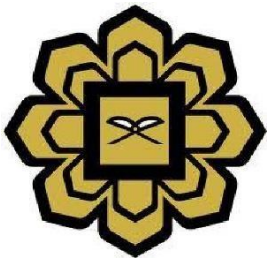
AUTOMATED VISUAL INSPECTION OF SHIP HULL SURFACE USING AI

BOUFARES HASSAN EL MEHDI 1717231



DEPARTMENT OF MECHATRONICS ENGINEERING KULLIYYAH OF ENGINEERING
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
JUNE 2022

Automated Visual Inspection of Ship Hull Surface using AI

BOUFARES HASSAN EL MEHDI 1717231

Supervisor: Assoc. Prof. Dr. Zulkifli Bin Zainal Abidin

A REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR A DEGREE OF BACHELOR OF ENGINEERING (B.ENG) IN MECHATRONICS ENGINEERING

ABSTRACT

Biological fouling was and still an issue that affects the marine vessel hulls. It creates a high resistance between the water and the ship hull which hinders the movement of the ship, leading to an increment of the overall cost. Previously, Biological fouling were detected through divers that check the vessel hull status between shipments, not mentioning the difficulties to see underwater under 10m of depth. Another way is by studying the time that the biological fouling takes to grow, and then deploy divers to perform the cleaning services. Therefore, to increase the efficiency of the detection services, an AI based automated visual inspection of ship hull surface is introduced. To develop an automated visual inspection system, a model controller is required to send analyse, send, and receive the data image from other components. In addition, an image acquisition and a lighting system is required to take footage under the water. The collected image may require some image enhancement process to clarify and perform a equal distribution of colour among the picture pixels. Using deep learning convolutional neural network, a model system is constructed to detect the biological fouling, corrosion, and metal loss on the marine vessel hull surface.

Keyword: Marine Vessel Hull, Underwater Visual Inspection, Deep Learning, Image Processing, Biological Fouling Recognition.

ACKNOWLEDGEMENT

In the name of Allah, the Most Gracious and the Most Merciful. Bless to Prophet Muhammad S.A.W, His companions and the people who follow His path. Thanks to Allah for His permission, allowing me to enter my final year after struggling some years to learn several useful courses based on my studies. Now, it is time for me to apply the knowledge into real applications in my Final Year Project (FYP). FYP is a compulsory course as it is a part of graduation requirement. First and foremost, I would like to express my humble gratitude and thank to my respected supervisor, Associate Professor Dr Zulkifli Zainal Abidin, for guiding me on this project. Even though this project is done on tough time due to the pandemic of Covid-19 and the recent floods, but his supervision helps me to complete this FYP 1 in any possible way. Also, I would like to thank to Dr. Hasan Firdaus for helping me to understand the concept of custom training of deep learning models. Last, thanks to Br. Ammar and Br Faris from Altus Project for accepting me to assist on this project, and for their assist on giving further explanation regarding the project objectives. Last but not least, thanks to everyone including my family and friends for their supports to encourage me to do well in my FYP project. Surely, FYP is a different experience from taught courses where it encourages the student to do an extensive study on some industrial topic by balancing between applying the knowledge learned from studies and exploring other related fields. Thus, I really hope this experience will help me on my career development.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 CHAPTER 1

1.1 BACKGROUND

Ship hull inspection has been and still one of the most essential aspects in the marine industry. That is due to the very sensitive material and shipments that is being carried. Things such as oil or nuclear materials can cause a catastrophe that its effect might last for decades, this is excluding the significant amount of money that must be paid as a compensation for the damages. Therefore, ship hull inspection must be given high priorities.

Ship hull inspection has developed throughout the previous decades. Around 40 years ago, inspection services were being applied manually, where divers are the ones who inspect and look for the technical damages, which could lead to human error. Based on (G. Pattofatto, 1991) paper, he mentioned that human errors are a type of error that is classified to be a medium-term issue. In other words, manual inspection is no longer an option to identify technical issues, that is due to the difficulties to inspect underwater, and the diving equipment fees. Therefore, companies begin to adapt new technologies and implement it in the marine industry. Such as, ROUV (Remotely Operated Underwater Vehicle), which is an underwater robot that conduct different type of services.

Visual inspection is considered to be one of the most preferred inspection methods. (G.J.Straits, 1991) mentioned in his paper about ship hull inspection method that inspection equipment and methods developed for other industries are beginning to show promise in aiding in the inspection of these vessels such as arrayed multi-processor controlled ultrasonic scanning, remote automated ultrasonic scanning, magnetic flux leakage,. tomography, and acoustic emission. Currently, we feel that the most effective inspection utilizes ultrasonic" and mechanical measurement techniques, visual and photographic records. In other words, visual inspection was and still one of the best methods that can be applied in the inspection services. However, ship hull visual inspection is currently used to identify things that sensors have difficulty to detect.

1.1 PROBLEM STATEMENT

Visual inspection is one of the most popular inspection methods that can be applied to detect technical issues. Due to its high efficiency and low cost of equipment, it is being implemented in almost every industrial field. In the marine industry, visual inspection used to be conducted manually by divers. Nowadays, with the entry of the inspection robots in the marine services industry. Real-time visual inspection services are being conducted using these inspection robots. Such a method requires the monitoring of an engineer the whole time, which can be tiring and inaccurate. In addition, underwater visuality is hazy, which makes it challenging to identify the technical issues. As a result, the goal of this project is to create an AI based ship hull visual inspection to collect images, enhance, and identify technical issues.

Ship's hull technical issues can vary from biofouling and corrosions to defects and holes. Most of the technical issues can be detected using advanced sensors, that uses technologies, such as, magnetic flux leakage and ultrasonic technologies. These inventions can detect mechanical issues precisely. However, there are things that cannot be detected using advanced sensors, such as, biofouling. Therefore, this project will detect biofouling spots which cannot be detected using sensors.

1.2 SCOPE OF THE STUDY

The research study focuses on adapting the visual inspection method to apply inspection services on marine vessel hull to detect biofouling spots and states its percentage in that spot. In addition, some of the IR 4.0 pillars will be implemented, supporting the Malaysian government's IR 4.0 initiative. Furthermore, it supports the fourteenth SDG's (Sustainable Development Goals), referring to life under water.

1.3 OBJECTIVES

- To investigate and study the suitable detection architecture for underwater environment.
- To develop a marine vessel hull visual inspection system using deep learning.
- To validate the integration of biofouling detection.

1.4 METHODOLOGY

This project is going to follow a specific method to achieve all the project objectives. It is going to aim on accomplishing small milestones which assists in achieving the whole project objectives. This method will make it easier to understand the topic and solve the problem.

1st MILESTONE: Understanding the techniques.

Collecting enough data on the technologies, detection algorithms, lighting designs, and image enhancements algorithms, from articles, reports, research, and

2nd MILESTONE: Constructing the software.

In this milestone, based on the collected information. A system will be constructed using that data as a reference. Which will be the system that detects biofouling spots.

3rd MILESTONE: Training the machine.

The machine will be trained on different data batches of biofouling spots. Covering almost every form of biofoulings.

4th MILESTON:

A simple prototype will be constructed to try the overall system, which is made of a camera and raspberry pi.

1.5 GANTT CHART

FYP I

Table 1 FYP1 GANT CHART

No	Task	Week													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Title Selection														
2	Intro, background, Objectives, etc														
3	Literature Review														
4	Software and Architecture														
5	Conceptual Design														
6	Components Selection														
7	Result and Analysis														
8	Discussion														
9	Conclusion														
10	Future Work														
11	Submission														
12	Recorded														

	Presentation														

FYP II

Table 2 FYP2 GANTT CHART

No	Task	Week													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Experimental Setup														
3	Collect the training data														
4	Code Development														
5	Model Deployment														
6	Result and Analysis														
7	Report writing														
8	Discussion														
9	Submission														

2 CHAPTER 2

LITERATURE REVIEW

2.1 OVERVIEW

In this chapter articles, reports, journals, research, and study cases will be reviewed looking for a potential algorithm or technology that can be used in detecting biofouling spots through visual methods. There are four primary systems/ algorithms that is going to be discussed in this chapter which represents the essential parts of every AI based visual inspection which are, object detection, lighting system, and image processing. In addition, each system/ algorithm will be reviewed, discussing its advantages and disadvantages and the reason of implementation.

2.2 BIOFOULING

Biofouling (Biological Fouling) is categorized as an issue that hinders the progress of marine vessels (Figure. 1). Marine vessel hull biofouling is a well-known problem for the shipping industry, leading to increased resistance and fuel consumption (Oliveira et al., 2018). An increasement in fuel consumption will definitely lead to an increment in the total costs. Thus, such issue must be detected and removed as soon as it begins to occur.

Figure 1. Biofouling stuck on the boat hull.

When it comes to detecting biofouling spots, many methods are introduced. Based on (Coraddu et al., 2019) the identification of the most influential parameters for the prediction of hull and propeller fouling in real operations based on data measured by the onboard automation systems. In addition, (Flemming & Flemming,

n.d.) mentioned that monitoring devices can be classified as systems which detect the kinetics of deposition of material and changes of thickness of deposit layer but cannot differentiate between microorganisms and abiotic deposit components. Furthermore, divers used to manually detect the biofouling spots under the ship hull cleaning process. Therefore, it can be concluded that there are many methods to identify and detect biological fouling spots, yet, these methods are not practical and efficient in long term operation, especially when it is implemented in an inconstant environment such as oceans.

2.3 DEEP LEARNING TRAINING METHODS

2.3.1 FINE-TUNING

Training the machine is considered to be one of the most important stages in AI (Artificial Intelligence) based projects. The better the machine is trained, the better it will perform while operating. However, the model's training is time and resource consuming. To train a deep learning neural network, tons of dataset are required for it to perform as intended. Therefore, a fine-tune algorithm is required to solve such an issue.

Fine-tune is basically applying a simple adjustment on previously existed data to achieve the desired output (Figure 2). For example, if a deep learning neural network is already trained to detect bicycles, then that neural network with small adjustments can be applied on motorcycles, due to the similarities that existed on both objects. By that, no training process on motorcycles is required. In addition, (Howard & Ruder, 2018)"mentioned that Fine-tuning has been used successfully to transfer

between similar tasks, e.g., in QA (Min et al., 2017), for distantly supervised sentiment analysis (Severyn and Moschitti, 2015), or MT domains (Sennrich et al., 2015)". Thus, it can be concluded that the closer the data sets are to each other, the better fine-tuning will operate.

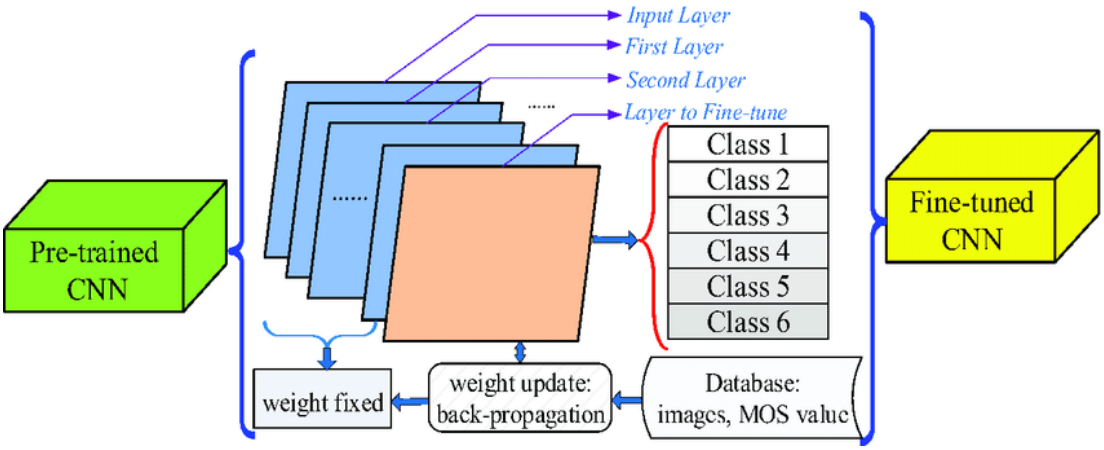


Figure 2 Fine-Tuning Process (Xu et al., 2019)

2.3.2 STraTA

STraTA (Self-Training with Task Augmentation) , is a new advanced unlabeled data architecture which is based on auxiliary fine-tuning deep learning training method. STraTA have proved its effectiveness comparing with other fine-tuning algorithms, where in just 8 training examples it reached an accuracy of 90% (Figure 3). Based on (Vu et al., 2021) paper, it was mentioned that Task augmentation and self-training provide complementary ways to leverage task-specific unlabeled data for improved downstream performance. While task augmentation utilizes unlabeled texts to synthesize a large amount of in-domain data for an auxiliary training task, self-training uses a model's predictions on unlabeled examples to improve the model itself.

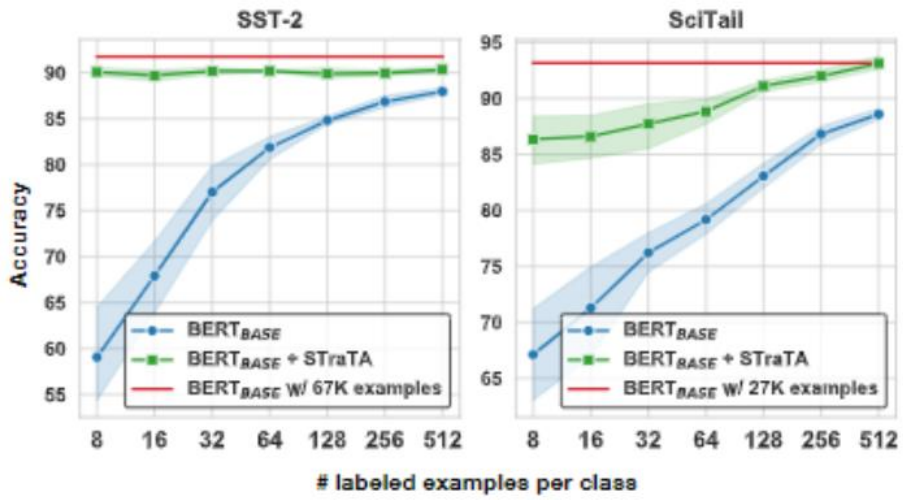


Figure 3. Performance of STraTA vs other architectures. (Source: (Vu et al., 2021)*)*

2.3.3 SUMMARY

To sum up fine-tuning and STraTA can be categorized as a process to train the data. Which is a method to train the data. Unlike other architectures, where it consists of a network architecture that is made for training the data.

2.4 OBJECT RECOGNITION

2.4.1 FASTER R-CNN

Region based convolutional neural network (R-CNN) is a neural architecture search algorithm (Figure. 4). R-CNN consist of 3 different types. R-CNN, fast R-CNN, faster RCNN. Where those versions have been developed and improved over time. Analyzing time of images used to take 45 seconds on R-CNN, on the other hand, it takes 2 seconds on faster RCNN. This is how far the R-CNN evolved. However, there are some limitations to such an algorithm, where it still undergoes some issues such as, identifying small objects, overlaps objects, and operates on a low FPS, which mean running on bad resolution. Furthermore, the existence of other architectures that are faster than R-CNN, takes it out of the list. (Girshick et al., n.d.) mentioned that It

is worth noting that OverFeat has a significant speed advantage over R-CNN: it is about 9x faster.

