. Five random images of corroded structures from the dataset used to train the model.

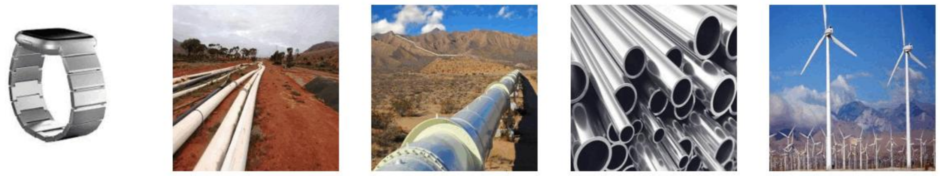


Fig. 7. Five random images of no corroded structures from the dataset used to train the model.
.

An overfitted model typically demonstrates strong performance on the validation data but performs poorly on new, unseen data. This performance degradation is most evident when the model is evaluated on the test set, which consists of data withheld from both the training and validation phases.

To mitigate the risk of overfitting, data augmentation techniques were applied exclusively to the images within the training set. This process artificially expands the dataset by generating new, modified versions of the original images through various transformations. These transformations included adjustments to brightness, the application of blur, random cropping, and rotation. By exposing the model to these varied conditions, augmentation encourages the learning of more generalizable features. For this task, the open-source Python library Albumentations (*Albumentations*, n.d.) was employed to implement these transformations. Fig. 8 demonstrates the effect of this technique, showing six augmented variations derived from a single image randomly selected from the training dataset.

4.3 LibAuc library

The decision to employ the LibAUC library was motivated by its unique optimization objective. Unlike standard approaches that maximize common metrics like accuracy or the F1-score, LibAUC is specifically designed to optimize the AUC metric directly. As detailed in Section 3.5, the AUC metric measures the inherent discrimination power of a classifier, representing its ability to rank positive instances higher than negative ones. To accomplish this direct optimization, the library implements a loss function known as AUCM, which is minimized during training using a stochastic gradient descent variant called the PESG optimizer. This specific combination of loss function and optimizer was integrated into the training stage for all CNN models in this study.

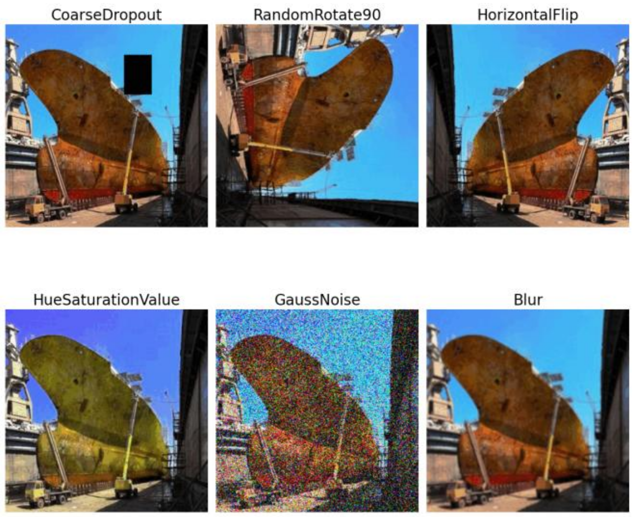


Fig. 8. Augmentations applied to a random image from the training dataset.

4.4 Training

The training procedure was designed to ensure both computational efficiency and statistical robustness. To achieve this, ten independent training runs were executed for each model architecture. This approach provides a sufficient basis for statistical comparison while effectively managing computational time. The resulting set of ten trained models per architecture generates a distribution of performance metrics, which allows for a rigorous comparison of AUC values from the validation and test sets using non-parametric statistical tests, such as the Wilcoxon signed-rank test.

Each run was set for a maximum of fifty epochs. This limit was not always reached, as an early stopping callback was employed to automatically terminate training if the validation AUC failed to improve over several consecutive epochs. This technique serves two primary purposes: it prevents the model from overfitting the training data, and it optimizes computational resources by concluding the process once peak performance is achieved, often before the fiftieth epoch.

Fine-tuning models. A fine-tuning technique was applied to the models. The feature extraction layers were frozen for the first ten epochs. This freezing allowed only the classifier layers to adapt to the specific task. After this period, these layers were unfrozen. The entire model then underwent the complete training process. These sequential steps comprise the standard fine-tuning technique.

Early Stop. A mechanism known as early stop was applied to the training process. This technique halts training if a monitored metric fails to improve after a set number of epochs. The specific metric monitored for this purpose was AUC. Implementation of this mechanism commenced after the initial ten epochs. The use of early stopping serves to prevent overfitting.

Store Best Models. The best performing model from each of the ten independent training runs was stored during the validation stage. Model selection was based on the highest AUC score achieved on the validation set. This procedure resulted in ten optimal models for each architecture. These stored models will undergo final evaluation using test data.

