Spring 5-2024

Autonomous Fish Identification for the Remotely Operated Vehicle Control System

Jacob Wildes
University of Maine - Main, jacob.c.wildes@maine.edu

AUTONOMOUS FISH IDENTIFICATION

FOR THE REMOTELY OPERATED VEHICLE CONTROL SYSTEM

by

Jacob Wildes

A Thesis Submitted in Partial Fulfillment
of the Requirements for a Degree with Honors
(Computer Engineering)

The Honors College

University of Maine

December 2024

Advisory Committee:

Vikas Dhiman, PhD., Assistant Professor of Electrical and Computer Engineering, Advisor

Prabuddha Chakraborty, PhD., Assistant Professor of Electrical and Computer Engineering

Melissa Ladenheim, PhD., Associate Dean of The Honors College

Gregory Studer, PhD., Staff at The Advanced Structures and Composites Center

ABSTRACT

A Remotely Operated Vehicle was designed, constructed, and programmed as a senior design project in Electrical and Computer Engineering by Dyllon Dunton and Jacob Wildes. The system was intended to be an alternative means to inspect the underside of ships. Given the small footprint of the system, it can be easily extended into other applications. In this thesis project, the observation system is modified to detect if a fish is present or not, classify the species, localize where in the image the fish is, and mask the fish by separating fish pixels from non-fish pixels. Additionally, the original design will be analyzed and compared with the new system. Finally, existing issues with both systems and next steps are discussed. Open-source solutions were systematically searched, and several candidate solutions were found. The candidates were then evaluated in terms of ease of use and similarity to the goals of this project. Eventually, custom solution using the YOLOv8 backbone, and a custom labeled dataset was developed which achieved a precision of 86.5%, and a mAP-50 of 91%.

DEDICATION

To my family and friends for supporting me through the entire process. I could not have done it without the support of each and every one of you.

ACKNOWLEDGEMENTS

Dyllon Dunton – For having my back all throughout our time as undergraduates, and for being part of the senior project.

Dr. Vikas Dhiman – For taking me under his wing early in my undergraduate career and supporting me through the rest of the way.

My Honors Thesis Committee – Without the support of all of you this project would not have gotten to the stage that it has.

Dr. Bob Klose – For supporting me through the complexities of academia the last four years.

My Hannaford Crew – I could not have asked for better people in my corner. The outpouring of support for all of my pursuits and letting me vent about the complexities of navigating the academic world.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

INTRODUCTION

Native Maine fish are at constant risk of being outcompeted and run out by invasive fish, such as the Bluegill infestation in Savade Pond discovered in 2017 [1]. During a fishing excursion, wildlife biologists discovered over 30 Bluegill fish, while only two native fish were caught. In 2018 the biologists returned and discovered that in a mere few hundred yards of habitat, dozens of Bluegills had amassed, and the native Pumpkinseed sunfish had been entirely run out of the area. The Maine Department of Inland Fisheries and Wildlife (MDIFW) is unsure of when the Bluegills were first introduced, but the invasion began years ago, undetected [2]. To address this problem, the design of an underwater ROV capable of detecting invasive fish early is described. Other than the MDIFW, it can be used by other lake goers should there be concern of an infestation. This spreads the effort of verifying infestations can be spread to those who use the watersheds as well, not just wildlife authorities. This increases the potential to catch these silent infestations before they grow to the point that native species are run out of their habitat.

The ROV was a senior design project designed by Dyllon Dunton and Jacob Wildes in the Electrical and Computer Engineering department at the University of Maine at the start of the Spring 2023 semester [3]. The project was designed to dive underneath ships and inspect hulls for damage, mitigating the hazard of human divers or the cost of putting a ship in dry dock. During the research phase of the project, it became clear that commercially available ROV systems were incredibly expensive. As such, cost cutting decisions were made that limited what the ROV could accommodate. Most notably, the sensing equipment was extremely limited, using a cheap IMU, budget lights

and a cheap camera. These hardware limitations directly impact the primary focus of this thesis project. The initial ROV was only capable of passing images from the camera to the operator with the goal of simply allowing the operator to decide if the hull was in operating condition. This project aims to address invasive fish detection by using the existing ROV platform and extend it to identify fish while still maintaining the core hardware components. Such an extension requires determining what new functionalities are needed, and how to mesh them with the existing system. Additionally, ethical concerns such as privacy and security will be explored. Finally, the two designs will be compared with each other for similarities, differences, and drawbacks.

The original design of the ROV included two primary systems – the controller hardware and software which consisted of a Raspberry Pi 4B and an STM32L452RE-P microcontroller with custom software, and the submarine hardware and software which also featured a Raspberry Pi 4B, as well as an STM32L4R5ZI microcontroller. For the purpose of this project, only the controller hardware and software received any modification, as the submarine hardware and software was primarily under Dyllon Dunton's purview and are therefore outside the scope of this thesis project.

This thesis paper is divided into seven sections. In the second section, a review of relevant literature will be discussed to provide pertinent technical information and context. Subsequently, ethical implications of the ROV will be discussed, followed by the methods employed to design the neural network residing on the ROV for fish detection and identification. Results from the design will be covered next. Finally, future work and closing thoughts will be given to provide a thorough summary of the ROV project in totality.

LITERATURE REVIEW

Automated object identification is not a new field of study. Brief searches for "automated object detection" yield quite literally thousands of results. However, narrowing the focus down to specifically identifying fish brings the results down from 11,000+ to around 1,000 results (gathered from the University of Maine's OneSearch tool). This is still a vast amount of research related to fish detection, but this is an umbrella term. Many of these works are related to identifying fish once they are outside of their natural habitat. Another issue commonly faced, which will be discussed later in this section, is that some of the software that is more in line with this project is rather outdated. The goal of this project is to provide a model built using modern tools capable of identifying a fish in each image frame while the fish is still in its natural habitat.

