

Article

Leading Edge Erosion Classification in Offshore Wind Turbines Using Feature Extraction and Classical Machine Learning

Oscar Best ^{1,*}, Asiya Khan ¹, Sanjay Sharma ¹, Keri Collins ¹ and Mario Gianni ²¹ School of Engineering, Computing and Mathematics, Faculty of Science and Engineering, University of Plymouth, Plymouth PL4 8AA, UK² School of Electrical Engineering, Electronics and Computer Science, University of Liverpool, Liverpool L69 3BX, UK

* Correspondence: oscar.best@plymouth.ac.uk

Abstract: Leading edge (LE) erosion is a type of damage that inhibits the aerodynamic performance of a wind turbine, resulting in high operation and maintenance (O&M) costs. This paper makes use of a small dataset consisting of 50 images of LE erosion and healthy blades for feature extraction and the training of four types of classifiers, namely, support vector machine (SVM), random forest, K-nearest neighbour (KNN), and multi-layer perceptron (MLP). Six feature extraction methods were used with these classifiers to train 24 models. The dataset has also been used to train a convolutional neural network (CNN) model developed using Keras. The purpose of this work is to determine whether classical machine learning (ML) classifiers trained with extracted features can produce higher-accuracy results, train faster, and classify faster than deep learning (DL) models for the application of LE damage detection of wind turbine blades. The oriented fast and rotated brief (ORB)-trained SVM achieved an accuracy of $90\% \pm 0.01$, took 80.4 s to train, and achieved inference speeds of 63 frames per second (FPS), compared to the CNN model, which achieved an accuracy of $79.4\% \pm 2.07$, took 4667.4 s to train, and achieved an inference speed of 1.3 FPS. These results suggest that classical ML models can be more accurate and efficient than DL models if the appropriate feature extraction method is used.

Keywords: machine learning; damage detection; feature extraction; offshore devices



Citation: Best, O.; Khan, A.; Sharma, S.; Collins, K.; Gianni, M. Leading Edge Erosion Classification in Offshore Wind Turbines Using Feature Extraction and Classical Machine Learning. *Energies* **2024**, *17*, 5475. <https://doi.org/10.3390/en17215475>

Academic Editors: Manuel Pineda-Sanchez, Javier Martínez-Roman, Martín Riera-Guasp, Angel Sapena-Bano and Jordi Burriel-Valencia

Received: 23 September 2024
Revised: 22 October 2024
Accepted: 29 October 2024
Published: 1 November 2024



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

1. Introduction

Offshore wind turbines are susceptible to surface damage over time. The blades of the structure are particularly affected, and damage is worsened by exposure to extreme environmental conditions [1]. Such damages include delamination, which is the degradation of the resin between the laminate plies covering the blade that is mostly found along the tip of the blade. Weakened areas of delamination are exposed to further cracking due to the fatigue load on the wind turbine blade [2]. Splitting of the two blade halves along the trailing edge can result in longitudinal cracks where these halves are bonded. This type of damage is mostly found at the base and tip of the blade and is a result of reduced blade stiffness [3]. LE erosion is another type of damage caused by particulates, such as rain and hail, removing material from the blade's leading edge. This has become an important issue for the offshore industry, as LE erosion can negatively impact the aerodynamic efficiency of the turbine [4].

Operation and maintenance (O&M) costs for offshore farms account for almost a quarter of the total life cycle cost. If offshore wind turbines are damaged and need repair, the limited options are either to tow the device to port for repairs or deploy installation vessels offshore [5]. Both methods are costly and time-consuming and cannot be sustained with the growing number of offshore wind farms. Offshore wind farms are particularly costly compared to onshore wind farms. This is due to the limited accessibility caused by the vast

distance to shore and high water depths [6]. With the operation and maintenance (O&M) cost of these offshore renewable energy (ORE) devices being so high [7], it is important to facilitate the identification of LE erosion so that repairs can be made. Computer vision and deep learning have proven their worth by allowing for faster and safer inspections of surface damage on wind turbine blades, as opposed to more conventional practices with crewed teams [8]. Drones fitted with cameras are more able to retrieve data, which can be further analysed. CNNs are neural networks designed for image analysis [9], and they are an effective tool for many vision-based applications, already making significant impacts in the application of ORE device inspection. The issue with this approach is that these DL models require a significant amount of data to train them to be effective [10]. Secondly, they are computationally expensive [11].

In the case of Shihavuddin et al. [12], they were provided with a non-public inspection dataset from the private company EasyInspect ApS (Brøndby, Denmark), which contained 4000×3000 -pixel images of different types of wind turbine blade damage. With this, they trained faster region-based CNN (faster R-CNN) models with different backbones, such as InceptionV2, Inception-ResNet-V2, ResNet-50, and ResNet-101. For comparison, the dataset was used to train and evaluate single-shot detectors (SSD) and region-based fully convolutional networks (RFCN). Their results showed that faster R-CNN models outperformed in terms of accuracy when real-time processing was not needed, and they concluded that larger models generally resulted in higher mean average precision (mAP) scores. The largest model, Inception-ResNet-V2, achieved an accuracy of 81.1%, compared to 71.67%, 71.93%, and 72.86% for Inception-V2, ResNet-50, and ResNet-101, respectively. For their work, they chose four classes of surface damage, which include LE erosion, a vortex generator (VG) panel, a VG panel with missing teeth, and a lightning receptor. Their reasoning for choosing these was because the classes produce visual traits that are identifiable by humans rather than the significance of the damages' impact on the operation of the wind turbine.

Aird et al. [13] specifically describe the significance of LE erosion, stating that this damage type is responsible for decreasing blade performance and longevity, increasing maintenance costs, and causing reductions in annual energy production (AEP). They worked with a small dataset of 140 images containing deep and shallow LE erosion classes. These data were used to train and evaluate two ML models. The first was a supervised ML model that employed CNNs for feature extraction from an annotated training dataset. The second was an unsupervised ML model that aggregated pixel intensity thresholding through the calculation of pixel-by-pixel shadow ratio to identify the features within the image [14–16]. Both models were used to determine the percentage area of the blade that was damaged, with both models successfully identifying approximately 65% of the surface damage area. Deeper damage was easier to detect for both models; however, the supervised model proved more successful at identifying shallow damage types. The paper explains the choice of LE erosion as the focus damage type because it is an important contributing factor to the wind turbine lifespan. In addition, the use of machine learning, both supervised and unsupervised, was explored to identify two different instances of LE erosion.

Similarly, Deng et al. [17] have used machine learning to classify blade surface damage. They used a dataset of 1200 images, with each of the four classes containing 300 images. The four damage types were nick, crack, blister, and spot. The authors used an improved particle swarm optimisation (PSO) method to allow for adaptive image filtering before applying the Log-Gabor filter to extract edge features [18,19]. The resulting features were used to train an SVM that would discriminate among the four classes. The SVM trained with features extracted without PSO achieved a mean accuracy of 87%, whereas with PSO, the trained SVM achieved a mean accuracy of 94%. The paper highlights the importance of image preprocessing prior to feature extraction because the quality of the features is affected by the quality of the image.

Yang et al. [20] have also pre-processed images prior to feature extraction. Their work is more concerned with the issue of visual noise generated by the background of the image

as opposed to the clarity of the image. They used a maximum inter-class variance threshold segmentation method known as the Otsu method to eliminate the background in the image, leaving only the blade [21]. The binary image containing just the blade is superimposed onto the original image, in which all background pixels are set to black. This method allows just the blade to contain noticeable features.

Feature extraction using the histogram of oriented gradients (HoG) approach was used by Wang and Zou [22]. This method extracted texture features from 507 images containing crack and non-crack damage classes. The dataset was split 70:30 for training and testing, and extracted features were used to train an SVM classifier. The authors reported a training accuracy of 89.02% and a test accuracy of 33.99%. The paper compares the results of the trained SVM to a CNN trained on the same dataset. The DL model achieved a training accuracy of 96%; however, there was no report for testing accuracy to make a true comparison between the two methods. The paper further mentions how larger datasets are needed for training DL models compared to ML models.