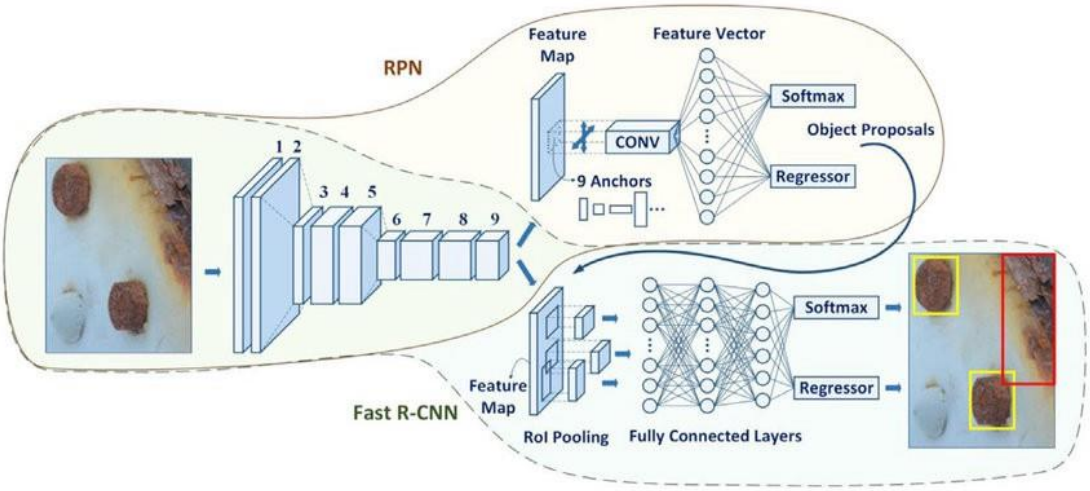


*Figure 4 R-CNN Architecture (Source: (*Faster RCNN Object Detection. Introduction | by Achraf KHAZRI | Towards Data Science, *n.d.))*

2.4.2 YOU ONLY LOOK ONCE (YOLO)

In normal methods, object detection algorithms are used based on the desired number of objects detecting. That is, to detect one object, it has to be stated that only one object will be detected in an image, which can be very inefficient when it comes to an image with multi objects. Therefore, YOLO takes part in detecting multi objects in one image in high speed and accuracy. YOLOv4 are located on the Pareto optimality curve and are superior to the fastest and most accurate detectors in terms of both speed and accuracy (Bochkovskiy et al., n.d.). Thus, figure. 5 YOLOv4 object detection architecture outperforms other architectures and algorithms. Furthermore, figure 6 shows the architecture of YOLO model.

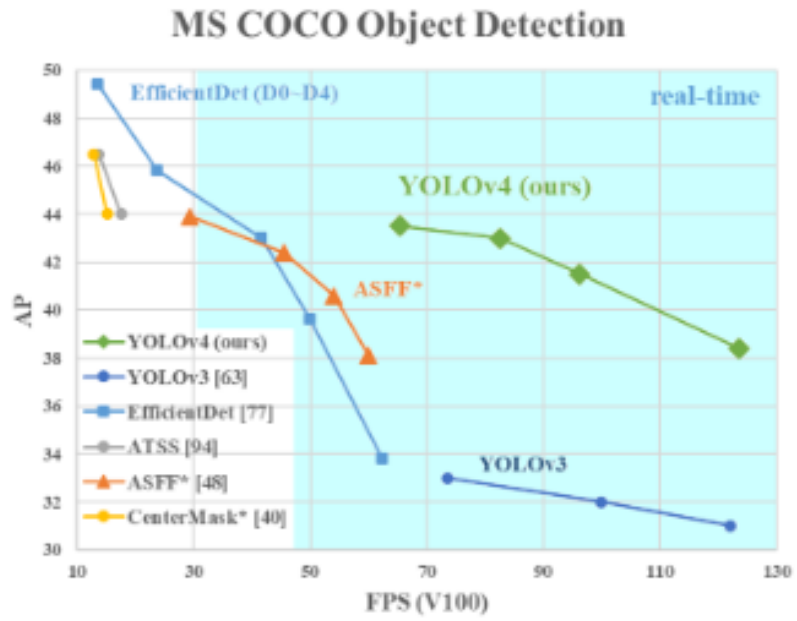


Figure 5. Performance of different object detection architecture. (Source: (Bochkovskiy et al., n.d.)*)*

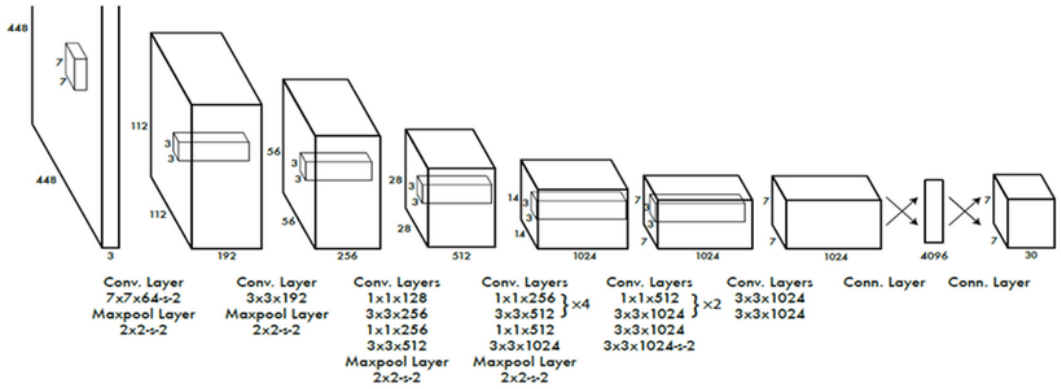


Figure 6 YOLO Architecture (Source: (Wu, 2018)*)*

2.4.3 U-NET

U-NET architecture was initiated to perform image segmentation for biomedical purposes. It consists of a total of 28 layers (Figure 7). Divided into two parts. A contracting path and an expansive path. The contracting path contracts the images size. On the other hand, the expansive path expands the image size after the information has been transferred from the contracting path. This architecture uses

image segmentation in its model. Therefore, segmentation annotation has to be performed to the training images. Segmentation annotation is the labeling of the object's pixels. Unlike the bounding box method, this labeling must be very accurate to obtain the right outcomes. Making this architecture obtain a high precision using a few training data. (Ronneberger et al., n.d.) mentioned in their paper that this architecture such that it works with very few training images and yields more precise segmentations. In other words, due to the image segmentation and this unique architecture, the model can have outputs with high precision.

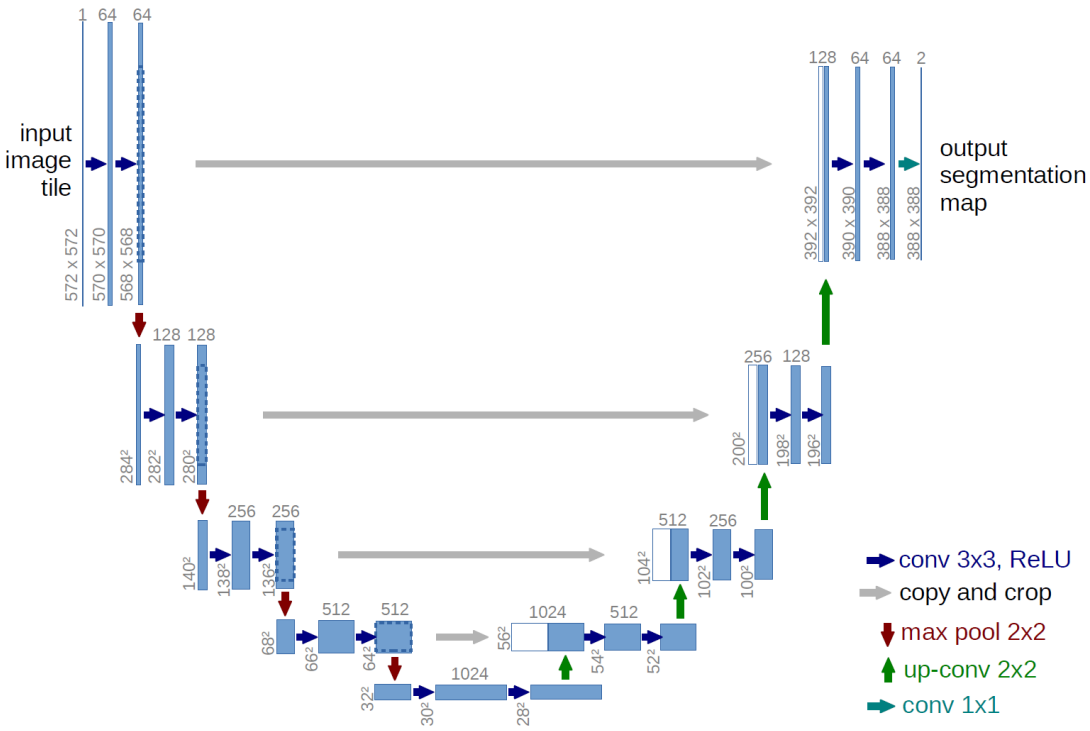


Figure 7 U-NET network architecture (Source: (Ronneberger et al., n.d.)*)*

Table 3 Object detection summary

OBJECT RECOGNITION (SUMMARY)			
(YOLO)		R-CNN	
ADVANTAGES	DISADVANTAGES	ADVANTAGES	DISADVANTAGES

- Detects far objects fast. - Can detect multi objects	- Does not state the area percentage of an object in the bounding box.	- Detects objects. - Set bounding box on it. .	- It is very slow compared to other architecture.
U-NET		SegNet	
ADVANTAGES	DISADVANTAGES	ADVANTAGES	DISADVANTAGES
- Uses image segmentation to recognize the objects. - Applies instance segmentation, which recognizes the background as well.	- Bad or Wrong annotation can lead to an inaccurate result.		

2.5 LIGHTING SYSTEM

The difficulty of underwater vision increases along the depth distance. Due to the limited amount of light that emits through the water, implementing a lighting system becomes a primary requirement to produce a better visual condition. Yet, most of the lighting systems are designed to fulfill the objectives based on a specific condition. Thus, the main objective in detecting biofouling spots underwater is to provide enough light which will enhance the visual condition.

When it comes to designing the lighting system, various aspects must be placed into consideration. Number of lights, light's position, light's color, and reflectors must be considered while designing a lighting system. (Jian et al., 2017) implemented a lighting system that contains six switchable LED lights (Figure 8) that are positioned creating a hexagon shape. This design is real time remotely controlled, which means that lights operate in a way that satisfies the user. On the other hand, a single LED light can be used in the lighting system, with the assist of reflectors, which distributes the light to a wide range of area. (Jionghui et al., 2015) introduced a method to combine refractor and a reflector, which will enhance the lighting system, making light affects a larger area. (Figure. 9)

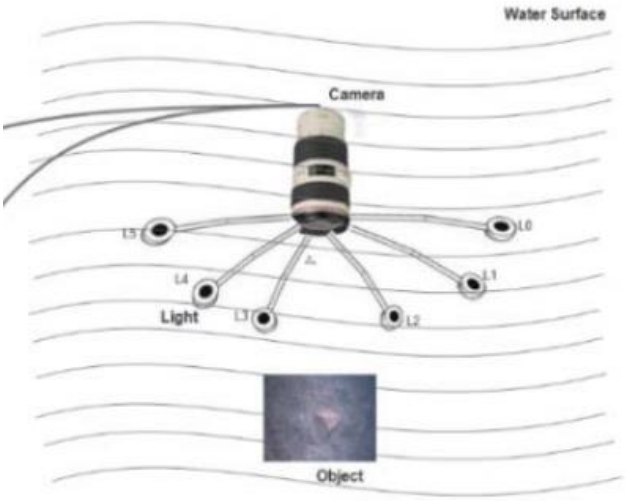


Figure 8. Multi LED lights lighting system. (Source: (Jian et al., 2017)*)*

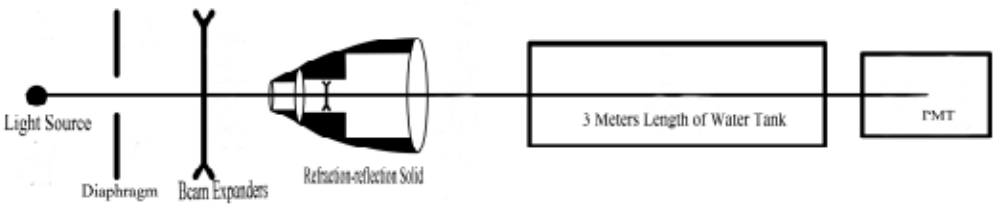


Figure 9. Reflector based lighting system. (Source: (Jionghui et al., 2015)*)*

On the other hand, when it comes to imaging, the lighting system parameters changes. The acquisition device (Camera), snaps its pictures by converting the object reflected light into an analog form that the device can understand, using a photodetector sensor. Therefore, if the reflected light was disturbed by particles, it will create noise in the image. The presence of complex biological and abiotic particles, such as soil particles and the floating excrement of marine animals, can change the direction of the light path in a complex manner (Zhu et al., 2021). This is what happens in underwater vision. Where the underwater turbidity interrupts the reflected light creating noise in images, which will affect the inspection process. Figure 10 shows how the light is affected by the underwater particles.

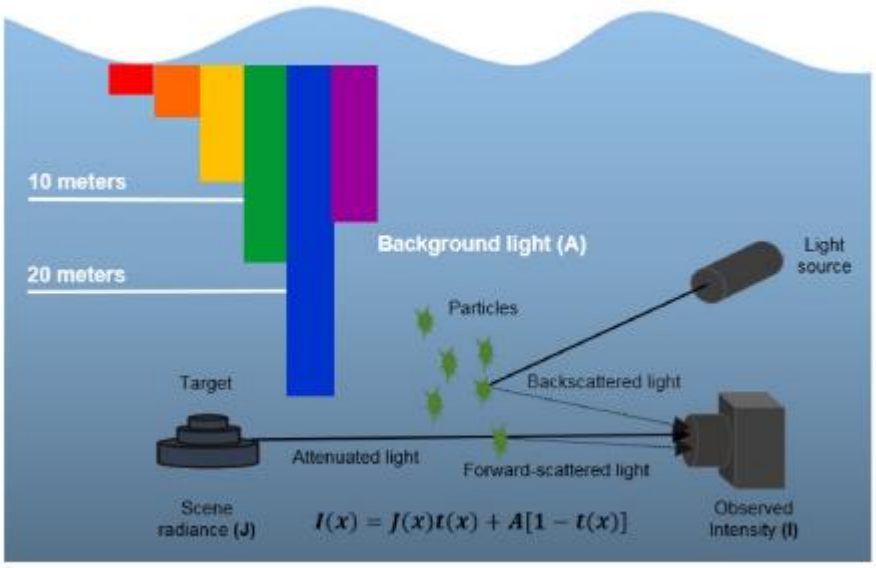


Figure 10 (Amer et al., 2019)

(Zhu et al., 2021) suggested two types of solutions that can reduce the occurrence of this phenomena. An image restoration approach, and a physical approach, which is using polarization. Polarization is filter that affects the light waves (transverse waves) which changes its emission angle. The purpose of this method is that due to polarization. The emitted

light will not hit the underwater particles. In addition, the emitted light that will reach the object will be reduced due to this process.

