Original Article

SHM Structural Health Monitoring

Structural Health Monitoring
2018, Vol. 17(5) 1110–1128

DOI: 10.1177/1475921717737051
journals.sagepub.com/home/shm
SAGE

Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection

Deegan J Atha[1] and Mohammad R Jahanshahi[2]

Abstract

Corrosion is a major defect in structural systems that has a significant economic impact and can pose safety risks if left untended. Currently, an inspector visually assesses the condition of a structure to identify corrosion. This approach is time-consuming, tedious, and subjective. Robotic systems, such as unmanned aerial vehicles, paired with computer vision algorithms have the potential to perform autonomous damage detection that can significantly decrease inspection time and lead to more frequent and objective inspections. This study evaluates the use of convolutional neural networks for corrosion detection. A convolutional neural network learns the appropriate classification features that in traditional algorithms were hand-engineered. Eliminating the need for dependence on prior knowledge and human effort in designing features is a major advantage of convolutional neural networks. This article presents different convolutional neural network–based approaches for corrosion assessment on metallic surfaces. The effect of different color spaces, sliding window sizes, and convolutional neural network architectures are discussed. To this end, the performance of two pretrained state-of-the-art convolutional neural network architectures as well as two proposed convolutional neural network architectures are evaluated, and it is shown that convolutional neural networks outperform state-of-the-art vision-based corrosion detection approaches that are developed based on texture and color analysis using a simple multilayered perceptron network. Furthermore, it is shown that one of the proposed convolutional neural networks significantly improves the computational time in contrast with state-of-the-art pretrained convolutional neural networks while maintaining comparable performance for corrosion detection.

Keywords

Convolutional neural network, corrosion detection, sliding window, color space

Introduction

Motivation

Corrosion is a major problem both in the US and globally. It is estimated that the global cost of corrosion is US$2.5 trillion and US$451.3 billion for the United States. This equates to 3.4% of the global gross domestic product (GDP) and 2.7% of the GDP of the United States.[1] In addition to the economic costs of corrosion, it can pose safety risks for those using or around the damaged structure or undue burdens such as traffic delays. Fast, accurate, and efficient methods to identify corrosion are needed.

Currently, there is interest in using unmanned aerial vehicles for visual inspection of structures.[2–4] Robotic inspections of structures will increase the diagnostic speed and reduce the costs associated with inspections. Furthermore, with the abundance of smartphones with high-quality cameras, the increasing computing power of mobile platforms, and the expansion of cloud computing, there is an opportunity for people without a background in structural engineering to use these platforms for automated inspection. Traditionally, inspections have been performed by an expert who visually inspects the structure. This can be time-consuming, costly, and tedious. With the use of these robotic and mobile platforms for inspection, image processing and

[1]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA
[2]Lyles School of Civil Engineering, Purdue University, West Lafayette, IN, USA

Corresponding author:
Mohammad R Jahanshahi, Lyles School of Civil Engineering, Purdue University, 550 Stadium Mall Drive, West Lafayette, IN 47907-2050, USA.
Email: jahansha@purdue.edu

computer vision inspection algorithms are the next step to a fully autonomous inspection system.

There have been several proposed algorithms for corrosion detection since vision-based data collection is contactless and ideal for incorporation into robotic systems. To this end, texture and color analyses in a statistical- or filter-based approach have been popular techniques. One of the most common filter-based approaches is the color wavelet filter bank. This approach requires a set of texture and color features. The accuracy of corrosion detection suffers if the optimal features are not identified.

Recently, the classification capabilities of computers have increased due to the use of convolutional neural networks (CNNs).[5] Currently, CNNs have been demonstrated to achieve human-level classification accuracy on the ImageNet challenge.[6] The success that has been achieved by these CNNs suggests that corrosion detection through computer vision algorithms could also achieve human-level accuracy. Unlike previous approaches, CNNs do not require any features to be determined by a human expert. The network learns the features from the training data. Furthermore, the networks have shown to be robust enough to classify objects with different scales, orientations, and levels of occlusion. It is expected that CNNs can detect corrosion more accurately than previous approaches.

The focus of this study is to quantify the performance of corrosion classification on a wide range of types of corroded regions and to find the optimal inputs into the CNN as well as CNN's optimal architecture. The results of this study can be used to create robust systems for a wide range of applications related to corrosion detection.

Background

There have been several studies in the area of vision-based corrosion detection.[7] Gray level co-occurrence matrices (GLCMs) are used by Haralick et al.[8] as statistical texture features for corrosion detection. GLCMs represent the frequency of different gray level combinations within an image.

Valois et al.[9] proposed that human brains operate as two-dimensional (2D) spatial filters in a psychophysical study. These insights inspired the use of multi-scale techniques for damage detection purposes.[10]

Furuta et al.[11] proposed a NN with hue/saturation/value (HSV) color space features to determine the level of deterioration of paint films. This system helped inexperienced inspectors to assess the damage of structural corrosion.

Livens et al.[12] used the wavelet decomposition of images with a learning vector quantization (LVQ) network to classify images with corrosion into two classes. This approach was shown to outperform a Gaussian classifier.

Gunatilake et al.[13] detected corrosion on aircraft skins using wavelets of order 6 proposed by Daubechies.[14] The features used were energy values which were found from 8×8 pixel regions. Corroded versus non-corroded regions were classified with a nearest neighbor classifier.

Gunatilake and Siegel[15] classified image blocks as corrosion with high confidence or low confidence, or as corrosion free using the luminance/in-phase/quadrature (YIQ) color space, the Battle–Lemaire (BL)[16] wavelet transform filter, and a feed-forward NN. The image blocks were non-overlapping 32×32 pixel regions.

Choi and Kim[17] used the hue/saturation/intensity (HSI) color space, principal component analysis, and varimax approaches to extract classification features to classify multiple types of corrosion defects. Feature extraction based on the azimuth difference of points on a surface was performed with the GLCM method. This approach was not applicable for images from a digital camera since the magnification factors of the tested images varied between 50 and 500.

Using a statistical analysis on Red/Green/Blue (RGB) color channels, Lee et al.[18] were able to classify rusts on steel bridge coatings. To classify corroded and non-corroded regions, Palakal et al. and Pidaparti used a K-means classifier with classification features computed by the Haar wavelet transform. The material loss from corrosion was estimated using a back-propagation NN.[19,20]

Xie[10] identified several downsides of the GLCM method such as the challenges of reducing the gray levels to overcome computational constraints, finding the appropriate number of entries for each matrix, and optimizing the displacement vector. Medeiros et al.[21] detected corrosion in carbon steel storage tanks and pipelines within a petroleum refinery through the combination of HSI color space statistics and GLCM probabilities.

Bonnin-Pascual and Ortiz[22] proposed a weak classifiers cascade that chained different weak but fast classifiers to create an improved global classifier. Through the use of a GLCM, roughness was measured first. Then, color information was obtained through the use of the HSV color space. This corrosion detection algorithm was used to help inspectors assess the hulls of vessels.

Pidaparti et al.[23] detected corrosion on nickel aluminum bronze metal using the Haar wavelet transform and fractals. It was shown that using features from the fractal dimension and the Shannon[24] entropy of the wavelet filter bank decomposition, cracks and pits were linearly separable.

Pakrashi et al.[25] proposed a method based on optical contrast of the corroded region compared to its

surrounding regions for detection and quantification of pitting corrosion on aluminum surfaces. The corroded region was identified using edge detection. However, not all forms of corrosion have the same dominant features as pitting corrosion; therefore, this approach is not applicable to other forms of surface corrosion.

Ghanta et al.[26] was able to detect defects on images of steel coating bridge surfaces using a one-level Haar wavelet transform applied to an RGB subimage and using entropy and energy in the wavelet domain as classification features. The optimal subimage size found was 8×8 pixels.

Jahanshahi and Masri[7] improved the reliability of color and texture analysis by evaluating the effects of different parameters in color wavelet–based texture analysis algorithms. It was shown the wavelet-based features obtained from CbCr color channels lead to best performance with respect to other color space channels (e.g. YIQ, RGB, HSI).

There are numerous approaches for autonomous vision-based corrosion detection in civil infrastructure systems. A survey and evaluation of many of the best methods is presented by Jahanshahi et al.[27] The previous approaches require an understanding of corrosion and the manual identification and experimentation of features. CNNs, however, are able to learn features and can be applied more broadly to multiple types of corrosion. There have been few studies on CNNs for structural damage detection. One of the few uses a CNN for crack detection.[28]

Contribution

The main contribution of this work is to present a CNN and sliding window approach for corrosion detection that outperforms the existing state-of-the-art corrosion detection algorithm. The effects of different sliding window sizes and color spaces as input into a CNN for corrosion detection are evaluated. In addition, the CNN architecture and the effect of fine-tuning a CNN are evaluated for both signal-to-noise ratio and computational performance to identify the optimal architecture and approach to train a CNN for corrosion classification. Two state-of-the-art CNNs designed for ImageNet as well as two proposed CNN architectures are evaluated.

Scope

The rest of this article is organized as follows: Section "CNNs" provides an overview of CNNs and the basics of the different layers within them. Section "Proposed approach" discusses the approach used to detect corrosion on metallic surfaces. This includes a discussion of the sliding window approach, color spaces, network architectures, and hyperparameters that were used for evaluation. Additionally, the section discusses the creation of a dataset and fine-tuning existing CNNs. Section "Experimental results and discussion" provides the precision, recall, and computational performance of using different image input parameters and CNN architectures. The section also provides results of the comparison between CNN features and wavelet features. Section "Summary and conclusion" provides the conclusions and recommendations for future work.

CNNs

CNNs were shown to be successful at image-based tasks in 1989.[29] However, due to the computational complexity and inadequate computing power at the time, there was minimal continuing research interest into them. In 2009, a database of over 10 million images and several thousand object classes, known as ImageNet, was created.[30] The ImageNet Challenge was created to evaluate and compare classification and detection algorithms on a dataset containing 1000 classes from the ImageNet database. With the development of powerful graphical processing units (GPUs), the creation of ImageNet, and the success of CNNs on the ImageNet Challenge,[5] there has been a significant increase in research interest in CNNs. They have been used for object classification,[5,6] object detection,[31,32] object segmentation,[33] action recognition,[34] the medical field,[35] and others. CNNs have been shown to outperform state-of-the-art classification algorithms in most fields they are utilized. CNNs are machine learning approaches that combine convolution and pooling layers as well as fully connected layers to perform image analysis.