These papers have explored limited techniques for feature extraction, achieving varying results. Because of this, it is difficult to determine whether alternative feature extraction methods would have been more appropriate for their application. Model efficiency is an issue that has been seemingly overlooked and should be seriously considered for practical implementations. Insufficient data for training deep learning models is also an issue echoed by multiple papers. The purpose of this work is twofold. The first aim is to identify which feature extraction method and classical ML classifier produces a model with a high classification accuracy, short training time, and low inference latency using a small dataset of 50 images given in [23]. The second aim is to compare the accuracy and efficiency of this ML model to a CNN model to determine whether it is more beneficial to use classical ML approaches instead of deep learning for the application of wind turbine LE erosion classification. DL models can be computationally expensive and require a significant amount of data to be useful; therefore, ML classifiers have been used as an alternative approach and have been provided with features that have been manually extracted using different feature extraction techniques. Using a limited dataset, the classifiers will discriminate between healthy blades and blades with LE erosion.

The structure of this paper is as follows: Section 2 explains the methods used for this work. The first is how image preprocessing has been applied to the small dataset. The second is which feature extraction methods have been used to collect training features. The third is which classical ML models have been used for classification, and finally, the parameters and conditions for training and inference are described for all classical ML models and a CNN model. Section 3 lists the accuracy results, training times, and inference performance of each classical ML model and CNN model. The robustness of the ORB-SVM model is tested on various noise-induced images. A comparison is made between the ORB-SVM algorithm and current existing algorithms. Lastly, the discussion and conclusion are contained within Section 4.

2. Materials and Methods

2.1. Data Preprocessing

A small dataset [23] containing 50 images, 25 images of wind turbine blades with LE erosion and 25 healthy blades, was created from selected images from online sources [24–30]. All images were first converted to grayscale to reduce the image dimension from 3 channels to 1, and they were resized to 160×160 . TensorFlow's (version 2.11.0) ImageData-Generator class has been used to expand the dataset from a total of 50 images to 500 images during runtime, with an equal number of class samples. This is important because too few images may appear similar and not fully represent the class to which they belong. By applying image augmentations, the intra-class variance is increased, which allows for a greater generalisation of the classification model. The types of augmentation that occur include width and height shifts of 10% of the original image size (after resizing), rotations of 0–45 degrees, vertical and horizontal flipping, channel shifts from 0 to 20% of the original pixel values,

zoom ranging from 0 to 150% of the original image, and shear ranging from 0 to 20 degrees. Figure 1 shows a selection of both LE erosion and healthy blade samples included in the dataset after augmentation.

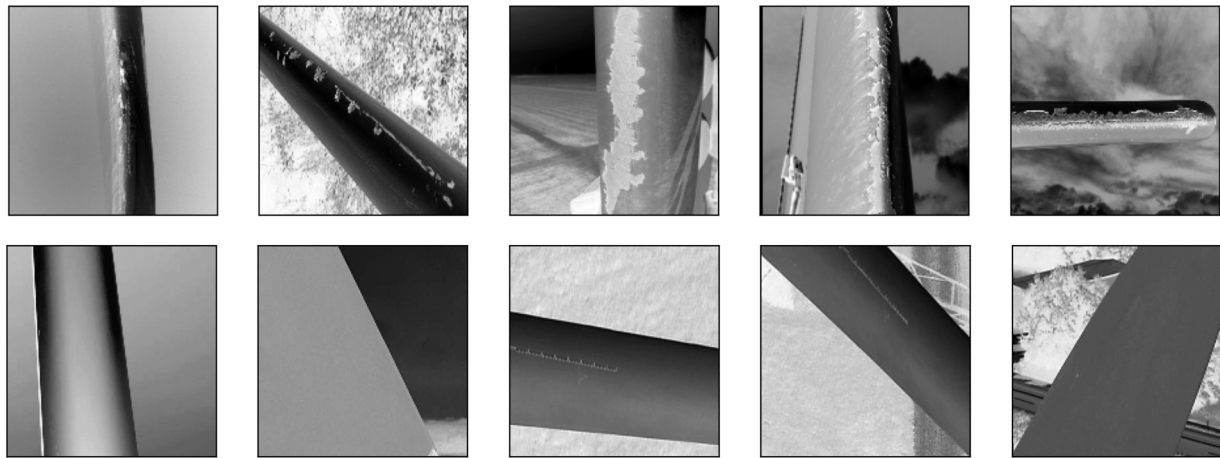


Figure 1. Dataset samples of LE erosion (**top row**) and healthy blades (**bottom row**).

2.2. Feature Extraction Methods

A selection of feature extraction techniques was used for this dataset to process the images and reduce the dimensionality of the data. The chosen feature extraction methods are well-known and have been used for image processing in numerous applications, particularly for damage detection.

2.2.1. Two-Dimensional Discrete Wavelet Transform (2D-DWT)

Wavelet analysis is used to divide signal information within an image into two components, which are approximation and detail coefficients. The image is passed through two filters, a high-pass and a low-pass filter. The 2D discrete wavelet transform splits an image into four coefficients, namely, cA, cH, cV, and cD. The decomposition creates an approximation of the original image, represented as cA, which is the result of the low-pass filter and is equal to the original image without the remaining coefficients (cH, cV, and cD) [31]. These remaining coefficients represent the detail of the original image and are the result of the high-pass filter, with each containing different details: horizontal details (cH), vertical details (cV), and diagonal details (cD). The approximation can be further decomposed into another four coefficients in the same way [32]. This technique can continue through multiple levels; however, for this paper, a 3-level decomposition was chosen. The goal was to extract sparse coefficients containing only important details originating from the original input image, much like the feature maps obtained from convolutional layers in a CNN. Figure 2 shows the three level 3 detail coefficients extracted from the original image.

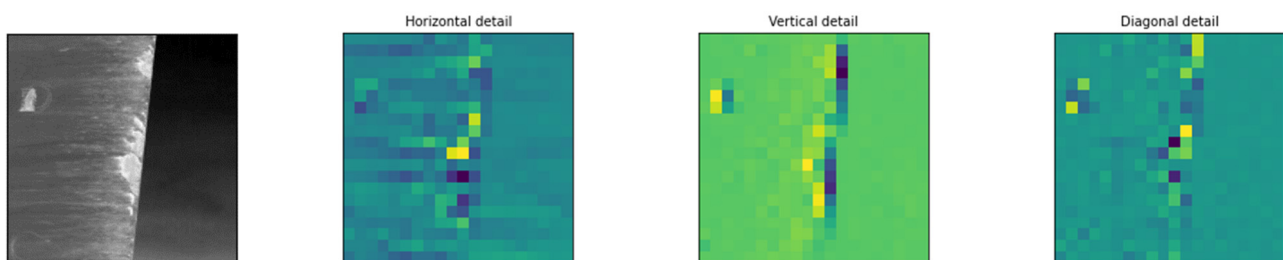


Figure 2. Original image (**left**) and level 3 decomposition detail coefficients (**right**).

2.2.2. Discrete Fourier Transform (DFT)

Discrete Fourier transform was used to decompose each image into its sine and cosine components. The spatial information was transformed into the frequency domain at which each point in the transform magnitude represents a certain frequency contained within the spatial image. Only a set sample of frequencies is contained within the image, which is enough to describe the spatial image and its important features whilst removing unnecessary frequencies. The DFT of a spatial image f with size $M \times N$ can be represented as an image F of the same size, as defined by the following equation:

$$F(u, v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) e^{-j2\pi(\frac{um}{M} + \frac{vn}{N})} \quad (1)$$

Figure 3 shows the magnitude of the DFT of the original image. Low-frequency signals are concentrated around the centre of the image and represent areas in which pixel intensities slowly evolve from one pixel to the other. High-frequency signals are represented near the boundary of the image and correspond to pixels within the image with sharp-intensity transitions. Due to the shape and intensity of the low-frequency signal near the horizontal axis, we can deduce that the image contains a distinct contrast in pixel values along a vertical area of the image. This most likely represents the difference in pixel intensity between the leading edge and the background.

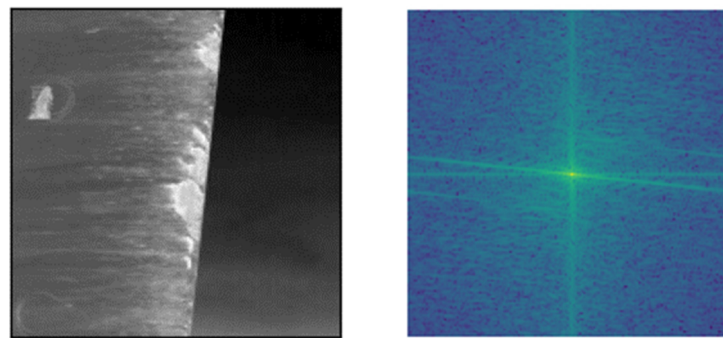


Figure 3. Original image (left) and discrete Fourier transform amplitude (right).