Table 4 Lighting system summary

LIGHTING SYSTEM (SUMMARY)				
MULTI LIGHTS		SINGLE LIGHT and REFLECTOR-BASED		POLARIZATION
ADVANTAGES	DISADVANTAGES	ADVANTAGES	DISADVANTAGES	
- Controllable light switches. - High brightness. - Reduce of underwater illumination.	- Difficult to implement on a multipurpose robot, due to its big size.	- Small size. - Large light coverage area. - Less components.	- High chance that the emitted light will intersect with the underwater particles and not reach the object.	- Reduces the amount of emitted light. - Digital purposes are more effective.

To sum up, a reflector-based lighting system tend to be more efficient and reliable. Due to the limited number of used components, which will give space to implement and use other technologies. In addition, a fixed lighting system is preferred when it comes to automating the inspection service. However, polarization seems to be a method to avoid the underwater particles, but when it comes to underwater darkness. Brightness is always prioritized.

2.6 IMAGE ENHANCEMENT

Either in real-time streaming or in collecting images. The collected data must be enhanced. Underwater vision is not similar to above water vision. (C. Y. Li et al., 2016) mentioned that “Images captured under water are usually degraded due to the effects of absorption and scattering. Degraded underwater images show some limitations when they are used for display and analysis. For example, underwater images with low contrast and colour cast decrease the accuracy rate of underwater object detection and marine biology recognition”. Furthermore, Underwater image enhancement as an indispensable step to improve the visual quality of recorded images has drawn much attention(C. Li et al., 2020). Therefore, image enhancement must be used to enhance and clarify the image in order to apply the required inspection services perfectly.

(C. Li et al., 2020) Introduced a CNN based image enhancement, which applies different image enhancement methods on a raw underwater image, choosing the enhancement method that suits the image condition. Figure 11 shows a simple example of how this algorithm function, as it can be seen, different enhancement methods are applied on the same picture, and when the perfect enhancement method is found, the system then simply choose it (chosen image is surrounded by a red boarder).

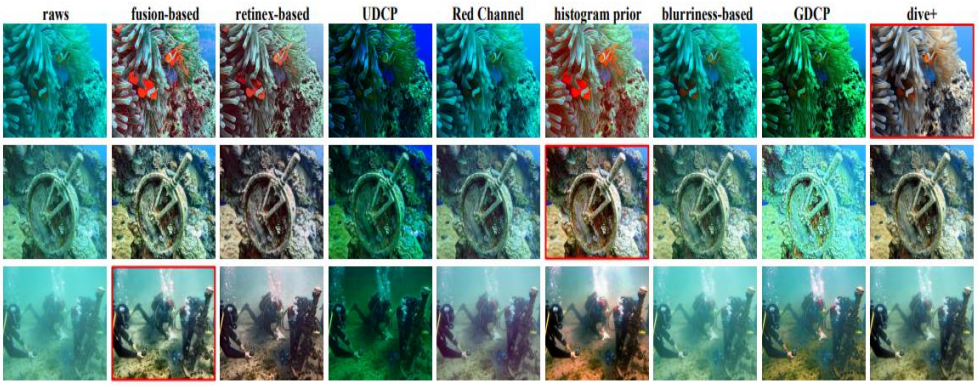


Figure 11. CNN based image enhancement process. (Source:(C. Li et al., 2020)*)*

3 CHAPTER 3

3.1 OVERVIEW

This chapter will go through the overall system design for this project. It has been divided into many subsections. Each section represents an essential pillar in the system model. The first subsection will discuss about the conceptual design illustrating the model's system architecture. The second subsection will discuss about the used software and architecture to develop the model. The third subsection will discuss about the implemented lighting system. Finally, the last part will go through the model preparation process. There are two prototypes that are being designed in this section. An underwater prototype and a on ground prototype. The on-ground prototype is to mimic the tasks that the underwater prototype will have to do.

3.2 CONCEPTUAL DESIGN

3.2.1 MODEL DEVELOPMENT FLOWCHART

Model preparation flowchart shows the process of preparing the deep learning model. Beginning from data collection until system deployment. Figure 12 displays the model preparation flowchart of AI based vessel hull visual inspection system.

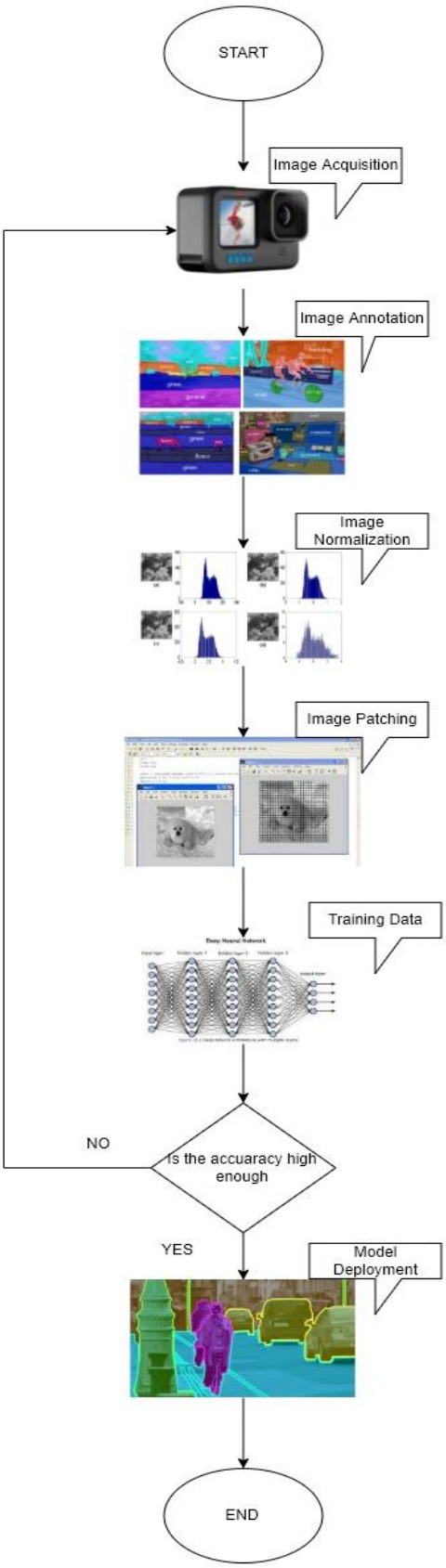


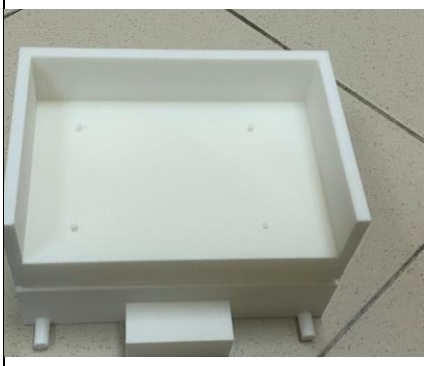
Figure 12 Model Development Flowchart

3.2.2 COMPONENT SELECTION

Table 5 represents the selected components for this project. These are the most essential components that will be implemented to build the ship hull visual inspection deep learning model. The selected components were, the underwater camera tray to contain the whole model, raspberry pi waterproof case, 1 TB SD card for data collection, raspberry pi to control the system, GoPro Hero 8 for image acquisition, Rolan flashlight as a lighting system, and a gaming laptop to train the data. On the other hand, the components that will be used for the on-ground prototype are, jetson nano 4GB, 3D printed structure, four mini wheels, 1 TB SD card, web camera, and a GOUI powerbank.

Table 5 List of components

Underwater Camera Tray with Plastic Carbon Arms and YS Adapter  RM 600	- Double handles with with 1-inch ball joint 25 mm - Action camera adapter for GoPro, sj4000 cam, dazzne, nilox, eye Cam ® - Dimensions: 260 x 60 x h 8 mm - Material: techno polymer.

Raspberry Pi waterproof case  RM 400	- Raspberry Pi waterproof case. - Temperature from -40°C to 55°C.
3D printed jetson nano structure 	- A simple structure to demonstrate this project. - Contain a place for the Jetson nano on the top of the structure. - The battery is placed at the bottom of the structure. - There are four places for wheels on the structure. - The camera will be placed on the sides.

Triple-S 2-inch wheels  RM 10	- Dia. 2inch: 2.5": outer 7.6mm+-, inner 5.9mm+- - Max load 45kg (2inch / 50mm)
1 TB SanDisk SD Card  RM 1,424	- Sequential Read Performance Up to 160MB/s and 90MB/s write speed.

Jetson Nano 4GB  RM 899	- GPU NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores. - CPU Quad-core ARM Cortex-A57 MPCore processor. - Memory 4 GB 64-bit LPDDR4, 1600MHz 25.6 GB/s - Storage 16 GB eMMC 5.1. - Video Encode 250MP/sec 1x 4K @ 30 (HEVC) 2x 1080p @ 60 (HEVC) 4x 1080p @ 30 (HEVC) 4x 720p @ 60 (HEVC) 9x 720p @ 30 (HEVC).
GoPro Hero 8 	- Video: 4K60, 2.7K120, 1080P240. - Photo: 12MP. - Dimensions: 66.3W x 48.6H x 28.4D (mm). - Digital lenses: SuperView, Wide, Linear, Narrow.

RM 1,400	
Web Camera  RM 26.90	- Item color: Black - DSP chip: Driverless - Image sensor: CMOS - Lens: Superior glass lens - High definition: 2M pixels - Dynamic - Resolution: 1920*1080 - Frame rate: 30fps
Rolan Waterproof Flashlight  RM 165	- Material: Aluminium alloy. - Brightness: 18000lm. - Bulbs: 15pcs 5050 white light XML2, 6pcs XPE red light R5, 6pcs XPE blue light R5. - Waterproof: IPX8. - Reachable depth of water: under 80 meters of water.

GOUI power bank  RM 120	- Li-lon Battery. - Micro USB：5V2. 1A. - Type-C PD in：5V3A，9V2A，12V 1. 5A. - Total output：5V 3A. - Capacity: 17000 mAh.
HP Pavilion Gaming Laptop 15-cx0xxx  RM 3,400	- Operating System: Windows 11 with 64- bit architecture. - Processor: Core i7 Intel. - GPU: NVIDIA® GeForce® GTX 1050Ti with 4 GB GDDR6 memory. - Port: HDMI/USB

3.2.3 SYSTEM ARCHITECTURE FLOWCHART

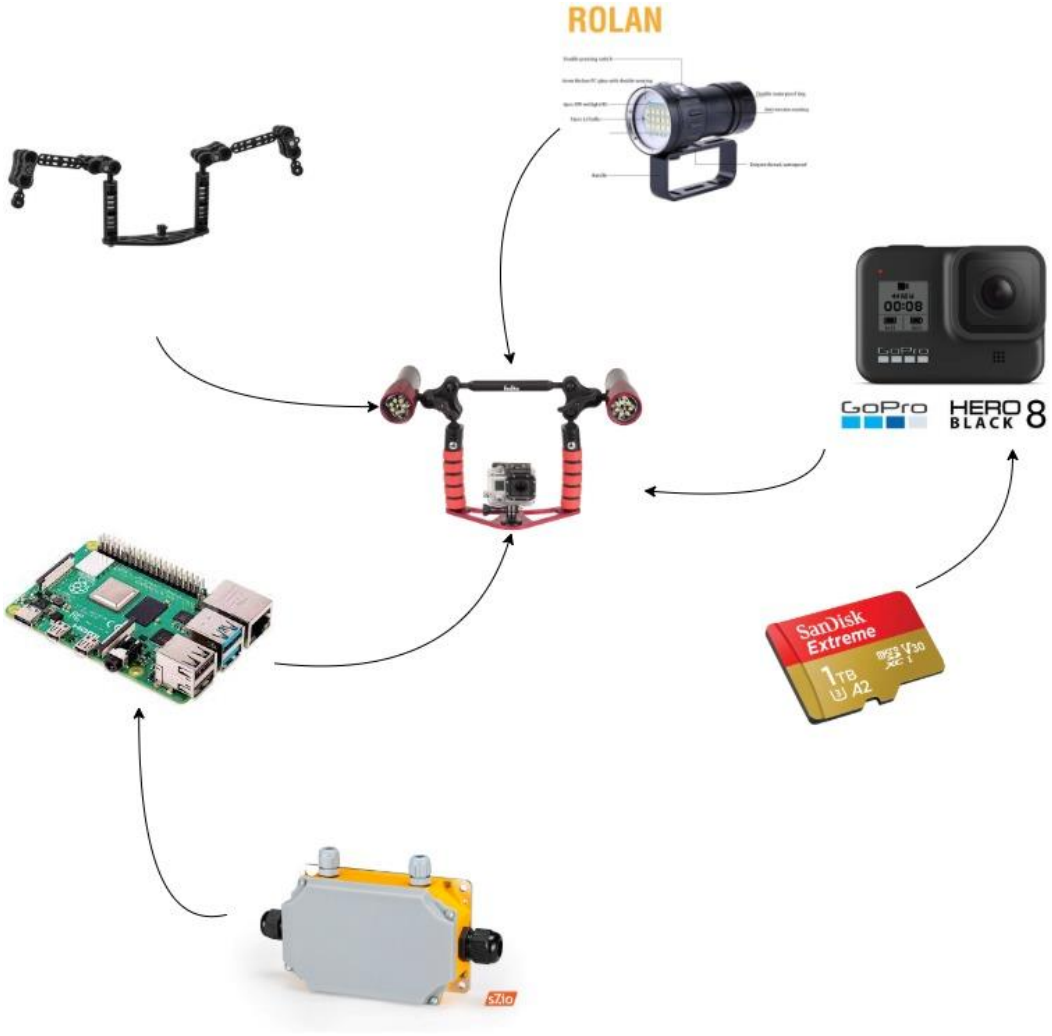


Figure 13 System Architecture Flowchart

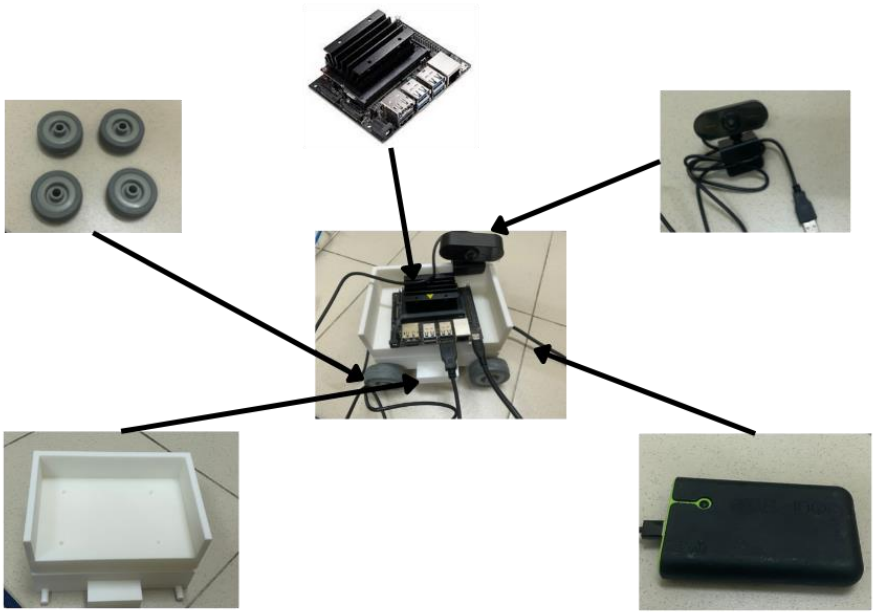


Figure 14 The made demonstration system

3.3 SOFTWARE AND ARCHITECTURE

3.3.1 SOFTWARE

USED SOFWARES AND PLATFORMS	
colab	Google Colaboratory is a programming notebook platform, similar to Jupyter notebook. It supports Python programming language. Providing free cloud GPU (Graphic Processing Unit) up to 12 GB and storage, making it easy to run complex and difficult codes.
ANACONDA	Anaconda is a programming platform that gathers a various type of libraries, platforms, and environments in one simple and easy to use program. Anaconda operates on the local computer graphic card. That is why it might

	require to use a high-quality graphic card for the best experience.
	Python is a programming language and considered to be one of the easiest programming languages to learn and use. That is due to its simple and understandable commands. It can be used for many purposes. However, it is highly used on Artificial Intelligence purposes.
	Numpy is a library package for scientific computing in Python. It has the ability to compute complicated math problems faster than the Python built-in functions.
	Apeer is a data annotation platform that is used to label the data images, prior the training process.
	Is a computing platform that uses the computer's graphic card to make certain tasks. It is usually used in deep learning and data visualization purposes.
	Ubuntu is a Linux operating system that is mostly being used on embedded system. It is considered to be simple and easy to use for beginners.
	Open Neural Network Exchange is a platform similar to Keras and Pytorch. As its models can be easily operated by the embedded systems such as Jetson nano. Usually the models made by Keras must be converted to ONNX before being able to deploy it into the jetson nano.

