Aquatic Identification and Sorting System

A

project

submitted by

John Schulz and Reiner Lintag

For

Senior Capstone Project

Department of Electrical Engineering

Peoria, IL

5/7/2019

Abstract

In many parts of the world, non-native species populations are growing at a rate faster than humans can actively remove them. The goal of the aquatic identification and sorting system (A.I.S.S.) is to develop a self-contained system that utilizes digital image processing and artificial intelligence inferencing to collect data and successfully sort a desired targeted species. The embedded acquisition and identification system developed is implemented with low latency inferencing hardware for millisecond response times. A battery backup power supply provides support for the onboard electronics and servo motors. Overall, the design strategy of the project utilizes a modular approach to be able to meet the needs of various environments, while maintaining maximum safety for both humans and marine wildlife.

Table of Contents

Table of Figures

Table of Tables

Table of Equations

I. Introduction

There exists little to no practical or efficient solutions for controlling marine species populations in the world's oceans and local waterways. The first step in managing a population is identifying the deployment environment where the system will target and classify species. The identification side of this project implements machine learning algorithms and image processing to classify data such as the type of species; whereas, the mobility side of the project involves creating a power system to support the various digital systems in a remote environment for an extended period. Lastly, the management portion of the project constructs a proactive approach to harnessing the data collected from the identification portion of the project to be able to sort the system's target object. The project involves three main areas of development: species identification, system mobility, and power management.

A. Objective

The current approach to controlling or removing aquatic species requires intense human interaction through manual extraction methods. These techniques include: hunting with spearing, electrofishing, or netting to identify and remove exotic marine animals. All these approaches are ineffective financially and are an inefficient use of one's time when spent attempting to control a population's size. In addition, financial bounties for removing the invasive species of reef fish, known as lionfish, have proved ineffective at tackling this problem in the state of Florida [12]. Due to the shortfalls of the previously stated removal methods, this project could improve upon the capture rate of invasive species by utilizing a remotely deployable system that implements artificial intelligence and machine learning hardware to identify, sort, and size a species that it has been trained to determine as exotic. Thus, the artificial intelligence fish sorting project has five primary objectives to analyze and solve:

- Develop a remote, portable, modular, sorting device, and power system.
- Maintain safety for animals and humans
- Implement artificial intelligence inferencing in an edge environment.
- Utilize low power and cost-efficient embedded components.
- Data log all interactions with marine wildlife.

B. Constraints

Based on the four primary goals, compromises in the design were a consideration throughout the course of the project. First, the developed system needs to be able to withstand a variety of remote marine environments from salt-water to fresh-water and various levels of water clarities, while not attached to a mainland power source. A portable product of this variety means that different species of marine life may be a target of this system. Therefore, modularity is essential in expanding the machine to sort a 40-pound animal versus one weighing 2 pounds. Likewise, expansion to accommodate different animal lengths may be crucial to some sorting applications.

Holding safety to the highest regard for all animal and human bi-standards is a crucial point in the engineering choices made throughout this project. Mistakenly harming a living organism is not tolerable and defeats the appeal of such a device. Since the system needs to be remotely deployable, the artificial intelligence system cannot rely on cloud computing architectures to make the inferences on the species the system comes across. Therefore, an embedded system must be able to compute machine inferencing with low latency and power requirements. Avoiding exotic componentry is essential because this system may get damaged or destroyed in some capacity and increases the overall price of the machine. Lastly, for research and debugging purposes, a data log is a necessary system feature.

II. Related Work

The related work section covers a literature search of projects at other educational institutions that are seeking to solve problems similar to the Aquatic Identification and Sorting System, while the commercial application subsection covers the real-world use cases for the research conducted in this project.

A. Literature Search

Three other universities are currently pursuing similar machine learning based engineering projects: the University of Alberta, Oregon State University, Carnegie Mellon University with Microsoft. The University of Alberta's research consists of managing invasive aquatic species using only machine learning algorithms focusing on decision trees [9]. The research team in Alberta is targeting only intercoastal species of plants and shellfish. Oregon State is also exclusively applying a decision tree approach, but they are focusing on only Lion Fish [10]. Carnegie Mellon and Microsoft are not targeting animals; instead, they are utilizing machine learning and game theory to track patrol routes of poachers to help wildlife authorities better protect endangered species of animals on wildlife reservations in Africa [11].

B. Commercial Applications

Two primary commercial uses exist for such a product: management of aquatic populations and aquaculture, which was rated to be a $128 billion-dollar industry in 2012 [1]. The targeted use case for the marine management sector would allow for the sorting and removal of invasive species from the environment by private or public entities. Currently, the removal of invasive species is a pressing issue for governments such as in the case of keeping Asian Carp out of the Great Lakes. The problem is estimated to cost from $15 to $18 billion and take 25 years to complete using traditional human-made barriers, as assessed by the Army Corps of Engineers [2]. The second commercial use case of the AISS is as a tool for the on and offshore aquaculture industries to systematically farm fish based on an individual size rather than on the collective batch size, thus helping to reduce inefficient harvesting that amounts to a $50 billion a year loss globally, as stated in a 2013 report by The World Bank.

III. Methodology

The methodology section discusses the basis of how the team viewed and approached the aquatic identification and sorting system project before the implementation phase began.

A. Proposed System

The fish sorting system is designed in a modular fashion to accommodate dimensional scaling depending on the target environment, the needs of the user, and the size of the targeted sorted specimens. The design principle of this project is to contain only internal moving parts to maintain a level of safety for the sortable object and human interaction. Ease of deployment is also critical to the success of the device, and the device must be able to survive and function in a remote area where access to the power grid is not always viable. The system is built around the prospect of low noise and vibrations as not to disturb the environment.

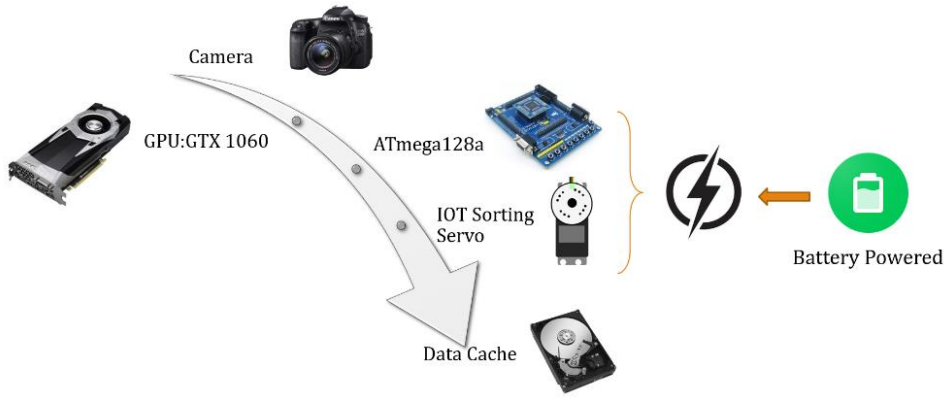

Figure 1 - Hardware Pipeline Overview

Figure 1 conveys all the hardware components required to implement the proposed system. The pipeline begins with an imaging device, which sends its imaging data to a neural network on either a GPU or FPGA for artificial intelligence inferencing to occur. Once a prediction on the species classification is determined, then the system will send a command to the ATmega128a microcontroller to close servos that will trap the fish. Lastly, valuable data is saved to a non-volatile storage element consisting of neural network accuracy and an image of the classified subject. Data retrieval is a crucial part of the pipeline because it aids in future research efforts and provides a log for future improvements.

B. Machine Learning

The approach to machine learning in this project is to design a model that can correctly inference the different classes of fishing lures. Since this model will not rely on cloud computing to perform its inferencing, this type of inferencing is called edge based neural intelligence. For image recognition tasks a convolutional neural network is best suited to identify visual patterns.

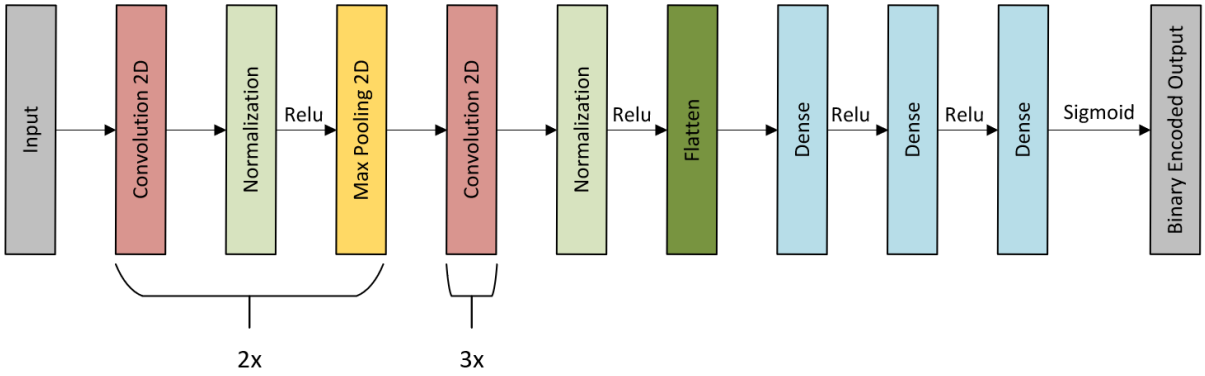

Figure 2 - AlexNet Model Hierarchy

Figure 2 shows the AlexNet layered structure that will be implemented in this project because AlexNet is widely known as a successfully applied convolutional neural network model [4][6]. The only problem with this model is that in its current state shown above it is too computationally expensive for an edge application; therefore, some layers will undergo a trimming in size, reduction in the total number of nodes and fixed point accuracy, or the removal of some layers to optimize latency and compute performance.

