sensors

MDPI

Article

Autonomous Image-Based Corrosion Detection in Steel Structures Using Deep Learning

Amrita Das [1,*], Sattar Dorafshan [1] and Naima Kaabouch [2]

[1] Department of Civil Engineering, College of Engineering & Mines, University of North Dakota, Grand Forks, ND 58202, USA; sattar.dorafshan@und.edu
[2] Department of Electrical Engineering, School of Electric Engineering & Computer Science, University North Dakota, Grand Forks, ND 58202, USA; naima.kaabouch@und.edu
* Correspondence: amrita.das@und.edu

Abstract: Steel structures are susceptible to corrosion due to their exposure to the environment. Currently used non-destructive techniques require inspector involvement. Inaccessibility of the defective part may lead to unnoticed corrosion, allowing the corrosion to propagate and cause catastrophic structural failure over time. Autonomous corrosion detection is essential for mitigating these problems. This study investigated the effect of the type of encoder–decoder neural network and the training strategy that works the best to automate the segmentation of corroded pixels in visual images. Models using pre-trained DesnseNet121 and EfficientNetB7 backbones yielded 96.78% and 98.5% average pixel-level accuracy, respectively. Deeper EffiecientNetB7 performed the worst, with only 33% true-positive values, which was 58% less than ResNet34 and the original UNet. ResNet 34 successfully classified the corroded pixels, with 2.98% false positives, whereas the original UNet predicted 8.24% of the non-corroded pixels as corroded when tested on a specific set of images exclusive to the investigated training dataset. Deep networks were found to be better for transfer learning than full training, and a smaller dataset could be one of the reasons for performance degradation. Both fully trained conventional UNet and ResNet34 models were tested on some external images of different steel structures with different colors and types of corrosion, with the ResNet 34 backbone outperforming conventional UNet.

Keywords: steel structure; corrosion; semantic segmentation; transfer learning; artificial intelligence

check for updates

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

Academic Editor: Yi Qin

Received: 10 May 2024
Revised: 27 May 2024
Accepted: 29 May 2024
Published: 4 June 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Corrosion is defined as changes in metal properties due to reactions with a surrounding corrosive environment, which deteriorates the metallic object's functionality [1]. External or internal corrosion can reduce any steel structure's lifespan [2].

Many types of defects occur in highway ancillary structures; however, corrosion is the most common [3]. Highway ancillary structures are exposed to weathering effects such as snow and rain, which lead to corrosion over time. The annual cost of corrosion damage estimated by the National Association for Corrosion Engineers (NACE) in 2016 was USD 2.5 trillion, which is equivalent to 3.4% of global GDP; however, current corrosion inspection techniques can save 15–35% of this estimated amount [4]. Among numerous corrosion monitoring techniques, ultrasonic testing is the most common one that can be used to get an instantaneous indication of cracks or corrosion. In this case, the traveling speed of the wave is generally considered for structural condition assessment. However, the authors of [5] reported a discrepancy in results in the case of multiple layers on a substrate. However, the performance of acoustic emission and eddy current technology in corrosion detection is not dependent on the number of layers and the size of defects [6]. These two methods may suffer from the recording of noisy signals and sensitivity to lift-off effects [6,7]. Regularly implemented inspection methods are not typically non-contact but non-destructive techniques (NDTs) that require a human inspector on-scene.

Inspection accuracy depends on the inspector's expertise and the accessibility of the defect location [8]. In addition, the road closures and traffic controls required for the maintenance and protection of traffic (MPT) could make inspections challenging to perform [9].

Many departments of transportation (DOTs) use unmanned aerial systems (UASs) for bridge inspections, allowing them to mitigate the problems related to manual inspections [10], which might be followed by the use of conventional image processing methods to detect steel corrosion [11–14]. The problems related to conventional image processing techniques are inaccurate user-defined parameters and adverse lighting condition effects [11]. The effect of over-pixelated regions in images may make conventional image-based methods less accurate [11]. Most researchers have used images collected in the lab to solve this problem, since the environment is controlled; however, this technique may produce a less generic defect detection model [12–14]. Using artificial intelligence (AI) as an alternative to the existing manned corrosion detection techniques can be a feasible solution to this problem.

Deep learning methods have already achieved a high degree of success in fields such as civil engineering, health care, cybersecurity intelligence, and smart cities [15]. This method's intelligent performance and learning capacity from big data make it a key component in building smart data-driven systems [16]. Deep learning and machine learning are the two primary branches of artificial intelligence. The primary concept of artificial intelligence is the transfer of intelligence by duplicating the architecture of the human brain [17]. The primary difference between deep learning and machine learning is network depth and network architecture complexity [16]. Feature extraction in deep learning is also autonomous through the convolution layers. K-mean clustering, support vector machines, random forest, and decision tree are some user-defined shallow structures used for feature extraction in machine learning [18].

A machine learning algorithm may be overwhelmed by the exponential growth in data volume, unlike deep learning models [16]. The exponential development of new deep learning models could be overwhelming for structural defect detection, where annotated datasets are rare and expensive to create. Some already developed model architectures, such as Faster RCNN [19,20] and Mask RCNN [21], have been used by researchers to detect corrosion in the structural elements of steel bridges for this reason. Other deep convolution neural networks, such as ResNet50 [22,23], AlexNet [23], VGG-16 [23], and GoogleNet [23], have been used to detect metallic corrosion. AlexNet outperformed the other models in terms of correct corroded image predictions; however, the training dataset was imbalanced, and the proportion of damage in the test dataset was not reported. The authors of [24] developed a shallow convolutional neural network to predict corrosion in visual images using sliding windows of different sizes and scored a 98% recall value for a window size of 128 × 128 megapixels. Bolton et al. [25] tested five image classification models to detect corrosion in steel structures by initializing the random weights, including VGG16, ResNet50, AlexNet, Bastian Net, and ZF Net. VGG16 and ResNet50 were used in both transfer learning and fully trained modes. Two pre-trained models, VGG16 and ResNet 50, yielded a 98% recall value in detecting metallic corrosion in 900 images.

Researchers have come to a consensus on the suitability of models for different types of corrosion. For example, Jin et al. [19] suggested using Faster RCNN to detect corrosion; however, Rahman et al. [26] suggested using a supervised pre-trained semantic segmentation model, DeepLab, to detect corrosion at various levels instead of using an object detection model such as Faster RCNN, since a bounding box might not be suitable for corroded pixels with an irregular shape. The DeepLab model was tested on 100 images and detected corroded pixels with an accuracy of 72%; however, the model accurately predicted 93% of the non-corroded pixels. Forkan et al. [27] used ensemble networking with shallow networks, such as VGG16, to identify corrosion and Mask-RCNN to separate the region of interest. Ahuja et al. [28] established that residual UNet could efficiently detect pitting corrosion with a 98% recall value, which was higher than RCNN and SVM. Bastian et al. [29] used modified ZFNet to detect and localize different levels of corrosion in

a pipeline with a 98.8% classification accuracy. Huang et al. [30] developed two lightweight models with residual blocks by modifying ShuffleNetV2 to segment metallic corrosion with a more than 85% mean IOU. Researchers have also fine-tuned some layers of pre-trained networks, such as ZFNet [31], PSPNet [32], DeepLab [32], and SegNet [32], to improve model performance.

Corrosion of industrial steel structures is also a serious matter to consider. Pre-trained PSPNet and Mask RCNN efficiently separated corroded pixels in a steel structure, with mean IOU values of 84.1% and 86.4%, respectively [33]. Katsamenis et al. [34] revealed that the Mask RCNN semantic segmentation model outperformed a fully convolution neural network and UNet, an encoder–decoder network, in terms of F1 score. Jiang et al. [35] used fusion attention UNet to detect corrosion in a steel box girder. Woo et al. [36] added a fusion module in place of the skip connection and a convolution block attention module (CBAM) to create a model they called modified UNet, which had a 77.75% mean IOU [35].

