Defect Detection on Wind Turbine Blades using Computer Vision and Image Processing Techniques

by

© **Syed Zeeshan Haider Rizvi, B.Eng.**

A thesis submitted to the School of Graduate
Studies in partial fulfillment of the requirements for
the degree of Master of Engineering.

Department of Electrical and Computer Engineering
Memorial University

October 2024

St. John's, Newfoundland and Labrador, Canada

Abstract

The research presented in this thesis enhances the accuracy and efficiency of wind turbine blade (WTB) inspections using advanced image processing and computer vision techniques. It addresses significant challenges in existing methods by introducing novel segmentation and defect detection strategies for WTBs, utilizing the high-resolution Blade30 drone imagery dataset.

The thesis is comprised of two main components. First, it develops an improved U-Net model, named Pixel U-Net, which incorporates pixel shuffle and unshuffle operations to enhance binary segmentation of WTBs. This model is specifically tailored to overcome the complex backgrounds in drone images that typically hinder accurate segmentation. Extensive testing shows that Pixel U-Net and its variations outperform existing models by effectively isolating WTB areas from challenging backgrounds, thus setting the stage for more reliable defect detection.

Second, the study introduces a cascaded approach that combines refined WTB images with YOLO-based object detection to identify and classify defects. This method significantly reduces false positives in complex backgrounds, enhancing detection reliability. The thesis evaluates various YOLO configurations, showing that the proposed methodology outperforms traditional WTB defect detection techniques.

This work significantly advances automated visual inspection systems for renewable energy assets, enhancing both the precision of defect detection and the operational efficiency of maintenance protocols for wind turbines. These improvements could lead to more reliable and cost-effective maintenance strategies, which are vital for the sustainable operation of wind energy projects. The findings have the potential to influence future developments in the field, promoting more effective maintenance approaches for wind turbines.

Acknowledgements

I would start by thanking God for helping me academically grow during the course of this research.

This work is a dedication to my late father, who would have been very proud to see me achieve this feat, and also to my mother, my two sisters, and my dearest wife for their unwavering support throughout this journey. You have all been my strength!

I want to thank my supervisor, Dr. Mohsin Jamil and my co-supervisor, Dr. Weimin Huang, for their expert mentorship during the course of this thesis work.

I also wish to thank my friends for their emotional and moral support throughout the course of this degree.

I further want to thank the examiners for taking the time to review this document and all the individuals involved in the peer review process for the manuscripts presented in this thesis.

Co-authorship Statement

I, Syed Zeeshan Haider Rizvi (Z.R.), am the primary author and copyright owner for all the chapters in this thesis.

The first article presented in this thesis titled, 'Pixel U-Net: An Improved Version of U-Net for Binary Segmentation of Wind Turbine Blades' is published in the Signal, Image and Video Processing journal. The original draft was prepared by Z.R. The codes, methods and experiments for this article were developed by Z.R. The manuscript was reviewed by Mohsin Jamil (M.J.) and Weimin Huang (W.H.). W.H. provided recommendations for further experiments to improve quality.

The second article presented in this thesis titled, 'Enhanced Defect Detection on Wind Turbine Blades using Binary Segmentation Masks and YOLO' is published in the Computers and Electrical Engineering journal. The original draft was prepared by Z.R. The methodology and codes were developed by Z.R. Review of the original draft was done by M.J. and W.H. Improvements to experiments were suggested by W.H.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

WTB	Wind Turbine Blade
YOLO	You Only Look Once
HS	Hierarchical Split
CV	Computer Vision
DL	Deep Learning
CNN	Convolutional Neural Network
FPS	Frames per Second

Chapter 1

Introduction

1.1 Problem Statement

Wind turbine technicians remain at a significant risk of fatal injuries, and their job is considered one of the most dangerous jobs in the energy sector. The responsibilities of a wind turbine technician involve the inspection of wind turbine blades (WTBs) for any defects and contamination on the blade surface. With recent advancements in the domain of computer vision, images captured by drones have been used to analyze wind turbines for any defects or contaminations. Images of different WTBs captured by a drone are given in Figure 1.1. These images are taken from the publicly available Blade30 dataset [1]. The process of WTB surface inspection and monitoring can be automated by training object detection algorithms to recognize any defect or contamination on the WTB surface. Two noticeable problems exist with the currently available methods in this domain.

The first problem is related to the resolution of the images. The images of WTB captured by drones are usually high-resolution images, as can also be noticed from the

Figure 1.1: Drone images of different wind turbine blades taken from Blade30 dataset

size of drone images in the Blade30 dataset. Since wind turbines are big structures with heights reaching over 260 meters and blade lengths reaching about 110 meters, these high-resolution images ensure that details on the WTB surface are captured by the drone. However, with high-resolution images, the training process of object detection algorithms becomes slow.

The second problem is related to the training results of object detectors on WTB images. Since WTB images consist of challenging and diverse backgrounds, the object detection algorithm can often confuse the background as a region of interest as well. This can be noticed in the detection result presented in Figure 1.2. The object detection algorithm classified a section from the background as 'surface contamination - dirt.' In object detection tasks, this is referred to as a false positive detection, as it was a positive detection that was made by the algorithm, but it was incorrect.

Recently, researchers have proposed different methods to identify defects and contamination on WTBs [2] [3] [4]. However, these methods are limited to architectural changes in the neural networks used for the detection. They introduced changes in the You Only Look Once v5 (YOLOv5) architecture and used the modified architectures to identify a maximum of 4 different categories of defects. In actual scenarios, and for a more accurate inspection of defects, the types of defects can be divided into many different categories.

In the domain of defect detection on WTBs, it was crucial to address these issues

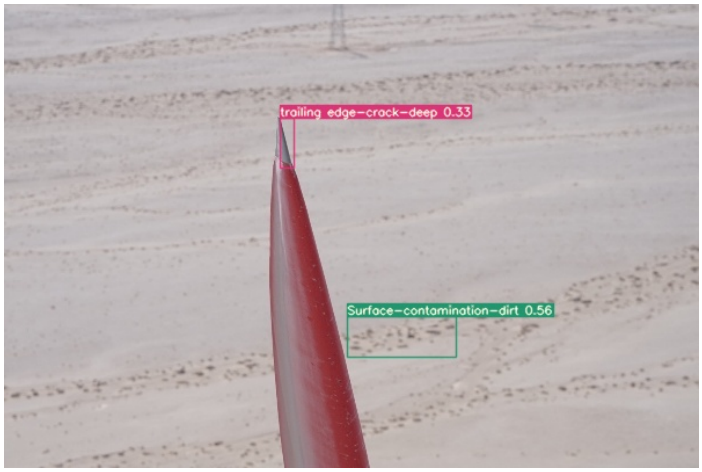


Figure 1.2: Problem of false positives on wind turbine blade images

and ensure that the performance of object detectors to identify defects and contaminations on WTB surface was efficient, robust and accurate. It was also necessary to perform the task for a greater number of categories of defects and contamination than the categories available in the current literature.

1.2 Research Objectives

To tackle the problems mentioned above, this research aimed to develop an efficient methodology to generate accurate defect detection results on WTB images. The main objectives of this thesis are as follows:

- Experiment with segmentation techniques (such as U-Net and its variations) to extract WTB areas from drone images.
- Experiment with image processing techniques (such as edge detection and sliding window) to additionally extract the area around the edges of WTBs.
- Train and test object detection algorithms (such as different versions of YOLO) for defect detection on WTB images.

- Prove the novelty and superiority of the proposed methodology by comparison with existing techniques in the domain of WTB defect detection (such as AFB-YOLO [2], MI-YOLO [3], and MC-YOLO [4]).

1.3 Structure of Thesis

This thesis follows a manuscript format. It is divided into 5 chapters. A brief overview of the contents of each of these chapters is provided below:

- **Chapter 1** contains the problem statement, research objectives, structure of thesis, the main contributions of the thesis, list of publications, and the dataset availability statement.
- **Chapter 2** consists of a thorough literature review into segmentation techniques, object detection in computer vision, and finally, image processing and defect detection on WTBs.
- **Chapter 3** is the research work published in the Signal, Image and Video Processing journal [5]. This chapter covers the first part of the proposed methodology in this thesis, which is the segmentation of WTB images. The chapter proposes a novel architecture for binary segmentation of WTBs called Pixel U-Net and its variations, which outperform existing architectures in training, validation and testing.
- **Chapter 4** is the research work that has been accepted for publication in the Computers and Electrical Engineering journal. This chapter describes the second part of the proposed methodology of this thesis, which is the image processing and defect detection on segmented WTB images. In this chapter, the

proposed methodology has also been compared with existing methods for defect detection on WTBs, and has been proven to outperform those methods.

- **Chapter 5** provides a thorough conclusion of the objectives and findings from the complete research work. Furthemore, this research work proposes a future direction for further continuation of this work.

Please note that parts of Chapter 2 are taken from the published work in Chapter 3 titled,"Pixel U-Net: An Improved Version of U-Net for Binary Segmentation of Wind Turbine Blades", and the published work in Chapter 4 titled, "Enhanced Defect Detection on Wind Turbine Blades Using Binary Segmentation Masks and YOLO".

1.4 Contributions of Thesis

The thesis aims to provide a robust and effective methodology for the purpose of defect detection on WTBs. The main contributions of the thesis work can be summarized as follows:

1. A novel architecture, Pixel U-Net, that outperforms all other architectures under study in validation performance for the binary segmentation of WTB images.

2. Two architectural variations of Pixel U-Net, namely U-Net + HS-Block + Pixel Unshuffle and Pixel U-Net + Attention, that prove that the inclusion of pixel shuffle and pixel unshuffle operations in architecture improves the overall performance of WTB segmentation.

3. A novel pre-processing pipeline using binary masks and edges of WTBs to generate a new version of WTB image dataset that optimizes storage space, training speed, and accuracy.

4. A comparative analysis using different versions of YOLO to demonstrate the performance of YOLO object detectors on WTB images.

1.5 List of Publications

As a part of this thesis, two publications have been presented, which have gone through a peer-review process. The two publications are listed as follows:

1. Syed Zeeshan Rizvi, Mohsin Jamil, and Weimin Huang. Pixel u-net: an improved version of u-net for binary segmentation of wind turbine blades. *Signal, Image and Video Processing*, 18:6299–6307, 2024. The full text of the published paper is available to read at this link: *https://rdcu.be/dKLtV*.

2. Syed Zeeshan Rizvi, Mohsin Jamil, and Weimin Huang. Enhanced defect detection on wind turbine blades using binary segmentation masks and yolo. *Computers and Electrical Engineering*, 120:109615, 2024.

1.6 Data Availability Statement

This thesis work makes use of the Blade30 dataset. The original version of the Blade30 dataset can be found at this link: *https://github.com/cong-yang/Blade30*. In Chapter 4, a refined version of the Blade30 dataset was used, which consisted of 307 images and 9 classes. This refined version of the original Blade30 dataset has been made publicly available for ease of further research and can be found at this link: *https://shorturl.at/VdPOn*.

1.7 Bibliography

[1] Cong Yang, Xun Liu, Hua Zhou, Yan Ke, and John See. Towards accurate image stitching for drone-based wind turbine blade inspection. *Renew. Energy*, 203:267–279, 2023.

[2] Xiukang Ran, Shang Zhang, Hengtao Wang, and Zhaoyang Zhang. An improved algorithm for wind turbine blade defect detection. *IEEE Access*, 10:122171–122181, 2022.

[3] Zhu Xiaoxun, Hang Xinyu, Gao Xiaoxia, Yang Xing, Xu Zixu, Wang Yu, and Liu Huaxin. Research on crack detection method of wind turbine blade based on a deep learning method. *Applied Energy*, 328:120241, 2022.

[4] Zhiming Zhang, Chaoyi Dong, Ze Wei, Weidong Zan, Jianfei Zhao, Fu Hao, and Xiaoyan Chen. A real-time wind turbine blade damage detection method based on an improved yolov5 algorithm. In *International Conference on Image and Graphics*, pages 298–309. Springer, 2023.

[5] Syed Zeeshan Rizvi, Mohsin Jamil, and Weimin Huang. Pixel u-net: an improved version of u-net for binary segmentation of wind turbine blades. *Signal, Image and Video Processing*, 18:6299–6307, 2024.

Chapter 2

Literature Review

As presented in Figure 2.1, the complete methodology proposed in this thesis can be broadly divided into 3 basic parts in terms of image processing and computer vision: image segmentation, edge detection, and object detection. Hence, the literature review has been split into sections that discuss these techniques separately and give an overview of the progress of research in these domains. The last section discusses these techniques in the light of recent research done specifically in the domain of WTB image processing and defect detection.

2.1 Image Segmentation

As displayed in Figure 2.2, image segmentation techniques can be broadly classified into two categories – traditional and deep-learning based. The traditional methods consist of different image processing techniques, such as thresholding, region-based segmentation, edge segmentation, clustering-based segmentation, and active contour segmentation [1]. Thresholding is a simple method used to separate the object from

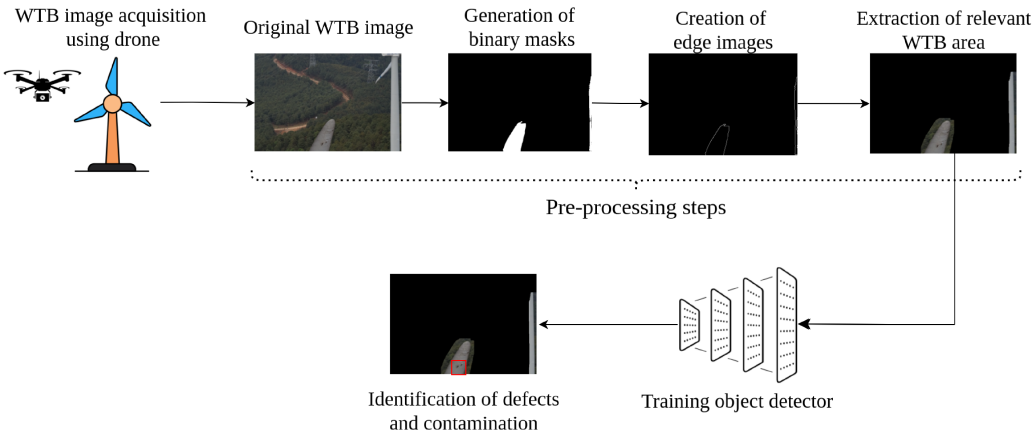


Figure 2.1: Complete methodology for identification of defects and contamination on wind turbine blades

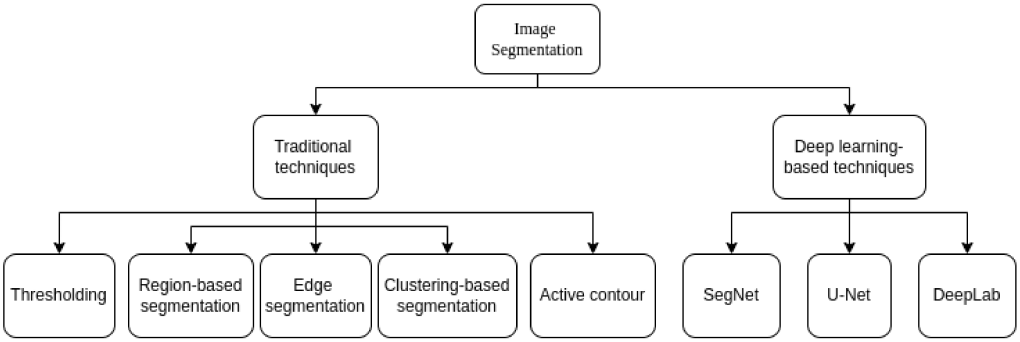


Figure 2.2: Classification of image segmentation techniques

the background by creating a binary image. If pixel values exceed the threshold value, they are assigned the value of one (which represents white) and a value of zero (which represents black) if the values are below the threshold. Thresholding is useful for applications in which a sharp change in contrast is present between the foreground and background. In region-based segmentation, the neighbouring pixel values with similar properties (for example, colour or intensity) are grouped together into regions. This technique is particularly useful when the regions of interest are homogenous. As the name suggests, the edge segmentation technique makes use of edges to identify

the boundaries of different objects, thereby segmenting them. In clustering-based segmentation, pixels are grouped into clusters using clustering algorithms. Each cluster then represents a different region of the image. This can be a useful technique when the image consists of distinct regions with colour intensities. Active contour methods use evolving curves within an image to identify object boundaries. These methods can be useful when dealing with objects which have complex shapes, as the curve can be aligned until it fits the object's boundary accurately.

The more recent segmentation techniques fall under the umbrella of deep learning. These modern techniques use convolutional neural networks (CNNs) to derive feature information from images and identify patterns. Unlike the standard neural networks which receive a flattened input, CNNs are unique and better suited for use with images based on their ability to preserve spatial information as they can work with multi-dimensional arrays. CNNs learn to identify the features of complex objects in an image, thereby developing a better dynamic understanding of what is present in an image. This is done by utilizing mathematical operations, some of which are convolution, logistic regression, normalization and activation. One of the earliest deep learning-based image segmentation architectures was introduced in 2015 by the name of SegNet [2]. This architecture discussed the idea of 'encoder' and 'decoder' in an architecture designed for image segmentation. The encoder path extracts low-level features from the image by reducing the spatial dimension, and increasing the number of filters. On the other hand, the decoder increases the spatial dimension by decreasing the number of filters again. This way, the encoder develops a good understanding of the features, whereas the decoder develops a rich understanding of the spatial information. In the same year, the popular U-Net architecture [3] was published, which was also built on the idea of encoder and decoder layers, connected together with skip connections to prevent information loss and generate better feature

representation. The skip connections also ensure that the developed understanding of features in the encoder and the developed understanding of spatial information in the decoder are regulated properly throughout the architecture. In the U-Net architecture, the encoder and decoder are also connected by a bottleneck layer that extracts the most abstract and high-level features. Following this work, DeepLab [4] was introduced, which introduced the techniques of atrous convolutions and atrous spatial pyramid pooling to further increase the efficiency of segmentation models.