4.5 Evaluation of the models

Each of the ten models per architecture was evaluated using the images from the test set. The performance metrics obtained from this final evaluation were then compared to the results previously achieved on the validation set. AUC served as the primary metric for this comparison. Additional support metrics provided further performance context; these metrics included accuracy, precision, recall, F1-score, and specificity.

5 Experiment and results

Experiments were conducted using a Jupyter Notebook on Kaggle.com, and the system specifications are listed in Table 4Table 4. The data employed to train the model was described in Subsection 4.2. The primary metric considered was AUC, the most accurate technique to compute AUC is trapezoidal integration (Namdar et al., 2021). Other metrics to measure the model performance are widely used in machine learning and are derived from the terms of the confusion matrix, as seen in Subsection 3.6. They are defined from (7) to (11). Table 5 summarizes the interpretation of each of these metrics.

Table 4. Specifications of the Notebook where the experiments of this work were carried out

Component	Specification
CPU	Intel(R) Xeon(R) CPU @ 2.00GHz
CPU Speed	2000.162 MHz
CPU Cores	2
Operating System	Linux f2257cf573ca 6.6.56+ x86_64
GPUs	2 GPU Tesla T4 @ 15360 MiB
CUDA Version	12.6
Python Version	3.10.12
LibAUC Version	1.3.0

$$Accuracy = \frac{TP + TN}{TN + TP + FP + FN} \tag{7}$$

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$Recall\ (TPR) = \frac{TP}{TP + FN} \tag{9}$$

$$Specificity = \frac{TN}{TN + FP} \tag{10}$$

$$F1\ scores = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{11}$$

Table 5. Interpretation of the performance metrics for the model in the binary classification of corrosion

Expression	Interpretation
Accuracy	A high value means that the model classifies images correctly, either with or without corrosion.
Precision	A high value indicates that the model is correctly classifying most images of corroded surfaces, thereby avoiding the incorrect classification of images with no corroded surfaces as corroded.
Recall	A high value indicates that the model is correctly classifying most images of corroded surfaces, thereby avoiding the incorrect classification of images with corroded surfaces as non-corroded.
Specificity	A high value indicates that the model accurately classifies most images of non-corroded surfaces, avoiding the misclassification of images with non-corroded surfaces as corroded.
F1 score	A high value means that the model has an equilibrium between precision and recall.

5.1 Experiment

Each of the three models (DenseNet121, ResNet50, and EfficientNetV2s) was trained ten times. Each training run consisted of a 50 epoch process that involved two stages: training and validation. In the training stage, batches of augmented images were fed to the CNN. The AUCM loss function computed the difference between the model predictions and the actual classes. Backpropagation then calculated the loss gradient, which the PESG optimizer used to update the network weights. In the validation stage, the model made predictions on a non-augmented validation dataset. The Area Under the Curve (AUC) score was then computed from these predictions, and the version of the model that achieved the highest AUC score was saved.

For the first ten epochs of each run, the feature extraction layers were frozen and then unfrozen for the remaining epochs. During this second phase, an early stopping mechanism was also applied, which halts the training if the AUC score fails to improve after a specified number of epochs. This process yielded ten optimized binary classification models for each of the three architectures. Finally, each group of trained models was evaluated with the test data. Metrics were computed by applying a threshold of t = 0.5 to the output probability of the model. As shown in expression (12), this threshold classified an image as either having corrosion (label 1) or not having corrosion (label 0).

$$\hat{y} = \begin{cases} 1, & if\ P(y = 1|x) \geq t \\ 0, & if\ P(y = 1|x) < t \end{cases}. \tag{12}$$

Where $\hat{y}$ represents the model prediction.

5.2 Results and discussion

This section presents the results from the evaluation of three distinct CNN architectures for binary corrosion classification: DenseNet121, ResNet50, and EfficientNetV2s. Table 6, Table 7, and Table 8 detail the performance of each of the ten trained models per architecture across both the validation and test datasets. The performance metrics for all ten models from each architecture are detailed in their respective tables. Table 6 provides a complete summary of the results for the DenseNet121 models. Table 7 and

Table 8 present the corresponding data for the ResNet50 and EfficientNetV2s models, respectively. Each table displays the following metrics for every individual model (labeled 1 through 10) on both the validation (Val) and test (Test) datasets: AUC, Accuracy, Precision, Recall, F1-score, and Specificity.

