

SCHOOL *of* BUSINESS AND TECHNOLOGY

Department of Engineering and Aviation Sciences

**Clam Activity Detection System**

**Shanice Nurse**

**Ashley Afueh**

Advisor: Dr. Zhang

April 23, 2020

Clam Activity Detection System

By

Shanice Nurse & Ashley Afueh

Submitted to the Department of Engineering and Aviation Sciences in partial fulfillment of the requirements for the degree of Bachelor of Science in Engineering at the

UNIVERSITY OF MARYLAND EASTERN SHORE

Date

The author hereby grants the University of Maryland Eastern Shore permission to reproduce and distribute publicly, paper and or electronic copies of this document in whole or in parts.

Authors Shanice Nurse, Ashley Afueh

Signature

Date

Department of Engineering and Aviation Sciences

Certified by

**List of Contents**

[**List of Contents 2**](#_30j0zll)

[**List of Figures 4**](#_1fob9te)

[**List of Tables 5**](#_3znysh7)

[**Abstract 6**](#_2et92p0)

[**1.**](#_tyjcwt) **Introduction 7**

[**1.1**](#_3dy6vkm) **Backgound/Motivation 7**

[**1.2**](#_4d34og8) **Objective 7**

[**1.3**](#_2s8eyo1) **Design Requirements 7**

[**1.4**](#_17dp8vu) **Design Constraints 7**

[**1.5**](#_3rdcrjn) **Design Methods 7**

[**2.**](#_26in1rg) **Project Description 9**

[**2.1**](#_lnxbz9) **System Description 9**

[**2.2**](#_35nkun2) **System Diagram 9**

[**2.3**](#_44sinio) **System Functions 9**

[**3.**](#_2jxsxqh) **Implementation Plan 11**

[**3.1**](#_z337ya) **Tasks 11**

[**3.2**](#_3j2qqm3) **Team Organization 12**

[3.2.1.](#_1y810tw) Responsibility of Team Member 1. 12

[3.2.2.](#_4i7ojhp) Responsibility of Team Member 2. 12

[**3.3**](#_2xcytpi) **Timeline/Milestones/Delivery Plan 12**

[**4.**](#_3whwml4) **Implementation 13**

[**4.1**](#_2bn6wsx) **Implementation of Task 1. 13**

[4.1.1.](#_qsh70q) Implementation of Subtask 1.1 13

[**4.2**](#_3as4poj) **Implementation of Task 1. 13**

[**5.**](#_1pxezwc) **Conclusion (Discussion and Future Plans) 14**

[**Acknowledgment 15**](#_2p2csry)

[**Appendix 16**](#_147n2zr)

[**A.**](#_3o7alnk) **Component Specs 16**

[1.](#_23ckvvd) Specs of Arduino Due 16

[2.](#_ihv636) Specs of Raspberry Pi 16

[**B.**](#_32hioqz) **Source Code. 16**

[1.](#_1hmsyys) Source Code of Graphic User Interface 16

[2.](#_41mghml) Source Code of Robotic Arm 16

[**REFERENCES 17**](#_2grqrue)

**List of Figures**

[Figure. 1.](#_1t3h5sf) Mussel Active Filtering (A), Voltage Output (B), and Mussel Resting (C). 6

[Figure. 2.](#_1ksv4uv) Design Methods. 7

**List of Tables**

[Table 1.](#_1ci93xb) Project Timeline and Delivery Plan 10

**Abstract**

By the end of the project, summarize the project into short text and put here (can be waived for Senior Design I). The abstract needs to be more than half page.

1. **Introduction**

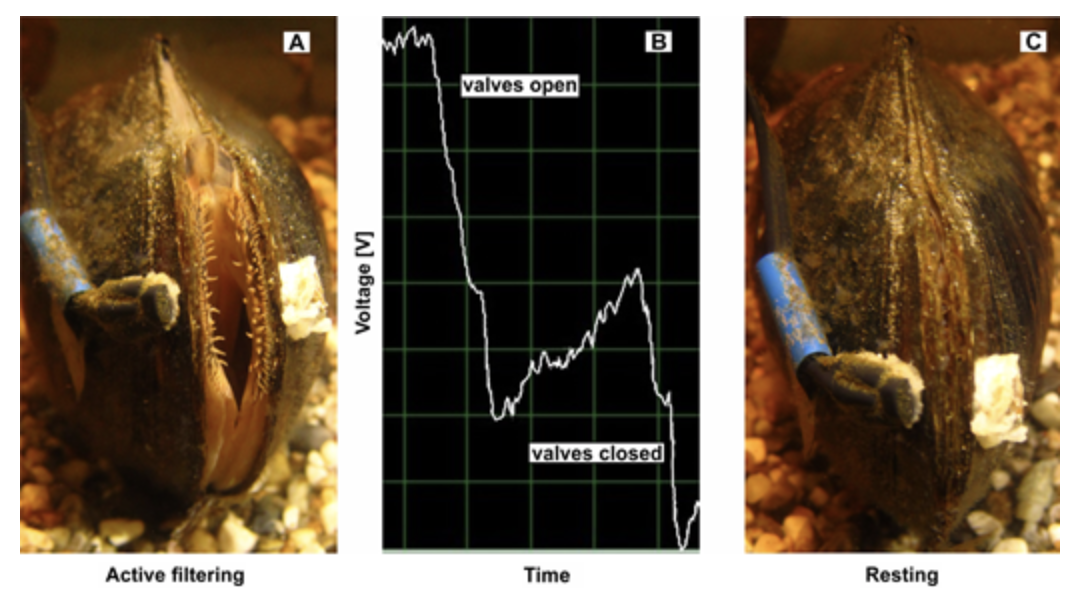
## 1.1 Background/Motivation

Clams provide a positive environment for the coastal and estuarine waters. Shellfish cultivating can give communities an assortment of environmental benefits, whose worth can be quantified. Clams have the ability to clean the water which they’ve grown in by filter feeding. By this way of feeding, clams filter phytoplankton, microorganisms, and detritus. By filtration, clams improve water quality by diminishing sediment loads and turbidity and expelling abundance supplements from inshore coastal waters. Clear water quality permits more daylight to infiltrate, which helps in the development of significant seagrasses and expands oxygen.

The capability of remotely monitoring the behavior of clams and their relationship with their habitat has restructured how ecologists conduct research. Shellfish cultivating can give nearby communities an assortment of environmental benefits, whose worth can be quantified. Freshwater mussels (Unionoida) are target species in aquatic conservation and belong to one of the fastest diminishing taxa globally (Bogan, 1993; Geist, 2010, 2015). Most recently, there has been a critical effort to put resources into improving the conception of the reasons for the decrease of Unionoida. Biologists have urged the discovery of a solution to observe the behavior, physiology, and ecological conditions experienced by creatures as they progress and communicate with their habitat. On account of the mechanical advances made in the course of the most recent 40 years, a wide range of approaches have been created to precisely comprehend the lives of creatures that go overlooked by the observation of an ecologist. Comprehensively, the strategies used to monitor clams are directly related to the "biotelemetry", which can be characterized as the remote detection of physiological, or behavioral activity information. It is effective to include biotelemetry into the study of clams in order to have a precise understanding of its preservation to the ecosystem. The standard amount of biotelemetry incorporated in research, differs enormously, from the utilization of data recorded, to transmitters that send data to either land-based receivers, or to a satellite circumnavigating the earth. For this reason, researchers and different articles suggest it useful to attach sensors to the mussels’ shell for ecological investigations and preservation ventures.

Various research projects range from, but are not limited to, mark-recapture studies to translocation and restocking programs, and mussel behavior research (Smith, Villella & Lemarié, 2003; Villella, Smith & Lemarié, 2004; Kurth *et al.*, 2007; Wilson *et al.*, 2011; Hartmann *et al.*, 2015). For instance, the controlled release of captive-reared freshwater pearl mussels (*Margaritifera margaritifera*) requires effective tools to monitor the success of such conservation projects (Gum, Lange & Geist, 2011). Adjoining sensors to a mussel may bring about strain for the mussel and increase endangerment to their vulnerability to harmful mixes from the glue. To connect an item includes the expulsion of the mussel from its condition, cleaning and drying the shell, applying the paste, squeezing the remote article against the shell and, for some paste', the procedure additionally includes keeping the mussel out of the water until the paste is dry. Currently, there are experimental studies of filtration-behavior and biological rhythm for freshwater mussels. Biologists perform their research by attaching objects to bivalve shells for conservation projects, mark-recapture studies, and behavioral analysis. The “marking” process is performed by using a magnet and a rubber-coated Hall sensor glued with cyanoacrylate adhesive attached to a mussel shell known as the “Anodontia Anatina.” These attachments were built to last for nine months after attachment. This system was designed for a filtration-behavior experiment, which identified a circadian rhythm. This experiment was performed by placing twenty-six mussels that were randomly selected were placed into two different closed recirculating aquarium systems. After allowing the mussels to adapt to their laboratory habitat for about 12 days, a Honeywell SS495A linear position Hall sensor (Honeywell, USA) and magnet were then attached for monitoring behavior. Additionally, this system allowed ecologists to observe the water quality parameters, dissolved oxygen (DO), pH and EC daily. Figure 1 depicts the measurement of a mussel’s filtration behavior using the change in proximity of a magnet (in right valve) to a Hall sensor (on left valve). In Figure 1, photograph B depicts a graph showing a measurement of the voltage output transduced by the Hall sensor when the mussel is active and resting.

Figure 1.



1. Mussel Active Filtering (A), Voltage Output(B), Mussel Resting(C).

This project is intended to assist in research for the Food & Science Department at the University of Maryland Eastern Shore. By developing a Clam Activity Detection System, a dataset from the activity of the clams will be derived from the activity observed without putting addition strain on the clams. This would be extremely beneficial to ecologist and researchers as they can remotely monitor precise data from their experiments. The dataset will be the basis for the development, evaluation, and use of the neural network developed. During this project, there will be models developed for photograph classification, and object detection to further investigate the activity of clams. By creating this development, the department will have a better insight of the clams’ behavioral activity with its environment. The Deep learning Convolutional Neural Network will be trained with an image classification model to observe each clam individually. This neural network is intended to monitor the clams’ amount of interaction, length of interaction, growth, and its overall adaptivity to its surroundings. This deep learning model will have the ability to integrate the feature extraction and classification process into a whole to record when the clam is most comfortable.

The purpose of this project is to develop a SMART system that will precisely monitor the activity and interaction of clams without the tedious strain imposed on their shell. By using deep learning approach to monitor the clam’s activity the data will be recorded with increased accuracy. As previously stated, there methods on the market that will accomplish similar tasks as our system. However, the methods do not offer the precision and comfortability our system can guarantee.