2.2.3. Laplacian of Gaussian (LoG)

To extract features using the Laplace transform, each image is passed through a Gaussian filter to remove noise. A Laplacian filter is then used to detect edges by computing the second derivatives of each image. This technique determines whether a change in adjacent pixel value is from an edge or not. The resulting extracted features contain key points, which represent regions with sudden changes in pixel intensity. The Laplacian $L(x, y)$ of an image with pixel intensity values $I(x, y)$ are given by the following equation [33]:

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \quad (2)$$

When combined with a Gaussian smoothing filter, the 2D $LoG(x, y)$ of an image can be represented as the following equation, with Gaussian standard deviation σ centred on zero:

$$LoG(x, y) = -\frac{1}{\pi\sigma^4} \left[1 - \frac{x^2 + y^2}{2\sigma^2} \right] e^{-\frac{x^2 + y^2}{2\sigma^2}} \quad (3)$$

Figure 4 shows the original image with the applied LoG filter. The edges from erosion are highlighted by this feature extraction method, which are fine details represented in the original image.

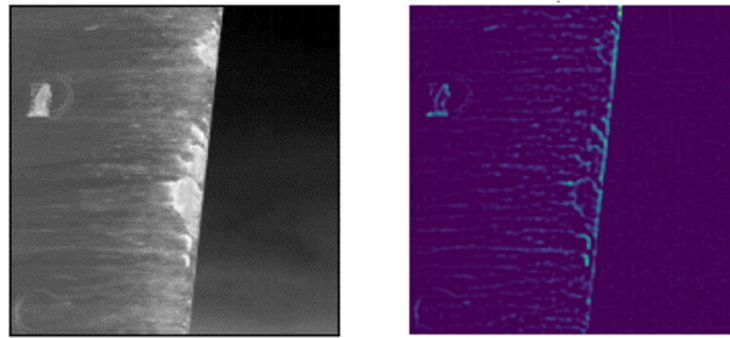


Figure 4. Original image (left) and LoG transform (right).

2.2.4. Histogram of Oriented Gradients (HoG)

For this method, the gradient is calculated for the pixel intensities in both the vertical G_y and horizontal G_x directions of the image [34]. These two feature maps are used to calculate the gradient magnitude G and direction θ using the following equations:

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \quad (4)$$

Each image is then split into cells of size 8×8 , whereby the histogram of the gradient is calculated for each cell. The sum of the magnitudes with corresponding gradient angles is represented by 9 bins before concatenating all cells into a single array for normalisation using the L2 Norm. HoG features are finally collected from all 2×2 blocks and concatenated to form a final feature vector representing the entire image. Figure 5 shows the HoG descriptor for the original image. The 400 cells contain both magnitude and direction for each gradient. Intense magnitudes along the leading edge show the contrast between blade and background pixel intensities, whereas the varying directions of the middle cells suggest a disturbance of the leading edge surface.

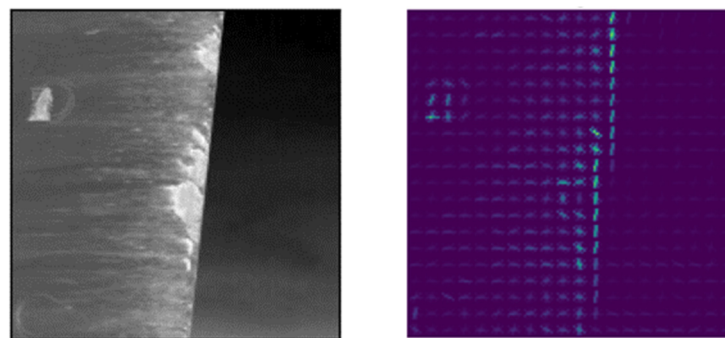


Figure 5. Original image (left) and HoG descriptor (right).

2.2.5. Oriented FAST and Rotated BRIEF (ORB)

ORB is a feature extraction technique developed at OpenCV Labs by Rublee E. et al. It was created as a viable opponent to SURF and SIFT because these are patented algorithms. ORB builds on the features from accelerated segment test (FAST) key point detector and binary robust independent elementary feature (BRIEF) descriptor algorithms. First, FAST with a multiscale image pyramid is applied to the image to identify scale-invariant key points. A rotation-aware BRIEF (rBRIEF) method is then used to group all identified key points and convert them into a binary feature vector [35]. Figure 6 shows the detected key points of the original image. Many of the key points chosen by the algorithm are heavily concentrated around the leading edge of the blade. This area of interest is likely caused by surface erosion differing from the appearance of the remaining blade surface.

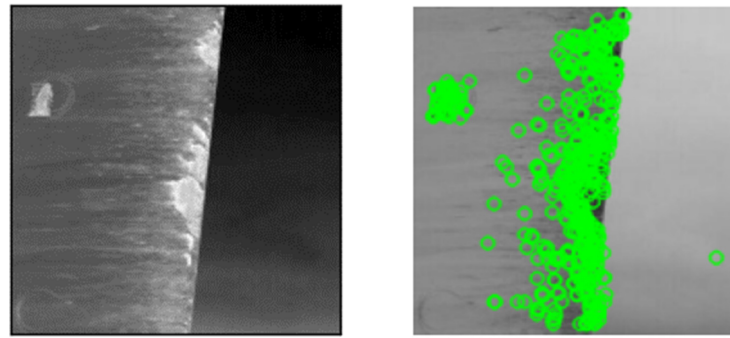


Figure 6. Original image (left) and ORB key points (right).

2.2.6. Canny Edge

An adaptive canny edge feature extraction method was used to automatically select appropriate upper and lower pixel thresholds. The first step of this process is to smooth the image using a Gaussian filter, which will help to remove noise from the image. The second step involves a pair of convolution masks applied to the image to detect vertical and horizontal details. The gradient magnitude and direction are then calculated using the same equation used for HoG [36]. Finally, adaptive thresholding was used to determine appropriate upper and lower thresholds for hysteresis thresholding, and the median pixel intensity was calculated for each image. The thresholds were then determined around this median value and controlled by a fixed variable σ . These thresholds are shown by the following equations, where M is the median pixel intensity of the image and σ is a fixed variable:

$$\begin{aligned} \text{Upper Threshold} &= (1 + \sigma)M \\ \text{Lower Threshold} &= (1 - \sigma)M \end{aligned} \quad (5)$$

Figure 7 shows the original image after the adaptive thresholding canny edge has been applied. The thresholding has successfully retained detail of interest, showing detected edges of the LE erosion.

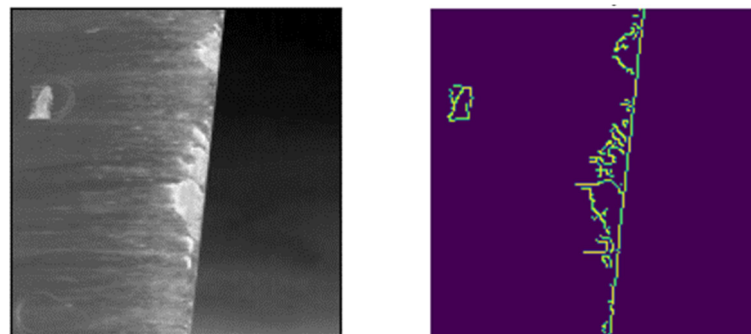


Figure 7. Original image (left) and canny edge transform (right).

2.3. Classical Machine Learning Methods

2.3.1. Support Vector Machine (SVM)

The support vector machine is currently one of the most widely used supervised ML models. This algorithm can be used for both regression and classification problems. For this paper, it is used for image classification. The SVM works by finding the hyperplane that separates the different classes in the feature space. The idea is to maximise the distance between the closest points of the different classes so that input data can be better distinguished. These feature points closest to the hyperplane are known as support vectors because without these points the hyperplane would shift [37]. Figure 8 shows an example of an SVM using a linear kernel. This type of SVM is useful for classifying between two classes; however, there are also polynomials, radial basis functions (RBFs), and sigmoid kernels, which can be used to solve more complex non-linear problems.

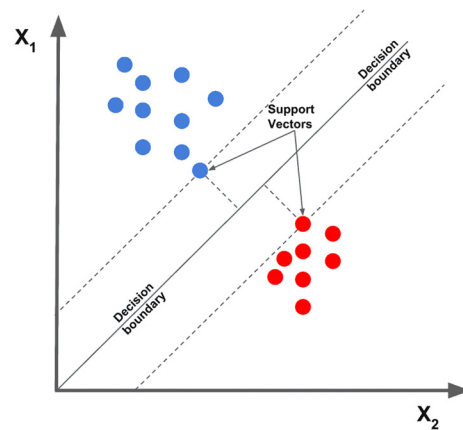


Figure 8. Decision boundary determined by maximising support vector distance. Red and blue points represent different classes [38].