C. Energy System

The energy system designed for this project places a priority on mobility and management of the attached load. To power the system the user will need a power source that can be a combination of highly functional and easily transportable. One of the design constraints is to have at least a 24-hour or more battery life during deployment. Power to the system will go to the development board and its corresponding sensors and four servos that will be outside of the sealed container filled with water and a fishing lure.

D. System Criteria

The success of the sorting system depends on answering six key points:

1. What is it?
2. What size is it?
3. Will a successful sort be possible?
4. Execute the mechanical sorting hardware.
5. Will the device be scalable and moveable?

The success of the power system depends on answering three key points:

1. Can it work in off-grid scenarios?
2. Is the power source sufficiently supplying the onboard electronics?
3. Will the power supply last at least 24-hours?

E. Electrical Safety Plan

Established under “Appendix V - Safety Management System Guidelines” is a documented created by the team with the approach of providing set procedures that outline the safety measures for the implementation phase of the Aquatic Identification and Sorting System project. The intent of this manual is for all students and staff that participate in any aspect of this project that includes possible hazards, incidents, or injury have a solid understanding of the safety procedures that they must adhere too and posses. The rules of operation stated must be applied to every team that wishes to continue the A.I.S.S. project. The document includes a safety summary, scope, contact information, purpose, team responsibilities, and safety procedure. The “Appendix VI – Emergency Contingency Plan” contains the order of operations that one should adhere to if an individual is injured.

IV. Architecture

Section IV on Architecture provides an analysis of how components are used to create the aquatic identification and sorting system. This section should answer these questions:

- How to design a simulation container?
- What specifications of a simulation container may affect other parts of the project?
- What types of machine learning platforms are available for this project?
- What to look for in a machine learning API?

A. Simulation Container Guidelines

The purpose of the simulation container is to create a controllable environment for collecting data, testing neural network inferencing, and implementing sorting control systems. This tank is not meant to be the deployed product. Figure 3, provides a high-level design overview of the essential components used to create the simulation tank which includes: a scaled reference point, a clear container, external environmental cladding, a pump for water flow, piping, and a mount point for the lure.

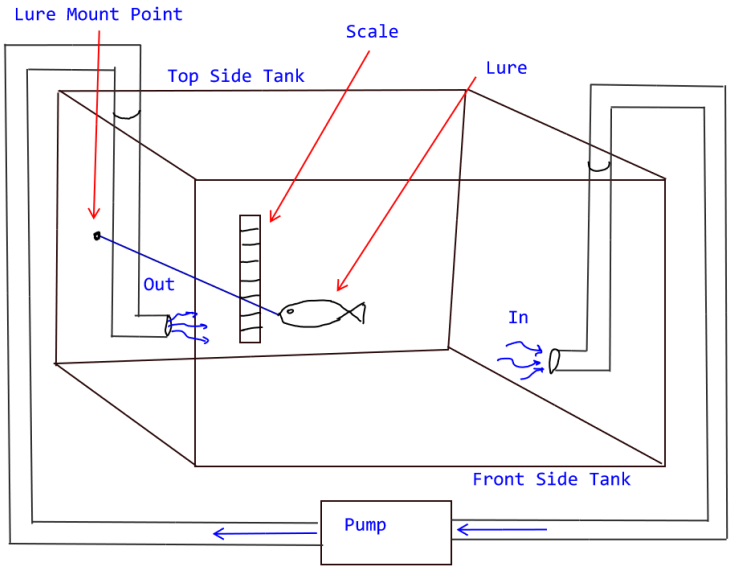

Figure 3 - Simulation Design

The scale on the back wall of the container in figure 3 is designed to provide a measurable reference point for parameters of fish height, length, and water clarity. The design requires there to be a specific placement of the scale out of view of the imaging feed going to the neural network and not present in any of the training data because it will inhibit the inferencing accuracy.

Containers size, shape, and features largely depended on the: lure dimensions, the distance required to focus on the subject, the method used to image the subject, ability to seal the container, drop protection, and volume of the tank. If the dimensions of the lure are too large, then the data collected may become compromised by forcing easily identifiable structures within the container like the pump or piping into the same image as the subject. A camera with a minimal focusing distance can be problematic because it will require the camera to be too far away from the subject, thus including external structure.

Shape and clarity are essential to consider because if the tank does not have a flat surface, the images taken will be distorted and unusable. A sealable tank with a lid provides extra protection from water spillage and is a safety requirement for this project. Also, the material of the tank is a safety concern if dropped; therefore, one should avoid glass containers. Lastly, the volume of water that the container can hold is crucial once all the previously stated requirements are satisfied. The depth of the water is a significant factor in reducing any surface turbulence and in providing enough room to image the subject without having the bottom or lid of the container in the image. The width of the tank must be a little wider than the pump. The length of the container must be long enough to fit all the required components inside, but if a tank is too short a circular current between the output pipe and the input might cause a tornado effect that may disrupt the lures swimming characteristics.

For optimal imaging accuracy, the background cladding of the tank should be a solid nonreflective color, free of blemishes and textures, provide minimal light bleed characteristics, and made of a light absorbent material. One of the goals of the background cladding is to make the imaging and data collection process reproducible from shot to shot and independent of the surrounding environmental lighting conditions. The physical qualities that the material must adhere too are either a molded shell that fits around the container or a flexible shell that can be installed and secured around the tank. Secondly, the materials must be able to withstand a corrosive environment. Ultimately, the choice in background materials must take into account light absorption and reflection effects.

Pump choice is another critical component of the simulation container during the data collecting process. The diameter of the tubing going to the pump in figure 3 is mostly dependent upon the size of the pump. There are two critical factors pertaining to pump choice: the flow rate measured in gallons or liters per second and the amount of aggregate debris that the system needs to handle without becoming bogged down. A lure requires a high volume of water pumped around it for its hydrodynamic properties of a side to side swimming action and position within the high-pressure stream coming out of the output pipe to function correctly. It is paramount to consider the number of kinks in the pipeline and length of tubing because it will affect the pumps energy consumption and rated volume per minute. Thus, it would be recommended to choose a relatively high-volume pump over a high-pressure model.

Different types of pumps and configurations are available on the market and suitable for building a simulation tank. The first requirement is a pump that is made to be in and around water; this functionality is a safety requirement since the simulation tank requires power to operate. This condition narrows down the types of pumps available to sump pumps, which are pumps that are designed to pump water out of basements and are usually submersible. Figure 3, shows the pump placed outside of the container, but it can also be inside. Since water spillage is a safety concern, it is safer to choose a submersible sump pump that can remain inside the tank to avoid the risk of a pipe failure at the neck of the pump.

Alternative piping materials are available for the simulation tank ranging in different rigidities, maximum lengths, and opacities. Whether the pump is inside or outside of the container, the piping must be rigid enough to stand on its own and maintain its factory circumference to reduce strain on the pump. Since the amount piping required to put together the tank is relatively minimal, the length should not limit one's choice in piping material. The opacity of the pipe comes down to convenience and functionality because it may be necessary to see through the exterior to spot a clogged section. If cost, rigidity, and compactness is a concern, then materials such as PVC pipe offer an excellent solution.

B. Machine Learning Hardware Platforms

Machine Learning (ML) hardware tasks can train and inference in a variety of hardware architectures: central processing units (CPUs), a graphics processing unit (GPUs), accelerated processing units (APUs), tensor processing units (TPUs), and field programmable gate arrays (FPGAs). For most applications, CPUs are avoided by the machine learning community because they do not have a high throughput or offer an extensive parallel computational architecture. In the field of neural networks, hobbyist and

educational instructors advise using GPUs, due to their ambiguous support by software developers. The downside to GPUs is that they are inefficient in their use of power.

However, an increasing number of APUs, which are graphics processing units combined with CPUs on the same die, are gaining popularity because they are inexpensive, readily available, and fast enough for many ML tasks. The usage of APUs for machine learning tasks in desktop computers down to industrial embedded products will likely increase due to the ubiquity these chips may pose in the marketplace. An APU's System In a Package (SIP) architecture is becoming cheaper to produce, which presents a cost advantage for manufacturers to implement these chiplets into the Internet of Things (IoT) product space for artificial intelligence applications.

Tensor Processing Units (TPUs), popularized by Google, is a term used to describe custom Application-Specific Integrated Circuits (ASICs) designed to accelerate specific machine learning tasks from the manufacturer's application programming interface (API) like Google's TensorFlow for Android or Apple's CoreML for iPhone [13]. While the creation of some TPUs are currently focusing on training machine learning models, these types of chips typically attempt to maximize the number of DSP blocks per cm^2 of silicon for inferencing applications. TPUs have an advantage over competitor architectures in that they can maintain a higher clock frequency, consume little power, and compute ML tasks in a parallel manner. It is speculative at the time of this report, but TPUs are not likely to gain much ground until an open source architecture can be determined because any hardware advancements are kept secret as highly guarded intellectual property by the producer.

Similarly, to ASICs, FPGAs are widely underrepresented in their usage in the machine learning community because of their high cost of entry and difficulty to execute. However, they offer an open-source architecture to program, high throughput, low latency, low power usage, and are thermally efficient. Thus, FPGAs and GPUs offer the best architecture for emedded applications.

C. Machine Learning APIs

Most artificial intelligence models, as of the writing of this report, are trained using the python or C/C++ coding language with the following frameworks and APIs [14]:

- TensorFlow by Google

- PyTorch by Facebook
- CNTK by Microsoft
- Machine Learning Toolbox by Matlab
- Caffe
- Keras
- Theano

After training a model, it is optional to export the structure and weights into two standard compressed packages with the file extensions of *h5* or *ONNX*. An *h5* package is a legacy structure for compressing large amounts of different data formats. An Open Neural Network Exchange (*ONNX*) compressed package is a newer format for exporting AI models to make the import, export, sharing of models easier among different machine learning APIs [15].