## Objective

The objective of this project is to create an automatic system that can precisely monitor and record activity of clams with 95% accuracy using a SMART design system.

## Design Requirements

1. The system will identify activity of each clam using a SMART system.
2. A time-lapse recording log will be implemented for visual data recording.
3. Access to a 24-hour data log will be offered with hourly updates.
4. Controlled parameters of the water quality along with the clam’s interaction to the environment will be accessed through a local network.

## Design Constraints

1. $200 budget
2. Light system is required for 24-hour monitoring

## Design Method

A system diagram for the design method can be seen in figure 2.

1. Design Method.

## Standards

List in this section all (industry) standards the project complied with.

1. **Project Description**

## System Description

This system intended to remotely monitor activity is controlled by a Deep Learning Convolutional Neural Network. For the project, the neural network is trained to identify various activities. A dataset will need to be prioritized with photographs of active clams and resting clams provided as a subset of photographs from a larger dataset. Using Keras the photos will be loaded to pre-process into standard directories. Python will then be used to create directories and subdirectories for both the “train” and “test” directories. After testing the dataset, a baseline Convolutional Neural Network will need to be developed. Different convolutional layers within the baseline model will be tested for accuracy. About three different VGG-based architecture models will be implemented to improve the performance by increasing the model capacity. This technique will alter the performance by allowing the model to learn features that will maximize the training dataset. After the process of model improvements, a final model configuration is selected and adopted. A final model normally includes all available data; the combination of train and test datasets. Through a web application, researchers will be able to view recorded datasets.

## System Diagram (or Flow Chart)

A screenshot of a cell phone

Description automatically generated

1. System Flow Chart

## System Functions

This system uses a camera which will monitor the clam’s various activity; either resting or active. This information is then communicated to the Raspberry Pi. In the microcomputer, there is a system design which utilizes Artificial Intelligence and a Convolution Neural Network to process the data extracted from the images recorded. The data is then transmitted to a local network for visual recordings.

1. **Implementation Plan**

## Tasks

## Team Organization

## Tasks

* Task 1: Lab Setup
* Subtask 1.1: Mound Cameras to tank
* Subtask 1.2: Mound Sensors to tank
* Subtask 1.3. Setup Raspberry pi for experiment
* Subtask 1.4: Setup any more other configurations
* Task 2: Design software for practice image identification
  + Subtask 2.1: Load Data
  + Subtask 2.2: Generate Dataset
  + Subtask 2.3: Create Model
  + Subtask 2.4: Save/Load Model
  + Subtask 2.5: Save/Load Model (again)
* Task 3: Design CNN (convolutional neural network)
  + Subtask 3.1: Preprocess design
  + Subtask 3.2: Feed neural network
  + Subtask 3.3: Obtain intrinsic dataset of picture
  + Subtask 3.4: Save pictures and results in database
* Task 4: Design software for clam identification
* Subtask 4.1: Photo dataset preparation
* Subtask 4.2: Develop a baseline CNN (convolutional neural network) model
* Subtask 4.3: Improvement on CNN model
* Subtask 4.4 Develop Transfer learning
* Subtask 4.5: Finalize CNN model

* Task 5: Design software for clam activity level
* Subtask 5.1: Photo dataset preparation
* Subtask 5.2: Develop a baseline CNN (convolutional neural network) model
* Subtask 5.3: Improvement on CNN model
* Subtask 5.4 Develop Transfer learning
* Subtask 5.5: Finalize CNN model
* Task 6: Data Organization for Database
* Subtask 6.1: Sorting data files
* Subtask 6.2: Writing data to a file
* Subtask 6.3: Modeling Data for database
* Subtask 6.4: Work with data constructed
* Subtask 6.5: Ensuring all cameras are synchronizing with little to no error
  + Subtask 6.6: Create code to send pics from Camera to raspberry pi
  + Subtask 6.7: Create code to send sensor data to raspberry pi
* Task 7: Send Obtained Data to Cloud
* Subtask 7.1: Create channel for data
* Subtask 7.2: Create API key
* Subtask 7.3: Create python code for data
* Task 8: Generate Website Database
* Subtask 8.1: Organize layout for website
* Subtask 8.2: Sort data for general clam activity and activity level
* Subtask 8.3: Sort data for individual clam activity and activity level
* Subtask 8.4: Create figures for data
* Subtask 8.5: Run website, fix errors
* Task 9: System evaluation
* Subtask 9.1: Determine accuracy levels of 4 cameras and of each sensor.
* Subtask 9.2: Test entire system
* Subtask 9.3: Fix any errors
* Subtask 9.4: Evaluate system
* Subtask 9.5: Refine system (if necessary)

**3.2: Team Organization**

Team Member 1: Shanice Nurse

Team Member 2: Ashley Afueh

***3.2.1: Responsibility of Team Member 1:***

Task 1: Lab Setup (half)

Task 3: Design CNN (convolutional neural network)

Task 5: Design software for clam activity level

Task 7: Send obtained data to cloud

Task 8: Generate website database (half)

Task 9: System Evaluation (half)

***3.2.2: Responsibility of Team Member 2:***

Task 1: Lab Setup (half)

Task 2: Design software for practice image identification of stars

Task 4: Design software for clam identification

Task 6: Data organization for database

Task 8: Generate website database (half)

Task 9: System Evaluation (half)

1. **Project Timeline and Delivery Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| Time | Task | Comment | Responsible Personnel |
| Week 1 | Subtask 1.1, 1.2 | Task 1: Lab Setup | Shanice Nurse, Ashley Afueh |
| Week 2 | Subtask 1.1, 1.2 |
| Week 3 | Subtask 1.3, 1.4 |
| Week 4 | Subtask 1.3 |
| Week 5 | Subtask 1.4 |
| Week 6 | Subtask 2.1 | Task 2: Design software for practice image identification | Ashley Afueh |
| Week 7 | Subtask 2.1 |
| Week 8 | Subtask 2.2, |
| Week 9 | Subtask 2.2, 2.3 |
| Week 10 | Subtask 2.3, 2.4 |
| Week 11 | Subtask 2.4, 2.5 |
| Week 12 | Subtask 3.1 | Task 3: Design CNN | Shanice Nurse |
| Subtask 3.2 |
| Subtask 3.3 |
| Subtask 3.4 |
| Week 13 | Subtask 4.1-4.2 | Task 4: Design software for clam identification | Ashley Afueh |
| Subtask 4.3 |
| Subtask 4.4 |
| Subtask 4.5 |
| Week 14 | subtask 5.1-5.3 | Task 5: Design software for clam activity level | Shanice Nurse |
| Week 15 | Subtask 5.4-5.5 |
| Week 16 | Subtask 6.1 – 6.2 | Task 6: Data organization for Database | Ashley Afueh |
| Subtask 6.3 – 6.4 |
| Week 17 | Subtask 7.1 – 7.3 | Task 7: Send obtained data to Cloud | Shanice Nurse |
| Week 18 | Subtask 7.4 – 7.5 |
| Week 19 | Subtask 8.1 – 8.3 | Task 8: Create Website Database | Ashley Afueh, Shanice Nurse |
| Week 20 | Subtask 8.4 – 8.5 |
| Week 21 | Subtask 9.1 – 9.2 | Task 9: System Evaluation | Ashley Afueh, Shanice Nurse |
| Subtask 9.3 – 9.4 |

1. **Implementation**

We have used Keras image preprocessing layers for image standardization and data augmentation.

**Intro: Installation and Setup**

We used the following command to install the Keras.

pip install keras

Train and Test folders are placed in the same folder as the python file directory.

NeuralNetwork

->DeepLearning.py

->train

->car

->planes

->random (other)

->test

->car

->planes

->random (other)

**Subtask 2.1:** **Load the data**

This step filtered out badly-encoded images that do not feature the string "JFIF" in their header. It iterated through each file from the “train” folder, and the sub folders were considered as the classes. Classes were used for the classification.

*num\_skipped = 0*

*for folder\_name in ("cars", "planes", "random (other)"):*

*folder\_path = os.path.join("train", folder\_name)*

*for fname in os.listdir(folder\_path):*

*fpath = os.path.join(folder\_path, fname)*

*try:*

*fobj = open(fpath, "rb")*

*is\_jfif = tf.compat.as\_bytes("JFIF") in fobj.peek(10)*

*finally:*

*fobj.close()*

*if not is\_jfif:*

*num\_skipped += 1*

*# Delete corrupted image*

*os.remove(fpath)*

*print(fpath)*

*print("Deleted %d images" % num\_skipped)*

**Subtask 2.2: Load the data**

This step was used to generate the dataset using the following code segment.

*image\_size = (224, 224)*

*batch\_size = 32*

*train\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory("train",validation\_split=0.5,subset="training",seed=1337,image\_size=image\_size,batch\_size=batch\_size)*

*val\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory("train",validation\_split=0.5,subset="validation",seed=1337,image\_size=image\_size,batch\_size=batch\_size)*

It's a good practice to artificially introduce sample diversity by applying random yet realistic transformations to the training images, such as random horizontal flipping or small random rotations. This helps expose the model to different aspects of the training data while slowing down overfitting.

*data\_augmentation = keras.Sequential(*

*[*

*layers.experimental.preprocessing.RandomFlip("horizontal"),*

*layers.experimental.preprocessing.RandomRotation(0.1),*

*]*

*)*

**Subtask 2.3 - Create Model**

This step created the Kera`s Model using the following code segment. The data augmentation was applied to the input dataset. Adding 2 layers of convolution layers. Convolution layers are the core components of the Neural Network. The first layer has 32x32 kernel size and the second layer has 64x64 kernel size. (This can be adjusted in order to achieve better result).

*def make\_model(input\_shape, num\_classes):*

*inputs = keras.Input(shape=input\_shape)*

*# Image augmentation block*

*x = data\_augmentation(inputs)*

*# Entry block*

*x = layers.experimental.preprocessing.Rescaling(1.0 / 255)(x)*

*x = layers.Conv2D(32, 3, strides=2, padding="same")(x)*

*x = layers.BatchNormalization()(x)*

*x = layers.Activation("relu")(x)*

*x = layers.Conv2D(64, 3, padding="same")(x)*

*x = layers.BatchNormalization()(x)*

*x = layers.Activation("relu")(x)*

*previous\_block\_activation = x # Set aside residual*

*for size in [128, 256, 512, 728]:*

*x = layers.Activation("relu")(x)*

*x = layers.SeparableConv2D(size, 3, padding="same")(x)*

*x = layers.BatchNormalization()(x)*

*x = layers.Activation("relu")(x)*

*x = layers.SeparableConv2D(size, 3, padding="same")(x)*

*x = layers.BatchNormalization()(x)*

*x = layers.MaxPooling2D(3, strides=2, padding="same")(x)*

*# Project residual*

*residual = layers.Conv2D(size, 1, strides=2, padding="same")(*

*previous\_block\_activation*

*)*

*x = layers.add([x, residual]) # Add back residual*

*previous\_block\_activation = x # Set aside next residual*

*x = layers.SeparableConv2D(1024, 3, padding="same")(x)*

*x = layers.BatchNormalization()(x)*

*x = layers.Activation("relu")(x)*

*x = layers.GlobalAveragePooling2D()(x)*

*if num\_classes == 2:*

*activation = "sigmoid"*

*units = 1*

*else:*

*activation = "softmax"*

*units = num\_classes*

*x = layers.Dropout(0.5)(x)*

*outputs = layers.Dense(units, activation=activation)(x)*

*return keras.Model(inputs, outputs)*

**Subtask 2.4: Save/Load Model**

This step was used to save/load the trained model. Since the saved model was present in the same folder as the python file, we loaded the saved model with the highest Epoch value.

