ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 11 Issue 3 March 2024

Application of Edge Detection to Image Classification for Wind Turbine Blade Defects

[1] Zachary Ward, [2] Jeremiah Engel, [3] Mohammad A.S. Masoum, [4] Mohammad Shekaramiz, [5] Abdennour Seibi

[1] [2] [3] [4] [5] Engineering Department, Utah Valley University Orem, UT 84058, USA
Corresponding Author Email: [1] Zachary.Ward@uvu.edu, [2] Jeremiah.Engel@uvu.edu, [3] mmasoum@ieee.org, [4] mshekaramiz@uvu.edu, [5] aseibi@uvu.edu

Abstract*— Wind turbines can become damaged during operation, and wind turbine blades are especially susceptible. Machine learning algorithms are often used to classify images, and these images are commonly processed prior to their use. One of these preprocessing methods is edge detection, which isolates the areas of an image that contain high-frequency information, such as edges. This paper explores the effect of applying edge detection as a preprocessing method for machine learning algorithms trained to classify defects in images of wind turbine blades. Specifically, edge detection is applied to the Xception and VGG19 convolutional neural networks. Conclusions as to the efficacy of edge detection as a preprocessing method for this type of data are drawn by comparing the classical performance of the selected machine learning algorithms to the performance of the same algorithms after implementing edge detection. If found to be successful, this technique can be used to improve automated detection of faulty wind turbines, which has implications for the reduction of energy and revenue loss at wind farms due to wind turbine downtime.

Index Terms*— Deep learning, edge detection, VGG19, wind turbine, Xception.

I. INTRODUCTION

The demand for renewable energy has increased. This is especially true of renewable energy generated from wind through wind farms. In the last three years (2020-2022), wind power has experienced its top three years of growth in history [1]. This conversion of wind energy to sustainable electricity is accomplished using wind turbines. Despite the increased popularity of the energy produced by these machines, wind turbines are susceptible to damage, especially by the environment they reside in [2, 3, 4]. The blades of the wind turbine are no exception. In a survey studying over 1,000 wind turbines across a period of fifteen years, the wind turbine blades were found to cause over 5% of wind turbine failures; failures such as these can result in downtimes greater than ten days [5, 6].

Wind turbine failures and downtime can be prevented by condition monitoring. Unmanned aerial vehicles and autonomous drones are a growing area of interest in industry and academia for addressing this need [7, 8, 9, 10]. Machine learning has been investigated through algorithms such as support vector machines (SVMs) and convolutional neural networks (CNNs) as a successful method to enable these vehicles to perform condition monitoring without human intervention [11, 12, 13]. One of the key aspects of condition monitoring is the identification of cracks and other defects on a wind turbine's blades. This research aims to evaluate the efficacy of edge detection, a data preprocessing method, for wind turbine blade crack detection by comparing the performance of Xception and VGG19, two common CNN architectures, implementing this preprocessing stage to the classical performance of those architectures on a dataset containing 6,000 images of healthy and faulty wind turbine blades. If successful, this method will increase the accuracy of machine learning architectures in detecting defects on wind turbine blades and thus benefit the field of autonomous condition monitoring of wind turbines.

In the remainder of this paper, the methodology guiding the evaluation process of edge detection is explained in Section II, the results are presented in Section III, and conclusions are drawn and recommendations made in Section IV.

II. METHODOLOGY

Edge detection will be evaluated by training the Xception and VGG19 CNN architectures on a dataset containing images of healthy and faulty wind turbine blades. An overview of this process is provided in Fig. 1. This section presents an elaboration on the Xception and VGG19 CNN architectures, the wind turbine dataset, edge detection, and the procedure used to evaluate edge detection as a data preprocessing method.

A. Xception

Xception is an implementation of the CNN. CNNs are neural networks consisting of pooling, fully-connected, and convolutional layers. Each layer is connected to the layers directly preceding and following it, and each layer is formed from weight-storing nodes. These weights are gradually updated during the model training process and enable the model to make inferences on new data. Each type of layer performs specific functionality. The pooling layer reduces the size of data it receives, and the fully-connected layer implements a mapping of each input node with all of the

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 11 Issue 3 March 2024

nodes on its output. The namesake of the CNN is the convolutional layer. It performs the convolution operation on the matrix of 2-dimensional (2D) data it receives. During this operation, the elementwise (Hadamard) product is computed between every region of the input data and another 2D matrix known as the kernel. The summation of all elements of the resulting matrix constitutes an entry in the convolutional layer's output feature map. The full feature map is formed as this operation is repeated for every region of the input matrix [14].