D. *Power Hardware*

The team wants to implement a sustainable power source to supply the entirety of the system with solar panels where backup power is stored in automotive grade A batteries to use when the solar panels are not producing enough energy, due to a variety of environmental factors. The solar panels chosen are the BP350 panels. This particular panel can generate a maximum of 150 [W] with a solar energy conversion percentage of 24%. The solar panel and lithium battery pack will power the 23 [W] Jebao DCP water pump, 4 [W] Mingdak branded LED aquarium light bar, 36 [W] Xilinx ZYNQ XC7Z020 FPGA board, and two separate 5 [W] power rails for additional IOT componentry. In the case that the solar panel array was not obtainable, the team will use a portable power station to support the 200 [W] requirement of the aquatic identification and sorting system. The power bank chosen is the Floureon 42,000 [mAh] power station which features a two 120 [V] AC power outlets, 4 USB ports, with an optional solar input port to charge the power bank. Thus, the Floureon power bank is a versatile second choice that can be adapted to use the BP350 solar panels. The power bank offers a convenient pre-built suitable solution, which is produced by two vendors Floureon and Powkey.

E. Sorting Hardware

The AISS team selected three types of embedded system architectures to control the sorting hardware. The first option is to implement a Xilinx ZYNQ FPGA and use the boards integrated dual-core ARM Cortex processor to operate the servos. The second choice is to utilize an ATmega128a from Microchip that receives the servo control commands from an FPGA directly over UART. The third system is to use a GPU on a laptop, and have the laptop CPU communicate over UART to an ATmega128a to control the sorting servos. Due to the familiarity that the team has with inferencing on a GPU and working with motor control systems on an ATmega128a, this approach to implementing the AISS embedded hardware architecture was selected.

The ATmega128a's operating voltage is from 2.7 [V] to 5.5 [V]. The ATmega128a will be performing at 5 [V] instead of the optional 3.3 [V]. The ATmega will receive the characters "O" and "C" via UART to begin either an open or close operation of the Nema CNC stepper motors. The two 2.8 [A] stepper motors chosen for the AISS are 56 [mm] 4-wire stepper motors and offer high torque and accuracy. Controlling the position of the sorting door is a requirement of the Nema servo, which can move in 0.000625-inch increments. The holding torque of this motor is 1.26 [Nm], and each step is 1.8 degrees. This particular motor is capable of paring with the ATmega128a through the use of an externally powered H-Bridge. The approach to testing this motor architecture is to make sure that the motor is connected to a constant DC power source or chopper drive controller before connecting the Nema servos to the AISS power bank.

V. Implementation

Section V on Implementation covers the design constraints that the team created and encountered while working to develop the AISS.

A. Sorting System

The team's approach to implementing an aquatic identification and sorting system comes from the design created in figure 4 by John Schulz. The team focused the project around engineering the electrical related components such as the power system, camera module, computing architecture, and the stepper motor controller.

1. Sorting Vessel

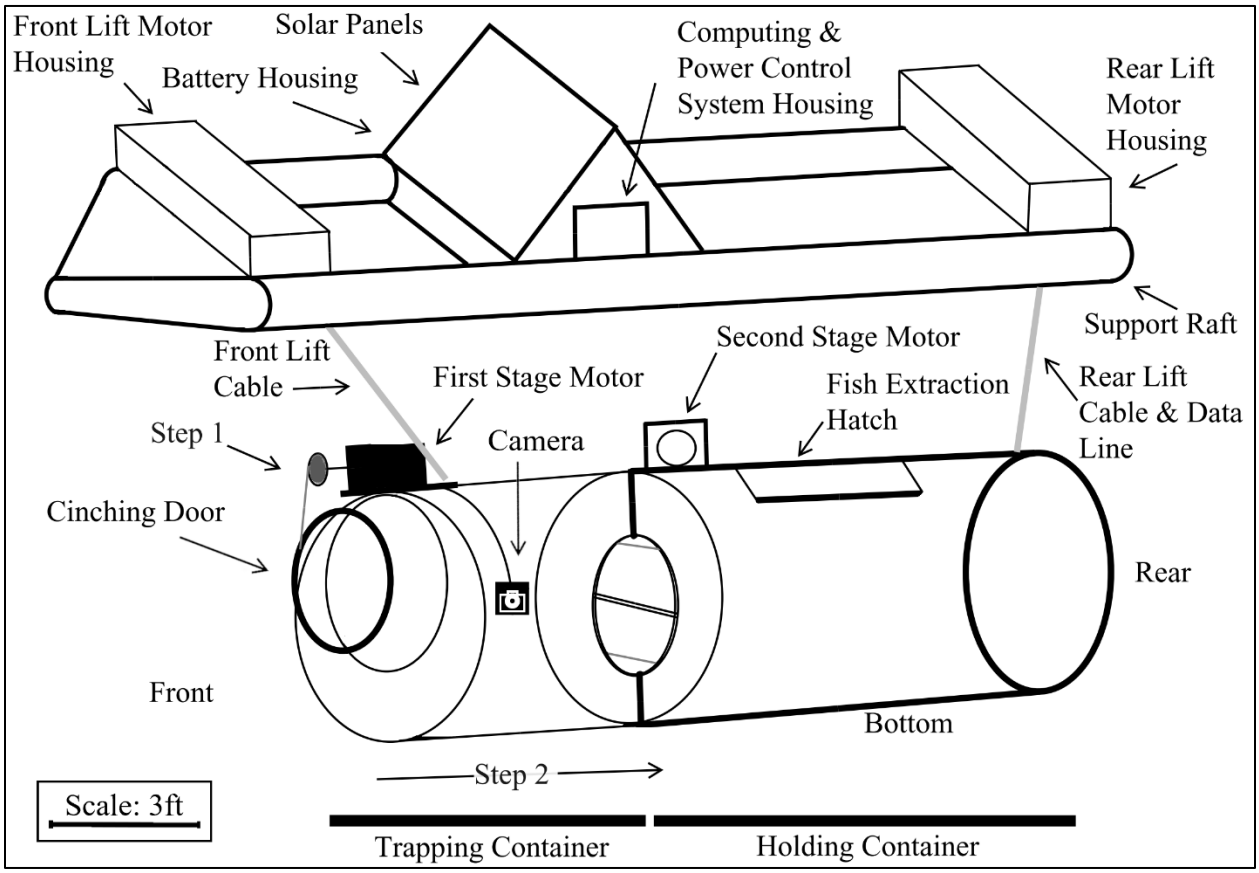

Figure 4 - Aquatic Identification and Sorting System

The entire system exists of two main components: the floating dingy and the submerged sorting container. The dingy is a vessel made of two pontoons that support:

1. The computing hardware for inferencing.
2. The power controller that supplies power to all electronics and charges the onboard battery from the solar panel.
3. The motors used to hoist the sorting container to the surface.
4. Keep the submerged container at a set height within the water column.

Intertwined in the *front lift cable* is a data line that connects the underwater electronic hardware and servos with the above water power system and computer.

2. Mechanical Sorting Process

The sorting process is a two-step process carried out by two submerged stepper motors. The inferencing system and camera start from its idle state when a sensor detects that an object is in front of the cinching door. Once a fish is identified inside the first chamber, labeled above as the *trapping container*, the motor in step 1 quickly closes the *cinching door*. After closing the cinching door, the trapped fish is unable to escape the trapping container. Then step 2 commences with the *second stage motor* slowly retracting the trapping container into and towards the *holding container* in the direction of the step 2 arrow. This operation is performed slowly over the course of 5 minutes to push the fish through the one-way door and into the holding container.

3. Sorting Algorithm

The steps of the algorithm in figure 5 are used to begin stage 1 of the sorting process.

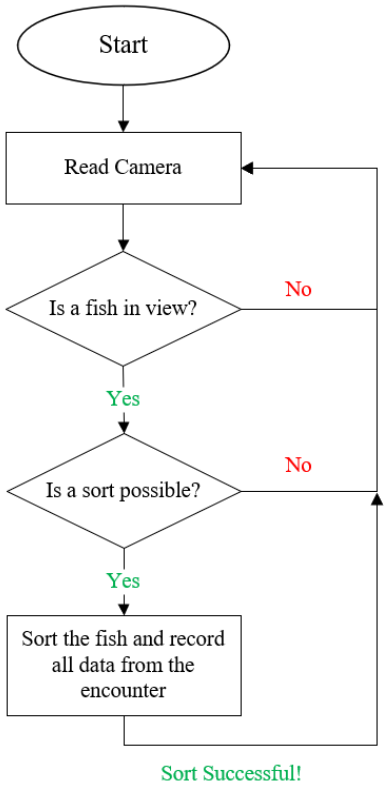

Figure 5 - Sorting Process

Start:

The start position in the algorithm, shown in figure 5, initiates after a sensor detects an object nearing the front of the cinching door; thereby, bringing the inferencing hardware and camera out of idle state.

Read Camera:

Once out of idle state, imagery from the camera is sent into the neural network for inferencing.

Is a Fish in View?:

This step is the most complex and involves receiving the accuracy percentages from the neural network. The algorithm is carried out on the CPU of the computer as follows:

> Central to the sorting process is the variable t_{stay} $[ms]$, which represents a threshold for the period of time $t_i - t_{i-1}$ that a fish classification remains the top prediction with an accuracy above 50%. A moving average filter takes into account the t_{stay} variable with the horizontal axis

of the filter window consisting of a 0 to 1 [s] time frame. The magnitude of the filter is calculated from the neural network's accuracy percentages that each have a range from 0 to 100%. When the accuracy percentages are above 75%, the t_{stay} values from the current window length at time t_i are totaled. If the total time exceeds the user adjustable variable t_{sort} $[ms]$ then a sort can commence. The variable t_{sort} $[ms]$ ranges from 0 to 1 [s].

Is a Sort Possible?:

This stage occurs before the mechanical portion of *step 1* begins. This part of the algorithm checks that a sort is not already in progress and that all mechanical systems are in a position for sorting to begin.