This detailed presentation of the data allows for a thorough assessment of the performance and generalization capabilities of each model. These results, in turn, provide the basis for a robust comparative analysis across the three architectures. Within the results for each architecture, bolded values identify the best-performing model. This designation is given to the model that achieved the highest AUC score on the test dataset, the primary metric used for comparison

Table 6. Performance metrics results for DenseNet121 models on validation and test data

Model	AUC		Accuracy		Precision		Recall		F1_score		Specificity	
	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test
1	0.993	0.994	0.960	0.978	0.982	0.979	0.944	0.979	0.962	0.979	0.979	0.975
2	0.992	0.993	0.942	0.945	0.993	0.968	0.899	0.929	0.944	0.948	0.993	0.963
3	0.995	**0.996**	0.972	0.961	0.988	0.979	0.960	0.949	0.974	0.964	0.986	0.975
4	0.995	0.993	0.960	0.972	0.966	0.970	0.960	0.979	0.963	0.974	0.959	0.963
5	0.993	0.992	0.972	0.956	0.994	0.969	0.955	0.949	0.974	0.959	0.993	0.963
6	0.994	0.991	0.969	0.956	0.994	0.978	0.949	0.939	0.971	0.958	0.993	0.975
7	0.995	0.993	0.966	0.972	0.988	0.970	0.949	0.979	0.968	0.974	0.986	0.963
8	0.996	0.995	0.975	0.978	0.983	0.970	0.972	0.989	0.977	0.980	0.979	0.963
9	0.994	0.994	0.957	0.972	0.982	0.979	0.938	0.969	0.960	0.974	0.979	0.975
10	0.993	0.995	0.957	0.956	0.994	0.978	0.927	0.939	0.959	0.958	0.993	0.975

Table 7. Performance metrics results for ResNet50 models on validation and test data

Model	AUC		Accuracy		Precision		Recall		F1_score		Specificity	
	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test
1	0.993	**0.996**	0.972	0.956	0.967	0.950	0.983	0.969	0.975	0.960	0.959	0.939
2	0.991	0.992	0.951	0.950	0.993	0.968	0.916	0.939	0.953	0.953	0.993	0.963
3	0.993	0.989	0.960	0.961	0.994	0.969	0.932	0.959	0.962	0.964	0.993	0.963
4	0.996	0.989	0.978	0.967	0.988	0.969	0.972	0.969	0.980	0.969	0.986	0.963
5	0.993	0.991	0.966	0.950	0.988	0.959	0.949	0.949	0.968	0.954	0.986	0.951
6	0.992	0.991	0.972	0.967	0.994	0.969	0.955	0.969	0.974	0.969	0.993	0.963
7	0.993	0.986	0.957	0.945	0.976	0.949	0.944	0.949	0.960	0.949	0.973	0.939
8	0.991	0.988	0.966	0.945	0.994	0.958	0.944	0.939	0.968	0.948	0.993	0.951
9	0.994	0.991	0.972	0.956	0.977	0.959	0.972	0.959	0.974	0.959	0.973	0.951
10	0.996	0.988	0.975	0.961	0.977	0.960	0.977	0.969	0.977	0.964	0.973	0.951

Fig. 9 displays box plots of the AUC scores for the ten models trained for each architecture: DenseNet121, ResNet50, and EfficientNetV2s. The plots show the score distributions on both the validation (blue) and test (red) datasets. The results indicate that the EfficientNetV2s model achieved the highest performance, with median AUC values of approximately 0.997 on the validation dataset and 0.993 on the test dataset. In comparison, the DenseNet121 model shows consistent performance, with a median AUC score of approximately 0.994 across both datasets.

The ResNet50 model exhibits the most variability, particularly on the test dataset, showing lower median AUC values and a wider distribution range. To assess the generalization capacity of each architecture, a Wilcoxon signed-rank test was performed on the paired AUC values from the validation and test sets. The significance level (α) was set to 0.05. For each architecture, the null hypothesis (H0) stated that there is no significant difference between the median AUC scores of the validation and test sets.

DenseNet121
H0: There is no statistically significant difference in AUC between validation data and test data across the DenseNet121 models.

ResNet50
H0: There is no statistically significant difference in AUC between validation data and test data across the ResNet50 models.

EfficieNetV2s
H0: There is no statistically significant difference in AUC between validation data and test data across the EfficieNetV2s models.