3.3.2 ARCHITECTURE

The used architecture for this project is U-NET architecture. That is due to its high ability to distinguish and localize the borders, through applying classification on each pixel. As mentioned in the literature review, it was made to use it on biomedical purposes. However, this project is going to adapt U-NET architecture and use it to classify and identify the different types of biofoulings and detect corrosion.

U-Net architecture is aptly named, where its diagram is in a form of U shape, representing the structure of the architecture (Figure. 15). This architecture consists of two parts. The first part is called the “Contracting Path”, which is the left part of the diagram, and the second part is called the “Expansive Path”, which is the right part of the diagram.

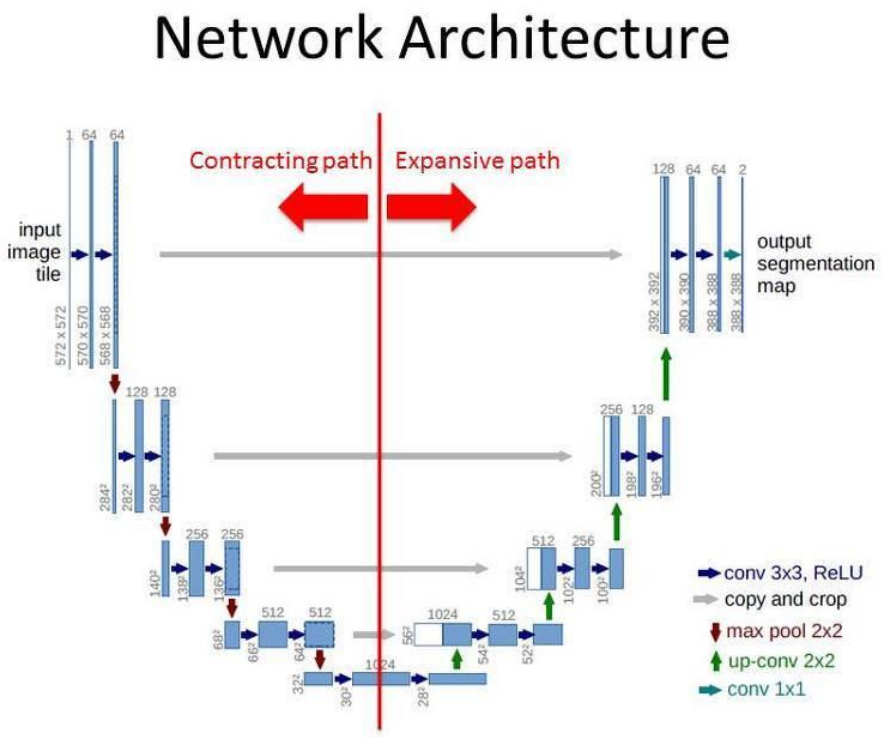


*Figure 15 U-NET Network Architecture (Source: (*Explain Fully Convolutional Network and Unet - 知乎, *n.d.))*

The contracting path consist of four, two convolutional layers and one max pooling layer, this process contracts the input image. Until the bottommost of the network, where there is another two convolutional layers, which transfers the information from a part to another. In the expansive path the image is going to be expanded to its original size. Going through the same process of the contracting path. It undergoes a four, two convolutional layers and one up sampling convolution layer. Reaching to the top of the network where the output is being reshaped. One of the requirements to use U-NET architecture is that the images must come in a specific size. That size must be dividable by 2 for four times without making a fraction.

3.4 IMAGE ACQUISITION

The image acquisition device that is going to be implemented in this project is GoPro Hero 8 (Figure 16). Due to its special design and robust structure, where it can take can bear up to 30m of water pressure. In addition, it has a wide FOV (Field of View), hence, it will cover more surface area of vessel hull. Along with its high resolution, outstanding stabilization system, small size, light weight, and its suitable price. It makes the perfect camera that can be used in the project.

*Figure 16 GoPro Hero 8 (Source: (*GoPro HERO 8 Black More Leaked Images & Specs – Camera News at Cameraegg, *n.d.))*

However, as the GoPro camera could not be obtained. It was replaced by a USB webcam purchased from Shopee online store. The camera supports HD resolution, MJPG pixel format, and works on laptop and the embedded system. Figure 17 shows some other features that the camera contain.

Figure 17HD Autofocus Web Camera

GoPro Hero 8 comes with a 12 Megapixel resolution and a CMOS photodetector sensor. It can shoot videos up to 4K/60fps resolution, providing high quality images showing every single detail in the image. Its FOV is 155°, being able to shoot in wide angle (Figure. 18), and it supports HDR (Hyper Dynamic Range). Meanwhile GoPro Hero 10 is available in the market as well, and it has better specifications than GoPro Hero 8, but these advantages are not important when it comes to visual inspection, more details will be given in the image normalization part regarding the image resolution and the training process.

*Figure 18 Wide angle FOV shot compared with linear FOV shot. (Source: (*How To Use GoPro Field of View (FOV) Like A Jedi [2021 Update], *n.d.))*

However, the only downside of the GoPro Hero cameras is the barrel distortion, which is the bending that happens to the top and bottom part of the image as it can be seen in figure 19. Due to the wide-angle lenses. It is not curtained that this issue can affect the inspection service, but it is better to avoid it. By using a python code, it can be easily getting rid of the image bending, and images will look normal.

Figure 19 Barrel Lens Distortion (Source: ((4) What Is Barrel Distortion in Photography? - Quora, *n.d.))*

The camera and the lighting system will be extended by a metal arm, hanging it in a specific position, creating a 90° angle with the ship hull surface. The distance between the camera and the ship hull surface will be then determined based on the area covered by the robot.

3.5 LIGHTING SYSTEM

The lighting system will consist of a Rolan IPX8 18000lm 500M Flashlight (Figure 20). Contains 15 pieces of 5050 white light XML2, 6 pieces of XPE red light R5, and 6 pieces of XPE blue light R5. Positioned above the camera and away from the robot, avoiding any shadows that can appear as an illusion to the model. Providing a strong light emission, which is suitable for underwater vision.

*Figure 20 Rolan Waterproof Flashlight (Source: (*Rolan IPX8 18000lm 500M Flashlight Waterproof Lamp Light Underwater Outdoor Camp | Shopee Malaysia, *n.d.))*

Initially, only one torch will be implemented in the project. Based on the results of the upcoming lab experiments, further changes might be applied. However, underwater vision is not ideal. Not all oceans have a clear see-through water. Underwater turbidity varies from a place to another. Therefore, the camera-lighting system will highly depend on the underwater environment, hence, the lab experiment

will simulate the underwater turbidity by adding milk into the water. (Amer et al., 2019) mentioned in his paper that, "In this part we present our first tests that were carried out in a controlled laboratory water tank (15 *L* of natural water) environment with different turbidity conditions by adding milk". In short, it means that milk creates a perfect haziness that mimics the underwater turbidity.

3.5.1 MODEL PREPARATION

3.5.1.1 DATA COLLECTION

Data is the new oil, for this project as much data as possible must be collected, which will lead to enhancing the performance of the model. In addition, high variety data is significantly important as well, which expands the range of recognized biofouling species and type of corrosions by the model. Furthermore, the type of background also takes an essential part of the data collection stage. As the collected data must have the same background to be able to perform normally.

However, finding a pre collected data related to biological fouling on the internet is difficult and inaccurate, because the collected data does not reflect the project robot's point of perspective. As the distance between the camera and the object in the data found on the internet is different from the one is going to be used in the robot. Therefore, initially raw data must be collected manually, then further collection can be conducted after finalizing the model. The collected data then, must be divided as training, validating, and testing into 80%, 10%, and 10% of the overall data respectively, which will be used to develop and enhance the model.

For this project two types of datasets will be used, a marine vessel ship hull corrosion dataset that was taken from a website, and a wall biological fouling that was collected manually. In terms of high variation, different species of biological fouling

can be taken into consideration. However, the extreme dark biological fouling are the types of dirt that is being focused on.

In addition, corrosion and biological fouling can occur in many forms, which differ based on its stage. Its growth is random and non-uniform. Which does not form a solid one object. Such a dataset requires a specific system architecture to be trained properly.

Initially, due to the lack of biological fouling dataset. A corrosion dataset from Mendeley Data has been used instead. The dataset consists of a high-resolution image (Figure. 21) and a low-resolution image (Figure. 22) of marine vessel corrosion. The data was prelabeled as well (Figure. 23). The number of the low-resolution images are 39 images, and the number of the high-resolution images are 37 images.

Figure 21 Low-Resolution Data (Source: (Chliveros et al., 2021)*)*

Figure 22 High-Resolution Data (Source: (Chliveros et al., 2021)*)*

Figure 23 High-Resolution Data Annotation (Source: (Chliveros et al., 2021)*)*

However, due to the bad camera posture in the Marine vessel corrosion dataset. It is not suitable to be used as a main dataset for this project as its position is too far from the surface and covers more area than the robot will be able to cover. Therefore, a dataset of a wall green mold (Fungus) has been collected manually to mimic the appearance of biological fouling on ship hull. The images have been collected from a distance that vary between 10 cm to 17 cm. Normal to the surface of

the wall. On a sunny day. These were the conditions that the following image in figure 24 were collected in.

Figure 24 Sample image of the wall fungus data

The condition that the dataset images were collected in is the most essential part in the dataset collection procedure. As it will be the weights that the model will use to predict the outputs of the image.

The wall fungus dataset contained 54 images, extracted from a video that has been taken of one of the campuses building on IIUM. The video has been shot using a Insta360 ONE X camera in 3K resolution and a 360 degrees coverage. In figure 25, the camera's software was used to read the recorded video, as it comes with a file type that can not be read by normal media player programs.

Figure 25 Insta360 ONE X Camera

The extraction of the image from the video degrades the image quality. As many details that used to appear in the video became fuzzy after taking screenshots from the video. That is due to the angle coverage that the camera had, as the whole 3k pixels were distributed on 360 degrees instead of 160 degrees. Therefore, the resolution of the image is not good. However, it is still applicable and can be used in the training stage as long as the same image resolution is going to be used in the prediction stage. In figure 26, is a sample of the collected images that will be used in the training process.

Figure 26 Sample from the wall fungus image dataset.

3.5.1.2 IMAGE ANNOTATION

Image annotation is the process of labeling the corrosion or fungus on the image data set, based on that the type of labeling then will be chosen. For object detection (Bounding Box) purposes, it is preferred to be applied on objects that is complete in structure, like, humans, animals, cars, etc... On the other hand, fragmented objects, such as, corrosions, biological fouling, and human cells uses image segmentation to label each specific class. Choosing or implementing the right annotation method can abbreviate the labeling process and increase the overall accuracy.

The type of annotation used in this project is semantic segmentation annotation (Figure 27). Which is a pixel-based annotation. Targeting the main pixels that represent the object, and performing a very precise labeling process, where only the pixels related to the object must be labeled.

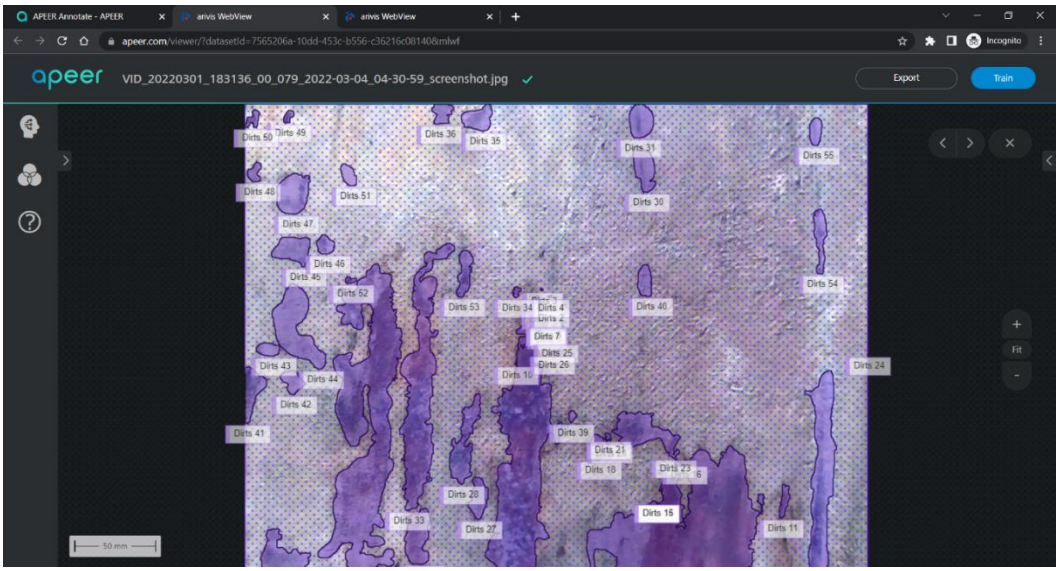


Figure 27 Annotation using apeer

This type of annotation has been chosen due to the type of dataset. Taking corrosion as an example, it can be seen that the corrosion spots are not gathered in one place, but it is spread all over the surface as it can be seen in figure 28. Furthermore, it does not appear in one size, but in different size and shapes. Therefore, bounding box annotation is not a suitable choice for such a dataset, because while labeling, every corrosion spot in the picture must be annotated, and annotation should fit the size of corrosion. Any excess pixels that are irrelevant to the object itself will affect the training process, which will lead to have a weak image segmentation process.