2.2 Edge Detection

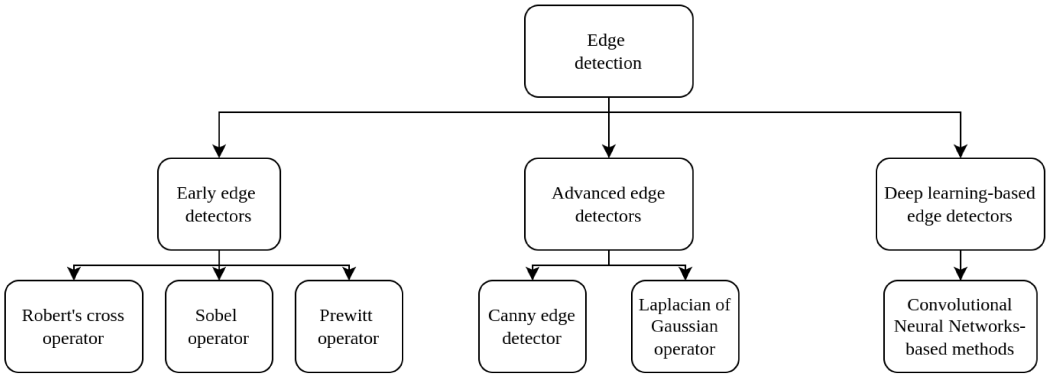


Figure 2.3: Classification of edge detection techniques

In image processing, edges are defined by an abrupt change in pixel intensities in the image. The work done to identify edges in images has progressed through the years, starting from early-stage detectors to advanced edge detectors to deep learning-based edge detectors. These categorizations are displayed in Figure 2.3. Robert's cross operator [5] is one of the earliest edge detection algorithms that was proposed. It uses diagonal gradients to compute the approximate magnitude of the spatial gradient of an image. It identifies points in an image where the brightness changes sharply, effectively highlighting the edges by applying small 2x2 kernels to calculate differences

diagonally across pixel values. In the following years, the Sobel edge detector [6] was introduced, which enhanced the detection of edges by combining results from horizontal and vertical edge detectors based on 3x3 convolution kernels. In the category of early edge detectors, the Prewitt operator [7] was also introduced, which works in a similar way to the Sobel operator but has uniform values in the kernel, unlike the Sobel operator, which places more weight towards the center of the kernel.

Consequently, in the category of advanced edge detection algorithms, the Canny edge detector [8] was introduced, which continues to be a very popular edge detector due to its accurate and refined edge detections. Canny edge detector works in a multi-step process, where it first removes the noise from an image, and then it calculates the strengths and direction of edges to refine them further and make them sharper. Finally, a thresholding operation is performed that ensures only the most significant edges are kept. Another advanced edge detector is the Laplacian of Gaussian (LoG) operator [9] which applies the Laplacian operator on images to enhance regions of rapid intensity change, thereby making the edges prominent.

With the emerging research in deep learning (DL), researchers have also experimented with convolutional neural networks (CNNs) to perform the task of edge detection [10]. These modern methods perform the task of edge detection by learning from vast amounts of data and not relying on hand-crafted features. However, as with any other DL methods, more computational resources and execution time is required for these methods.

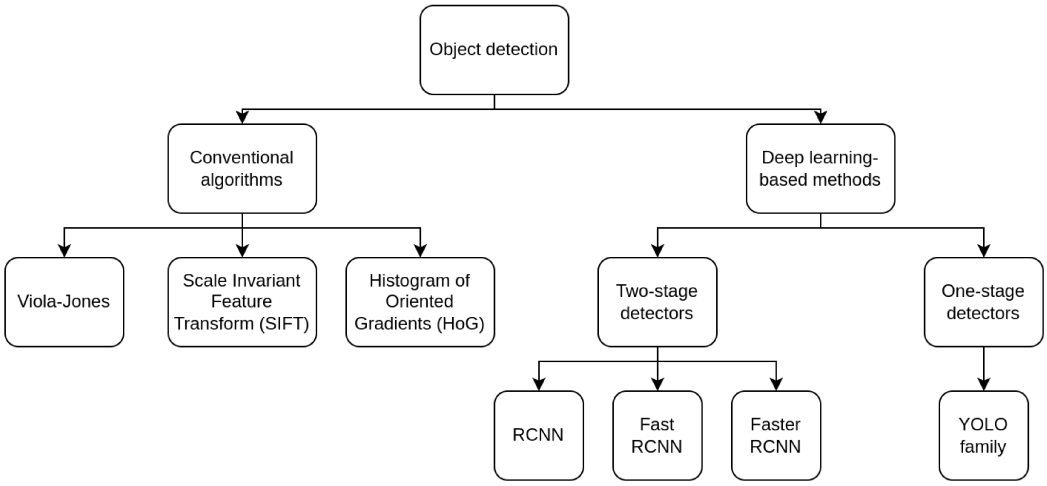


Figure 2.4: Classification of object detection techniques

2.3 Object Detection

Figure 2.4 presents an overview of different object detection methods. Before intensive research in the DL domain for object detection, conventional algorithms were used. For these algorithms, features had to be extracted before passing them to a classification method. One of the earliest of these algorithms was Viola-Jones object detection [11], which proposed the approach of making integral images and then performing detection using Haar features. Building on the research in object detection and recognition, in 2004, Scale Invariant Feature Transform (SIFT) [12] was used to extract features from an image and compare them with an existing database of features to recognize the object present in the image. Following this, in 2006, the Histogram of Oriented Gradient (HOG) [13] approach was introduced, which presented a new approach to object detection using support vector machine as a classifier. The extraction of features from the frequency domain for object detection has also been studied in various research works [14], [15]. Deep learning methods can perform object detection using neural networks without the need for extracting features as a pre-processing step. These algorithms use convolutional neural networks (CNNs) to perform computation

on images. Within deep learning, object detection algorithms can be broadly divided into two categories: two-stage, and one-stage detectors. Region-based convolutional neural network (RCNN) [16] is one of the earliest two-stage detectors proposed. It extracts different proposed regions for an object in an image and then passes each of these proposed regions to a CNN to make predictions for the class of the object. This makes the object detection process slow, and it cannot be used for real-time object detection. To tackle the issue, Fast RCNN [17] and Faster RCNN [18] were proposed as improved versions of RCNN. However, these algorithms continued to be additions to the two-stage detectors.

You Only Look Once, or YOLO [19], was the first single-stage detector proposed in 2016. This algorithm could predict all the objects present in an image by passing the image to a CNN just once, which defines the algorithm's name. YOLO allows faster object detection and can perform well on real-time applications. YOLO architectures are generally divided into three main parts: backbone, neck and head. The backbone extracts features from the image, the neck merges multi-scale features to assist with detections of variable sizes, and the head outputs the class and bounding box for the object. Many improved versions of YOLO have been proposed over recent years. Focusing on the advancements made in the YOLO series since 2020, YOLOv5 [20] streamlined the object detection pipeline by introducing different model variants for balancing speed and performance, advanced data augmentation techniques, and auto anchor calculation. The modular design introduced with YOLOv5 contributed significantly to scalability, as it made it easy to adjust the network size for different applications. YOLOv5 uses the Cross Stage Partial Darknet (CSPDarknet) as its backbone, which is an improvement of the original Darknet backbone used in the earliest versions of YOLO. It further uses the Path Aggregation Network (PANet) in the neck to enhance feature propagation between different scales and improve object

detection, particularly for small objects. In 2022, YOLOv6 [21] was introduced, which was focused on industrial-grade performance and offered further improvements in the architecture. YOLOv6 incorporated anchor-free detection heads and a more efficient backbone network using EfficientRep, improving inference speed and accuracy on small objects. It further used RepVGG-based blocks to optimize performance of the backbone. The seventh version of YOLO, referred to as YOLOv7 [22], was presented in the same year. YOLOv7 further optimized the backbone by using an extended efficient layer aggregation network (E-ELAN), which improves feature fusion and the model's ability to learn better representations from different layers. YOLOv7 also incorporated new strategies for feature pyramid network (FPN) and PANet optimizations in the head part, leading to better multi-scale object detection. With further improvements in the architecture, YOLOv8 [23] was released in 2023. YOLOv8 refined the advancements in the YOLO series by adopting a more flexible framework with enhancements in the detection head and feature extraction, achieving state-of-the-art performance across a wider range of tasks.

2.4 Image Processing and Defect Detection on WTBs

Segmenting the WTB area from a challenging background is a crucial pre-processing step in identifying defects on the WTB surface. Once a segmented WTB image is obtained, it can be mapped onto the original WTB image to extract only the relevant area, which can be used to train an object detector for surface defects. A method proposed in [24] uses "regression crop" to crop the surface of WTB out of images and perform detection only across the surface. A technique presented in [25] gives an

improved version of U-Net that can effectively segment WTB from infrared images using hierarchical split depthwise separable convolution operations. Another improved version of U-Net specifically to segment WTB areas was proposed in [26], which made use of an efficient channel attention network and point wise spatial attention network. In terms of segmentation of WTB images, state-of-the-art performance was reported in another improved version of U-Net called Pixel U-Net [27], in which the max pooling and transpose convolution operations in the U-Net architecture were replaced by pixel shuffle and unshuffle operations.

Since WTB surfaces are large and the whole blade surface cannot be captured in high definition in a single drone image, researchers have proposed image stitching techniques to generate accurate images of the complete WTB image. One such technique was proposed in [28], in which authors perform different geometrical calculations to align images belonging to the same WTB and create a panoramic image. Another technique was proposed in [29] that features a two-level process that combines coarse-grained initial alignment using blade edges and drone-blade distances with fine-grained adjustments optimized by texture and shape losses. Other than extracting relevant areas from images of WTBs, work has also been done to improve the quality of WTB images captured in varying lighting conditions. In [30], a method that uses cartoon and texture maps to enhance the illumination of WTB images is proposed.

In addition to research on pre-processing techniques used on WTB images, researchers have experimented with DL to build networks that can be trained for the specific application of identifying defects on WTBs. A method was proposed in [31] that can identify small defects on WTB using an improved version of YOLOv5 to perform the task. In another work [32], an attention and feature-balanced version of

Table 2.1: Recently proposed methods for WTB defect detection

	mAP @ 0.5 (%)	FPS (frames/sec)	Number of Classes
AFB-YOLO [32]	83.7	63.4	2
MI-YOLO [33]	92.1	-	1
MI-YOLO + Alpha IoU [33]	93.2	-	1
MC-YOLO [34]	94.2	25.1	4

YOLO (AFB-YOLO) is proposed, which helps in gathering more feature information for improved detection. In this improved architecture for YOLOv5s (where s stands for small), the authors also replace the intersection over union (IoU) loss with efficient IoU (EIoU) loss for bounding box calculations. In another work [33], the authors introduced an improved version of YOLOv5s called multivariate information YOLO (MI-YOLO) for defect detection on WTBs. In this architecture, authors replace some parts from the original YOLOv5s backbone and also infuse the MobileNetv3 backbone in the network. In their ablation study, they also extend their workings to replace the IoU loss with Alpha-IoU loss for bounding box calculations. In [34], another improvement to YOLOv5s was proposed called MC-YOLO. This modified architecture replaced the complete original backbone of YOLOv5s with the MobileNetv3 backbone. For real-time monitoring of WTB, the inference speed is very important. This is addressed in [35], where the researchers proposed an improved version of YOLOX-Nano which greatly increases the inference speed.

Table 2.1 gives an overview of the recently proposed methods for defect detection on WTBs. As it can be noted from the table, these methods used custom datasets that had different number of defects, ranging from a minimum of 1 to a maximum of 4 defects. The accuracy comparisons have been provided in the respective research papers in the form of mean average precision (mAP) score, which is one of the most commonly used metric in the domain of object detection. A higher frames per second (FPS) score makes the method more suitable for real-time application, which is

another useful calculation to have when deploying the trained algorithm on a drone to identify defects on WTBs in real-time. The obtained FPS score was not provided in [33].

2.5 Bibliography

[1] B Basavaprasad and S Hegadi Ravindra. A survey on traditional and graph theoretical techniques for image segmentation. *Int. J. Comput. Appl*, 975:8887, 2014.

[2] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv:1505.07293*, 2015.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Med. Image. Comput. Comput. Assist. Interv.–MICCAI 2015: 18th Int. Conf., Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern. Anal. Mach. Intell.*, 40(4):834–848, 2017.

[5] Georgios Gennis, Argyro Kamperi, Vassilis Alimisis, Christos Dimas, and Paul P Sotiriadis. An area-efficient, analog integrated image edge detector based on the robert's cross operator. In *2023 12th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pages 1–4. IEEE, 2023.

[6] P Vinista and M Milton Joe. A novel modified sobel algorithm for better edge detection of various images. *International journal of emerging technologies in engineering research (IJETER)*, 7(3):25–31, 2019.

[7] Sri Rahmawati, Retno Devita, Ruri Hartika Zain, Eva Rianti, Najla Lubis, and Anjar Wanto. Prewitt and canny methods on inversion image edge detection: an evaluation. In *Journal of Physics: Conference Series*, volume 1933, page 012039. IOP Publishing, 2021.

[8] Renjie Song, Ziqi Zhang, and Haiyang Liu. Edge connection based canny edge detection algorithm. *Pattern Recognition and Image Analysis*, 27:740–747, 2017.

[9] Bickey Kumar Shah, Vansh Kedia, Rohan Raut, Sakil Ansari, and Anshul Shroff. Evaluation and comparative study of edge detection techniques. *IOSR Journal of Computer Engineering*, 22(5):6–15, 2020.

[10] Xavier Soria Poma, Edgar Riba, and Angel Sappa. Dense extreme inception network: Towards a robust cnn model for edge detection. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1923–1932, 2020.

[11] Modesto Castrillón, Oscar Déniz, Daniel Hernández, and Javier Lorenzo. A comparison of face and facial feature detectors based on the viola–jones general object detection framework. *Machine Vision and Applications*, 22:481–494, 2011.

[12] Zahra Hossein-Nejad and Mehdi Nasri. An adaptive image registration method based on sift features and ransac transform. *Computers & Electrical Engineering*, 62:524–537, 2017.

[13] Nooshin Nabizadeh and Miroslav Kubat. Brain tumors detection and segmentation in mr images: Gabor wavelet vs. statistical features. *Computers & Electrical Engineering*, 45:286–301, 2015.

[14] M Ahsan, MA Based, J Haider, M Kowalski, et al. An intelligent system for automatic fingerprint identification using feature fusion by gabor filter and deep learning. *Computers and Electrical Engineering*, 95:107387, 2021.

[15] Haewon Byeon, Mohammad Shabaz, Kapil Shrivastava, Anjali Joshi, Ismail Keshta, Rajvardhan Oak, Pavitar Parkash Singh, and Mukesh Soni. Deep learning model to detect deceptive generative adversarial network generated images using multimedia forensic. *Computers and Electrical Engineering*, 113:109024, 2024.

[16] Hao Wang, Mengjiao Li, and Zhibo Wan. Rail surface defect detection based on improved mask r-cnn. *Computers and Electrical Engineering*, 102:108269, 2022.

[17] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[18] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[20] Glenn Jocher. Ultralytics yolov5, 2020.

[21] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, Yiduo Li, Bo Zhang, Yufei Liang, Linyuan Zhou, Xiaoming Xu, Xiangxiang Chu, Xiaoming Wei, and Xiaolin Wei. Yolov6: A single-stage object detection framework for industrial applications, 2022.

[22] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023.

[23] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.

[24] Weijie Zhou, Zijun Wang, Minshu Zhang, and Liwen Wang. Wind turbine actual defects detection based on visible and infrared image fusion. *IEEE Transactions on Instrumentation and Measurement*, 72:1–8, 2023.

[25] Junfeng Yu, Yunze He, Hao Liu, Fan Zhang, Jie Li, Gaosen Sun, Xiaofei Zhang, Ruizhen Yang, Pan Wang, and Hongjin Wang. An improved u-net model for infrared image segmentation of wind turbine blade. *IEEE Sens. J.*, 23(2):1318–1327, 2022.

[26] Long Wang, Jinxu Yang, Chao Huang, and Xiong Luo. An improved u-net model for segmenting wind turbines from uav-taken images. *IEEE Sens. Lett.*, 6(7):1–4, 2022.

[27] Syed Zeeshan Rizvi, Mohsin Jamil, and Weimin Huang. Pixel u-net: an improved version of u-net for binary segmentation of wind turbine blades. *Signal, Image and Video Processing*, 18:6299–6307, 2024.

[28] Junfeng Yu, Yunze He, Fan Zhang, Gaosen Sun, Yuejun Hou, Hao Liu, Jie Li, Ruizhen Yang, and Hongjin Wang. An infrared image stitching method for wind turbine blade using uav flight data and u-net. *IEEE Sensors Journal*, 23(8):8727–8736, 2023.

[29] Cong Yang, Xun Liu, Hua Zhou, Yan Ke, and John See. Towards accurate image stitching for drone-based wind turbine blade inspection. *Renew. Energy*, 203:267–279, 2023.

[30] Yeping Peng, Weijiang Wang, Zhen Tang, Guangzhong Cao, and Shengxi Zhou. Non-uniform illumination image enhancement for surface damage detection of wind turbine blades. *Mechanical Systems and Signal Processing*, 170:108797, 2022.

[31] Rui Zhang and Chuanbo Wen. Sod-yolo: a small target defect detection algorithm for wind turbine blades based on improved yolov5. *Advanced Theory and Simulations*, 5(7):2100631, 2022.

[32] Xiukang Ran, Shang Zhang, Hengtao Wang, and Zhaoyang Zhang. An improved algorithm for wind turbine blade defect detection. *IEEE Access*, 10:122171–122181, 2022.

