Deep Neural Networks for external corrosion classification in industrial above-ground storage tanks

Anibal Alviz-Meza¹ and Dario Y. Peña-Ballesteros¹

¹Affiliation not available

April 16, 2024

Abstract

This paper explores EfficientNet smaller models for a multiclassification task of corrosion types in above-ground storage tanks through transfer learning and finetuning approaches. Data augmentation was used to increment the data an oil and gas company provided, reaching a dataset of around 5000 images. The images were stored in Google Drive and imported by Colab to obtain the models using TensorFlow and Keras. After the hyperparameters’ tuning a transfer learning model was selected and explored with fine tuning. The EfficientNetB0 model delivered from fine-tuning accomplished 94% performance. This work is the first attempt to deploy an artificial vision automatic tool for being implemented during tank inspection in the industrial sector. In a further development, this model can be coupled with one based on object detection for the remote identification of failures due to external corrosion during tank inspection; improving safety and reliability in the oil and gas industry.

Hosted file

Deep Neural Networks for external corrosion classification in industrial above-ground storage tanks

Anibal Alviz-Mezaa,*, Dario Y. Peña-Ballesterosa

aGrupo de Investigaciones en Corrosión, Universidad Industrial de Santander, Parque Tecnológico Guatiguará, Piedecuesta 681011, Colombia

anibalalviz@gmail.com*

Abstract: This paper explores EfficientNet smaller models for a multiclassification task of corrosion types in above-ground storage tanks through transfer learning and fine-tuning approaches. Data augmentation was used to increment the data an oil and gas company provided, reaching a dataset of around 5000 images. The images were stored in Google Drive and imported by Colab to obtain the models using TensorFlow and Keras. After the hyperparameters’ tuning a transfer learning model was selected and explored with fine tuning. The EfficientNetB0 model delivered from fine-tuning accomplished 94% performance. This work is the first attempt to deploy an artificial vision automatic tool for being implemented during tank inspection in the industrial sector. In a further development, this model can be coupled with one based on object detection for the remote identification of failures due to external corrosion during tank inspection; improving safety and reliability in the oil and gas industry.

Keywords: Convolutional Neural Networks; Transfer Learning; Fine-tuning; External Corrosion; Storage Tanks.

1. Introduction

Oil and gas industries are still the main pillars of the world economy. For the 2020-2030 decade, it is expected the maximum production of conventional oil, around 4.5–4.8 billion tons per year [1]. These industries are facing challenges such as transitioning to more sustainable operations and energy sources [2]–[4], as well as incorporating artificial intelligence in their processes [5]. This last aspect has proved to have a huge niche and applicability in atmospheric or external corrosion detection in metallic structures from many facilities. For instance, in above-ground storage tanks containing hydrocarbons, pitting corrosion -the main concern- can be identified through machine learning models
to avoid the release of toxic and flammable compounds [6]. It is not only a matter of safety but productivity, since without said models, industries must stop the operation to make regular preventive inspections.

External corrosion of above-ground storage tanks (ASTs) initiates when the painting or coating that covers the metallic structure fails. The most commonly associated failures are pitting, crevice, galvanic, blistering, and uniform corrosion [7]. Pitting is evidenced as leaks, holes, and gouges (Figure 1a), but it can also be associated with microbiologically induced corrosion. Crevice corrosion generates crevices (Figure 1b), galvanic corrosion to section loss (Figure 1c), and blistering corrosion the apparition of bubbles below the painting/coating (Figure 1d). Uniform corrosion (Figure 1e) is the result of homogenous chemical reactions along with the surface extension of a piece. Stress corrosion cracking is less common in storage tanks, however, it can be present at the bottom of tanks when the storage medium becomes saltier, when the end of their service lives is approaching [8] or under the influence of ethanol [9].

![Figure 1. External corrosion in above-ground storage tanks.](image)

The introduction of corrosion modeling in industry has been performed to cover several issues such as corrosion detection, prediction, and classification [10], [11]. In particular, Convolutional Neural Networks (CNNs) have become the most used models for corrosion detection. CNN extracts features from large image datasets without the need for manual extraction typical of shallow machine learning models [12]. The parallelization provided by GPU computing, and the use of cloud computing, have allowed CNNs equal human
capacity for image classification and segmentation [13]. The tasks linked to obtaining CNN models include image acquisition and preprocessing, architecture selection, tuning of hyperparameters, and evaluation.

Most studies available in the scientific literature of CNN’s models for corrosion detection in the oil and gas industry have been developed for pipelines [5], [14]. Custom algorithms detect corrosion and evaluate its magnitude between low, medium, and high after a training process with a dataset of 140K images [15]. It has been used in object detection for pitting, galvanic, and intergranular corrosion risk assessment in oils and gas facility systems, based on 36 images, using data augmentation, and Faster R-CNN with ResNet50 architecture [16]. Offshore pipelines have been studied to deliver models for corrosion detection and classification according to their grade, based on 4K underwater images, achieving an accuracy of 81% [17]. Other approaches for corrosion assessment on metallic surfaces have reached 98.5% performance using more than 67K images for corrosion detection through fine-tuning the VGG_16 model [18]. The structural damage detection method based on Faster R-CNN can also detect concrete cracks, steel high and medium corrosion, bolt corrosion, and steel delamination based on 2.3K images, reaching 89.7% testing performance. Moreover, semantic segmentation by PSPNet and Mask R-CNN has led to corrosion detection with a performance of 73.2%, being implemented in industrial inspections [19]. Likewise, tools such as RustSEG have emerged for segment images for automated corrosion detection without the requirement of per-pixel labeled datasets for training, accomplishing a confidence of 99% [20].