*epochs = 2*

*savedFileName = "save\_at\_{0}.h5".format(epochs)*

*if(not(path.exists(savedFileName))):*

*model = make\_model(input\_shape=image\_size + (3,), num\_classes=3)*

*keras.utils.plot\_model(model, show\_shapes=True)*

*callbacks = [*

*keras.callbacks.ModelCheckpoint("save\_at\_{epoch}.h5"),*

*]*

*model.compile(*

*optimizer=keras.optimizers.Adam(1e-3),*

*loss="binary\_crossentropy",*

*metrics=["accuracy"],*

*)*

*model.fit(*

*train\_ds, epochs=epochs, callbacks=callbacks, validation\_data=val\_ds,*

*)*

*model = keras.models.load\_model(savedFileName)*

**Step 5: Save/Load Model (again)**

This step classified the test images with either saved model or the newly created model. Using the predict method from the model, prediction value was used to classify the image into the corresponding class and the count was incremented.

*carCount = 0*

*planeCount = 0*

*othersCount = 0*

*for folder\_name in ("cars", "planes", "random (other)"):*

*folder\_path = os.path.join("train", folder\_name)*

*for fname in os.listdir(folder\_path):*

*fpath = os.path.join(folder\_path, fname)*

*img = keras.preprocessing.image.load\_img(*

*fpath, target\_size=image\_size*

*)*

*try:*

*img\_array = keras.preprocessing.image.img\_to\_array(img)*

*img\_array = tf.expand\_dims(img\_array, 0) # Create batch axis*

*predictions = model.predict(img\_array)[0]*

*if(predictions[0] > predictions[1] and predictions[0] > predictions[2]):*

*carCount = carCount + 1*

*elif(predictions[1] > predictions[0] and predictions[1] > predictions[2]):*

*planeCount = planeCount + 1*

*elif(predictions[2] > predictions[0] and predictions[2] > predictions[1]):*

*othersCount = othersCount + 1*

*finally:*

*fobj.close()*

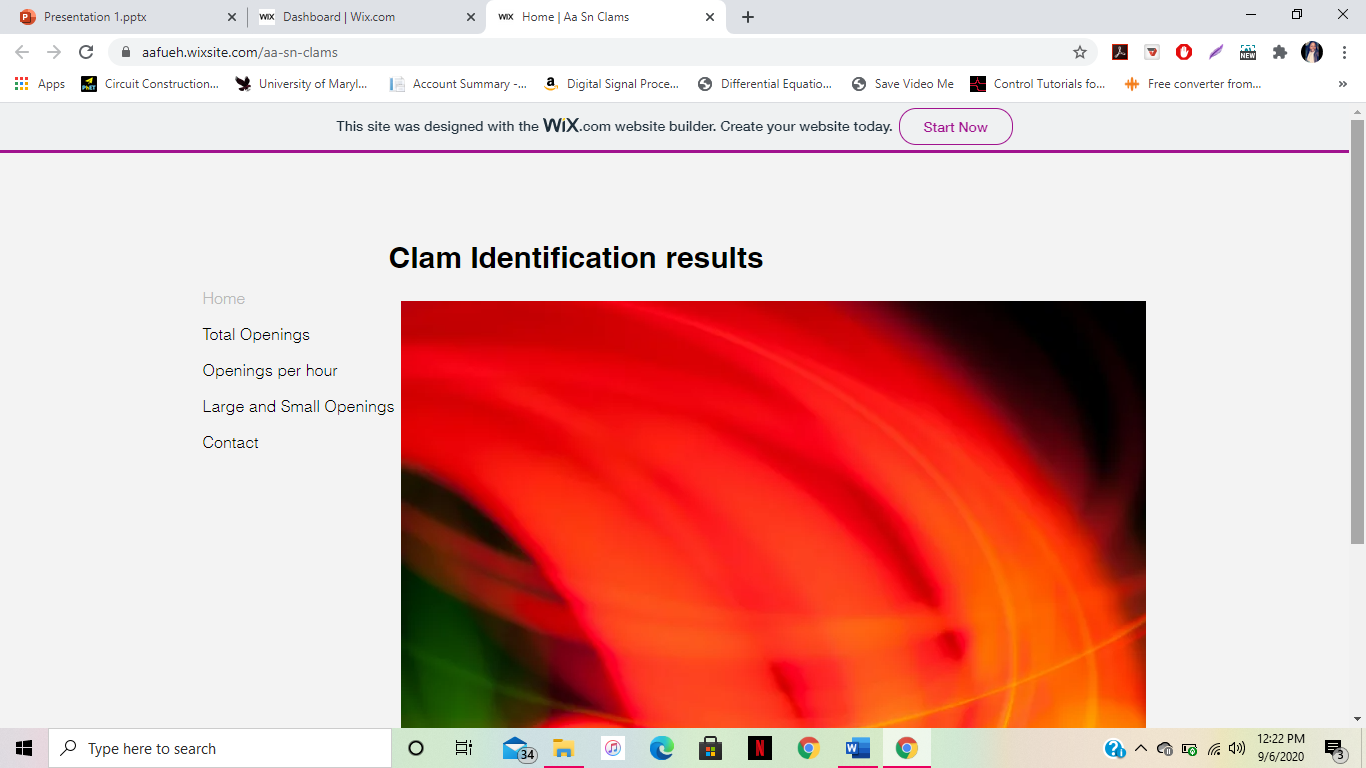
*print("car Count : ", carCount)*

*print("plane Count : ", planeCount)*

*print("others Count : ", othersCount)*

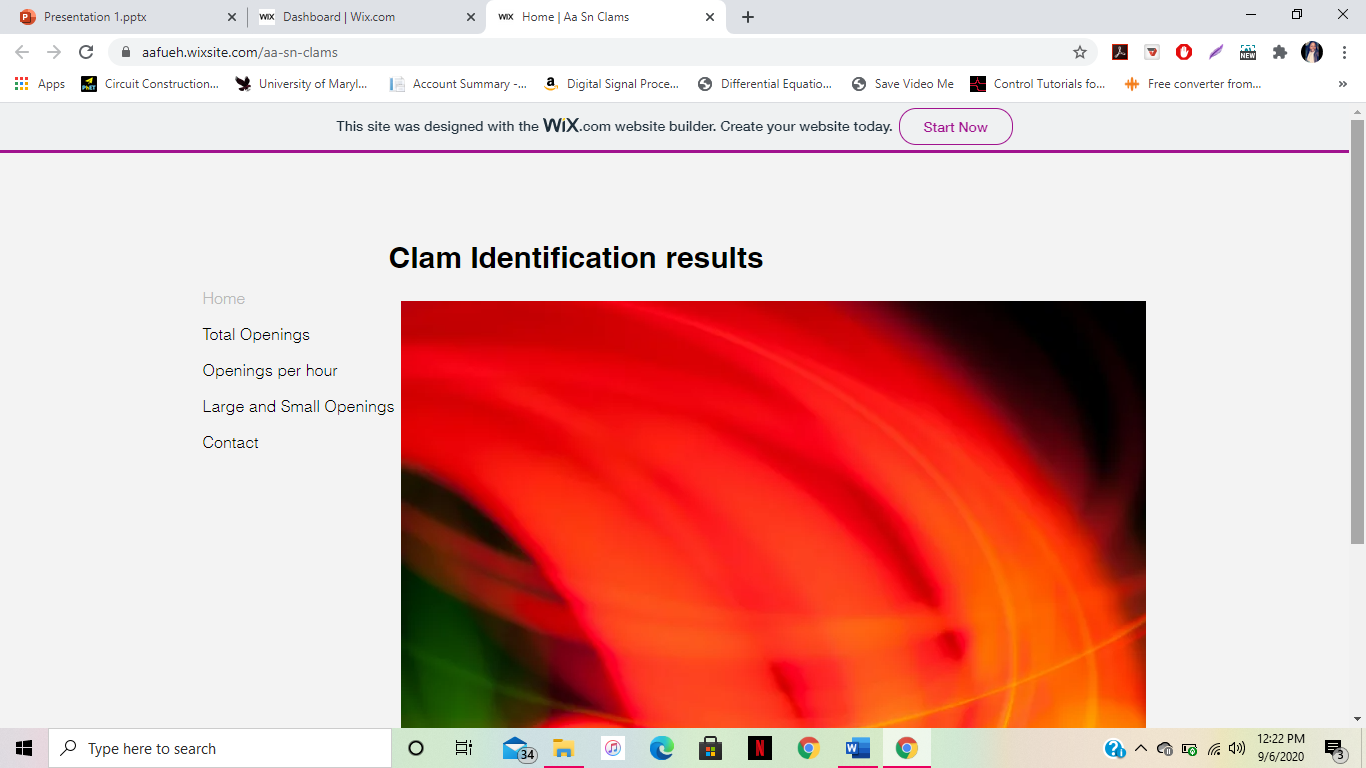
**Subtask 8.1-8.3 – Creating Website Database**

A website was also created that will display the activity of the clams. The data of the clam activity will be displayed from this website. It will have different pages for the total openings in a day,