Architecture overview

The basic structure of a CNN has multiple convolution, activation, and pooling layers followed by the fully connected and classification layers. An overview of the basic structure of a CNN is presented in Figure 1. The first part of the network consists of a number of convolution layers. Each convolution layer is followed by a rectified linear unit (ReLU) for activation. Additionally, a pooling layer can follow a convolution layer. ReLUs have been the primary activation layer since it was shown that they achieve better accuracy and faster computations.[36] The second part of the network resembles a normal artificial neural network (ANN) with fully connected hidden layers followed by a classification layer (e.g. Softmax). The transition between a feature map output to The Fully Connected

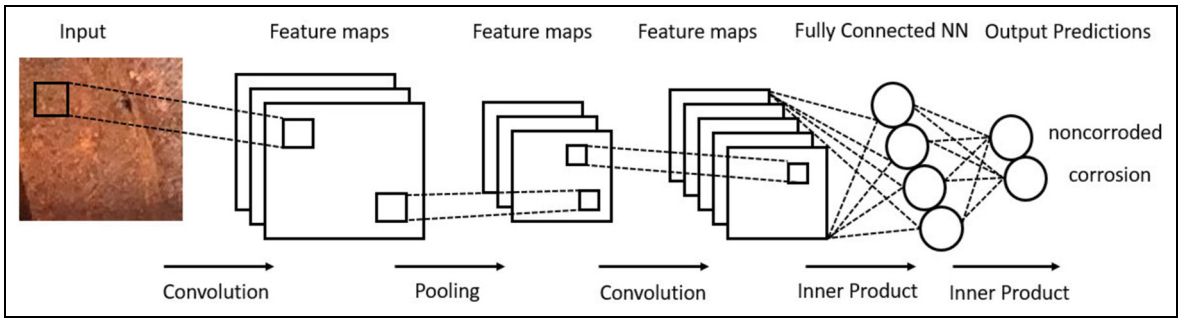

Figure 1. Overview of a CNN architecture.

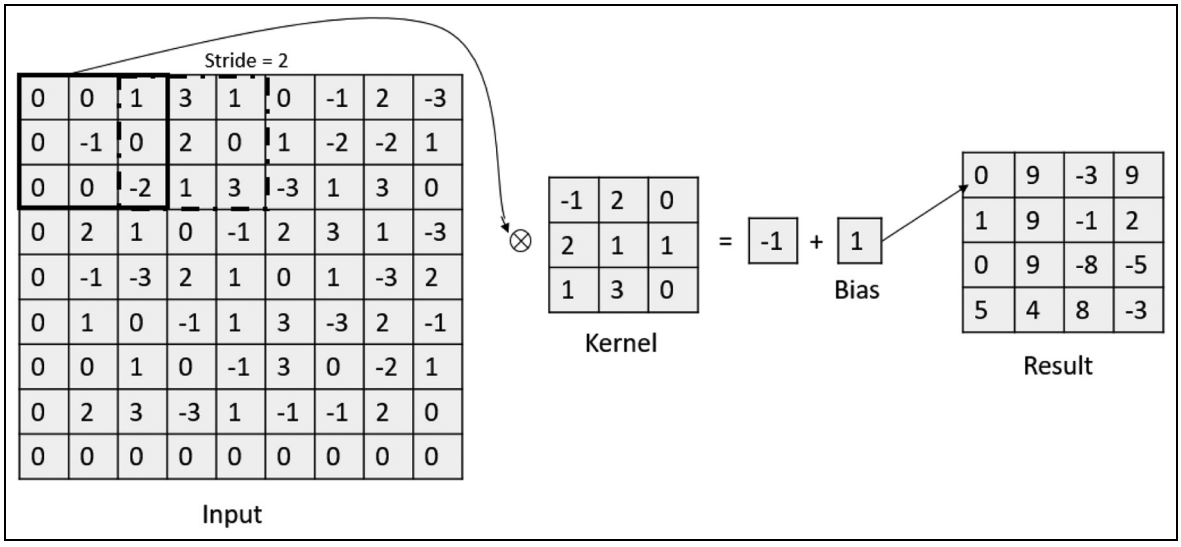

Figure 2. Diagram of a convolution process using a zero-mean normalized input.

NN layer contains full connections from each feature in the feature map to every node in the first fully connected layer. The primary differences between multilayered perceptron (MLP) and CNNs are the convolution and pooling layers which are discussed further.

Convolution layer. Convolution layers are the means by which a CNN extracts features from an input image. A convolution layer consists of a set of small (e.g. 3×3 or 5×5) receptive fields known as filters or kernels which operate on subarrays of equal size from the input data. The weight values for the kernels are typically initialized with random values and updated during training by a stochastic gradient descent algorithm. These weight values can also be initialized with pretrained values in a process known as fine-tuning which is discussed in section "Fine-tuning." A dot product is computed between the kernels and the subarrays of the input data. The multiplied values are summed and a bias value is added. A diagram of a sample convolution process is shown in Figure 2. To operate on the entire input area, the kernels are moved over the input data. The number of indices moved for each new calculation is known as the stride. A smaller stride increases the number of features extracted but is also more computationally expensive.

Pooling layer. Pooling layers of CNNs are used to downsample the data to decrease the computational costs of a network. To downsample the data, each subarray of the data is condensed into a single value. The two main methods are max and mean pooling. In max pooling, the greatest value in the subarray becomes the new value for the subarray. In mean pooling, the values of the subarray are averaged to generate the new value. As with the convolution layer, the stride of a pooling layer determines how many indices to shift to perform

the next pooling operation on a new subarray. Most CNNs currently use overlapping subarrays after the success of AlexNet.[5]

Dropout layer. Dropout layers are used to prevent overfitting.[37] Overfitting occurs when the network is so well optimized for the training data that the network performs poorly on the test data because it cannot handle the new variations within the test images. A dropout layer sets a random set of activations to zero during the forward pass while the network is being trained. This removal of certain activations during training forces the network to be redundant, which prevents the network from overfitting by training the network to provide the correct classification even if some of the activations are set to zero.

Existing architectures

There has been significant research into designing the best network architecture. Krizhevsky et al.[5] proposed AlexNet, an eight-layer architecture with five convolution layers. AlexNet was improved through the creation of ZF Net[38] based on an in-depth analysis of what the CNN was learning. The VGG networks[39] were developed by the Visual Geometry Group at Oxford University where architectures of 16–19 layers were evaluated, and the networks were able to outperform the previous state-of-the-art networks of smaller sizes. One of the main challenges of training a deeper network is the computational costs. GoogLeNet[40] is an architecture with 22 layers that was able to have a greater depth and width without increasing the computational costs. ResNet,[6] a 152-layer architecture, was developed to maximize the depth of the network while limiting the complexity. This architecture is eight times deeper than VGG networks, while it has lower complexity than them. These networks are primarily evaluated on the ImageNet Challenges; however, they are developed to be effective on any dataset of images.

Proposed approach

In this study, it is proposed to use a CNN to classify image regions as corroded or not. In this section, the methodology on how CNNs are used to classify corrosion is presented. To this end, two existing networks that have shown significant success on ImageNet,[30] VGG16[39] and ZF Net,[38] are used. In addition, two networks are proposed that are smaller and faster than ZF Net and VGG16 but are shown to achieve similar signal-to-noise ratios as presented in section "Experimental results and discussion." In order to identify the optimal color space for corrosion detection using a CNN, the color spaces RGB, YCbCr, CbCr, and grayscale are used to train and test ZF Net. It was demonstrated in Jahanshahi et al.[7] that identifying corrosion is not always the most accurate in the RGB color space. After determining the optimal color space, different architectures are evaluated for that color space. In addition, to classify the regions of an image as corroded or not, multiple sliding window sizes (128×128, 64×64, and 32×32) are used. A sliding window approach is utilized to determine corroded regions within an image. Analyzing these different window sizes will help identify the optimum window size for a sliding window approach.

Sliding window

To classify the different regions as corroded or noncorroded, a sliding window approach is used. A diagram of this approach is shown in Figure 3. With a smaller sliding window size, the corroded region can be more accurately localized. However, the smaller the input image size, the less number of features a CNN can learn. This decrease in the number of features could lead to a decrease in signal-to-noise ratio. To evaluate the impact of sliding window sizes, images of 128×128, 64×64, and 32×32 pixels are used and the overall performance of the proposed approach is evaluated for these sliding window sizes. The proposed approach does not have overlapping window. For example, the 128×128 pixel sliding window region is shifted by 128 pixels horizontally or vertically for a new region. Without overlap between the windows, each window size classifies the same amount of image area. This facilitates a fair comparison between different sliding window sizes.