[33] Zhu Xiaoxun, Hang Xinyu, Gao Xiaoxia, Yang Xing, Xu Zixu, Wang Yu, and Liu Huaxin. Research on crack detection method of wind turbine blade based on a deep learning method. *Applied Energy*, 328:120241, 2022.

[34] Zhiming Zhang, Chaoyi Dong, Ze Wei, Weidong Zan, Jianfei Zhao, Fu Hao, and Xiaoyan Chen. A real-time wind turbine blade damage detection method based on an improved yolov5 algorithm. In *International Conference on Image and Graphics*, pages 298–309. Springer, 2023.

[35] Chunsheng Hu, Yong Zhao, Fangjuan Cheng, and Zhiping Li. Multi-object detection algorithm in wind turbine nacelles based on improved yolox-nano. *Energies*, 16(3):1082, 2023.

Chapter 3

Pixel U-Net: An Improved Version of U-Net for Binary Segmentation of Wind Turbine Blades

Syed Zeeshan Rizvi, Mohsin Jamil, Weimin Huang

Department of Electrical and Computer Engineering, Memorial University of Newfoundland and Labrador

This chapter has been published in the Signal, Image and Video Processing journal.

The full text of the published paper is available to read at this link: https://rdcu.be/dKLtV.

3.1 Abstract

Wind turbine technicians face significant risks of fatal injuries, which can be mitigated by utilizing drones to capture images of wind turbine blades (WTBs) for remote inspection and maintenance. Different computer vision and image processing techniques can be applied to these captured drone images to automate the process of WTB inspection. However, the captured drone images consist of challenging backgrounds, due to which the WTB area needs to be extracted from the image as a pre-processing step. This paper introduces Pixel U-Net, an enhanced U-Net architecture tailored for binary segmentation of WTB images, improving segmentation accuracy with pixel shuffle and unshuffle operations. Evaluated against baseline U-Net architecture and its variations using the publicly available Blade30 dataset, Pixel U-Net achieves an average validation accuracy of 99.0% and surpasses existing methods in WTB image segmentation. Additionally, novel architectural variations, namely U-Net + HS-Block + Pixel Unshuffle and Pixel U-Net + Attention, have also been proposed in this study, which exhibit superior performance with an average training loss of 0.012 and an average testing accuracy of 98.3%, respectively. Qualitative comparisons of the results further highlight the efficacy of deep learning-based segmentation techniques in advancing wind turbine inspection and maintenance practices.

3.2 Introduction

Image segmentation is one of the most widely used techniques in computer vision. It involves separating different areas of interest in an image. Each segmented area corresponds to a specific object, shape, or category within the image. The technique has been widely used for numerous different applications, such as autonomous driving

[1], video surveillance [2], remote sensing using satellite imagery [3], and augmented reality [4].

The use of image segmentation for the identification of wind turbine blades (WTBs) has recently gained researchers' interest. Routine maintenance is required to ensure the smooth functioning of wind turbines. The manual inspection of wind turbines by human subjects is a risky job and can prove to be fatal as well [5]. With the emerging trends in technology, researchers have used drones to capture images of WTBs [6] for remote analysis. These captured images and datasets have paved the way for further studies on the automatic inspection of WTBs using deep learning methods. Image segmentation techniques allow for separating the blade area from the background, consisting of landscapes, varying colours of the sky, ocean, or infrastructure. These challenging backgrounds can confuse object detection models, resulting in false positive detections [7]. A good pre-processing technique to prepare the dataset for training object detectors is to segment the relevant area of WTBs from the image and feed only the area of interest to training models to automatically detect defects on the WTB surface.

Image segmentation techniques can be broadly classified into two categories – traditional and deep learning-based. The traditional methods consist of different image processing techniques, such as thresholding, region-based segmentation, edge segmentation, and clustering-based segmentation [8]. The more recent segmentation techniques fall under the umbrella of deep learning. These modern techniques use convolutional neural networks (CNNs) to derive feature information from images and identify patterns. One of the earliest deep-learning based image segmentation architectures was introduced in 2015 by the name of SegNet [9]. This architecture discussed

the idea of 'encoder' and 'decoder' in an architecture designed for image segmentation. In the same year, the popular U-Net architecture [10] was published, which was also built on the idea of encoder and decoder layers, connected together with skip connections to prevent information loss and generate better feature representation. Following this work, DeepLab [11] was introduced, which introduced the techniques of atrous convolutions and atrous spatial pyramid pooling to further increase the efficiency of segmentation models.

Segmenting the WTB area from a challenging background is a crucial pre-processing step in identifying defects on the WTB surface. Once a segmented WTB image is obtained, it can be mapped onto the original WTB image to extract only the relevant area, which can be used to train an object detector for surface defects. Referring to the application of image segmentation on WTBs, different variations of deep learning-based techniques have been utilized to segment WTB areas from images. In [12], different attention mechanisms are fused into the original U-Net architecture to improve feature extraction. In another work [13], the hierarchical split depth-wise separable convolution operations are utilized instead of the basic convolution operations in the U-Net architecture to improve segmentation performance. In this work,

Figure 3.1: Sample image set from Blade30 dataset

the convolution block utilizing hierarchical split depthwise separable convolution operations is referred to as the hierarchical split block (HS-Block). In this research, further improvements to the U-Net architecture are proposed. These improvements involve integrating pixel shuffle and pixel unshuffle operations [14] in the U-Net architecture instead of the transpose convolution and max pool operations, respectively. Three different architectural improvements are proposed using the pixel shuffle and pixel unshuffle operations, namely U-Net + HS-Block + Pixel Unshuffle, Pixel U-Net, and Pixel U-Net + Attention. The proposed architectures are compared with existing architectures, namely baseline U-Net, Attention U-Net [15], and U-Net + HS-Block. For the experiments in this work, the Blade30 dataset [6] is used, which is a publicly available dataset of WTB images. The Blade30 dataset consists of drone images of 33 different blades, making it a dataset rich in varying background information. A pair of a WTB image with its corresponding segmentation mask from the Blade30 dataset is provided in Figure 3.1. Five different splits of the Blade30 dataset were used in the experiments, and Pixel U-Net achieves the best accuracy on each validation set, whereas the other two proposed variations, U-Net + HS-Block + Pixel Unshuffle, and Pixel U-Net + Attention, achieve the best performance in training and testing, respectively.

The main contributions of this research work can be listed as follows:

1. A novel architecture, Pixel U-Net, that outperforms all other architectures under our study in validation performance for the binary segmentation of WTB images.

2. Two architectural variations of Pixel U-Net, namely U-Net + HS-Block + Pixel Unshuffle and Pixel U-Net + Attention, that prove that the inclusion of pixel shuffle and pixel unshuffle operations in architecture improves the overall performance of WTB segmentation.

This research work is divided into 4 sections, including this section. The next section will discuss the architectures under study, and the proposed architectural changes. The third section will include details regarding the experimental setup, pre-processing of the dataset, and discussion of our obtained results. Finally, the conclusion of this study will be provided in Section 3.5.

3.3 Methodology

In this study, various segmentation models are explored and expanded upon. The methodology section is divided into different sub-sections to aid understanding, each detailing components of the final proposed model, Pixel U-Net. These include an overview of the baseline U-Net model, the concept of attention in U-Net, the functioning of hierarchical split convolution operations, and finally, the architectural details of the proposed model – Pixel U-Net, and its proposed variations.

3.3.1 Baseline U-Net Model

The original U-Net architecture [10] can be visualized in the shape of the alphabet U, which can be symmetrically divided from the centre. The left side of the architecture represents the contraction path, which is also referred to as the encoder. This side of the U-Net receives an image as an input. The image is passed twice through a series of a convolution blocks, followed by a batch normalization layer to reduce over-fitting, followed by the rectified linear unit (ReLU) as an activation layer to introduce non-linearity in the training process. After this, the image is down-sampled using a maxpool operation. Using a 2x2 window for maxpooling reduces the spatial dimensions by half. The function of the contraction path is to extract low-level features

from the input image. This is done by reducing the spatial dimension and increasing the number of filters as the input passes through the described operations.

The right side of the U-Net architecture represents the expansion path, also called the decoder. This side follows the same sequence of convolution blocks, batch normalization, and activation layers described above. However, the maxpool operation is replaced by transpose convolution. In doing so, the spatial dimension of feature maps increases, whereas the number of filters decreases. A 2x2 transpose convolution is applied, which doubles the spatial dimensions and reduces the number of filters by half. In U-Net architecture, skip connections connect the encoder with the decoder. These skip connections concatenate filters of the same spatial dimensions from the encoder and decoder. This is particularly important during backpropagation. If the skip connections did not exist, the encoder would have received very small gradients during backpropagation, and learning low-level features would have been slow. Skip connections ensure that the model develops a good understanding of the low-level features and spatial information.

At the bottom of the U-shaped architecture, the encoder and decoder are joined by a bottleneck layer that extracts the most abstract and high-level features. The final layer of the decoder consists of a 1x1 convolution block, which gives an output feature map with the number of channels equal to the number of classes being predicted. Since we are working on a binary segmentation problem, the prediction, x, is passed through the sigmoid activation function to scale output predictions between 0 and 1. The formula for the sigmoid activation function is given below.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

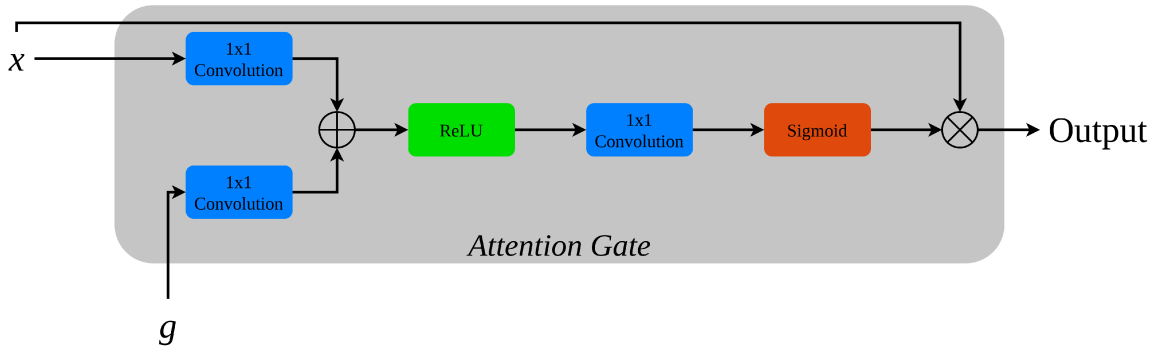


Figure 3.2: Attention gate takes two inputs: x comes from the corresponding level of the encoder, and g comes from the lower level of the decoder

3.3.2 Attention U-Net

The concept of attention in U-Net was introduced in [15]. Attention U-Net builds upon the basic U-Net architecture described in the earlier sub-section and includes an attention mechanism to the architecture. This attention mechanism allows the architecture to focus on areas of interest while performing segmentation. The technique is particularly useful in the binary segmentation of WTBs as we are only interested in the WTB area and not the challenging background.

Attention can be broadly divided into two categories – hard attention and soft attention. Attention U-Net uses the idea of soft attention, as different weights are assigned to different parts of the image. These weights help in identifying the regions of higher importance for the architecture. Regarding the architectural details of Attention U-Net, the attention block is placed at the end of the skip connections from the encoder to the decoder. The use of soft attention on the skip connection helps suppress the features of less interest. A detailed representation of the attention block is provided in Figure 3.2. The attention block takes in two inputs, g and x. The feature map from the previous lowest level in the decoder of the network is represented by g, and the feature map from the corresponding level in the encoder is represented

by x. Since g comes from a deep part of the whole architecture, it is rich in feature information, whereas x is rich in spatial information since it comes from a relatively earlier part of the network. The attention block uses spatial and feature information and concatenates them after passing through 1x1 dimensional filters. This helps in further increasing the weights of relevant regions on the image, and further reducing the weights of irrelevant regions. The outputs are then passed through a ReLU activation function before passing through another filter of 1x1 dimension. The output from that filter is passed through the sigmoid activation function. Once the weights are in the range of 0-1, they are multiplied with the input x to determine the areas of interest in the image.

3.3.3 Functioning of HS-Block

In [13], an improved version of U-Net was proposed, built on hierarchical split depthwise separable convolutions, originally proposed by Yuan et al. [16]. A visualization of the hierarchical split operation is presented in Figure 3.3. Adding pointwise convolution operations before and after the hierarchical split depthwise separable convolution operation, and concatenating the output of this operation with the original input gives the HS-Block. The structuring of the HS-Block is presented in Figure 3.4.

In the proposed architecture of improved U-Net using HS-Block, the authors replace one of the series of convolution blocks, batch normalization, and ReLU activation with the HS-Block. The authors state that this helps extract better multi-scale feature representations. The HS-Block splits the input feature map into a number of groups. As shown in Figure 3.3, these groups go through a depth-wise separable convolution [17], which is a combination of depth-wise and point-wise convolution operations in the specific order. The depth-wise separable convolution operations have

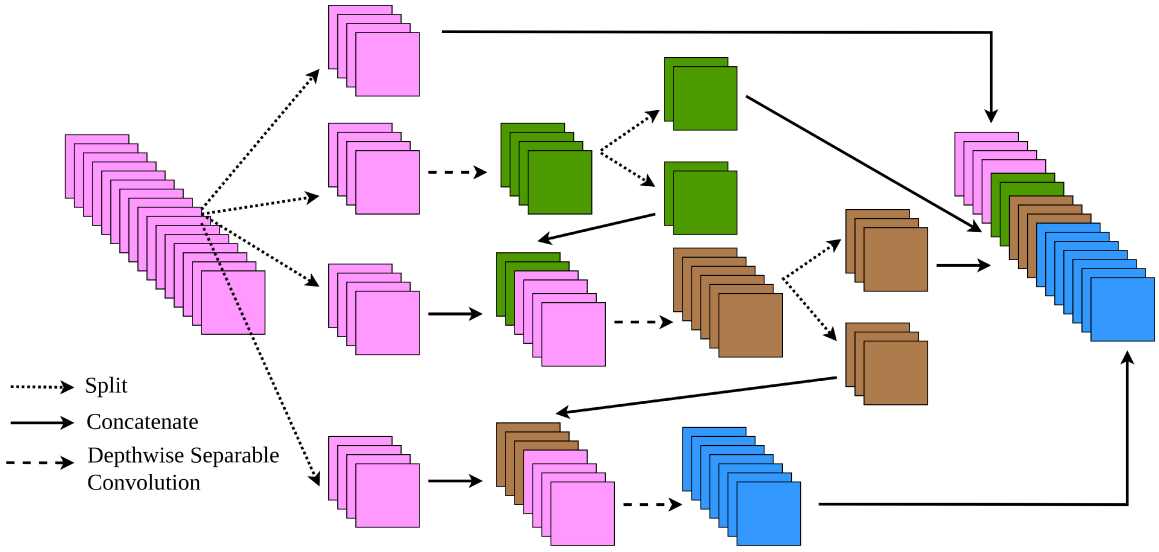


Figure 3.3: Heirarchical split depthwise separable convolution operation

been proven to reduce computation times [18] as fewer parameters need to be learned with these operations, compared to the traditional convolution operations. The outputs of each group are concatenated in a hierarchical fashion, where each output from the depth-wise separable convolution operation is concatenated with the features from the subsequent group.