Despite the above studies, there is limited information regarding CNN models for corrosion classification in ASTs. The closest solved corrosion classification problems found in literature are related to metallic corrosion [21], [22]. These works trained CNN models, one to classify between five types of metal corrosion with 563 images, reaching a 93.8% performance, based on a custom CNN model [22], while the other used corrosion detection for four types of failures relayed on Yolov3-tiny architecture to get a performance of 85% [21]. However, none of them represent the specific corrosion issues found in ASTs. In this study, we aim to solve this artificial vision issue by using a small dataset provided by one industrial oil and gas company, using transfer learning pre-trained EfficientNet models to classify between pitting, crevice, galvanic, blistering, and uniform corrosion. This study delivers EfficientNet customized models to be implemented
for external corrosion classification in ASTs. Besides, this development contributes to remote monitoring and the development of safe operations in the oil and gas sector.

2. Corrosion classification approach

2.1 Efficient-Net

Thanks to the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), object detection, image indexing, and image classification the growth of artificial vision has been enriched by the deployment of powerful models’ architectures. These models currently are used for transfer learning and fine-tuning in multiple tasks that require artificial vision. One of the best architectures was recently reached by Efficient-Net models through new scaling methods that uniformly scale all dimensions of depth/width/resolution by using constant ratios, based on MobileNets and ResNet CNNs. In particular, EfficientNet-B7 achieves state-of-the-art 84.3% in top-1 accuracy on ImageNet (see Figure 2), while being 8.4x smaller and 6.1x faster on inference than the best existing CNNs [23].

**Figure 2.** Model Size vs. ImageNet Accuracy [23].
The EfficientNet architecture uses the AutoML MNAS framework to optimize accuracy and efficiency. Moreover, it employs mobile inverted bottleneck convolution (MBConv) to increase the Floating-Point Operations per Second (FLOP) budget. The basic model called EfficientNetB0 is shown in Figure 3, which was scaled up to get the EfficientNetB7 current model [24].

![Figure 3. The architecture of network EfficientNet-B0.](image)

### 2.2 Dataset acquisition and preprocessing

For corrosion classification problems there are no free available data sources on the internet, especially, if dealing with ASTs. In that case, some of the few ways to get the images are from private companies. The images were taken from 5 years of fieldwork during tank inspection. The diversity of the images was just enough to organize 5 files with images corresponding to galvanic, crevice, blistering, pitting, and uniform corrosion. However, due to the abundance of poor-quality, edited, and non-relevant images, the dataset was consolidated with only 800 images. Stress corrosion cracking and other types of corrosion were not identified in the tanks inspected from 2017 to 2023. All the images were labeled and resized to 224x224 pixels in Colab as required for transfer learning with EfficientNet. The 5 files with images carefully selected to describe each class were compressed and deposited as one zip file in Google Drive for being called from Google Colab notebooks.

### 2.3 Data augmentation

Several techniques for data augmentations were applied to gather the highest possible image dataset to avoid overfitting the customized EfficientNet-based models selected (the ones with the smaller number of parameters). Slightly rotation (nearest fill mode), brightness, shear, and zoom changes, as well as minor channel/width/height shifts and horizontal/vertical flips, were considered to increase the dataset. Table 1 shows the distribution of images for training, validation, and testing after data augmentation. 10-15% of the original data was randomly destined for testing, and 15-30% of the remaining for validation, stratifying the data. It is worth mentioning that for all these experiments, a random seed was set in Python to obtain a similar data distribution across runs, to ensure comparable performances.
Table 1. Images of each type of corrosion are used for training, validation, and testing after data augmentation.

<table>
<thead>
<tr>
<th>Type</th>
<th>Original</th>
<th>Training (85/70%)</th>
<th>Validation (15/30%)</th>
<th>Testing (15/10%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pitting</td>
<td>200</td>
<td>1296/1134</td>
<td>26/54</td>
<td>30/20</td>
</tr>
<tr>
<td>Crevice</td>
<td>156</td>
<td>1017/882</td>
<td>20/42</td>
<td>23/16</td>
</tr>
<tr>
<td>Blistering</td>
<td>140</td>
<td>918/792</td>
<td>17/38</td>
<td>21/14</td>
</tr>
<tr>
<td>Galvanic</td>
<td>154</td>
<td>999/603</td>
<td>20/42</td>
<td>23/15</td>
</tr>
<tr>
<td>Uniform</td>
<td>150</td>
<td>981/855</td>
<td>19/40</td>
<td>22/15</td>
</tr>
</tbody>
</table>