Finding Related Work

One of the first steps taken to accomplish this project was a search for existing projects in an attempt to see what preexisting work was already available. In stark contrast to the amount of research done regarding fish identification, searches on the ubiquitous code sharing platform GitHub yielded a shockingly small number of existing projects to gain inspiration from, let alone potentially build from – that number being a meager 97 results. Of those results, some were incomplete or far removed from the focus topic. Of the yielded results, only 12 were deemed to have potential to mesh well with this project. These decisions were based off a few criteria. The first was documentation: the selected works either referenced one another or referenced a research paper (or a combination of both). Additionally, the documentation was deemed considerably better if it detailed the training methodology, or at minimum the data used by the author to

complete the project. Secondly, two features of GitHub – “Issues” and “Pull Requests” were reviewed. It is often difficult to determine if a system will work for a use case from simply reviewing the provided information. However, the “Issues” tab allows prospective users to review concerns other users have found, as well as actual issues with the project. Finally, age of the software was a determining factor. The tools used to build these projects are consistently being updated. As such, as age increases, the likelihood for incompatibility with some existing dependency also increases. For example, TensorFlow (a neural network programming library) underwent a massive overhaul in 2019 which effectively made all projects built using a previous version of TensorFlow break.

Neural networks are a set of algorithms that are modeled loosely after the human brain and designed to recognize patterns. There can be multiple layers in a neural network, and each layer consists of neurons. Broadly, artificial neurons work by receiving one or more input signals. These signals can be from other neurons, or raw data. They then perform calculations and send output signals to neurons in subsequent layers through a synapse. It is important to note that not all neurons must have synapses that feed into subsequent neurons. Additionally, there are weights associated with each neuron. These are factored into the calculations done in each neuron and dictate the value of the data provided by the neuron. As such, the weights are optimized during the network training. Increasingly complicated neural networks require more computational power. Since this project is built using an NVIDIA Orin Nano, the lack of computational power is compensated for by an increase in inference time. You Only Look Once (YOLO) is a neural network architecture that was introduced by Redmon et al in 2016 [4]. Since its introduction there have been several iterations which have improved the overall

implementation and architecture. The latest version, YOLOv8, by Ultralytics, was used. The architecture of the initial version is shown in Figure 1.

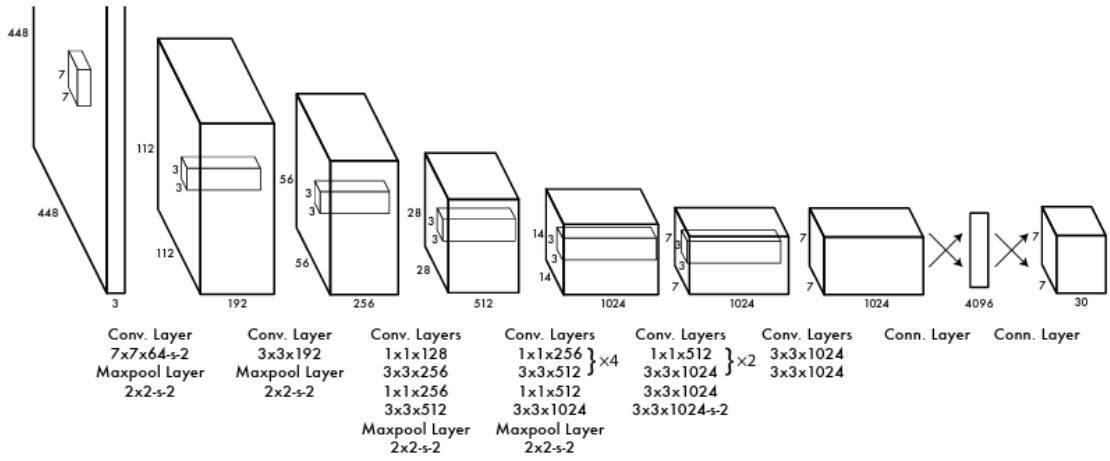

Figure 1. YOLO Architecture developed by Redmon et al [4]

As shown in the above figure, the YOLO architecture is quite extensive. From left to right, the input image is 448 pixels wide, 448 pixels tall, and 3 pixels (red, green, and blue) deep. This is then convolved and maxpooled to extract features from the image. A two-dimensional (2D) convolution operating between finite dimensional kernel weights $g \in \mathbb{R}^{K \times L}$ and a finite image $f \in \mathbb{R}^{I \times J}$ is denoted as $f * g \in \mathbb{R}^{I-K+1, J-L+1}$ and defined as

$$(f * g)(x, y) = \sum_{i=x-1}^{x-1+K} \sum_{j=y-1}^{y-1+L} f(i, j) g(x - i, y - j)$$

$$\text{for all } x \in \{1, \dots, I - K + 1\}, y \in \{1, \dots, J - K + 1\}$$

Equation 1. 2D Convolution

Normally these images have a third dimension which is the channel dimension (red, green, blue). In most neural network libraries, the convolution kernel over the third dimension is the same as the channel dimension. If the input is $f \in \mathbb{R}^{I \times J \times C}$, the kernel weights are $g \in \mathbb{R}^{K \times L \times C}$ where C is the number of input channels, then the output