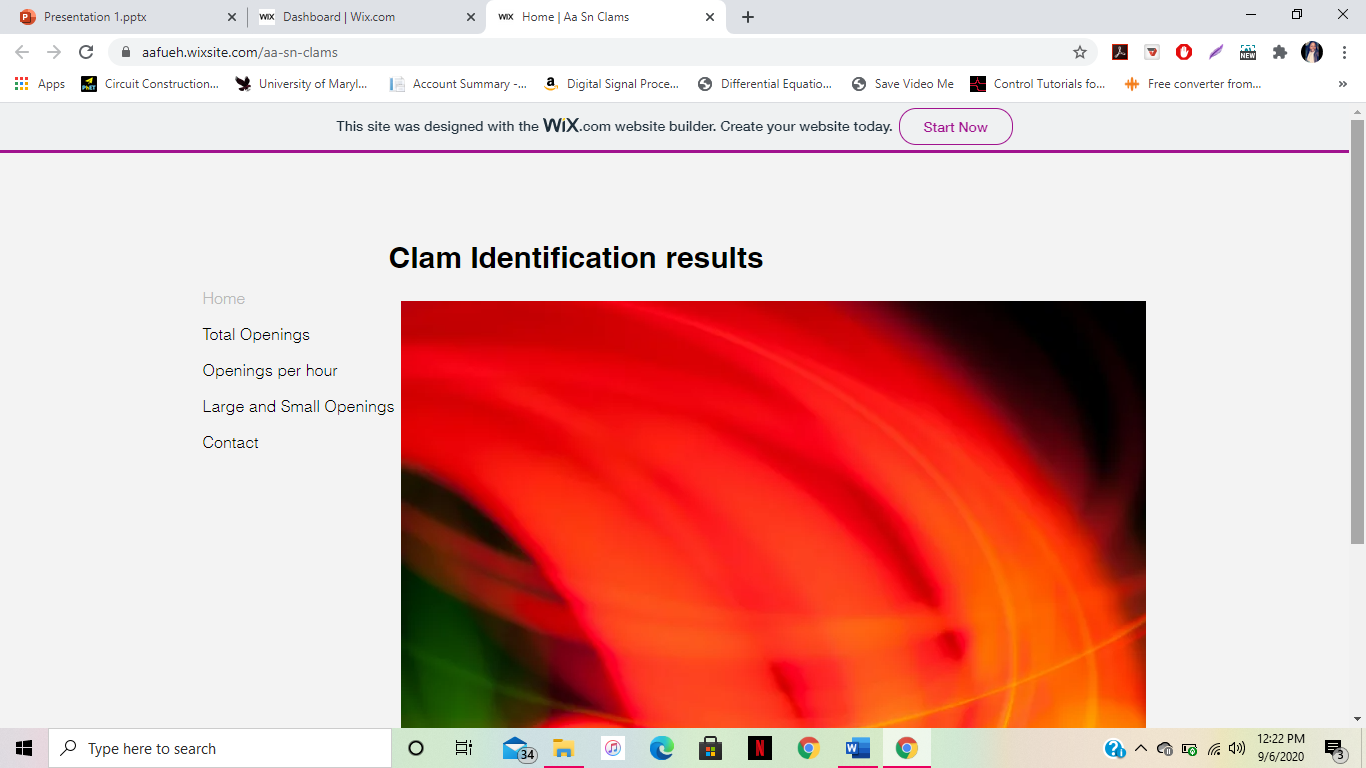
*Tab for total openings*

the number of openings per hour,



*Tab for openings per hour*

and large and small openings.



*Tab for large and small openings*

**Subtask 4.1: Photo dataset preparation**

The data that will be supplied to this website will be coming from an embedded link that contains all of the data. It will be shown in the version of a bar graph.

Our project includes using the raspberry pi 4 computer and the raspberry pi camera with the module v2. During the week, we began setting up the hardware and software components for the raspberry pi and camera. For hardware components, we physically installed the cameras into the raspberry pi computer and for the software components, we installed the raspberry pi operating system.

*Hardware components Software Components*

For the hardware component, we installed the raspberry pi cameras into the raspberry pi. In order to do this, we first unlocked the slots on the raspberry pi's for the camera. We then inserted the blue strip at the tail of the raspberry pi cameras into the slot for the cameras on the board, with the blue side of the strip facing adjacent to the black side of the slot. We then secured the slot to make sure the camera tail would not become loose.



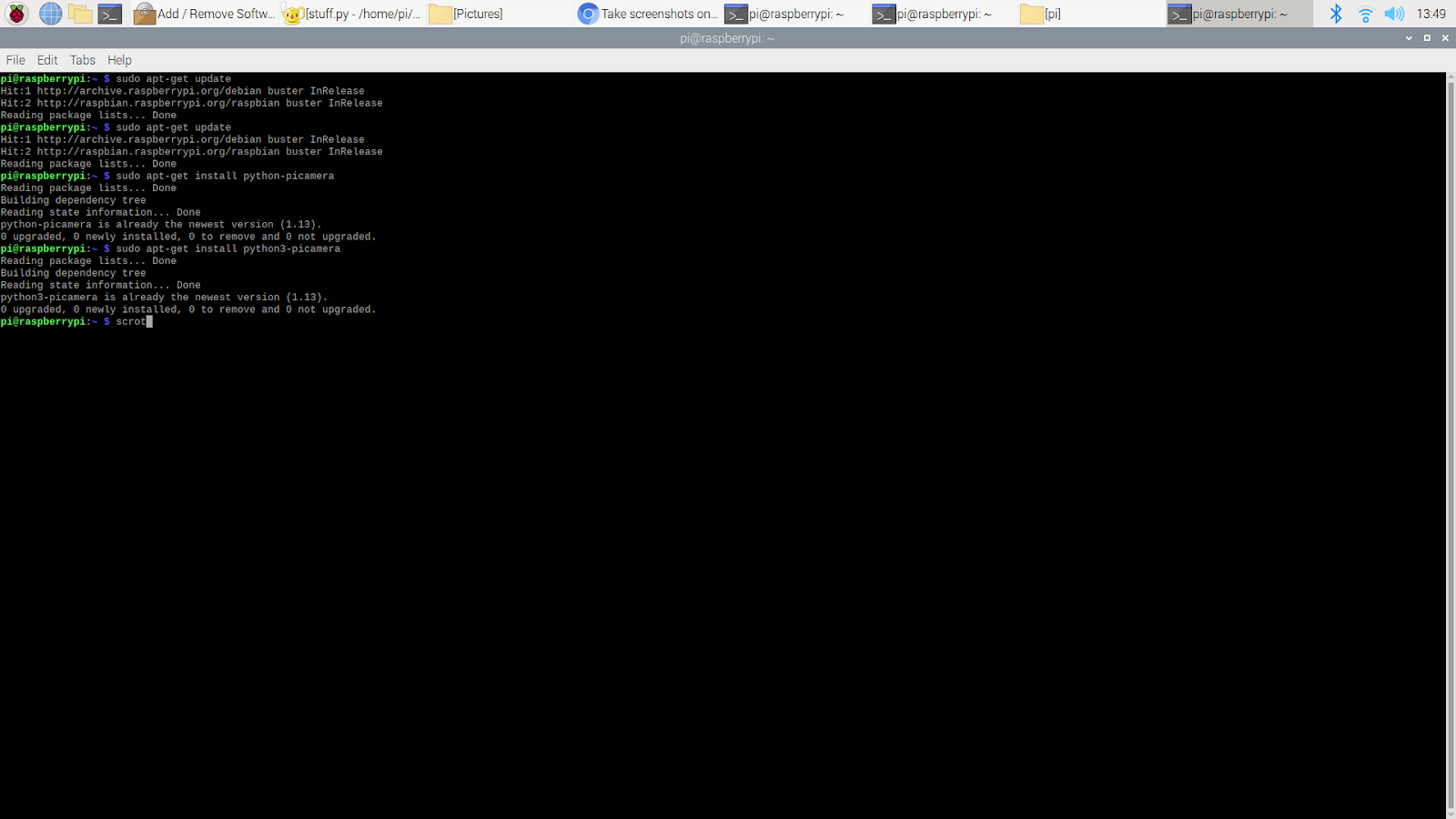
*Camera Installation*

We installed the raspberry pi cameras into the raspberry pi. In order to do this, we first unlocked the slots on the raspberry pi's for the camera. We then inserted the blue strip at the tail of the raspberry pi cameras into the slot for the cameras on the board, with the blue side of the strip facing adjacent to the black side of the slot. We then secured the slot to make sure the camera tail would not become loose.

We installed the necessary software needed for the raspberry pi, which is the Raspbian operating system. The 32 bit operating system worked best with our computer. After doing this, we tested that this operating system was installed properly by properly. As we receive the essential parts needed for the raspberry pi, we will then go further with the software installations.

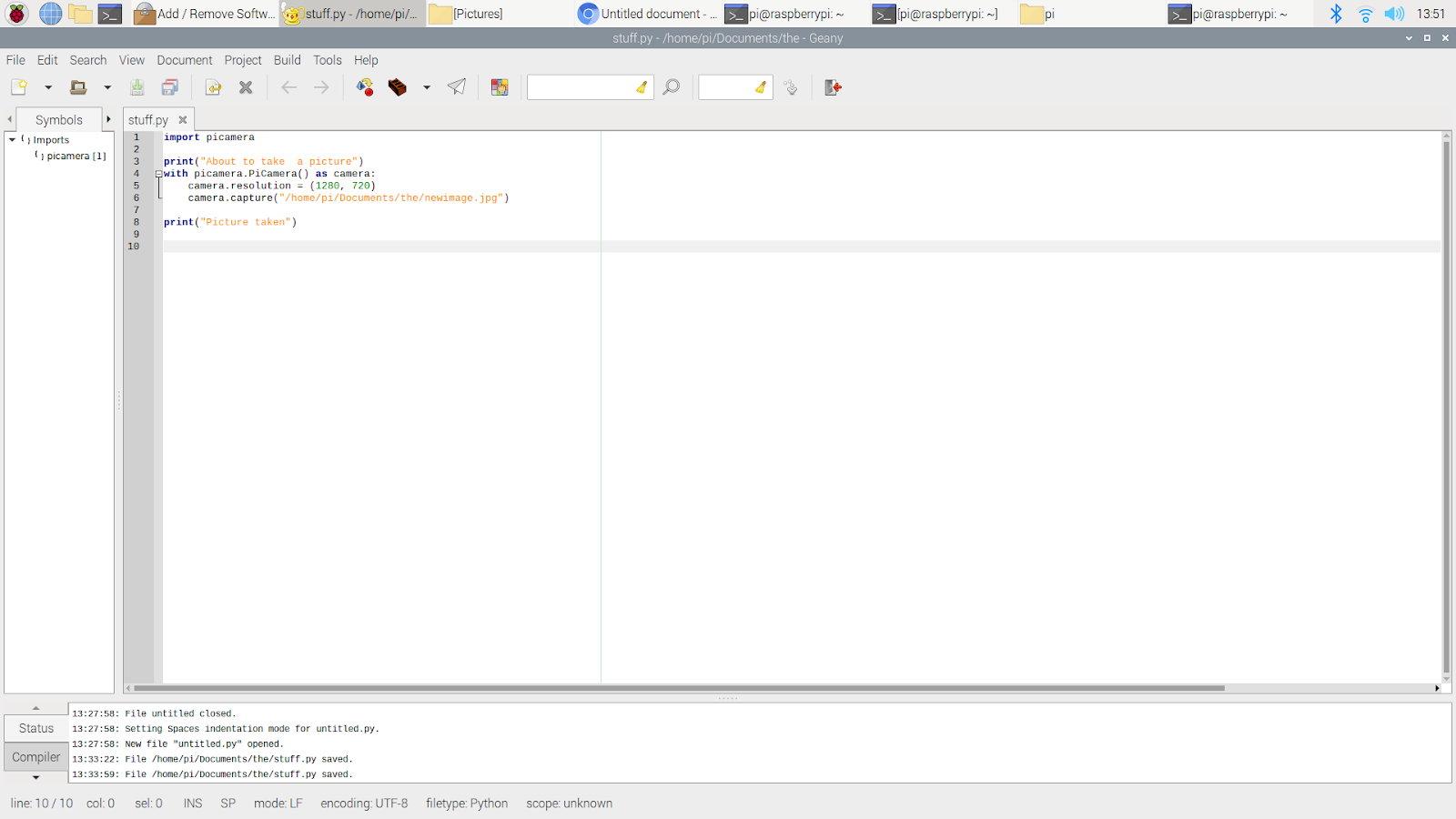
After the necessary components for the raspberry pi were obtained, pictures were tabled with the raspberry pi. Taking pictures are essential to our project because they will be needed to take images of clams. Since our class have not arrived as of yet, we practiced taking pictures of everyday items in our tank.