2.4 Evaluation metrics

Accuracy, precision, recall, and F1 score (harmonic mean of precision and recall) were the metrics used to evaluate the capacity of generalization of the model. However, to be fear of the slight imbalance of classes in our dataset, we decided to focus our attention on the weighted F1 score during training. The mathematical expressions for such metrics are defined in Equations (1)-(3). Otherwise, we used a confusion matrix to show the performances of the trained models in testing with unseen images.

\[
\text{Precision} = \frac{\sum \text{true positives}}{\sum \text{true positives} + \sum \text{false positives}} \quad (1)
\]

\[
\text{Recall} = \frac{\sum \text{true positives}}{\sum \text{true positives} + \sum \text{false negatives}} \quad (2)
\]

\[
F1 \text{ Score} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \quad (3)
\]

2.5 Implementation details

![Flowchart of the methodology approach used for this work.](image)

We used Google Colab Notebook with a CPU virtual machine of 12.7 GB RAM and 107.7 GB of disk, which can be used without cost. The experiments were performed using
Python (3.10.12), Keras (2.12.0), and Tensorflow (2.12.0). The complete scheme of the methodology used is shown in Figure 4.

3. Results

3.1 Data preprocessing

The images provided for ASTs of petrochemical industries are usually charged with more information than needed to detect corrosion (see Figure 5a). In such cases, the images require to be cropped to not difficult the training of CNN’s models. Besides, as shown in Figures 5b and 5c, some images are taken from far away or counted with various corrosion types, respectively. These images were not used for training since the model had low performance with them due to our data scarcity. Of course, the limitation of this decision is that the obtained model would lead to better inferences when receiving clean images, focused on the corrosion issue, as seen in Figure 1. On the other hand, the images fed to the models had sizes close to those required to carry out transfer learning, therefore, it did not represent a big deal.

![Figure 5](image)

**Figure 5.** Illustration of some of the images received by the company.

3.2 Data augmentation

This technique allowed us to get a better dataset for training and validation. The changes made to the original images for its augmentation added minor changes to them to not create misperception in the key features to be learned by the models (see Figure 6). For instance, the rotation of images generates new space in the original frame that needs to be filled by the nearest pixels, constant pixels, reflected pixels, and so on. For this particular problem, we decided to use the nearest pixel fill mode offered by Keras since the other types gave place to black spaces easy to confuse with black holes from pitting corrosion.
3.3 Models proposed based on EfficientNet architectures

We decided to use EfficientNetB0 and EfficientNetV2B0-B1 models for transfer learning. These last architectures belong to the new family of EfficientNetV2 models, which train much faster than state-of-the-art models while being up to 6.8x smaller, adjusting regularization along with image size (progressive learning) [25]. These selections were based on the relatively small image dataset provided by one petrochemical company to avoid overfitting induced by large trainable parameters (heavier models). These EfficientNet models have not been tried for corrosion classification tasks in the oil and gas industry as far as we know. We initially tried MobileNetV2 for our task since it is a smaller model, but it delivered poor performances and was discarded for deeper analysis.

Our experimental approach intended to first define the best models’ architecture and hyperparameters through transfer learning (TL) to then apply fine tuning (FT). TL is commonly employed to outperform CNNs in image classification tasks [26]; however, FT is well-known for enhancing TL top-1 performances of artificial vision practical problems [27]. The details of the architecture are presented in Table 2 and Table 3.

The selection of the number of intermediate layers to be introduced in our customized CNNs was made based on several experimental runs, as shown in Table 4. At the end of the frozen EfficientNets layers used for TL (features’ extractor), we added a global_average_pooling2D, reducing the computational complexity of subsequent layers. A sequence of inner dense layers (IL) was explored to get an optimum of one dense layer with a ReLU activation function for TL. L1 (Lasso) and L2 (Ridge) regularizations - Elastic Net- were added to this layer (fixed at 0.001), followed by a batch normalization layer, and a 20% dropout layer; for both, TL and FT. In the case of FT, the Dropout layer
was increased to 50% and the intermediate layers were duplicated to reduce overfitting (see Table 3). In both cases, the output dense layer counted with five neurons and a softmax activation function to complete the artificial vision system for the classification of corrosion in ASTs.

Table 2. CNN’s proposed models with 1 intermediate dense layer for TL.

<table>
<thead>
<tr>
<th>Layers</th>
<th>Output shape</th>
<th>No. of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>EfficientNet (B0-V2B0-V2B1)</td>
<td>$7 \times 7 \times 1280$</td>
<td>0</td>
</tr>
<tr>
<td>Global_average_pooling2D</td>
<td>1280</td>
<td>0</td>
</tr>
<tr>
<td>Dense</td>
<td>50</td>
<td>64050</td>
</tr>
<tr>
<td>Batch_normalization</td>
<td>50</td>
<td>200</td>
</tr>
<tr>
<td>dropout</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>Dense_1</td>
<td>5</td>
<td>255</td>
</tr>
<tr>
<td><strong>Total params:</strong> B0 = 4114076; V2B0 = 5983817; V2B1 = 6995629</td>
<td><strong>Trainable params:</strong> B0 = V2B0 = V2B1 = 66405</td>
<td><strong>Non-trainable params:</strong> B0 = 4049671; V2B0 = 5919412; V2B1 = 693.224</td>
</tr>
</tbody>
</table>

Table 3. CNN’s proposed models with 2 intermediate dense layers for FT.