Color spaces

Furuta et al.,[11] and Jahanshahi and Masri[7] previously have shown that RGB is not always the best color space for color analysis since the channel values are not independent. RGB images are the combination of the red, green, and blue light that is captured within an image. A criticism of this color space is the fact that the amount of red, green, and blue light captured is correlated to the amount of light hitting an object. Often times, other color spaces attempt to minimize this correlation. An example of a color space different from RGB is the YCbCr color space seen in Figure 4. This color space has a luma signal (Y) and two chroma components (Cb and Cr). Equation (1) shows how to convert from RGB to YCbCr. Grayscale is tested to evaluate the effectiveness of only using a single pixel value as opposed to using multiple channels

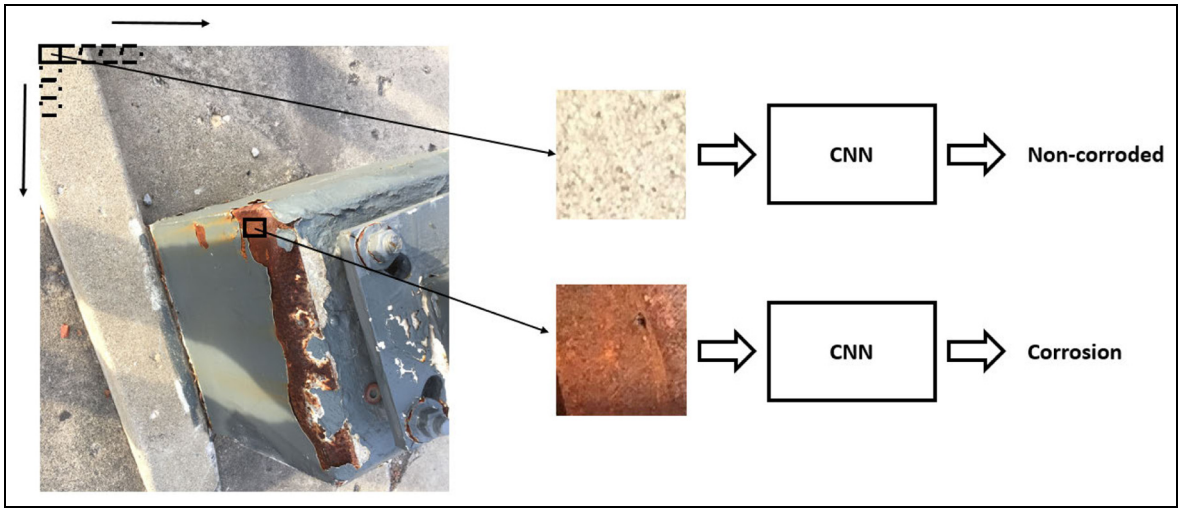

Figure 3. Sample sliding windows used for region classification.

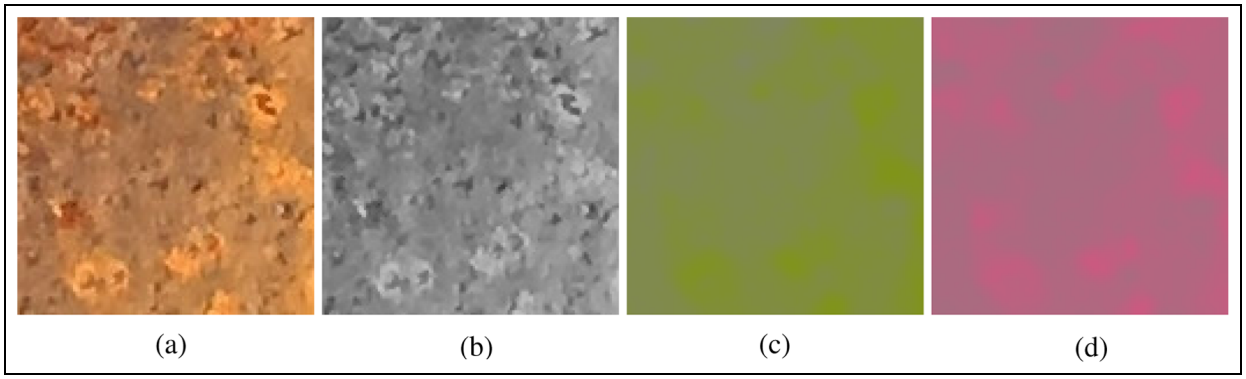

Figure 4. The channel breakdown for a sample YCbCr image: (a) RGB image, (b) Y channel, (c) Cb channel, and (d) Cr channel.

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix} \quad (1)$$

There has been limited research into the effect of different color spaces for CNNs. In Bojarski et al.,[41] YUV images were used for the input into a CNN and Ng et al.[42] studied the effect of different color spaces for gender classification. To identify the optimal color space to classify corrosion, RGB, YCbCr, CbCr, and grayscale color channels are evaluated in the current study. CbCr was shown to be the most robust for corrosion detection using wavelet decomposition by Jahanshahi and Masri.[7] To test the effect of different color spaces, the sliding window regions are converted to the aforementioned color spaces before being input into the CNN.

Network architecture

To analyze the impact of the network architecture, two existing architectures and two proposed architectures are studied. ZF Net[38] is a small, fast, efficient CNN and VGG16[39] is a deeper and more computational expensive CNN. These two networks were chosen because they have exhibited state-of-the-art performance on the ImageNet Challenge and provide a comparison between classification and computational performance of different size networks. ZF Net consists of five convolution layers followed by three fully connected layers. A softmax classifier is used to determine whether the sliding window is corroded or not. A table of ZF Net's architecture is in Table 1. VGG16 contains 13 convolution layers followed by three fully connected layers. A softmax classifier is also used by VGG16. The architecture of VGG16 is in Table 2. Additionally, VGG15 is utilized to compare the use of CNNs with wavelet features input into a NN. VGG15 utilizes the

Table 1. Architecture of ZF Net.

Layer	Input	Kernel shape	Number of kernels	Stride	Output	Number of variables
Conv1	128 × 128 × 3	7 × 7 × 3	96	2	64 × 64 × 96	14,112
Pool1	64 × 64 × 96	3 × 3 × 1	–	2	33 × 33 × 96	–
Conv2	33 × 33 × 96	5 × 5 × 96	256	2	17 × 17 × 256	614,400
Pool2	17 × 17 × 256	3 × 3 × 1	–	2	9 × 9 × 256	–
Conv3	9 × 9 × 128	3 × 3 × 256	384	1	9 × 9 × 384	884,736
Conv4	9 × 9 × 384	3 × 3 × 384	384	1	9 × 9 × 384	1,327,104
Conv5	9 × 9 × 384	3 × 3 × 384	256	1	9 × 9 × 256	884,736
pool3	9 × 9 × 256	3 × 3 × 256	–	2	4 × 4 × 256	–
FC1	4 × 4 × 256	2048	4096	–	4096	8,338,608
FC2	4096	4096	4096	–	4096	16,777,216
FC3	4096	4096	2	–	2	8192

Table 2. Architecture of VGG16.

Layer	Input	Kernel shape	Number of kernels	Stride	Output	Number of variables
Conv1	128 × 128 × 3	3 × 3 × 3	64	1	128 × 128 × 64	1728
Conv2	128 × 128 × 64	3 × 3 × 64	64	1	128 × 128 × 64	36,864
Pool1	128 × 128 × 64	2 × 2 × 1	–	2	64 × 64 × 64	–
Conv3	64 × 64 × 64	3 × 3 × 64	128	1	64 × 64 × 128	73,728
Conv4	64 × 64 × 128	3 × 3 × 128	128	1	64 × 64 × 128	147,456
Pool2	64 × 64 × 128	2 × 2 × 1	–	2	32 × 32 × 128	–
Conv5	32 × 32 × 128	3 × 3 × 128	256	1	32 × 32 × 256	294,912
Conv6	32 × 32 × 256	3 × 3 × 256	256	1	32 × 32 × 256	589,824
Conv7	32 × 32 × 256	3 × 3 × 256	256	1	32 × 32 × 256	589,824
Pool3	32 × 32 × 256	2 × 2 × 1	–	2	16 × 16 × 256	–
Conv8	16 × 16 × 256	3 × 3 × 256	512	1	16 × 16 × 512	1,179,648
Conv9	16 × 16 × 512	3 × 3 × 512	512	1	16 × 16 × 512	2,359,296
Conv10	16 × 16 × 512	3 × 3 × 512	512	1	16 × 16 × 512	2,359,296
Pool4	16 × 16 × 512	2 × 2 × 1	–	2	8 × 8 × 512	–
Conv11	8 × 8 × 512	3 × 3 × 512	512	1	8 × 8 × 512	2,359,296
Conv12	8 × 8 × 512	3 × 3 × 512	512	1	8 × 8 × 512	2,359,296
Conv13	8 × 8 × 512	3 × 3 × 512	512	1	8 × 8 × 512	2,359,296
Pool5	8 × 8 × 512	2 × 2 × 1	–	2	4 × 4 × 512	–
FC1	4 × 4 × 512	8192	4096	–	4096	33,554,432
FC2	4096	4096	4096	–	4096	16,777,216
FC3	4096	4096	2	–	2	8192

same fully connected layer sizes as the NN proposed by Jahanshahi and Masri[7] but uses VGG16's convolution layers as shown in Table 3.

In addition, two networks are proposed. Corrosion7 is seven-layer network with five convolution layers followed by two fully connected layers. This network contains half as many feature maps for each convolution layer compared to ZF Net. Decreasing the number of feature maps will decrease the computation time of the network. In addition, the first fully connected layer has 1024 neurons and the second is the classification layer and has two neurons. This is compared to two layers of 4096 neurons followed by the classification layer of two neurons in ZF Net. It is proposed that accuracy will not suffer because to classify corrosion, less features are needed to perform a binary classification of corrosion compared to classifying 1000 different objects for which ZF Net was designed. Corrosion5, the second proposed CNN, is a five-layer network with three convolution layers followed by two fully connected layers. The two fully connected layers are the same as Corrosion7. Corrosion5 is used to determine whether or not reducing the number of convolution layers or feature maps result in a better performance. A full breakdown of the layers in Corrosion7 and Corrosion5 is in Tables 4 and 5, respectively. A comparison of total parameters is in Table 6.

Hyperparameters

There are several hyperparameters that need to be set for CNNs. These parameters include the initial learning rate, batch size, number of iterations, and number of

Table 3. Proposed architecture of VGG15.