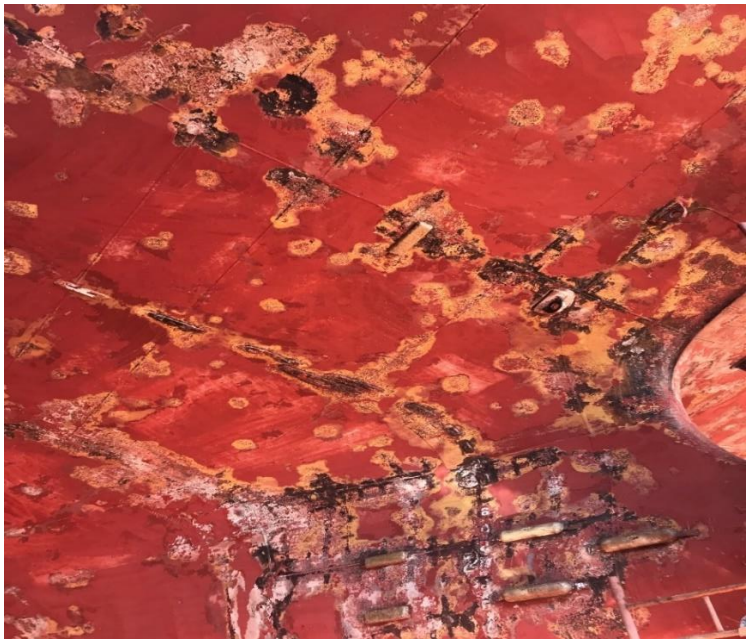


Figure 28 The distribution of corrosion on vessel hull. (Source: (Chliveros et al., 2021)*)*

The masks of the Marine Vessel Corrosion dataset were a coloured mask as shown in figure 29. It was a prelabelled mask, and it was included along with the dataset.

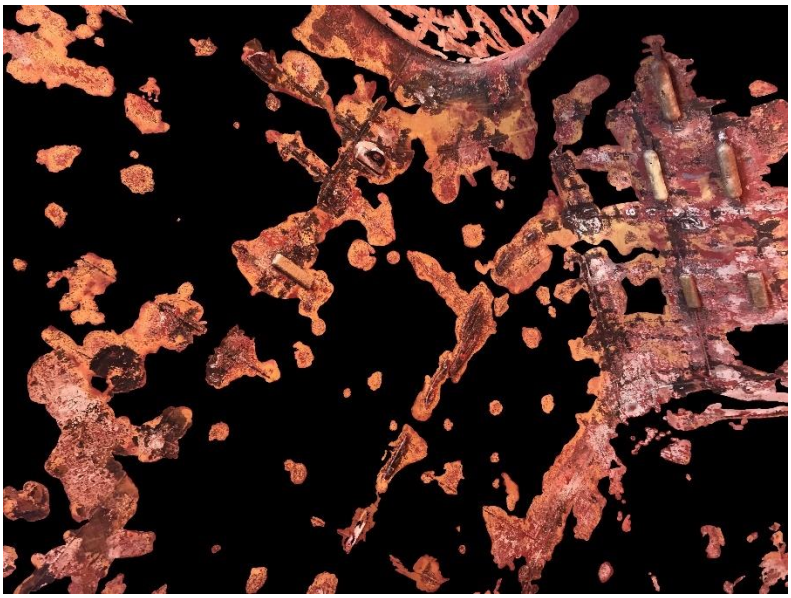


Figure 29 Colored mask of Marine Vessel Corrosion

Same concept can be applied on the wall fungus dataset. Furthermore, the collected data then will be uploaded to apeer website to label the images. There are many tools that can be used on apeer to label the image. It helps in terms of the object shape, as some shapes are edgy while others are round. In addition, there are color filters that can be used to enhance the vision of the object, as some objects are harder to see, such as human cells.

After finalizing the image annotation, then a mask is being created. The masks are basically a copy of the made annotation, except it shows the position of the labelled object on the image, so it can be recognized in the training session. In figure 30, it can be seen that the mask comes in two colors, representing the object and the background. This type of mask is called binary mask as it comes in black and white. The black color represents the background, and the white color represents the labelled object.

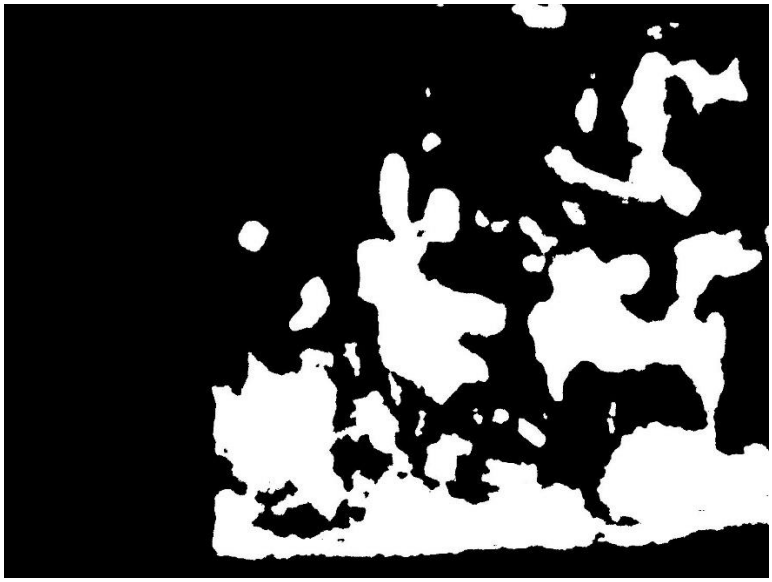


Figure 30 A sample mask of the wall fungus dataset

3.5.1.3 IMAGE NORMALZATION

Image normalization is to normalize and set all images into the same range of pixels. In image segmentation, normalization is the equalization of image dimensions between the test images and its masks. That is due to the training process of segmented data. In the training process, the model applies its training only on the pixels selected in the mask.

In this project, image normalization takes place to ensure that all the data and its mask are in the same size. This is a compulsory step in preparing the data because the model will not initiate the training process until all the images and masks are in the same resolution.

3.5.1.4 IMAGE PATCHING

Image patching is the process of cutting or cropping images into smaller parts. In other words, it is the subsection of an input image. It fastens the training process. This happens due to the decrement of the image dimension, where it becomes easy for the model to go through the image. However, it may decrease the accuracy, it is done due to the limited GPU memory.

As previously mentioned in part 3.6.1 (Data Collection). There are two types of collected data on vessel hull corrosion. A high-resolution data (4032 x 3024) and a low-resolution data (1920 x 1080). These image's dimensions are considered to be significantly large, because the more pixels that the image contains, the more GPU is required to train the model. Therefore, the images must be patched in order to suit the GPU's capabilities. For example, if a data image with a resolution of (512-pixel x 512-pixel) is going to be trained. Then at least a GPU of 6 GB is required for such a task.

Since Google Colab provides a 12 GB GPU, then an image of (512 x 512) resolution can be effectively trained. Hence, in this project, the patching process has been applied into the low-resolution dataset creating two data outputs (Figure 31). One with a (256 x 256) resolution and the other with (512 x 512) resolution. The variation of the dataset sizes is due to the impact of the image size on the training process. Therefore, a varied number of resolutions must be prepared to check which will ensue the optimal result.

Figure 31 Applying image patching on low-resolution data. (Source: (Chliveros et al., 2021)*)*

In addition, same concept is going to be applied on the wall fungus dataset. Except for this dataset the images are going to be divided into 360 x 360 pixels, that is due to the original image pixels and the requirements of using U-NET architecture. Figure 32 shows how does the patched wall fungus image looks like.

Figure 32 A patched image of the wall fungus dataset

Based on the concept of patching and since one image is going to be divided into 12 smaller images. In addition, since the usual size of an image that is used in the model is going to be 360 x 360 pixels and considering applying patching on all 54 images. Therefore, it can be said that the new number of images in the dataset is 648 images.

3.5.1.5 TRAINING MODEL

After applying image patching on the raw images and masks. The data then is going to be imported into the model system. Since this project is initially using Google Colab for the training process, then the data will first be uploaded into Google Drive, then it can be imported into the code (Appendix A).

Referring to appendix A, it contains the U-NET architecture code(Appendix B). Which contains the U-NET convolutional neural network layers, optimizer, and the

loss function. The U-NET architecture network has been illustrated in section 3.2.1. The optimizer is the function that finds the value of the weights parameters that minimizes the loss. In addition, the loss function, is a function that calculates the average loss of the model. On the other hand, the second part contains the running code. Where it imports the necessary libraries, imports the images and masks from Google drive, reset the image size (resize), checks if the images are normalized, calls the U-NET model (Appendix B), and then runs the model for training on a specific batch size and epochs. To finally generates the accuracy, IoU (Intersection over Union), the loss, and a graph that represents the mentioned variables.

3.5.2 APPLIED FUNCTIONS FOR TRAINING

In regards the implemented functions in the training process. There were two main functions that were used in the model. An activation function and a loss function. The activation function is a function that do a simple decision making of whether to activate a specific neuron or not. In addition, the loss function is a function that calculates the error of the predicted output and the actual output. Sigmoid activation function and binary cross entropy loss function were used to build this model as it is mentioned in appendix B. Figure 33 and 34 indicates the equations of both functions.

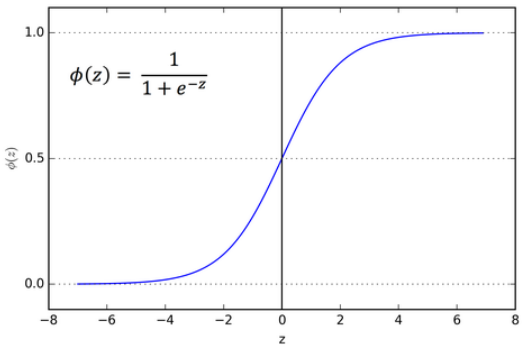


Figure 33 Sigmoid Activation

$$\text{Loss} = -\frac{1}{\substack{\text{output} \\ \text{size}}} \sum_{i=1}^{\substack{\text{output} \\ \text{size}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

Figure 34 Cross Entropy loss function

3.5.3 MODEL SAVING

After the training process the model has to be saved. Which contains all the training weights. The trained model will be saved as an (.hdf5) file which is a type of file that Keras platform saves its model on. The following model can then be imported into any code and used normally for prediction purposes. A sample of the code that can be used to extract the weights from the model and apply it to predict images can be found in appendix C.

3.5.4 MODEL PREDICTION

The prediction stage requires calling the saved model, extracting the training weights, then apply prediction on the images. The prediction images must be different than the training images, otherwise the prediction will be similar to the user's annotation. Further illustration about the model's prediction results will be discussed in chapter 4.

3.5.5 MODEL DEPLOYMENT

To improve the quality of the project and to simulate the implementation of the model in the AI based visual inspection of ship hull surface. The model is going to be implemented into jetson nano as shown in figure 35. Jetson nano is an embedded system that is preferrable to be used in machine/ deep learning purposes, due to its ability of good performance in this field. The used jetson nano is the 4GB version as it contains 4 Giga Bytes of ram.

Figure 35 Jetson nano 4GB

After installing Ubuntu operating system on the jetson nano, further packages and libraries must be installed to ensure the smoothness of the model's prediction process, as shown in figure 36. However, when it comes to applying the prediction process using Keras's (.hdf5) model, the jetson nano will not be able to handle it due to its limited ram capacity. Therefore, the file type must be changed and replaced with a (.onnx) file type, which is smoother than Keras and does not require a high ram to operate. Appendix D shows a simple code used on google colab that converts (.hdf5) file type to (.onnx) file type. The jetson nano can then run the model and apply prediction on the image easily and within few seconds.

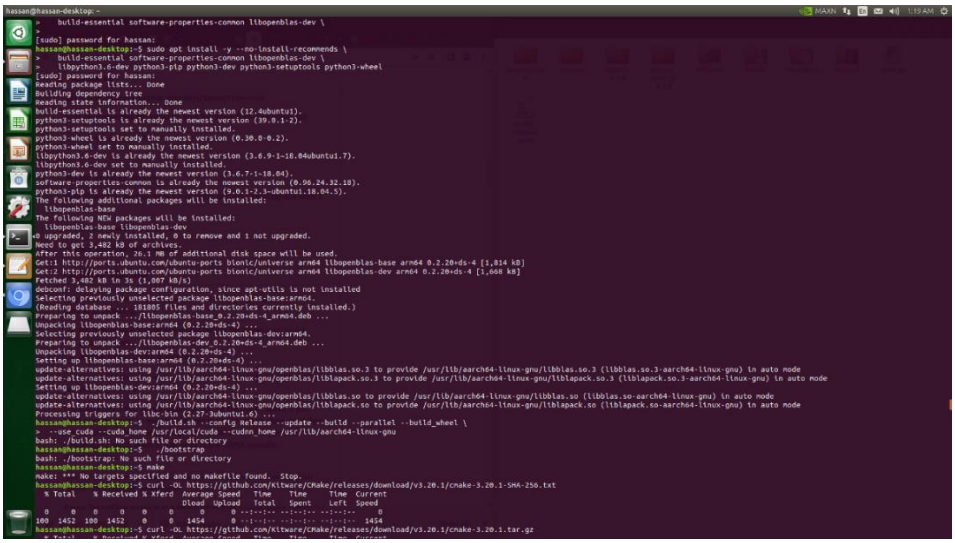


Figure 36 Installing packages on Jetson nano

3.5.6 SYSTEM DEMONSTRATION

To demonstrate the system and simulate the environment that it might be in. A simple structure has been designed and printed using 3D printer, the model has been applied on the real-time footage collected by the camera that was mentioned earlier. The following figure 37 shows the design of the system that will simulate the experience of predicting images of wall fungus.

Figure 37 System demonstration

Furthermore the model has been given a position to be placed in the main robot as it can be seen in the following figure 38. Beside the camera, arduino, and raspberry pi.

Figure 38 Model deployment

4 CHAPTER 4

4.1 OVERVIEW

The outcomes of chapter 3 will be discussed in this chapter. All the initial trials and results will be discussed. Illustrating and analyzing the behavior of the training model and the system in general. Initially, the data used for the training is a vessel hull corrosion data.

4.2 RESULTS AND ANALYSIS

As the first stage into developing an AI based ship hull visual inspection system, there were many made trials of training the model on marine vessel hull corrosion and wall fungus datasets. In the first trial YOLOv5 architecture (Figure. 39) were used to identify and classify the corrosion in the dataset.

*Figure 39 YOLOv5 Architecture (Source: (*Ultralytics/Yolov5: YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite, *n.d.))*

YOLOv5 is considered to be one of the most advanced versions of YOLO architectures. Therefore, it is one of the most recommended architectures to use for

object detection. Hence, to begin with, the first trial on training custom data using YOLOv5 is uploading the dataset on roboflow, annotate it, then export the labeled data as a link to use it in the YOLOv5 architecture to train it (Figure. 40). Therefore, the vessel hull corrosion data has been uploaded into roboflow, labeled manually image by image (Figure. 41, 42, and 43), then exported as link and placed in the code (Figure. 44).