<table>
<thead>
<tr>
<th>Layers</th>
<th>Output shape</th>
<th>No. of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>EfficientNet (B0-V2B0-V2B1)</td>
<td>$7 \times 7 \times 1280$</td>
<td>0</td>
</tr>
<tr>
<td>Global_average_pooling2D</td>
<td>1280</td>
<td>0</td>
</tr>
<tr>
<td>Dense</td>
<td>100</td>
<td>128100</td>
</tr>
<tr>
<td>Batch_normalization</td>
<td>100</td>
<td>400</td>
</tr>
<tr>
<td>dropout</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>Dense_1</td>
<td>50</td>
<td>5050</td>
</tr>
<tr>
<td>Batch_normalization_1</td>
<td>50</td>
<td>200</td>
</tr>
<tr>
<td>Dropout_1</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>Dense_2</td>
<td>5</td>
<td>255</td>
</tr>
<tr>
<td><strong>Total params:</strong> 4183576</td>
<td><strong>Trainable params:</strong> 133705</td>
<td><strong>Non-trainable params:</strong> 4049871</td>
</tr>
</tbody>
</table>

3.4 Mathematical expression behind the proposed architecture

The easiest understanding of CNNs for a general audience can be achieved experimentally by reviewing the website https://poloclub.github.io/cnn-explainer. This interactive page explains what are convolutions, strides, kernels, activations functions, pooling layers, input layers, and flatten layers [28]. However, for the particular functions used for regularization such as L1, L2, batch normalization, and dropout, their mathematical expressions are the following. Equation (4) is associated with the categorical cross-entropy loss function, widely used for multiclass classification problems. We also used categorical focal cross entropy, Equation (5), to explore performances for hard to classy classes; where, $p_i = \text{output if } y_{true} = i, \text{else } 1 - \text{output}$ [29]. Equations (6) and (7) are linked with L1(Lasso) and L2 (Ridge) normalizations, which penalizes the complexity of the models. The batch normalization
regulator aims to fight the internal covariate shift problem. In Equation (8), \( \mu_B \) is the batch mean, \( \sigma_B^2 \) the batch variance, and \( \gamma \) and \( \beta \) are learned during training. Equation (9) is the dropout, which selects neurons to be disabled based on the dropout probability \( p \), rescaling the remained neurons’ weights (\( w_i \)) to \( 1/(1-p) \).

\[
\text{CategoricalCE} = -\sum_{i=1}^{N} y_{true_i} \cdot \log (y_{pred_i})
\]

\[
\text{CategoricalFCE} = \alpha \cdot (1-p)^\gamma \cdot \text{CategoricalCE}(y_{true_i}, y_{pred_i})
\]

\[
\text{loss function} + \alpha \cdot \frac{1}{N} \sum_{j=1}^{N} |w_i|
\]

\[
\text{loss function} + \alpha \cdot \frac{1}{2N} \sum_{j=1}^{N} w_i^2
\]

\[
y_i = \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta
\]

\[
w_i^* = \begin{cases} 
0, & \text{with } p \\
\frac{w_i}{1-p}, & \text{otherwise}
\end{cases}
\]

### 3.5 Hyperparameter tuning

Regarding the hyperparameters not explored in the model, we used the Adam optimizer and 150 epochs. The batch size was explored for 16, 32, and 64 values. The learning rate (Lr) was evaluated for 0.01, 0.001, and 0.0001 magnitudes. When using the categorical focal cross-entropy loss function, the values of gamma and alpha were tested in the ranges (1-4) and (0.25-2), respectively. Moreover, we used EarlyStopping, ReduceLROnPlateau, and ModelCheckpoint callbacks from the Keras library to manage overfitting. EarlyStopping with patience (ESP) of 5 was used for training while 10 for the best models found. For ReduceLROnPlateau, a patience (RP) of 3 and a factor (RF) of 0.25 were destined for training while RP= 5 and RF = 0.50 for the best models. This last hyperparameter fine-tuned the model, reducing its learning rates in small proportions. Finally, the hyperparameter optimization was done manually to reduce the computational cost and time required when covering large ranges of values through Keras Tunner. Table 4 shows the experimental trials (see also supplemental material), where 2 EfficientNetB0-TL models (15/16_replica and 17/18_replica) were selected as the best ones. One is based on the categorical cross-entropy loss and the other on categorical focal cross-entropy. Both models were evaluated with FT, indicating better results when two intermediate dense layers were used in the architectures since it avoided overfitting (see section 3.6).
Moreover, the data partition of 10% for testing, and 30% of the remaining for validation was the best distribution found to reduce the loss values.