Table 8. Performance metrics results for EfficieNetV2s models on validation and test data

Model	AUC		Accuracy		Precision		Recall		F1_score		Specificity	
	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test
1	0.998	0.993	0.972	0.945	0.988	0.968	0.960	0.929	0.974	0.948	0.986	0.963
2	0.996	0.990	0.972	0.928	0.994	0.947	0.955	0.919	0.974	0.933	0.993	0.939
3	0.998	0.993	0.978	0.967	1.000	0.989	0.960	0.949	0.980	0.969	1.000	0.987
4	0.998	**0.995**	0.966	0.967	0.982	0.989	0.955	0.949	0.968	0.969	0.979	0.987
5	0.996	0.994	0.966	0.950	0.988	0.978	0.949	0.929	0.968	0.953	0.986	0.975
6	0.995	0.989	0.972	0.923	0.994	0.929	0.955	0.929	0.974	0.929	0.993	0.915
7	0.998	0.991	0.975	0.945	1.000	0.968	0.955	0.929	0.977	0.948	1.000	0.963
8	0.997	0.994	0.963	0.950	0.988	0.978	0.944	0.929	0.965	0.953	0.986	0.975
9	0.998	0.993	0.966	0.956	0.994	0.989	0.944	0.929	0.968	0.958	0.993	0.987
10	0.997	0.991	0.972	0.950	0.982	0.950	0.966	0.959	0.974	0.954	0.979	0.939

H0 is rejected if the p-value < α and accepted if the p-value > α.

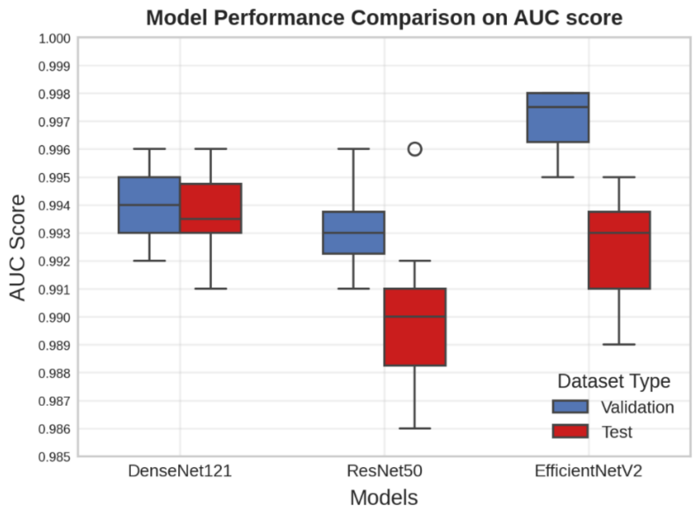


Fig. 9. AUC score distributions for the three CNN architectures on validation (blue) and test (red) datasets. Results from the evaluation of ten models per architecture.

The performance metric statistics for all thirty models that completed the fine-tuning process are consolidated in Table 9 and **Error! Reference source not found.**. These tables provide a comprehensive overview of model effectiveness. Table 9 presents statistical summaries for AUC, Accuracy, and Precision. **Error! Reference source not found.** provides the same statistical measures for Recall, F1 score, and Specificity. For each metric, the calculated statistics include the average value (AVG), standard deviation (SD), minimum value (MIN), and maximum value (MAX).

The Wilcoxon signed-rank test showed different generalization capabilities among the architectures. The null hypothesis (H0) was rejected for two of the models: ResNet50, with a p-value of 0.031, and EfficientNetV2s, with a p-value of 0.004. This outcome suggests that these two architectures do not generalize well to unseen data. In contrast, the null hypothesis was not rejected for the DenseNet121 architecture (p-value = 0.431). This result indicates that the model generalizes effectively, as its performance did not significantly differ between the validation and test datasets.

Table 11 presents a comparison of the AUC performance of the individual architectures when evaluated on validation data and test data. The numbers in bold represent the average ± standard deviation of the AUC corresponding to DenseNet121 when evaluated with test data; this result indicates that this architecture exhibits the best stability and yields the highest AUC values on average.

Figures Fig. 10 and Fig. 11 present the confusion matrix and ROC curves, respectively, for the models with the best performance from each architecture, specifically those with the highest AUC scores. Table 12 identifies the corresponding models for these architectures, showing the key metrics.

Table 9. Summary statistics of AUC, accuracy, and precision for Fine-Tuned DenseNet121, ResNet50, and EfficientNetV2s models

Architecture	Statics	AUC		Accuracy		Precision	
		Val	Test	Val	Test	Val	Test
DenseNet121	AVG	0.994	0.993	0.963	0.964	0.986	0.974
	SD	0.001	0.001	0.010	0.011	0.008	0.004
	MIN	0.992	0.991	0.942	0.945	0.966	0.968
	MAX	0.996	0.996	0.975	0.978	0.994	0.979
ResNet50	AVG	0.993	0.990	0.966	0.955	0.984	0.961
	SD	0.001	0.002	0.008	0.008	0.009	0.007
	MIN	0.991	0.986	0.951	0.945	0.967	0.949
	MAX	0.996	0.996	0.978	0.967	0.994	0.969
EfficientNetV2s	AVG	0.997	0.992	0.970	0.948	0.991	0.968
	SD	0.001	0.001	0.004	0.014	0.006	0.020
	MIN	0.995	0.989	0.963	0.923	0.982	0.929
	MAX	0.998	0.995	0.978	0.967	1.000	0.989