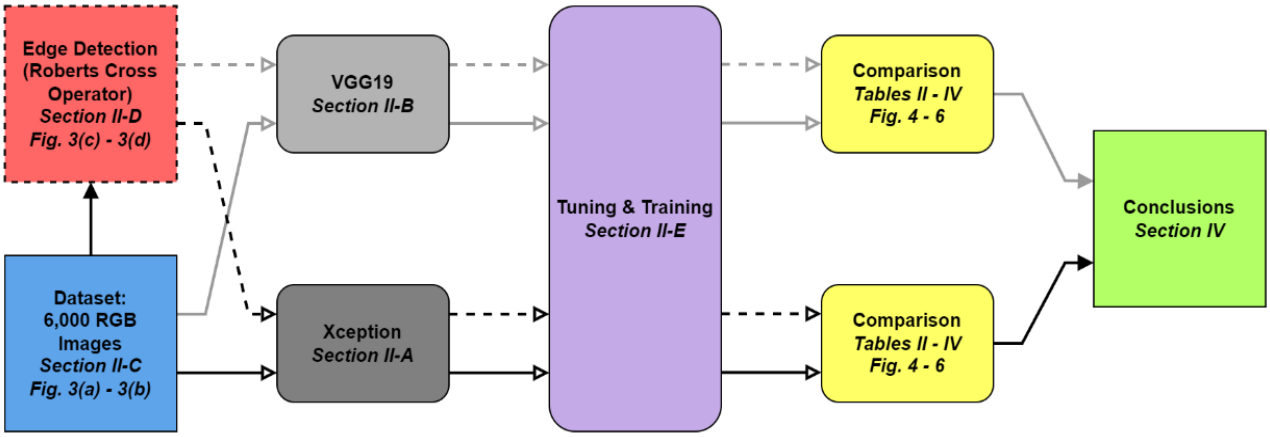


Fig. 1. Implementation, training, and performance of Xception and VGG19 machine learning algorithms with and without edge detection preprocessing.

The core idea distinguishing Xception from other CNNs is the Xception module, which replaces the typical convolutional layer. This module implements layers performing depthwise separable convolution, which separates the computation of spatial and cross-channel correlations. Xception also includes residual connections between layers [15]. A visualization of the Xception architecture as implemented in this paper is shown in Fig. 2.

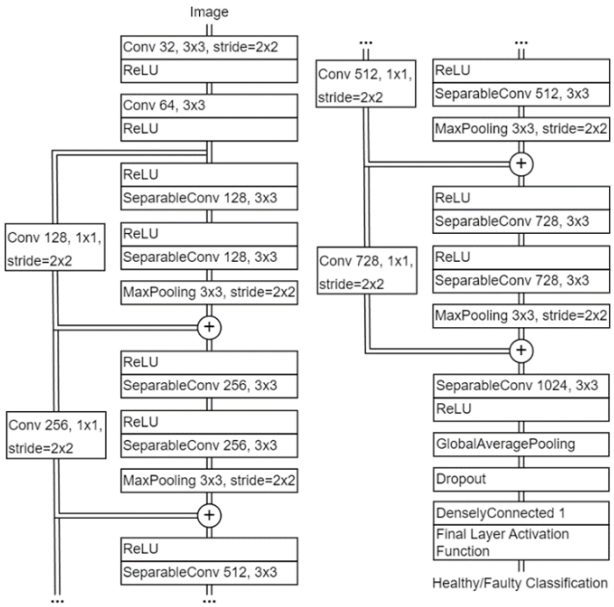


Fig. 2. Structure of the Xception architecture implemented in this reserch.

B. VGG19

VGG19 is another implementation of the CNN. The VGG network is one of the two CNN architectures explored in this paper, a fundamental yet renowned machine learning algorithm introduced by Andrew Zisserman and Karen Simonyan [16]. The VGG model analyzes the depth of layers using a relatively small convolutional filter size (3 × 3) [17]. The pre-trained Visual Geometry Group model 19 (VGG19) was trained on the ImageNet database of roughly 14,197,122 images that are categorized according to the WordNet hierarchy. All images in this dataset are RGB with size 224 × 224 [18]. The pre-trained model serves as a base for which wind turbine fault image processing can be added to the classifier.