**Table 4.** Test performed to explore the best CNN’s model.

<table>
<thead>
<tr>
<th>Batch</th>
<th>Lr</th>
<th>$\alpha$</th>
<th>$\gamma$</th>
<th>ESP</th>
<th>RP</th>
<th>RF</th>
<th>F1 Score</th>
<th>Val loss</th>
<th>IL</th>
<th>Mod</th>
<th>Test /Val</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>0.01</td>
<td>-</td>
<td>5</td>
<td>3</td>
<td>0.25</td>
<td>0.8320</td>
<td>1.1238</td>
<td>1</td>
<td>B0TL</td>
<td>15%T/(30%V)</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>0.01</td>
<td>-</td>
<td>5</td>
<td>3</td>
<td>0.25</td>
<td>0.7959</td>
<td>1.9799</td>
<td>1</td>
<td>B0TL</td>
<td>15%T/(30%V)</td>
</tr>
<tr>
<td>3</td>
<td>32</td>
<td>0.001</td>
<td>-</td>
<td>5</td>
<td>3</td>
<td>0.25</td>
<td>0.7984</td>
<td>1.2244</td>
<td>1</td>
<td>B0TL</td>
<td>15%T/(30%V)</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>0.001</td>
<td>-</td>
<td>5</td>
<td>3</td>
<td>0.25</td>
<td>0.7345</td>
<td>1.3761</td>
<td>1</td>
<td>B0TL</td>
<td>15%T/(30%V)</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>0.0001</td>
<td>-</td>
<td>5</td>
<td>3</td>
<td>0.25</td>
<td>0.7938</td>
<td>2.3206</td>
<td>1</td>
<td>B0TL</td>
<td>15%T/(30%V)</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>0.0001</td>
<td>-</td>
<td>5</td>
<td>3</td>
<td>0.25</td>
<td>0.7553</td>
<td>3.1431</td>
<td>1</td>
<td>B0TL</td>
<td>15%T/(30%V)</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>0.01</td>
<td>-</td>
<td>5</td>
<td>3</td>
<td>0.25</td>
<td>0.8234</td>
<td>0.8297</td>
<td>1</td>
<td>B0TL</td>
<td>15%T/(30%V)</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>0.01</td>
<td>-</td>
<td>10</td>
<td>5</td>
<td>0.25</td>
<td>0.7822</td>
<td>0.8272</td>
<td>1</td>
<td>B0TL</td>
<td>15%T/(30%V)</td>
</tr>
<tr>
<td>9</td>
<td>32</td>
<td>0.01</td>
<td>-</td>
<td>10</td>
<td>5</td>
<td>0.25</td>
<td>0.8135</td>
<td>1.1726</td>
<td>1</td>
<td>B0TL</td>
<td>15%T/(30%V)</td>
</tr>
<tr>
<td>10</td>
<td>32</td>
<td>0.01</td>
<td>-</td>
<td>10</td>
<td>5</td>
<td>0.25</td>
<td>0.8327</td>
<td>1.7406</td>
<td>2</td>
<td>B0TL</td>
<td>15%T/(30%V)</td>
</tr>
<tr>
<td>11</td>
<td>32</td>
<td>0.01</td>
<td>-</td>
<td>10</td>
<td>5</td>
<td>0.25</td>
<td>0.8132</td>
<td>2.7606</td>
<td>2</td>
<td>B0TL</td>
<td>15%T/(30%V)</td>
</tr>
<tr>
<td>12</td>
<td>32</td>
<td>0.01</td>
<td>-</td>
<td>10</td>
<td>5</td>
<td>0.25</td>
<td>0.8115</td>
<td>1.0490</td>
<td>1</td>
<td>V2B0TL</td>
<td>15%T/(30%V)</td>
</tr>
<tr>
<td>13</td>
<td>32</td>
<td>0.01</td>
<td>-</td>
<td>10</td>
<td>5</td>
<td>0.25</td>
<td>0.8141</td>
<td>1.0323</td>
<td>1</td>
<td>B1TL</td>
<td>15%T/(30%V)</td>
</tr>
<tr>
<td>14</td>
<td>32</td>
<td>0.01</td>
<td>-</td>
<td>10</td>
<td>5</td>
<td>0.25</td>
<td>0.8153</td>
<td>0.7639</td>
<td>1</td>
<td>B0TL</td>
<td>10%T/(30%V)</td>
</tr>
<tr>
<td>15</td>
<td>32</td>
<td>0.01</td>
<td>-</td>
<td>10</td>
<td>5</td>
<td>0.50</td>
<td>0.8240</td>
<td>0.7204</td>
<td>1</td>
<td>B0TL</td>
<td>10%T/(30%V)</td>
</tr>
<tr>
<td>16</td>
<td>32</td>
<td>0.01</td>
<td>-</td>
<td>10</td>
<td>5</td>
<td>0.50</td>
<td>0.8152</td>
<td>0.7367</td>
<td>1</td>
<td>B0TL</td>
<td>10%T/(30%V)</td>
</tr>
<tr>
<td>17</td>
<td>32</td>
<td>0.01</td>
<td>0.25</td>
<td>2.0</td>
<td>10</td>
<td>5</td>
<td>0.50</td>
<td>0.8180</td>
<td>0.0702</td>
<td>1</td>
<td>B0TL</td>
</tr>
<tr>
<td>18</td>
<td>32</td>
<td>0.01</td>
<td>0.25</td>
<td>2.0</td>
<td>10</td>
<td>5</td>
<td>0.50</td>
<td>0.8057</td>
<td>0.0744</td>
<td>1</td>
<td>B0TL</td>
</tr>
<tr>
<td>19</td>
<td>32</td>
<td>0.01</td>
<td>0.25</td>
<td>2.0</td>
<td>10</td>
<td>5</td>
<td>0.25</td>
<td>0.8073</td>
<td>0.073</td>
<td>1</td>
<td>B0TL</td>
</tr>
<tr>
<td>20</td>
<td>32</td>
<td>0.01</td>
<td>0.5</td>
<td>2.0</td>
<td>10</td>
<td>5</td>
<td>0.50</td>
<td>0.7994</td>
<td>0.1533</td>
<td>1</td>
<td>B0TL</td>
</tr>
<tr>
<td>21</td>
<td>32</td>
<td>0.01</td>
<td>0.9</td>
<td>2.0</td>
<td>10</td>
<td>5</td>
<td>0.50</td>
<td>0.8060</td>
<td>0.2935</td>
<td>1</td>
<td>B0TL</td>
</tr>
<tr>
<td>22</td>
<td>32</td>
<td>0.01</td>
<td>0.9</td>
<td>1.5</td>
<td>10</td>
<td>5</td>
<td>0.50</td>
<td>0.7872</td>
<td>0.3724</td>
<td>1</td>
<td>B0TL</td>
</tr>
<tr>
<td>23</td>
<td>32</td>
<td>0.01</td>
<td>1.0</td>
<td>1.5</td>
<td>10</td>
<td>5</td>
<td>0.50</td>
<td>0.8039</td>
<td>0.4089</td>
<td>1</td>
<td>B0TL</td>
</tr>
<tr>
<td>24</td>
<td>32</td>
<td>0.01</td>
<td>1.0</td>
<td>1.0</td>
<td>10</td>
<td>5</td>
<td>0.50</td>
<td>0.7811</td>
<td>0.4971</td>
<td>1</td>
<td>B0TL</td>
</tr>
<tr>
<td>25</td>
<td>32</td>
<td>0.01</td>
<td>1.0</td>
<td>2.0</td>
<td>10</td>
<td>5</td>
<td>0.50</td>
<td>0.8087</td>
<td>0.3189</td>
<td>1</td>
<td>B0TL</td>
</tr>
<tr>
<td>26</td>
<td>32</td>
<td>0.01</td>
<td>1.0</td>
<td>3.0</td>
<td>10</td>
<td>5</td>
<td>0.50</td>
<td>0.8040</td>
<td>0.2364</td>
<td>1</td>
<td>B0TL</td>
</tr>
</tbody>
</table>
3.6 Training and validation results