Table 10 Summary statistics of recall, F1-score, and specificity for Fine-Tuned DenseNet121, ResNet50, and EfficientNetV2s models

Architecture	Statics	AUC		Accuracy		Precision	
		Val	Test	Val	Test	Val	Test
DenseNet121	AVG	0.945	0.960	0.965	0.966	0.984	0.969
	SD	0.020	0.021	0.009	0.010	0.010	0.006
	MIN	0.899	0.929	0.944	0.948	0.959	0.963
	MAX	0.972	0.989	0.977	0.980	0.993	0.975
ResNet50	AVG	0.954	0.957	0.969	0.958	0.982	0.953
	SD	0.021	0.012	0.008	0.007	0.011	0.009
	MIN	0.916	0.939	0.953	0.948	0.959	0.939
	MAX	0.983	0.969	0.980	0.969	0.993	0.963
EfficientNetV2s	AVG	0.954	0.935	0.972	0.951	0.989	0.963
	SD	0.007	0.012	0.004	0.013	0.007	0.024
	MIN	0.944	0.919	0.965	0.929	0.979	0.915
	MAX	0.966	0.959	0.980	0.969	1.000	0.987

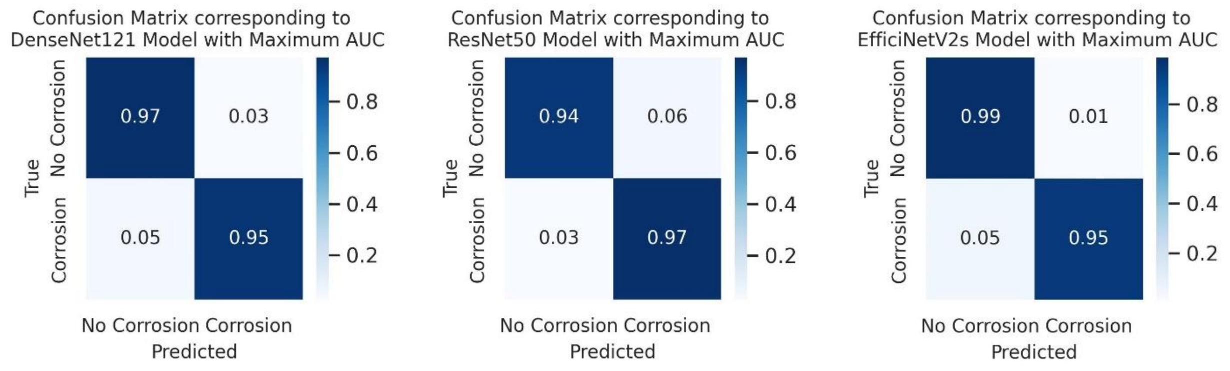
Table 11. AUC performance of individual architecture evaluated on validation and test data

Architecture	Validation			Test		
	AUC (AVG ± SD)	MIN	MAX	AUC (AVG ± SD)	MIN	MAX
DenseNet121	0.994 ± 0.001	0.992	0.996	**0.993 ± 0.001**	0.991	0.996
ResNet50	0.993 ± 0.001	0.991	0.996	0.990 ± 0.002	0.986	0.996
EfficientNetV2s	0.997 ± 0.001	0.995	0.998	0.992 ± 0.001	0.989	0.995

The confusion matrix and ROC curves of the best models, along with the data in Table 6, Table 7, Table 8, Table 9 and Table 10, provide evidence that all architectures (DenseNet121, ResNet50, EfficientNetV2s) have excellent performance for the binary classification of corrosion. On the test data, the architectures achieved high AUC values averaging 0.993, 0.994, and 0.997, respectively. This performance underscores their considerable potential for corrosion detection, a critical task in predictive maintenance and industrial safety. These results suggest the models are highly effective at classification when optimized with LibAUC.