3.3.4 Proposed Architecture of Pixel U-Net

In this research work, the different versions of U-Net described in the previous subsections are further built upon. Pixel shuffle [14] is a technique that is typically used in image super-resolution techniques to increase the size of feature maps by utilizing

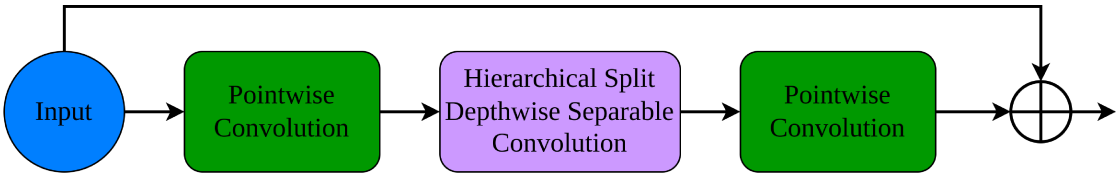


Figure 3.4: HS-Block

information present in different filters, thereby reducing the number of filters while doing so. The formulae by which the size of the resulting feature map and number of filters is calculated using the pixel shuffle and pixel unshuffle operations are given in equations (3.2) - (3.7). The width and height of the input feature map are given by W_{in} and H_{in}, respectively, whereas they are defined by W_{out} and H_{out}, respectively, for the output feature map. The number of filters in the input and output feature maps is given by N_{in} and $N_{filters}$, respectively. The term $factor$ is used to define the up-scaling factor in (3.2), (3.3) and (3.4), and the down-scaling factor in (3.5), (3.6) and (3.7). It can be observed from (3.2), (3.3) and (3.4) that pixel shuffle operation results in the same spatial dimensions of width and height as a normal upscaling function would have, however, this new information is derived from the existing filters instead of interpolation or convolution operations. Hence, the number of filters also changes, as shown in equation (3.4).

$$W_{out} = W_{in} \times factor \tag{3.2}$$

$$H_{out} = H_{in} \times factor \tag{3.3}$$

$$N_{filters} = \frac{N_{in}}{factor^2} \tag{3.4}$$

Likewise, pixel unshuffle refers to the opposite operation, in which the size of feature maps is reduced, and the information present in the feature map is used to create additional filters, thereby increasing the number of filters present. A comparison between max pool and pixel unshuffle operations to reach the same spatial dimensions is provided in Figure 3.5. It can be observed that while the max pool operation selects

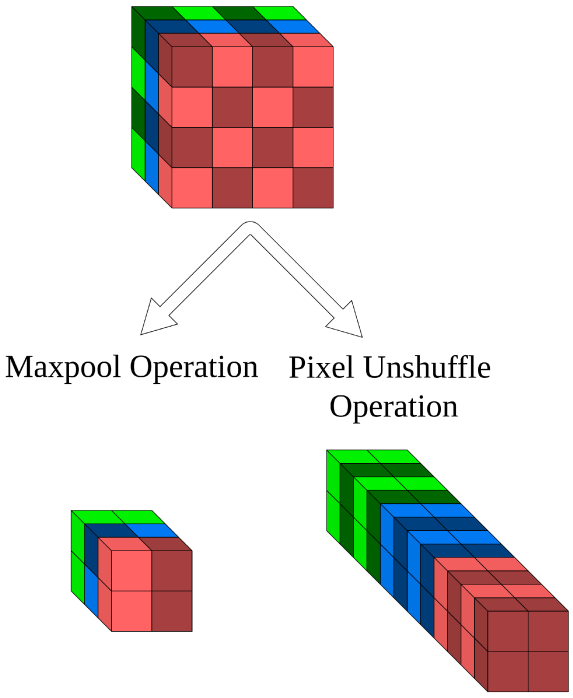


Figure 3.5: Downsampling by a factor of 2 using max pooling and pixel unshuffle operations. While some spatial information is lost after max pooling, all spatial information is preserved in pixel unshuffle operation by creating new filters

the feature with the highest weight in the 2x2 window and keeps information from all channels separate, the pixel unshuffle operation makes use of all spatial information that is present and combines it to form additional channels. The formulae by which the spatial dimensions and number of filters change in pixel unshuffle operations are given in equations (3.5), (3.6), and (3.7).

$$W_{out} = \frac{W_{in}}{factor} \tag{3.5}$$

$$H_{out} = \frac{H_{in}}{factor} \tag{3.6}$$

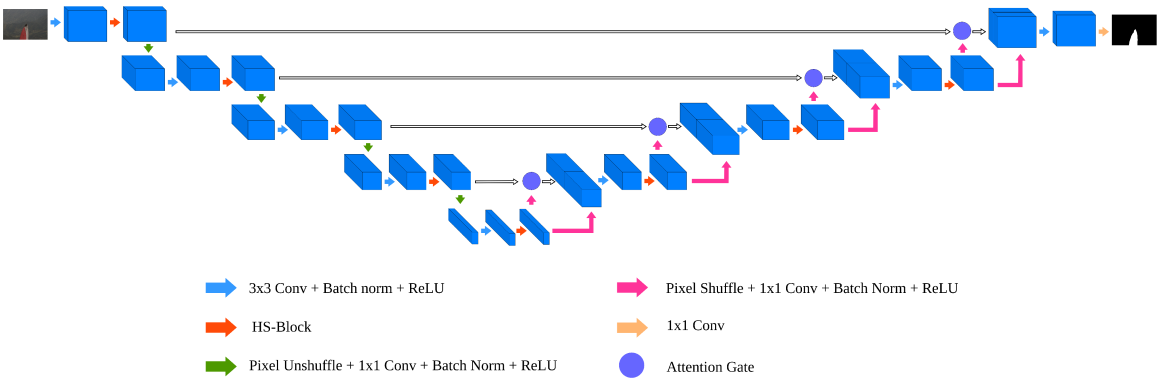


Figure 3.6: Proposed architecture for Pixel U-Net + Attention. The proposed architecture for Pixel U-Net can be visualized by removing the attention gate, and by further removing the pixel shuffle operation and replacing it with transpose convolution, the proposed architectural structure for U-Net + HS-Block + Pixel Unshuffle can be visualized

$$N_{filters} = N_{in} \times factor^2 \tag{3.7}$$

In the proposed improved architectures, different variations of pixel shuffle and unshuffle operations were tested. Pixel U-Net refers to the variation which replaces the max pool operations in the U-Net architecture with pixel unshuffle operation during encoding, and replaces the transpose convolution in the U-Net architecture with pixel shuffle operation during decoding. A 1x1 convolution operation follows each pixel unshuffle or pixel shuffle operation to keep the number of filters the same as expected in the original U-Net architecture. Pixel U-Net also replaces one convolution block in each encoding and decoding stage with the HS-Block mentioned earlier.

The different variations of Pixel U-Net tested consisted of only including the pixel unshuffle operation during the encoding process (referred to as U-Net + HS-Block + Pixel Unshuffle), including both pixel unshuffle and pixel shuffle operations during encoding and decoding respectively (referred to as Pixel U-Net), and including attention mechanism in the shortcut paths from the encoder to the decoder along with

pixel unshuffle and pixel shuffle operations (referred to as Pixel U-Net + Attention). Figure 3.6 presents a representation of the proposed novel architecture of Pixel U-Net + Attention.

The loss function that is used is a combination of binary cross entropy loss [19] and dice loss [20]. The formula for the dice score is provided in equation (3.8). Dice score is a commonly used metric for image segmentation tasks. In the numerator, the formula calculates the product between the predicted probability of pixels and the corresponding values in the ground truth mask (either 0 or 1). In the ground truth masks, the value of 0 represents the image background, and a value of 1 represents the WTB area. The product is multiplied by 2 to give more weight to the numerator, shifting more focus towards the area of interest only. The denominator is the sum of predicted pixel values and the ground truth mask values. A small value of ϵ is added to avoid division by zero. Adding the binary cross entropy loss with the dice loss gives the loss function provided in equation (3.9), which is used for training in this research work. Each ground truth pixel value is given by y, whereas x gives the predicted value. The total number of pixel values is given by N. The binary cross entropy loss is focused on the entire image, whereas the dice loss is focused on the region of overlap. Where the binary cross entropy loss ensures each pixel is accurately classified, the dice loss ensures the overlapping of the predicted foreground with the ground truth improves. This is what makes the combination of both these losses a good metric to calculate the training loss.

$$S_{dice} = \frac{2 \times \sum (y_{true} \times x_{pred})}{\sum y_{true} + \sum x_{pred} + \epsilon} \tag{3.8}$$

$$L_{BCE+Dice} = -\frac{1}{N}\sum_{i=0}^{N}[y_i \cdot \log x_i + (1-y_i) \cdot \log(1-x_i)] + (1 - S_{dice}) \tag{3.9}$$

3.4 Experiment and Results

3.4.1 Experimental Setup

For the experiments, 1240 images were picked from the Blade30 dataset [6] along with their corresponding binary segmentation masks. By randomly splitting the 1240 images into different train, validation and test folders, five different random arrangements were created. This was done to ensure the robustness of the obtained results by observing if all arrangements give the same collective outcome. Each arrangement is referred to as a *split* in this text. In each split, 72% images were used for training, 13% were used for validation, and the remaining 15% were kept separately to test trained models. It was important to ensure W_{in} and H_{in} in equations 3.5 and 3.6 were always divisible by 2 to avoid any loss of information during downscaling when running the pixel unshuffle operations. It was also important to maintain the original aspect ratio of the images. Due to these reasons, the images and the masks were reshaped to a size of 608 x 416. For all the trainings, a batch size of 4 was chosen, and training was continued for 30 epochs. Adam optimizer was used as the optimization algorithm as it offers efficient convergence and robustness while optimizing gradients during back-propagation. Some initial experiments were conducted using different learning rate schedulers, and cosine annealing gave the best results; therefore, it was chosen for all subsequent experiments. Cosine annealing uses the cosine function to adjust the learning rate over time. Initially, the learning rate was set at 0.0001. An

Table 3.1: Training, validation, and testing results

	Model	Avg Training Loss (last 10 epochs)	Avg Validation Dice Score (last 10 epochs)	Avg Testing Dice Score (best epoch)	Training Time
Split 1	U-Net	0.01937	0.9862	0.9835	11m 30s
	Attention U-Net	0.08559	0.9880	0.9834	12m 25s
	U-Net+HS-Block	0.01351	0.9913	0.9829	8m 38s
	U-Net+HS-Block+Pixel Unshuffle	**0.01056**	0.9895	0.9811	10m 13s
	Pixel U-Net	0.01207	**0.9916**	**0.9835**	10m 27s
	Pixel U-Net+Attention	0.01664	0.9905	0.9832	11m 41s
Split 2	U-Net	0.01956	0.9884	0.9809	11m 36s
	Attention U-Net	0.01912	0.9744	0.9803	12m 38s
	U-Net+HS-Block	0.01581	0.9902	0.9821	8m 57s
	U-Net+HS-Block+Pixel Unshuffle	0.01304	0.9884	0.9789	10m 18s
	Pixel U-Net	0.01786	**0.9912**	**0.9828**	10m 28s
	Pixel U-Net+Attention	**0.01237**	0.9905	0.9812	11m 35s
Split 3	U-Net	0.01630	0.9845	0.9823	15m 14s
	Attention U-Net	0.01201	0.9847	0.9221	12m 44s
	U-Net+HS-Block	0.01139	0.9901	**0.9839**	8m 58s
	U-Net+HS-Block+Pixel Unshuffle	**0.01098**	0.9867	0.9838	10m 11s
	Pixel U-Net	0.01157	**0.9905**	0.9822	11m 35s
	Pixel U-Net+Attention	0.01244	0.9899	0.9828	10m 25s
Split 4	U-Net	0.03332	0.9833	0.9831	11m 34s
	Attention U-Net	0.04754	0.9748	0.9828	12m 48s
	U-Net+HS-Block	0.01391	0.9800	0.9815	9m 3s
	U-Net+HS-Block+Pixel Unshuffle	**0.01292**	0.9826	0.9830	10m 22s
	Pixel U-Net	0.01523	**0.9858**	0.9825	10m 26s
	Pixel U-Net+Attention	0.01366	0.9793	**0.9840**	11m 42s
Split 5	U-Net	0.01572	0.9885	0.9822	11m 37s
	Attention U-Net	0.01271	0.9885	0.9815	12m 54s
	U-Net+HS-Block	0.01897	0.9874	**0.9826**	9m 11s
	U-Net+HS-Block+Pixel Unshuffle	0.01503	0.9901	0.9815	10m 23s
	Pixel U-Net	0.01388	**0.9902**	**0.9826**	10m 19s
	Pixel U-Net+Attention	**0.01059**	0.9899	0.9825	11m 32s

image scaling factor of 0.5 was chosen to speed up training, which implies that the architectures received images of size 304 x 208 as inputs for training.

To calculate the training loss, a combination of cross entropy and dice loss was used as described in equation (3.9). During validation, only the dice score was used as the metric to determine performance, the functioning of which was described in equation (3.8). A dice score of 1.0 reflects a perfect overlap between predicted and ground truth images, whereas a dice score of 0 reflects no overlap.

Different variations of the proposed architecture, along with already existing U-Net architectures, were trained and results were compared. The results of baseline U-Net, Attention U-Net and U-Net + HS-Block were compared with the proposed

architectural variations, namely U-Net + HS-Block+Pixel Unshuffle, Pixel U-Net, and Pixel U-Net + Attention. For graphical analysis, the results of training and validation are provided in Figure 3.7 of all splits. Table 3.1 presents a numerical representation of the results. For this table, the obtained numerical values of training loss and validation dice of the last 10 epochs were averaged in each split. Table 3.1 also presents the average training times on all splits for each architecture. To test the trained models, the respective weights for each model that generated the best results were identified. Using those weights, the dice score was calculated between each obtained prediction and its corresponding ground truth and then averaged for all test images in each split. These obtained testing results are also presented in Table 3.1. To summarize the analysis of all splits, results of each model from all splits were added and are presented in Table 3.2. Finally, some of the obtained predictions on the testing set are displayed in Figure 3.8 for a qualitative analysis.

3.4.2 Analysis of Results

We experimented with training the existing and proposed architectures for 20, 30 and 40 epochs. It was noted that training for 20 epochs resulted in underfitting and training for 40 epochs made it difficult to distinguish between the performance of all architectures. Training for 30 epochs offered a good balance and a good representation of the performance of all architectures. The graphs have been presented in this work

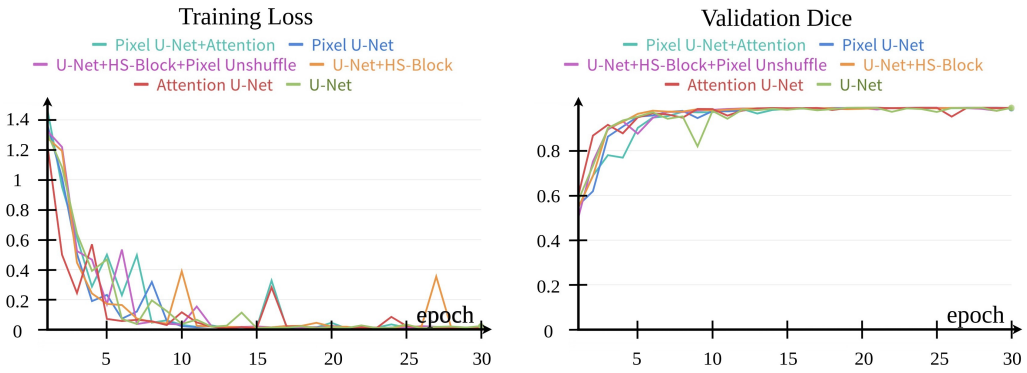
Table 3.2: Sum of results from 5 splits

	Training Loss	Validation Dice Score	Testing Dice Score
U-Net [10]	0.10427	4.9309	4.9120
Attention U-Net [15]	0.17697	4.9104	4.8501
U-Net+HS-Block [13]	0.07359	4.9390	4.9130
U-Net+HS-Block+Pixel Unshuffle	**0.06138**	4.9374	4.9083
Pixel U-Net	0.07061	**4.9493**	4.9136
Pixel U-Net+Attention	0.06570	4.9401	**4.9137**

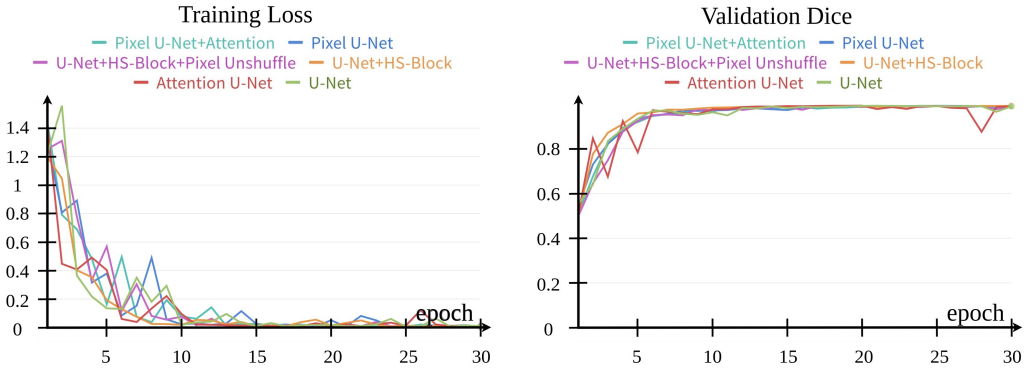
to analyze the progress of training and validation, and the presence of outliers for all architectures. From the graphs, it can be noted that with the exception of U-Net and Attention U-Net, all architectures generally follow a smooth training and validation progress. After about 20 epochs of training, the training and validation performance becomes difficult to distinguish from the graphs, necessitating a clearer representation of the results. This is achieved in Table 3.1 for the last 10 epochs.

The quantitative results in Table 3.1 show that the proposed architecture, Pixel U-Net, outperforms other architectures in the validation phase in all splits. The other variations of Pixel U-Net (U-Net + HS-Block + Pixel Unshuffle, and Pixel U-Net + Attention) demonstrate the best performance in training and testing. These results, along with the summarized analysis in Table 3.2, can provide promising grounds for more future research to conclude that the inclusion of pixel shuffle and pixel unshuffle operations in the U-Net architecture can help in improving binary segmentation accuracy for WTB images by preserving spatial information. The run times for all experiments presented in Table 3.1 make it evident that introducing HS-Block in U-Net architectures helps reduce training times, as discussed earlier in Section 3.3. However, introducing pixel shuffle and pixel unshuffle operations with the standard HS-Block increases the training times because of the additional convolution operations. However, comparing Pixel U-Net with baseline U-Net, and Pixel U-Net + Attention with Attention U-Net, we can notice that the training concludes in less time with the proposed architectural changes.

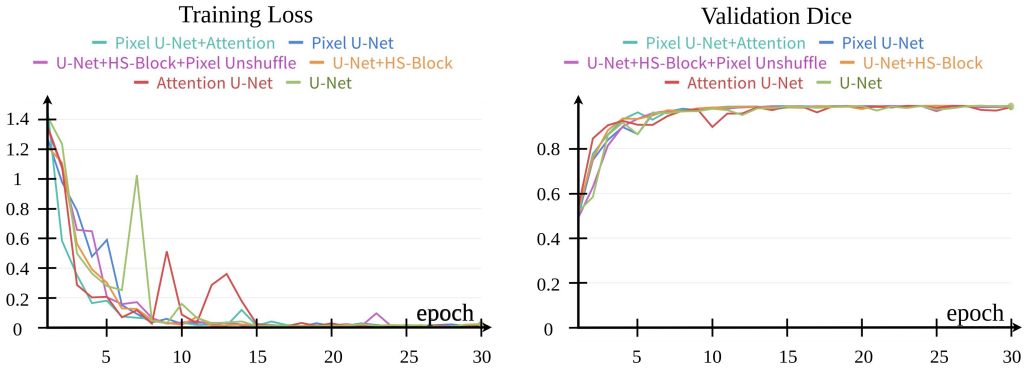
The qualitative analysis is presented in Figure 3.8. It can be observed that the creators of the Blade30 dataset considered the near-field wind turbines as areas of interest, and not the wind turbines which were present in far-field. The same pattern is learned by the existing and proposed architectures that are tested in this article.