The models that showed the best performances during training reached 82% in the F1 score metric, surpassing the rest of the models trained by only 2% (see Table 4). We attach that behavior to the EfficientNet architecture used for transfer learning and to the data quality. Note that more advanced models such as EfficientNetV2B0 and V2B1 did not suit our images better than version B0; that is, the pre-trained features learned by the simplest model align well with our classification problem. The best B0TL models shared the following hyperparameters: one intermediate dense layer with 50 units, learning rate = 0.01, batch size = 32, ESP = 10, RP = 5, RF = 0.5 (see runs 15-18 in Table 4). Their difference was the loss function. As noted in Table 4, the categorical focal cross entropy function with alpha = 0.25 and gamma = 2, leads to lower losses. Nevertheless, it is not necessarily associated with more capabilities of generalization of the model. When using an alpha value of 0.25 the training focused the most on the minority class of the dataset, in our case, the blistering class; but paying less attention to the most abundant class, pitting [30]. However, if the performance of the model is stuck due to the majority class, this loss function, with these hyperparameters, does not offer the best solution for the desired task. Therefore, the good approaches to zero of the focal loss can be an illusion. Figure 7 shows the limitations that still face this model for classifying corrosion types, despite its final loss of 0.0702. Meanwhile, the model focused on the categorical cross-entropy function showed similar results (see Figure 8), but for a final loss value of 0.7204. Then, this last model, with a final loss value ten times higher than the based on the focal loss, is more in line with the reality of the model when tested with never-before-seen images.
Figure 7. Confusion matrix for the TL model trained using a focal function with the lower loss value.

Figure 8. Confusion matrix for the TL model trained categorical cross-entropy function with the lower loss value.
On the other hand, blistering and pitting corrosion were identified as the hardest classes for the classification problem addressed. As seen in Figure 8, the blisters formed below a coating can be easily confused with the pits caused by microorganisms at the bottom of the tank. Figures 1a and 6 show how hard it can be, even for a human eye, to distinguish between these two classes. This limitation can also be observed in the confusion matrix of other runs in supplementary material. Hence, it is not only a problem of the loss function used but also of data scarcity to train the models, since microbial corrosion is a not frequent type of pitting in the studied ASTs. The 25% of the pitting dataset corresponded to microbial pitting.

Being aware of the above-mentioned limitations we decided to select the model with the higher final loss value, the one that used the categorical cross-entropy function. Figure 9 shows its corresponding loss and F1-score curves during training and validation of the B0TL model. The optimal model performance was found around epoch-51 since in the following epochs it begins to overfit. The classes uniform and galvanic corrosion deliver the best performances while pitting the worst (see Figure 10) in coherence with previous statements. When the B0TL model converges the final learning rate reached is 6.25e-04, after 4 decreasing steps from 1e-2 (see supplemental material S39).