2.3.2. Random Forest

The random forest algorithm is a machine learning technique that creates numerous decision trees during training. The decision tree is a supervised learning branched model that consists of a hierarchy of decision nodes. Each node splits the data based on a specific test of an attribute, creating branches to further nodes until a decision leaf is reached. This decides the resulting label. Training a decision tree consists of a greedy selection of the best splits, which minimises a cost function. With random forest, each tree is trained on a random subset of the input features. For classification, the majority class identified by all decision trees is represented as the final output class [37]. By forcing a subset of data to each decision tree and analysing collective decisions, the random forest algorithm becomes more robust to overfitting. Figure 9 shows the general structure of a random forest model.

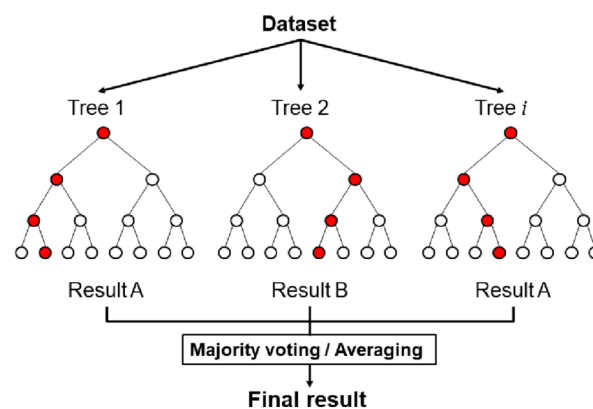


Figure 9. Random forest structure. Red circles show decision path for each tree. Result is determined by a majority voting [39].

2.3.3. K-Nearest Neighbour (KNN)

KNN is a supervised learning classification model that uses proximity to assign class labels to feature points. It works by taking a new feature point and calculating the distance of this point to all prior classified points. The value of K represents the number of nearest neighbouring points that are included when deciding the majority class of surrounding points. The new feature point is then assigned to this class. For image classification, the majority class assigned to the total feature points of an image decides the overall class of the input image [40]. Figure 10 shows an example of KNN with $K = 3$, where two of the surrounding three neighbours belong to class B, which results in the new feature point adopting the same label.

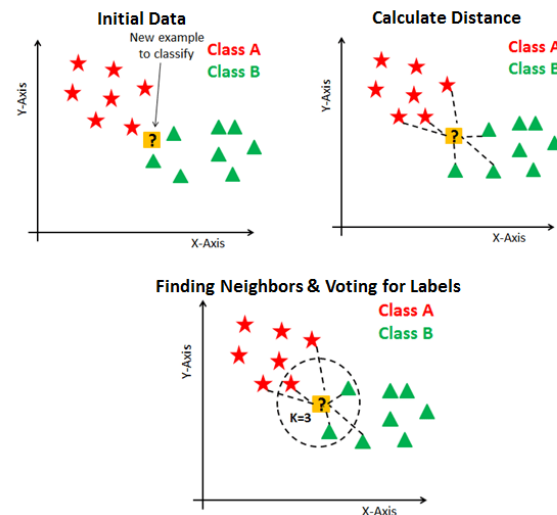


Figure 10. Stages of KNN. A new feature point adopted by the majority class of its nearest neighbour, determined by the value of K [41].

2.3.4. Multi-Layer Perceptron (MLP)

The multi-layered perceptron is a type of artificial neural network (ANN) that consists of an input layer, an output decision layer, and a variable number of fully connected hidden layers that connect the input to the output [42]. Each hidden layer is made up of neurons, with an associated weight that determines its connection strength to the proceeding layers. As training commences, input features pass through the network, in which an output is generated by the last layer. The predicted output from the MLP is measured against the true output using a cost function to produce a network error, which is the difference between the two values. Backpropagation is the process whereby this error is propagated back through the network to tune the neuron weights and minimise the error value. As training continues, this process is repeated until the error can no longer be further reduced, thus producing a predicted output close to that of the true output. Figure 11 shows the structure of a simple MLP with a single hidden layer.

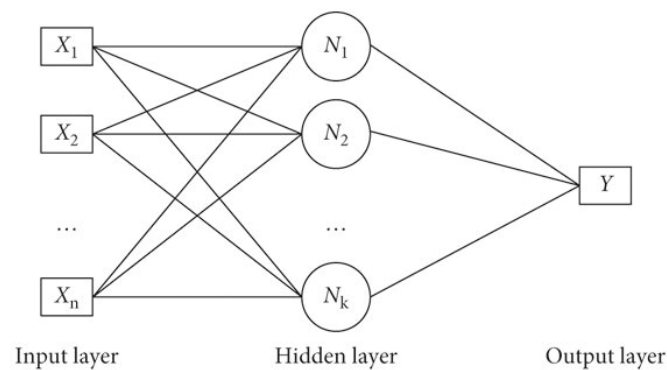


Figure 11. Schematic of an MLP with a single hidden layer [43].

2.4. Training and Inference

2.4.1. Machine Learning Models

Training for all models was performed using all 8 cores of an Intel(R) Core(TM) i7-7820X CPU @ 3.60 GHz, 3600 Mhz, manufactured by Intel and sourced from Hsinchu, Taiwan. The data was first pre-processed by resizing and grey-scaling to produce single-channel $160 \times 160 \times 1$ images before data augmentation was used to expand the dataset. Feature extraction for each method took place after data pre-processing. The extracted features were used as input to all four classifiers. Nested Kfold cross-validation (CV) was used for training each classifier using 4 folds for both inner and outer CV. Outer CV would

split the dataset into 4 folds, consisting of 3 training and 1 testing fold. The 3 training folds were further split into 4 folds in the same manner using inner CV. Here, Halving GridSearchCV was used across the inner folds to determine the optimal hyperparameters for the first outer fold. This process took place for each of the outer folds. The mean and standard deviation for all outer fold accuracies have been measured per model. The accuracy can be determined using the following formula, where true positive (TP) is the correct detection of the positive class; true negative (TN) is the correct detection of the negative class; false positive (FP) is the incorrect detection of the positive class; and false negative (FN) is the incorrect detection of the negative class:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (6)$$

The training time of the nested Kfold CV for each model was recorded in seconds. The inference FPS was measured using a Jetson Nano 2gb developer board, manufactured by NVIDIA and sourced from Shenzhen, China.

2.4.2. CNN Model

The dataset was further used to train a convolutional neural network so that a comparison could be made between deep learning and classical machine learning models. Figure 12 shows the structure of a simple convolution neural network implementation using Keras. The model is composed of five convolutional layers using the activation function ReLU, with a 3×3 kernel, zero padding, and a stride of 1. Each convolutional layer outputs double the number of input channels, whilst the output feature map dimensions are affected by the following formulas, where W and H are the width and height of the feature map, respectively, K is the kernel size, S is the stride, and P is the padding:

$$\begin{aligned} W_{out} &= \frac{W_{in} - K + 2P}{S} + 1 \\ H_{out} &= \frac{H_{in} - K + 2P}{S} + 1 \end{aligned} \quad (7)$$

The input size for the network is $160 \times 160 \times 1$, which allows for the greyscale images in the dataset to be used. Two stacked convolutional layers follow the input layer before a maximum pooling layer is applied. This is to allow for a richer feature extraction before the spatial information is reduced further into the network [44]. The maximum pooling layers in the network have a pool size of 2×2 , which reduces the input feature dimensions by half. Finally, the output feature maps are flattened and passed through a fully connected layer consisting of 256 units, followed by a 25% dropout layer, which temporarily removes 25% of the previous nodes during both forward and backward propagation. This method helps avoid model overfitting by creating a new network architecture from the parent network [45]. Finally, an output layer containing a single unit with a sigmoid activation function is used for the binary classification. The fully connected layers in the model are responsible for classifying the inputs and tuning the model's weights.

For training the model, an Adam optimiser with a learning rate of 0.00002 and a binary cross entropy loss function were optimised parameters predetermined by a hyperband parameter tuner used on each fold of 4-fold cross-validation. A batch size of 32 was used to train for 100 epochs per fold. The mean and standard deviation for all fold accuracies were measured. After cross-validation, the final model was trained on 80% of the entire dataset, leaving 20% for validation using the same predetermined parameters. The training time for the 4-fold cross-validation was measured in seconds. Inference was measured on a Jetson Nano 2gb developer board where the trained model was deployed.