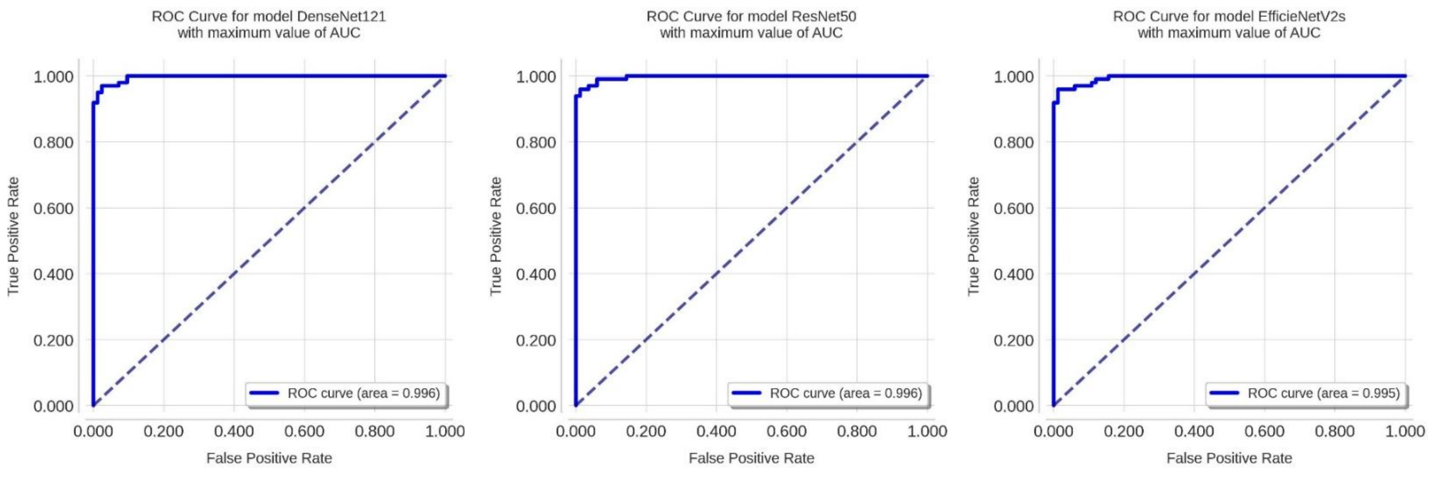
Although high AUC values above 0.99 for all architectures suggest excellent performance, an analysis of generalization capacity reveals a critical difference. The p-values from our hypothesis test show this disparity. Only DenseNet121 demonstrated robust generalization, indicated by a p value of 0.431 that permitted us to accept the null hypothesis. DenseNet121, therefore, emerges as the optimal choice.

In contrast, ResNet50 (p = 0.031) and EfficientNetV2s (p = 0.004) both rejected the null hypothesis, which indicates their ability to generalize is less robust. This weakness was particularly evident in EfficientNetV2s. The robust generalization of DenseNet121 stems from its use of LibAUC, highlighting the potential of directly optimizing the AUC metric. This approach is powerful for binary classification tasks like corrosion detection, which require a classifier with high discriminatory power, a quality the AUC metric directly represents.



(a) (b) (c)

Fig. 10. Confusion matrix of the models with the maximum AUC value for each architecture (a) DenseNet121, (b) ResNet50, and (c) EfficienNetV2s.



(a) (b) (c)

Fig. 11. Roc Curve of the models with the maximum AUC value for each architecture (a) DenseNet121, (b) ResNet50, and (c) EfficienNetV2S.

Table 12. Metrics of evaluation for the best models according to the maximum value of AUC on test data taken from Table 6, Table 7 and Table 8. Numbers in bold represent the best value for each metric

Model	AUC	Accuracy	Precision	Recall	F1_score	Specificity
(3) DenseNet121	**0.996**	0.961	0.979	0.949	0.964	0.975
(1) ResNet50	**0.996**	0.956	0.950	**0.969**	0.960	0.939
(4) EfficieNetV2s	0.995	**0.967**	**0.989**	0.949	**0.969**	**0.987**

5.3 Comparison with two state-of-the-art research studies

We compared our results with two state-of-the-art studies to contextualize the performance of our model. This comparison highlights explicitly the AUC scores because they represent the discriminatory power of a classifier. Table 13 presents the complete data from this analysis, allowing for a direct evaluation against the other works.

Table 13. Comparison of the results of the metrics of the best model obtained related to AUC, with the results reported in two state-of-the-art studies. Values in bold letters represent the best values

Research	Model	AUC	Accuracy	Precision	Recall	F1-score	Specificity
(Z. Zhao et al., 2024)	EfficientNetV2	0.97	0.9176	0.9146	0.9036	0.9091	NR
(Farooqui et al., 2024)	CNN from scratch	NR	**1.0**	**1.0**	**1.0**	**1.0**	NR
This work	LibAUC-DenseNet121	**0.996**	0.961	0.979	0.949	0.964	0.975

NR = No reported.

6 Conclusions

This study examined three deep learning architectures (DenseNet121, ResNet50, and EfficientNetV2s) for binary corrosion classification, using the LibAUC library to optimize the AUC metric directly. All architectures initially proved effective, achieving high AUC values that confirmed their strong potential for corrosion detection. However, a rigorous generalization analysis using a hypothesis test revealed a critical difference in their robustness. Only DenseNet121 demonstrated superior generalization, as its ability to maintain high performance on unseen data led to the acceptance of the null hypothesis. This result establishes it as the best option for this classification task. In contrast, despite their high initial AUC values, ResNet50 and EfficientNetV2s showed significant limitations, suggesting a high risk of overfitting.