Split 1

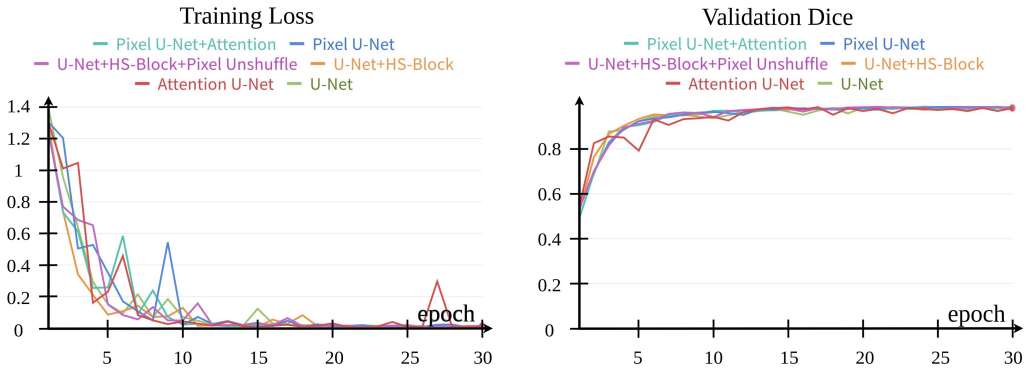


Split 2

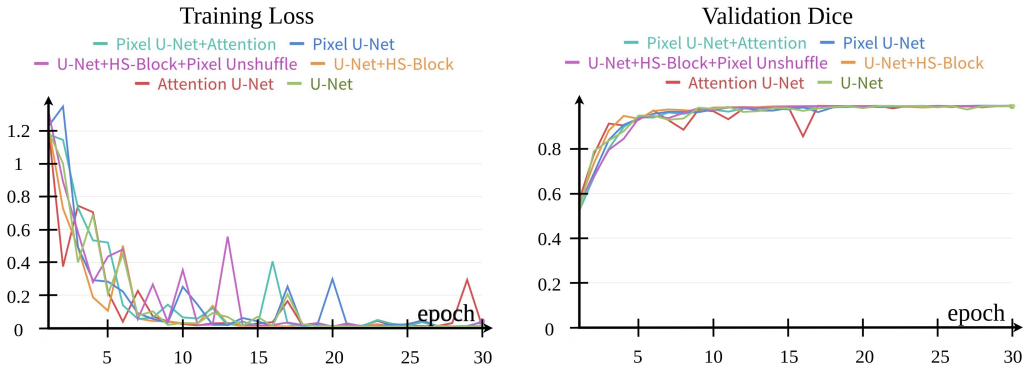


Split 3

Figure 3.7: Training and validation graphs for each split



Split 4



Split 5

Figure 3.7: Training and validation graphs for each split *(cont.)*

The figure displays better predictions of the proposed architectural changes than those made by existing architectures. It can be noted that introducing the pixel shuffle and pixel unshuffle operations in the U-Net architecture results in better identification of edges of the relevant area of WTB and hence better segmentation predictions on WTBs against challenging backgrounds.

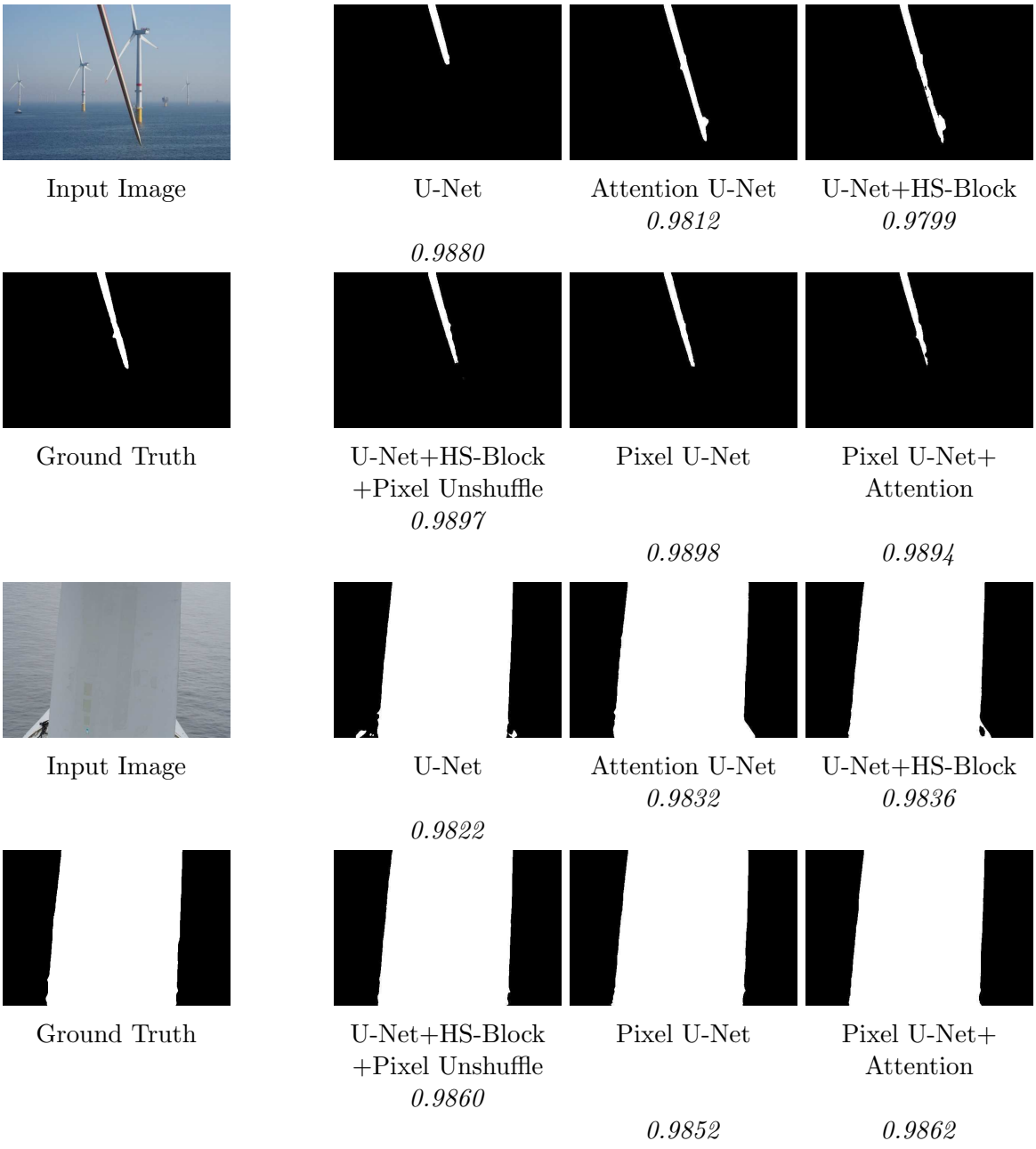


Figure 3.8: Qualitative analysis on test images. Dice score was calculated of predicted masks with ground truth masks, and have been provided for each architecture

3.5 Conclusion and Future Work

In this study, an improved version of the U-Net architecture that is tailored for binary segmentation of wind turbine blade (WTB) images was proposed. Through a comprehensive evaluation using the Blade30 dataset, it was demonstrated that Pixel U-Net outperforms existing architectures, namely baseline U-Net, Attention U-Net, and U-Net + HS-Block, in terms of validation performance for WTB image segmentation, with an average accuracy of 99.0% across 5 different splits.

This research additionally contributes novel architectural variations of Pixel U-Net, namely U-Net + HS-Block + Pixel Unshuffle, and Pixel U-Net + Attention, which outperformed the existing architectures in the study in both training and testing phases, respectively. These findings underscore the potential of pixel shuffle and pixel unshuffle operations in improving segmentation accuracy and performance for WTB image segmentation.

In our experiments with the Blade30 dataset, we use U-Net + HS-Block + Pixel Unshuffle to increase understanding of low-level features, Pixel U-Net to increase understanding of both low-level features and spatial information, and Pixel U-Net + Attention to infuse attention mechanisms to WTB image segmentation. Moving forward, future research could explore further optimizations and refinements to the Pixel U-Net architecture, investigate the applicability of the proposed techniques to other domains, and explore additional datasets to validate the generalizability of the findings from this research. By addressing these avenues, we can continue to drive innovation in WTB inspection methodologies and support the sustainable growth of renewable energy infrastructure.

3.6 Bibliography

[1] Der-Hau Lee and Jinn-Liang Liu. End-to-end deep learning of lane detection and path prediction for real-time autonomous driving. *Signal, Image and Video Processing*, 17(1):199–205, 2023.

[2] Hefeng Wu, Ning Liu, Xiaonan Luo, Jiawei Su, and Liangshi Chen. Real-time background subtraction-based video surveillance of people by integrating local texture patterns. *Signal, Image and Video Processing*, 8:665–676, 2014.

[3] Sai Lei, Mingming Lu, Jieqiong Lin, Xiaoqin Zhou, and Xuemei Yang. Remote sensing image denoising based on improved semi-soft threshold. *Signal, Image And Video Processing*, 15:73–81, 2021.

[4] Madjid Maidi, Fakhreddine Ababsa, Malik Mallem, and Marius Preda. Hybrid tracking system for robust fiducials registration in augmented reality. *Signal, Image and Video Processing*, 9:831–849, 2015.

[5] Sobhan Asian, Gürdal Ertek, Cagri Haksoz, Sena Pakter, and Soner Ulun. Wind turbine accidents: A data mining study. *IEEE Syst. J.*, 11(3):1567–1578, 2016.

[6] Cong Yang, Xun Liu, Hua Zhou, Yan Ke, and John See. Towards accurate image stitching for drone-based wind turbine blade inspection. *Renew. Energy*, 203:267–279, 2023.

[7] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *Eur. Conf. Comput. Vis.*, pages 340–353. Springer, 2012.

[8] B Basavaprasad and S Hegadi Ravindra. A survey on traditional and graph theoretical techniques for image segmentation. *Int. J. Comput. Appl*, 975:8887, 2014.

[9] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv:1505.07293*, 2015.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Med. Image. Comput. Comput. Assist. Interv.–MICCAI 2015: 18th Int. Conf., Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[11] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern. Anal. Mach. Intell.*, 40(4):834–848, 2017.

[12] Long Wang, Jinxu Yang, Chao Huang, and Xiong Luo. An improved u-net model for segmenting wind turbines from uav-taken images. *IEEE Sens. Lett.*, 6(7):1–4, 2022.

[13] Junfeng Yu, Yunze He, Hao Liu, Fan Zhang, Jie Li, Gaosen Sun, Xiaofei Zhang, Ruizhen Yang, Pan Wang, and Hongjin Wang. An improved u-net model for infrared image segmentation of wind turbine blade. *IEEE Sens. J.*, 23(2):1318–1327, 2022.

[14] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1874–1883, 2016.

[15] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard

Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

[16] Pengcheng Yuan, Shufei Lin, Cheng Cui, Yuning Du, Ruoyu Guo, Dongliang He, Errui Ding, and Shumin Han. Hs-resnet: Hierarchical-split block on convolutional neural network. *arXiv preprint arXiv:2010.07621*, 2020.

[17] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1251–1258, 2017.

[18] Anisha Pal, Shourya Jaiswal, Swarnendu Ghosh, Nibaran Das, and Mita Nasipuri. Segfast: A faster squeezenet based semantic image segmentation technique using depth-wise separable convolutions. In *Proc. 11th Indian Conf. Comput. Vis., Graph. Image Process.*, pages 1–7, 2018.

[19] Usha Ruby and Vamsidhar Yendapalli. Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(10), 2020.

[20] Toufique A Soomro, Ahmed J Afifi, Junbin Gao, Olaf Hellwich, Manoranjan Paul, and Lihong Zheng. Strided u-net model: Retinal vessels segmentation using dice loss. In *2018 Digit. Image Comput.: Tech. Appl. (DICTA)*, pages 1–8. IEEE, 2018.

Chapter 4

Enhanced Defect Detection on Wind Turbine Blades Using Binary Segmentation Masks and YOLO

Syed Zeeshan Rizvi, Mohsin Jamil, Weimin Huang

Department of Electrical and Computer Engineering, Memorial University of Newfoundland and Labrador

This chapter has been published in the Computers and Electrical Engineering journal.

4.1 Abstract

The detection of defects and contamination on wind turbine blades (WTBs) using neural networks is a rapidly growing research area. One problem with publicly available datasets of wind turbine blade images is that the image size is quite large, which

results in slower training of object detectors. A new pre-processing pipeline is proposed in this research, in which the Blade30 dataset is used to create a new masked version of the dataset that effectively takes about 50% less disk space. To provide a thorough comparative analysis and to prove the robustness of the proposed approach, the masked version of the dataset is trained with different YOLO object detectors, namely YOLOv5, YOLOv6, YOLOv7 and YOLOv8. The obtained results reflect that the time taken for the proposed pre-processing approach and completing 300 epochs of training with YOLO object detectors helps in cutting down around 1-2 hours compared to training with the original version of the dataset. Validation results using the proposed technique demonstrate a gain in mean average precision (mAP) scores ranging from 0.8-7.9% across different versions of YOLO compared with the results on the original dataset, and a gain in mAP scores ranging from 1.1-22.7% compared with different existing methods for WTB defect detection. Testing with the trained weights obtained through the masked version of the dataset shows a significant gain in mAP scores, ranging from 27.3-33.7%.

4.2 Introduction

The recent advancements in research in computer vision have opened doors to numerous new possibilities. Object detection continues to be one of the most valuable applications in computer vision. It refers to identifying objects of interest in an image and drawing a bounding box around them to localize them. Object detection has been effectively used to solve many real-world problems, such as person detection in challenging scenarios [1], brain tumour diagnosis [2], crowd counting [3], disease detection on crops [4], and video surveillance [5], to name a few. One such area in which object detection is gaining the interest of computer vision researchers is damage detection

Figure 4.1: Sample image from Blade30 dataset showing the WTB and the respective segmentation mask

on wind turbine blades (WTB).

Wind turbines are a source of clean and renewable energy. Since wind turbines are subject to strong winds and adverse weather conditions, they require routine maintenance for continued efficient performance. Different methods are used to perform non-destructive testing of wind turbine blades. Some of these methods include ultrasonic testing [6], tap testing [7], and infrared thermography [8]. Visual inspection is one of the most common methods to identify structural damage on wind turbines [9]. The use of drones has become a popular method for visually inspecting wind turbines [10]. WTB inspection using drones can significantly reduce turbine downtime from 1.5 hours (using conventional methods) to an average of just 20 minutes [11]. Using drones for WTB inspection also mitigates the risks of human injury or even death. A study suggests that most deaths occur during the construction or maintenance of wind turbines [12]. Furthermore, since drones can capture and store images of WTBs, they also help maintain a better record of the condition of the turbine, which can be revisited at any time.

Researchers have used various image processing techniques to get a better representation of wind turbine images captured using drones. Experiments performed

in [13] reconstruct a 3D model of a WTB using multiple images. A technique was proposed in [14] to measure strain on wind turbines using drone images. In another work presented in [15], images of WTBs were synthetically generated to address the problem of motion blur when drones are used. With continued research in computer vision, captured drone images have been used to train computer vision algorithms to identify damages on WTBs automatically [16]. However, the availability of WTB image datasets in the public domain remains a problem for researchers. Recently, the Blade30 dataset [11] was made publicly available, which offers a collection of annotated images for the purpose of identifying different types of defects and contamination on WTB. As shown in Figure 4.1, Blade30 consists of images of WTBs and binary segmentation masks of corresponding images. One major problem with WTB images captured using drones is the size of the images. Each image present in the Blade30 dataset is roughly of the size 5400 x 3600. Images of a higher resolution result in slow training of object detection algorithms [17]. Moreover, datasets with images of higher resolution also need a good amount of storage space on the hard drive. It is also worth mentioning that the presence of a challenging background can increase the possibility of false positive detections by the object detector algorithm. One such case for this is shown in Figure 4.2.

This research proposes a novel pre-processing pipeline that helps in creating a new version of WTB images. The Blade30 dataset has been used in this research to demonstrate the effectiveness of the proposed approach. Through different experiments, it is proven in this work that the new version of the Blade30 dataset prepared using the proposed approach significantly reduces the storage space required, takes less time to complete training, and achieves better training results compared to the original version of the Blade30 dataset. In this work, a comparative analysis of the performance of different You Only Look Once (YOLO) models for defect detection

Figure 4.2: Background in images results in the detection of false positives

on WTB surfaces is also presented.

The main contributions of this research are summarized below:

1. A novel pre-processing pipeline using binary masks and edges of WTBs to generate a new version of WTB image dataset that optimizes storage space, training speed, and accuracy.

2. A comparative analysis using different versions of YOLO to demonstrate the performance of YOLO object detectors on WTB images.

The next sections in this paper are organized as follows. Section 4.3 discusses the related work performed in the domains of object detection and WTB defect detection. Section 4.4 presents the complete methodology. Section 4.5 details the experimental design, and Section 4.6 discusses the obtained results. Finally, Section 4.7 constitutes the conclusion.

4.3 Related Work

The detection of defects on WTB using computer vision techniques falls under the category of object detection. For this specific problem, objects are classified as different types of damages, defects, or contaminations on surfaces of WTBs. Reviewing the literature on object detection algorithms to understand their development with time is vital. Due to this reason, this section is divided into two parts. The first part gives a brief overview of the literature in the object detection domain, and the second part discusses work performed using deep learning (DL) to process WTB images and identify damages on them automatically.