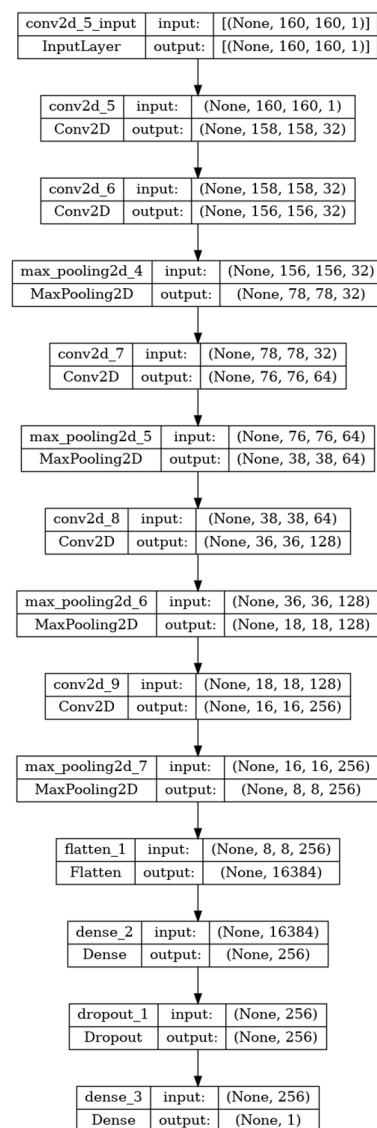


Figure 12. CNN model architecture.

3. Results

3.1. Machine Learning Results

Table 1 shows the mean and standard deviation of the nested fourfold cross-validation accuracies obtained for each classification model trained with different extracted features.

Table 1. Accuracy (%), mean, and standard deviation across all outer folds of each model.

Feature Extraction	Canny Edge	DWT	DFT	Laplace	HoG	ORB
MLP	69.2 ± 0.06	50.8 ± 0.02	59 ± 0.05	74.8 ± 0.03	62.4 ± 0.03	75.6 ± 0.08
SVM	72.6 ± 0.06	72.6 ± 0.03	83.8 ± 0.01	72.8 ± 0.006	67 ± 0.06	90 ± 0.01
Random Forest	68.6 ± 0.03	76.4 ± 0.03	81.2 ± 0.05	70 ± 0.03	70 ± 0.03	74.6 ± 0.05
KNN	50 ± 0.004	50.2 ± 0.007	78.2 ± 0.04	53.6 ± 0.01	69.4 ± 0.04	78 ± 0.02

The SVM trained with ORB features obtained the highest accuracy of 90%, with a low standard deviation of 0.01. The best parameters were found to be the polynomial kernel, $C = 0.1$, and $\gamma = 0.001$.

The receiver operation characteristic (ROC) is a two-dimensional measure of classification performance [46]. The curve plots the probability of correctly classifying the positive

class compared to the rate of falsely identifying the positive class at varying thresholds. The area under the curve (AUC) value is a key measure obtained from the curve that represents how well the classifier distinguishes between classes. Figure 13 shows the ROC curves for each fold of the ORB-SVM, which gave the best mean accuracy of 90%. Here we can observe that the difference between the curves is marginal, which is representative of a robust model. The red dotted line equates to a 50% chance of positive classification. The mean curve is close to the vertical axis, which is further above this line and means that the model has a high chance of positive classification. This can also be deduced by the high average AUC value. The ROC analysis shows that the model successfully identifies more of the positive class with fewer false positives.

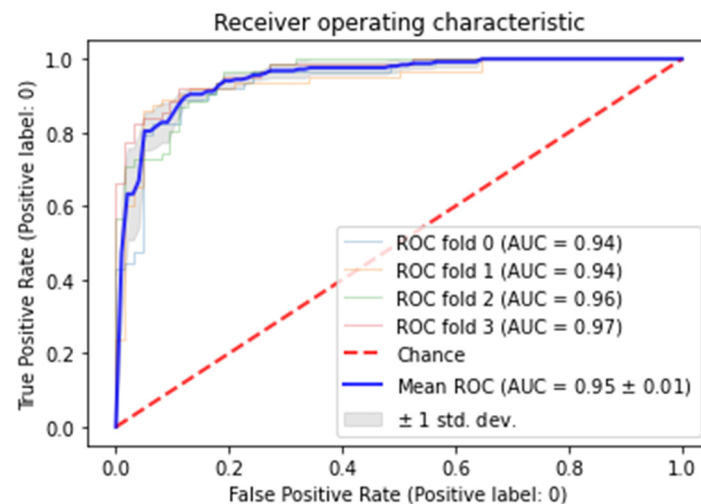


Figure 13. ROC curve of 4-fold cross-validation of the ORB-SVM.

Table 2 shows the time taken for each model to complete training using fourfold cross-validation. In most cases, the random forest classifiers took less time than the remaining three types of classifiers, whereas the MLP models took considerably longer to train. DWT produced the least extracted features, which is why the training times for each model are much smaller than others. There is a correlation between training time and the number of extracted features used to train the model.

Table 2. Training times in seconds for each model.

Feature Extraction	Canny Edge	DWT	DFT	Laplace	HoG	ORB
MLP	865.2	36.3	672.1	978.4	470.3	540.8
SVM	139.2	8.7	125.2	142.4	70.0	80.4
Random Forest	40.0	38.5	63.3	41.1	47.8	49.6
KNN	184.4	12.7	228	130.2	106.5	136.4

The inference results displayed in Table 3 show that the MLP models classify images faster than the remaining machine learning models, with most KNN models producing the lowest FPS. Again, ML models trained using DWT features classify significantly faster due to the small number of extracted features.

Table 3. Inference times of models represented as frames per second (FPS).

Feature Extraction	Canny Edge	DWT	DFT	Laplace	HoG	ORB
MLP	59	526	135	91	77	182
SVM	17	303	37	13	63	63
Random Forest	12	13	43	16	13	45
KNN	0.83	100	6	1.6	4	3

To ensure the ORB-SVM algorithm showed robustness to image interference, various types of noise were added to a test image of LE erosion. The types of noise reflect conditions that may be faced by a drone whilst collecting images such as brightness, cloud cover, and wind turbulence. Figure 14 shows the test image with different applied noise. To simulate brightness, all pixels were increased by the value of 100. Similarly, to simulate cloud cover, all pixels were decreased by 100 to create a ‘dimming’ effect. Vertical and horizontal motion blur was added to simulate heavy turbulence caused by winds. Each image was classified using the ORB-SVM algorithm, which provided a class label and confidence value. Despite the simulated interference, the algorithm correctly identified that the image contained LE erosion with minimal change in confidence. Vertical motion blur has decreased the model’s prediction confidence the most. This can be explained by the reduction in pixel intensity changes within the image, resulting in fewer FAST key points detected by the ORB algorithm. Overall, the ORB-SVM algorithm has shown robustness to image external interference and proves a level of reliability for practical use in an industrial setting.

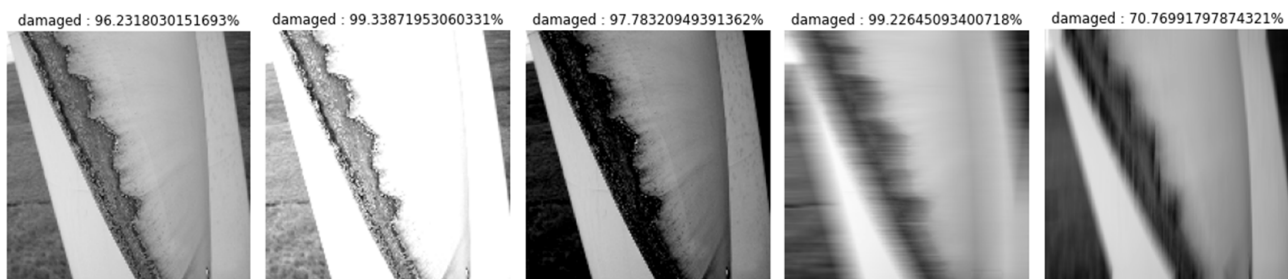


Figure 14. Test image with various noise. From left to right: original image, brightened, dimmed, horizontal blur, and vertical blur.