Layer	Input	Kernel shape	Number of kernels	Stride	Output	Number of variables
Conv1	128 × 128 × 3	3 × 3 × 3	64	1	128 × 128 × 64	1728
Conv2	128 × 128 × 64	3 × 3 × 64	64	1	128 × 128 × 64	36,864
Pool1	128 × 128 × 64	2 × 2 × 1	–	2	64 × 64 × 64	–
Conv3	64 × 64 × 64	3 × 3 × 64	128	1	64 × 64 × 128	73,728
Conv4	64 × 64 × 128	3 × 3 × 128	128	1	64 × 64 × 128	147,456
Pool2	64 × 64 × 128	2 × 2 × 1	–	2	32 × 32 × 128	–
Conv5	32 × 32 × 128	3 × 3 × 128	256	1	32 × 32 × 256	294,912
Conv6	32 × 32 × 256	3 × 3 × 256	256	1	32 × 32 × 256	589,824
Conv7	32 × 32 × 256	3 × 3 × 256	256	1	32 × 32 × 256	589,824
Pool3	32 × 32 × 256	2 × 2 × 1	–	2	16 × 16 × 256	–
Conv8	16 × 16 × 256	3 × 3 × 256	512	1	16 × 16 × 512	1,179,648
Conv9	16 × 16 × 512	3 × 3 × 512	512	1	16 × 16 × 512	2,359,296
Conv10	16 × 16 × 512	3 × 3 × 512	512	1	16 × 16 × 512	2,359,296
Pool4	16 × 16 × 512	2 × 2 × 1	–	2	8 × 8 × 512	–
Conv11	8 × 8 × 512	3 × 3 × 512	512	1	8 × 8 × 512	2,359,296
Conv12	8 × 8 × 512	3 × 3 × 512	512	1	8 × 8 × 512	2,359,296
Conv13	8 × 8 × 512	3 × 3 × 512	512	1	8 × 8 × 512	2,359,296
Pool5	8 × 8 × 512	2 × 2 × 1	–	2	4 × 4 × 512	–
FC1	4 × 4 × 512	8,192	50	–	50	409,600
FC3	50	50	2	–	2	100

Table 4. Proposed architecture of Corrosion7.

Layer	Input	Kernel shape	Number of kernels	Stride	Output	Number of variables
Conv1	128 × 128 × 3	7 × 7 × 3	58	2	64 × 64 × 58	8526
Pool1	64 × 64 × 58	3 × 3 × 1	–	2	33 × 33 × 58	–
Conv2	33 × 33 × 58	5 × 5 × 58	128	2	17 × 17 × 128	185,600
Pool2	17 × 17 × 128	3 × 3 × 1	–	2	9 × 9 × 128	–
Conv3	9 × 9 × 128	3 × 3 × 128	192	1	9 × 9 × 192	221,184
Conv4	9 × 9 × 192	3 × 3 × 192	192	1	9 × 9 × 192	331,776
Conv5	9 × 9 × 192	3 × 3 × 192	128	1	9 × 9 × 128	221,184
pool3	9 × 9 × 128	3 × 3 × 128	–	2	4 × 4 × 128	–
FC1	4 × 4 × 128	2048	1024	–	1024	2,097,152
FC2	1024	1024	2	–	2	2048

Table 5. Proposed architecture of Corrosion5.

Layer	Input	Kernel shape	Number of kernels	Stride	Output	Number of variables
Conv1	128 × 128 × 3	7 × 7 × 3	96	2	64 × 64 × 96	14,112
Pool1	64 × 64 × 96	3 × 3 × 1	–	2	33 × 33 × 96	–
Conv2	33 × 33 × 96	5 × 5 × 96	256	2	17 × 17 × 256	614,400
Pool2	17 × 17 × 256	3 × 3 × 1	–	2	9 × 9 × 256	–
Conv3	9 × 9 × 256	3 × 3 × 256	384	1	9 × 9 × 384	884,736
FC1	9 × 9 × 384	31,104	1024	–	1024	31,850,496
FC2	1024	1024	2	–	2	2048

epochs. The batch size is the number of images per iteration. For each sliding window size, the batch size is 256 images. 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. The initial learning rate for all three window sizes is 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 of the images in

Table 6. A comparison of parameters between different architectures.

Dataset	Parameters
ZF Net	28,849,104
Corrosion7	3,067,470
Corrosion5	33,365,792
VGG16	65,050,304
VGG15	15,120,164

the training set. For 128 × 128 window sizes, the number of iterations used is 8800. It is increased by four for each decreasing window size due to the increase in the number of images, which is discussed in section "Dataset generation." The increase in iterations keeps the number of epochs constant at 22 for each window size. Therefore, for each sliding window size, the networks are trained on the same amount of image area. Since the training loss and validation loss had converged by 22 epochs while evaluating the hyperparameters' impact on training, the number of epochs was set to 22 during training. Further training on the network would risk overfitting, which is when the training loss and validation loss separate. Further discussion of how to find the best hyperparameters can be found in Bottou[43] and Bengio.[44]

Fine-tuning

Pretrained CNN architectures are typically evaluated on ImageNet,[30] which has millions of images and 1000 classes. Typically, as networks have gotten deeper, their accuracy has increased on ImageNet.[6] Corrosion classification as analyzed in this article is a binary classification and the dataset is significantly smaller. Furthermore, corrosion is not typically thought of as an "object," so in the paper, the effectiveness of fine-tuning is analyzed. Fine-tuning is when pretrained convolution layer weights are used as the initial weights to train a CNN on a new dataset. Pretrained weights are obtained from training a network on a very large dataset such as ImageNet so that the convolution layers learn robust features. The pretrained weights for ZF Net and VGG16 used in the paper were trained by their respective authors on the ImageNet.

Dataset generation

To create the dataset, 926 images were gathered using different digital cameras. These images were then cropped into regions that were fully corroded and regions that contained no corrosion. These regions were then split into 128 × 128 images, resulting in 33,039 corroded images and 34,148 non-corroded images. The images were then randomly divided into a training, validation, and test set with a target of 70% training, 10% validation, and 20% testing. In order to keep the sets independent, cropped regions from a particular image were included in only one set. For example, if there were 150 cropped images from a full size image, all 150 would be in only one of the datasets. For this reason, the final breakdown for each set does not exactly match the target percentages. Table 7 shows a sample breakdown of the training, validation, and testing datasets. Ten different combinations of training, validation, and test sets were generated.

To create the different sliding window sizes, the 128 × 128 images were divided into fourths for the 64 × 64 dataset and into sixteenths for the 32 × 32 dataset. Therefore, with each decrease in sliding window size, there is a corresponding increase in the number of images in the dataset. However, the actual amount of corroded area remains constant between all three sliding window sizes. Since the amount of training and testing area remains constant across all three sliding window sizes, their performance can be compared fairly.

The dataset of 33,039 corroded and 34,148 non-corroded images of 128 × 128 pixels, obtained from the original 926 images, is relatively small compared to the size of ImageNet. In this case, overfitting could be an issue. The use of pretrained networks described in section "Fine-tuning" can help limit overfitting. Since a network has already learned a significant number of parameters to extract features, only slight optimizations are required which can be performed with significantly less images.

Experimental results and discussion

This section details the results of training a CNN with different input parameters and CNN architectures.

Table 7. A sample of the number of images in each dataset when 128 × 128 sliding window is used.

Dataset	Corroded	Non-corroded	Total	Percentage
Training	24,806	25,280	50,086	74.55%
Validation	2517	2339	4856	7.23%
Testing	5716	6529	12,245	18.23%

Table 8. Performance of the different input parameters for ZF Net.

Window size	Color space	Recall (%)		Precision (%)		F1 score (%)	
		Mean	Stdev	Mean	Stdev	Mean	Stdev
128 × 128	RGB	96.89	1.42	96.68	0.53	96.78	0.65
128 × 128	YCbCr	96.08	1.28	97.37	1.18	96.75	0.83
128 × 128	CbCr	94.01	1.37	96.10	1.34	95.03	0.89
128 × 128	Grayscale	83.63	5.23	89.01	4.21	86.05	2.21
64 × 64	RGB	96.45	1.50	95.31	1.19	95.87	0.82
64 × 64	YCbCr	95.17	1.23	97.162	0.87	96.12	0.87
64 × 64	CbCr	93.49	1.75	96.17	1.27	94.79	0.93
64 × 64	Grayscale	84.77	3.34	87.62	2.36	86.14	2.27
32 × 32	RGB	90.59	3.53	93.50	2.28	91.99	2.31
32 × 32	YCbCr	93.03	1.33	96.59	0.96	94.77	1.05
32 × 32	CbCr	93.11	1.54	93.64	2.43	93.36	1.55
32 × 32	Grayscale	82.57	2.76	86.68	2.97	86.12	2.59

The recall, precision, F1 score, and computational performance of the different tests are evaluated. The equations for recall, precision, and F1 score can be seen in equations (2) to (4), respectively. F1 score is a measurement of classification performance that considers both the precision and recall. It is the harmonic mean of precision and recall where the arithmetic average on the reciprocal of precision and recall is performed. Precision and recall have different denominators but their reciprocals both have true positives as their denominators. This is why the harmonic mean is used

$$recall = \frac{\sum (true\ positives)}{\sum (true\ positives) + \sum (false\ negatives)} \quad (2)$$

$$precision = \frac{\sum (true\ positives)}{\sum (true\ positives) + \sum (false\ positives)} \quad (3)$$

$$F1 = \frac{2 \times recall \times precision}{recall + precision} \quad (4)$$

Caffe[45] is an open-source deep learning framework developed by researchers at the University of California, Berkley and was used to train the networks. The training was performed on three different computers. The main computer used was a server with four Nvidia Titan X Pascal GPUs, two Intel Xeon E5 CPUs, and 264 GB of RAM. The other two computers were a workstation with a Nvidia Gigabyte GTX 1070 GPU, 16 GB RAM, and an Intel i5 6600k CPU, and a computer with a Nvidia Quadro GPU, 32 GB of RAM, and an Intel Xeon CPU. All performance testing occurred on the server using a single Titan X GPU.

Effect of input image parameters

The results of using different color spaces and sliding window sizes with ZF Net can be seen in Table 8. Each combination was trained and tested 10 times using the 10 different generated training and testing datasets. The *t*-test is utilized to compare the effect of different parameters. The *t*-test assumes that the data are normally distributed. To verify that the data are normally distributed, the Kolmogorov–Smirnov test is performed on the data. The test did not reject the null hypothesis that the data are normally distributed. Therefore, it is acceptable to use a sample size of 10 in this study (i.e. the data are symmetric, unimodal, and without outliers).[46] Based on Central Limit Theorem, the objective of using a large sample size (e.g. n30) is to ensure that sample data are approximately normally distributed which is already satisfied with the sample size of 10 in this study. The best F1 score is achieved using the 128 × 128 window size. RGB, YCbCr, and CbCr color spaces show a decrease in the mean F1 score at each decreasing window size as shown in Figure 5(c). Grayscale does not demonstrate a change in mean F1 score, but does show increasing standard deviation as the window size decreases. Using the *t*-test, the RGB color space with a 128 × 128 sliding window size is shown to have a greater mean F1 score than the 64 × 64 window size with 99.3% confidence.