4.3.1 Progress of Object Detection Algorithms

Before intensive research in the DL domain for object detection, conventional algorithms were used. For these algorithms, features had to be extracted before passing them to a classification method. One of the earliest of these algorithms is Viola-Jones object detection [18], which proposed the approach of making integral images and then performing detection using Haar features. Building on the research in object detection and recognition, in 2004, Scale Invariant Feature Transform (SIFT) [19] was used to extract features from an image and compare them with an existing database of features to recognize the object present in the image. Following this, in 2006, Histogram of Oriented Gradient (HOG) [20] approach was introduced, which presented a new approach to object detection using support vector machine as a classifier. The extraction of features from the frequency domain for object detection has also been studied in various research works [21] [22]. Deep learning methods can

perform object detection using neural networks without the need for extracting features as a pre-processing step. These algorithms use convolutional neural networks (CNNs) to perform computation on images. Within deep learning, object detection algorithms can be broadly divided into two categories, two-stage, and one-stage detectors. Region-based convolutional neural network (RCNN) [23] is one of the earliest two-stage detectors proposed. It extracts different proposed regions for an object in an image and then passes each of these proposed regions to a CNN to make predictions for the class of the object. This makes the object detection process slow, and it cannot be used for real-time object detection. To tackle the issue, Fast RCNN [24] and Faster RCNN [25] were proposed as improved versions of RCNN. However, these algorithms continued to be additions to the two-stage detectors.

You Only Look Once, or YOLO [26], was the first single-stage detector proposed in 2016. This algorithm could predict all the objects present in an image by passing the image to a CNN just once, which defines the algorithm's name. YOLO allows faster object detection and can perform well on real-time applications. Many improved versions of YOLO have been proposed over recent years. Focusing on the advancements made in the YOLO series since 2020, YOLOv5 [27] streamlined the object detection pipeline by introducing different model variants for balancing speed and performance, advanced data augmentation techniques, and auto anchor calculation. In 2022, YOLOv6 [28] was introduced, which was focused on industrial-grade performance and offered further improvements in the architecture. The seventh version of YOLO, referred to as YOLOv7 [29], was presented in the same year. YOLOv7 uses extended efficient layer aggregation network to improve model's efficiency. With further improvements in the architecture, YOLOv8 [30] was released in 2023 that offered state-of-the-art performance on benchmark datasets for object detection.

4.3.2 Image Processing and Defect Detection on WTB Images

As shown in Figure 4.2, the background in WTB images results in the detection of false positives on images. As discussed in the next section in equation (4.9), false positives directly affect the precision of the object detector. Due to this reason, work has been done on pre-processing WTB images to extract only the relevant area. A method proposed in [31] uses "regression crop" to crop the surface of WTB out of images and perform detection only across the surface. A technique presented in [32] gives an improved version of U-Net that can effectively segment WTB from infrared images using hierarchical split depthwise separable convolution operations. Another improved version of U-Net specifically to segment WTB area was proposed in [33], which made use of an efficient channel attention network and point-wise spatial attention network. In terms of segmentation of WTB images, state-of-the-art performance was reported in another improved version of U-Net called Pixel U-Net [34], in which the max pooling and transpose convolution operations in the U-Net architecture were replaced by pixel unshuffle and pixel shuffle operations.

Since WTB surfaces are large and the whole blade surface cannot be captured in high definition in a single drone image, researchers have proposed image stitching techniques to generate accurate images of the complete WTB image. One such technique was proposed in [35], in which authors perform different geometrical calculations to align images belonging to the same WTB and create a panoramic image. Another technique was proposed in [11] that features a two-level process that combines coarse-grained initial alignment using blade edges and drone-blade distances with fine-grained adjustments optimized by texture and shape losses. Other than extracting relevant areas from images of WTBs, work has also been done to improve

the quality of WTB images captured in varying lighting conditions. In [36], a method that uses cartoon and texture maps to enhance the illumination of WTB images is proposed.

In addition to research on pre-processing techniques used on WTB images, researchers have experimented with DL to build networks that can be trained for the specific application of identifying defects on WTBs. A method was proposed in [37] that can identify small defects on WTB using an improved version of YOLOv5 to perform the task. In another work [38], an attention and feature-balanced version of YOLO (AFB-YOLO) is proposed, which helps in gathering more feature information for improved detection. In this improved architecture for YOLOv5s (where *s* stands for small), the authors also replace the intersection over union (IoU) loss with efficient IoU (EIoU) loss for bounding box calculations. In another work [39], the authors introduced an improved version of YOLOv5s called multivariate information YOLO (MI-YOLO) for defect detection on WTBs. In this architecture, authors replace some parts from the original YOLOv5s backbone and also infuse the MobileNetv3 backbone in the network. In their ablation study, they also extend their workings to replace the IoU loss with Alpha-IoU loss for bounding box calculations. In [40], another improvement to YOLOv5s was proposed called MC-YOLO. This modified architecture replaced the complete original backbone of YOLOv5s with the MobileNetv3 backbone. For real-time monitoring of WTB, the inference speed is very important. This is addressed in [41], where the researchers propose an improved version of YOLOX-Nano which greatly increases the inference speed.

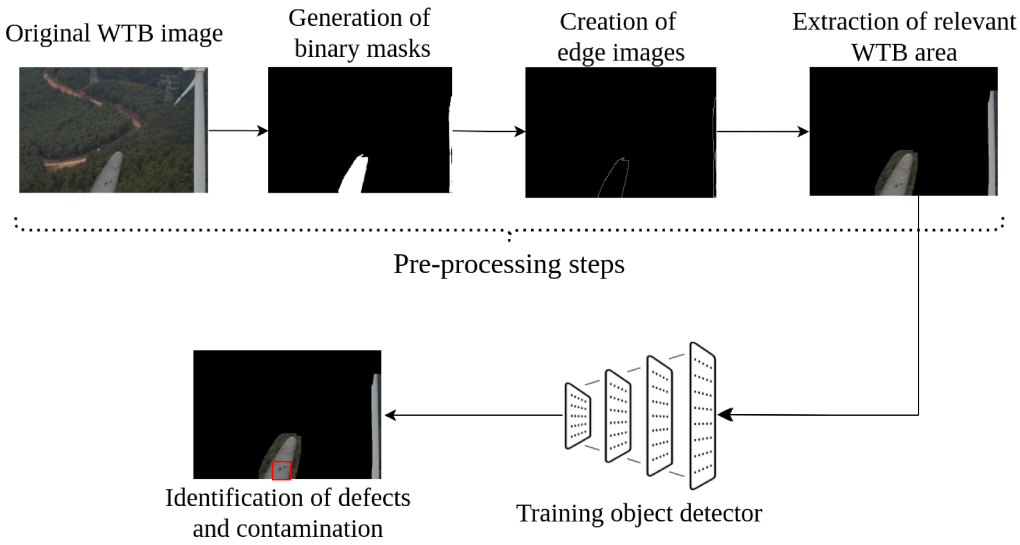


Figure 4.3: Proposed training pipeline including the pre-processing steps

4.4 Methodology

An overview of the complete training process is given in Figure 4.3. The method proposed in this research can be divided into three simple steps – generation of binary segmentation masks using Pixel U-Net, extraction of relevant WTB area, and object detection using YOLO. In the proposed pipeline for preparing the dataset for training, the captured drone images of WTBs are passed through a segmentation model called Pixel U-Net to generate binary segmentation masks. Using the binary segmentation masks, the proposed method creates a masked version of the original dataset. This dataset is then used to train different version of YOLO to identify defects and contamination on WTBs. The methodology is explained in detail below.

4.4.1 Generation of Binary Segmentation Masks using Pixel U-Net

Pixel U-Net [34] is an improved version of the baseline U-Net segmentation model. The architectural improvements proposed in Pixel U-Net tailor it for the task of binary segmentation of WTBs. The authors also prove with different experiments that Pixel U-Net outperforms baseline U-Net for the task of binary segmentation of WTBs. Pixel U-Net builds further upon the concept of hierarchical split depthwise separable convolutions in U-Net architecture proposed in [32]. In the proposed architecture of Pixel U-Net, the max pooling operations from the original U-Net architecture are replaced with pixel unshuffle operations in the encoding process, and the transpose convolution operations are replaced by the pixel shuffle operations during the decoding process. This helps the segmentation model develop a better understanding of low-level features and spatial information.

Since the goal is to segment the WTB area into foreground and everything else into background, the problem is treated as a binary segmentation problem and the model is optimized using binary cross-entropy loss. The dice score is also used for the calculation of loss during training process. Equation (4.1) presents the formula for the calculation of dice score, in which the ground truth pixel values of the binary mask are represented by y_{true} and the predicted pixel values of the binary mask are represented by x_{pred}. To avoid division by zero, a small value of ϵ is added to the fraction. The complete loss function in equation (4.2) is simply a sum of the binary cross entropy loss and dice loss. In this formula, the ground truth pixel values are represented by y_i, and the predicted pixel values are represented by x_i, whereas the total number of pixels is given by N. The training process for Pixel U-Net works towards optimizing the performance of binary segmentation using this loss function,

Figure 4.4: Actual WTB image (left), ground truth binary segmentation mask (middle), binary segmentation mask generated by Pixel U-Net (right)

and the trained model is then able to generate binary segmentation masks of WTBs such as the one shown in Figure 4.4 (right).

$$S_{dice} = \frac{2 * \sum (y_{true} * x_{pred})}{\sum y_{true} + \sum x_{pred} + \epsilon} \tag{4.1}$$

$$L_{BCE+Dice} = -\frac{1}{N} \sum_{i=0}^{N} [y_i \cdot \log x_i + (1 - y_i) \cdot \log(1 - x_i)] + (1 - S_{dice}) \tag{4.2}$$

4.4.2 Extraction of Relevant WTB area

Figure 4.5: Binary segmentation mask (left) and the obtained edge image from it using Canny edge detector (right)

Once the binary segmentation masks are obtained, they are used to create the

mapped version of WTB image. First, the WTB image and mask are resized to the same size. A new image is created, with pixel values P. The pixel values in the WTB image are represented by I, whereas the pixel values in binary segmentation mask are represented by S. The following formula is used to fill in pixel values in the new image.

$$P = \begin{cases} I, & \text{if } S = 1 \\ 0, & \text{if } S = 0 \end{cases} \tag{4.3}$$

This results in the image shown in Fig. 4.6 (middle). However, the problem with images like these is that due to an absence of background close to the WTB edges, the object detection algorithm fails to learn defects or contamination on the edges. To solve this problem, the edge images of WTB blades are obtained. The edges of WTB can be accurately extracted using the binary segmentation masks. Extraction of edges of WTB is vital for the method proposed in this sub-section. Experiments were done with different edge detectors, and it was found that the Canny edge detector [42] gives the best results on the binary segmentation masks. Experiments were also done to apply the Canny edge detector directly on the original WTB images to extract the edges of WTBs, but the challenging and diverse nature of backgrounds in WTB

Figure 4.6: Original WTB image from Blade30 dataset (left), masked version using binary segmentation mask only (middle), masked version using binary segmentation mask and edge image (right)

images made it difficult to obtain consistent edges directly on original WTB images. Fig. 4.5 shows an obtained edge image from the binary segmentation mask. The Canny edge detector works by first smoothing the mask using a Gaussian filter [43]. The smoothing of the mask is done to remove noise from the image. After smoothing, the mask is convolved with Sobel filter [44]. The Sobel operator identifies edges both in the x-direction (K_x) and y-direction (K_y) by using the filter operations below.

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{4.4}$$

The convolved result of the mask with Sobel operators in the x-direction and y-direction is represented by I_x and I_y, respectively. These values collectively represent a gradient vector in the form of (I_x , I_y) that can be used to calculate the magnitude M and direction D using the following equations.

$$M = \sqrt{I_x^2 + I_y^2} \tag{4.5}$$

$$D = \arctan\left(\frac{I_y}{I_x}\right) \tag{4.6}$$

These obtained results are used to perform non-maximum suppression, which suppresses repetitive edge lines and gives a clean edge image, as shown in Fig. 4.5.

Once both the binary segmentation mask and its corresponding edge images are obtained, they are used to extract not just the WTB surface, but also a clean area around the edge of WTB. A sliding window of a fixed size f (this size is set to be 1/10th of the width or height of the image, whichever is greater) is applied on the

edge image. The window moves through the whole edge image with pixel values E. The fxf window of pixel values around the pixel value I in the original WTB image is represented by I_w. The pixel values of the new image are represented by P and the fxf window of pixel values around P is denoted by P_w. The pixel values of binary segmentation mask are represented by S. This modified approach to obtain pixel values P for the new image can be represented by the formula below.

$$P_w = \begin{cases} I_w, & \text{if } E = 1 \\ I, & \text{if } S = 1 \\ 0, & \text{if } S = 0 \end{cases} \tag{4.7}$$

Applying this methodology results in mapped images such as the one shown in Fig. 4.6 (right), which are used in further experimentation.

4.4.3 Object Detection using YOLO

The original version of the Blade30 dataset and the new version of the dataset are trained for 9 different types of defects and contamination using different versions of YOLO, namely YOLOv5, YOLOv6, YOLOv7 and YOLOv8. In terms of architecture, YOLO model architectures are divided into three basic parts. First is the backbone, in which features are extracted from the image. Second is the neck, in which features are combined to learn patterns in the dataset. And third is the head, which predicts the class and location of objects in the image. Once these predictions are made, the result is further refined by the algorithm by applying the non-maximum suppression technique. This technique uses the intersection over union (IoU) value of all predicted boxes, and if this value is greater than a set threshold value, the bounding box with a

lower confidence score is suppressed. The IoU is defined by the formula given below. A and B define two different bounding boxes for which the IoU is to be calculated.

$$IoU = \frac{(A \cap B)}{(A \cup B)} \tag{4.8}$$

To measure the accuracy of the training process, the algorithm uses mean average precision (mAP) metric. Precision is defined as the ratio of correct predictions of a class to the total number of predictions for that class. Recall is the ratio of correct predictions to the total number of objects from that class present. The formulae for precision and recall are given in equation (4.9) and (4.10), respectively. Correct predictions are referred to as true positives (TP) and true negatives (TN), whereas incorrect predictions are referred to as false positives (FP) and false negatives (FN). Once the precision and recall values are calculated, the F1-score relates both metrics with the formula given in equation (4.11).

$$\text{Precision } = \frac{TP}{TP + FP} \tag{4.9}$$

$$\text{Recall } = \frac{TP}{TP + FN} \tag{4.10}$$

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.11}$$

Average precision (AP) is defined as the area under the precision-recall curve and is calculated separately for each class. As the name defines, mAP is the mean of all AP values obtained for each class. In the formula in equation 4.12, N represents the total number of classes.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{4.12}$$

The performance of YOLO algorithms is compared in this research using obtained mAP scores at different IoU thresholds on the original, as well as the masked version of the Blade30 dataset. For example, $mAP_{0.50}$ represents the mAP scores obtained when IoU threshold was set at 0.50. The metric $mAP_{0.50:0.95}$ gives a better overview of performance as it reflects the average obtained by setting IoU thresholds at intervals of 0.05, starting at 0.50 and ending at 0.95. The performance comparison of the proposed methodology in this work with existing techniques proposed by researchers for the same problem is done using mAP, precision, recall, and F1 metrics.

4.5 Experimental Setup

4.5.1 Pre-processing of Dataset

The Blade30 dataset consists of a total of 1302 high-resolution images, along with their binary segmentation masks. For the first step, Pixel U-Net was trained to generate binary segmentation masks. A total of 1054 image-mask pairs were used for this purpose, out of which 15% images were used for validation. The discarded images were removed from the dataset based on inaccurate ground truth binary segmentation masks. To achieve quicker training, the images were resized to 608 x 416, and a scaling factor of 0.5 was used. A batch size of 4 was used, and the initial learning rate was kept at 0.0001. The algorithm was trained for 30 epochs. The training and validation results from this step are provided in Figure 4.7. At the end of the training process, the model is able to achieve a training loss value of 0.00789 and a validation dice score

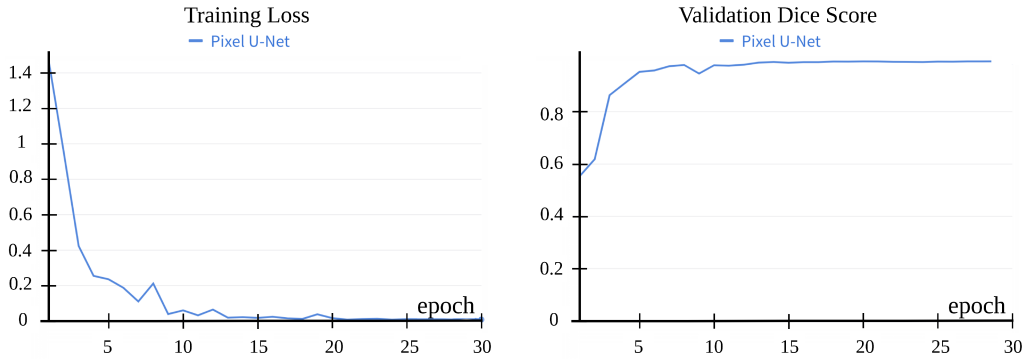


Figure 4.7: Training and validation results for binary segmentation using Pixel U-Net of 0.992.

The Blade30 dataset contains 263 labelled images, and the rest can be treated as background images for training an object detector. The labelled images consist of different classes representing defects and contaminations on the WTB surface. For the training of object detectors, a total of 307 images were chosen from the Blade30 dataset. The next step consisted of generating the binary masks for these images using the trained segmentation model. For this purpose, the images were first down-scaled from their original size of 5456 x 3632 to 608 x 416. The images were then passed through the trained model. Once binary segmentation masks were obtained, the method described in sub-section 4.4.2 was applied, which consisted of generating edge images and then the relevant area from WTB image, as shown in Figure 4.6 (right). To map the relevant blade area onto the original-sized images, lower-sized images were resized to the original size, and again changed to a binary mask based on wherever pixel values were present. Using the same methodology presented in equation (4.3), the relevant area was then obtained from the original-sized images using the new mask. A detailed breakdown of the time taken for each of these steps is given in Table 4.1.