3.2. CNN Results

After 100 epochs per fold, the mean accuracy attained by the model was $79.4\% \pm 2.07$, with an average validation loss of 0.44 in a total of 4667.4 s. The results for validation loss and validation accuracy per fold are represented in Figure 15. The accuracy of the model is likely limited by the size of the dataset; however, the variation between folds is visible but not vast, which suggests that the dataset is consistent, and the model is not dependent on specific train-split configurations to achieve a higher accuracy.

Table 4 shows the comparison between the CNN and ORB-SVM in terms of accuracies and speeds for both training and inference. From Table 4, ORB-SVM achieved an accuracy 10.6% higher than that of the CNN model, with a lower cross-validation variance, suggesting a more efficient generalisation of the training data. Using the system CPU, the CNN took longer to train at 4667.4 s compared to the ORB-SVM at 327.5 s. For inference on the embedded system, ORB-SVM achieved an FPS rate of 63, compared to 1.3 from the CNN model. From this comparison, the ORB-SVM algorithm would be more suitable for deployment on an embedded system.

Table 4. Comparison between ORB-SVM and CNN model.

Algorithm	Accuracy (%)	Training Time (s)	Inference (FPS)
ORB-SVM	90 ± 0.01	327.5	63
CNN	$79.4\% \pm 2.07$	4667.4	1.3

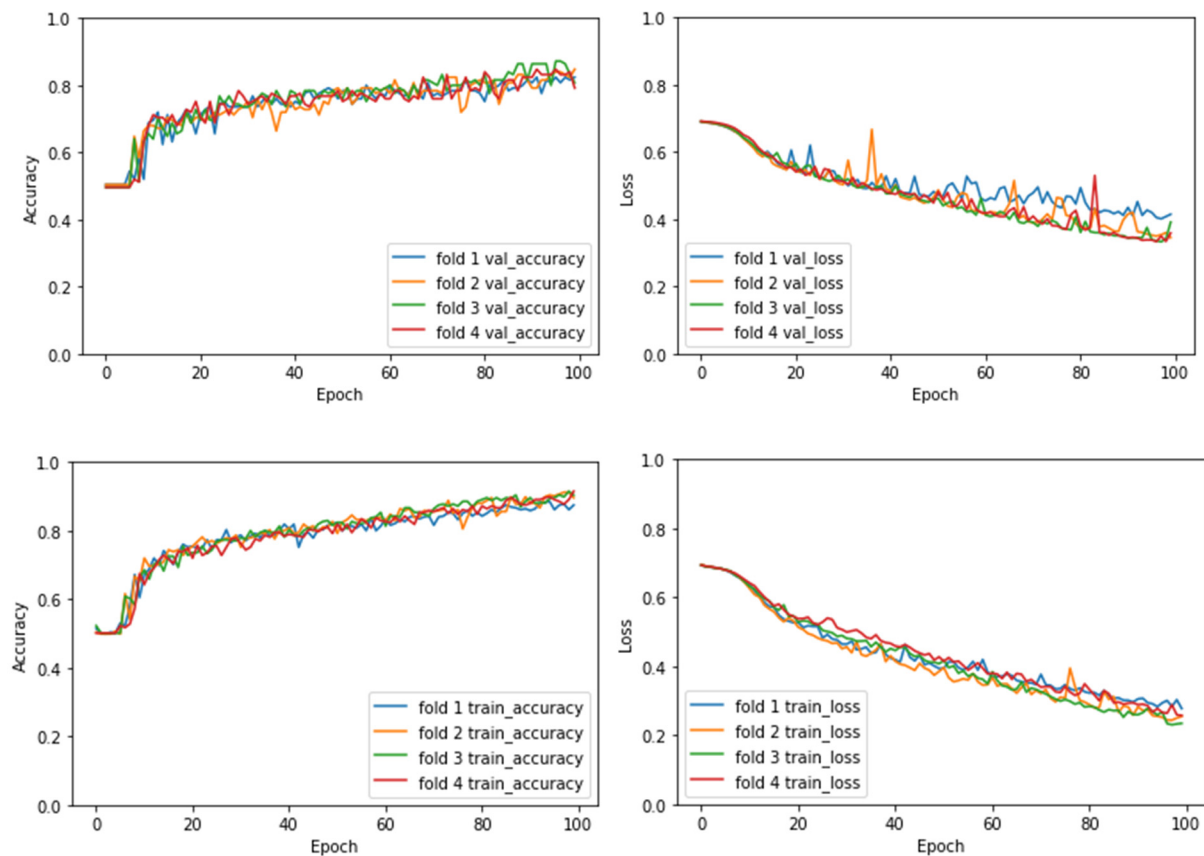


Figure 15. CNN validation, training accuracy, and loss graphs for 4-fold cross-validation.

3.3. Comparison Experiments

To demonstrate the effectiveness of the ORB-SVM algorithm, a comparison was made with a selection of advanced lightweight algorithms, namely, EfficientNetB0, MobileNetV2, and DenseNet121. For this comparison, computations were made using an AMD Ryzen 9 CPU, NVIDIA Geforce 3050 Ti laptop, manufactured by ASUS and sourced from Suzhou, China. All four algorithms used the same number of images (500) with the same augmentations. Fourfold cross-validation accuracies, training times, and inference FPS can be found in Table 5.

Table 5. Comparison between ORB-SVM, EfficientNetB0, MobileNetV2, and DenseNet121.

Algorithm	Accuracy (%)	Training Time (s)	Inference (FPS)
ORB-SVM	90 ± 0.01	327.5	20,134.2
EfficientNetB0	78.4 ± 0.8	1051	24.8
MobileNetV2	85.2 ± 5.32	560.6	29.8
DenseNet121	91 ± 0.35	1283	29.9

In Table 5, ORB-SVM shows an accuracy that is 11.6% higher than that of EfficientNetB0 and 4.8% higher than that of MobileNetV2, with a lower cross-validation variance, which suggests the model generalises more efficiently to the training data. DenseNet121 showed an increase of 1% to that of the ORB-SVM algorithm but showed significantly less inference FPS in comparison. ORB-SVM shows an improvement in inference speed at a minimal cost to classification accuracy. Figure 16 shows the memory consumption during training and inference for ORB-SVM (green), EfficientNetB0 (blue), MobileNetV2 (red), and DenseNet121 (black).

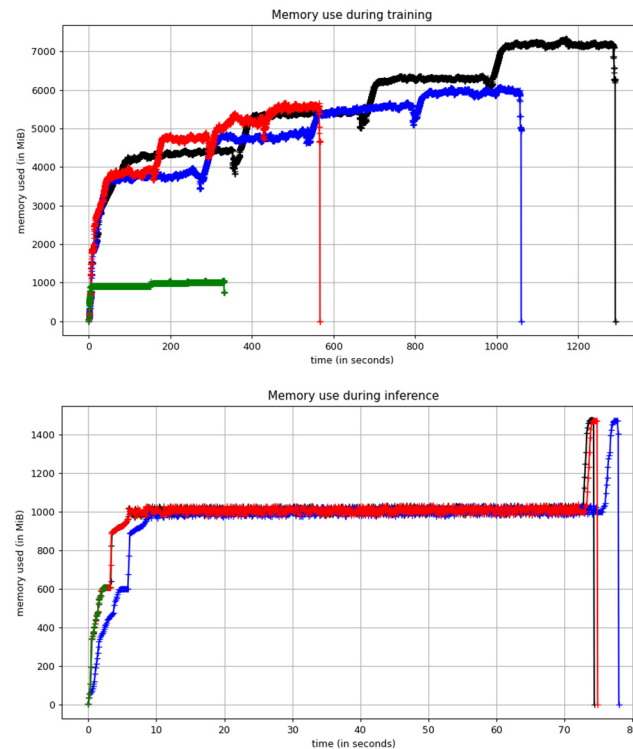


Figure 16. Training and inference memory consumption of ORB-SVM, EfficientNetB0, MobileNetV2, and DenseNet121.

In Figure 16, ORB-SVM shows a fraction of the memory consumption when compared to the other algorithms during training. It consumed approximately 1000 MiB, compared to DenseNet121, which consumed the highest amount of over 7000 MiB. For inference, ORB-SVM consumed 600 MiB compared to the remaining three algorithms, which all consumed approximately 1500 MiB. The significant reduction in memory consumption suggests that ORB-SVM is more efficient than the remaining models.

4. Discussion

In this study, a small dataset was created to classify LE erosion. The dataset, consisting of 50 images, was small and sought to address the challenges associated with insufficient data. Four types of ML classifiers were selected, including MLP, SVM, random forest, and KNN, and trained on various feature extraction methods to produce 24 models. A CNN model was also trained on the dataset for comparison. This was to identify how each classifier would perform given different features as input.