$f * g \in \mathbb{R}^{I-K+1 \times J-L+1}$ is two-dimensional,

$$(f * g)(x, y) = \sum_{i=x-1}^{x-1+K} \sum_{j=y-1}^{y-1+L} \sum_{c=1}^{C} f(i, j, c) g(x-1, y-j, c)$$

$$\text{for all } x \in \{1, \dots, I-K+1\}, y \in \{1, \dots, J-L+1\}$$

Equation 2. 2D Convolution with multiple input channels

To generate multiple output channels, as many kernels as output channels needed are initialized. Let C_0 be output channels, and C_i be input channels. Kernel weights are then initialized of size $g \in \mathbb{R}^{C_0 \times K \times L \times C_i}$ and the input image is of size $f \in \mathbb{R}^{I \times J \times C_i}$. The output size is $f * g \in \mathbb{R}^{I-K+1 \times J-L+1 \times C_o}$

$$(f * g)(x, y, c_o) = \sum_{i=x-1}^{x-1+K} \sum_{j=y-1}^{y-1+L} \sum_{c_i=1}^{C} f(i, j, c_i) g(c_o, x-i, y-j, c_i)$$

$$\text{for all } x \in \{1, \dots, I-K+1\}, y \in \{1, \dots, J-K+1\}, c \in \{1, \dots, C_o\}$$

Equation 3. 2D Convolution with multiple input and output channels

This step is responsible for extracting a feature map of the input. The next step is the maxpooling step, which helps make the network resilient to small shifts in the position of the features in an image. Additionally, it reduces the number of parameters in the network. The maxpooling 2D layer takes in an image of size $f \in \mathbb{R}^{I \times J \times C}$ where *C* is the channel dimension and a kernel size of K, L, and outputs another image $M \in \mathbb{R}^{\lfloor \frac{I}{K} \rfloor \times \lfloor \frac{J}{L} \rfloor}$. In this case, $\lfloor x \rfloor$ denotes the largest integer smaller than x. The maxpooling operation is

$$M(x, y, c) = max_{(i,j) \in W_{x,y}} f_c(i, j, c)$$

$$for\ all\ x \in \left\{1, \dots, \left\lfloor \frac{I}{K} \right\rfloor\right\}, y \in \left\{1, \dots, \left\lfloor \frac{J}{L} \right\rfloor\right\}\ where$$

$$\mathcal{W}_{x,y} = \{(i,j)|(x-1)K \leq i \leq xK, (y-1)L \leq j \leq yL, i \leq I, j \leq J, i \in \mathbb{Z}, j \in \mathbb{Z}\}$$

Equation 4. 2D Maxpooling

In the above equation, M_c represents the output of the maxpooling layer, for a particular channel, *c.* f_c is the input from the three-dimensional feature map, and W_{xy} is the window of the pooling operation over the input. The maximum value in the window is selected as it slides across the input. The convolution process is done a total of ten times in the data flow from input to output. After each a leaky rectified linear unit (ReLU) activation function is applied. The maxpooling process is applied four times. Finally, there are two fully connected layers at the end which allows for non-linear combinations of high-level features pulled from the convolutional layers. The fully connected equation is shown in Equation 5.

$$\mathrm{y} = \mathrm{Wx} + \mathrm{b}$$

Equation 5. Fully Connected Layers

In the above equation, x is the input. In this case, that input is flattened into a vector since it is three-dimensional. y is the output vector, b is the bias vector. Each element in the output vector is the sum of the elements in x with the bias term. Convolution is a linear operation. Combinations of linear functions result in linear results, and solely linear models are not as effective in image recognition tasks as non-linear models. Additionally, real-world data is typically not linear. To fix this, activation functions like the leaky ReLU after each convolution introduce some non-linearity. Activation functions introduce the non-linearity.

The cumulative knowledge gained from these individual works was crucial in the development of the custom model. Of the five related works, Calvo [5], Anar [6], and

Govostes [7] all relied on the third version of YOLO as the basis of the model. As discussed by Abdulla [8] and empirically confirmed by Calvo, it is absolutely necessary to have a large number of images per class when training a multi-class neural model. In order to make the dataset, an online tool was used to annotate and augment the data. Additionally, the commonality of architecture among the works was heeded, and the YOLO architecture was selected, as it is claimed to be a state-of-the-art model which is fast, accurate, and easy to use [9].

METHODOLOGY

In order to accomplish this project, there were three primary steps that had to be taken. First, a framework for the neural model had to be chosen. Next, a practical dataset had to be found, as well as a method to label and outline the fish. Finally, a training method had to be chosen and executed. The detail behind each process is detailed below.

Selecting a Neural Model Framework

As discussed in the Literature Review, there is not much recent work regarding underwater fish identification. In addition, much research involves having the fish out of its natural habitat or is based on much older frameworks. Since this system is designed with intent to identify fish in their natural habitat as quickly as possible, it was necessary to find a model framework capable of accomplishing that task. As previously discussed, YOLO was the most popular framework used in other projects. As a result, the newest version, YOLOv8 was used in this project. Accompanying this version are a multitude of different pretrained model weights. YOLO offers a variety of operational modes: detection, segmentation, pose, oriented detection, and classification. Under each of these categories, there are five options for each mode. There are nano, small, medium, large, and extra-large models. These models tradeoff accuracy with speed and size. According to Jocher et al., the smallest YOLO model had a single frame prediction time when running on a Central Processing Unit (CPU) of 96.1 ms [10]. In order to maintain a somewhat visually appealing framerate when making predictions, a rate of 15 frames per second was chosen. This means that a frame must be processed approximately every 67 ms. Unfortunately, the desired framerate was unobtainable, at least with a CPU only. The speed of predictions on individual frames increases significantly with the introduction of

a graphics processing unit (GPU). However, the YOLO models were evaluated on an NVIDIA A100 GPU – hardware significantly more powerful than what would be available on a single board computer such as a Raspberry Pi or Jetson Orin Nano. Currently a Raspberry Pi runs the controller. As a result, the less accurate, yet faster model was chosen.