Table 4.1: Breakdown of time for pre-processing steps

Pre-processing Step	Time (s)
Training Pixel U-Net	627.00
Resizing images to 608 x 416	6.53
Running inference on trained model	7.58
Obtaining edge images	0.76
Extracting relevant WTB area	2.99
Mapping onto original size	11.93
Total Time	656.79

4.5.2 Training Object Detectors

Following the pre-processing steps in the previous sub-section, two versions of the Blade30 dataset were obtained for experiments. The first version consisted of original images from the Blade30 dataset, whereas the second version consisted of masked images from the Blade30 dataset, as shown in Figure 4.6 (right). Before starting the training, both versions of the dataset were refined to remove under-represented classes. In some cases, under-represented classes were merged into visually similar classes. Moreover, classes which did not have good diversity were also removed. A total of 9 classes remained to train the YOLO object detectors for comparative analysis. Both versions of the dataset consisted of 249 labelled images with good class representation, and 58 background images, making a total of 307 images. The datasets were split into 73%, 17% and 10% for training, validation and testing respectively. It was ensured that both versions of the dataset had exactly the same images in the train, validation and test set. The experiments were performed on NVIDIA RTX 2000 Ada, and 8 GB of video RAM (VRAM) was used. A batch size of 8 was used for training, and a batch size of 4 was used for validation. For all YOLO models, the respective pre-trained weights on the COCO dataset were used, and the models were further trained for 300 epochs on both versions of the Blade30 dataset. All training experiments were performed using the MMYOLO toolbox by OpenMMLab [45]. A complete flow

chart of the proposed methodology in this research, including all pre-processing steps required during training and testing, is presented in Figure 4.8.

The proposed methodology was also compared for performance with recently proposed techniques for defect detection on wind turbine blades by researchers, namely AFB-YOLO [38], MI-YOLO [39], and MC-YOLO [40]. For this comparison, training with these methods was also performed for 300 epochs with the same dataset split described above, and using pre-trained weights on the COCO dataset. For better adaptability with the pre-trained weights, all the methods were trained using the complete intersection over union (CIoU) loss function [23] for the bounding box loss. It is to be noted that the authors of [38] originally worked with the efficient intersection over union (EIoU) loss function [46], but the experiments conducted in this research gave a better performance for all architectures with the CIoU loss when pre-trained weights were used, and hence it was eventually used for comparison. The training progression for all methods is provided in Figure 4.9. To make the graphical comparison clear, a time-weighted exponential moving average with a factor of 0.7 was used to smooth the precision and recall graphs. The original results without smoothing can be seen at the backdrop of the precision and recall graphs. The actual final values obtained at epoch 300 from these graphs for the performance metrics are given in Table 4.2. Using the respective weights from the best-performing epoch of each method, testing was conducted and the obtained results are provided in Table 4.3.

Table 4.2: Comparison of training performance of proposed methodology with existing methods

	YOLOv5 Original [27]	**AFB-YOLO [38]**	**MI-YOLO [39]**	**MC-YOLO [40]**	**YOLOv5 Masked**
Precision ↑	**0.861**	0.820	0.654	0.562	0.850
Recall ↑	0.841	0.810	0.827	0.660	**0.863**
F1 ↑	0.851	0.815	0.730	0.607	**0.856**
$mAP_{0.50}$ ↑	0.864	0.846	0.820	0.710	**0.886**
$mAP_{0.50:0.95}$ ↑	0.620	0.571	0.501	0.404	**0.631**

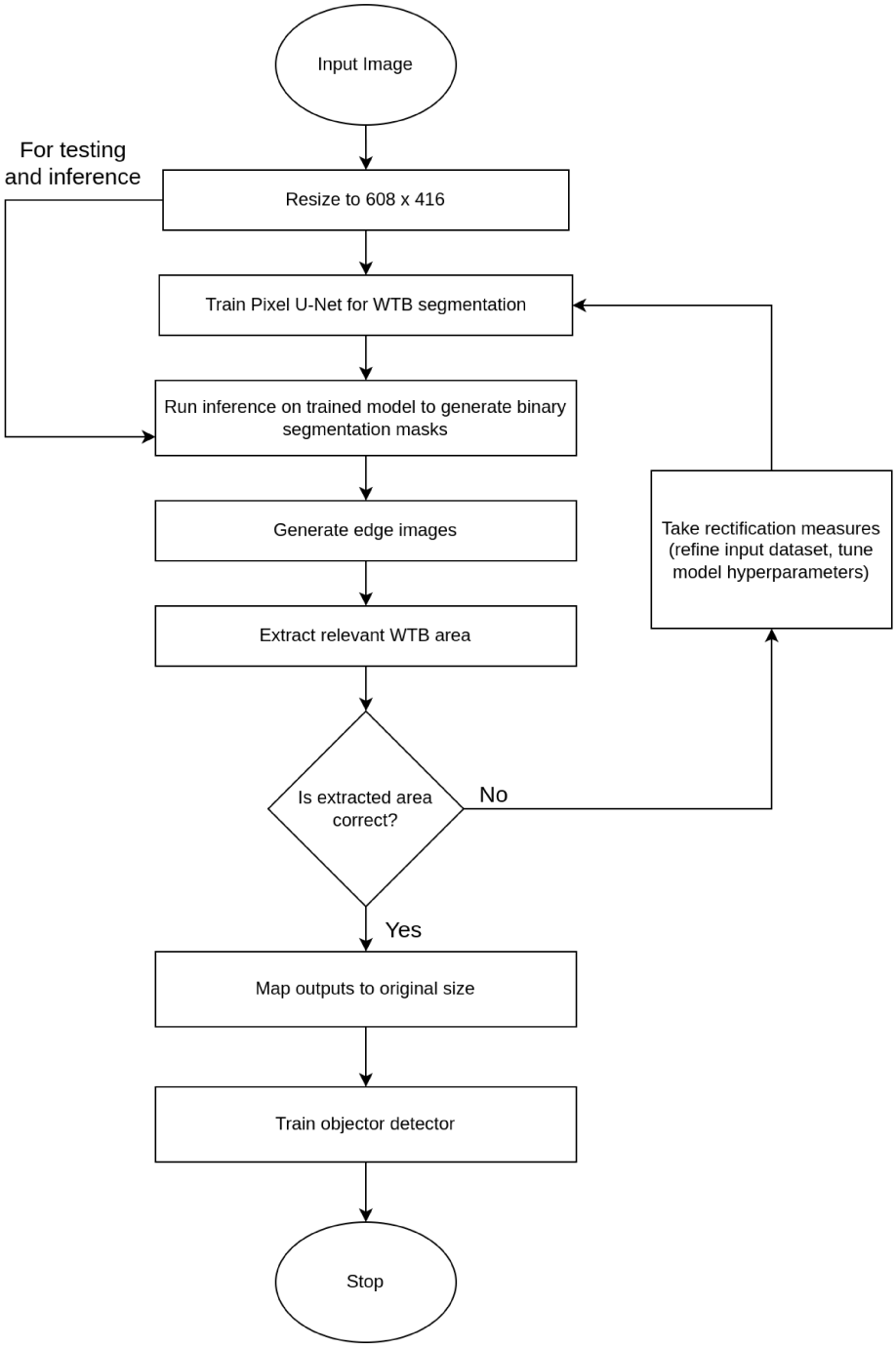


Figure 4.8: Detailed flowchart for proposed methodology

Table 4.3: Comparison of testing performance of proposed methodology with existing methods

	YOLOv5 Original [27]	**AFB-YOLO [38]**	**MI-YOLO [39]**	**MC-YOLO [40]**	**YOLOv5 Masked**
Precision ↑	**0.356**	0.320	0.287	0.343	0.652
Recall ↑	0.426	0.604	**0.750**	0.688	**0.674**
F1 ↑	0.388	0.418	0.415	0.458	**0.663**
$mAP_{0.50}$ ↑	0.739	0.647	0.579	0.603	**0.838**
$mAP_{0.50:0.95}$ ↑	0.315	0.379	0.350	0.356	**0.652**

4.6 Analysis of Results

Table 4.4: Storage space and training time comparison

		Original Version	**Masked Version**	**Time Difference (h)**
Dataset size (MBs) ↓		1200	589	-
Training time (hours) ↓	YOLOv5s [27]	9.183	7.642	1.541
	YOLOv6s [28]	9.860	8.686	1.174
	YOLOv7t [29]	12.20	10.10	2.100
	YOLOv8s [30]	9.397	8.156	1.241

In the tables in this manuscript, a down arrow represents that a lower value of the metric is better, whereas an up arrow suggests that a higher value of the metric is better. From Table 4.4, it can be observed that preparing the WTB image dataset using the proposed methodology results in a significant reduction in the required storage space, which leads to quicker training. The total time taken by the pre-processing steps required for this methodology is given in Table 4.1, which equates to 0.182 hours. Adding this time to the training time, it can be noted that the overall time for the proposed methodology still helps in reducing training time by about 1-2 hours.

It was also observed that the proposed data pre-processing methodology helps significantly improve the mAP scores obtained at the end of the training process. As presented in Table 4.5, almost every experiment noted a gain in mAP when the masked version of the dataset was used. This increase in performance is attributed to the object detector being able to focus only on the area of interest using masked images,

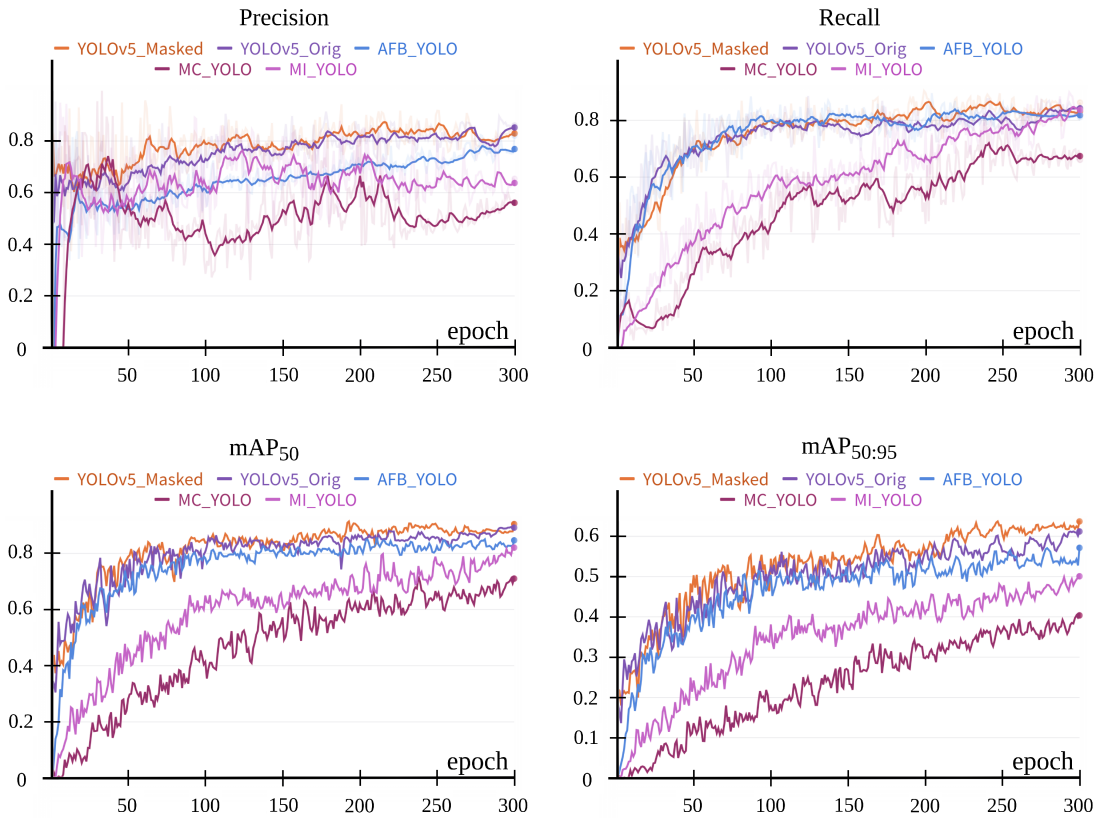


Figure 4.9: Comparison with existing methods

which reduces the possibility of errors and facilitates better learning. To determine the statistical significance of the obtained mAP scores (specifically the $mAP_{0.50:0.95}$ metric) on the masked version of the dataset across all YOLO models in this study, the results were compared with the mAP scores on the original version of the dataset using Wilcoxon Signed Rank. The significance level for this test was set at 0.05, which means that if the probability values fall below this threshold, the difference between both metrics is considered significant. Table 4.6 gives the obtained probability values with each YOLO model. The values are well below the significance level, which reflects that the results with the masked version of the dataset are statistically significant. A measure of the computational complexity of the YOLO models used in this research is provided in Table 4.7.

Table 4.5: Comparison of obtained mAP values from experiments

		Original Version	**Masked Version**	**Gain in mAP**
$mAP_{0.50} \uparrow$	YOLOv5s	0.864	0.886	2.2%
	YOLOv6s	0.929	0.902	-2.7%
	YOLOv7t	0.819	0.828	0.9%
	YOLOv8s	0.778	0.857	7.9%
$mAP_{0.75} \uparrow$	YOLOv5s	0.752	0.779	2.7%
	YOLOv6s	0.781	0.801	2.0%
	YOLOv7t	0.693	0.702	0.9%
	YOLOv8s	0.594	0.634	4.0%
$mAP_{0.50:0.95} \uparrow$	YOLOv5s	0.620	0.631	1.1%
	YOLOv6s	0.624	0.651	2.7%
	YOLOv7t	0.557	0.565	0.8%
	YOLOv8s	0.523	0.571	4.8%

Table 4.6: Wilcoxon Signed Test results

YOLOv5s	**YOLOv6s**	**YOLOv7t**	**YOLOv8s**
0.01160	0.00001	0.00494	0.00048

A comparative analysis of the training progression of different YOLO models on both versions of the dataset is given in Figure 4.10. It can be seen that the training progresses normally with little to no difference on both versions of the dataset. Comparing the performance of the object detectors, YOLOv5s achieved the lowest loss value in 300 epochs of training. YOLOv5s was hence chosen to perform testing on the two versions of the dataset. The weights from the epoch that obtained the best mAP scores were chosen for each version of the dataset, respectively. The quantitative results obtained through testing are presented in Table 4.3. In this table, YOLOv5 Original refers to training and testing using the original dataset, whereas YOLOv5

Table 4.7: Computational complexity of YOLO models

	YOLOv5s	**YOLOv6s**	**YOLOv7t**	**YOLOv8s**
Model Parameters	7.04M	17.19M	6.04M	11.14M
FLOPS	20.38G	56.03G	16.86G	36.56G

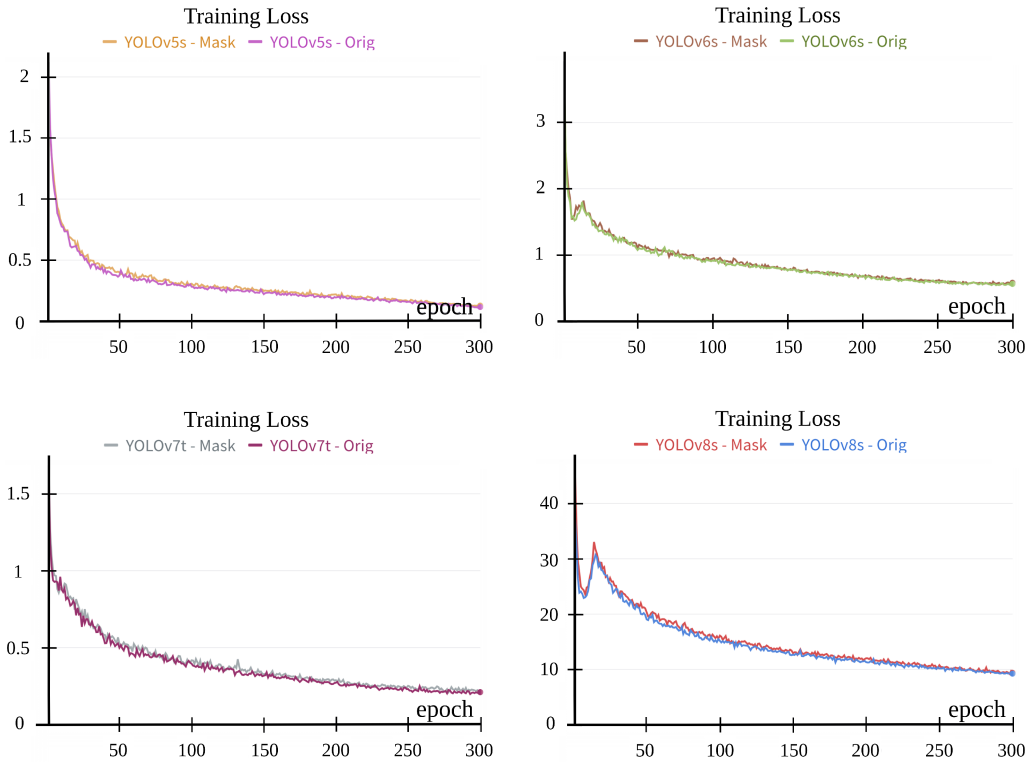


Figure 4.10: Training loss

Masked refers to the proposed methodology of training and testing using masked images of the dataset. Significant improvements in the F1 and mAP scores were noted using the masked version of the dataset, which have also been compared with existing methods that proposed different improvements to the YOLOv5s architecture. The improvements in results using the proposed methodology are also prominent in Figure 4.9 and Table 4.2, which provide a quantitative analysis of the training progression with different proposed methods for WTB defect detection. It can be noted that

Table 4.8: Comparison of testing time using YOLOv5s

	Time for pre-processing (s)	Time for inference (s)	Total time (s)
Original Version	-	2.17	2.17
Masked Version	0.01	2.03	2.04

shifting focus towards dataset pre-processing instead of architectural improvements helps in improving the efficiency of the object detector.