*Figure 40 The process of using YOLOv5 Architecture. (Source: (*Train Custom Data · Ultralytics/Yolov5 Wiki, *n.d.))*

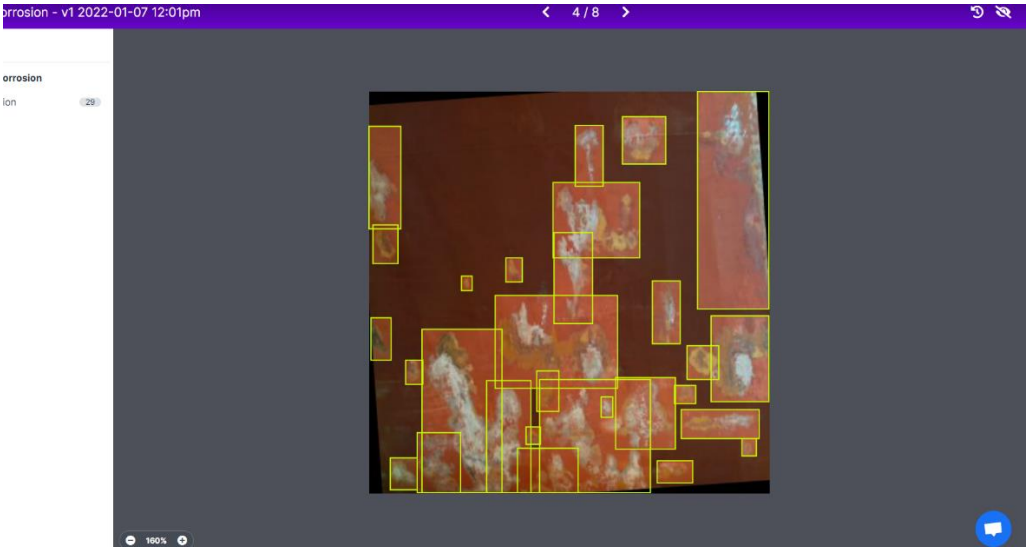


Figure 41 Bounding box labeling of ship hull corrosion

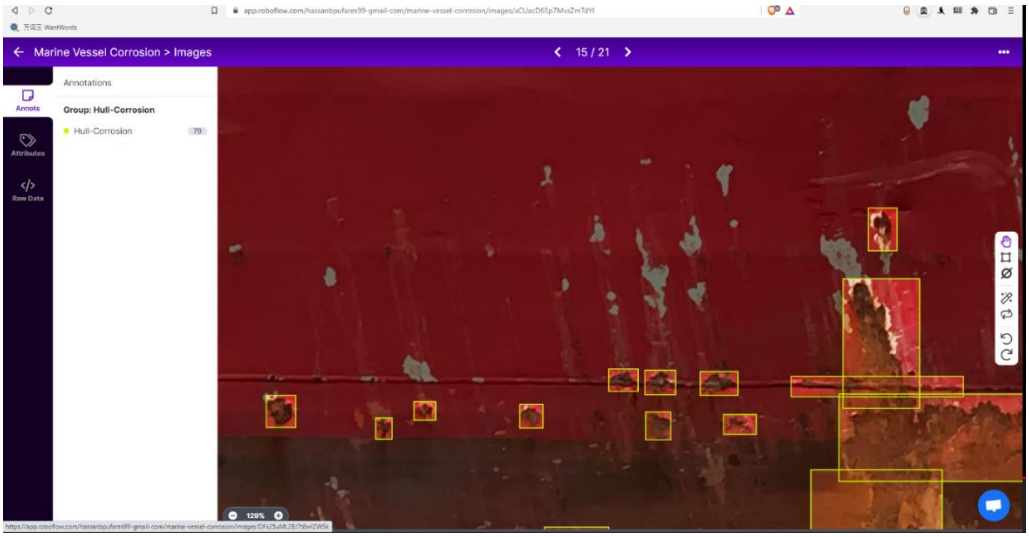


Figure 42 Bounding box labeling of ship hull corrosion

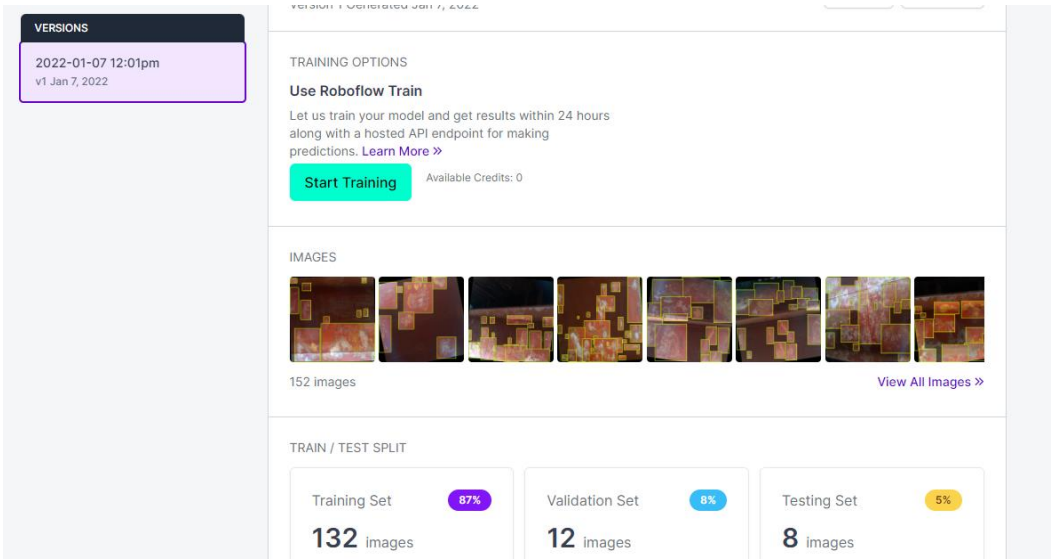


Figure 43 The collection of annotated data.

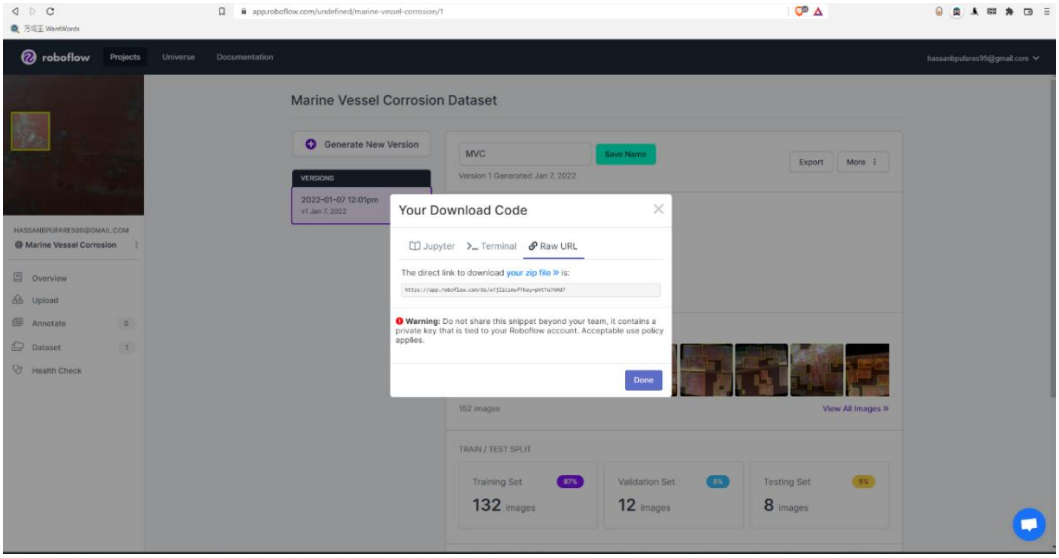


Figure 44 Exporting the labeled output images

There is also different type of models that can be used for training purposes, and it is based on the size of the dataset (Figure. 45). Since the ship hull corrosion dataset is small. Then in the first trial, yolov5n has been used as the training model.

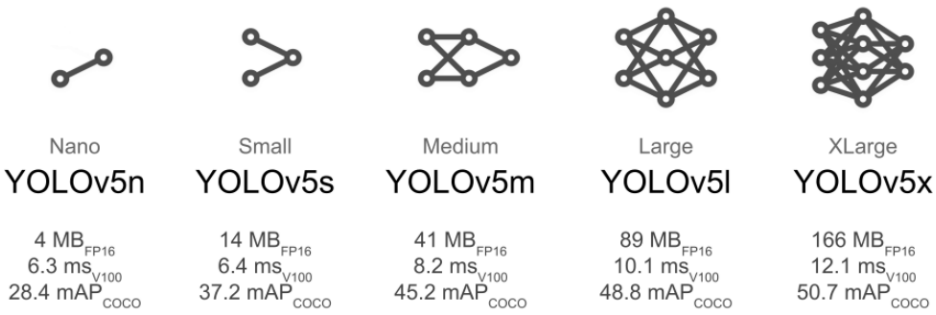


*Figure 45 Type of training models (Source: (*Train Custom Data · Ultralytics/Yolov5 Wiki, *n.d.))*

The first result that has been obtained using YOLOv5 model (Figure. 46 and Figure. 47). The overall result seems dissatisfactory, the acquired Mean Average Precision (mAP) was around 0.3. which is considered to be very low. Even after trying it on a test image as in figure 48 the accuracy was 0.3 as well. The occurrence of this result indicates that there is an issue either with the used model, parameters,

epochs, learning rate, or the type of labeling. It can be said that the type of labeling is incorrect. The ship hull corrosion distribution pattern is random and fragmented all around the surface and the image. Therefore, it will be difficult to annotate such a data using bounding box annotation method, due to its inaccuracy. It is even logically to be considered as a wrong type of annotation.

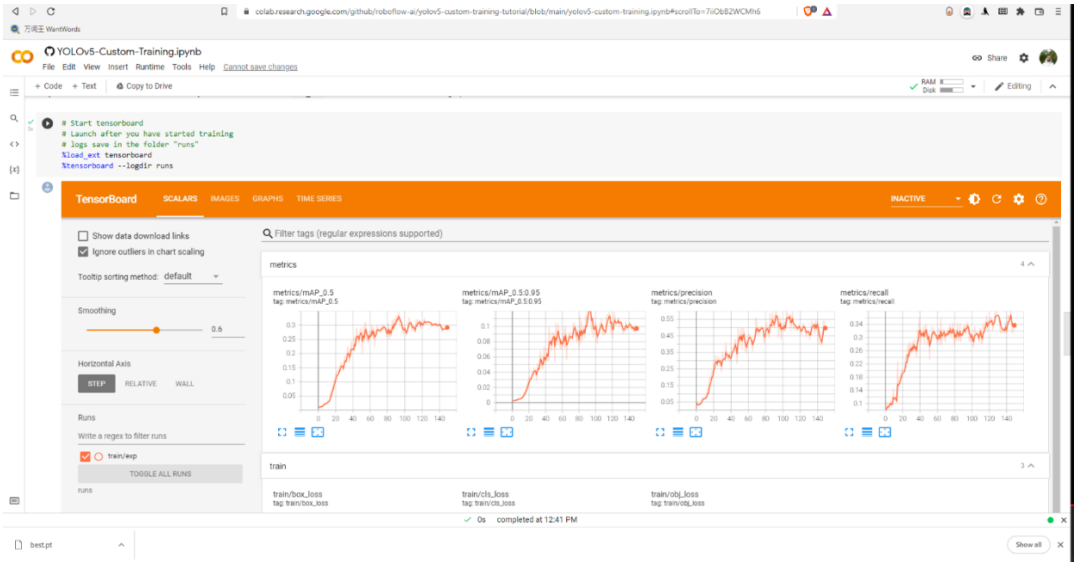


Figure 46 Result of YOLOv5n model in detecting ship hull corrosion.

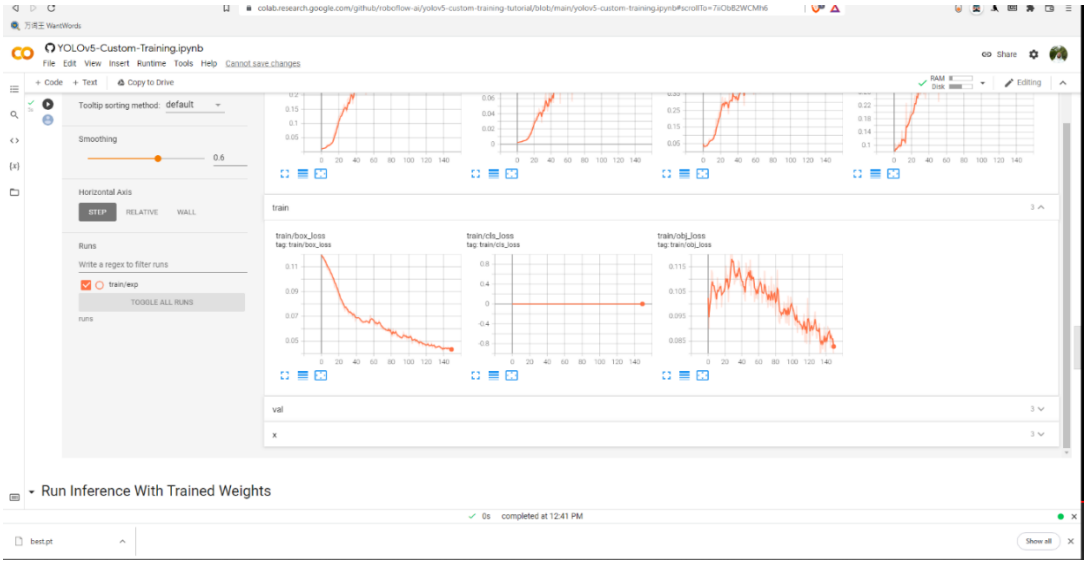


Figure 47 Result of YOLOv5n model in detecting ship hull corrosion.

Figure 48 Applying the model on a random corrosion image from Google.

The second and third trial was using U-NET architecture. The process is almost similar to the YOLOv5 model training process. The first step after preparing the code is to preform image normalization to the dataset, which consist of the raw images and the colored masks, to ensure that all images are in the same resolution. The second step is to perform image patching, which will crop the dataset images into smaller subsections, which will lead to a reduction in the use of GPU. The third step is that the data then is uploaded on Google Drive, and then it was ran using the codes in the appendices.

Regarding the result of the U-NET training model (Figure. 49 and Figure 50), it appears to be better than YOLOv5n, and that is due to the image segmentation annotation. Because the image segmentation tends to be more accurate and precise than the bounding box annotation. In the two trials, the training model had an accuracy higher than 50%, which is better than YOLOv5n. However, it seems that

there is zero IoU in the results. Which means that this training model isn't functioning well. This time it is due to the data annotation and the loss function. The implemented loss functions for the U-NET trials were binary loss function. Binary loss function operates masks that are in black and white color. While the masks that are available with the data is colored.