Sort the Fish and Record All Data:

Lastly, a sort is commencing during this stage. All video data and power metrics are recorded with time stamps. The collected data is critical for further training of the model, system debugging, researchers to analyze the data, and commercial fisheries to evaluate the harvesting of their crop.

B. Simulation Container

Utilizing a large 18"x26"x15" clear box, shown in figure 6 made from a food grade plastic to resist staining, this sealable container houses the environment used to create the database of lures that resemble real fish. The pump system is used to make the fake lure swim as water passes over the body of the lure and to circulate fine sediment debris that is added into the tank in stages of measured amounts. Imaging of a lure entails keeping approximately the same separation from the lens to subject while capturing the artificial bait from three different angles to improve neural network training. The minimal focusing distance of the camera lens sets how close a subject can be to the camera lens. After filming all 20 lures, then fine sediment debris can be added to reduce water visibility; then, the image capturing process will repeat.

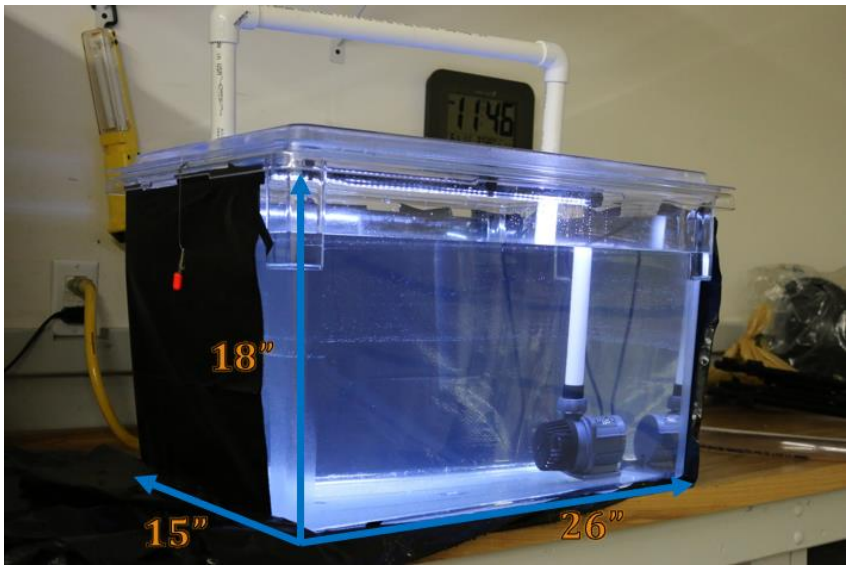

Figure 6 - Container Dimensions

A layering of black heavy-duty trash bags was cut and applied to half of the surface area of the container to reduce undesirable effects from external lighting, reflective glare on the subject and tank, and to limit background details that might confuse the neural network during testing. In the above image, no light blocking material is on the container’s frosted textured lid. The implementation of a controlled and constant background is meant to resemble the type of controlled environment that would exist in a fish sorting machine.

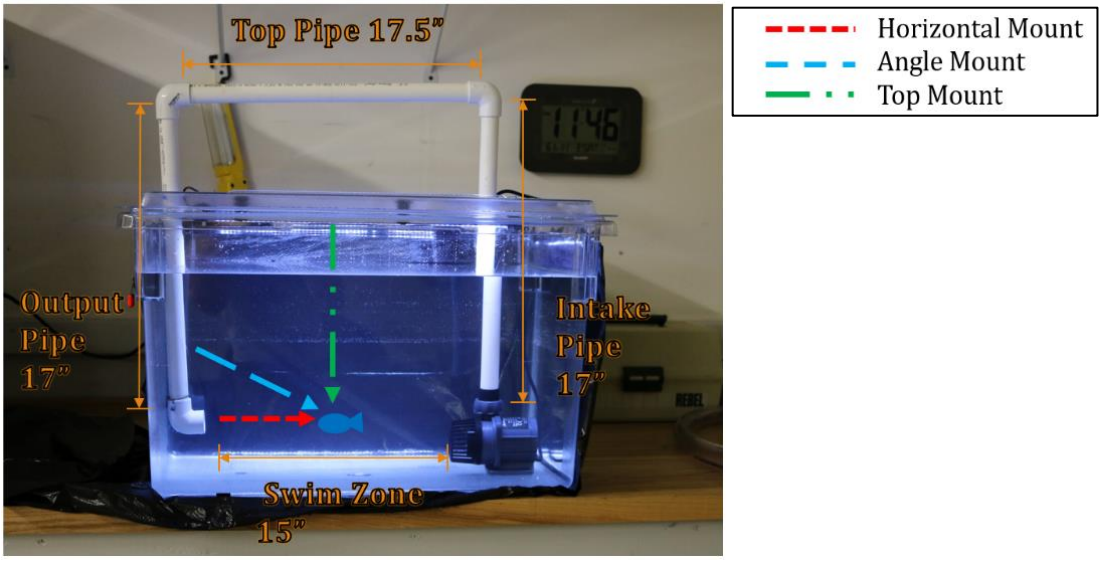

Figure 7 - Piping and Swim Zone Dimensions

Due to the circumference of the exit port on the sump pump, ¾" PVC piping was chosen for its rigidity, easy disassembly, and inexpensive cost. All tubing shown in figure 6-7 was cut with a hand saw, and hand fitted using no glue. The team cut three different lengths of output pipe: 14, 17, and 20-inches to provide further adjustment of the output nozzle height in reference to the bottom of the container. "Appendix III – Lure Datasheet," contains further information detailing the pipe lengths and mounting angles for each lure.

Figure 7 displays the attachment points for the clear monofilament fishing line to connect to the artificial bait to the top of the tank on the end where the water exits the hose. The swimming zone allows the lure to freely swim in the stream of water as would occur when reeling a lure on a fishing rod. Depending on the type of artificial fishing bait, some lures require different mounting points, as shown by the different dotted line patterns in the above figure. For example, the artificial baits that have their point of attachment on the top of their body, require a top mount solution. If a lure cannot permanently maintain its elevation from hitting the bottom of the container in a horizontal mount position but is not a top mount lure, then it can be mounted at an angle. The majority of artificial fishing bait on the market is designed to be attached using a horizontal mount to be reeled in from a perspective nearly parallel to the surface of the water.

C. Lure Design

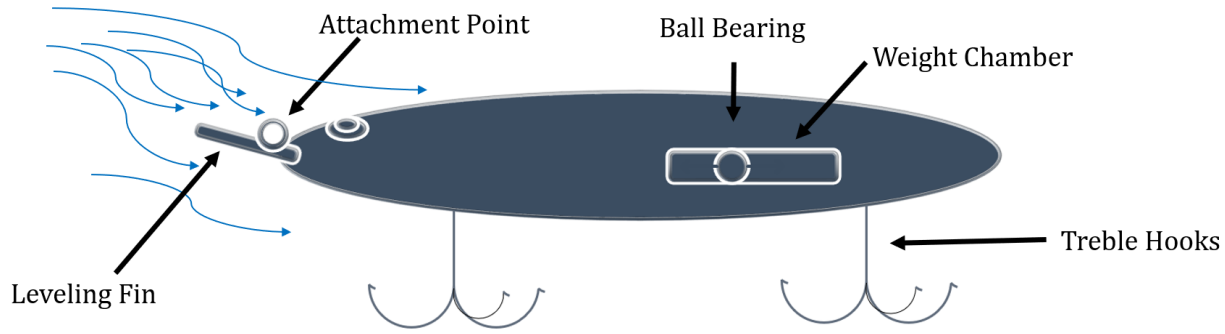

Figure 8 - Lure Components

Not all components shown in figure 8 are critical components to make a lure, but an attachment point and at least one hook must be a part of any sufficient design. For all angle and horizontal mounted artificial baits, the leveling fin is a hydrodynamic requirement to maintain an approximate height parallel to the water's surface. The larger the leveling fin, the deeper the lure will swim because it will create a larger

high-pressure zone on the top side of the spine, which will force the bait downward in the water table as water passes around its body. All classes of baits picked for this project are designed to create a shimmy motion from side to side like a dog wagging its tail. Some lures will contain a ball bearing to accentuate the side-to-side and vertical movements while swimming or to create a rattling sound that emulates a shoal of fish moving through the water.

D. Imaging System

A Canon 70D DSLR camera using a 35-50 mm, variable aperture, zoom lens was used to capture all the training data. For inferencing tasks, the Canon 70D is well suited to be used with an FPGA because it has an HDMI out port that can feed data into the programmable hardware. Whereas, in the use case of a graphics processing unit for inferencing, an internal webcam or external USB based webcam offers a more manageable setup experience. All imaging is to be shot at a resolution of standard 720p at 60 frames per second to help reduce blur of the lures swimming.

An internal LED strip mounted to the lid of the container aids in providing light to the top of the lure, which emulates real-world lighting conditions from the point of view of the camera. Also, a lid mounted light source allows for imaging to be done in a dark room; thereby, reducing reflections from the external environment. During filming, distance from the subject will remain as close as the focal length will allow without incurring image blur from the camera being unable to focus due to proximity. The video shot is stored on a solid-state SD card and then manually transferred to a computer for frame splitting and image processing.