The first sixteen layers of VGG19 are convolution layers, and the last three layers are dense or fully connected layers [18]. Five blocks of convolution are present in VGG19. Each block is combined with one MaxPool Layer. Block 1: The depth of filters is 64 with two convolution layers. Block 2: The depth of filters is 128 with two convolution layers. Block 3: The depth of filters is 256 with four convolution layers. Block 4 and Block 5: The depth of filters is 512 with four convolution layers [19]. The model uses kernels of size (3 × 3) with a stride size of one pixel, which allows the entire construct of each image to be covered. Spatial padding is used to preserve the spatial resolution of the image, and max pooling is performed over a (2 × 2) pixel window with a stride of 2. A rectified linear unit (ReLU) layer introduces non-linearity after those mentioned previously. This allows the model to classify more effectively, and it improves computational and running time. Three fully connected layers are used at the end: the first two are of size 4096; the third contains 1000 channels. Finally, a softmax function layer completes the architecture. The architecture of the model is summarized in detail in Table I [20].

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 11 Issue 3 March 2024

Table I. Vgg19 Layers Architecture.

Layer #	Layer Details	Layer #	Layer Details
1	Conv3x3 (64)	11	Conv3x3 (512)
2	Conv3x3 (64)	12	Conv3x3 (512)
-	MaxPool	-	MaxPool
3	Conv3x3 (128)	13	Conv3x3 (512)
4	Conv3x3 (128)	14	Conv3x3 (512)
-	MaxPool	15	Conv3x3 (512)
5	Conv3x3 (256)	16	Conv3x3 (512)
6	Conv3x3 (256)	-	MaxPool
7	Conv3x3 (256)	17	Fully Connected (4096)
8	Conv3x3 (256)	18	Fully Connected (4096)
-	MaxPool	19	Fully Connected (1000)
9	Conv3x3 (512)	-	SoftMax
10	Conv3x3 (512)	-	-

C. Dataset

In order to train the Xception and VGG19 CNNs on wind turbine defect classification and to evaluate the impact of edge detection as a data preprocessing method, a dataset of healthy and faulty wind turbine blades was created. In both categories, the subject of the image is a small-scale wind turbine prototype assembled at Utah Valley University (UVU). In total, the dataset contains 6,000 red-green-blue (RGB) images, of which 3,000 belong to the healthy class, and the other 3,000 belong to the faulty class. Wind turbine blades in the faulty class of images are distinguished from those in the healthy class by artificially generated defects including cracks, erosion, and holes. Additionally, both classes contain images captured outside using a Zenmuse L1 RGB camera connected to a DJI Matrice 300 RTK drone where the blades are mounted on the wind turbine prototype. Images are also included in the dataset that were captured inside with the blades isolated from the wind turbine prototype.

For training and evaluation, each image in the dataset was downscaled to a size of 300 × 300. This had the effect of decreasing the training time of the CNN models without impacting the performance comparison between the standard models and the models modified with the edge detection preprocessing technique. Furthermore, the dataset was divided into training, testing, and validation subsets: 4,200 images were allocated for the training subset, 1,200 for the testing subset, and 600 for the validation dataset. This allowed the evaluation of edge detection to proceed using fresh data previously unseen by the models. This process will be described in detail in Section II-E.

A sample of the wind turbine dataset is provided in Fig. 3 Particularly, Fig. 3(a) contains images from the healthy class of wind turbine blades, and Fig. 3(b) contains images from the faulty class. The two leftmost images in each category are taken from the subset of images captured outside; the rightmost images are from the subset captured inside.

D. Edge Detection

Edge detection is a class of algorithms that approximate the magnitude of an image's gradient. As a result, these algorithms output a 2D matrix where each element is a value between 0 and 1 and corresponds to the rate of change of the pixel intensity in the original image at that point. Values closer to 0 in the output matrix indicate a location in the original image with slowly changing features while values closer to 1 indicate a location with features changing more quickly. This means that edges in the original image, which are locations that experience sudden changes in pixel intensity, have higher values in the output matrix; the other, non-edge pixels have lower values.