Many researchers [35,37] have compared UNet's performance to that of existing convolutional neural networks, including AlexNet, Vgg16, and GoogleNet, to detect corrosion. The performance of modified UNet with the inclusion of a custom layer has also been investigated [35]. The purpose of this study is to evaluate the performance of UNet as a corrosion detection semantic segmentation model with different backbones under transfer learning and fully trained modes. The motivation of choosing a U-shaped network for this investigation is pixel-level accuracy in corrosion detection [35,37] and the reduced number of images required for training [36]. Moreover, transferring the contextual information to the expansion path enables the model to generate a precise segmentation map [37]. There are several deep learning libraries and resources [38], such as Pytorch [39] and TensorFlow [39]. The authors used Keras, a high-level neural network library built on the TensorFlow library, to train the semantic segmentation model.

2. Theoretical Concept of Artificial Neural Network

2.1. General Architecture of Deep Learning Models

A deep learning model's skeleton is based on human neurons. The fundamental idea of a deep learning network unit is that an input (x) is weighted by w and bias (b), then summed. Here, bias (b) is a scalar value; however, the input (x) and weight (w) are vectors. The whole process can be described using Equation (1) [40]:

$$y = \varphi\left(w^T x + b\right) \tag{1}$$

where φ is the activation function that executes non-linear transformation on the outputs from the convolution layers.

All neurons are generally interconnected with the neurons of the next layer. Each connection represents a parameter in the network. Local connectivity between the neurons reduces the number of parameters [40]. For example, the chosen kernel dimension is $3 \times 3 \times 1$, and the number of filters is 1 for an input size of $10 \times 10 \times 1$; therefore, the weight in a single filter sized 3×3 would be 9, bias = 1, and the total number of parameters from each filter is (9 + 1) or 10. A kernel is shared between the local neurons to collect the feature information. All accumulated information from the neurons is stored in a matrix, which is denoted as an activation map. The number of activation maps depends on the number of kernels, which are defined in the hyperparameters. The transferring of information is referred to as weight sharing in a convolutional neural network [41]. This working principle of convolution layers enables the network to handle large datasets. A simplified representation of a deep neural network structure is depicted in Figure 1.

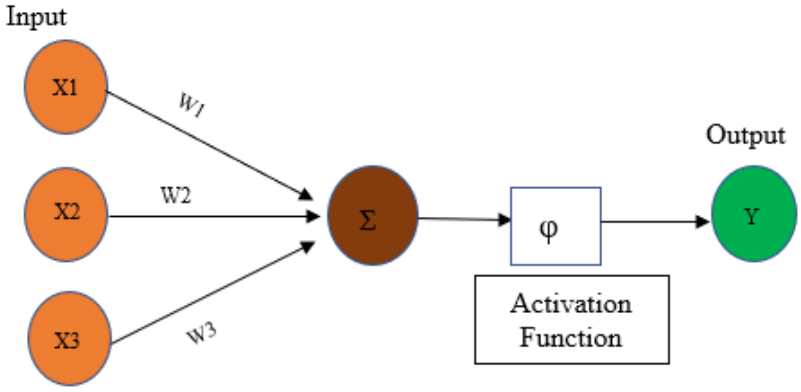


Figure 1. Simplified representation of a deep neural network architecture.

2.2. Convolution

Neural networks extract features from the input through the convolution layers by applying filters [41]. Filters are composed of differently sized kernels. Kernels operate in an area are of same size as the subarrays from the input data. Kernel weights can be initiated with random values and later updated as the training progresses. The weight values can be initiated randomly, or the pre-trained weights can be adopted in a pre-trained network. Kernels with specific sizes slide over the input image. The dot product is computed between each number of the kernel as the kernel moves, and the overlapped element of the input image is computed. The summation of the results from each multiplication identifies the output feature map (Figure 2a–c). This process is then repeated for the entire image. Kernel size is typically 3 × 3 but can be 5 × 5 or 7 × 7 [41].

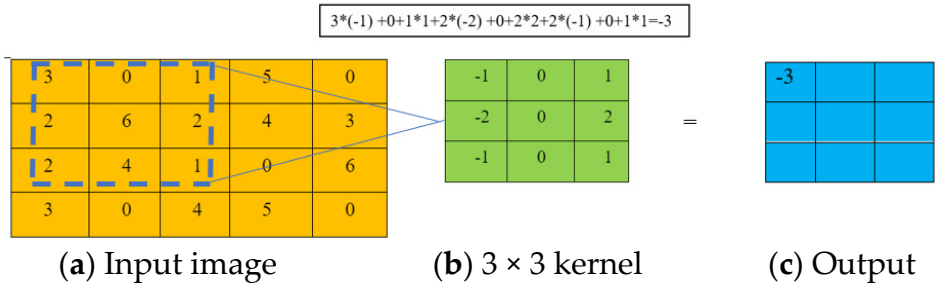
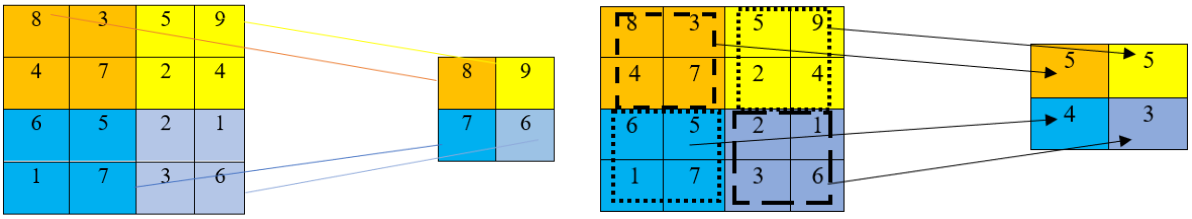


Figure 2. (**a**–**c**) Schematic presentation of convolutional operations.

2.3. Pooling Layer

Inserting a pooling layer between two convolutional layers plays a vital role in reducing the dimensionality of the output [42]. Pooling layers reduce the output dimension and the chance of overfitting [43]. The output image from one convolution layer is divided into small blocks during the pooling step, and a value from each block is selected to generate a compressed feature map. The two most used pooling methods are average pooling [44] and max pooling [45]. There may be one or more fully connected layers after the convolution and pooling layers [46–48]. Local features from the previous output are summed into global features by the fully connected neurons in these layers.

A sliding window passes over the input and feeds the window number to a pooling function. A pooling function can be classified as max and mean pooling [49]. The maximum value from each extracted patch is stored as output and ignores all other smaller values in a max-pooling operation. Max and average pooling operations with a 2 × 2 filter and a stride of 2 are described in Figure 3a,b. Plane dimensions, such as height and width, are downsampled by a factor of two during the pooling operation.



(**a**) Max pooling (**b**) Average pooling

Figure 3. (**a**,**b**) Schematic presentation of different pooling operations.

2.4. ReLu Activation Function

Neural networks cannot manage complex features without an activation layer such as Tanh, Sigmoid, Leaky ReLu (Rectified Linear Unit), or ReLu. Among these, ReLu has become popular because of its simplicity. The main advantage of ReLu is that it does not activate all neurons together, unlike the other activation functions. Equation (2) states the functionality of ReLu such that if the value is less than 0 it is replaced with 0 [50], which means that the neurons become deactivated for negative output values.