Acquiring a Dataset

Training neural models well requires a vast array of images, as discussed in the Literature Review. Additionally, since the ROV was not yet fit for underwater deployment and there was no budget to purchase underwater photography equipment, online databases of fish images were the next best alternative. This was not the ideal outcome, as the images are not necessarily local to Maine, and the orientation and visibility of the fish are unable to be controlled. However, it was the second-best option for getting fish images. After extensive research, the dataset hosting website Kaggle had the most applicable dataset developed by Lampa et al [11] of fish, which was chosen for this project. This was done because the fish are not local to Maine – originating from Cabuyao City, Philippines. The more universal fish - such as perch and tilapia - were selected from the dataset to train with as they are also located in the United States, and more particularly, Maine. Figure 2 shows a sample image of the Perch and more general Fish class from the dataset.

Figure 2. On left, the general "Fish" class. On right, the "Perch" Class

Annotating Data

After completing the task of acquiring data, the next step was to prepare it for use in the model. This included labeling each individual image and drawing a bounding box around each fish in each image. Given that there were so many images to annotate (approximately 700 raw images in the final working dataset), standard methods for annotating using tools such as AnyLabeling [12], OpenLabeling [13], and Labellmg [14] would be far too time consuming. Alternatively, there are expensive automated tools such as Adobe Acrobat, but it is unsuitable for reaching the proper format the neural network seeks when training. Similarly to how Kaggle [15] functions as a dataset hosting website, there is another website dedicated to generating neural network training datasets called Roboflow [16]. This tool vastly sped up the process of readying the dataset as it offered some “smart” bounding options which would automatically place bounding boxes over a desired area with relatively good accuracy. Figure 3 shows an example of both the Fish and Perch classes and their annotations.

Figure 3. "Fish" Class Annotation on Left, "Perch" Class Annotation on Right

This allowed a significant speedup in annotation time, as in most cases the only step was to add a label to the image and continue to the next one. Next, Roboflow facilitated further augmentation of the images by randomly orienting images to encourage a more robust model which is less prone to overfitting. It also publicly hosted the annotated dataset for easy training and validation access.

Training the Model

After determining the type of framework to use as well as constructing a dataset complete with annotations, the final step was training the model. As previously discussed, the dataset suffered from being on the small side, and unfortunately not entirely diverse. As a result, the Adaptive Moment Estimation (Adam) optimizer introduced by Kingma and Ba in 2016 [17] was used. Optimizers are used to optimize weights of a neural network to minimize the loss function. This was in part because it is popular, but also because it is less sensitive to hyperparameters. This saved time in tuning the hyperparameters, especially where the project is currently at a smaller scale. Additionally, it converges quickly, resulting in faster training. This in turn helps to protect the model from overfitting, where the model does not learn any meaningful patterns and instead memorizes the training data.

Using a dropout rate of 25% was useful, as it allowed the model to understand what a fish is from only viewing small parts of it by randomly deactivating neurons in each layer. This made the model much more robust, as it would allow it to potentially identify a fish even if a piece was missing. Additionally, a cosine annealing learning rate schedule was introduced in an effort to help the model escape local minima and explore different regions of loss. This creates a cyclic learning rate. It follows the cosine curve downward from the maximum learning rate until it reaches the minimum learning rate, where the cycle continues. This helps the model escape local minima and potentially find a more optimal solution.

RESULTS

The model performs generally as expected in real-world tests. As shown in Table 1, the model is reasonably precise, measuring 86.5% accurate during training and validation over 25 epochs. Figure 4 shows an example of the model working well.

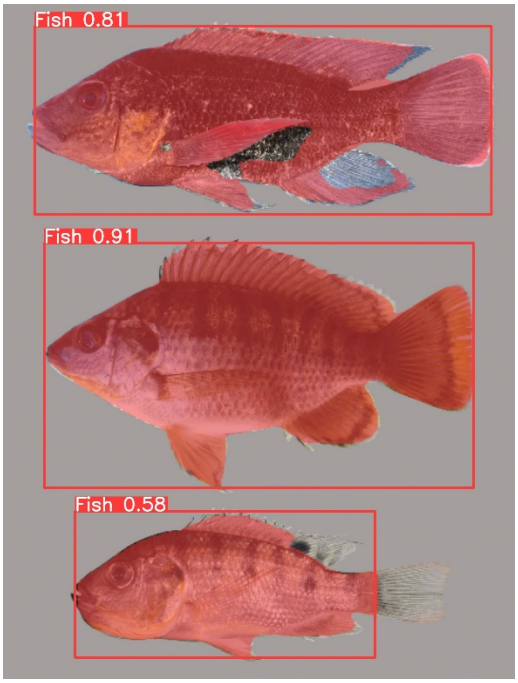

Figure 4. Model Accurately Identifying Fish.

As shown in Figure 4, the model works excellently on these fish which it has never seen previously. However, these fish are not in their natural habitat. The model is able to predict these fish very accurately because they are broadside to the camera, and all defining features of a fish are present. As discussed extensively to this point, the model struggles to identify fish that do not look similar to the fish that it was trained on. This is most certainly due to the fact that there simply were not diverse enough images for the model to train with. In order to remedy this, the dataset needs to be extensively added to.

Despite this, the model still functions well for a baseline prototype. Notice in Figure 5, the model struggles severely to predict any fish.

Figure 5. Model Unable to Detect Fish in Habitat.

These fish are "in situ," where the defining features are not readily obvious to the model, and as a result the ability for the model to accurately predict in this situation is severely hindered. Notice that the fish that it was only 28% certain was a fish is primarily uncovered. Its physical features are clear and unobstructed by any other fish in the image. Additionally, the fish are of similar color relative to themselves and to the environment, so the defining features are further lost. To expand upon the extensive addition to the dataset, images such as these are a must. Without them, the model will continue to suffer from being unable to reliably identify fish when they are not perfectly aligned with the camera.

It can be safely argued that the model does not overfit. Figure 6 shows the loss of the model in training and validation over a period of 25 epochs with an image batch size of eight.