Many algorithms exist for edge detection calculation: this research implements the Roberts cross operator through the scikit-image Python library for its computational simplicity. In this algorithm, the original image is convolved with two kernels: the positive diagonal kernel given in (1) and the negative diagonal kernel given in (2). Afterwards, the resulting matrices are combined to form the final output matrix using (3), where $\mathbf{E}_p$ is the matrix resulting from the convolution of the original image with (1), $\mathbf{E}_n$ is the matrix resulting from the convolution of the original image with (2), and $\mathbf{E}$ is the matrix output from the edge detection algorithm [21].

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{2}$$

$$\mathbf{E} = \sqrt{\frac{\mathbf{E}_p{}^2 + \mathbf{E}_n{}^2}{2}} \tag{3}$$

This algorithm requires a 2D matrix; however, the wind turbine dataset was captured in RGB format. This requires a 3D matrix, as the extra dimension is used to store information from the red, green, and blue data channels. To convert the dataset to a usable form, the dataset images were processed using (4) prior to entering the edge detection algorithm. In this equation, $\mathbf{R}$ is the 2D matrix of red-channel data from the original image, $\mathbf{G}$ is the green-channel data, $\mathbf{B}$ is the blue-channel data, and $\mathbf{S}$ is the 2D output matrix representing the converted grayscale image [21]. The resulting grayscale image consists of pixels that have an intensity that is equivalent to the intensity of the pixels in the original image.

$$\mathbf{S} = 0.2125\mathbf{R} + 0.7154\mathbf{G} + 0.0721\mathbf{B} \tag{4}$$

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 11 Issue 3 March 2024

For the purposes of evaluation, a copy of the captured wind turbine dataset was processed using edge detection, and training and evaluation occurred on this new, static dataset. Fig. 3(c) and Fig. 3(d) sample images from the respective Healthy and Faulty image classes from the edge detection dataset. In a real-time wind turbine condition monitoring system, this edge detection algorithm could be added to the beginning of a CNN architecture as a new layer to perform edge detection preprocessing dynamically on every image that enters the model for classification.

E. Evaluation Procedure

To evaluate edge detection preprocessing for use in wind turbine health classification by CNNs, parameters within the network, known as hyperparameters, were first identified that could be tuned to increase the performance of each CNN. These included each respective CNN's batch size, loss function, optimizer type, activation function, and dropout ratio. The batch size represents the amount of data that is allowed to pass through the CNN during training before the model's weights are updated. Smaller batch sizes suffer from noise due to the model's weights being frequently updated; this noise can cause a CNN not to converge on optimal weights during training. In larger batch sizes, the effects of noise are minimized, but the model is more likely to be trapped with non-optimal weights.

The loss function computes the error margin between the model's output and the correct output during training. The optimizer is used to update the model weights during training to increase the model's accuracy by traversing the gradients output by the loss function. The activation function imposes non-linear behavior on the values passed between the layers of the architecture. The dropout ratio is used to prevent the CNN from overfitting during training. It is a measure of the percentage of nodes in the architecture that are deactivated during training, which forces the CNN to continuing identifying new patterns as it is trained. For all of these hyperparameters, proper values must be found that balance the trade-offs for each hyperparameter on the specific dataset used for training. This occurs during hyperparameter tuning. In this research, the KerasTuner software was used to identify and tune these hyperparameters [22].

After tuning the hyperparameters, the optimal values found are fixed for use during the training of the CNN model. This is where the model weights are updated and the CNN learns the relationships of the training data to enable successful inference on future, unseen data. The training was accomplished using the Keras software, which was also used to programmatically define the Xception and VGG19 architecture [23].