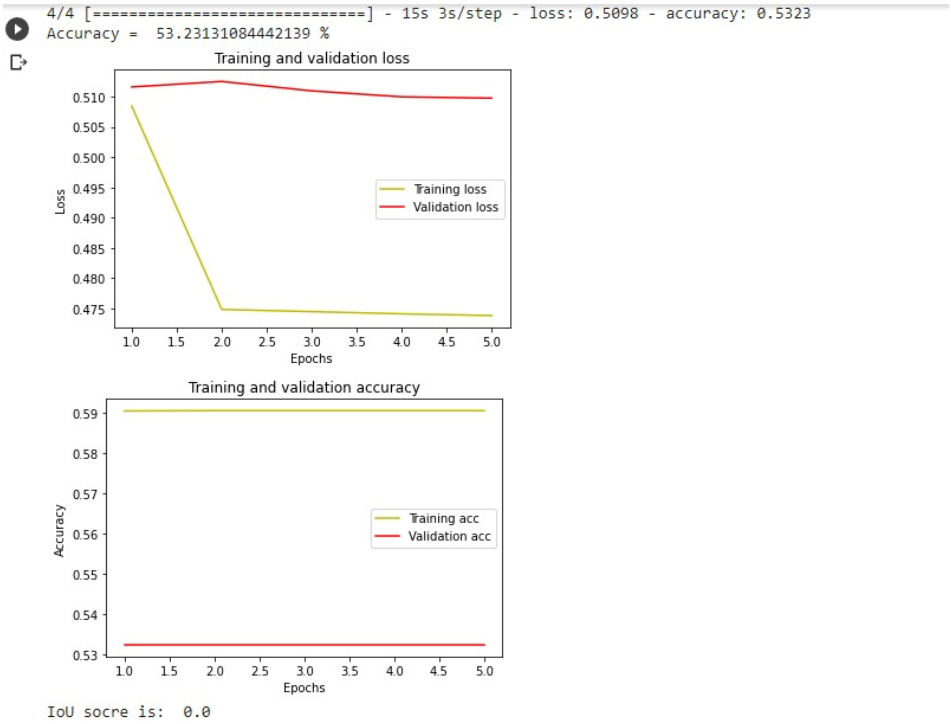


Figure 49 First U-NET training model trial

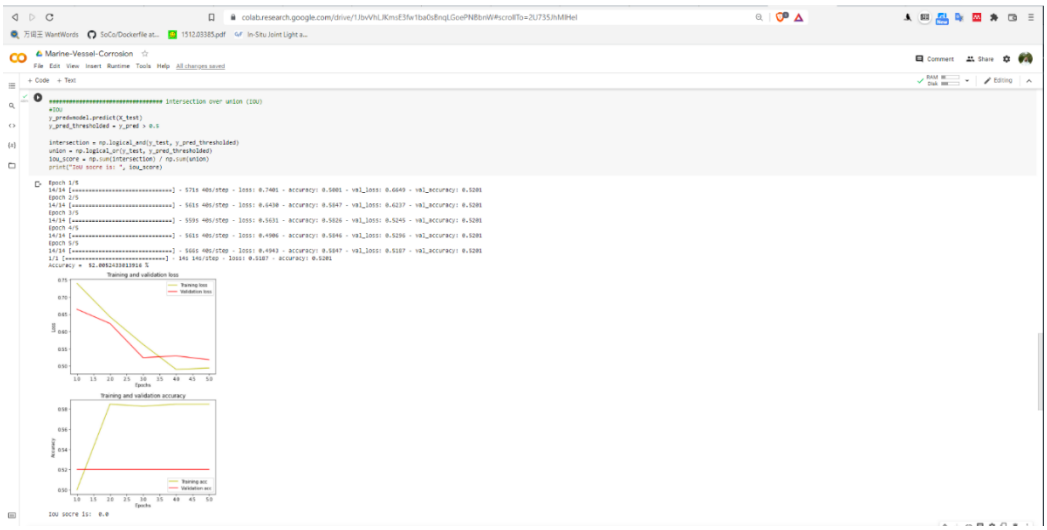


Figure 50 second U-NET training model trial

The fourth trial was applied on the wall fungus image dataset. Using U-NET architecture as well. Some adjustments have been made to improve the model's behavior. First, the data has been manually annotated. Second, the extracted mask was a binary mask (black and white colored), unlike the marine vessel corrosion masks which were colored masks. Third, the patching size was 360 x 360 pixel which is compatible with the original image size and can be extracted without leaving any missing spots.

Thenceforth, after making such an adjustment, the results were superior. Figure 51 contains the graph of the results after applying the mentioned adjustments. As it can be seen, the graphs have begun to look normal and logical, as all the bias and variance have been reduced. The first graph in figure 51 is the loss function graph, showing the value of the loss function per epoch. Showing the improvement of the gained weights, illustrating how close it is to the weights of the actual image. The second graph shows the accuracy of the model per epoch. In the second graph it can be observed that there is some divergence between the training accuracy and the validation accuracy. It is expected to be due to a simple error that might occur because

of the number of training epochs or the number of images in the dataset is low. In addition, the accuracy and the IoU have increased from the previous trials, as the accuracy have reached to 85% and the IoU have reached to 35%, meaning that the applied adjustments were on point.

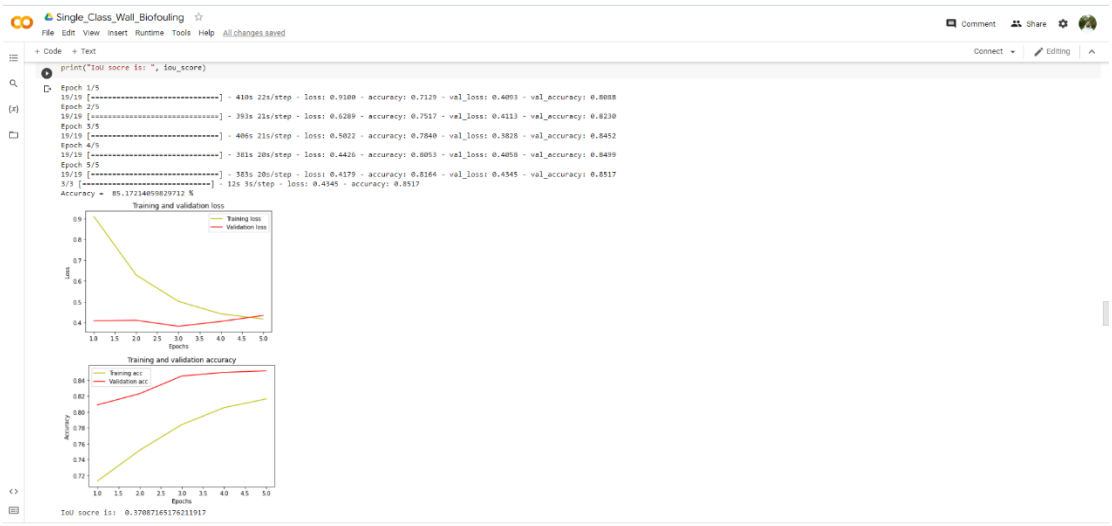


Figure 51 Results of the fourth training trial.

The saved model after the training stage is used to make sure that the model is effective, it has been applied on a set of images that has not been included in the training dataset. Figure 52 shows one of the images that has been predicted by the saved model. It can be seen that accuracy of the model is high with a minimal error percentage. This is considered to be an outstanding result based on the number of images that were used to train the model and the number of epochs in the training session.

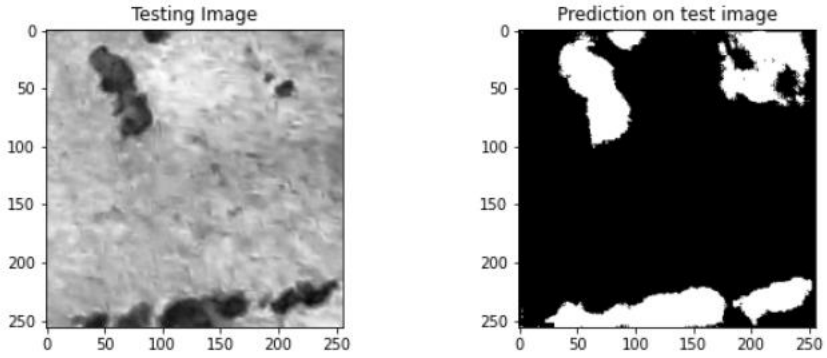


Figure 52 Results of the prediction process using the saved model.

After converting the model into (.onnx) file type as mentioned in chapter 3 and deploying it on jetson nano to apply the prediction on images. It worked perfectly and did not show any form of delay, figure 53 and figure 54 shows the prediction outcome using the same prediction code that was used on google colab, but now it was applied and relayed on the jetson nano GPU.

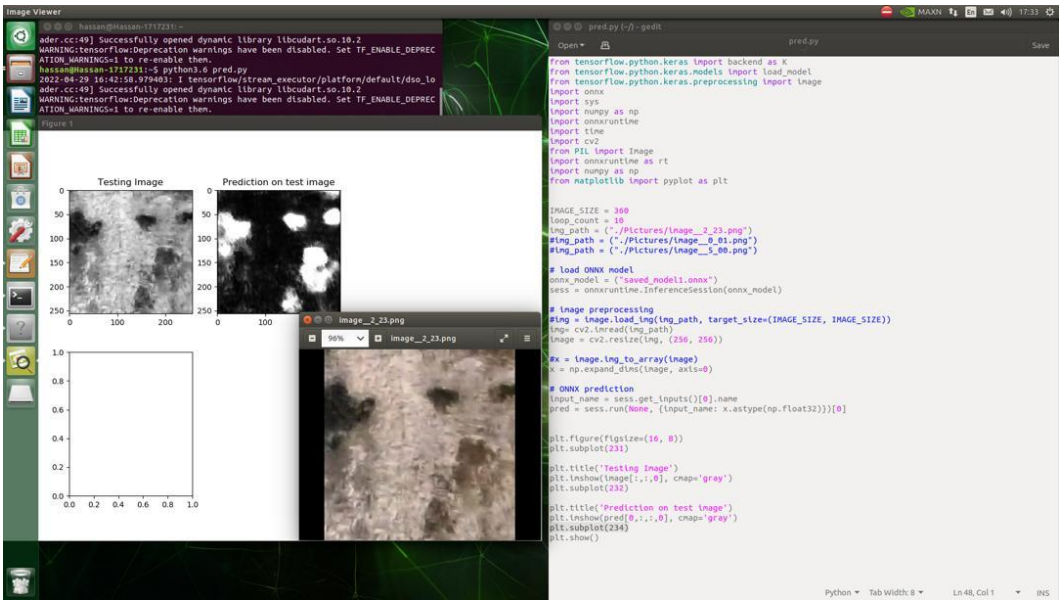


Figure 53 Normal image prediction on Jetson nano

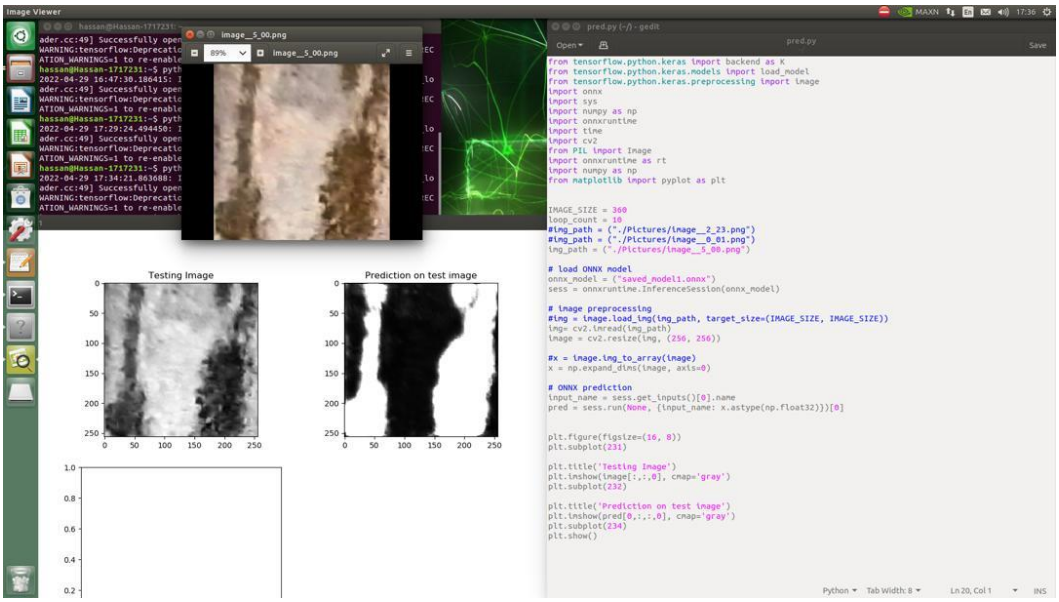


Figure 54 Normal image prediction on Jetson nano

In order to improve and enhance the experience of detecting wall fungus. A real-time semantic segmentation has been added to further simulate the process of the visual inspection system. The only difference between this process and the previous one, is that the input in the previous process was an image, but in this process the input is a webcam frame. As shown in figure 55, this is a simple demonstration of the system using the camera and predicting the wall fungus on one of the images.

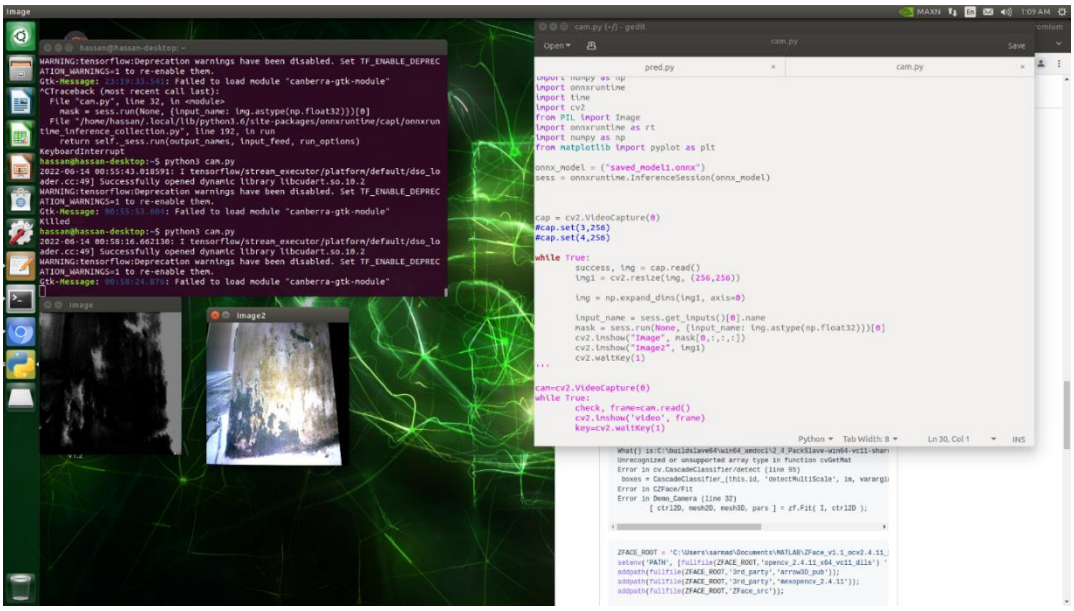


Figure 55 Real-Time video prediction on jetson nano

4.3 DISCUSSION AND LIMITATION

Overall, the process of building a deep learning model is based on the purpose of the model. As each data type may require a specific conditions and rules that must be followed to make sure that the model operates perfectly. To begin with, the first step of building the model is to collect the data and it must be collected in different variation to make it a robust model. Also, the collected data must mimic the same conditions of the purpose that it will be used for. As any changes in one of the conditions in the prediction stage, the model might not be able to recognize the object it is looking for. The second step is to annotate the collected data. Those two stages require patience as it is one of the most time-consuming stages, and any error may affect the outcome in the training process. Therefore, these two processes must be done cautiously. Then the data preparation process takes place, as the data is then normalized and patched to fit capabilities of the system to be trained. After the training session the model is ready to be used or retrained on new image dataset.