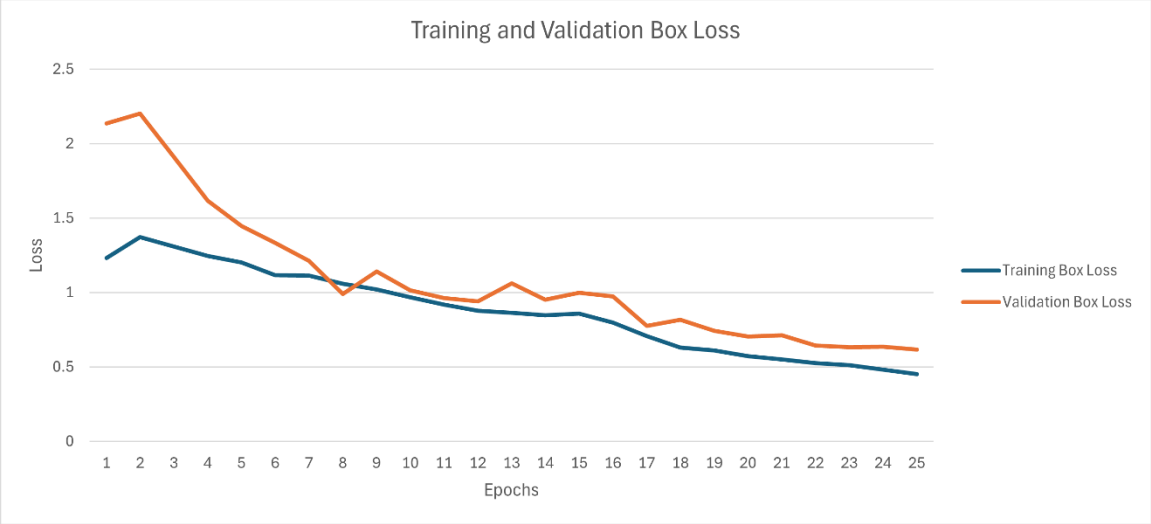

Figure 6. Box loss of the Model during Training and Validation

As shown in the above figure, both the training and validation losses trend downward, which is the desired result. Notice, however, that the loss is still decreasing. Further training would decrease loss and increase accuracy. Key signs of overfitting include the validation loss massively increasing while the training loss continues to decrease. In the figure above, after the sixth epoch, both the training and validation trend downward at a similar rate. As such, the model is suffering less from poor optimization, and more from a poorly constructed dataset. The data is not diverse enough to give the model enough data points to accurately infer fish in multiple different orientations. In the technical sense, the model does not overfit. It does, however, overfit in the sense that the data is biased. This means that this overfit cannot be solved with optimizations outside reworking the dataset.

ETHICAL ANALYSIS

This project is autonomous observation based, which leads to certain ethical questions. In order to better analyze the ethics of this project, some questions proposed by Green [14] are answered. Of note, there are questions of technical safety, transparency and privacy of the data being collected, beneficial use capacity, and malicious use capacity. It is imperative that the system be completely transparent about the data it collects, how and if it stores data, and the potential damages of a compromised system.

Under the umbrella of "technical safety" the goal is to assert that the neural network will function as detailed. If it does fail, answer what the fallout of a failure would entail. At the forefront, as will be discussed in greater detail in the Results section, the neural network does function as intended, albeit in a more limited capacity. If the network is shown an image of a fish, it is about 50% successful in identifying the fish as a fish, as shown in Figure 7.

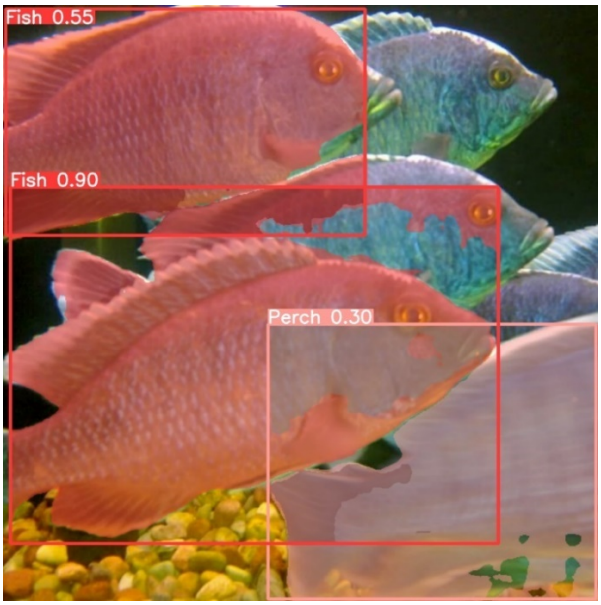

Figure 7. Fish being identified by the Model.

As shown in the above figure, the network is able to identify fish if the majority of the fish is visible and largely uncovered. The worst case scenario is that the model fails and displays some unexpected output. This model solely takes images in, and if applicable, draws a box and labels around a fish. If it were to fail, the worst thing that it could do is improperly label, improperly box, or simply not do anything. The system has no other data to work with than that in the image, so the operator is also safe from any harm in the network having a catastrophic failure.

Thankfully, this system has little foreseeable for malicious use. The neural network is incredibly naïve, able to only take images in, make a prediction based on what is in the image, and save the image. Malicious use of the system would struggle to do anything outside of its operating parameters. Assuming that this system was compromised, and the images were lost continuously, fish invasions would continue to remain undetected. As such, this system is designed to be used as one tool in a multifaceted approach at fish invasion detection. Pre-existing approaches to detection, such as electrofishing, are still imperative to employ.

Regarding data use and collection, this system solely saves images where a fish is predicted to be in the frame. If there is no fish, the image is not saved at all. Those images that are saved do not have any other information on them than what the model appends to it, such as that in Figure 2. Additionally, no data is transmitted. When implemented on the ROV, the entire system is disconnected from the internet, so any data transaction must be done locally.