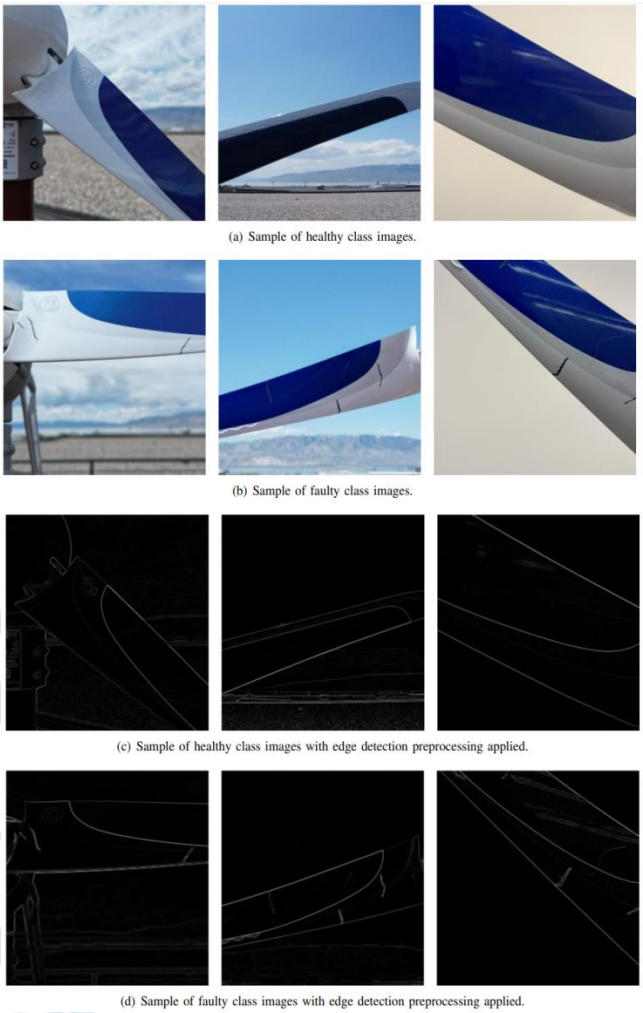


Fig. 3. Selected images from the wind turbine dataset (6,000 images) captured at Utah Valley University (UVU) using a small-scale wind turbine prototype. Samples are shown for both the original RGB dataset and the corresponding images modified by edge detection preprocessing.

To enhance the training and evaluation process, the wind turbine dataset was subdivided into three separate datasets: one used for training, one for testing, and one for validation. The training subset was used exclusively during model training, and the testing dataset was used to periodically evaluate progress during training on fresh data. The validation subset was used only for final evaluation after the training process was completed. This ensures the results are only based on inferences: they aren't biased with data already learned during training.

To evaluate the results, each network architecture was independently trained ten times using the hyperparameters identified during hyperparameter tuning. This allowed the average performance to be calculated, which more accurately represents the performance obtainable if these architectures were to be trained and implemented for an autonomous wind turbine condition monitoring system. After each independent training session, a confusion matrix was generated. This

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 11 Issue 3 March 2024

represents results in terms of true positive, true negative, false positive, and false negative. True positive inferences are where the model correctly classified an image as, in this case, a healthy blade. True negative inferences are where the model correctly classified an image as containing a faulty blade. A false positive inference means the model incorrectly classified a faulty blade as healthy, and a false negative means a healthy blade was incorrectly classified as faulty.

A variety of statistics can be derived from a confusion matrix. This research focuses on accuracy, precision, hit rate, miss rate, specificity, fall-out, and F1 score. Accuracy represents the ratio of correct inferences to the total number of inferences. Precision is the ratio of correct healthy inferences to the total number of healthy inferences while hit rate is the ratio of correct healthy inferences to the total number of healthy images. On the other hand, the miss-rate is the ratio of incorrect faulty inferences to the total number of healthy images. Additionally, specificity is the ratio of correct faulty inferences to the total number of faulty images, and fall-out is the ratio of incorrect healthy inferences to the total number of faulty images. F1 score is the harmonic mean of precision and hit rate. This means ideal models possess larger values of accuracy, precision, hit rate, specificity, and F1 score and lower values of miss rate and fall-out. All values are real numbers and range from 0 to 1. For each of the ten independent training sessions, these statistics were derived from the corresponding confusion matrix. The final statistics used for evaluation are the averages of these statistics from each of the ten training sessions.

The process of tuning hyperparameters, training the model, and deriving average results was repeated for each dataset – the original RGB dataset, the dataset processed with edge detection, and the portion of the edge detection dataset containing only images captured outdoors – and for each CNN architecture – Xception and VGG19. Results are presented in Section III.

III. RESULTS

Edge detection for wind turbine defect classification was evaluated through the average accuracy obtained by each selected CNN architecture when implementing edge detection preprocessing in comparison with the obtained average accuracy on the original RGB dataset unprocessed with edge detection. The average accuracies for the selected CNN architectures on the original RGB dataset are presented in Table II. The average accuracies obtained when edge detection preprocessing was implemented are given in Table III. Both tables are sorted by highest accuracy and include the average statistics described in Section II-E for reference.