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x > 0 \end{cases} \tag{2}$$

2.5. Learning Mode of Deep Learning Models

Deep learning models can be trained in two learning modes, namely transfer learning and fully training mode. Full training mode refers to the stage when the models are trained from scratch; however, there are two types of approaches in transfer learning, namely (a) using a pre-trained model as a classifier and (b) including a fine-tuning step when the model is trained on the specific dataset [16]. Deep learning models typically need a large dataset compared to conventional machine learning models [18], which could limit their applications when these datasets are scarce or expensive to generate. It is also sometimes difficult to collect training data that have the same features as the test data [51]. Using a pre-trained model can solve this issue if the training dataset is relevant to the test set [52]. Transferring knowledge from an already trained model will save time and optimize the model's learning.

Transfer learning has already been tested for steel defect detection [53]. The authors of [51] described transfer learning in two ways, namely as homogenous and heterogeneous. In the case of homogeneous transfer learning, the feature space of the source domain (f_s) is the same as the feature space of the target domain (f_t). Otherwise, this would be heterogeneous transfer learning. Transfer learning has its benefits; however, it could have adverse effects due to the irrelevance between the source and target domain [54]. This type of knowledge transfer is negative transfer learning. Negative transfer learning could be a result of class imbalance and the conditional difference between the source and target domains [55].

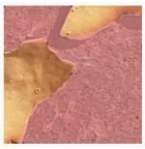
3. Methodology

This section first presents how the dataset was constructed. The learning efficiency of any machine learning model depends on the input dataset. Deep learning models have more layers than machine learning models, which help in improving the learning process. Despite this, an adequate number of representative training datasets still plays a significant role in artificial intelligence model performance, regardless of the number of layers [56,57]. Building up a network with a sufficiently large and representative dataset is challenging. Data augmentation can be a feasible solution for enabling a model to learn better [53]. The developed data augmentation scheme and metrics used for evaluation are reviewed below, and implementation details are provided.

3.1. Dataset Construction and Annotation

No datasets related to corrosion damage segmentation were available prior to this work; therefore, a new dataset was constructed from scratch. A cellphone camera (resolution: 13 megapixels) was used to capture 300 images of four in-service traffic poles with heights of 7 m between 10 a.m. and 12 p.m. in sunny conditions. The poles were in Grand Forks, North Dakota. These poles were later replaced due to the presence of severe corrosion. The image size was 2322 pixels by 4128 pixels. The images were collected for differing degrees of corroded and sound regions with different background scenes. Some representative images were selected for annotation after collection. The Python label-studio image labeler was used for annotation. All representative images were split into sections with dimensions of 227 by 227 pixels. A total of one hundred pieces with corroded pixels were used for annotation (Table 1). Only the corroded pixels were annotated; therefore, the method was semantic segmentation. Two other datasets were prepared with images collected by the authors using UAS and the Internet for qualitative model testing [58].

Table 1. Examples of data annotation.

Original Image	Annotated	Ground Truth (Binary)
		

3.2. Data Augmentation

Some image augmentation libraries, such as imgaug [59], torchvision [39], Augmentor [60], CLoDSA [61], SOLT [62], and Automold [63], have been used over the last few years. The disadvantage of these libraries is the focus on the specific dataset/domain. Buslaev et al. [64] developed an augmentation tool named Albumentaion, which can provide versatility in image augmentation. The authors used the Python Albumentation library for data augmentation. Python LabelStudio in python 3.8 platform was used to semantically generate 100 binary masks from 100 representative images. The images and the masks were augmented to 2000 each, for a total of 2000 images and 2000 masks. Horizontal flip, vertical flip, translation, and grid distortion were used for augmentation (Table 2). The optimum number of augmentations per image was tested, and the optimum number range of the augmented images for each image was 20–40. The number of augmented images from one image was 20. Changes in the images were difficult to identify visually when the number of augmentations reached 40.

Table 2. Results obtained from different augmentation operations of the Albumentation library.

Original	Vertical Flip	Horizontal Flip	Random Rotate 90	Transpose	Grid Distortion
					
					

3.3. Studied Model Architecture

In this study, UNet, which was developed for medical image segmentation [65], was used as a corroded pixel identifier with a different mode of training and pre-trained backbone. The details of different UNet architectures are discussed in the following sections.

3.3.1. Encoder

The encoder consists of contraction blocks. The architecture of each block is similar to the conventional convolutional network (Figure 4). This study used three state-of-the-art networks, namely ResNet34, DenseNet121, and EfficientNetB7, as backbone feature extractors. The basic architecture of these convolutional neural networks is discussed below.

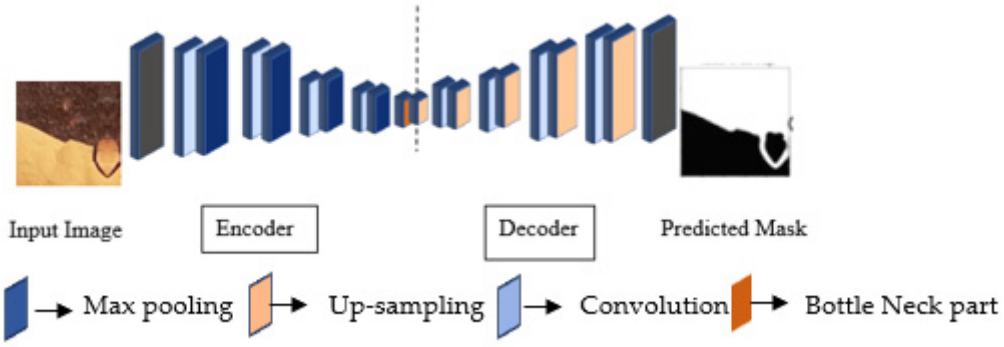


Figure 4. Schematic diagram of the U-shaped architecture of the encoder–decoder network.

ResNet34

ResNet 34 was introduced by He et al. [66]. The primary objective of this innovation was to reduce the difficulties in deeper neural network training. In a conventional convolution neural network, the output from the Nh layer is forwarded to the $(N+1)$th layer [67]; then, the output image is $X_N = H_N\,(X_N - 1)$. In the case of ResNet, a residual block was developed, which can be expressed using Equations (3) and (4).

$$Y1 = h(X_N) + F(X_N, W_N) \tag{3}$$

$$X_{N+1} = f(Y1) \tag{4}$$

where X_N and X_{N+1} are the input and output of the Nth unit, respectively; F is a residual function; and W_N is the weight in the Nth layer. The term $h(X_N) = X_N$ is identity mapping, and f is a ReLU [68]. This residual block helps in performing identity mapping without any computational complexities; therefore, it is termed a shortcut connection [67] (Figure 5a).

DenseNet121

DenseNet121 was developed by Huang et al. [67], with each layer connected to every other layer in a forward-feeding mechanism. DenseBlock (Figure 5b) is the primary building component of DenseNet121 [68]. Feature maps from the preceding layers are concatenated and passed to the subsequent layers with the current layer's feature map to continue the feed-forward mechanism. For example, X_0, X_1, X_2....., and $X_N - 1$ are the feature maps from all preceding layers of the Nth layer, and input of this layer can be expressed by Equation (5) [64].

$$X_N = H_N\,(X_0,\ X_1,\ X_2,\ \ldots\ldots,\ X_{N-1}) \tag{5}$$

where H_N represents the Nth layer, X_N is the output of the Nth layer, and $(X_0,\ X_1,\ X_2,\ \ldots\ldots,\ X_{N-1})$ reflects the concatenation operation.