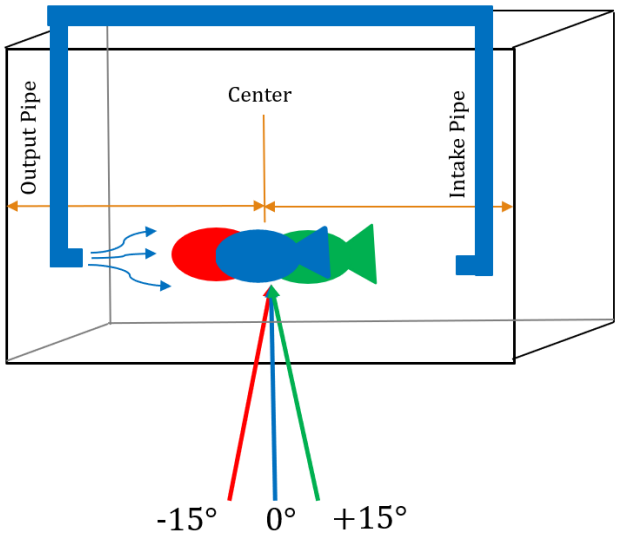

Figure 9 - Imaging Angle

Figure 9, shows the method designed to shift the subject away from the center of the container to eliminate the possibility during filming training data at alternative angles of having the reflection or physical view of the piping or the sump pump, in the images. The reason for filming the subject at different angles is to provide a variety of features on the lure that may not be present at angles other than zero degrees. Therefore, the shifting method provides a pseudo-3-D mapping of the lure due to the symmetrical nature of the subject.

E. Image Processing

Implementation of image processing in this project can be broken down into these four areas:

1. In-camera image processing
2. Post-processing of video
3. Processing of video into still frames saved as PNGs
4. Processing of PNGs into the neural network

1. Processing Pipeline

Stage 1: No application of in-camera image processing. The camera was set to default settings and filmed in auto-mode at 720p @ 60 FPS.

Stage 2: No post-processing was conducted on the video data once stored in non-volatile memory on the camera's SD card or laptop's SSD.

Stage 3: The video was processed into 8-bit RGB frames in the resolution of 150 x 200 x 3 using the Matlab function *imresize*. Images after this stage of processing are stored in the *RAWFrame* folder shown in "Appendix IV – Database Layout."

Stage 4: The 150 x 200 x 3 frames were processed such that each R, G, and B 8-bit values [0, 255] were converted to the closet 4-bit representation and then scaled to the same *uint8* range of [0, 255]. If an 8-bit pixel value is evenly split between two nearest 4-bit approximated values, the algorithm will round up with the *ceiling* function. Thereby, making images brighter, rather than darker when using the *floor* function. After the color depth conversion, the duplication of each image occurs, and the original image is stored along with an altered version of the original to add variety. The image processed to add database variety contains Gaussian noise using the *imnoise* Matlab function and then randomly processed to receive either a left-right flip affine transformation (*lrflip*), up-down affine transformation (*udflip*), or both. After this stage of processing, images are stored as PNGs in the *batch* folder shown in "Appendix IV – Database Layout."

2. Image Intensity Compression

Image Intensity is the number of levels of color accuracy provided for each R, G, and B color pixel value, which in this project is 256 levels of intensity. These values are typically unsigned 8-bit integer values ranging from 0 to 255. If an image does not have a large enough image intensity, false contouring will occur. False contouring is the appearance of lines of color that exist due to a lack of image intensity. The term color depth in this project refers to the number of bits per colored pixel, which in this project is 4-bits.

The design of the color compression process is to lower inferencing latency on an FPGA or a supported Tensor-based Nvidia GPU. Four bits of color allows for an adequate amount of imaging information to

enter the neural network. Since the artificial intelligence model utilizes convolutional layers, the system focuses on reoccurring patterns, and a 4-bit image should produce comparable results to a network using 8-bit imagery. The following four images were crucial in the design phase in helping the AISS team to compromise on the 4-bit color depth example in figure 12.

Figure 10 - Color Depth 2-Bit

One can see the apparent false contouring that occurs in figure 10, as the result of low RGB color depth. There is not enough level of color available to provide a smooth finish on the lure. The level of false contouring from the above figure is an attribute to avoid.

Figure 11 - Color Depth 3-Bit

The image in figure 11 also contains excessive levels of false contouring.

Figure 12 - Color Depth 4-Bit

At a 4-bit color depth, the lure in figure 12 provides a sufficient middle ground of low false contouring and a high enough level of image intensity. In a side-by-side comparison test, the team found it difficult to tell the difference between figure 12 and figure 13 once resized down to a resolution of 150 x 200 x 3.

Figure 13 - Color Depth 8-Bit

Thus, the color depth in figure 12 was chosen over figure 13 for implementation in the project.

F. Database

The database in figure 14 consists of all 20 classes of lures; however, neural network training will only occur on 17 of the 20 types. All images in figure 14 were shot from a camera on a tripod in a fixed position from the subject to provide the size comparison between lures. The collection of 20 artificial baits is designed to test the accuracy of the Convolutional Neural Network (CNN) in these aspects:

1. Obvious outliers

2. Distinguishing between similar sizes and colors
3. Testing 3 of the classes not trained, against their smaller duplicate
4. Ability to recognize images with only 4-bits of color per RGB pixel

Figure 14 - Sample Image of Each Neural Network Class

The reason for a few outlier classes, such as the mouse, worm, or frog, is to test the network on peak accuracy because these classes are highly recognizable. A few classes include a combination of matching in size and/or color with other classes; this aspect is meant to mimic the detection of hybrid species of fish like bluegills from sunfish bluegills. The three left out classes are designed to test the neural network against the possibility of classifying a species that is larger than those it has trained on previously. Lastly, all images sent into the neural network will be converted to a 4-bit color depth per RGB pixel. The reason for a color depth that is not 8-bit color per RGB pixel is to test the network's accuracy on the loss of 12-bits of color data per image pixel.

Figure 15 - Creation of Database Flowchart

The process of receiving the data from the camera and producing a trainable dataset can be seen above in figure 15. Ideally, the database should consist of at least 10,000 images per class. Thus, by keeping every 5th frame from every second of video recorded at 60 frames per second, then each class must be filmed for at least 14 minutes for all video that contains only clear water to reach 10,000 images value. Additionally, the subjects were imaged at different water clarities to increase the networks recognition accuracy in a real-world marine setting. Then the original pictures will be duplicated by MATLAB code, applied Gaussian white noise, and randomly selected for an affine transformation consisting of either a left-right, up-down, or both flipping operations.

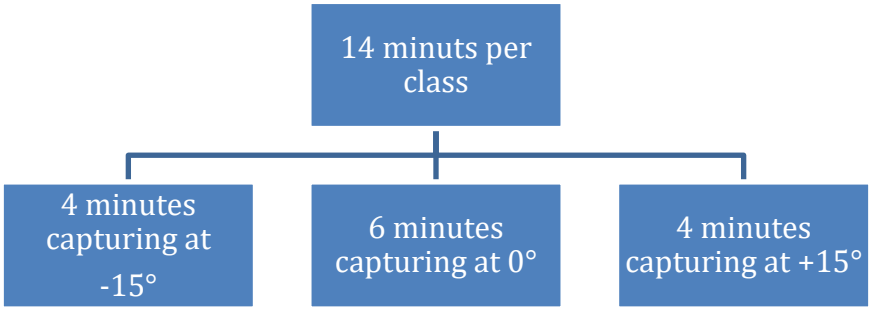

Figure 16 - Capturing Time Schedule

When filming for no debris footage, figure 16 provides time and camera angle details concerning the recording schedule for each class. Footage captured containing debris should be shot at zero degrees for four minutes. Ultimately the guidelines for the time constraints in figure 16 should produce a consistent database to utilize during training. The estimated size of the database with a duplicated set of images with noise and image rotation added combined with a total of two debris levels would result in a database of approximately 40,000 photos per class and a total database size of around 700 thousand pictures. For a complete overview of the file hierarchy created for the database used in this project, figure 41 under Appendix IV displays the naming structure and the file path relationship between the training scripts and the training, testing, and verification folder locations.

G. Embedded Neural Network

Typically, neural networks train and inference on GPUs; however, this project is focusing on utilizing FPGAs to perform the inferencing tasks. This project will use an embedded neural network, which means that optimizations of the model's structure are focused on meeting the requirements of a hardware system

with limited resources, like an FPGA. In designing the aquatic identification model, the basis of the network comes from the structure of AlexNet shown in figure 2. The original AlexNet model showcases: five convolutional layers in red, two normalization layers in light green, two max-pooling layers in yellow, one flattening layer in dark green, three dense layers, and one binary encoded output in gray at the end. AlexNet is a well-known image classification model that famously won the 2012 ImageNet LSVRC-2012 competition [5].

The structure of AlexNet best suites our use case because it is small in the number of layers and can be modified to fit onto a mid-level performing FPGA. Once on the FPGA, the neural network will utilize a signed-magnitude 12-bit integer bus to optimize the usage of the 24-bit multiplier blocks (DSP blocks). Thus, for every 12-bit number multiplied by another 12-bit number, only one DSP block is used. All neural network weights will be quantized into a 12-bit fixed sign-magnitude notation to save on logic cell usage. Reducing the number of convolutional layers down by two and dense layers to only two layers with 500 nodes each will further save on logic cells. Lastly, implementing the ReLu activation function on all layers except the last will minimize the need for costly look-up-tables (LUTs) that are an inefficient use of space.