When edge detection was applied to the portion of dataset images captured indoors, it became evident that glare from the indoor lighting was problematic. After edge detection has been applied, cracks are virtually indistinguishable from lighting reflections. This is evident in the right-most image of Fig. 3(d) where glare is present on the upper portion of the blade and cracks exist on the lower portion of the blade. To obtain an accurate estimate of edge detection's performance, it was also evaluated on a subset of the original dataset: the portion exclusively containing images captured outdoors. This removes the long, narrow reflections caused by indoor lighting that full-scale wind turbines would not possess. These results are presented in Table IV.

Table II. Average Wind Turbine Blade Defect Classification Performance Without Edge Detection Preprocessing Using Xception And Vgg19.

Algorithm	VGG19	Xception
Accuracy	0.9867	0.9792
Precision	0.9733	0.9709
Hit Rate	1.0000	0.9880
Miss Rate	0.0000	0.0120
Specificity	0.9740	0.9703
Fall-Out	0.0260	0.0297
F1 Score	0.9865	0.9794

Table III. Average Wind Turbine Blade Defect Classification Performance With Edge Detection Preprocessing Using Xception And Vgg19.

Algorithm	VGG19	Xception
Accuracy	0.9300	0.9345
Precision	0.8800	0.9477
Hit Rate	0.9778	0.9200
Miss Rate	0.0222	0.0800
Specificity	0.8909	0.9490
Fall-Out	0.1090	0.0510
F1 Score	0.9263	0.9335

Table IV. Average Wind Turbine Blade Defect Classification Performance With Edge Detection Preprocessing On Images Captured Exclusively Outdoors.

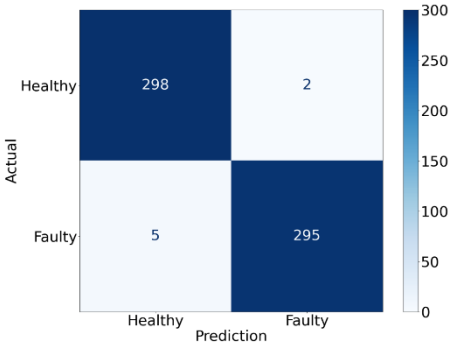
Algorithm	VGG19	Xception
Accuracy	0.8800	0.9575
Precision	0.7800	0.9767
Hit Rate	0.9750	0.9380
Miss Rate	0.0250	0.0620
Specificity	0.8167	0.9770
Fall-Out	0.1833	0.0230
F1 Score	0.8667	0.9565

ISSN (Online) 2394-2320

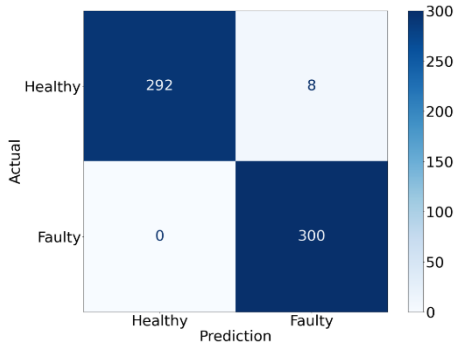
International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 11 Issue 3 March 2024

A visualization of the confusion matrices is also provided. These are the confusion matrices generated during the ten independent training sessions that possessed the highest accuracies `for each CNN architecture. Specifically, the top confusion matrices from the original RGB dataset are given in Fig 4, and the top confusion matrices from the dataset processed with edge detection are given in Fig. 5. The top confusion matrices from the exclusively outdoors portion of the edge detection dataset are provided in Fig 6.

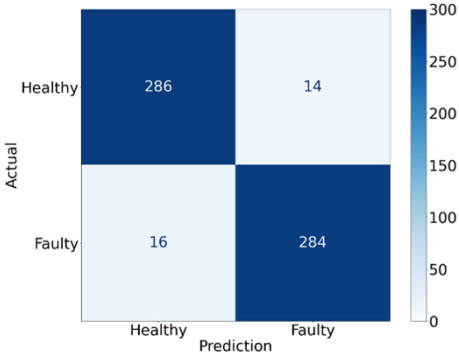


(a) Xception

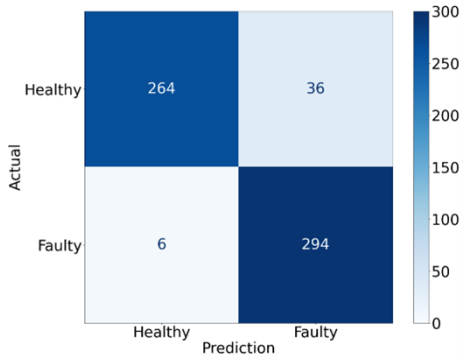


(b) VGG19

Fig. 4. Highest accuracy confusion matrices for wind turbine blade defect classification without edge detection preprocessing using Xception and VGG19.