Using a 128 × 128 window size, RGB and YCbCr are shown to achieve a similar F1 score. Using the *t*-test, the means of the RGB color space with a 128 × 128 sliding window size and the YCbCr color space with a 128 × 128 window size are the same with 92.7% confidence. Using the RGB color space with a 128 × 128 sliding window size has a larger mean F1

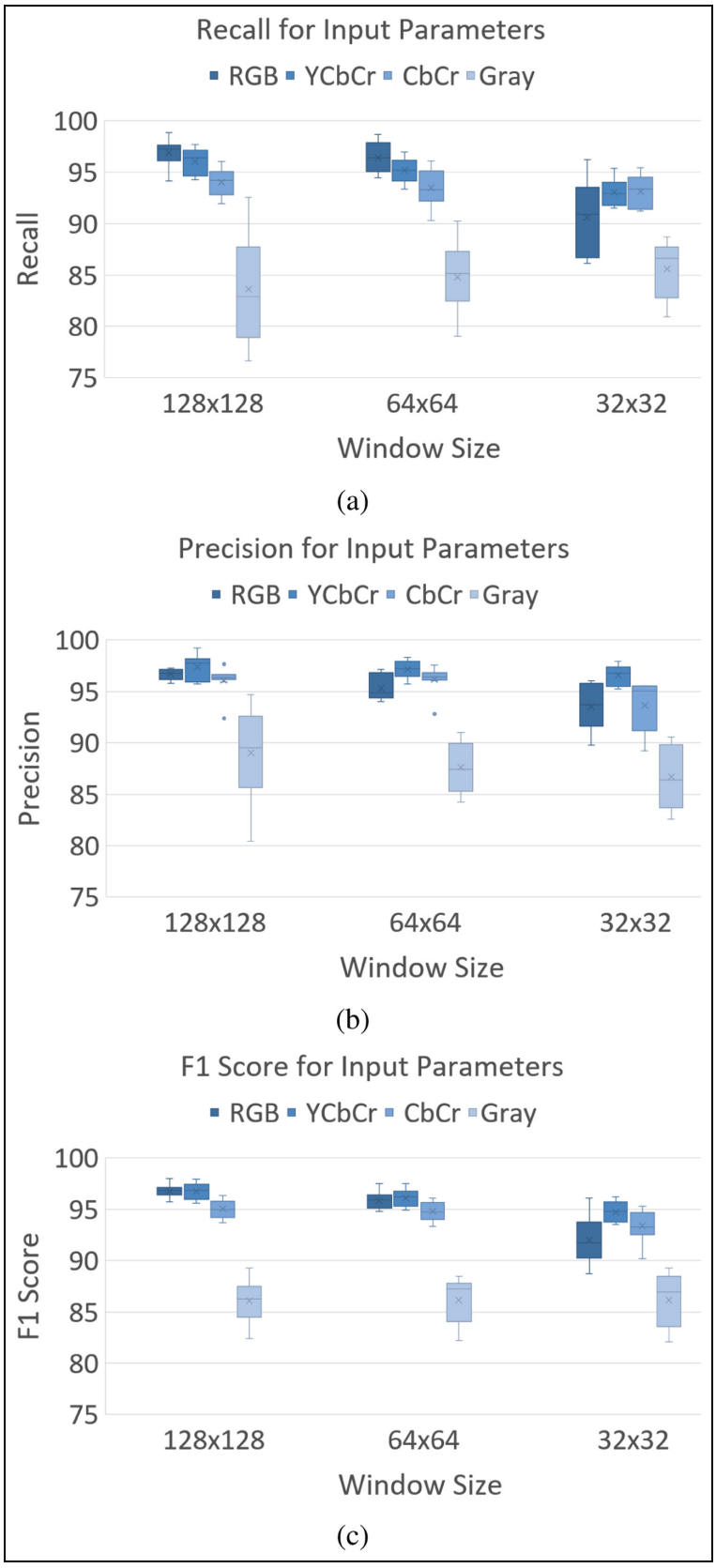

Figure 5. Effect of the input parameters in box plots where the bottom of the rectangle is the first quartile and the top is the third quartile: (a) box plot of recall for different input parameters, (b) box plot of precision for different input parameters, and (c) box plot of F1 score for different input parameters.

score than using the CbCr color space with a 128×128 sliding window size with over 99.9% confidence using the *t*-test. CNNs learn to identify the important features to detect corrosion during training. Since CbCr contains one less channel of input data compared to YCbCr, there are less features for a CNN to learn. Thus, it has worse performance than YCbCr. Furthermore, since RGB and YCbCr contain the same image but different representations, it was expected that a CNN should be able to identify the prominent features for each color space with similar abilities. This was the case at the 128×128 sliding window size; however, at the 32×32 sliding window size, the YCbCr color space has a greater mean F1 score compared to the RGB color space with 99.8% confidence. This means that ZF Net identifies small features within the YCbCr color space more effectively than RGB while at larger sliding window sizes there are enough features present for a CNN to perform similarly for either color space. It is also interesting to note even though the F1 scores are similar for RGB and YCbCr with a 128×128 sliding window size, the precision and recall for the two color spaces do differ slightly. Visually in Figure 5(a), it can be seen that the RGB color space tends to have a higher recall, but the YCbCr color space tends to have a higher precision, which is seen in Figure 5(b).

A visual representation of applying the trained CNNs to an image with a sliding window approach can be seen in Figure 6. The smaller sliding window size allows for a finer detection of corroded regions; however, it can result in more false positives and inaccuracies. This increase in false positives is seen when comparing Figure 6(d) with Figure 6(b) and Figure 6(p) with Figure 6(n). Visually, the different CNNs perform quite well overall.

Effect of CNN architectures

To compare the effect of the CNN architectures, ZF Net,[38] VGG16,[39] Corrosion7, and Corrosion5 are used. For fine-tuning, the weights were obtained from the pretrained models of ZF Net and VGG16 trained by their respective authors on the ImageNet dataset. The RGB color space was utilized for the architecture tests because it was shown to be the best along with YCbCr in section "Effect of input image parameters." Additionally, the RGB color space is what was used for the pretrained weights of ZF Net and VGG16. The visualization of the first convolutional layer kernels for each network is shown in Figure 7. The two networks that are fine-tuned are noticeably different than the networks that are not fine-tuned. Since the fine-tuned networks were trained on ImageNet, which has 1000 classes compared to the binary problem in this study, the networks learned a greater variety of features, such as edge, shape, color, and texture features as shown in Figure 7(b) and (f). The networks that were trained exclusively on the corrosion dataset primarily use color and texture features demonstrated in Figure 7(a) and (c)–(e). Additionally, the networks that are not fine-tuned have a much higher risk of overfitting. Figures 7(a) and (c)–(e) shows less variations in features and more signs of overfitting compared to the fine-tuned networks in Figures 7(b) and (f).

The results of the different networks and fine-tuning can be seen in Table 9. For ZF Net with a window size of 128×128, fine-tuning the network results in an observed mean F1 score that is less than without fine-tuning. However, using the *t*-test, it cannot be concluded that the means between the two are different. This means that ZF Net is able to learn robust features from the corrosion dataset. Using the *t*-test VGG16 with fine-tuning is shown to have a greater mean F1 score compared to both VGG16 without fine-tuning and ZF Net without fine-tuning with over 99.9% confidence. A visual comparison of F1 scores is demonstrated in Figure 8(c). VGG16 with fine-tuning tends to outperform at every window size for F1 score as well as for recall and precision as demonstrated in Figure 8(a) and (b), respectively.

Since VGG16 outperformed ZF Net on the ImageNet Challenge, it was expected to outperform ZF Net on the corrosion dataset. However, due to VGG16's depth and the relatively small size of the corrosion dataset compared to ImageNet, VGG16 needs to be fine-tuned for it to be able to learn the necessary features to outperform ZF Net. Without fine-tuning, VGG16 tends to perform similar to ZF Net which is demonstrated in Figure 8. The reason is that the corrosion dataset does not include enough images for VGG16 to outperform ZF Net without fine-tuning. VGG16 has 14,710,464 variables associated with the convolutional layers compared to ZF Net's 3,725,088 variables. If the corrosion dataset was to approach the scale of ImageNet, it is expected that VGG16 without fine-tuning would outperform ZF Net and the impact of fine-tuning would be minimized similar to what was shown with ZF Net in this experiment. The reason is that the VGG16 architecture has a greater number of variables to optimize compared to ZF Net.