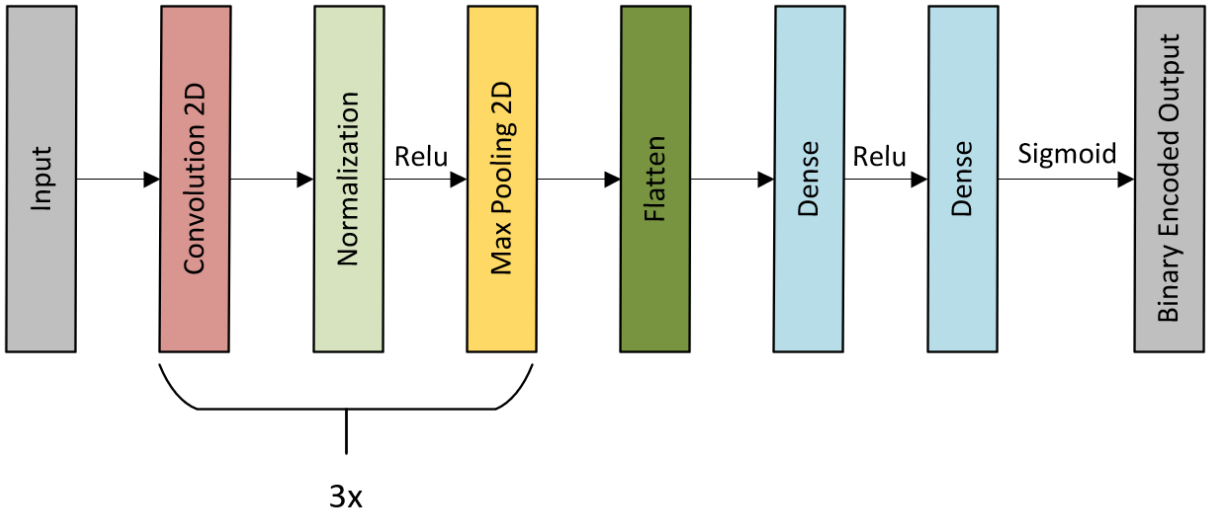

Figure 17 - Embedded Neural Network Structure

The result of using logic saving techniques is a modified AlexNet structure shown in figure 17. While this is a small network in the number of layers, the output accuracy will be hot-key encoded, so the evaluation

of the system is based off the model's ability to produce a logic ‘1’ for the correct class. For example, a real-world prediction of 95% equates to the same logic ‘1’ value if the prediction was only 75%.

The reason for choosing an FPGA for this project is because the implementation of the problem the team is trying to solve requires low latency AI inferencing. It is system critical when dealing with fast-moving marine species for the artificial intelligence system to decide within milliseconds to initiate a sort or prevent one. The second reason for choosing field programmable gate arrays as a platform is to offer future expandability for the addition of another camera to increase inferencing accuracy. The only reason this project would revert to using a GPU for inferencing is if the team is unable to fit the VHDL neural network model onto the PYNQ FPGA.

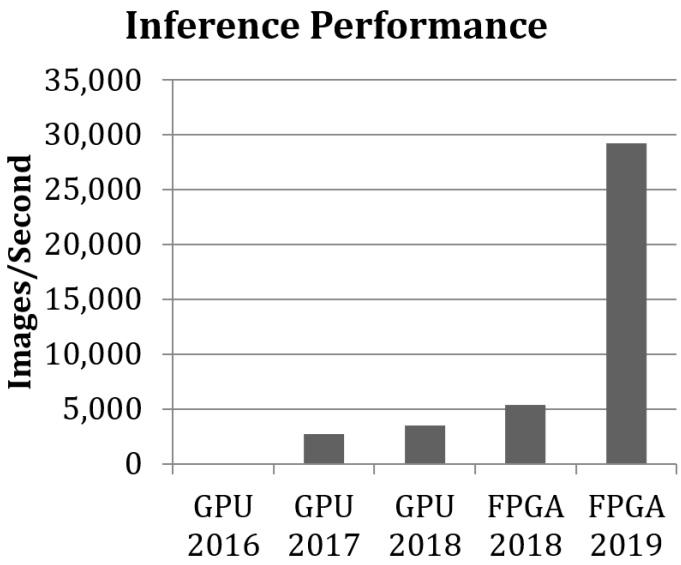

Figure 18 - FPGA Inferencing Performance [4]

Figure 18 includes data from the 2018 Xilinx Developer Conference that show how an FPGA offers unparalleled inferencing performance in the images per second that a network can decide upon [4]. Unfortunately, GPUs cannot provide both high throughput and low latency. According to Xilinx, FPGAs outperform GPU in performance per watt in AI applications by 16 fold. Thus, the primary implementation constraints of putting a model on an FPGA is enough logic cell memory (BRAM), an HDMI input for video, capable of inferencing at 60 images per second, and can operate at under 50 watts or less.

H. Software

The following subsections contain a variety of software programs, packages, and toolboxes required to implement this project.

1. Software Packages

Software required for this project include:

1. Windows 10 OS (Recommended)
2. Xilinx Vivado 2016 or newer design suite
3. MathWorks Matlab 2018b with the following:
    3.1. Communications Toolbox
    3.2. Computer Vision Toolbox
    3.3. Data Acquisition Toolbox
    3.4. Deep Learning Toolbox
    3.5. Fixed-Point Toolbox
    3.6. Image Acquisition Toolbox
    3.7. Parallel Computing Toolbox
    3.8. Text Analytics Toolbox
    3.9. Matlab Support Package for USB Webcams
4. Python 3.6 or newer
5. Rodeo IDE for Python
6. Machine Learning APIs (Python)
    6.1. Keras
    6.2. Tensorflow
7. Atmel Studio 7.0 with ATmega 8-bit microcontroller package

2. Machine Learning

Training of the neural network will be done on the lure database using an Nvidia GTX 1060 graphics card inside a 2018 Alienware 13” of the R3 generation. The Matlab Deep Learning Toolbox will be utilized to create and train the network on the graphics card. Exporting trained models consists of the network structure and weights in the machine learning standard *.onnx* compressed package file type. From there, the *.onnx* package is imported into MATLAB where software developed by the team will analyze the neural networks structure and generate VHDL code for the network to run on the FPGA.

I. Hardware

1. Power System Overview

Possible solutions to support an aquatic identification and sorting system include a renewable energy source combined with a backup battery support system. The reason that a battery support system is necessary is for two purposes: the harvesting of energy from a renewable source is inconsistent, or the renewable resource cannot supply enough peak current to the sorting hardware. Since a battery must be involved, the electrical system must provide power to the onboard electronic and sorting motors using discrete power regulating components.

2. Electronic Shorting Prevention

All external wiring and connections must be socketed with sufficient erosive resistant gaskets to withstand longterm water submersion and to prevent electrical shorts. Wiring that connects submerged electronics to above water electronics should be sealed within the conduit and securely mounted to the structure of the AISS. The goal is to prevent wiring from breaking due to vibration and from wire conduit wearing down causing a short with the metal structure of the sorting machine. Silicon-based dielectric marine grade grease to lubricate moving parts may provide further shorting protection. Dielectric grease is not water soluble and offers an alternative to silicone spray lubricants. Hot glue made for PCB boards or another type of silicone resin may be used to secure components to PCB surfaces, plastic, or fiberglass materials to reduce unnecessary vibration related wear.

3. Hydro-Generator

Since system deployment to a river is possible, a hydro-generator was researched to provide power to the system from the natural flow of water surrounding the sorting machine. Implementation of a hydro-electric generator offers a reliable and relatively constant source of energy. Conventional naming for a hydrogenator that would be used to satisfy an electrical load required by the AISS is often called low-impact hydro, micro-hydro, or run-of-steam hydroelectric generators.

A simple hydro-based system is possible to build into our project, but the team concluded that due to the time limit of this years capstone project, full implementation of the power system was not feasible. However, if one were to pursue this method, two main parts should be considered, as shown below in figure 19. First, the stator component inside a hydrogenator is a stationary part that is equipped with coils

of wires to collect electricity from the rotor disk. The rotor disk is the second component which is the moving component that rotates at high rounds per minute (RPMs) as water exerts force in the blades attached to the motor's armature. The magnets inside the rotor induce electricity in the coils from the stator. Finding multiple sources from IEEE offers a general idea of how to produce a hydro generator.

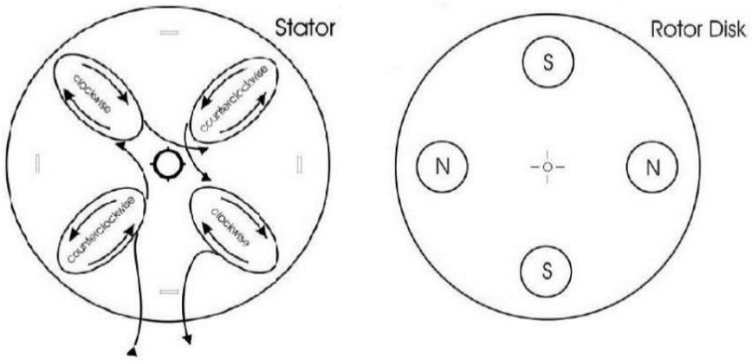

Figure 19 - Stator and Rotor Models [16]

The stator consists of four or more coils containing 200 turns each made of copper wire. The winding direction of the coils must alternate between clockwise and counterclockwise to ensure that an electron will follow the path of the Right-Hand rule; thereby, creating the motion in the magnetic field during a clockwise rotation for the magnets inside the motor generating electricity. To check if the coils were connected correctly, a digital multi-meter should produce a reading of around 10Ω or more. Next, the team thought about using neodymium magnets to create our rotor. The use of 4 or more neodymium magnets is sufficient to build the generator. Checking that the polarities of the magnets are in orientation with the opposite polarity of the previous magnet. Note that all magnets are facing the motor's armature, which is a crucial step in building a functioning generator. For example, the motor design in figure 19 has the magnets arranged with the polarities in an alternating (N-S-N-S) orientation forming a circle.

After assembly of the stator, rotor disk, magnets, and propeller; testing of the generator's performance can occur inside the simulation container with the pump providing the downstream pressure to spin the generator's armature, as shown in figure 3. The pump would supply a constant stream of force, where testing and recording of the generator's power output based on propeller types can occur. Testing can provide a baseline of the flow required for the sorting machine to function.

Hydrology has many statistical methods when creating a reliable hydrogenator. A book called “Statistical Methods in Hydrology” by Charles T. Haan provides mathematical tools that are helpful to engineers and hydrologists in demonstrating applications and methods to solve hydro-electric problems. Haan’s book is a valuable resource into the background, statistical analysis, and the probability associated with implementing a successful hydro-electric system, as covered in chapters 7, 14, and 15. Lastly, his book has been a great asset to the team and offers excellent insight for future Bradley engineering teams wishing to understand the viability of hydro-generators for use in their project.