In the raspberry pi terminal, a module called “pycamera” was downloaded. This will give us the necessary packages to take pictures.



*Installing essential packages to take pictures*

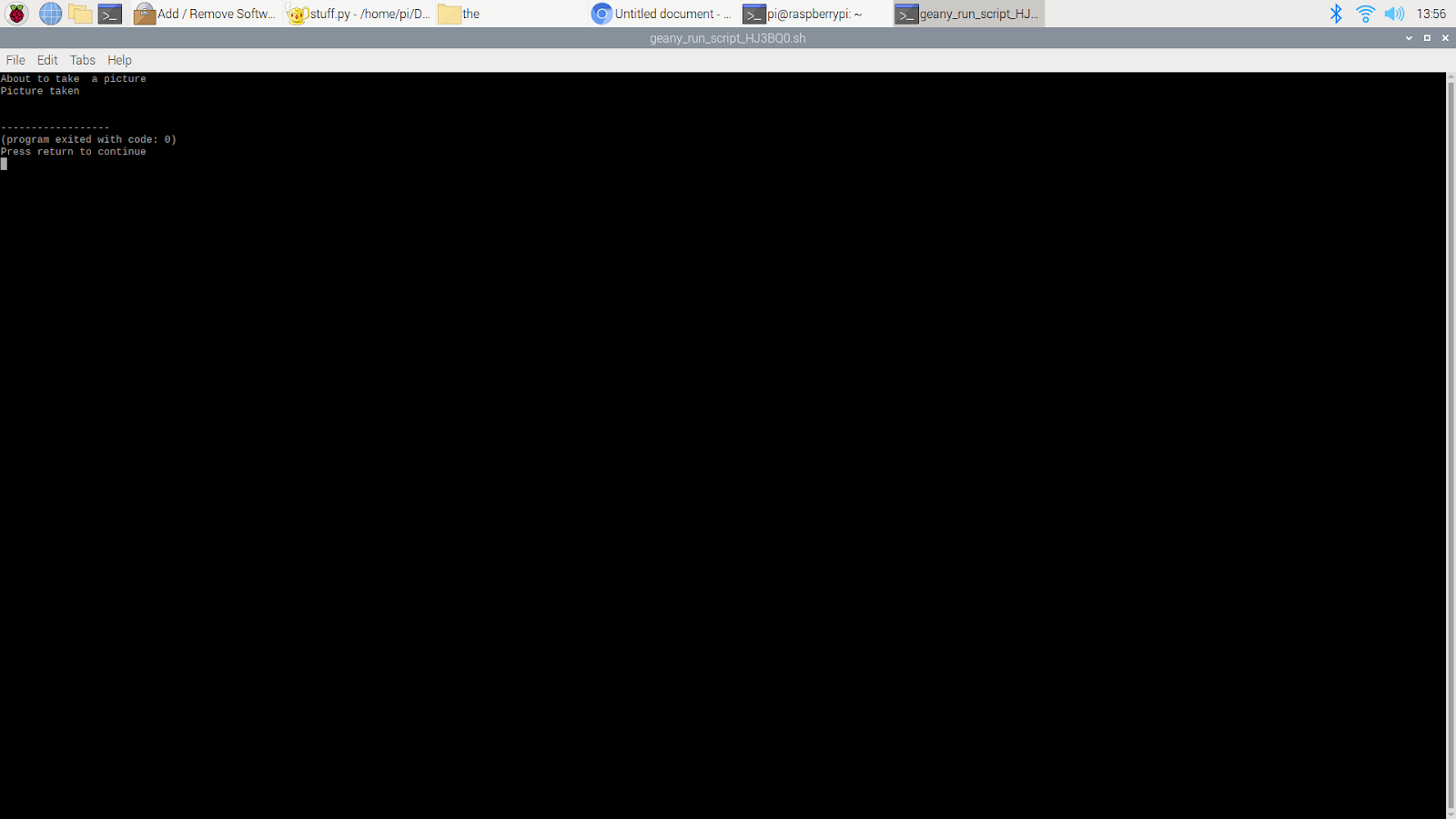
We then wrote a script in python to set up the camera:



*Python Script to take pictures with pi camera*

We made sure the resolution of the camera is high to be able to capture the images in very good quality. It will especially need to be in good quality since the pictures will be taken of the images underwater.

After executing the file, the following screen that displayed afterwards confirmed that the image was taken



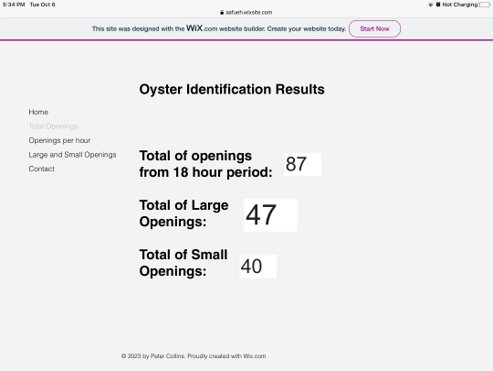
*Picture of execution of file*

We took the images of the air pump inside of the tank from various different angles.