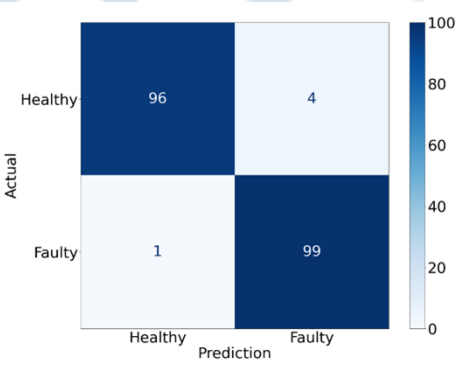


(a) Xception

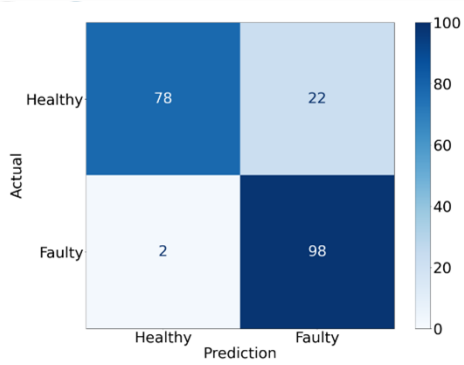


(b) VGG19

Fig. 5. Highest accuracy confusion matrices for wind turbine blade defect classification with edge detection preprocessing using Xception and VGG19.



(a) Xception



(b) VGG19

Fig. 6. Highest accuracy confusion matrices for wind turbine blade defect classification with edge detection preprocessing on images captured exclusively outdoors using Xception and VGG19.

IV. CONCLUSIONS

Edge detection preprocessing failed to improve the performance of wind turbine defect classification using convolutional neural networks (CNNs). In fact, it drastically decreased the performance. On the original RGB dataset, both CNN architectures performed in excess of 97% accuracy – VGG19 had the highest accuracy at 98.67% and Xception had slightly lower accuracy at 97.92%. When the

edge detection preprocessing algorithm was introduced, the accuracy for both architectures fell below 94%: Xception performed at 93.45% and VGG19 performed at 93.00%. This decrease in performance is also visible across the other averaged statistics.

In the portion of the wind turbine dataset processed with edge detection that exclusively contained images captured outdoors, which was intended to mitigate unnatural glare from indoor lighting interfering with the edge detection algorithm, a similar trend is visible. The Xception architecture, for instance, obtained an average accuracy of 95.75%. This is higher than the accuracy obtained on the edge detection dataset containing both indoor and outdoor images, which indicates the removal of the indoor glare did benefit the edge detection algorithm. However, this accuracy was still lower than the accuracy obtained on the original RGB dataset. The performance of the VGG19 architecture experienced a further decrease in performance: it obtained an average accuracy of 88.00%. The architecture being unable to converge on optimal weights due to the reduced size of the dataset resulting from the removal of images captured indoors is the likely cause of this decrease.

In summary, this research evaluated the efficacy of using edge detection implemented through the Roberts cross operator to increase the performance of an autonomous wind turbine condition monitoring system. To meet this goal, a dataset of 6,000 RGB images containing healthy and faulty wind turbine blades was created, to which edge detection preprocessing was applied. Two common CNN architectures, Xception and VGG19, were trained on both the original RGB dataset and the dataset processed with edge detection to compare their respective performance in each scenario. The performance of each architecture decreased when edge detection was introduced. These results indicate that edge detection implemented through the Roberts cross operator is not suitable for increasing the performance of an autonomous, CNN-based wind turbine condition monitoring system.

ACKNOWLEDGMENTS

This work is supported by the Office of the Commissioner of Utah System of Higher Education (USHE)-Deep Technology Initiative Grant 20210016UT.

REFERENCES

[1] M. Hutchinson and F. Zhao. GWEC — Global Wind Report 2023. Mar. 2023.

[2] H. Long et al. “Data-Driven Wind Turbine Power Generation Performance Monitoring”. In: IEEE Transactions on Industrial Electronics 62.10 (2015), pp. 6627–6635. DOI: 10.1109/TIE.2015.2447508.