In summary, this study validates the efficacy of CNNs with direct AUC optimization for binary corrosion classification and emphasizes that generalization analysis is essential for model selection. The findings identify DenseNet121 as the most robust candidate for this task. Future work will focus on validating this model with more diverse, real-world datasets and exploring techniques to enhance its superior generalization capabilities further.

Finally, we compared our results with those of two related studies (Farooqui et al., 2024; Z. Zhao et al., 2024). The study by Farooqui et al., reported perfect scores (100%) for accuracy, precision, recall, and F1 score, but omitted AUC and specificity metrics. Such perfect scores can suggest potential overfitting or limited dataset diversity, issues our study mitigated with fine-tuning and early stopping techniques. Our DenseNet121-LibAUC model outperformed the EfficientNetV2 model from Z. Zhao et al., across all key metrics. Specifically, our model achieved a 2.68% improvement in AUC (0.996 vs. 0.97) and a notable increase in accuracy (0.961 vs. 0.9176).

References

Agatonovic-Kustrin, S., & Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, *22*(5), 717–727. https://doi.org/10.1016/S0731-7085(99)00272-1

Albumentations. (n.d.). Retrieved February 13, 2025, from https://albumentations.ai/docs/1-introduction/#what-is-image-augmentation

Anzures-Garcia, M., Sánchez Gálvez, L. A., Larios Gómez, M., Tapia Rodríguez, A., & Aguirre Agustín, R. (2024). IAA-CNN: Intelligent Attendance Algorithm based on Convolutional Neural Networks. *International Journal of Combinatorial Optimization Problems and Informatics*, *15*(5), 51–63. https://doi.org/10.61467/2007.1558.2024.v15i5.557

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*. https://doi.org/10.1016/j.ndteint.2019.102134

Calders, T., & Jaroszewicz, S. (n.d.). Efficient AUC Optimization for Classification. In *Knowledge Discovery in Databases: PKDD 2007* (pp. 42–53). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-74976-9_8

Casas, E., Ramos, L., Romero, C., & Rivas-Echeverría, F. (2024). A comparative study of YOLOv5 and YOLOv8 for corrosion segmentation tasks in metal surfaces. *Array*, *22*. https://doi.org/10.1016/j.array.2024.100351

Das, A., Dorafshan, S., & Kaabouch, N. (2024). Autonomous Image-Based Corrosion Detection in Steel Structures Using Deep Learning. *Sensors*, *24*(11), 3630. https://doi.org/10.3390/s24113630

Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, & Li Fei-Fei. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

Fan, J., Upadhye, S., & Worster, A. (2006). Understanding receiver operating characteristic (ROC) curves. *CJEM*, *8*(01), 19–20. https://doi.org/10.1017/S1481803500013336

Farooqui, M., Rahman, A., Alsuliman, L., Alsaif, Z., Albaik, F., Alshammari, C., Sharaf, R., Olatunji, S., Althubaiti, S. W., & Gull, H. (2024). A Deep Learning Approach to Industrial Corrosion Detection. *Computers, Materials & Continua*, *81*(2), 2587–2605. https://doi.org/10.32604/cmc.2024.055262

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Fondevik, S. K., Stahl, A., Transeth, A. A., & Knudsen, O. O. (2020). Image Segmentation of Corrosion Damages in Industrial Inspections. *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, *2020-November*, 787–792. https://doi.org/10.1109/ICTAI50040.2020.00125

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition*. http://arxiv.org/abs/1512.03385

Holm, E., Transeth, A. A., Knudsen, O. O., & Stahl, A. (2020). Classification of corrosion and coating damages on bridge constructions from images using convolutional neural networks. In Wolfgang Osten and Dmitry P. Nikolaev (Ed.), *Twelfth International Conference on Machine Vision (ICMV 2019)* (p. 45). SPIE-Intl Soc Optical Eng. https://doi.org/10.1117/12.2557380

Hoo, Z. H., Candlish, J., & Teare, D. (2017). What is an ROC curve? *Emergency Medicine Journal*, *34*(6), 357–359. https://doi.org/10.1136/emermed-2017-206735

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2016). *Densely Connected Convolutional Networks*. https://doi.org/10.48550/arXiv.1608.06993

Indolia, S., Goswami, A. K., Mishra, S. P., & Asopa, P. (2018). Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. *Procedia Computer Science*, *132*, 679–688. https://doi.org/10.1016/j.procs.2018.05.069

Koushik, J. (2016). *Understanding Convolutional Neural Networks*. https://doi.org/10.48550/arXiv.1605.09081