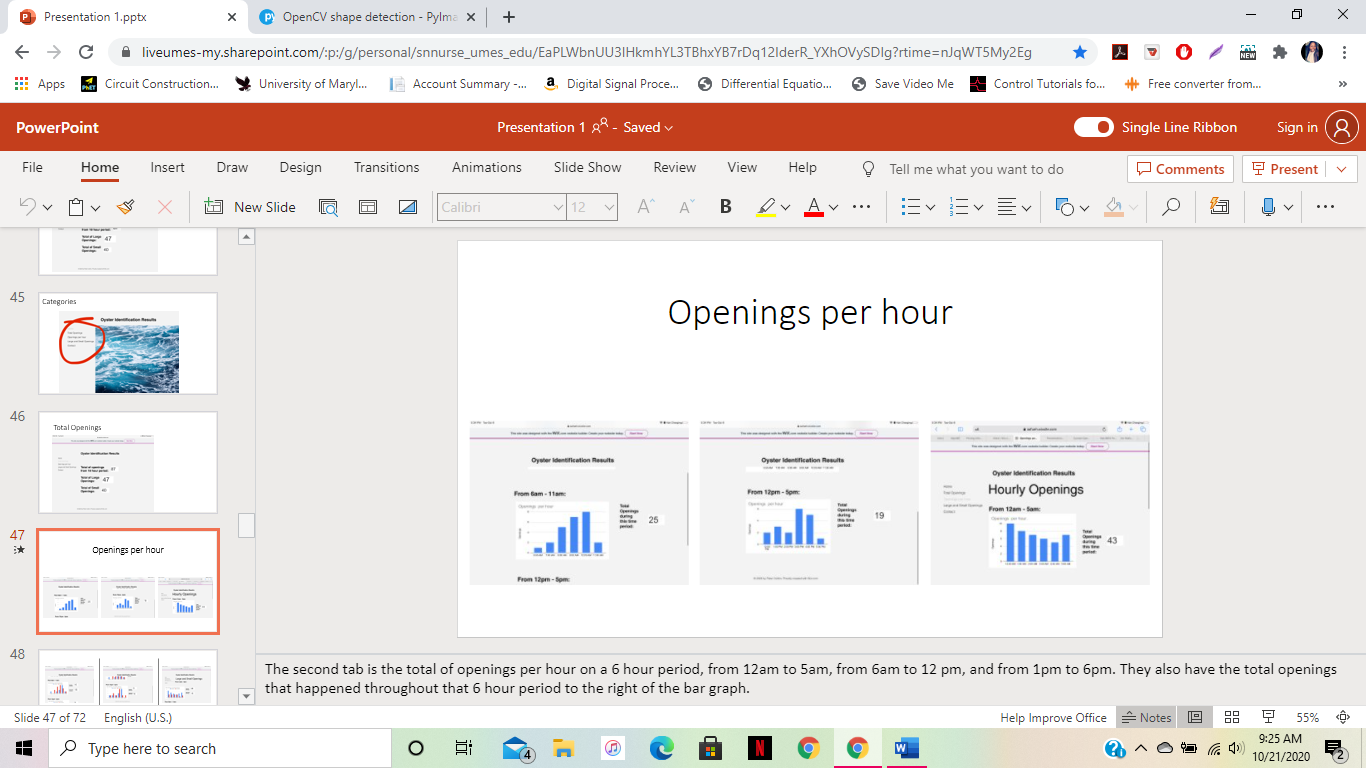


*images of air pump at different angles*

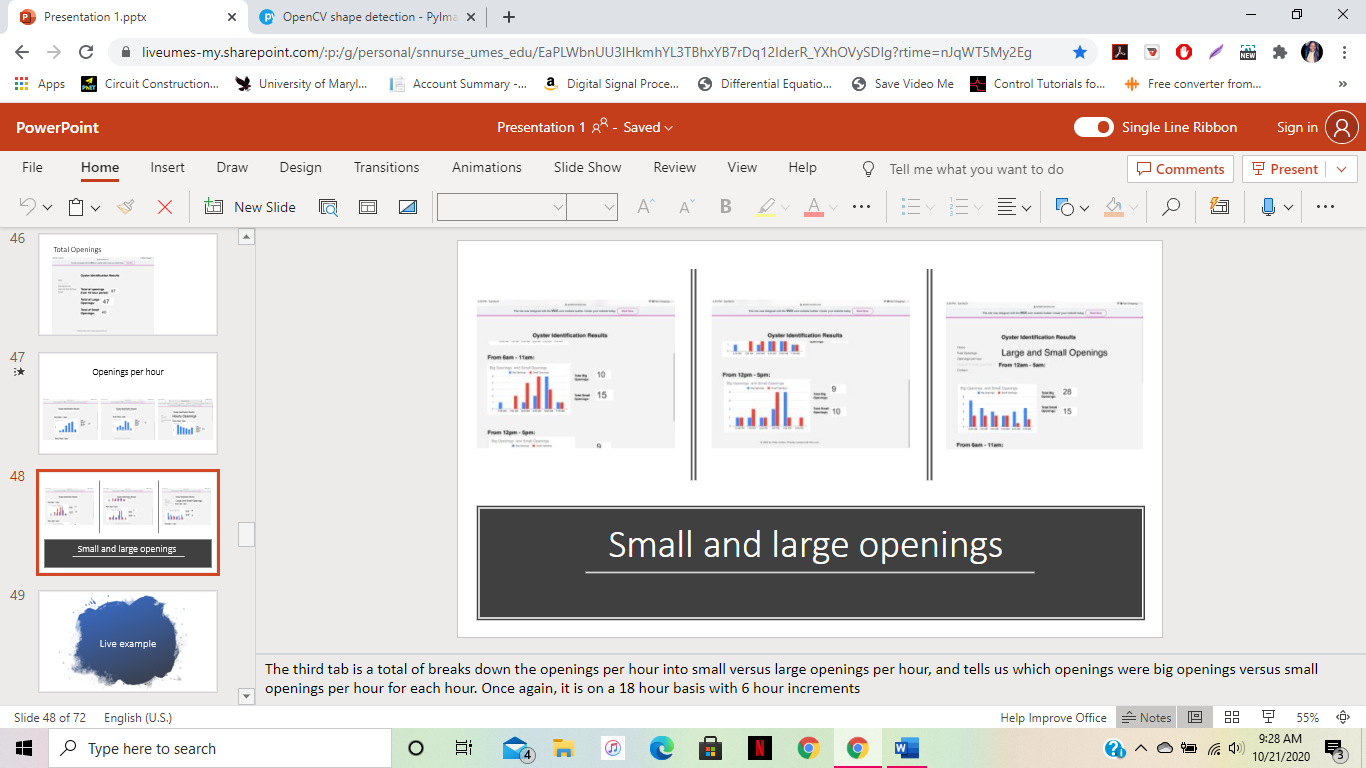
After this, we worked on the website. We were able to present data on our website via google sheets. This way, the user will have a way to visually view the data and extrapolate essential data needed from that data. This data will be embedded from a file from google sheets, which is a data source that we will use with our project due to it being compatible with the code we use for this project. There are 3 categories of data that will be shown: total openings, oyster openings per hour, and the level of openings. They all will be on a shown over an 18 hour period, with 6 hour increments, even though the data will be continuously running.



Above is the first tab of the website, which is total openings. This data explains the total openings of an 18 hour basis and breaks it down briefly into total openings, and the small and large openings out of the total openings. We used random data as an example of how the data works.

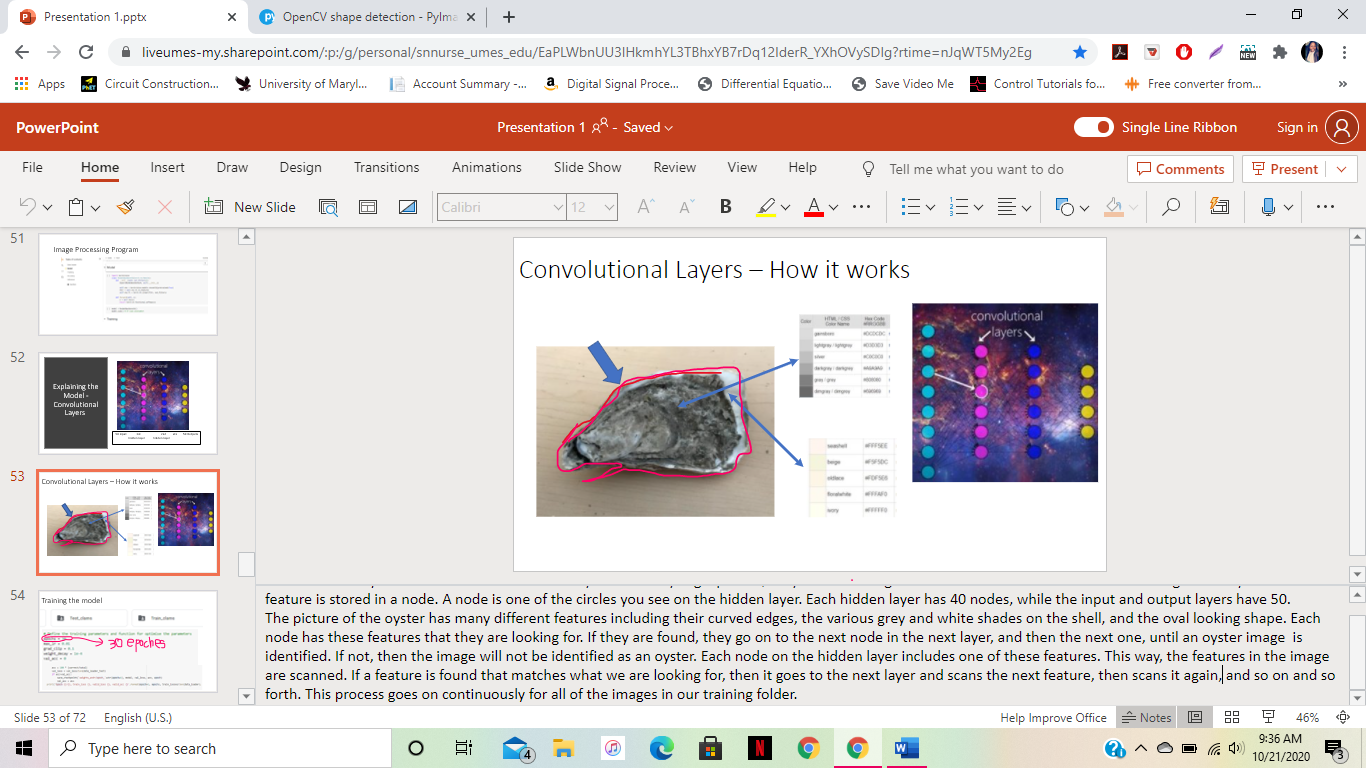


The second tab is the total of openings per hour on a 6 hour period, from 12am to 5am, from 6am to 12 pm, and from 1pm to 6pm. They also have the total openings that happened throughout that 6 hour period to the right of the bar graph. ​



The third tab is a total of breaks down the openings per hour into small versus large openings per hour, and tells us which openings were big openings versus small openings per hour for each hour. Once again, it is on a 18 hour basis with 6 hour increments​.

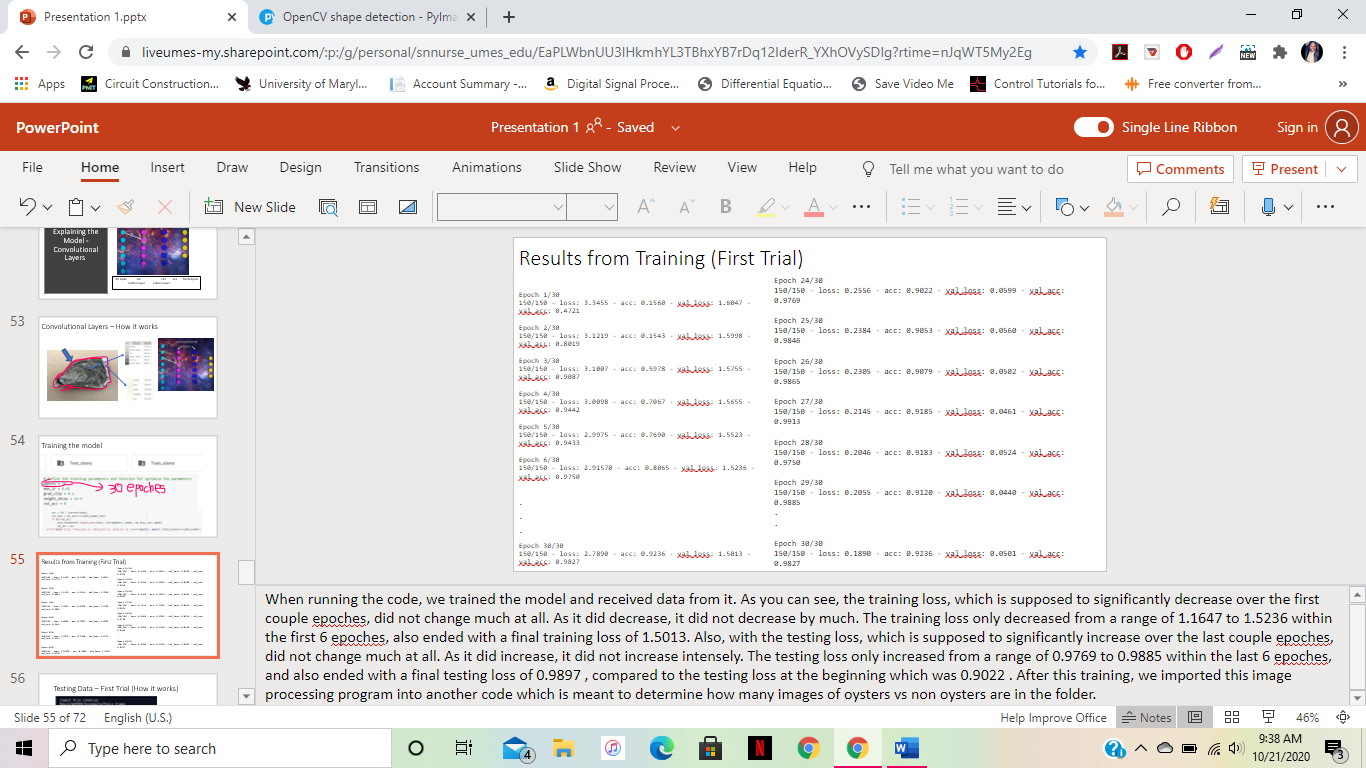
We also designed a program that determines if an image is an image of an oyster or not. We created an image processing program that allows us to identify images of oysters. We created an image processing program that allowed us to identify images of oysters. We achieved this task by using convolutional layers in a neural network. Convolutional layers are layers in a neural network that notice patterns in images that help us identify an image. They help us with image processing by us being able to determine which image is an oyster because of the similarities in pattern. ​Convolutional layers consist of input layers, hidden layers, and output layers. The input layers consist of how many categories we plan on feeding to the neural network. The hidden layers are the various amounts of different patterns In the image that the algorithm picks up on. If it continuously notices the same types of things in an image, it goes to the next hidden layer, so on and so forth, but if not, it will not identify the image. We started off with 30 nodes in the input layer and 30 nodes in the output layer, but we later chose to use 50 nodes in the input layer and output layer and 50 nodes in the output layer. There are also 10 hidden layers in between these input and output layers. The next slide will explain what these layers are doing exactly. ​