The two proposed networks, Corrosion7 and Corrosion5, perform with similar mean F1 scores to ZF Net according to the *t*-test with sliding window sizes of 128×128 and 64×64 and the RGB color space. The goal of these two networks was to decrease the computational time while maintaining similar signal-to-noise ratio as state-of-the-art architectures for the ImageNet Challenge. The computational

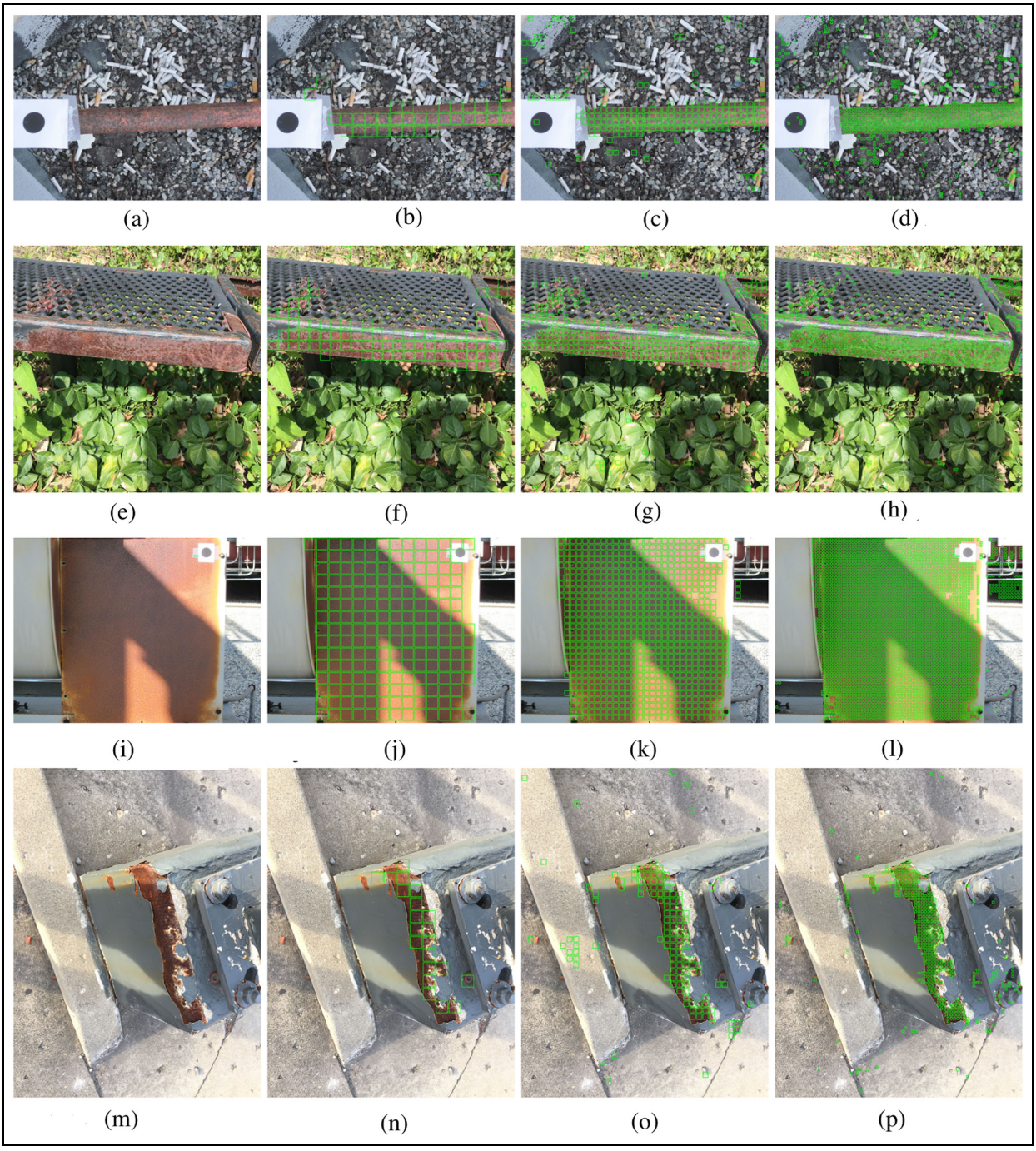

Figure 6. Corrosion detection using ZF Net applied to RGB sample images where green boxes are regions identified as corrosion: (a) original image; (b) ZF Net, 128 × 128; (c) ZF Net, 64 × 64; (d) ZF Net, 32 × 32; (e) original image; (f) ZF Net, 128 × 128; (g) ZF Net, 64 × 64; (h) ZF Net, 32 × 32; (i) original image; (j) ZF Net, 128 × 128; (k) ZF Net, 64 × 64; (l) ZF Net, 32 × 32; (m) original image; (n) ZF Net, 128 × 128; (o) ZF Net, 64 × 64; and (p) ZF Net, 32 × 32.

performance results of the networks are discussed in section "Computational performance." These networks were able to achieve similar signal-to-noise ratios at every window size compared to ZF Net as well as compared to each other. Using a 32 × 32 window size, the recorded mean F1 score for Corrosion5 is greater than ZF Net and Corrosion7 as shown in Table 9; however, using the *t*-test, it cannot be concluded that the means are different. Figure 8(c) demonstrates that although the recorded mean F1 score for Corrosion5 at the

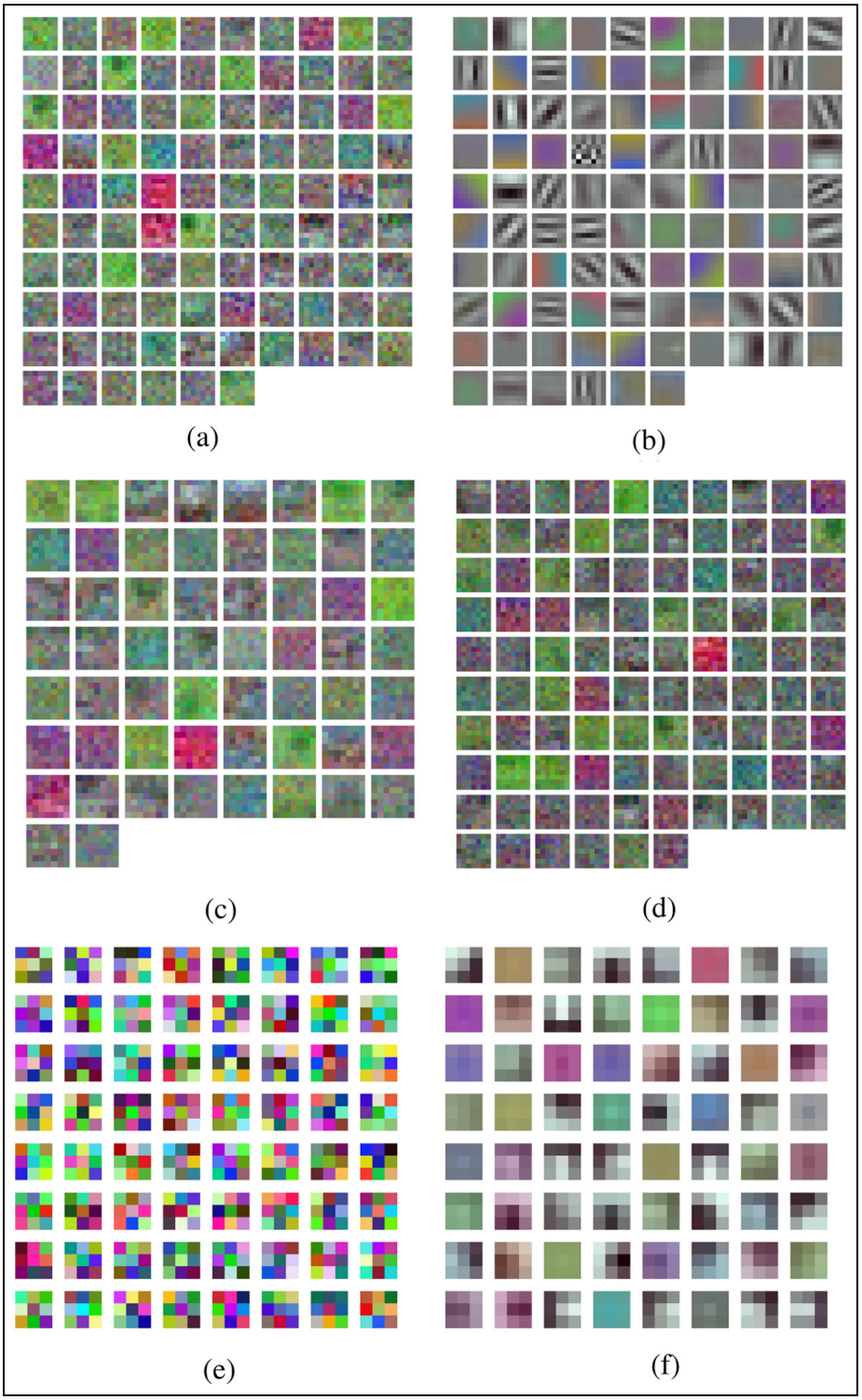

Figure 7. Visualization of the first convolutional layer kernels for each network: (a)–(d) are maps of 7 × 7 kernels where as (e) and (f) are 3 × 3. (a) ZF Net, (b) ZF Net Fine-tuned, (c) Corrosion7, (c) Corrosion5, (e) VGG16, and (f) VGG16 Fine-tuned.

32 × 32 window size is greater, the F1 score has a large standard deviation and the box plots of different networks have significant overlaps. These results indicate that for the binary classification of corrosion, the CNN architecture can be smaller compared to architectures designed for ImageNet while achieving similar signal-to-noise ratios. The similarities can be viewed in the box plots of Figure 8.