[3] J. Ribrant and L.M. Bertling. “Survey of Failures in Wind Power Systems With Focus on Swedish Wind Power Plants During 1997–2005”. In: IEEE Transactions on Energy Conversion 22.1 (2007), pp. 167–173. DOI: 10.1109/TEC.2006.889614.

[4] N. Honjo. “Detail survey of wind turbine generator and electric facility damages by winter lightning”. In: The 31st Wind Energy Utilization Symposium. Vol. 35. 2013, pp. 296–299.

[5] B. Hahn, M. Durstewitz, and K. Rohrig. “Reliability of wind turbines”. In: Wind energy. 2007, pp. 329–332.

[6] W. Qiao and D. Lu. “A Survey on Wind Turbine Condition Monitoring and Fault Diagnosis—Part I: Components and Subsystems”. In: IEEE Transactions on Industrial Electronics 62.10 (2015), pp. 6536–6545. DOI: 10.1109/TIE.2015.2422112.

[7] B. Coffey. Taking Off: Nevada Drone Testing Brings Commercial UAVs Closer To Reality. https://www.ge.com/news/reports/taking-off-nevada-drone-testing-brings-commercial-uavs-closer-to-reality. 2019.

[8] GEV Wind Power. Wind Turbine Blade Inspection. https://www.gevwindpower.com/blade-inspection/. 2021.

[9] B. Pinney et al. “Exploration and Object Detection via Low-Cost Autonomous Drone”. In: 2023 Intermountain Engineering, Technology and Computing (IETC). 2023, pp. 49–54. DOI: 10.1109/IETC57902.2023.10152139.

[10] B. Pinney et al. “Drone Path Planning and Object Detection via QR Codes; A Surrogate Case Study for Wind Turbine Inspection”. In: 2022 Intermountain Engineering, Technology and Computing (IETC). 2022, pp. 1–6. DOI: 10.1109/IETC54973.2022.9796739.

[11] J. Miller et al. “Hyperparameter Tuning of Support Vector Machines for Wind Turbine Detection Using Drones”. In: 2023 Intermountain Engineering, Technology and Computing (IETC). 2023, pp. 55–60. DOI: 10.1109/IETC57902.2023.10152252.

[12] C. Seibi et al. “Locating and Extracting Wind Turbine Blade Cracks Using Haar-like Features and Clustering”. In: 2022 Intermountain Engineering, Technology and Computing (IETC). 2022, pp. 1–5. DOI: 10.1109/IETC54973.2022.9796823.

[13] L. N’Diaye et al. “Residual and Wavelet based Neural Network for the Fault Detection of Wind Turbine Blades”. In: 2022 Intermountain Engineering, Technology and Computing (IETC). 2022, pp. 1–5. DOI: 10 .1109/IETC54973.2022.9796852.

[14] K. O’Shea and R. Nash. An Introduction to Convolutional Neural Networks. 2015. DOI: 10.48550/ARXIV.1511.08458. URL: https://arxiv.org/abs/1511.08458.

[15] F. Chollet. “Xception: Deep Learning with Depthwise Separable Convolutions”. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017, pp. 1800–1807. DOI: 10.1109/CVPR.2017.195.

[16] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-scale Image Recognition”. In: 2015 International Conference on Learning Representations (ICLR). 2014. DOI: arXivpreprintarXiv:1409.1556820.

[17] D. Sarkar. “A comprehensive hands-on guide to transfer learning with real-world applications in deep learning”. In: Towards Data Science (2018).

[18] O.G. Yalcin. “4 Pre-Trained CNN Models to Use for Computer Vision with Transfer Learning”. In: Towards Data Science (2020).

[19] F. Chollet. Deep learning with Python. Simon and Schuster, 2021.

[20] A. Kaushik. Understanding the VGG19 architecture. https://iq.opengenus.org/vgg19-architecture/. 2020.

[21] S. van der Walt et al. “scikit-image: image processing in Python”. In: PeerJ 2 (June 2014), e453. ISSN: 2167-8359. DOI: 10.7717/peerj.453. URL: https://doi.org/10.7717/peerj.453.

[22] T. O’Malley et al. KerasTuner. https://github.com/keras-team/keras-tuner. 2019.

[23] F. Chollet et al. Keras. https://keras.io. 2015.