There were many limitations in this project which hindered the progress of making the model. The first limitation is the unavailability of the biological fouling of the ship hull. This was due to some technical issues that occurred in the company at that time, which made it difficult for the diving team to collect some images to be used on this project. Therefore, one of the closest situations or conditions that appears to be similar to ship hull biological fouling is the wall fungus. Supported by the fine-tuning part in the chapter 2. The model can be easily transferred into ship hull biological fouling segmentation due to its similarity in shape and color. The second limitation is that the progress on the master students was obstructed as one of the had to leave. Therefore, it was difficult to implement this project's model into their robot structure. The third limitation is the used components as it was underqualified, and the

results would have been better if the used components in the data collection and the model deployment were better.

5 CHAPTER 5

5.1 CONCLUSION

In conclusion, all objectives have been successfully achieved, which is to investigate and study the suitable detection architecture for underwater environment. Based on the literature review and the methodology it can be concluded that biological fouling requires the segmentation method to segment the dirty spots on the ship hull. In addition, the second objective, developing a marine vessel hull visual inspection system using deep learning. As the built system has been already installed on the embedded system and ready to be integrated and attached into the main robot. Finally, the third objective, validating the integration of biofouling detection. Has been shown in chapter 4, as the system have managed to identify a single wall fungus class from the background.

5.2 FUTURE WORK

In future work the system will include the real data of biological fouling, thenceforth, the same procedure will be followed to create the deep learning model. A better component will be used to enhance the whole process of training and prediction. In addition, the underwater prototype will be built, all the mentioned filtering techniques will be used on the upcoming underwater dataset as it might not be as clear as the wall fungus dataset, and the system will use a multiclass model which include biological fouling, corrosions, and color loss technical issues to enhance the performance of the visual inspection.

6 REFERENCES

(4) What is barrel distortion in photography? - Quora. (n.d.). Retrieved January 25, 2022, from https://www.quora.com/What-is-barrel-distortion-in-photography

Amer, K. O., Elbouz, M., Alfalou, A., Brosseau, C., & Hajjami, J. (2019). Enhancing underwater optical imaging by using a low-pass polarization filter. Optics Express, 27(2), 621. https://doi.org/10.1364/oe.27.000621

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (n.d.). YOLOv4: Optimal Speed and Accuracy of Object Detection. Retrieved November 26, 2021, from https://github.com/AlexeyAB/darknet.

Chliveros, G., Tzanetatos, I., & Kamzelis, K. (2021). MaVeCoDD Dataset: Marine Vessel Hull Corrosion in Dry-Dock Images. 1. https://doi.org/10.17632/RY392RP8CJ.1

Coraddu, A., Lim, S., Oneto, L., Pazouki, K., Norman, R., & Murphy, A. J. (2019). A novelty detection approach to diagnosing hull and propeller fouling. Ocean Engineering, 176(January), 65–73. https://doi.org/10.1016/j.oceaneng.2019.01.054

explain fully convolutional network and Unet - 知乎. (n.d.). Retrieved January 24, 2022, from https://zhuanlan.zhihu.com/p/65398511

Faster RCNN Object detection. Introduction | by Achraf KHAZRI | Towards Data Science. (n.d.).

Retrieved January 25, 2022, from https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4

Flemming, H.-C., & Flemming, H.-C. (n.d.). Microbial Biofouling: Unsolved Problems, Insufficient Approaches, and Possible Solutions. Springer Series on Biofilms, 5. https://doi.org/10.1007/978-3-642-19940-0_5

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (n.d.). Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5). Retrieved January 25, 2022, from http://www.cs.berkeley.edu/˜rbg/rcnn.

GoPro HERO 8 Black More Leaked Images & Specs – Camera News at Cameraegg. (n.d.). Retrieved January 22, 2022, from https://www.cameraegg.org/gopro-hero-8-black-more-leaked-images- specs/

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. ACL 2018

- 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers), 1, 328–339. https://doi.org/10.18653/v1/p18-1031

Introducing Image Segmentation At Labelbox. (n.d.). Retrieved January 25, 2022, from https://labelbox.com/blog/introducing-image-segmentation/

Jian, M., Qi, Q., Dong, J., Yin, Y., Zhang, W., & Lam, K. M. (2017). The OUC-vision large-scale underwater image database. Proceedings - IEEE International Conference on Multimedia and Expo, 1297–1302. https://doi.org/10.1109/ICME.2017.8019324

Jionghui, R., Wenming, Y., Linqiang, W., Liao, Y., Zhang, W., Jiang, D., Wang, W., & Brambilla, G. (2015). Using the Combination Refraction-reflection Solid to Design Omni-directional Light Source Used in Underwater Wireless Optical Communication. 9679, 96790–96791. https://doi.org/10.1117/12.2199222

Li, C., Guo, C., Ren, W., Cong, R., Hou, J., Kwong, S., & Tao, D. (2020). An Underwater Image Enhancement Benchmark Dataset and beyond. IEEE Transactions on Image Processing, 29, 4376–4389. https://doi.org/10.1109/TIP.2019.2955241

Li, C. Y., Guo, J. C., Cong, R. M., Pang, Y. W., & Wang, B. (2016). Underwater image enhancement by Dehazing with minimum information loss and histogram distribution prior. IEEE Transactions on Image Processing, 25(12), 5664–5677. https://doi.org/10.1109/TIP.2016.2612882

Oliveira, D., Larsson, A. I., & Granhag, L. (2018). Biofouling The Journal of Bioadhesion and Biofilm Research Effect of ship hull form on the resistance penalty from biofouling Effect of ship hull form on the resistance penalty from biofouling. https://doi.org/10.1080/08927014.2018.1434157

Rolan IPX8 18000lm 500M Flashlight Waterproof Lamp Light Underwater Outdoor Camp | Shopee Malaysia. (n.d.). Retrieved January 25, 2022, from https://shopee.com.my/Rolan-IPX8- 18000lm-500M-Flashlight-Waterproof-Lamp-Light-Underwater-Outdoor-Camp- i.120984801.7215299891

Ronneberger, O., Fischer, P., & Brox, T. (n.d.). U-Net: Convolutional Networks for Biomedical Image Segmentation. Retrieved January 25, 2022, from http://lmb.informatik.uni-freiburg.de/

Train Custom Data · ultralytics/yolov5 Wiki. (n.d.). Retrieved January 25, 2022, from https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

ultralytics/yolov5: YOLOv5 in PyTorch > ONNX > CoreML > TFLite. (n.d.). Retrieved January 25,

2022, from https://github.com/ultralytics/yolov5

Vu, T., Luong, M.-T., Le, Q. v., Simon, G., & Iyyer, M. (2021). STraTA: Self-Training with Task Augmentation for Better Few-shot Learning. http://arxiv.org/abs/2109.06270

Wide-Angle Photography Underwater - Underwater Photography Guide. (n.d.). Retrieved January 26, 2022, from https://www.uwphotographyguide.com/wide-angle-photography

Wu, J. (2018). Complexity and accuracy analysis of common artificial neural networks on pedestrian detection. MATEC Web of Conferences, 232. https://doi.org/10.1051/matecconf/201823201003

Xu, X., Shi, B., Gu, Z., Deng, R., Chen, X., Krylov, A. S., & Ding, Y. (2019). 3D No-Reference Image Quality Assessment via Transfer Learning and Saliency-Guided Feature Consolidation. IEEE Access, 7, 85286–85297. https://doi.org/10.1109/ACCESS.2019.2925084

Zhu, Y., Zeng, T., Liu, K., Ren, Z., & Lam, E. Y. (2021). Full scene underwater imaging with polarization and an untrained network. Optics Express, 29(25), 41865. https://doi.org/10.1364/oe.444755

APPENDIX A

```
PROJECT_PATH="/content/drive/MyDrive/MVC3"
#Installing Libraries
import os
import tifffile as tiff
from importlib.machinery import SourceFileLoader
#SourceFileLoader( 'simple_unet_model', os.path.join(PROJECT_PATH,
 'simple_unet_model.py')).load_module()
SourceFileLoader( 'simplest_unet_model', os.path.join(PROJECT_PATH
, 'simplest_unet_model.py')).load_module()
import random
#from simple_unet_model import simple_unet_model   #Use normal une
t model
from simplest_unet_model import simplest_unet_model
from keras.utils.np_utils import normalize
import cv2
from PIL import Image
import numpy as np
from matplotlib import pyplot as plt
#Calling images and masks from Google Drive
image_directory = '/content/drive/MyDrive/MVC3/patches/images/'
mask_directory = '/content/drive/MyDrive/MVC3/patches/masks/'

#Setting image size
SIZE = 360
image_dataset = []
mask_dataset = []
#loading images
images = os.listdir(image_directory)
images.sort()
for i, image_name in enumerate(images):
    if (image_name.split('.')[1] == 'png'):
        #print(image_directory+image_name)
        image = cv2.imread(image_directory+image_name, 0)
        image = Image.fromarray(image)
        image = image.resize((256, 256))
        image_dataset.append(np.array(image))
#loading masks
masks = os.listdir(mask_directory)
masks.sort()
for i, image_name in enumerate(masks):
    if (image_name.split('.')[1] == 'tiff'):
```

```
        image = cv2.imread(mask_directory+image_name, 0)
        image = Image.fromarray(image)
        image = image.resize((256, 256))
        mask_dataset.append(np.array(image))

#Normalize images
image_dataset = np.expand_dims(np.array(image_dataset), 3)
#D not normalize masks, just rescale to 0 to 1.
mask_dataset = np.expand_dims(np.array(mask_dataset),3) /255.
#split data into train, test, validate.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(image_dataset,
 mask_dataset, test_size = 0.10, random_state = 0)

X_train_quick_test, X_test_quick_test, y_train_quick_test, y_test_
quick_test = train_test_split(X_train, y_train, test_size = 0.9, r
andom_state = 0)
#Calling the model architecture and passing the values
def get_model():
    return simplest_unet_model(IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS
)

model = get_model()
#Training
history = model.fit(X_train, y_train,
                    batch_size = 32,
                    verbose=1,
                    epochs=5,
                    validation_data=(X_test, y_test),
                    shuffle=False)
#plot the training and validation accuracy and loss at each epoch
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# acc = history.history['acc']
acc = history.history['accuracy']
# val_acc = history.history['val_acc']
val_acc = history.history['val_accuracy']

plt.plot(epochs, acc, 'y', label='Training acc')
```

```
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
y_pred=model.predict(X_test)
y_pred_thresholded = y_pred > 0.4

intersection = np.logical_and(y_test, y_pred_thresholded)
union = np.logical_or(y_test, y_pred_thresholded)
iou_score = np.sum(intersection) / np.sum(union)
print("IoU socre is: ", iou_score)
```

APPENDIX B

```
from keras.models import Model

from keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D, concatenate, Conv2DTranspose, BatchNormalization, Dropout, Lambda

from keras import backend as K

def simplest_unet_model(IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS):

#Build the model

    inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))

    #s = Lambda(lambda x: x / 255)(inputs)   #No need for this if we normalize our inputs beforehand

    s = inputs


    #Contraction path

    c1 = Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(s)

    c1 = Dropout(0.1)(c1)

    c1 = Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(c1)

    p1 = MaxPooling2D((2, 2))(c1)


    c2 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(p1)

    c2 = Dropout(0.1)(c2)
```

```
    c2 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(c2)

    p2 = MaxPooling2D((2, 2))(c2)


    c3 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(p2)

    c3 = Dropout(0.2)(c3)

    c3 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(c3)

    p3 = MaxPooling2D((2, 2))(c3)


    c4 = Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(p3)

    c4 = Dropout(0.2)(c4)

    c4 = Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(c4)

    p4 = MaxPooling2D(pool_size=(2, 2))(c4)


    c5 = Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(p4)

    c5 = Dropout(0.3)(c5)

    c5 = Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(c5)


    #Expansive path

    u6 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(c5)

    u6 = concatenate([u6, c4])

    c6 = Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(u6)

    c6 = Dropout(0.2)(c6)

    c6 = Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(c6)


    u7 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(c6)

    u7 = concatenate([u7, c3])
```

```
    c7 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(u7)

    c7 = Dropout(0.2)(c7)

    c7 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(c7)


    u8 = Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(c7)

    u8 = concatenate([u8, c2])

    c8 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(u8)

    c8 = Dropout(0.1)(c8)

    c8 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(c8)


    u9 = Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same')(c8)

    u9 = concatenate([u9, c1], axis=3)

    c9 = Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(u9)

    c9 = Dropout(0.1)(c9)

    c9 = Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same')(c9)


    outputs = Conv2D(1, (1, 1), activation='sigmoid')(c9)


    model = Model(inputs=[inputs], outputs=[outputs])

    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    model.summary()


    return model
```

APPENDIX C

```
PROJECT_PATH="/content/drive/MyDrive/MVC3"
```

```
import os
import tensorflow as tf
import random
from importlib.machinery import SourceFileLoader
SourceFileLoader( 'simplest_unet_model', os.path.join(PROJECT_PATH, 'simplest_u
net_model.py')).load_module()
from simplest_unet_model import simplest_unet_model
from keras.utils.np_utils import normalize
import cv2
from PIL import Image
import numpy as np
from matplotlib import pyplot as plt
def get_model():
    return simplest_unet_model(256, 256, 3)

model = get_model()

model.load_weights('/content/drive/MyDrive/MVC3/dirts4.hdf5')
image = cv2.imread(r'/content/drive/MyDrive/MVC3/prediction/image__1_23.png',1)
image = cv2.resize(image, (256, 256))
image_norm = np.expand_dims(np.array(image),0)
prediction = (model.predict(image_norm) > 0.2).astype(np.uint8)

plt.figure(figsize=(16, 8))
```

```
plt.subplot(231)


plt.title('Testing Image')

plt.imshow(image[:,:,0], cmap='gray')

plt.subplot(232)


plt.title('Prediction on test image')

plt.imshow(prediction[0,:,:,0], cmap='gray')
```

APPENDIX D

```
pip install onnxruntime
pip install -U tf2onnx
!git clone https://github.com/onnx/tensorflow-onnx

PROJECT_PATH="/content/drive/MyDrive/multiclass_dirts&cracks/dirts
4.hdf5"
python -m tf2onnx.convert --PROJECT_PATH tensorflow-model-path --
output model.onnx
```