From the results in Table 1, the model with the highest accuracy was the ORB-trained SVM, which achieved 90% accuracy. This suggests that reducing the dimensionality of the image to only key features can prove beneficial to classification performance. This result suggests that the model has a small percentage of incorrect predictions, with most predictions correctly detecting instances of LE erosion and no LE erosion. The reliability of correct classifications is crucial for operational decisions. Alerting to damage on the blade when there is none could waste time if the decision is made to act on the incorrect classification. Similarly, undetected damage that is left unattended may cause further damage to the structure of the blade in the future and increase maintenance costs. Both scenarios demonstrate why minimising the FNs and FPs of the classification model is important. The selection of both the classifier and feature extraction method is important. In Table 1, it can be seen that the canny edge, LoG, and DWT features have produced underperforming KNN models, with the lowest accuracy of 50%. These models would be inappropriate for industrial use because an accuracy of 50% means that there is an equal chance that a false classification is made as a correct one, which results in a ‘guessing’ model.

When trained from scratch on the dataset, the CNN model achieved a mean accuracy of 79.4%, which is 10.6 percentage points lower than that of the ORB-SVM. This result further backs the conclusions of other researchers that deep learning models require a large amount of data to achieve a higher accuracy. Although this is not an underperforming result, the margin for error is higher than that of the ORB-SVM, which could prove more costly, especially if the model would be used to analyse a large volume of images. With respect to cost and safety, the rate of FNs should be considered more carefully compared to FPs because missed damages can result in destructive errors further down the line. This would not only prove more costly but may increase the risk of danger to those nearby, whereas non-existent detected damage may cause an unnecessary further inspection, but this would be the extent of the consequence caused by the incorrect classification.

With a training time of 4667.4 s, the CNN model took considerably longer to train than all machine learning classifiers, with the DWT-SVM having the lowest time taken to train at 8.7 s. High training times can be problematic if there is a limited timeframe for an inspection to take place. Any changes to the model or added training images requiring the model to be re-trained would result in a lengthy wait each time; therefore, it is important for this process to be minimised. For inference, the MLP models were the fastest, with the highest FPS of 526 from the DWT-MLP. Whether inference takes place offline or in real time, it will affect the overall processing time. If inference takes place onboard a drone, then fast and efficient detection is even more important to allow for all frames to be processed within the given flight time and reduce the processing power needed to process each frame. It is important to find the appropriate trade-off between accuracy and efficiency when evaluating a model. For example, the ORB-SVM, with an accuracy of 90%, achieved 63 FPS during inference compared to the fastest inference of 526 FPS by the DWT-MLP model. This model only achieved an accuracy of 50.8%, which is poor; therefore, it would be best to select the ORB-SVM model. Indubitably, the accuracy of the model should take more priority over efficiency because an accuracy that is too low can cause further delays to the damage detection process, as mentioned.

The conclusion of this work suggests that machine learning methods can compete with current state-of-the-art deep learning models, especially when training data are a limiting factor. This is a common issue, and public data are scarcely made available by private companies. This is largely due to the difficulties faced when obtaining the images. For the most part, the wind turbine would need to be shut down to allow the drone to approach close enough to the blades to attain detailed images. The weather conditions can further limit this opportunity, specifically, strong winds and heavy rain, which would require waterproofing and a robust control system. Situations in which the turbine is in motion introduce the issue of motion blurring to captured images, and these would need pre-processing first before being used in a dataset.

With the recent advances in AI, realistic images can be generated from key phrase inputs to aid with dataset building. This can be a strong tool for collecting images of damaged blades that closely resemble those of a real damaged blade. This is a unique way for a model to learn trends and patterns exhibited in real-world images and learn to classify based on this; however, real-world images would be more desirable because these would be 100% representations of what the model would be exposed to during inference, including not just the blade and damage itself but also the effects of environmental factors, such as weather, lighting, and noise.

This work shows that less computationally expensive approaches can be used in this domain while still providing competitive performance. This is becoming more important as the focus shifts more toward real-world applications of this technology [47]. It is useful to experiment with cutting-edge systems to discover new possible solutions; however, if these findings cannot be translated into scalable, practical solutions, then their use is limited. Refining existing technology and ensuring its reliable and efficient use in the field is what is needed for the technology to achieve its intended purpose. Since the batteries of a drone can only allow for short flights at a time [48], it is crucial to maximise this flight

time when possible, which includes reducing the onboard computer power requirements whilst minimising the impact on performance. For damage detection, this could further be improved by looking at which features contribute to a certain class more than others. The interpretation of a classifier's output may give more insight into the true effects of selected features and allow us to make informed decisions for model tuning or feature filtering and further reduce unnecessary computations. Future work should be developed and tested on real-world scenarios that emulate the true environment to ensure efficiency and feasibility on a larger scale.

5. Conclusions

In this paper, four machine learning classifiers were trained using six different feature extraction methods to create a total of 24 models. The combination of ORB features and the support vector machine created the model with the highest accuracy of 90%. The model was tested on an image with different interferences applied to simulate environmental challenges. The model correctly classified the images with little change in model confidence, proving adequate robustness. It was then compared with a CNN model trained on the same dataset. The ORB-SVM produced a classification accuracy that was 10.6% higher than that of the CNN model. The ORB-SVM classified images faster at 63 FPS, compared to the 1.3 FPS speed of the CNN model. The model was further compared with three current advanced algorithms, namely, EfficientNetB0, MobileNetV2, and DenseNet121. ORB-SVM was found to classify images significantly faster, with little impact on classification accuracy, and, for both training and inference, ORB-SVM consumed less memory than the other three models.

Author Contributions: Methodology, O.B.; software, O.B.; supervision, A.K., S.S., M.G. and K.C.; writing—original draft, O.B.; writing—review and editing, O.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Engineering and Physical Sciences Research Council (EPSRC).

Data Availability Statement: The publicly available dataset can be found at the following web address: LE_erosion (<https://www.kaggle.com/datasets/oscarbest/le-erosion>, accessed on 11 September 2024).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Feijóo, M.d.C.; Zambrano, Y.; Vidal, Y.; Tutivén, C. Unsupervised damage detection for offshore jacket wind turbine foundations based on an autoencoder neural network. *Sensors* **2021**, *21*, 3333. [CrossRef] [PubMed]
2. Katsaprakakis, D.A.; Papadakis, N.; Ntintakis, I. A comprehensive analysis of wind turbine blade damage. *Energies* **2021**, *14*, 5974. [CrossRef]
3. Wang, W.; Xue, Y.; He, C.; Zhao, Y. Review of the typical damage and damage-detection methods of large wind turbine blades. *Energies* **2022**, *15*, 5672. [CrossRef]
4. Herring, R.; Dyer, K.; Martin, F.; Ward, C. The increasing importance of leading edge erosion and a review of existing protection solutions. *Renew. Sustain. Energy Rev.* **2019**, *115*, 109382. [CrossRef]
5. Shi, J.; Hu, M.; Zhang, Y.; Chen, X.; Yang, S.; Hallak, T.S.; Chen, M. Dynamic Analysis of Crane Vessel and Floating Wind Turbine during Temporary Berthing for Offshore On-Site Maintenance Operations. *J. Mar. Sci. Eng.* **2024**, *12*, 1393. [CrossRef]
6. Zhang, K.; Pakrashi, V.; Murphy, J.; Hao, G. Inspection of Floating Offshore Wind Turbines Using Multi-Rotor Unmanned Aerial Vehicles: Literature Review and Trends. *Sensors* **2024**, *24*, 911. [CrossRef]
7. Rinaldi, G.; Thies, P.R.; Johanning, L. Current status and future trends in the operation and maintenance of offshore wind turbines: A review. *Energies* **2021**, *14*, 2484. [CrossRef]
8. Chung, H.-M.; Maharjan, S.; Zhang, Y.; Eliassen, F.; Strunz, K. Placement and routing optimization for automated inspection with unmanned aerial vehicles: A study in offshore wind farm. *IEEE Trans. Ind. Inform.* **2020**, *17*, 3032–3043. [CrossRef]
9. Mishra, M. Convolutional Neural Networks, Explained. Towards Data Science 2020. Available online: <https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939> (accessed on 14 August 2024).
10. Xiao, C.; Liu, Z.; Zhang, T.; Zhang, X. Deep learning method for fault detection of wind turbine converter. *Appl. Sci.* **2021**, *11*, 1280. [CrossRef]
11. Thompson, N.C.; Greenewald, K.; Lee, K.; Manso, G.F. The computational limits of deep learning. *arXiv* **2020**, arXiv:2007.05558.