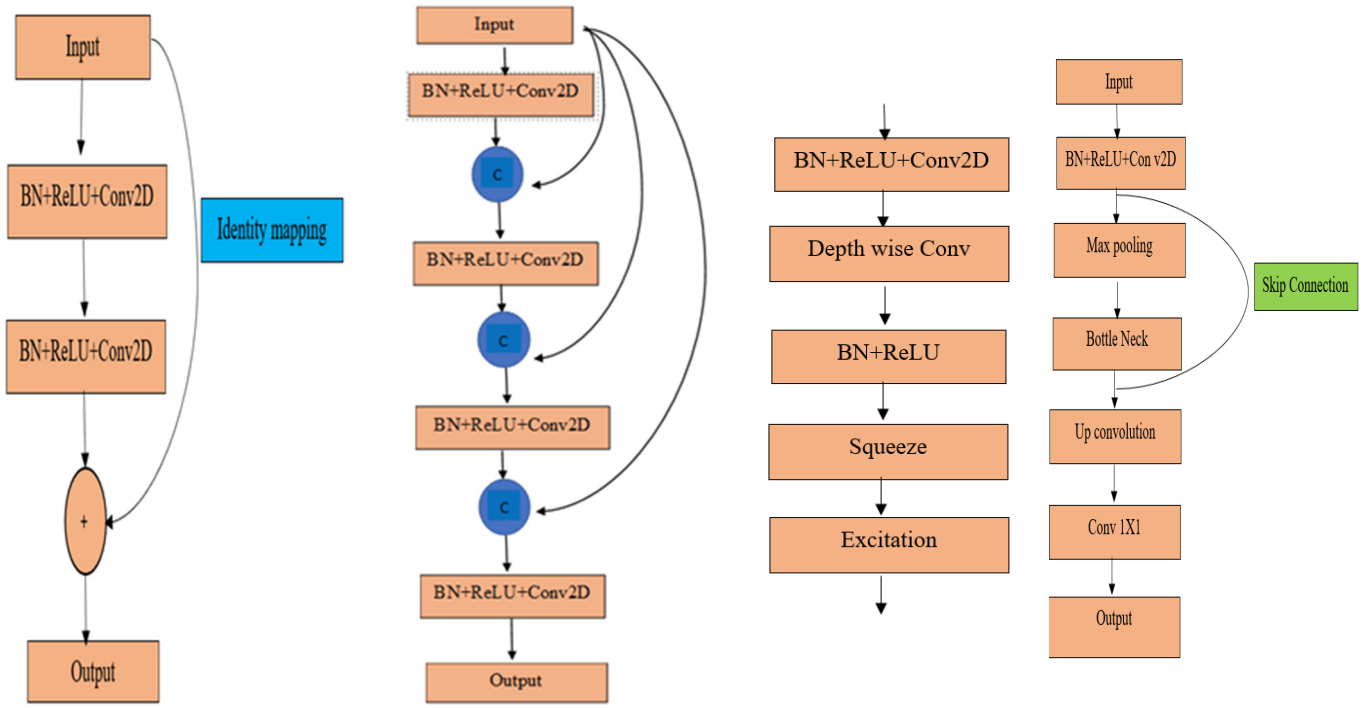


Figure 5. Unit block: (**a**) ResNet34, (**b**) DenseNet121, (**c**) EfficientNetB7, and (**d**) UNet.

EfficientNetB7

EfficientNetB7 is a modified version of EfficientNetB0. A compound scaling method has been implemented on a baseline network, EfficientNet B0, to confirm the uniform scaling up of depth, width, and resolution, developing the advanced version known as EfficientNetB7 [69]. Mobile inverted bottleneck convolution (MBConv) is the basic building block of any EfficientNet architecture with squeeze and excitation optimization (Figure 5c) [70,71]. The output from the depth-wise convolution is forwarded to a new channel by pointwise convolution. Two layers of EfficientNet blocks are used for to squeeze and extend the channels [61]. Rectified Linear Unit (ReLU) and Batch Normalization (BN) are used to prevent information loss in the final layer of each block [72].

Original UNet

The structure of this network's encoder block is the same as that mentioned in [65]. The input image passes through two 3 × 3 convolution layers and the activation function, ReLU. The down-sampling operation takes place with a max pooling operation with a stride of 2. The number of filters is doubled compared to the previous layer during down-sampling. This operation continues until reaching the bottleneck zone. The up-sampling of the feature map continues at the end of the bottleneck zone, which is the reverse of down-sampling. All feature maps from the corresponding layer are concatenated to avoid any loss of border pixels in the previous convolution layers. This concatenation operation is called a skip connection. The size of the final convolution layer is 1 × 1, confirming that the output size is the same as the input.

3.3.2. Decoder

Up-sampling is performed in the decoder (Figure 6) to keep the size of the input and output images the same. Some features may be missing because of the use of max pooling during down-sampling. This problem is mitigated during up-sampling by using a filter size for each step that the same as the encoder. The output feature from each filter operation is copied and concatenated to construct the resultant segmented image of the

same size as the input. This process is called transpose convolution. When up-sampling, all local information after filtering is integrated to obtain the full-resolution image. Each network's decoder block (Figure 6) consists of sampling, concatenation, convolution, batch normalization, and activation operations. The number of filters reduces by half after concatenation, from 256 to 16, for all networks except original UNet. In the decoder part, the number of filters in UNet only decreases by half, from 256 to 64.

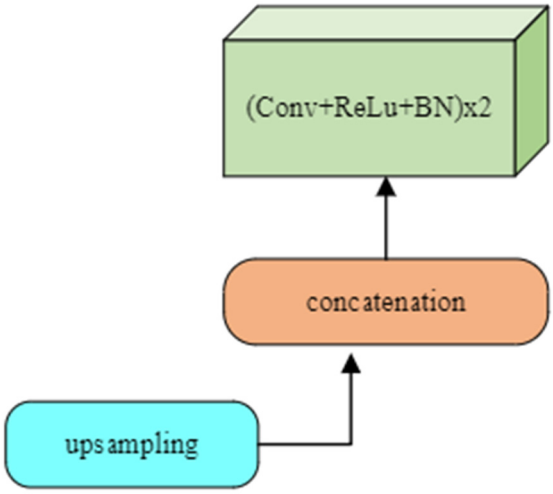


Figure 6. Unit block of decoder layers.

3.4. Model Training

All networks in transfer learning (TL) and fully trained (FT) mode were retrained/trained on the new dataset with 4000 images. Among the 2000 images and 2000 masks, 80% were used for training, 10% for validation, and 10% for testing. Augmentation was performed before training to confirm that the images with their respective mask should be picked by the model during training. The representative loss curve of Resnet34 is depicted in Figure 7. The model did not suffer from overfitting due to dataset augmentation. Figure 7 shows that there is a substantial gap between the training and validation loss curves until approximately 7–8 epochs. After epoch number 10, both the curves reached a point of stability. Hyperparameter selection was performed empirically after attaining knowledge from the literature review. The batch size is the number of images per iteration. The batch size was 16 images for each epoch. The initial learning rate is a parameter that determines the rate of learning. A higher learning rate means the network will learn faster, but the network might settle at a worse value. In this study, the learning rate was 0.001. The number of iterations is the number of times the network weights are updated. The number of epochs is the number of times the network has been updated for all the images in the training set. Here, the numbers of epochs and iterations for each model were 25 and 160, respectively. The number of parameters and training times are given in Table 3.

Table 3. Number of parameters for different models.