![Figure 9. B0TL loss and F1-score curves during training and validation.](image)

Once the TL model, FT was introduced to explore better performances. This approach consists of two phases; one called warming up and a second one called adjustment. In the first phase, all layers were trained with a 1e-2 learning rate. As our task is not the same as ImageNet, we decided to learn features relatively quickly for over 10 epochs, without freezing any layers. In the next phase, we freeze the lower layers (feature extraction layers) and adjust the upper layers (task-specific layers) to our data using a learning rate of 1e-3, and the same hyperparameters employed to train the B0TL model (see runs 31-38 in Table 4). Two intermediate dense layers with 100 and 50 units, respectively, were optimal to avoid overfitting.
**Figure 10.** Class-wise F1-score over epochs for the B0TL model.

**Figure 11.** B0FT loss and F1-score curves during training and validation.

Figure 11 shows its corresponding loss and F1-score curves during training and validation of the B0FT model. The optimal model performance was found around epoch 57. The classes uniform and galvanic corrosion deliver the best performances while blistering and pitting the worst (see Figures 12 and 13). Crevice corrosion gets misclassified in several cases due to its similar appearance to galvanic corrosion when dealing with bolted joints or with pitting when coating fails around the junctions (see supplemental material S42).
However, the final loss reached for this model was a bit lower than after training the B0TL model (see supplemental material S36 y S41). As argued before for the B0TL model, the performances of the B0FT model suggest that more data are required to overcome the limitations of high similarity between some of the classes. When the B0TL model converges the final learning rate reached was 2.5e-04, after 2 decreasing steps from 1e-3 (see supplemental material S40).

Figure 12. Class-wise F1-score over epochs for the B0FT model.

Figure 13. Confusion matrix for the FT model trained with categorical cross-entropy function with the lower loss value.
3.7 Model testing and inference

Testing results were obtained by using the B0TL and B0FT models, with categorical cross-entropy loss functions, due to their outstanding performances in the training and validation stages. As shown in Table 5, B0TL reached a mean F1_score (weighted) of 85% while B0FT gained 8% over TL. These numbers exceed the values found in the validation phase, probably related to the particular distribution of the validation dataset favored the learning of important patterns. In addition, the pitting and blistering corrosion classes delivered the lowest performances, as expected from the training process.

<table>
<thead>
<tr>
<th></th>
<th>TL Precision</th>
<th>TL Recall</th>
<th>TL F1Score</th>
<th>FT Precision</th>
<th>FT Recall</th>
<th>FT F1Score</th>
<th>Img.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blistering Corrosion</td>
<td>0.78 ± 0.07</td>
<td>0.75 ± 0.04</td>
<td>0.93 ± 0.07</td>
<td>0.86 ± 0.00</td>
<td>0.89 ± 0.03</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>Galvanic corrosion</td>
<td>0.93 ± 0.01</td>
<td>0.84 ± 0.04</td>
<td>0.94 ± 0.06</td>
<td>0.93 ± 0.00</td>
<td>0.94 ± 0.04</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>Crevice corrosion</td>
<td>0.81 ± 0.02</td>
<td>0.94 ± 0.00</td>
<td>0.92 ± 0.03</td>
<td>0.97 ± 0.03</td>
<td>0.94 ± 0.00</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>Pitting corrosion</td>
<td>0.92 ± 0.03</td>
<td>0.78 ± 0.03</td>
<td>0.84 ± 0.01</td>
<td>0.98 ± 0.03</td>
<td>0.93 ± 0.00</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>Uniform corrosion</td>
<td>0.80 ± 0.01</td>
<td>0.94 ± 0.07</td>
<td>0.86 ± 0.02</td>
<td>0.87 ± 0.00</td>
<td>0.93 ± 0.00</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.85 ± 0.01</td>
<td></td>
<td></td>
<td>0.85 ± 0.01</td>
<td>0.93 ± 0.02</td>
<td></td>
<td>80</td>
</tr>
<tr>
<td>Macro Avg.</td>
<td>0.85 ± 0.01</td>
<td>0.85 ± 0.01</td>
<td>0.84 ± 0.01</td>
<td>0.93 ± 0.02</td>
<td>0.93 ± 0.02</td>
<td></td>
<td>80</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>0.85 ± 0.01</td>
<td>0.85 ± 0.01</td>
<td>0.85 ± 0.01</td>
<td>0.94 ± 0.02</td>
<td>0.93 ± 0.02</td>
<td>0.93 ± 0.02</td>
<td>80</td>
</tr>
</tbody>
</table>

Figure 14. Inferences of the B0FT model for an image with pitting corrosion.

We used the B0FT model to make inferences and observe how it worked. Figure 14 shows how the pitting induced by microbial corrosion is confused with blisters while in less
proportion to uniform corrosion, due to the homogeneous background. Besides, Figure 15 indicates a correct classification of blistering with a small proportion of pitting, that in fact, may be present in the image. Likewise, an image with multiple types of corrosion was explored (see Figure 16), pointing out that our model bases its answers on the type of corrosion prevalent. In this particular case, opting for crevice corrosion, while galvanic corrosion and pitting are also present. Therefore, it is a reminder that artificial vision should work together with an expert eye to confirm and enhance the inferences made by the models.