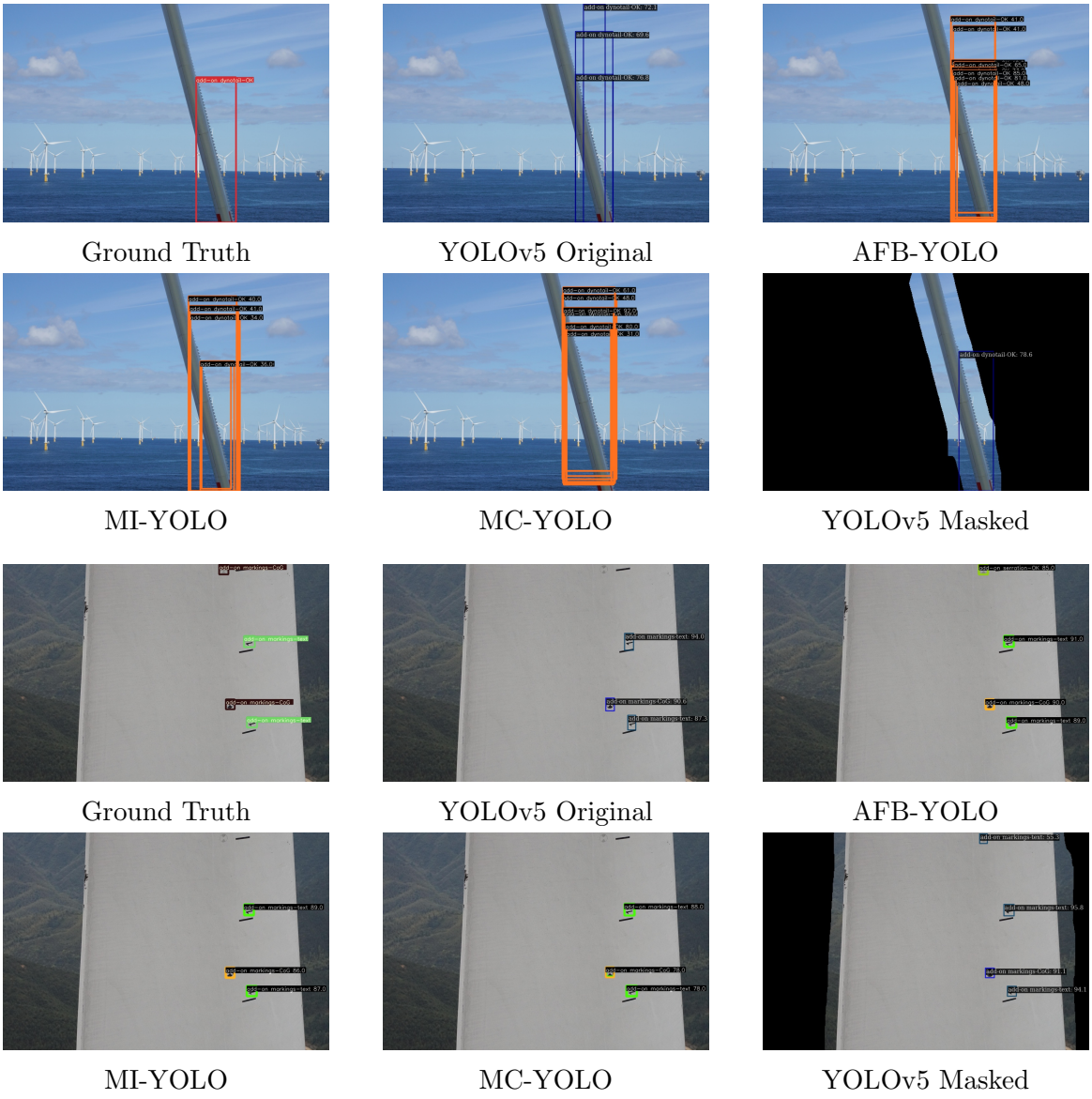


Figure 4.11: Qualitative comparison of detections of proposed method with existing methods for WTB defect detection

Using the proposed methodology, the time taken per image was also reduced. For testing, the time taken by the first pre-processing step in Table 4.1 can be ignored (Training Pixel U-Net), and the remaining time durations are added. This gives a

total time of 29.79 seconds. A total of 307 images were used for the steps in Table 4.1, giving the time taken per image for the remaining steps to be 0.01 seconds. Adding this time to the inference time per image as shown in Table 4.8, the total time for testing using the proposed method is still lower than the time for testing using the original version of the dataset.

For a qualitative comparison, some of the results from the testing phase are presented in Figure 4.11. The common observation that can be made from the results is that given the relevant area for detection, the model trained and tested on the masked version of the dataset (YOLOv5 Masked) gives a better performance than the different models that were compared. The confidence threshold was set to be 0.3 for all the models during testing, and the results display that YOLOv5 Masked has a better detection accuracy with robust performance compared to other methods. Using other methods resulted in repeated detections and lower confidence in detections. This can be attributed to the challenging nature of backgrounds in WTB images during training, which were avoided by YOLOv5 Masked due to the masking of the dataset.

4.7 Conclusion

A novel pre-processing technique to aid with identifying defects and contamination on the surface of WTB is proposed in this work. The binary segmentation masks of WTB images are used to create edge images in this work, which are then collectively used to extract only the area of interest from the WTB image, resulting in a mapped version of the original dataset. Through extensive experiments with different versions of YOLO, namely YOLOv5, YOLOv6, YOLOv7 and YOLOv8, it is proven that the proposed technique helps in reducing the storage space required for WTB image datasets to

half, and reduces training time for object detectors by about 1-2 hours. The obtained results with the proposed method during training demonstrate a gain in mAP scores ranging from 0.8-7.9% across different versions of YOLO, and a gain in mAP scores ranging from 1.1-22.7% across different existing methods. A significant gain in mAP scores was noted with the proposed method during testing, ranging from 27.3-33.7%.

Data Availability Statement

The original Blade30 dataset is publicly available at this link: `https://github.com/cong-yang/Blade30`. The refined version of the Blade30 dataset that has been used in this research consisting of 307 images with 9 classes is publicly available at this link: `https://universe.roboflow.com/memorial-university-of-newfoundland-y9kcb/blade-damage-detection-v2/dataset/7`. The codes and the results associated with this manuscript can be made available to the reader upon request.

4.8 Bibliography

[1] C Jayavarthini and C Malathy. An improved deep-layer architecture for real-time end-to-end person recognition system. *Computers & Electrical Engineering*, 96:107550, 2021.

[2] Dillip Ranjan Nayak, Neelamadhab Padhy, Pradeep Kumar Mallick, and Ashish Singh. A deep autoencoder approach for detection of brain tumor images. *Computers and Electrical Engineering*, 102:108238, 2022.

[3] Lixian Yuan, Yandong Chen, Hefeng Wu, Wentao Wan, and Pei Chen. Crowd counting via localization guided transformer. *Computers and Electrical Engineering*, 104:108430, 2022.

[4] Nirmal Raj, Senthil Perumal, Sanjay Singla, Girish Kumar Sharma, Shamimul Qamar, and A Prabhu Chakkaravarthy. Computer aided agriculture development for crop disease detection by segmentation and classification using deep learning architectures. *Computers and Electrical Engineering*, 103:108357, 2022.

[5] Sannasi Ganapathy and Devansh Ajmera. An intelligent video surveillance system for detecting the vehicles on road using refined yolov4. *Computers and Electrical Engineering*, 113:109036, 2024.

[6] Melody A Drewry and George A Georgiou. A review of ndt techniques for wind turbines. *Insight-Non-Destructive Testing and Condition Monitoring*, 49(3):137–141, 2007.

[7] Anne Jüngert. Damage detection in wind turbine blades using two different acoustic techniques. *The NDT Database & Journal (NDT)*, 2075, 2008.

[8] Bin Yang, Lixin Zhang, Weidong Zhang, and Yibo Ai. Non-destructive testing of wind turbine blades using an infrared thermography: A review. In *2013 International Conference on Materials for Renewable Energy and Environment*, volume 1, pages 407–410. IEEE, 2013.

[9] Kyungil Kong, Kirsten Dyer, Christopher Payne, Ian Hamerton, and Paul M Weaver. Progress and trends in damage detection methods, maintenance, and data-driven monitoring of wind turbine blades–a review. *Renewable Energy Focus*, 44:390–412, 2023.

[10] Mahmood Shafiee, Zeyu Zhou, Luyao Mei, Fateme Dinmohammadi, Jackson Karama, and David Flynn. Unmanned aerial drones for inspection of offshore wind turbines: A mission-critical failure analysis. *Robotics*, 10(1):26, 2021.

[11] Cong Yang, Xun Liu, Hua Zhou, Yan Ke, and John See. Towards accurate image stitching for drone-based wind turbine blade inspection. *Renew. Energy*, 203:267–279, 2023.

[12] Sobhan Asian, Gürdal Ertek, Cagri Haksoz, Sena Pakter, and Soner Ulun. Wind turbine accidents: A data mining study. *IEEE Syst. J.*, 11(3):1567–1578, 2016.

[13] D Zhang, K Burnham, L Mcdonald, C Macleod, G Dobie, R Summan, and G Pierce. Remote inspection of wind turbine blades using uav with photogrammetry payload. In *56th Annual British Conference of Non-Destructive Testing-NDT 2017*, 2017.

[14] Ashim Khadka, Arash Afshar, Mehrdad Zadeh, and Javad Baqersad. Strain monitoring of wind turbines using a semi-autonomous drone. *Wind Engineering*, 46(1):296–307, 2022.

[15] Yeping Peng, Zhen Tang, Genping Zhao, Guangzhong Cao, and Chao Wu. Motion blur removal for uav-based wind turbine blade images using synthetic datasets. *Remote Sensing*, 14(1):87, 2021.

[16] Long Wang and Zijun Zhang. Automatic detection of wind turbine blade surface cracks based on uav-taken images. *IEEE Transactions on Industrial Electronics*, 64(9):7293–7303, 2017.

[17] Sergio Saponara and Abdussalam Elhanashi. Impact of image resizing on deep learning detectors for training time and model performance. In *International*

Conference on Applications in Electronics Pervading Industry, Environment and Society, pages 10–17. Springer, 2021.

[18] Modesto Castrillón, Oscar Déniz, Daniel Hernández, and Javier Lorenzo. A comparison of face and facial feature detectors based on the viola–jones general object detection framework. *Machine Vision and Applications*, 22:481–494, 2011.

[19] Zahra Hossein-Nejad and Mehdi Nasri. An adaptive image registration method based on sift features and ransac transform. *Computers & Electrical Engineering*, 62:524–537, 2017.

[20] Nooshin Nabizadeh and Miroslav Kubat. Brain tumors detection and segmentation in mr images: Gabor wavelet vs. statistical features. *Computers & Electrical Engineering*, 45:286–301, 2015.

[21] M Ahsan, MA Based, J Haider, M Kowalski, et al. An intelligent system for automatic fingerprint identification using feature fusion by gabor filter and deep learning. *Computers and Electrical Engineering*, 95:107387, 2021.

[22] Haewon Byeon, Mohammad Shabaz, Kapil Shrivastava, Anjali Joshi, Ismail Keshta, Rajvardhan Oak, Pavitar Parkash Singh, and Mukesh Soni. Deep learning model to detect deceptive generative adversarial network generated images using multimedia forensic. *Computers and Electrical Engineering*, 113:109024, 2024.

[23] Hao Wang, Mengjiao Li, and Zhibo Wan. Rail surface defect detection based on improved mask r-cnn. *Computers and Electrical Engineering*, 102:108269, 2022.

[24] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[26] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[27] Glenn Jocher. Ultralytics yolov5, 2020.

[28] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, Yiduo Li, Bo Zhang, Yufei Liang, Linyuan Zhou, Xiaoming Xu, Xiangxiang Chu, Xiaoming Wei, and Xiaolin Wei. Yolov6: A single-stage object detection framework for industrial applications, 2022.

[29] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023.

[30] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.

[31] Weijie Zhou, Zijun Wang, Minshu Zhang, and Liwen Wang. Wind turbine actual defects detection based on visible and infrared image fusion. *IEEE Transactions on Instrumentation and Measurement*, 72:1–8, 2023.

[32] Junfeng Yu, Yunze He, Hao Liu, Fan Zhang, Jie Li, Gaosen Sun, Xiaofei Zhang, Ruizhen Yang, Pan Wang, and Hongjin Wang. An improved u-net model for infrared image segmentation of wind turbine blade. *IEEE Sens. J.*, 23(2):1318–1327, 2022.

[33] Long Wang, Jinxu Yang, Chao Huang, and Xiong Luo. An improved u-net model for segmenting wind turbines from uav-taken images. *IEEE Sens. Lett.*, 6(7):1–4, 2022.

[34] Syed Zeeshan Rizvi, Mohsin Jamil, and Weimin Huang. Pixel u-net: an improved version of u-net for binary segmentation of wind turbine blades. *Signal, Image and Video Processing*, 18:6299–6307, 2024.

[35] Junfeng Yu, Yunze He, Fan Zhang, Gaosen Sun, Yuejun Hou, Hao Liu, Jie Li, Ruizhen Yang, and Hongjin Wang. An infrared image stitching method for wind turbine blade using uav flight data and u-net. *IEEE Sensors Journal*, 23(8):8727–8736, 2023.

[36] Yeping Peng, Weijiang Wang, Zhen Tang, Guangzhong Cao, and Shengxi Zhou. Non-uniform illumination image enhancement for surface damage detection of wind turbine blades. *Mechanical Systems and Signal Processing*, 170:108797, 2022.

[37] Rui Zhang and Chuanbo Wen. Sod-yolo: a small target defect detection algorithm for wind turbine blades based on improved yolov5. *Advanced Theory and Simulations*, 5(7):2100631, 2022.

[38] Xiukang Ran, Shang Zhang, Hengtao Wang, and Zhaoyang Zhang. An improved algorithm for wind turbine blade defect detection. *IEEE Access*, 10:122171–122181, 2022.

[39] Zhu Xiaoxun, Hang Xinyu, Gao Xiaoxia, Yang Xing, Xu Zixu, Wang Yu, and Liu Huaxin. Research on crack detection method of wind turbine blade based on a deep learning method. *Applied Energy*, 328:120241, 2022.

[40] Zhiming Zhang, Chaoyi Dong, Ze Wei, Weidong Zan, Jianfei Zhao, Fu Hao, and Xiaoyan Chen. A real-time wind turbine blade damage detection method based on an improved yolov5 algorithm. In *International Conference on Image and Graphics*, pages 298–309. Springer, 2023.

[41] Chunsheng Hu, Yong Zhao, Fangjuan Cheng, and Zhiping Li. Multi-object detection algorithm in wind turbine nacelles based on improved yolox-nano. *Energies*, 16(3):1082, 2023.

[42] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[43] Guang Deng and LW Cahill. An adaptive gaussian filter for noise reduction and edge detection. In *1993 IEEE conference record nuclear science symposium and medical imaging conference*, pages 1615–1619. IEEE, 1993.

[44] O Rebecca Vincent, Olusegun Folorunso, et al. A descriptive algorithm for sobel image edge detection. In *Proceedings of informing science & IT education conference (InSITE)*, volume 40, pages 97–107, 2009.

[45] MMYOLO Contributors. MMYOLO: OpenMMLab YOLO series toolbox and benchmark. `https://github.com/open-mmlab/mmyolo`, 2022.

[46] Zheng Wang, Yan Liu, Siyuan Duan, and Hongguang Pan. An efficient detection of non-standard miner behavior using improved yolov8. *Computers and Electrical Engineering*, 112:109021, 2023.

Chapter 5

Conclusion

In this thesis work, an efficient and robust methodology was developed to detect defects and contamination on WTBs automatically. In this methodology, the WTB area is effectively segmented from the original drone image using Pixel U-Net, which is a novel architecture proposed as part of this thesis work. Pixel U-Net is an improved version of the U-Net architecture. The Pixel U-Net architecture replaces the max pooling and transpose convolution operations in the original U-Net architecture with pixel unshuffle and pixel shuffle operations respectively. Experiments in this work demonstrate a better segmentation performance on WTBs compared to the performance of existing architectures. As part of ablation studies carried out in this work, two different variations of Pixel U-Net have also been proposed, namely HS-Block + Pixel Unshuffle, and Pixel U-Net + Attention.

As part of the next step in the proposed methodology, image-processing techniques have been used to extract the relevant WTB area once the binary segmentation masks are obtained. In addition to the actual WTB area, the relevant WTB area has been defined to be some part around the edges of WTB as well. This ensures a better

detection of defects and contamination on the edges of WTBs. Once the relevant area is obtained, different versions of YOLO were trained to identify defect and contamination on WTBs. Through experiments, it has been proven in this research that the proposed methodology not only outperforms existing methods in WTB defect detection, but also helps in reducing the size of the dataset by about half, and reduces training time by about 1-2 hours.

5.1 Future Work

As WTB defect detection is a relatively new area in the domain of computer vision, there is a scarcity of WTB datasets that are publicly available. All the experiments in this research were carried out using the publicly available Blade30 dataset. To help future researchers build up on this work, the refined version of the Blade30 dataset used in this research, with the exact training, validation, and test splits, has been made available at this link: *https://shorturl.at/VdPOn*. As more WTB datasets become publicly available, a good future direction will be to validate the results presented in this thesis work on other WTB datasets as well. To assist future researchers in the domain of WTB defect detection, if resources permit, it would also be very helpful to gather a new WTB dataset captured using drones, annotate it for defects and contamination, and make it publicly available for the research community to use.

Additionally, while this thesis focused on WTBs, the potential of Pixel U-Net in other domains remains largely untapped. Applying it across different contexts could validate its universal effectiveness and lead to new adaptations for various segmentation tasks. Therefore, extending this research to other domains is a promising direction for future work.