Backbone	Total Parameters (M)	Training Mode	Trainable Parameters (M)	Training Time (hrs)
DenseNet121	12.1	FT TL	12 5.2	6.8 5.82
EfficientNetB7	75	FT TL	74.7 11.2	16.35 15.43
ResNet34	24.4	FT	24	8.13
UNet	31.06	FT	31.04	14.4

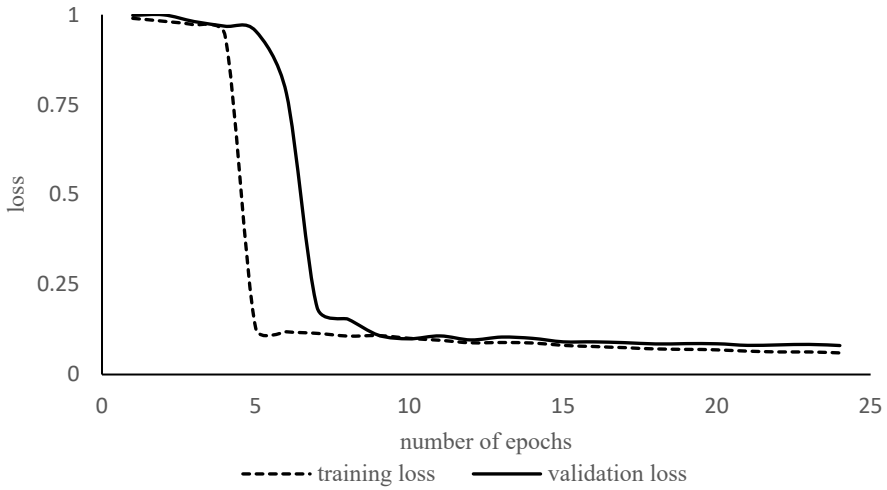


Figure 7. Training and validation loss curves of Resnet34 in full training mode.

3.4.1. Binary Cross-Entropy Loss Function

The binary cross-entropy loss function represents the performance map of a deep learning model that merges one or more variables into a real number [73,74]. There are several types of loss functions [75]. The most used loss function in machine learning is cross entropy [74], which is the difference between two probability distributions of a given set of variables. Selecting a loss function depends entirely on the type of segmentation and the number of classes. Binary cross entropy is used in a binary class model, and a categorical loss function can be used for a multiclass model. The target probability distribution (P) for the dataset would be 0 or 1 for our classification task, interpreted as sound and corroded pixels, respectively. Binary cross entropy can be defined using Equation (6).

$$H(P,Q) = -\sum^{\{x \epsilon X\}} P(x) \times log(Q(x)) \tag{6}$$

where each $\{x \epsilon X\}$ is a class label that is assigned to our dataset, P is the known probability of each class label, and Q is the probability predicted by the model. We established the P value as 0.5 or higher. The loss calculated by the binary cross entropy for a single image can be expressed using Equation (7).

$$H(P,Q) = -(P(\text{class0}) \times log(Q(class0)) + (P(\text{class1}) \times log(Q(class1)) \tag{7}$$

3.4.2. Optimizer

An optimizer's goal is to minimize the loss function by updating the weights and biases in each iteration of training using the training rate. The most used optimizer is stochastic gradient descent (SGD); however, this optimizer needs to be tuned throughout the training process [76]. To mitigate this problem, the functionality of the learning rate has been modified according to different aspects and named Adagrad, Adadelta, RMS-Prop, and ADAM [45,77]. In this study, the ADAM optimizer was used to optimize the network. The working principle [78] of this optimizer can be described using Equations (8)–(11).

$$\mu(t) = \beta_1 \, \mu_{(t-1)} \, + (1 - \beta_1 \,) g_t \tag{8}$$

$$\gamma(\mathrm{t}) = \beta_2 \gamma_{(\mathrm{t}-1)} - (\beta_2 - 1)[\mathrm{g_t}]^2 \tag{9}$$

$$\Delta w_t = -\frac{\alpha \times \mu(t)}{\sqrt{\gamma(t)} + \epsilon} g_t \tag{10}$$

$$w_{t+1} = w_t + \nabla w_t \tag{11}$$

where $\mu(t)$ and $\gamma(t)$ are the first and second moment, of the gradients, respectively. β_1 and β_2 are the decay rate of the average of the gradients. The default values of β_1 and β_2 are 0.9 and 0.99, respectively, unless defined by the user [78]. α and ϵ are the learning rate and regularization term, respectively. g_t is the gradient at time t. w_t and w_{t+1} are the weights at time t and $t+1$, respectively.

4. Results and Discussion

In this section, all results are presented and discussed in detail.

4.1. Evaluation Metrics

Semantic segmentation model performance evaluations should account for classification accuracy and localization correctness. In this study, model performance was evaluated using Equations (12)–(16).

$$\textit{True Positive Rate } (TPR) = \frac{TP}{TP + FN} \tag{12}$$

where *TP* indicates the number of correctly detected pixels, and *FN* (false negative) indicates how many pixels were falsely detected as a negative class.

$$\textit{True Negative Rate } (TNR) = \frac{TN}{TN + FP} \tag{13}$$

where *TN* indicates the number of correctly detected non-corroded pixels, and *FP* (false positive) indicates how many pixels were falsely detected as corroded.

This study evaluated the mean *IOU* or Jaccard Index for each model using Equation (14). Mean *IOU* is the metric by which the percentage of overlap between the ground-truth mask and the predicted mask is determined. This value ranges from 0 to 100%.

$$\textit{Mean IOU or Jaccard Index} = \frac{TP}{TP + FP + FN} \tag{14}$$

*F*1 score measures the balance between the model's precision and recall. *F*1 score is the metric obtained using Equation (15).

$$F1 \textit{ score} = \frac{2TP}{2TP + FP + FN} \tag{15}$$

Pixel accuracy is the metric used to report the percentage of correctly classified pixels and can be determined using Equation (16).

$$\textit{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{16}$$

4.2. Ablation Study

An ablation study investigates the effect of different layers of deep learning models by modifying the architecture or fine-tuning the network [78]. In this study, an ablation study was performed on UNet with different backbone variants, as mentioned below.

4.2.1. Benefits of Knowledge from Transfer Learning

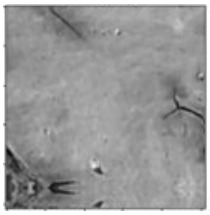
Two pre-trained convolution networks, DenseNet121 and EfficeintNetB7, were used as the backbone of the base model UNet. These two models were pre-trained on the ImageNet dataset. The encoder was kept frozen during training to evaluate the efficacy of the models' pre-trained weights. All evaluation metrics indicated that EfficientNetB7 outperformed Densenet121 (Table 4); however, both models reached an average IOU of 90% or more, and the segmentation results were comparable in most testing images (Table 5). EfficientNetB7 had a 5% greater IOU than DenseNet121. The innovative and

simultaneous compound scaling of network layers, width, and resolution results in a wider network with more feature storage [71]. As a result, EfficientNet reported approximately 4% fewer false-negative (FN) values. DenseNet was trained on 5 M of the parameters, corresponding to almost 50% of the total trainable parameters (12.2 M), which might have affected the memory in the retained weight from the pre-trained mode. Both models successfully detected non-corroded pixels, reporting TNR values above 95%. A detailed comparison between the DenseNet121 and EfficicentNetB7 predictions is presented in Table 5, establishing that 3823 pixels were annotated as corrosion in this specific test image. DenseNet121 predicted 4547 pixels as corrosion, out of which 724 were falsely predicted. EfficientNetB7 only predicted 420 pixels falsely, representing 9.8% of the total predictions.

Table 4. Evaluation metrics of UNet with different backbones in transfer learning mode.