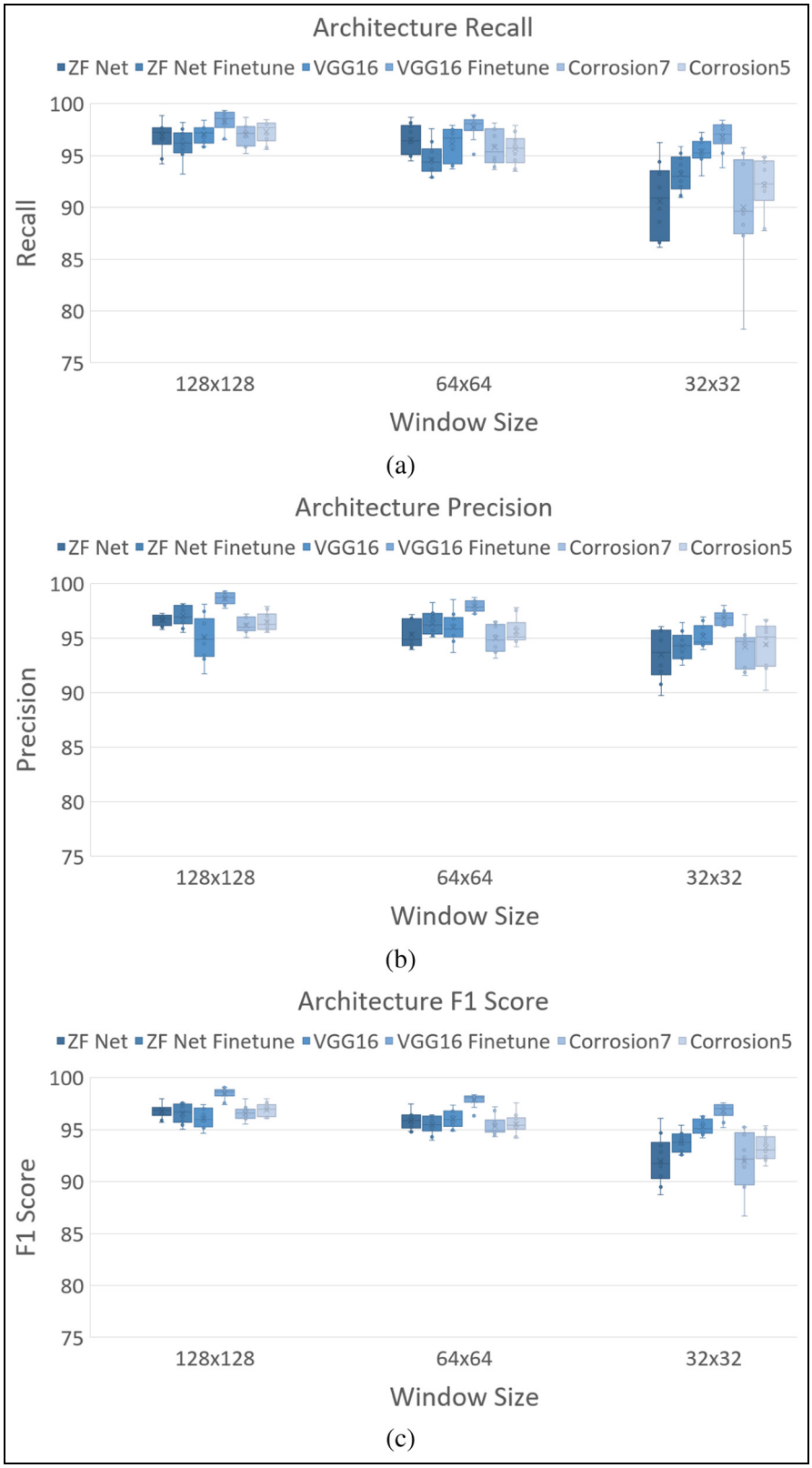

Figure 8. Effect of CNN architectures where the bottom of the rectangle is the first quartile and the top is the third quartile: (a) box plot of recall for different CNN architectures, (b) box plot of precision for different CNN architectures, and (c) box plot of F1 score for different CNN architectures.

Table 9. Performance of different CNN architectures on RGB images.

Network	Window size	Recall (%)		Precision (%)		F1 score (%)	
		Mean	Stdev	Mean	Stdev	Mean	Stdev
ZF Net	128 × 128	96.86	1.42	96.68	0.53	96.78	0.65
ZF Net Fine-tuned	128 × 128	96.11	1.42	97.04	0.92	96.57	0.92
VGG16	128 × 128	97.05	0.87	95.10	2.02	96.05	0.95
VGG16 Fine-tuned	128 × 128	98.31	1.02	98.64	0.57	98.47	0.57
Corrosion7	128 × 128	97.01	1.07	96.17	0.70	96.58	0.68
Corrosion5	128 × 128	97.32	1.05	96.47	0.84	96.89	0.65
ZF Net	64 × 64	96.45	1.50	95.31	1.19	95.87	0.82
ZF Net Fine-tuned	64 × 64	94.61	1.49	96.34	1.09	95.46	0.83
VGG16	64 × 64	96.11	1.60	95.99	1.39	96.04	0.86
VGG16 Fine-tuned	64 × 64	97.76	1.14	97.93	0.55	97.84	0.65
Corrosion7	64 × 64	95.78	1.68	94.99	1.30	95.33	0.96
Corrosion5	64 × 64	95.62	1.47	65.64	1.20	95.55	0.97
ZF Net	32 × 32	90.59	3.53	93.50	2.28	91.99	2.31
ZF Net Fine-tuned	32 × 32	93.20	1.73	94.30	1.28	93.73	0.99
VGG16	32 × 32	95.35	1.19	95.16	1.04	95.25	0.76
VGG16 Fine-tuned	32 × 32	96.80	1.40	96.83	0.66	96.81	0.80
Corrosion7	32 × 32	90.00	5.29	94.20	1.74	91.96	2.76
Corrosion5	32 × 32	92.15	2.57	94.45	2.13	93.25	1.29

Table 10. Performance of different algorithms using 128 × 128 window.

Algorithm	Color space	Recall (%)		Precision (%)		F1 score (%)	
		Mean	Stdev	Mean	Stdev	Mean	Stdev
VGG16 Fine-tuned	RGB	98.31	1.02	98.64	0.57	98.47	0.57
VGG15 Fine-tuned	RGB	98.10	1.32	98.76	0.49	98.42	0.68
Wavelet NN	RGB	93.01	1.77	93.65	0.95	93.32	1.14
Wavelet NN	YCbCr	85.41	3.12	89.85	1.53	87.53	2.27
Wavelet NN	CbCr	75.85	4.12	84.87	5.59	80.03	4.29

CNN features versus wavelet features

In order to compare a CNN approach with the existing state of the art, VGG16 with fine-tuning was compared to the wavelet NN proposed by Jahanshahi and Masri.[7] The wavelet NN approach proposed was optimal with 50 neurons in a single hidden layer. This is smaller than the size of the hidden fully connected layers of the CNNs used in this article. In order to compare the robustness of the features obtained by a CNN to the wavelet features, the VGG16 architecture was modified. One fully connected layer was removed and another was reduced to 50 neurons. The convolution layers were left unchanged. This created VGG15, whose fully connected layers resembled that of the wavelet NN. The primary difference between the two approaches was the features being input into the fully connected layers. The results from the tests can be seen in Table 10. Each scenario was repeated 10 times, once each on the 10 different datasets. Using the *t*-test, VGG15 with fine-tuning has a greater mean than the wavelet NN approach with over 99.9% confidence regardless of the color space used for the wavelet NN. From this, it can be concluded that using CNN, features outperform the use of wavelet features. As shown in Table 8, a CNN approach performs the best with the RGB or YCbCr color space. To utilize the benefits of fine-tuning, the RGB color space was used.

Another interesting observation through these tests was that using the VGG convolution layers with the fully connected layers proposed previously, VGG15 performs similarly to VGG16. Binary classification of corrosion is less complex of a problem than classifying the 1000 classes in ImageNet that VGG16's fully connected layers were designed for. Therefore, the most significant aspect of using a CNN architecture is the improved features generated from the convolution layers. The size increase in fully connected layers of modern CNNs compared to previous NN sizes have little impact on the classification of corrosion.

Table 11. Computation performance of different networks with 256 images per batch, 1 batch per iteration, and averaged over 50 iterations on a Nvidia Titan X Pascal GPU.

Network	Window size	Forward pass (ms)	Backward pass (ms)	Training time (ms)
ZF Net	128 × 128	70	246	316
VGG16	128 × 128	252	322	574
VGG15	128 × 128	245	313	557
Corrosion7	128 × 128	33	132	165
Corrosion5	128 × 128	60	219	279
ZF Net	64 × 64	22	68	90
VGG16	64 × 64	68	85	153
VGG15	128 × 128	65	80	145
Corrosion7	64 × 64	11	37	48
Corrosion5	64 × 64	16	59	75
ZF Net	32 × 32	10	23	33
VGG16	32 × 32	24	47	71
VGG15	128 × 128	22	44	67
Corrosion7	32 × 32	4	12	17
Corrosion5	32 × 32	6	18	24

Computational performance

The computational performance of the networks was evaluated on a Nvidia Titan X Pascal GPU. A batch size of 256 images was used for the performance testing. The performance results are summarized in Table 11. To train a network, both the forward and backward pass through the network are used. The expected total training time is the training time per batch multiplied by the number of iterations. For instance, for 8800 iterations, ZF Net, and a 128 × 128 sliding window size, the expected training time is a little over 46 min. In practice, this number will be slightly longer and vary as the network will be tested on a validation set periodically. To classify an image with the network after training, only the forward pass is utilized. VGG16 is the largest network and is the slowest, which is expected. VGG15 has one less and smaller fully connected layers and achieves a minor speed increase at each window size compared to VGG16. ZF Net is three times as fast compared to VGG16 in the forward pass. Corrosion7 and Corrosion5 were designed to improve the computational performance compared with ZF Net. Corrosion7 is able to cut the forward pass time in half at every window size compared to ZF Net's time. In addition, since Corrosion7 can classify 256 windows in 33 ms, it can be utilized for corrosion detection on an image of 2048 × 2048 pixels at 30 fps, the typical frame rate for a video feed. Corrosion5 did not experience as significant of a computational performance increase as corrosion7 did when the two are compared to ZF Net.

As shown in Table 11, at each decreasing sliding window size, the number of images increases by four times to cover the same amount of area. However, the times at each decreasing sliding window size do not decrease by four times, which means that it is faster to use a larger window size to analyze an image for corrosion. The computational time does not decrease at the same rate due to the parallel nature of CNNs and GPUs. With a larger image, there are more opportunities to perform computations in parallel.

Summary and conclusions

CNN is a powerful machine learning approach for many image detection and classification tasks. This study examined CNNs for corrosion detection of a sliding window over an image. Several input parameters for the region input into the CNN were evaluated. The impact of sliding window sizes of 128 × 128, 64 × 64, and 32 × 32 pixels, and the color spaces RGB, YCbCr, CbCr, and grayscale were studied. In addition, two state-of-the-art CNN architectures, ZF Net and VGG16, were evaluated and compared to three proposed CNNs, Corrosion7, Corrosion5, and VGG15, for corrosion detection. Additionally, ZF Net and VGG16 architectures were trained with random initialization of weights and pretrained weights from CNNs trained on ImageNet to evaluate the effect of fine-tuning.

The use of a CNN was shown to outperform the previous state-of-the-art corrosion detection approach where wavelet features were used with a NN. The most robust input parameters found were a sliding window size of 128 × 128 and the RGB or YCbCr color space. VGG16, the deeper network, with fine-tuning was shown to be the most robust architecture. Additionally, it was shown that VGG16 was the most robust due to its convolution layers and not due to its increase in fully connected layer size. It was observed that without fine-tuning, ZF Net and VGG16 performed similarly.