Convolutional layers work as detectives essentially. When analyzing a picture, they scan the image for various different features that the image of an oyster has. Each feature is stored in a node. A node is one of the circles you see on the hidden layer. Each hidden layer has 40 nodes, while the input and output layers have 50. ​

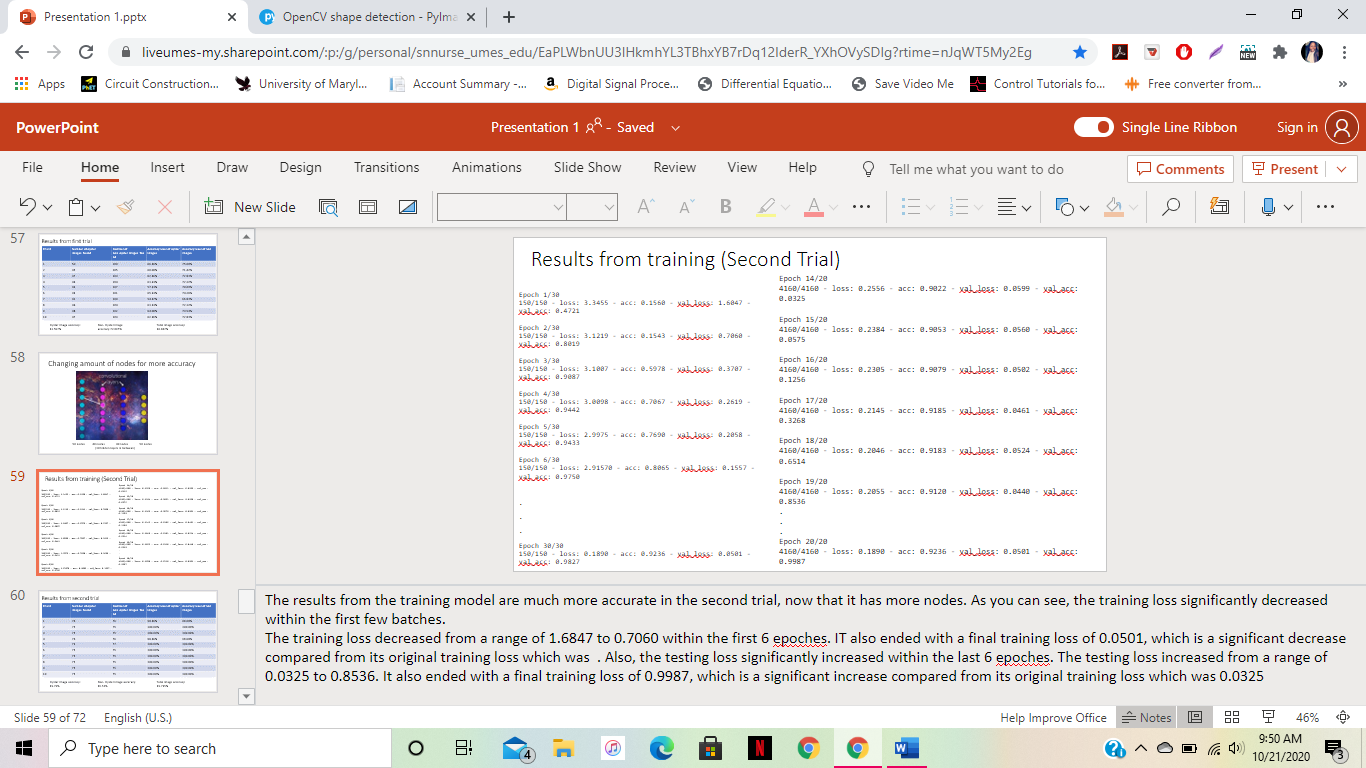
The picture of the oyster has many different features including their curved edges, the various grey and white shades on the shell, and the oval looking shape. Each node has these features that they are looking for. If they are found, they go on to the next node in the next layer, and then the next one, until an oyster image is identified. If not, then the image will not be identified as an oyster. Each node on the hidden layer includes one of these features. This way, the features in the image are scanned. If a feature is found that matches what we are looking for, then it goes to the next layer and scans the next feature, then scans it again, and so on and so forth. This process goes on continuously for all of the images in our training folder. ​

For this program, there are 2 categories – images of oysters in water, and images of sometime else, AKA non-oysters. For the pictures of oysters in water, there are 150 images of oysters in water and 150 images of other miscellaneous pictures. We included a lot of pictures of the empty tank with just water so the model would be able to determine the parts of the tank that are just water versus the parts of the tank with an oyster in it. We then defined the training parameters to be able to give us an accuracy numbers that would determine how well our training model worked. The training model went through 30 "epoches", which are 30 passes that the dataset completed. When the code is ran, It also will tell us the amount of images identified out of the testing folder, and the epoches, or passes it went through. It will also tell us the training loss and the testing loss. As described on the official "keras FAQ", THE training loss is the average of the losses over each batch of training data. Since the model is changing over time, the training loss over the first batches is generally higher, while the testing loss is generally lower over the last batches. ​



When running the code, we trained the model and received data from it. As you can see, the training loss, which is supposed to significantly decrease over the first couple epoches, did not change much at all. As it did decrease, it did not decrease by much. The training loss only decreased from a range of 1.1647 to 1.5236 within the first 6 epoches, also ended with a final training loss of 1.5013. Also, with the testing loss, which is supposed to significantly increase over the last couple epoches, did not change much at all. As it did increase, it did not increase intensely. The testing loss only increased from a range of 0.9769 to 0.9885 within the last 6 epoches, and also ended with a final testing loss of 0.9897 , compared to the testing loss at the beginning which was 0.9022 . After this training, we imported this image processing program into another code which is meant to determine how many images of oysters vs non oysters are in the folder. ​ When we run the code that obtains the image processing program in it, the code prompts us to type the file destination of the testing folder. This folder has 150 images, with 75 of them being images of oysters and 75 of them being non oyster images. The results will then tell us how many images were oyster images vs how many images were non oyster images. This will be ran 10 times and all results will be recorded for an accuracy of our training model. The accuracy level of identifying the oyster images came out to a total of 61.597% while the accuracy level of identifying the non oyster images came out to a total of 72.097%. It can be inferred from this data that our image processing program is not very strong nor accurate. ​

Because of our poor performance in the image processing results, we went back to our program and added more nodes to our layers in our neural network. Instead of 30 nodes in the input and output layer, we now have 50. We added 20 more nodes to our image processing code. These nodes identify the color, texture and shape of the oyster in the image much more specifically. We then retrained and retested the image processing code again.​

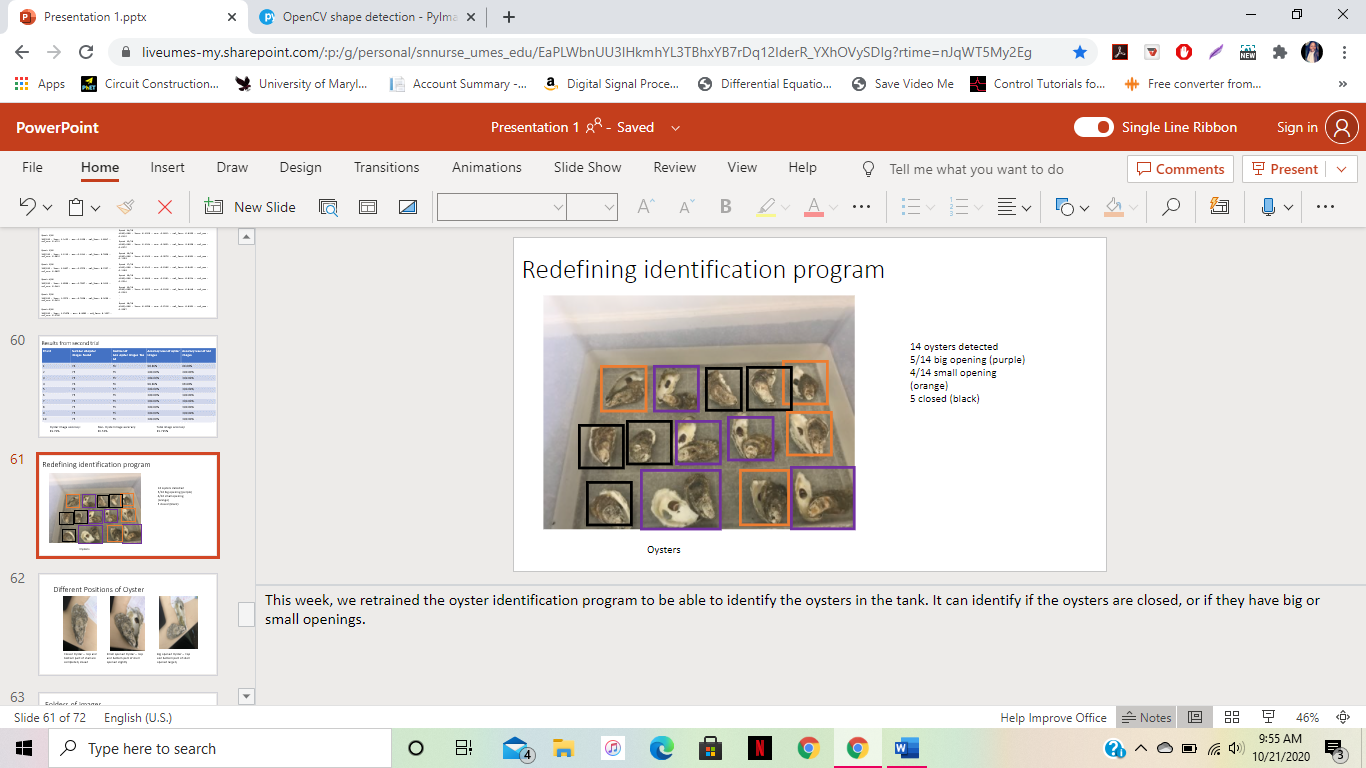


The results from the training model are much more accurate in the second trial, now that it has more nodes. As you can see, the training loss significantly decreased within the first few batches. ​

The training loss decreased from a range of 1.6847 to 0.7060 within the first 6 epoches. IT also ended with a final training loss of 0.0501, which is a significant decrease compared from its original training loss which was . Also, the testing loss significantly increased within the last 6 epoches. The testing loss increased from a range of 0.0325 to 0.8536. It also ended with a final training loss of 0.9987, which is a significant increase compared from its original training loss which was 0.0325​

We then tested our model. As you can see, the oyster image accuracy is now 99.73%, the non oyster image accuracy is now 99.74% and the total image accuracy is now 99.735. This ensures that our model has vastly improved its performance and is working greatly. ​

Afterwards, we decided to do the same method, but with all oysters in the tank at once instead of one oyster at a time, as shown



1. **Project evaluation**

In this chapter, please evaluate the performance of the solution completed in the project with the considerations of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors. And please consider improvement suggestion for the future on the aspect.