Backbone	TPR (%)	TNR (%)	FPR (%)	FNR (%)	IOU (%)	F1 Score (%)	Accuracy (%)
DenseNet121	93.4	98.5	1.5	6.6	90.77	95.17	96.78
EfficientNetB7	97.3	99.1	0.9	2.7	95.6	97.8	98.5

Table 5. Comparison of predictions between the transfer learning modes of EfficientNetB7 and DenseNet121.

Original Image (Gray Scale)	Ground Truth	DenseNet121 Prediction	EfficientNetB7 Prediction
			
Number of corroded pixels	3823	4547	4243

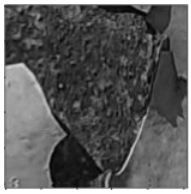
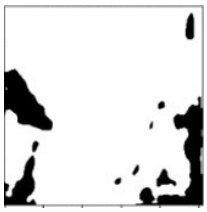
4.2.2. Effect of Training

The model that performed the best in the previous section, EfficientNetB7, was selected to investigate the encoder–decoder model's performance when trained from scratch. ResNet34 was chosen as a less deep network compared to EfficientNetB7. In addition to these models, a UNet with its original structure was built and trained from scratch. ResNet34 and UNet outperformed EfficientNetB7 in terms of all evaluation metrics (Table 6). The network performance of EfficientNetB7 dropped to a 35.5% TPR value when used in fully trained mode. EfficientNetB7 was proficient in detecting sound pixels, with a TNR value of 96.1%. It is difficult to obtain a standard dataset unless it is generated in a controlled environment; therefore, model performance may be affected when tested on images collected from the original structure that was exposed to the environmental conditions. The authors did not use any synthetic images to create a dataset with uniform corroded pixels to address this issue. Non-uniform defect sizes in terms of pixel value might deteriorate EfficientNetB7 performance. Srivastava et al. [75] established that deep networks with large parameters may learn complicated features; however, this property may lead the network to fit noise present in the training dataset when it is smaller, reducing the model's ability to generalize. Our results validated the statement mentioned by Srivastava et al. [78] (Table 7). Compound scaling in EfficientNetB7 was designed so that the model should be trained on a high-resolution dataset [63]; however, the images were resized to 256 × 256 in this study. For this reason, comparatively shallow networks with fewer parameters to train (ResNet34 and UNet) can outperform the deeper EfficientNetB7.

Table 6. Evaluation metrics of UNet with different backbones in fully trained mode.

Base	TPR (%)	TNR (%)	FPR (%)	FNR (%)	IOU (%)	F1 Score (%)	Accuracy (%)
ResNet34	93.4	98.5	1.5	6.6	90.77	95.16	96.78
EfficientNetB7	35.5	96.1	3.9	64.5	35.43	52.32	38.68
UNet	93.9	94	6	0.1	84.11	91.4	94.02

Table 7. Comparison of predictions between full training modes of ResNet34 and UNet.

Original Image (Gray Scale)	Ground Truth	ResNet34 Prediction	UNet Prediction	EfficientNetB7 Prediction
				
Number of corroded pixels	39,128	40,333	42,642	57,096

ResNet 34 and UNet predicted approximately 94% of the corroded pixels correctly due to their shallows depth and less complex architectures. The IOU value ranged from 84 to 90%, which was higher than any semantic segmentation; however, UNet classified 4.5% more sound pixels as corroded, and ResNet34 classified 6.5% more corroded pixels as sound pixels. ResNet34 predicted the edges more sharply than UNet. The evaluation metrics mentioned in Table 6 present the average performance of the mentioned models, while Table 7 presents the performance of each model in a specific image. All models reported false predictions. ResNet34 and UNet yielded false predictions of 2.98% and 8.24%, respectively, whereas EfficientNetB7 detected 31% of the non-corroded pixels as corrosion.

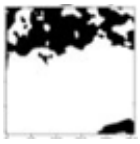
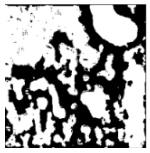
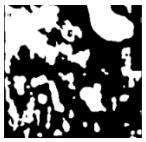
4.3. Qualitative Comparison of ResNet34 and UNet

The model pre-trained on the external dataset and with different color components yielded less noisy segmentation than the fully trained model (Table 8). Both models performed satisfactorily when automatically detecting corroded pixels. The segmentation of the corroded pixels from the external images revealed the robustness of the ResNet34 and UNet models. Fully trained UNet did not segregate the background properly from the corroded part in one image, possibly due to the color similarity between the background and the corroded structure. We implemented UNet and ResNet34 as classifiers and trained them on 3200 images and masks for this test. Our dataset was small compared to big datasets such as ImageNet and CIFAR10 and consists of images from ancillary structures, which are yellow; however, the results established that the models were not biased to this color. The most important component of ResNet34 is the residual block, whereas it is the skip connection for the UNet architecture; therefore, a combination of both architectural features might help the model maintain a gradient, resulting in good predictions. All UNet and ResNet34 predictions are listed in Table 8.

In this study, all models were trained on the same dataset without confirming the optimized number of images necessary for each model. The demand for the number of images for the network with more depth, such as EfficientNetB7, may be higher than the number of images used for training. For this reason, an iterative training process should be performed by varying the number of images and learning rates to confirm the performance

of the underperforming network, EfficientNetB7, in fully trained mode. This process is time-consuming and will be considered in future studies.

Table 8. Performance of fully trained UNet and ResNet34 on external images.

	Prediction by UNet	**Prediction by ResNet34**
		
		
		
		

5. Conclusions

This study investigated the performance of a U-shaped encoder–decoder network for semantically segmented corroded pixels in steel structures with different backbones, along with the original structure. EfficientNetB7, ResNet34, and DenseNet121. A comparison was also made between the pre-trained and fully trained UNet for the autonomous segmentation of corroded pixels in steel structures. The models exhibited remarkable performance in different training modes. For example, EfficientNet reported a TPR of 97% and an IOU of 96.7% for the transfer learning mode; however, it reported a TPR of 35.53% TPR and an IOU of 35.43% for the fully trained mode. ResNet34 and the original UNet reported TPR and IOU values of approximately 90% and 85%, respectively. The results from the fully trained ResNet34 and original UNet models revealed that the shallower network performed well in corroded pixel determination in fully trained mode.

Another novel aspect of this study is the investigation of model performance when external datasets were used for testing. Two fully trained models were tested on the dataset that were independent in terms of dissimilarity with the training set and the type of sensor used during data collection. UNet with a ResNet34 backbone outperformed UNet with its original structure when using an external dataset. The most important component of ResNet34 is the residual block, which helps the model maintain gradient, resulting in better corroded pixel segmentation. The combination of a residual block with feature concatenation through skip connections helped the network collect and transfer important features from the preceding neuron.

The authors did not determine the number of optimum images necessary in fully trained mode, which could be a contributing factor when training deeper networks such as EfficientNetB7. An iterative training process should be performed by varying the number of images and learning rates to confirm the performance of the underperforming network, EfficientNetB7, in fully trained mode. This process is time-consuming and will be

considered in future studies. A comparison of UNet with the other network architectures will also be included in further study.

Author Contributions: A.D., S.D. and N.K. conceived and designed the methods; A.D. collected the data in the field; A.D. annotated the data and constructed the dataset; A.D. trained and tested the model; A.D. prepared the manuscript; S.D. and N.K. reviewed the paper; and S.D. acquired research funding and supervised the research. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the North Dakota Department of Transportation (NDDOT) for the project "Smart UAS Inspection of Ancillary Structures in North Dakota". Contract no. 91210512.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are available upon request to the corresponding author.

Acknowledgments: We wish to thank Eberichi Ichi, Department of Civil Engineering, for helping with data collection.