This model is also entirely transparent. The method for fine tuning the model is entirely open source [19], and YOLO [10], the framework with which the model was

built on, is also entirely open source. The dataset found on Kaggle is also entirely open source [15]. Additionally, the dataset used to fine tune the model was used under the Community License Agreement – Sharing – Version 1.0. The method by which the fish images were collected is not publicized by the dataset authors, but permission to use the images is provided under the license agreement.

DISCUSSION

The autonomous ROV was a device intended to autonomously identify fish as it patrols an aquatic environment. Realistically, this system is capable of identifying fish, but the network falls victim to poor generality of the training data. As discussed in the Methodology section, there are not many applicable preexisting databases of fish images that are local to Maine. Gathering images manually is a long and strenuous task, and time was a major factor in the project. As such, fish that do not look similar to a tilapia or perch tend to go unnoticed by the model. Ideally, this model would have been built off a preexisting model, such as those built by the Fishial.AI group [20]. This would have provided a foundation for the model to start from, as opposed to a model built from scratch. Additionally, the training data would have been built off of fish images in situ, not those taken on dry land. As discussed in the Literature Review, water is a poor medium for identification. Water refracts light, and this is magnified by particularly contaminant filled watersheds, such as the Androscoggin River or Etna Pond. Additionally, environmental factors such as algal blooms were not taken into account when training the model. Each of these factors drastically impact the model's ability to make accurate predictions on images, as each frame has the potential to be obfuscated.

Since this project builds on top of the old system, it still shares the physical shortcomings of the initial ROV. Firstly, it is sparsely equipped. As discussed previously, cost was the determining factor in a lot of decisions made for the ROV. This is most apparent with the choice of lights and camera. The camera is a simple universal serial bus (USB) webcam with 720p resolution. This is fine for the model predicting, but a primary issue with the webcam is that anything that moves quickly across the frame is incredibly

blurry. The lights are more than bright enough, but the issue is that the team was not aware of the implications of light layout at the time of construction, so there is no empirical evidence that the lighting would benefit or detract from the model's ability to make inferences on an image. Barring an inertial measurement unit (IMU) used for stability control, there are no other sensors aboard the ROV.

A secondary issue is that the software was built to be modular. The hardware was built less so. This makes hardware changes a much more involved process. For example, the Raspberry Pi running the controller side is already under massive load from the controller graphical user interface (GUI) rendering as well as sending and receiving data to and from the ROV as shown in Figure 8. Alternative hardware choices such as phones, tablets, and laptops were decided against primarily due to the use of an ethernet for connection from controller to ROV. Phones and tablets can connect to an ethernet cable via dongle, but then the controller software would have to be ported from Linux to iOS or Android. Laptops could be used instead as well, but then a few possible issues arise; currently the ROV takes analog inputs to trim the propellers. Using a laptop without external interfaces would make the inputs entirely binary – on or off. Alternatively, if external interfaces are used, the user interface becomes clunky and usage environments for an ROV such as this do not lend themselves to spreading out with multiple wires being exposed. Finally, battery life can no longer be guaranteed. The laptop would be provided by the user, and it is entirely of unknown age and operating system. That being the case, the Raspberry Pi was decided upon since it can be purchased cheaply and loaded with the software designed prior to this project.

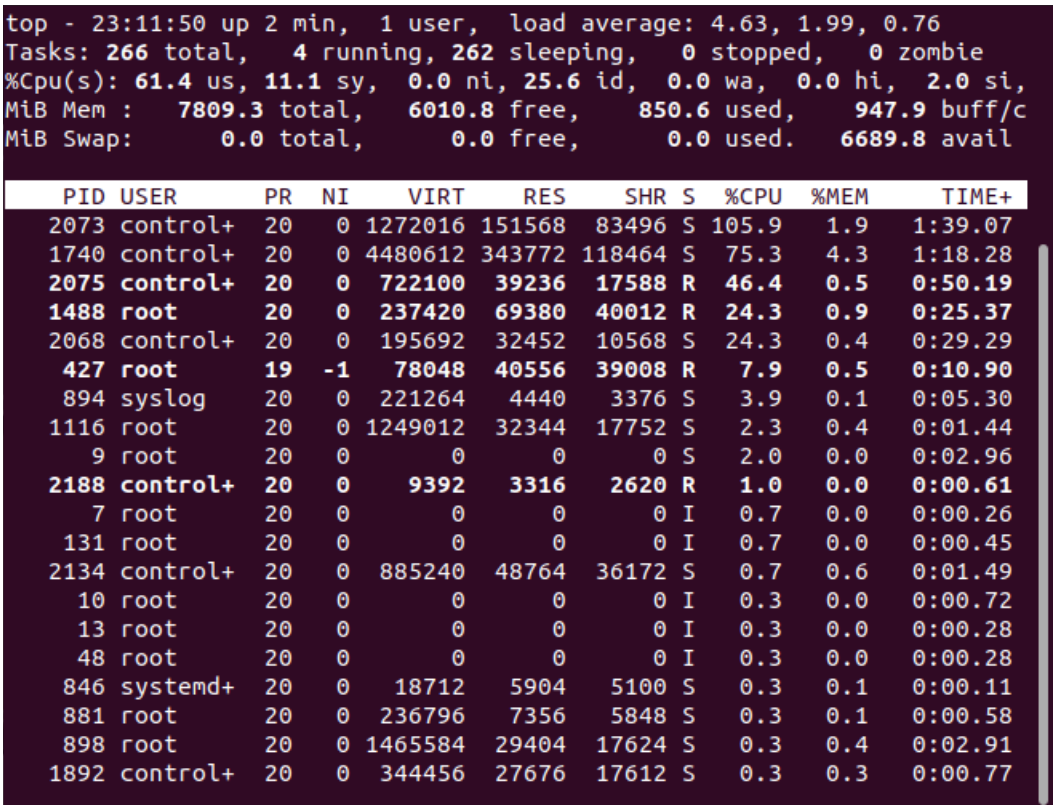

Figure 8. TOP Readout from the Controller Raspberry Pi

For context, the USER column shows multiple “control+” running programs. The controller software to run the ROV is run in the user space, and the username of the controller Raspberry Pi is called “controller.” The last three letters are truncated to a “+.” The data shown is slightly misleading. The Raspberry Pi has four cores, and each core represents 100% under the %CPU column. However, The GUI alone utilizes one core and then some, at 105.9%. This necessitates two hardware overhauls on the controller side. First, the Raspberry Pi would need to be swapped for a more powerful edge computing device, such as a NVIDIA Orin Nano. This device has a much higher power requirement, so the battery pack would need to be bulked out to support the new computer. This in turn would result in a new battery management system (BMS). All of this means the price of the system increases. In the case of the Orin Nano, significantly