4. Solar Panel

Since the AISS environment consists of waterways where the flow of water may not be enough to satisfy the needs of the system, solar combined with a battery backup offers an excellent alternative to a hydro-generator. In the past, capstone projects at Bradley Electrical Engineering Department have used a BP350 50 [W] solar panel. The dimensions of this panel are 33” wide by 24” tall. The last project to use a BP350 panel implemented a single panel connected to an automotive grade car battery to create a reliable power source. Unfortunately, the BP350 was not available to the AISS team. Thus, the analytical data covered in their project will form the basis for the aquatic identification and sorting system project.

Before implementing the fundamentals of their project into the AISS, data must be taken to find how much energy the sorting system hardware requires. This information is valuable in selecting the number of batteries, solar panels, and the size of capacitors needed to meet the power needs of the AISS project. In order to support high current draws when engaging the submerged stepper motors through a system of relays, capacitors can provide immediate power. When the solar panels are receiving enough sunlight to support the system the power controller will charge all batteries when the inferencing hardware is in an idle state. If the inferencing hardware is no longer in low energy consumption mode, the power controller will stop charging the batteries and switch all power resources to supply the equipment for sorting. In situations where the solar panel is not receiving sunlight, the power controller will run off the battery backup system.

The solar panel array that our team wanted to implement was a 200 [W_{max}] solar panel array, that was capable of supplying up to 12 [A]. The overall package of the system includes the following: a microcontroller, a discrete charge controller, wiring, capacitors, voltage regulators, and DC-DC converters. The microcontroller is used to read power sensors, control relays, estimate the reserve battery amount, and send power data to the inferencing system, which can be data logged. The discrete charge

controller is used to maintain the charge of the battery from the solar panel. The relays are used to switch motors permanently on or off and transition between either the solar panels or the battery source to power the inferencing hardware. The voltage regulators are used to maintain two constant 5 [V] rails. The first rail is for the submerged electronics and the second rail is for the above water electronics. Lastly, the DC-DC converter is used to convert between the batteries 12 [V] logic and the system 5 [V] logic.

5. Portable Power Bank

The aquatic identification and sorting system needs to be a hassle-free deployable system for any user. Since one of the design objectives is for the simple system disassembly, power system construction should include removable and repairable componentry. Four LED indicators showing the user the level of the battery in increments of 25%. The 42,000 [mAh] battery is made of a Grade A lithium polymer that only weighs 3.3 pounds. The whole power bank with controllers weight 4.2 pounds; thereby, making it easy to remove and replace, if necessary. This system would allow us to power our inferencing system along with other sorting components like a camera and two stepper motors.

Figure 20 - Portable Power Bank [17]

The portable power bank by Powkey in figure 20 was selected because the system was self-contained. For implementation in the AISS, direct wiring of the solar panel into the bank is the method that the team will

use to integrate both systems. The bank also supports dedicated 5 [V] rails over the USB ports. Lastly, the portable power bank is easily replaceable if damaged while deployed. Due to the Electrical Engineering department not having enough funding and no future use cases existing for the device, fulfillment of the team's request for the power bank in figure 20 was not possible. The team was notified to implement the system with a lab-based power supply by the lab director.

6. Artificial Intelligence Hardware

Inferencing is performed by a GTX 1060 GPU on an Alienware laptop running Matlab. Design of the Matlab script should include communication over USB-A to UART to issue servo control commands to the ATmega128a. Communication between the inferencing laptop and the ATmega128a microcontroller is performed using a baud rate of 9600, one parity bit, and an 8-bit data segment.

7. Imaging Hardware

A Canon 70D DSLR camera performs the imaging used to collect data with an 35-55 mm lens. The 35-55 mm lens uses a silent stepper motor (STM) for smooth focusing and includes in lens optical image stabilization.

VI. Results

The results collect occurred over the course of this capstone project from June of 2018 to May of 2019.

A. Sorting System

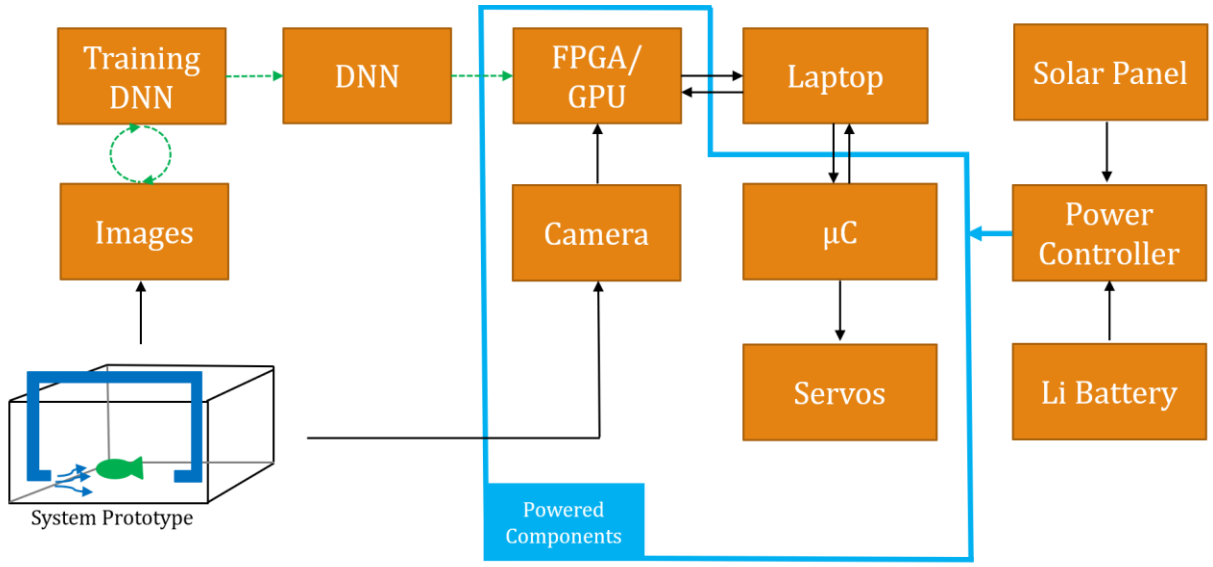

Figure 21 - High-Level System Overview

Figure 21 contains a high-level system overview of all the functioning hardware and software components implemented in this project. The hardware components in figure 21 represent the electrical related parts from figure 4 in the AISS sketch. The blue region shown above consists of the equipment that the power system developed can support. Due to a lack of development time, the decision was made in February to switch from developing the inferencing system on an FPGA over to a GPU. This choice was made by the team so that a functional inferencing system would be working in time for project judging in April. If FPGA development had continued, the laptop box in figure 21 would not exist because the dual-core ARM Cortex processor built into the Xilinx's ZYNQ line of FPGAs would replace the need for the laptop's CPU used to communicate with the ATmega128a microcontroller over UART.

The purpose of implementing a 16 [MHz] ATmega128a microcontroller, built by Microchip, is to control the servos and record data from the power controller. Also, the microcontroller offers another level of

protection from the more expensive FPGA because damaging an ATmega128a during development is not as significant.

B. Database

The database contains a total of 7 hours of footage with three levels of water clarity captured, which totals over 700,000 images covering all 20 classes. On average, each fishing lure category contains 33,000 images.

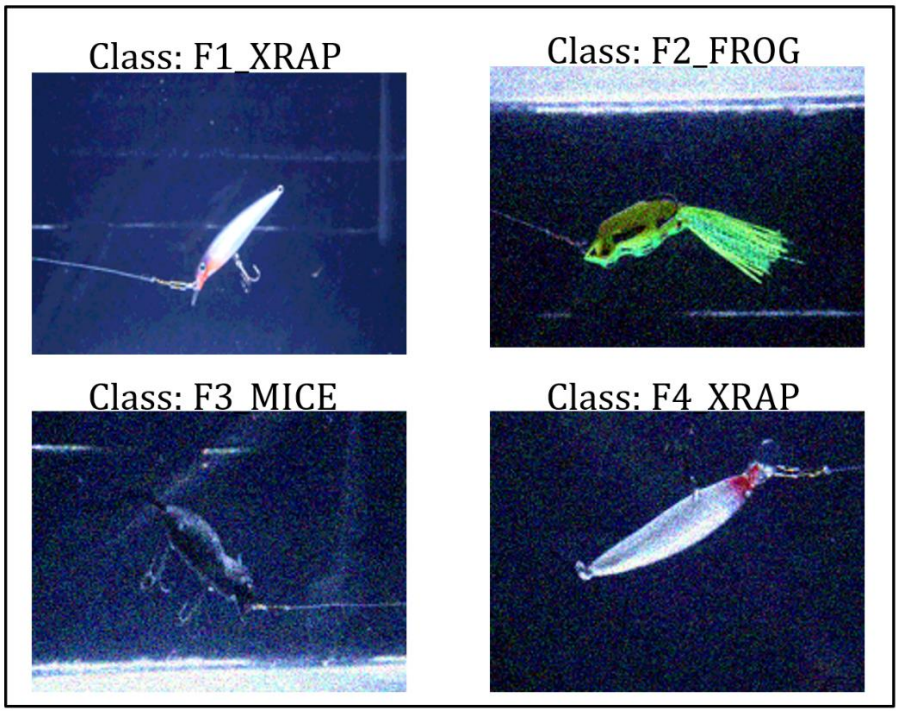

Figure 22 - Processing of Classes

Figure 22 contains images from the first 4 of 20 lure classes. All three image processing operations used to obscure the dataset in the second duplicate database image can be seen in classes F2 through F4, which is adding a combination of noise and a left-right or up-down flip. For example, class F2_FROG in figure 22 contains Gaussian noise, a left-right flip, and an up-down flip. Any processed image that has the string attached to the lure (i.e., leader) located on the right side of the picture, is an image that received a left-right flip. The picture of the class F1_XRAP contains no processing and offers an example of the original

image before white noise, and affine transformations were applied. The images in figure 22 and others are saved in the directory:

D:/data/batch/.../<class_name>/.png*

according to "Appendix IV – Database Layout."