Conflicts of Interest: There are no known declared conflicts of interest.

References

1. Czichos, H.; Saito, T.; Smith, L.E. (Eds.) *Springer Handbook of Metrology and Testing*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2011.
2. Di Sarno, L.; Majidian, A.; Karagiannakis, G. The effect of atmospheric corrosion on steel structures: A state-of-the-art and case-study. *Buildings* **2021**, *11*, 571. [CrossRef]
3. Kaczinski, M.R.; Dexter, R.J.; Van Dien, J.P. *Fatigue-Resistant Design of Cantilevered Signal, Sign and Light Supports*; Transportation Research Board: Washington, DC, USA, 1998; Volume 412.
4. Koch, G.; Varney, J.; Thompson, N.; Moghissi, O.; Gould, M.; Payer, J. International measures of prevention, application, and economics of corrosion technologies study. *NACE Int.* **2016**, *216*, 2–3.
5. Wu, R.; Zhang, H.; Yang, R.; Chen, W.; Chen, G. Nondestructive testing for corrosion evaluation of metal under coating. *J. Sens.* **2021**, *2021*, 6640406. [CrossRef]
6. Wu, K.; Byeon, J.W. Morphological estimation of pitting corrosion on vertically positioned 304 stainless steel using acoustic-emission duration parameter. *Corros. Sci.* **2019**, *148*, 331–337. [CrossRef]
7. Raude, A.; Bouchard, M.; Sirois, M. Stress Corrosion Cracking Direct Assessment of Carbon Steel Pipeline Using Advanced Eddy Current Array Technology. In Proceedings of the CORROSION 2018, Phoenix, Arizona, USA, 15–19 April 2018, NACE-2018-11150.
8. Feroz, S.; Abu Dabous, S. Uav-based remote sensing applications for bridge condition assessment. *Remote Sens.* **2021**, *13*, 1809. [CrossRef]
9. Garlich, M.J.; Thorkildsen, E.T. *Guidelines for the Installation, Inspection, Maintenance and Repair of Structural Supports for Highway Signs, Luminaires, and Traffic Signals*; No. FHWA-NHI-05-036; Federal Highway Administration: Washington, DC, USA, 2005.
10. Dorafshan, S.; Campbell, L.E.; Maguire, M.; Connor, R.J. Benchmarking Unmanned Aerial Systems-Assisted Inspection of Steel Bridges for Fatigue Cracks. *Transp. Res. Rec.* **2021**, *2675*, 154–166. [CrossRef]
11. Das, A.; Ichi, E.; Dorafshan, S. Image-Based Corrosion Detection in Ancillary Structures. *Infrastructures* **2023**, *8*, 66. [CrossRef]
12. Khayatazad, M.; De Pue, L.; De Waele, W. Detection of corrosion on steel structures using automated image processing. *Dev. Built Environ.* **2020**, *3*, 100022. [CrossRef]
13. Hoang, N.D. Image processing-based pitting corrosion detection using metaheuristic optimized multilevel image thresholding and machine-learning approaches. *Math. Probl. Eng.* **2020**, *2020*, 6765274. [CrossRef]
14. Naik, D.L.; Sajid, H.U.; Kiran, R.; Chen, G. Detection of corrosion-indicating oxidation product colors in steel bridges under varying illuminations, shadows, and wetting conditions. *Metals* **2020**, *10*, 1439. [CrossRef]
15. Sarker, I.H. Data science and analytics: An overview from data-driven smart computing, decision-making and applications perspective. *SN Comput. Sci.* **2021**, *2*, 377. [CrossRef]
16. Sarker, I.H. Machine learning: Algorithms, real-world applications and research directions. *SN Comput. Sci.* **2021**, *2*, 160. [CrossRef]
17. Sarker, I.H.; Furhad, M.H.; Nowrozy, R. Ai-driven cybersecurity: An overview, security intelligence modeling and research directions. *SN Comput. Sci.* **2021**, *2*, 173. [CrossRef]
18. Sarker, I.H. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Comput. Sci.* **2021**, *2*, 420. [CrossRef]
19. Jin, L.H.; Hwang, S.; Kim, H.; Sohn, H. Steel bridge corrosion inspection with combined vision and thermographic images. *Struct. Health Monit.* **2021**, *20*, 3424–3435.

20. Ta, Q.B.; Kim, J.T. Monitoring of corroded and loosened bolts in steel structures via deep learning and Hough transforms. *Sensors* **2020**, *20*, 6888. [CrossRef]
21. Ta, Q.B.; Huynh, T.C.; Pham, Q.Q.; Kim, J.T. Corroded Bolt Identification Using Mask Region-Based Deep Learning Trained on Synthesized Data. *Sensors* **2022**, *22*, 3340. [CrossRef]
22. Zhang, S.; Deng, X.; Lu, Y.; Hong, S.; Kong, Z.; Peng, Y.; Luo, Y. A channel attention based deep neural network for automatic metallic corrosion detection. *J. Build. Eng.* **2021**, *42*, 103046. [CrossRef]
23. Holm, E.; Transeth, A.A.; Knudsen, O.Ø.; Stahl, A. Classification of corrosion and coating damages on bridge constructions from images using convolutional neural networks. In Proceedings of the Twelfth International Conference on Machine Vision (ICMV 2019), Amsterdam, The Netherlands, 16–18 November 2019; Volume 11433, pp. 549–556.
24. Atha, D.J.; Jahanshahi, M.R. Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection. *Struct. Health Monit.* **2018**, *17*, 1110–1128. [CrossRef]
25. Bolton, T.; Bass, J.; Gaber, T. A comparison of deep learning techniques for corrosion detection. In Proceedings of the 8th International Conference on Advanced Intelligent Systems and Informatics 2022, Cairo, Egypt, 20–22 November 2022; Springer International Publishing: Cham, Switzerland, 2022; pp. 189–198.
26. Rahman, A.; Wu, Z.Y.; Kalfarisi, R. Semantic deep learning integrated with RGB feature-based rule optimization for facility surface corrosion detection and evaluation. *J. Comput. Civ. Eng.* **2021**, *35*, 04021018. [CrossRef]
27. Forkan, A.R.M.; Kang, Y.B.; Jayaraman, P.P.; Liao, K.; Kaul, R.; Morgan, G.; Ranjan, R.; Sinha, S. CorrDetector: A framework for structural corrosion detection from drone images using ensemble deep learning. *Expert Syst. Appl.* **2022**, *193*, 116461. [CrossRef]
28. Ahuja, S.K.; Shukla, M.K.; Ravulakollu, K.K. Optimized Deep Learning Framework for Detecting Pitting Corrosion based on Image Segmentation. *Int. J. Perform. Eng.* **2021**, *17*, 627.
29. Bastian, B.T.; Jaspreeth, N.; Ranjith, S.K.; Jiji, C.V. Visual inspection and characterization of external corrosion in pipelines using deep neural network. *NDT E Int.* **2019**, *107*, 102134. [CrossRef]
30. Huang, J.; Liu, Q.; Xiang, L.; Li, G.; Zhang, Y.; Chen, W. A Lightweight Residual Model for Corrosion Segmentation with Local Contextual Information. *Appl. Sci.* **2022**, *12*, 9095. [CrossRef]
31. Cha, Y.J.; Choi, W.; Suh, G.; Mahmoudkhani, S.; Büyüköztürk, O. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Comput.-Aided Civ. Infrastruct. Eng.* **2018**, *33*, 731–747. [CrossRef]
32. Munawar, H.S.; Ullah, F.; Shahzad, D.; Heravi, A.; Qayyum, S.; Akram, J. Civil infrastructure damage and corrosion detection: An application of machine learning. *Buildings* **2022**, *12*, 156. [CrossRef]
33. Fondevik, S.K.; Stahl, A.; Transeth, A.A.; Knudsen, O.Ø. Image segmentation of corrosion damages in industrial inspections. In Proceedings of the 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), Baltimore, MD, USA, 9–11 November 2020; pp. 787–792.
34. Katsamenis, I.; Protopapadakis, E.; Doulamis, A.; Doulamis, N.; Voulodimos, A. Pixel-level corrosion detection on metal constructions by fusion of deep learning semantic and contour segmentation. In *Advances in Visual Computing, Proceedings of the 15th International Symposium, ISVC 2020, San Diego, CA, USA, 5–7 October 2020, Proceedings, Part I*; Springer International Publishing: Cham, Switzerland, 2020; pp. 160–169.
35. Jiang, F.; Ding, Y.; Song, Y.; Geng, F.; Wang, Z. Automatic pixel-level detection and measurement of corrosion-related damages in dim steel box girders using Fusion-Attention-U-net. *J. Civ. Struct. Health Monit.* **2023**, *13*, 199–217. [CrossRef]
36. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
37. Duy, L.D.; Anh, N.T.; Son, N.T.; Tung, N.V.; Duong, N.B.; Khan, M.H.R. Deep learning in semantic segmentation of rust in images. In Proceedings of the 2020 9th International Conference on Software and Computer Applications, Langkawi, Malaysia, 18–21 February 2020; pp. 129–132.
38. Géron, A. *Hands-On Machine Learning with Scikit-Learn and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2017.
39. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*.
40. Emmert-Streib, F.; Yang, Z.; Feng, H.; Tripathi, S.; Dehmer, M. An introductory review of deep learning for prediction models with big data. *Front. Artif. Intell.* **2020**, *3*, 4. [CrossRef]
41. LeCun, Y. Generalization and network design strategies. *Connect. Perspect.* **1989**, *19*, 18.
42. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; et al. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
43. Jiang, X.; Satapathy, S.C.; Yang, L.; Wang, S.H.; Zhang, Y.D. A survey on artificial intelligence in Chinese sign language recognition. *Arab. J. Sci. Eng.* **2020**, *45*, 9859–9894. [CrossRef]
44. Wang, T.; Wu, D.J.; Coates, A.; Ng, A.Y. End-to-end text recognition with convolutional neural networks. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), Tsukuba, Japan, 11–15 November 2012; pp. 3304–3308.
45. Boureau, Y.L.; Ponce, J.; LeCun, Y. A theoretical analysis of feature pooling in visual recognition. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 111–118.

46. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014, Proceedings, Part I 13*; Springer International Publishing: Cham, Switzerland, 2014; pp. 818–833.
47. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv* **2012**, arXiv:1207.0580.
48. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
49. Yamashita, R.; Nishio, M.; Do, R.K.G.; Togashi, K. Convolutional neural networks: An overview and application in radiology. *Insights Imaging* **2018**, *9*, 611–629. [CrossRef] [PubMed]
50. Dorafshan, S.; Thomas, R.J.; Maguire, M. Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. *Constr. Build. Mater.* **2018**, *186*, 1031–1045. [CrossRef]
51. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 9. [CrossRef]
52. Han, X.; Hu, Z.; Wang, S.; Zhang, Y. A Survey on Deep Learning in COVID-19 Diagnosis. *J. Imaging* **2022**, *9*, 1. [CrossRef] [PubMed]
53. Damacharla, P.; Rao, A.; Ringenberg, J.; Javaid, A.Y. TLU-net: A deep learning approach for automatic steel surface defect detection. In Proceedings of the 2021 International Conference on Applied Artificial Intelligence (ICAPAI), Halden, Norway, 19–21 May 2021; pp. 1–6.
54. Ge, L.; Gao, J.; Ngo, H.; Li, K.; Zhang, A. On handling negative transfer and imbalanced distributions in multiple source transfer learning. *Stat. Anal. Data Min. ASA Data Sci. J.* **2014**, *7*, 254–271. [CrossRef]
55. Seah, C.W.; Ong, Y.S.; Tsang, I.W. Combating negative transfer from predictive distribution differences. *IEEE Trans. Cybern.* **2012**, *43*, 1153–1165. [CrossRef]
56. Mumuni, A.; Mumuni, F. Data augmentation: A comprehensive survey of modern approaches. *Array* **2022**, *16*, 100258. [CrossRef]
57. Hasanpour, S.H.; Rouhani, M.; Fayyaz, M.; Sabokrou, M. Let's keep it simple, using simple architectures to outperform deeper and more complex architectures. *arXiv* **2016**, arXiv:1608.06037.
58. Devian Art. Available online: www.devianart.com52 (accessed on 23 October 2022).
59. Jung, A.B.; Wada, K.; Crall, J.; Tanaka, S.; Graving, J.; Yadav, S.; Banerjee, J.; Vecsei, G.; Kraft, A.; Borovec, J.; et al. Imgaug. 2019. Available online: https://github.com/aleju/imgaug (accessed on 24 February 2023).
60. Bloice, M.D.; Roth, P.M.; Holzinger, A. Biomedical image augmentation using Augmentor. *Bioinformatics* **2019**, *35*, 4522–4524. [CrossRef] [PubMed]
61. Casado-García, Á.; Domínguez, C.; García-Domínguez, M.; Heras, J.; Inés, A.; Mata, E.; Pascual, V. CLoDSA: A tool for augmentation in classification, localization, detection, semantic segmentation and instance segmentation tasks. *BMC Bioinform.* **2019**, *20*, 323. [CrossRef] [PubMed]
62. Tiulpin, A. Solt: Streaming over Lightweight Transformations. 2019. Available online: https://zenodo.org/records/3702819 (accessed on 24 February 2023).
63. Buslaev, A.; Iglovikov, V.I.; Khvedchenya, E.; Parinov, A.; Druzhinin, M.; Kalinin, A.A. Albumentations: Fast and flexible image augmentations. *Information* **2020**, *11*, 125. [CrossRef]
64. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: Proceedings of the 18th International Conference, Munich, Germany, 5–9 October 2015, Proceedings, Part III 18*; Springer International Publishing: Cham, Switzerland, 2015; pp. 234–241.
65. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
66. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
67. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
68. Sarker, L.; Islam, M.M.; Hannan, T.; Ahmed, Z. COVID-DenseNet: A deep learning architecture to detect COVID-19 from chest radiology images. *Preprint* **2020**. [CrossRef]
69. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the 36th International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
70. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
71. Agarap, A.F. Deep learning using rectified linear units (relu). *arXiv* **2018**, arXiv:1803.08375.
72. Brownlee, J. Probability for Machine Learning: Discover How to Harness Uncertainty with Python. Machine Learning Mastery, 2019.
73. Aretz, K.; Bartram, S.M.; Pope, P.F. Asymmetric loss functions and the rationality of expected stock returns. *Int. J. Forecast.* **2011**, *27*, 413–437. [CrossRef]
74. Zhang, Z. Improved adam optimizer for deep neural networks. In Proceedings of the 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS), Banff, AB, Canada, 4–6 June 2018; pp. 1–2.
75. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.

76. Labhsetwar, S.R.; Haridas, S.; Panmand, R.; Deshpande, R.; Kolte, P.A.; Pati, S. Performance Analysis of Optimizers for Plant Disease Classification with Convolutional Neural Networks. In Proceedings of the 2021 4th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), NaviMumbai, India, 15–16 January 2021; pp. 1–6.
77. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
78. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.