so. The changes are going to be made to the controller instead of the ROV for a couple of reasons. With the addition of an Orin Nano, that would be the single most expensive piece of hardware in the entire system at almost $500. Since this is a budget project, placing such expensive equipment in an environment where water exposure is a constant threat is a poor idea at best. This decision sacrifices detection speed due to ethernet latency, but speed is less of a concern than system reliability and robustness.

CONCLUSION

There is a lot of future work left in this project. Firstly, the dataset needs to be massively overhauled to faithfully depict what the model would encounter in practice. This involves adding species of fish found in Maine watersheds, and introducing diverse photos of each species, as no two fish are identical. The data worked with is largely imperfect as it comes from outside of the geographical focus area. This step involves a potentially multi-season process, capturing images of fish from multiple different Maine watersheds. The most time consuming part of the process can be compared to fishing. Each fish species needs to have multiple images taken, and different times of day result in different fish species activity. Additionally, some fish species, like Brook Trout, are much more active during their spawning season from mid-November to early January [21]. Secondly, in order for the system to be fully integrated, the controller hardware needs to be overhauled to integrate the model with the ROV control system. However, a publicly available baseline model is available on Jacob's GitHub [19] to be extended to whatever needs necessary and has the potential to eventually better identify Maine local aquatic fauna at.

BIBLIOGRAPHY

[1] “An Undetected Fish Invasion | IFW Blogs,” www.maine.gov. https://www.maine.gov/ifw/blogs/mdifw-blog/undetected-fish-invasion (accessed Nov. 10, 2023).

[2] Maine Department of Inland Fisheries and Wildlife, “Fish Invasions Take A Toll On Native Fisheries”, 2016. https://www.maine.gov/ifw/blogs/mdifw-blog/fish-invasions-take-toll-native-fisheries (accessed Nov. 10, 2023).

[3] D. Dunton and J. Wildes, “Submersible Drone for Hull Inspections Senior Project Report,” 2024. https://github.com/jacobcwildes/Submarine_Capstone/blob/main/Capstone_Final.pdf (accessed Apr. 9, 2024).

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” 2016. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf (accessed Apr. 9, 2024).

[5] R. Calvo, “rocapal/fish_detection,” GitHub, Feb. 02, 2024. https://github.com/rocapal/fish_detection?tab=readme-ov-file (accessed Nov. 22, 2023).

[6] Y. C. Anar, “Fjolnirr/fish_tracker,” GitHub, Mar. 04, 2024. https://github.com/Fjolnirr/fish_tracker/tree/master (accessed Nov. 22, 2023).

[7] R. Govostes, “WHOIGit/FishDetector,” GitHub, Feb. 28, 2024. https://github.com/WHOIGit/FishDetector (accessed Oct. 23, 2023).

[8] Waleed Abdulla, "Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow", Github, 2017. https://github.com/matterport/Mask_RCNN (accessed Dec. 13, 2023).

[9] G. Jocher, A. Chaurasia, and J. Qiu, Ultralytics YOLO. 2023. https://github.com/ultralytics/ultralytics?tab=readme-ov-file (accessed Mar. 10, 2024).

[10] Ultralytics, “Instance Segmentation,” docs.ultralytics.com. https://docs.ultralytics.com/tasks/segment/ (accessed Apr. 7, 2024).

[11] M. D. Lampa, R. C. Librojo, and M. M. Calamba, ‘Fish Dataset’. Kaggle, 2022. https://www.kaggle.com/ (accessed Jan. 1, 2024).

[12] V. A. Nguyen, AnyLabeling - Effortless data labeling with AI support, Github. https://github.com/vietanhdev/anylabeling (accessed Mar. 2, 2024).

[13] Cartucho, “OpenLabeling: open-source image and video labeler,” GitHub, Nov. 30, 2022. https://github.com/Cartucho/OpenLabeling (accessed Mar. 2, 2024).

[14] “HumanSignal/labelImg,” GitHub, Aug. 17, 2023. https://github.com/HumanSignal/labelImg (accessed Mar. 2, 2024).

[15] ‘Kaggle’. [Online]. Available: https://www.kaggle.com (accessed Jan 1, 2024).

[16] Dwyer, B., Nelson, J., Hansen, T., et. al. (2024). Roboflow (Version 1.0) [Software]. Available from https://roboflow.com (accessed Mar. 12, 2024).

[17] D. P. Kingma and J. Ba, ‘Adam: A method for stochastic optimization’, arXiv preprint arXiv:1412. 6980, 2014. https://arxiv.org/pdf/1412.6980 (accessed Apr. 22, 2024).

[18] B. Green, “Artificial Intelligence and Ethics: Sixteen Challenges and Opportunities,” www.scu.edu, Aug. 18, 2020. https://www.scu.edu/ethics/all-about-ethics/artificial-intelligence-and-ethics-sixteen-challenges-and-opportunities/ (accessed Feb. 6, 2024).