It is expected that with a dataset approaching the size of ImageNet, VGG16 without fine-tuning would outperform ZF Net since VGG16 with fine-tuning outperformed ZF Net with fine-tuning. Additionally, smaller architectures such as the proposed Corrosion7 and Corrosion5 were able to achieve a similar signal-to-noise ratio as ZF Net. The largest effect from the different architectures was in the computational performance. Both VGG16 and VGG15 can provide a more robust classification but at the cost of a slowdown. Corrosion7 has a forward pass time that is twice as fast as ZF Net and over seven times faster than VGG16.

In future work, the use of CNNs in the study can be expanded to classify different types of corrosion. For the classification of multiple types of corrosion, the number of neurons in the fully connected layers is expected to have more significance in performance. Further evaluation of the number of neurons needed for the number of corrosion classes would be an interesting evaluation. Furthermore, CNNs can be utilized to quantify the area of the corroded region using the sliding window approach in this article and through continuing this research for corrosion segmentation. Having the ability to automatically detect corrosion, identify the type, and quantify the amount of corrosion will aid a human inspector and significantly cut down on the time and cost associated with inspecting civil infrastructure.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This study was supported in part by the Purdue Honors College.

References

1. Koch G, Varney J, Thompson N, et al. *International measures of prevention, application, and economics of corrosion technologies study*. Houston, TX: NACE International, 2016.
2. Metni N and Hamel TA. UAV for bridge inspection: visual servoing control law with orientation limits. *Automat Constr* 2007; 17(1): 3–10.
3. Nikolic J, Burri M, Rehder J, et al. A UAV system for inspection of industrial facilities. In: *Proceedings of the 2013 IEEE aerospace conference*, Big Sky, MT, 2–9 March 2013, pp. 1–8. New York: IEEE.
4. Li Z, Liu Y, Hayward R, et al. Knowledge-based power line detection for uav surveillance and inspection systems. In: *Proceedings of the 23rd international conference on image and vision computing*, Christchurch, New Zealand, 26–28 November 2008, pp. 1–6. New York: IEEE.
5. Krizhevsky A, Sutskever I and Hinton GE. ImageNet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L, et al. (eds) *Proceedings of the 25th international conference on neural information processing systems*, Lake Tahoe, NV, 3–6 December 2012, pp. 1097–1105. Red Hook, NY: Curran Associates.
6. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: *Proceedings of the 2016 IEEE conference on computer vision and pattern recognition*, Las Vegas, NV, 27–30 June 2016. New York: IEEE.
7. Jahanshahi MR and Masri SF. Parametric performance evaluation of wavelet-based corrosion detection algorithms for condition assessment of civil infrastructure systems. *J Comput Civil Eng* 2013; 27(4): 345–357.
8. Haralick RM, Shanmugam K and Dinstein I. Textural features for image classification. *IEEE T Syst Man Cyb* 1973; SMC-3(6): 610–621.
9. Valois RLD, Albrecht DG and Thorell LG. Spatial frequency selectivity of cells in macaque visual cortex. *Vision Res* 1982; 22(5): 545–559.
10. Xie X. A review of recent advances in surface defect detection using texture analysis techniques. *Elecron Lett Comput Vision Image Anal* 2008; 7(3): 1–22.
11. Furuta H, Deguchi T and Kushida M. Neural network analysis of structural damage due to corrosion. In: *Proceedings of the 3rd international symposium on uncertainty modeling and analysis and annual conference of the North American Fuzzy Information Processing Society*, College Park, MD, 17–20 September 1995, pp. 109–114. New York: IEEE.
12. Livens S, Scheunders P, Van de Wouwer G, et al. Classification of corrosion images by wavelet signatures and LVQ networks. In: *Proceedings of the 6th international conference on computer analysis of images and patterns*, Prague, 6–8 September 1995, pp. 538–543. London: Springer-Verlag.
13. Gunatilake P, Siegel M, Jordan AG, et al. Image understanding algorithms for remote visual inspection of aircraft surfaces. In: *Proceedings of volume 3029, machine vision applications in industrial inspection V*, San Jose, CA, 15 April 1997, pp. 2–13. Bellingham, WA: SPIE.
14. Daubechies I. *Ten lectures on wavelets*. Philadelphia, PA: SIAM, 1992.
15. Gunatilake P and Siegel M. Remote enhanced visual inspection of aircraft by a mobile robot. In: *Proceedings of the 1998 IMTC conference*, St. Paul, MN, 18–21 May 1998, pp. 49–58. New York: IEEE.
16. Strang G and Nguyen T. *Wavelets and filter banks*. Wellesley, MA: Wellesley-Cambridge Press, 1996.
17. Choi K and Kim S. Morphological analysis and classification of types of surface corrosion damage by digital image processing. *Corrosion Sci* 2005; 47(1): 1–15.
18. Lee S, Chang LM and Skibniewski M. Automated recognition of surface defects using digital color image processing. *Automat Constr* 2006; 15(4): 540–549.

19. Palakal MJ, Pidaparti RMV, Rebbapragada S, et al. Intelligent computational methods for corrosion damage assessment. *AIAA J* 2001; 39(10): 1936–1943.
20. Pidaparti RM. Structural corrosion health assessment using computational intelligence methods. *Struct Health Monit* 2007; 6(3): 245–259.
21. Medeiros FN, Ramalho GL, Bento MP, et al. On the evaluation of texture and color features for nondestructive corrosion detection. *EURASIP J Adv Sig Pr* 2010; 2010(1): 817473.
22. Bonnin-Pascual F and Ortiz A. Combination of weak classifiers for metallic corrosion detection and guided crack location. In: *Proceedings of 2010 IEEE conference on emerging technologies and factory automation*, Bilbao, 13–16 September 2010, pp. 1–4. New York: IEEE.
23. Pidaparti RM, Aghazadeh BS, Whitfield A, et al. Classification of corrosion defects in NiAl bronze through image analysis. *Corrosion Sci* 2010; 52(11): 3661–3666.
24. Shannon CE. A mathematical theory of communication. *Bell Syst Tech J* 1948; 27(3): 379–423.
25. Pakrashi V, Schoefs F, Memet JB, et al. ROC dependent event isolation method for image processing based assessment of corroded harbour structures. *Struct Infrastruct Eng* 2010; 6(3): 365–378.
26. Ghanta S, Karp T and Lee S. Wavelet domain detection of rust in steel bridge images. In: *Proceedings of 2011 IEEE international conference on acoustics, speech and signal processing*, Prague, 22–27 May 2011, pp. 1033–1036. New York: IEEE.
27. Jahanshahi MR, Kelly JS, Masri SF, et al. A survey and evaluation of promising approaches for automatic image-based defect detection of bridge structures. *Struct Infrastruct Eng* 2009; 5(6): 455–486.
28. Oullette R, Browne M and Hirasawa K. Genetic algorithm optimization of a convolutional neural network for autonomous crack detection. In: *Proceedings of the congress on evolutionary computation*, Portland, OR, 19–23 June 2004, pp. 516–521. New York: IEEE.
29. LeCun Y, Boser B, Denker JS, et al. Backpropagation applied to handwritten zip code recognition. *Neural Comput* 1989; 1(4): 541–551.
30. Deng J, Dong W, Socher R, et al. ImageNet: a large-scale hierarchical image database. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Miami, FL, 20–25 June 2009. New York: IEEE.
31. Ren S, He K, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks. In: *Proceedings of the 28th international conference on neural information processing systems*, Montreal, QC, Canada, 7–12 December 2015, pp. 91–99. Cambridge, MA: MIT Press.
32. Sermanet P, Eigen D, Zhang X, et al. OverFeat: integrated recognition, localization and detection using convolutional networks, CoRR abs/1312.6229, 2013.
33. Long J, Shelhamer E and Darrell T. Fully convolutional networks for semantic segmentation, CoRR abs/1411.4038, 2014.
34. Donahue J, Hendricks LA, Guadarrama S, et al. Long-term recurrent convolutional networks for visual recognition and description, CoRR abs/1411.4389, 2014.
35. Payan A and Montana G. Predicting Alzheimer's disease: a neuroimaging study with 3D convolutional neural networks, CoRR abs/1502.02506, 2015.
36. Nair V and Hinton GE. Rectified linear units improve restricted Boltzmann machines. In: Frnkranz J and Joachims T (eds) *Proceedings of the 27th international conference on international conference on machine learning*, Haifa, 21–24 June 2010, pp. 807–814. Madison, WI: Omnipress.
37. Hinton GE, Srivastava N, Krizhevsky A, et al. Improving neural networks by preventing co-adaptation of feature detectors, CoRR abs/1207.0580, 2012.
38. Zeiler MD and Fergus R. Visualizing and understanding convolutional networks, CoRR abs/1311.2901, 2013.
39. Simonyan K and Zisserman A. Very deep convolutional networks for large-scale image recognition, CoRR abs/1409.1556, 2014.
40. Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions, CoRR abs/1409.4842, 2014.
41. Bojarski M, Testa DD, Dworakowski D, et al. End to end learning for self-driving cars, CoRR abs/1604.07316, 2016.
42. Ng CB, Tay YH and Goi BM. Comparing image representations for training a convolutional neural network to classify gender. In: *Proceedings of the 1st international conference on artificial intelligence, modeling and simulation*, Kota Kinabalu, Malaysia, 3–5 December 2013, pp. 29–33. New York: IEEE.
43. Bottou L. *Stochastic gradient descent tricks*. Berlin; Heidelberg: Springer, 2012, pp. 421–436.
44. Bengio Y. Practical recommendations for gradient-based training of deep architectures, CoRR abs/1206.5533, 2012.
45. Jia Y, Shelhamer E, Donahue J, et al. Caffe: convolutional architecture for fast feature embedding. In: *Proceedings of the 22nd ACM international conference on multimedia*, Orlando, FL, 3–7 November 2014, pp. 675–678. New York: ACM.
46. El-Hajjar ST. *Basic & business course in statistics II*. Bloomington, IN: Author House, 2014.