![Figure 15](image1.png)  
**Figure 15.** Inferences of the B0FT model for an image with blistering corrosion.

![Figure 16](image2.png)  
**Figure 16.** Inferences of the B0FT model for an image with crevice corrosion in a cathodic current protection (CCP) connection system.
4. Discussion and comparison with the state-of-the-art

While many authors have dealt with deep learning algorithms to deal with corrosion detection issues [15], [21], [31]–[35], few scientists have focused on classifying corrosion types in different environments [13], [16], [22], [36], [37]. More importantly, no work has been done to classify corrosion types for ASTs, and using EfficientNet architectures. However, we compared our results with the available models trained through CNNs based on metallic infrastructures.

Petrica and coworkers gathered 1300 images of 256x256 pixels for "rust" and 2200 for the "non-rust" class, destining 80% for training and 20% for validation, to solve a binary classification problem. They used fine-tuning with the model bvlc_reference_caffenet, developed by the Berkeley Vision and Learning Center, obtaining a performance of 88% for a learning rate of 5e-5 [37]. For their part, by using convolutional neural networks, Ejimuda and collaborators used TL with the ResNet model to solve a multiclassification problem with object detection; considering, intergranular, galvanic, and pitting corrosion as classes. They used data augmentation since they only counted 36 images from the internet, reaching 336 images. 70% of de data was used for training and 30% for validation, achieving 83.3% of performance [16]. The main limitation of both models compared to this research was the robustness of the model and the data scarcity.

Table 6. Comparison with similar approaches.

<table>
<thead>
<tr>
<th>No. of Images</th>
<th>Model</th>
<th>Best F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Holm et al [36]</td>
<td>9300</td>
<td>VGG-16</td>
</tr>
<tr>
<td>Ahuja et al [22]</td>
<td>2000</td>
<td>VGG-16</td>
</tr>
<tr>
<td>This work</td>
<td>4562</td>
<td>EfficientNetB0</td>
</tr>
</tbody>
</table>

In a closer approach to our work, Holm et al solved a multiclass corrosion problem with the categories not-corrosion, paint flaking, and two red corrosion types related to bridge constructions. They counted with a large image dataset after data augmentation of 9300 images. The data partition was established at 80%, 10%, and 10% for training, validation, and testing, respectively. They used 4 pre-trained models: AlexNet, GoogLeNet, ResNet-50, and VGG-16. This last model delivered the best F1 score, accomplishing 95.53% [36]. Finally, the most similar research was made by Ahuja et al, who solved a multiclassification problem with six metallic corrosion categories, differing from our work by not considering the blistering corrosion. After data augmentation, the authors collected around 2000 images of 512x512 pixels. Also, a batch size of 8, an input size of 224x224, 10 epochs, an Adamax optimizer, and a learning rate of 1e-2 were implemented. A VGG-16, pre-trained and customized network was optimized to improve the classification accuracy to obtain a 93.8% in the F1-score metric [22].

Compared to our model, Holm’s approach counted the double images to accomplish their goal [36]. The study of Ahuja lacks a deeper analysis and reliability since, despite employing half of our images and a model with 8 times more trainable parameters (VGG-16), they achieved a very high performance [22]. We addressed a more complex classification task with two challenging classes and, through a clear and robust
methodology, achieved a 94% performance in the best trial. Then, EfficientNetB0 proved to be an excellent choice for corrosion classification tasks, allowing TL and FT approaches due to its small set of trainable parameters and lower computational cost.

5. Conclusions

This article presents an exploration of the performance of EfficientNet models B0, V2B0, and V2B1 for corrosion image classification in above-ground storage tanks. The images used were received from a petrochemical company and managed to gather around 5000 images after data augmentation. The types of corrosion of interest identified in the company were galvanic, interstitial, uniform, pitting, and blistering corrosion. The model with the best performance was obtained by fine-tuning EfficientNetB0, which yielded an F1-score of 94% in the best of cases. The blistering and pitting corrosion classes represented an obstacle for the trained model due to the limited amount of data used. In particular, these two types of corrosion represented a challenge because they have similar shapes, colors, and textures when the pitting corrosion is microbial. However, the model obtained has great potential for its application in automatic inspections of metallic infrastructures with corrosion problems similar to those presented in ASTs. Likewise, this model can be extended for other types of corrosion, taking advantage of its low computational cost and its low number of trainable parameters compared to other models available from ImageNet.

List of abbreviations

Not applicable.

Availability of data and materials

The database of images was provided by a private industry and it is not available.

Competing interests

The authors declare no conflict of interest.

Funding

No funding to declare.

Authors' Contributions

Alviz-Meza A. carried out the literature review, data analysis, and discussion of results, and drafted the manuscript. Peña-Ballesteros Dario Y. aimed to extract data and its preprocessing step. Alviz-Meza A. communicates with the editor. All authors read and approved the final manuscript.

Acknowledgments

Not applicable.
References