[19] “GitHub - jacobcwildes/Submarine_Capstone at honors_thesis,” GitHub. https://github.com/jacobcwildes/Submarine_Capstone/tree/honors_thesis (accessed Apr. 10, 2024).

[20] F. A. Project, “fishial/fish-identification,” GitHub, Mar. 12, 2024. https://github.com/fishial/fish-identification/tree/main (accessed Nov. 21, 2023).

[21] “Brook Trout: Species Information: Fisheries: Fish & Wildlife: Maine Dept of Inland Fisheries and Wildlife,” www.maine.gov. https://www.maine.gov/ifw/fish-wildlife/fisheries/species-information/brook-trout.html (accessed Mar. 15, 2024).

[22] Juan Carlos Ovalle, Carlos Vilas, Luís T. Antelo, On the use of deep learning for fish species recognition and quantification on board fishing vessels, Marine Policy, Volume 139, 2022, 105015, ISSN 0308-597X, https://doi.org/10.1016/j.marpol.2022.105015. (https://www.sciencedirect.com/science/article/pii/S0308597X22000628) (accessed Nov. 10, 2023).

APPENDICES

Appendix A: Additional Results

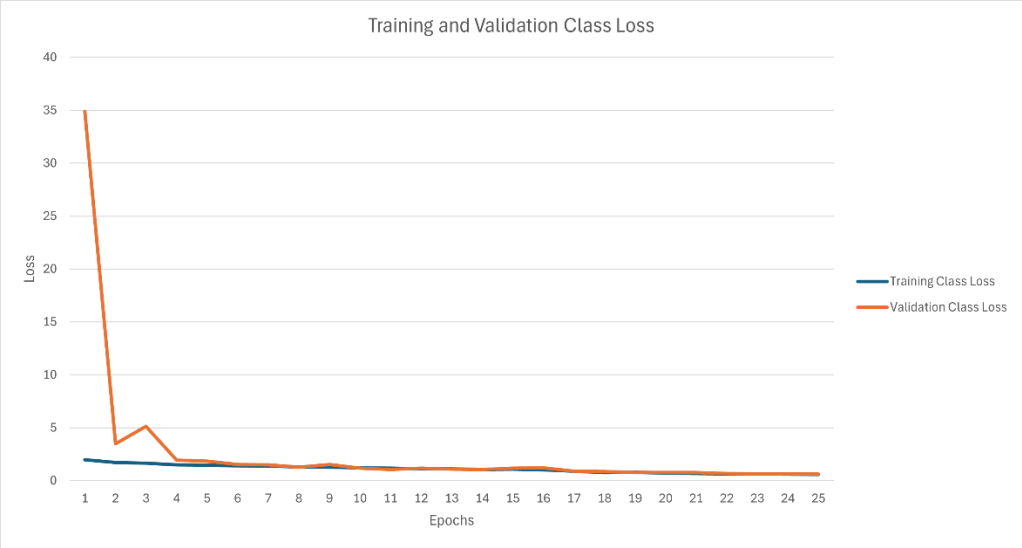

Figure 9. Class loss during Training and Validation

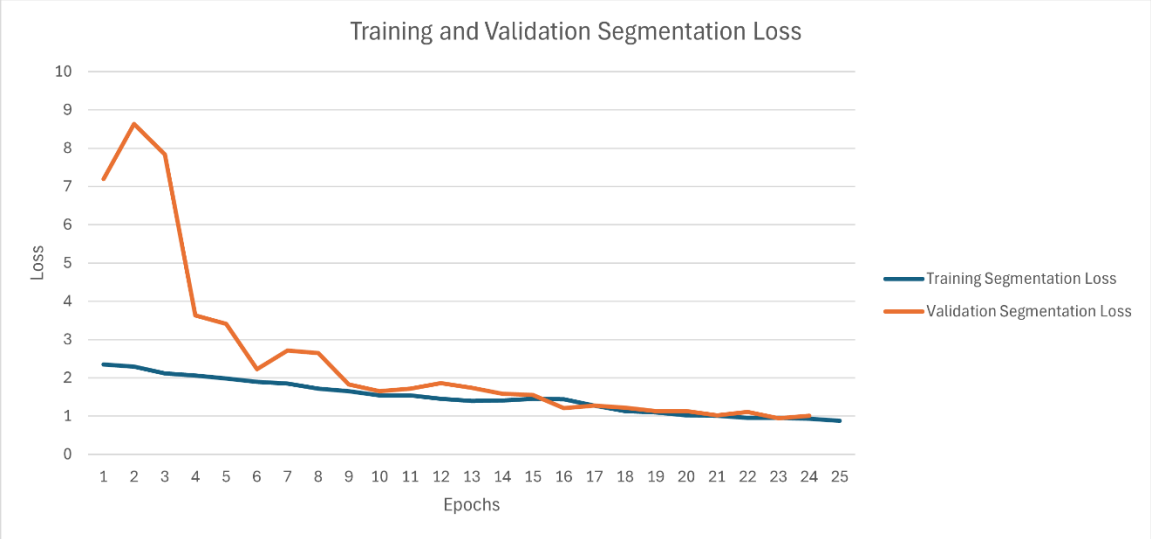

Figure 10. Segmentation loss during Training and Validation

Precision	Recall	mAP50	mAP50-95
.8649	.8876	.9106	.8031

Table 1. Performance Metrics

AUTHOR BIOGRAPHY

Jacob Wildes was born in Bangor, Maine on July 5th, 2002. He spent the entirety of his childhood living in Carmel, Maine and graduated from Hermon High School in 2020. He majored in computer engineering and minored in computer science. Jacob is an avid runner and outdoorsman. In 2019 he began working at Hannaford Supermarkets, and in 2022 he began working at the Advanced Structures and Composites Center. When not at work, he can be found with family or building model ships.

In the future, Jacob plans to wind down to one job and start his career in computer engineering, most likely in the field of embedded system software development or cluster computing.