Majnik, M., & Bosnić, Z. (2013). ROC analysis of classifiers in machine learning: A survey. *Intelligent Data Analysis*, *17*(3), 531–558. https://doi.org/10.3233/IDA-130592

Namdar, K., Haider, M. A., & Khalvati, F. (2021). A Modified AUC for Training Convolutional Neural Networks: Taking Confidence Into Account. *Frontiers in Artificial Intelligence*, *4*. https://doi.org/10.3389/frai.2021.582928

Nie, W. (2023). Artificial Intelligence-based Recognition of Wear and Corrosion on Steel Surfaces. *2023 4th International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering, ICBAIE 2023*, 284–287. https://doi.org/10.1109/ICBAIE59714.2023.10281351

O'Shea, K., & Nash, R. (2015). *An Introduction to Convolutional Neural Networks*. https://doi.org/10.48550/arXiv.1511.08458

Soares, L. B., De Oliveira Evald, P. J. D., Evangelista, E. A. D., Drews-Jr, P. L. J., Da Costa Botelho, S. S., & MacHado, R. I. (2023). An Autonomous Inspection Method for Pitting Detection Using Deep Learning∗. *IEEE International Conference on Industrial Informatics (INDIN)*, *2023-July*. https://doi.org/10.1109/INDIN51400.2023.10218256

Srivastava, A., Ji, G., & Singh, R. K. (2021, October 9). Application of Deep-Learning Architecture for Image Analysis based Corrosion Detection. *Proceedings - 1st International Conference on Smart Technologies Communication and Robotics, STCR 2021*. https://doi.org/10.1109/STCR51658.2021.9588887

Sun, P. (n.d.). *Phase5_Capstone-Project: Deep learning for automated corrosion detection*. Retrieved February 15, 2025, from https://github.com/pjsun2012/Phase5_Capstone-Project

Suresh Kumar, S., Gokul, K., Hemanand, I., Eaknath, M. S., & Jayabharanivelu, V. M. (2023). Video Analytics using Deep Learning in Cloud Services to Detect Corrosion - A Comprehensive Survey. *3rd International Conference on Innovative Mechanisms for Industry Applications, ICIMIA 2023 - Proceedings*, 949–955. https://doi.org/10.1109/ICIMIA60377.2023.10426026

Tan, M., & Le, Q. V. (2021). *EfficientNetV2: Smaller Models and Faster Training*. http://arxiv.org/abs/2104.00298

Tariber, S. S., Shreehari, A., Belaldavar, P. S., Poornachandra, M., & Shruthi, M. L. J. (2023). Performance Analysis of Subsea Pipeline Defect Classification Using Supervised Machine Learning. *Proceedings of IEEE International Conference on Modelling, Simulation and Intelligent Computing, MoSICom 2023*, 492–497. https://doi.org/10.1109/MoSICom59118.2023.10458750

Vriesman, D., Britto Junior, A., Zimmer, A., & Lameiras Koerich, A. (2019). Texture CNN for thermoelectric metal pipe image classification. *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, *2019-November*, 569–574. https://doi.org/10.1109/ICTAI.2019.00085

Yuan, Z. (n.d.). *LibAUC: A Deep Learning Library for X-Risk Optimization*. Retrieved February 2, 2025, from https://github.com/Optimization-AI/LibAUC/blob/1.4.0/examples/08_Optimizing_AUROC_Loss_with_DenseNet121_on_Melanoma.ipynb

Yuan, Z., Yan, Y., Sonka, M., & Yang, T. (2020). *Large-scale Robust Deep AUC Maximization: A New Surrogate Loss and Empirical Studies on Medical Image Classification*. https://doi.org/10.48550/arXiv.2012.03173

Yuan, Z., Yan, Y., Sonka, M., & Yang, T. (2021). Large-scale robust deep auc maximization: A new surrogate loss and empirical studies on medical image classification. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3040–3049. https://doi.org/10.1109/ICCV48922.2021.00303

Yuan, Z., Zhu, D., Qiu, Z. H., Li, G., Wang, X., & Yang, T. (2023). LibAUC: A Deep Learning Library for X-Risk Optimization. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 5487–5499. https://doi.org/10.1145/3580305.3599861

Zhao, R., Zhang, H., Zhou, J., Liao, L., & Xia, R. (2017). The testing scheme for steel corrosion in the reinforced concrete via near field effect of meter-band wave. *Progress in Electromagnetics Research Letters*, *66*(February), 127–134. https://doi.org/10.2528/PIERL17011403

Zhao, Z., Bakar, E. B. A., Razak, N. B. A., & Akhtar, M. N. (2024). Corrosion image classification method based on EfficientNetV2. *Heliyon*, *10*(17), e36754. https://doi.org/10.1016/j.heliyon.2024.e36754