1. Public health
2. Safety
3. Welfare
4. Global factor
5. Cultural factor
6. Social factor
7. Environmental factor
8. Economic factor
9. **Conclusion**

By the end of the project, conclude the project and your learning experience.

Please make sure you include the overall evaluation of the project and the **future plan** about how the project can be improved.

* Project summary

…

* Learning and practice experience

…

* Future plan (how to improve)

…

**Acknowledgment**

If you get help or support from someone else (besides the team member and the advisor) and want to show your appreciation, put here (**do not include the advisor**).

**Appendix**

**Source Code.**

1. ***Source Code of practice code***

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

import os

#>>>>>>>>>>> Step 1

num\_skipped = 0

for folder\_name in ("cars", "planes", "random (other)"):

folder\_path = os.path.join("train", folder\_name)

for fname in os.listdir(folder\_path):

fpath = os.path.join(folder\_path, fname)

try:

fobj = open(fpath, "rb")

is\_jfif = tf.compat.as\_bytes("JFIF") in fobj.peek(10)

finally:

fobj.close()

if not is\_jfif:

num\_skipped += 1

# Delete corrupted image

os.remove(fpath)

print(fpath)

print("Deleted %d images" % num\_skipped)

#>>>>>>>>>>> Step 2

image\_size = (224, 224)

batch\_size = 32

train\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory("train",validation\_split=0.10,subset="training",seed=1337,image\_size=image\_size,batch\_size=batch\_size)

val\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory("train",validation\_split=0.10,subset="validation",seed=1337,image\_size=image\_size,batch\_size=batch\_size)

data\_augmentation = keras.Sequential(

[

layers.experimental.preprocessing.RandomFlip("horizontal"),

layers.experimental.preprocessing.RandomRotation(0.1),

]

)

train\_ds = train\_ds.prefetch(buffer\_size=32)

val\_ds = val\_ds.prefetch(buffer\_size=32)

#>>>>>>>>>>> Step 3

def make\_model(input\_shape, num\_classes):

inputs = keras.Input(shape=input\_shape)

# Image augmentation block

x = data\_augmentation(inputs)

# Entry block

x = layers.experimental.preprocessing.Rescaling(1.0 / 255)(x)

x = layers.Conv2D(32, 3, strides=2, padding="same")(x)

x = layers.BatchNormalization()(x)

x = layers.Activation("relu")(x)

x = layers.Conv2D(64, 3, padding="same")(x)

x = layers.BatchNormalization()(x)

x = layers.Activation("relu")(x)

previous\_block\_activation = x # Set aside residual

for size in [128, 256, 512, 728]:

x = layers.Activation("relu")(x)

x = layers.SeparableConv2D(size, 3, padding="same")(x)

x = layers.BatchNormalization()(x)

x = layers.Activation("relu")(x)

x = layers.SeparableConv2D(size, 3, padding="same")(x)

x = layers.BatchNormalization()(x)

x = layers.MaxPooling2D(3, strides=2, padding="same")(x)

# Project residual

residual = layers.Conv2D(size, 1, strides=2, padding="same")(

previous\_block\_activation

)

x = layers.add([x, residual]) # Add back residual

previous\_block\_activation = x # Set aside next residual

x = layers.SeparableConv2D(1024, 3, padding="same")(x)

x = layers.BatchNormalization()(x)

x = layers.Activation("relu")(x)

x = layers.GlobalAveragePooling2D()(x)

if num\_classes == 2:

activation = "sigmoid"

units = 1

else:

activation = "softmax"

units = num\_classes

x = layers.Dropout(0.5)(x)

outputs = layers.Dense(units, activation=activation)(x)

return keras.Model(inputs, outputs)

from os import path

#>>>>>>>>>>> Step 4

epochs = 10

savedFileName = "save\_at\_{0}.h5".format(epochs)

if(not(path.exists(savedFileName))):

model = make\_model(input\_shape=image\_size + (3,), num\_classes=3)

keras.utils.plot\_model(model, show\_shapes=True)

callbacks = [

keras.callbacks.ModelCheckpoint("save\_at\_{epoch}.h5"),

]

model.compile(

optimizer=keras.optimizers.Adam(1e-3),

loss="binary\_crossentropy",

metrics=["categorical\_accuracy"],

)

model.fit(

train\_ds, epochs=epochs, callbacks=callbacks, validation\_data=val\_ds,

)

model = keras.models.load\_model(savedFileName)

#>>>>>>>>>>> Step 5

carCount = 0

planeCount = 0

othersCount = 0

for folder\_name in ("cars", "planes", "random (other)"):

folder\_path = os.path.join("train", folder\_name)

for fname in os.listdir(folder\_path):

fpath = os.path.join(folder\_path, fname)

img = keras.preprocessing.image.load\_img(

fpath, target\_size=image\_size

)

try:

img\_array = keras.preprocessing.image.img\_to\_array(img)

img\_array = tf.expand\_dims(img\_array, 0) # Create batch axis

predictions = model.predict(img\_array)

predictions = predictions[0]

if(predictions[0] > predictions[1] and predictions[0] > predictions[2]):

carCount = carCount + 1

elif(predictions[1] > predictions[0] and predictions[1] > predictions[2]):

planeCount = planeCount + 1

elif(predictions[2] > predictions[0] and predictions[2] > predictions[1]):

print(fpath)

othersCount = othersCount + 1

finally:

fobj.close()

print("car Count : ", carCount)

print("plane Count : ", planeCount)

print("others Count : ", othersCount)

**REFERENCES**

[1] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino*, et al.*, "Corona: System Implications of Emerging Nanophotonic Technology," in *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, 2008, pp. 153-164.

[2] X. Zhang and A. Louri, "A multilayer nanophotonic interconnection network for on-chip many-core communications," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, 2010, pp. 156-161.

[3] C. Batten, A. Joshi, J. Orcutt, A. Khilo, B. Moss, C. Holzwarth*, et al.*, "Building manycore processor-to-DRAM networks with monolithic silicon photonics," in *High Performance Interconnects, 2008. HOTI '08. 16th IEEE Symposium on*, 2008, pp. 21-30.

[4] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly: illuminating future network-on-chip with nanophotonics," in *IEEE/ACM Intl. Symp. on Computer Architecture (ISCA)*, 2009, pp. 429-440.

[5] N. Kirman, M. Kirman, R. K. Dokania, J. F. Martinez, A. B. Apsel, M. A. Watkins*, et al.*, "Leveraging Optical Technology in Future Bus-based Chip Multiprocessors," in *Microarchitecture, 2006. MICRO-39. 39th Annual IEEE/ACM International Symposium on*, 2006, pp. 492-503.

[6] J. M. Cianchetti, C. J. Kerekes, and H. D. Albonesi, "Phastlane: a rapid transit optical routing network," in *Proceeding of: 36th International Symposium on Computer Architecture (ISCA)*, 2009, pp. 441-450.

[7] A. Shacham, K. Bergman, and L. P. Carloni, "Photonic Networks-on-Chip for Future Generations of Chip Multiprocessors," *Computers, IEEE Transactions on,* vol. 57, pp. 1246-1260, 2008.

[8] A. Shacham, K. Bergman, and L. P. Carloni, "On the Design of a Photonic Network-on-Chip," in *First International Symposium on Networks-on-Chip, 2007. NOCS 2007*, 2007, pp. 53-64.

[9] M. Kwai Hung, Y. Yaoyao, W. Xiaowen, Z. Wei, L. Weichen, and X. Jiang, "A Hierarchical Hybrid Optical-Electronic Network-on-Chip," in *VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium on*, 2010, pp. 327-332.

[10] D. Ding and D. Z. Pan, "OIL: a nano-photonics optical interconnect library for a new photonic networks-on-chip architecture," presented at the Proceedings of the 11th international workshop on System level interconnect prediction, San Francisco, CA, USA, 2009.

[11] A. Joshi, C. Batten, K. Yong-Jin, S. Beamer, I. Shamim, K. Asanovic*, et al.*, "Silicon-photonic clos networks for global on-chip communication," in *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on*, 2009, pp. 124-133.

[12] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino*, et al.*, "Corona: system implications of emerging nanophotonic technology," in *Proc. 35th IEEE/ACM Int'l Symp. Computer Architecture (ISCA)*, 2008, pp. 153-164.

[13] L. Zhang, E. Regentova, and X. Tan, "A 2D-Torus Based Packet Switching Optical Network-on-Chip Architecture," presented at the *IEEE International Symposium on Photonics and Optoelectronics* (SOPO 2011), Wuhan, China, 2011.

[14] L. Zhang, E. E. Regentova, and X. Tan, "Packet switching optical network-on-chip architectures," *Comput. Electr. Eng.,* vol. 39, pp. 697-714, 2013.

[15] G. Huaxi, X. Jiang, and W. Zheng, "A novel optical mesh network-on-chip for gigascale systems-on-chip," in *Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on*, 2008, pp. 1728-1731.

[16] G. Huaxi, X. Jiang, and Z. Wei, "A low-power fat tree-based optical Network-On-Chip for multiprocessor system-on-chip," in *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE '09.*, 2009, pp. 3-8.

[17] Y. Yaoyao, X. Jiang, H. Baihan, W. Xiaowen, Z. Wei, W. Xuan*, et al.*, "3-D Mesh-Based Optical Network-on-Chip for Multiprocessor System-on-Chip," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* vol. 32, pp. 584-596, 2013.

[18] A. Shacham, K. Bergman, and L. P. Carloni, "Photonic networks-on-chip for future generations of chip multiprocessors," *IEEE Trans. Computers,* vol. 57, pp. 1246-1260, 2008.

[19] A. W. Poon, F. X. Xu, and X. Luo, "Cascaded active silicon microresonator array cross-connect circuits for WDM networks-on-chip," in *Proc. SPIE*, 2008, p. 689812.

[20] M. Lipson, "Compact Electro-Optic Modulators on a Silicon Chip," *IEEE Journal of Selected Topics in Quantum Electronics,* vol. 12, pp. 1520-1526, 2006.

[21] M. Lipson, "Guiding, modulating, and emitting light on Silicon-challenges and opportunities," *Lightwave Technology, Journal of,* vol. 23, pp. 4222-4238, 2005.