12. Shihavuddin, A.; Chen, X.; Fedorov, V.; Nymark Christensen, A.; Andre Brogaard Riis, N.; Branner, K.; Bjorholm Dahl, A.; Reinhold Paulsen, R. Wind turbine surface damage detection by deep learning aided drone inspection analysis. *Energies* **2019**, *12*, 676. [\[CrossRef\]](#)
13. Aird, J.A.; Barthelmie, R.J.; Pryor, S.C. Automated Quantification of Wind Turbine Blade Leading Edge Erosion from Field Images. *Energies* **2023**, *16*, 2820. [\[CrossRef\]](#)
14. Bradley, D.; Roth, G. Adaptive thresholding using the integral image. *J. Graph. Tools* **2007**, *12*, 13–21. [\[CrossRef\]](#)
15. Zulpe, N.; Pawar, V. GLCM textural features for brain tumor classification. *Int. J. Comput. Sci. Issues (IJCSI)* **2012**, *9*, 354.
16. Sirmacek, B.; Unsalan, C. Damaged building detection in aerial images using shadow information. In Proceedings of the 2009 4th International Conference on Recent Advances in Space Technologies, Istanbul, Turkey, 11–13 June 2009; pp. 249–252.
17. Deng, L.; Guo, Y.; Chai, B. Defect detection on a wind turbine blade based on digital image processing. *Processes* **2021**, *9*, 1452. [\[CrossRef\]](#)
18. Kumar, P.P.; Rao, I.K. Log Gabor filter based feature detection in image verification application. *Int. J. Sci. Res.* **2014**, *3*, 703–707.
19. Wang, J.-F.; Zhao, G.; Zhai, X.-Q.; Feng, L.-J. Study on the Improved PSO Algorithm Used in Coal Mine Safety Resource Allocation. In Proceedings of the 23rd International Conference on Industrial Engineering and Engineering Management 2016: Theory and Application of Industrial Engineering, Bali, Indonesia, 4–7 December 2016; pp. 149–156.
20. Yang, X.; Zhang, Y.; Lv, W.; Wang, D. Image recognition of wind turbine blade damage based on a deep learning model with transfer learning and an ensemble learning classifier. *Renew. Energy* **2021**, *163*, 386–397. [\[CrossRef\]](#)
21. Khambampati, A.; Liu, D.; Konki, S.; Kim, K. An automatic detection of the ROI using Otsu thresholding in nonlinear difference EIT imaging. *IEEE Sens. J.* **2018**, *18*, 5133–5142. [\[CrossRef\]](#)
22. Wang, Y.; Zou, L. Research on surface damage detection of wind turbinebladebased on machine vision. *Proc. J. Phys. Conf. Ser.* **2022**, *2184*, 012018. [\[CrossRef\]](#)
23. Best, O. LE Erosion Dataset. 2024. Available online: <https://www.kaggle.com/> (accessed on 11 September 2024).
24. Brscotia. Scottish Wind Turbine Close Up—Parrot Bebop 2 Power FPV Drone Footage. 2017. Available online: <https://www.youtube.com/watch?v=h069SuhUxZs> (accessed on 23 July 2024).
25. DefaultName. Drone Inspecting Wind Turbine. 2015. Available online: <https://www.youtube.com/watch?v=VB24RIm3yDE> (accessed on 15 July 2024).
26. MegaZoom. Wind Turbine Close Up/MegaZoom. 2022. Available online: <https://www.youtube.com/watch?v=5zVgJXyAe-I&t=18s> (accessed on 15 July 2024).
27. Mosegaard, I. Blade Inspection Video. 2014. Available online: <https://www.youtube.com/watch?v=meDXOM4poQA> (accessed on 18 July 2024).
28. Reynolds, N. Windmill Drone Footage. 2018. Available online: <https://www.youtube.com/watch?v=TzN--DftGuw> (accessed on 15 July 2024).
29. Services, H.D. Raw Video from a Wind Turbine Flyover. 2021. Available online: <https://www.youtube.com/watch?v=98oAdazXQgs> (accessed on 17 July 2024).
30. UAV, D. Wind Turbine Inspection Tests by DroneworX Technology Belgium. 2014. Available online: <https://www.youtube.com/watch?v=ZPa0ZIPWjok> (accessed on 15 July 2024).
31. Samra, A.S.; Allah, S.E.T.G.; Ibrahim, R.M. Face recognition using wavelet transform, fast Fourier transform and discrete cosine transform. In Proceedings of the 2003 46th Midwest Symposium on Circuits and Systems, Cairo, Egypt, 27–30 December 2003; pp. 272–275.
32. Tao, Z.; Wei, T.; Li, J. Wavelet multi-level attention capsule network for texture classification. *IEEE Signal Process. Lett.* **2021**, *28*, 1215–1219. [\[CrossRef\]](#)
33. Anand, A.; Tripathy, S.S.; Kumar, R.S. An improved edge detection using morphological Laplacian of Gaussian operator. In Proceedings of the 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India, 19–20 February 2015; pp. 532–536.
34. Zhou, W.; Gao, S.; Zhang, L.; Lou, X. Histogram of oriented gradients feature extraction from raw bayer pattern images. *IEEE Trans. Circuits Syst. II Express Briefs* **2020**, *67*, 946–950. [\[CrossRef\]](#)
35. Guangyun, W.; Zhiping, Z. An improved ORB feature extraction and matching algorithm. In Proceedings of the 2021 33rd Chinese Control and Decision Conference (CCDC), Kunming, China, 22–24 May 2021; pp. 7289–7292.
36. Mittal, H.; Garg, N. Hand Symbol Recognition Using Canny Edge Algorithm And Convolutional Neural Network. In Proceedings of the 2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), Gwalior, India, 14–16 March 2024; pp. 1–5.
37. Das, D.; Singh, M.; Mohanty, S.S.; Chakravarty, S. Leaf disease detection using support vector machine. In Proceedings of the 2020 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 28–30 July 2020; pp. 1036–1040.
38. Mallick, S. Support Vector Machines (SVM). 2018. Available online: <https://learnopencv.com/support-vector-machines-svm/> (accessed on 2 August 2024).
39. Christiansen, S.D. Ischemic Stroke Thrombus Characterization through Quantitative Magnetic Resonance Imaging. Doctoral Dissertation, The University of Western Ontario, London, ON, Canada, 2021.
40. Li, J.; Zhang, J.; Zhang, J.; Zhang, S. Quantum KNN classification with K Value selection and neighbor selection. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2023**, *43*, 1332–1345. [\[CrossRef\]](#)

41. Shafi, A. K-Nearest Neighbors (KNN) Classification with Scikit-Learn. 2023. Available online: <https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn> (accessed on 5 August 2024).
42. Juna, A.; Umer, M.; Sadiq, S.; Karamti, H.; Eshmawi, A.A.; Mohamed, A.; Ashraf, I. Water quality prediction using KNN imputer and multilayer perceptron. *Water* **2022**, *14*, 2592. [\[CrossRef\]](#)
43. Zhu, Z. Effects of environmental factors on ozone flux over a wheat field modeled with an artificial neural network. *Adv. Meteorol.* **2019**, *2019*, 1257910. [\[CrossRef\]](#)
44. O'shea, K.; Nash, R. An introduction to convolutional neural networks. *arXiv* **2015**, arXiv:1511.08458.
45. Dileep, P.; Das, D.; Bora, P.K. Dense layer dropout based CNN architecture for automatic modulation classification. In Proceedings of the 2020 National Conference on Communications (NCC), Kharagpur, India, 21–23 February 2020; pp. 1–5.
46. Rakotomamonjy, A. Optimizing Area Under Roc Curve with SVMs. In Proceedings of the ROCAI, Valencia, Spain, 22 August 2004; pp. 71–80.
47. Memari, M.; Shakya, P.; Shekaramiz, M.; Seibi, A.C.; Masoum, M.A. Review on the Advancements in Wind Turbine Blade Inspection: Integrating Drone and Deep Learning Technologies for Enhanced Defect Detection. *IEEE Access* **2024**, *12*, 33236–33282. [\[CrossRef\]](#)
48. Aquilina, J.P.; Farrugia, R.N.; Sant, T. On the energy requirements of UAVs used for blade inspection in offshore wind farms. In Proceedings of the 2019 Offshore Energy and Storage Summit (OSES), Brest, France, 10–12 July 2019; pp. 1–7.

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