Figure 23 - Graph of the Distribution of Images Captured Per Class

Figure 23 provides an overview of the distribution of the number of images captured per class. The reason this graph is significant is that the distribution among each class is relatively even. Only F3_MICE and F13_JERK standout as having more images in the database than the others. Overall, no lures contain a significant number of pictures trained on over another. During training, the limiting factor for the number of images per class allowed into training and testing data is calculated by the artificial bait with the least amount of PNGs. For example, F9_CD01 contains the least number of images then the user must set a value based on 70% or less than the total number of images collected for F9_CD01. Then Matlab uses

that value as a limit and only trains that many images per class, while the other 30% or more PNGs are used as untrained on (i.e., non-backpropagated) verification data. The outcome of this computation is that model's train on the same number of PNGs for each lure category; thereby, creating an evenly trained model. Additionally, the 70% limiting value fixes the issue of an unevenly distributed number of classes in a database.

In both Matlab and Keras models created for this project, dynamically feeding randomized data into the neural network model was implemented. It is important to note that frames were saved to their corresponding batch folders inside the database in a randomized manner out of their filming order. This method decreases the probability that any two images will be trained in the order in which they were filmed.

Filming time for each class in the database consists of a total of 14 minutes in clear water (cld_lvl 0), 4 minutes in semi-clear water (cld_lvl 1), and 4 minutes in cloudy water (cld_lvl 2). The time was shortened from 14 minutes to 4 minutes for debris-filled data due to the sheer amount of time the team took to collect the data. The team spent around 70 hours collecting this data because the only space for data collection was at John's house, an hour north of Bradley University. Filming could only occur at night to reduce the harsh reflections from exterior objects from appearing on the front of the simulation container.

Figure 24 - Fine Sediment Used for Cloudy Water

Figure 24 shows the fine sediment collected from John's family garden. Years of tilling this dirt allowed for a low build-up of non-decomposed solid materials, like leaves or sticks, to exist in the soil. The correct amount of sediment to add to the water would be to dampen two fingers and apply dry dirt to cover the two fingers. For electrical safety purposes, slowly insert one's hands in the tank with the pump off. The sediment will quickly rinse from the user's hands. The team found this method of adding debris to be a more controllable approach to fine-tuning the clarity of the water.

C. Neural Networks on FPGAs

A significant amount of time spent during the capstone project was used to develop a program in Matlab that receives a *.onnx* or *.h5* compressed neural network model and then converts the model to run on an FPGA in VHDL.

1. Exporting VHDL Models

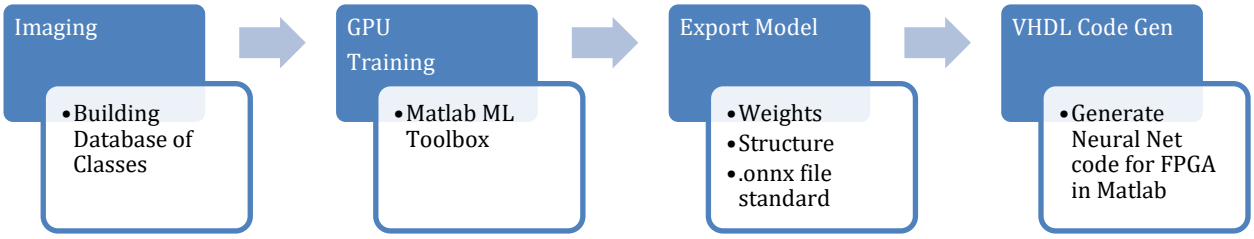

Figure 25 - Model Conversion Flowchart

Figure 25 contains the flowchart of the process to convert a trained model into VHDL code. The purpose of writing a high-level converter is to take advantage of Matlab's toolboxes to read the .json network structure, where the Text Analytics Toolbox is used to search for the layer types used by the training API like Keras or Tensorflow. Then Matlab's *importONNX* function from the Deep Learning Toolbox is implemented to read the weights from the dense layers, which then go through a quantization conversion process to match the user's requested customized sign-magnitude binary floating point (FP) structure. The selected FP width is applied to the bus of the neural network in the form of VHDL's *std_logic_vector* variables throughout the generated code. Thus, $A * B + C$ operations, activation layer Look-Up-Tables (LUTs), and dense layers are carried out by *std_logic_vector*'s with the same width as the specified quantified bus. The layers currently supported in the conversion software are ReLu, SoftMax, Sigmoid, Tanh, and Dense.

2. Program Generation

The conversion process in Matlab, shown in figure 26, starts by reading the user's input parameters, which includes the bus design and the name of models *.onnx* or *.h5* file. The *.onnx* or *.h5* should be placed under …/Main/model/h5/ and *.json* should be placed under …/Main/model/json/.

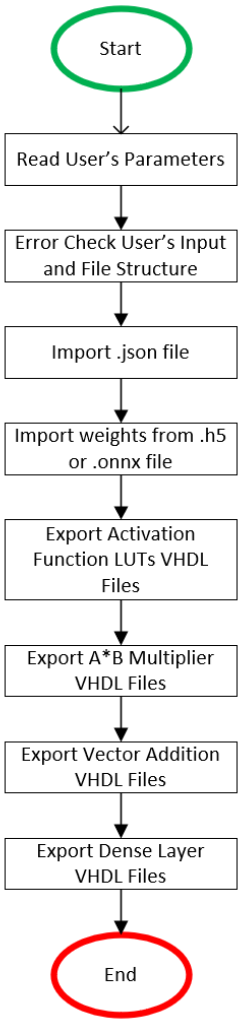

Figure 26 - Flowchart of VHDL Model Conversion Program

The script checks to make sure that the files entered by the user exist in the directory and the program checks that the bus value will work. Bus sizes can be up to 16-bit and must be even values greater than 3-bits. For example, FIX_12_8 will work, but FIX_11_8 is not supported. Import the JSON file is designed to get information about the layers of the network. Layers that the script supports will be generated as VHDL files. Next, the weights stored in either the *.onnx* or *.h5* files are converted into the user's

quantized fixed notation. Once the weights are converted, the generation of VHDL can proceed. First, the activation functions found from the *.json* are generated as LUTs and exported in a VHDL file under the folder …/Main/gen_vhd/ with all other created VHDL files. Second, the generation of multipliers occurs. Third, custom adders files are generated to handle vector addition in parallel. Lastly, the all *dense.vhd* file is written, which includes all the weights for each dense layer and implemented the previously written activation functions, multipliers, and adders.

3. FPGA Roadblocks

In this project, the approach used to generate a neural network model on an FPGA was inspired by the study of the human mind, which deploys a form of a full feedforward network in a non-linear manner. Although the nonlinear biochemical portion of the human mind cannot be implemented on an FPGA, the creation of the feedforward part is possible to recreate. The issue with trying to compute full feedforward inferencing on an FPGA in one clock cycle was unsuccessful in its implementation for four reasons:

1. The PYNQ board of ~100,000 logic cells could not hold a usable image recognition network.
2. DSP cells were exhausted quickly. Overstacking of DSP cells dramatically reduced the FPGAs clock cycle frequency. For example, the PYNQ board, utilizing its 100 [MHz] internal clock, ran a neural network that implemented an XOR logic gate with two hidden layers of 10 nodes each used more than 90% of the boards logic cells and ran at approximately half the internal clock speed [MHz].
3. The PYNQ board lacked enough BRAM, also known as block RAM, to hold all the weights of a deep neural network like the AISS model from figure 17 in logic cell memory all at one time.
4. The PYNQ board lacked enough available logic cells, after the implementation process of compiling the neural networks VHDL files, for an image processing pipeline to exist on the same board.

Ultimately, these four roadblocks caused the team in February to transition to using a GPU to meet the May deadline of having a working prototype of the AISS to present to the Electrical Engineering faculty and the Industrial Advisory Board (IAB) members.

4. Fixed-Point Neural Networks

Currently, neural network research related to FPGAs is experiencing a renaissance due to the needs of self-driving cars requiring low latency inferencing hardware. There are many different approaches to designing artificial intelligence systems, but the system the team built as referenced previously was a single clock cycle feedforward network. The neural net conversion software did not use any methods of probabilistic pruning to optimize the space taken up on the FPGA. Instead, the team focused on optimizing the inferencing pipeline by using smaller quantized bits in the AI's weights, bus size, and activation functions. For example, deeper models will fit better on an FPGA if the bus is in a FIX_8_4 representation and not FIX_16_10. The concept makes sense because the binary representation of the network's weights uses fewer resources during *Node*Weights + Bias* calculations. Likewise, decreased binary lengths allow for smaller activation function Look-Up-Tables. Additional gains can be made by using the ReLu function:

$$y = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Equation 1 - ReLu Function

The ReLu function in equation 1 does not require a Look-Up-Table and only requires logic cells for a comparison operation. Therefore, ReLu is the most logic cell efficient activation function for an embedded neural network, but forms of Leaky ReLu activation functions are not because they would require a LUT. Note that activation functions like Sigmoid, Tanh, or SoftMax are still necessary for output nodes.

Traditional activation functions like Sigmoid, Tanh, and SoftMax are calculated into Look-Up-Tables because BRAM based tables are quicker and do not require DSP cells. Although, Look-Up-Tables can be optimized in area usage for activation functions that are symmetrical around the y-axis, by using a DSP cell to implement a subtraction operation to get y-values where $x < 0$. In other words, functions like Sigmoid only require a Look-Up-Table for y values where $x > 0$.

Figure 27 and figure 28 illustrate the subtraction method used to optimize the size of Look-Up-Tables and show the accuracy difference between sigmoid in FP6 sign-magnitude FIX_6_3 form versus FP12 sign-magnitude FIX_12_8.

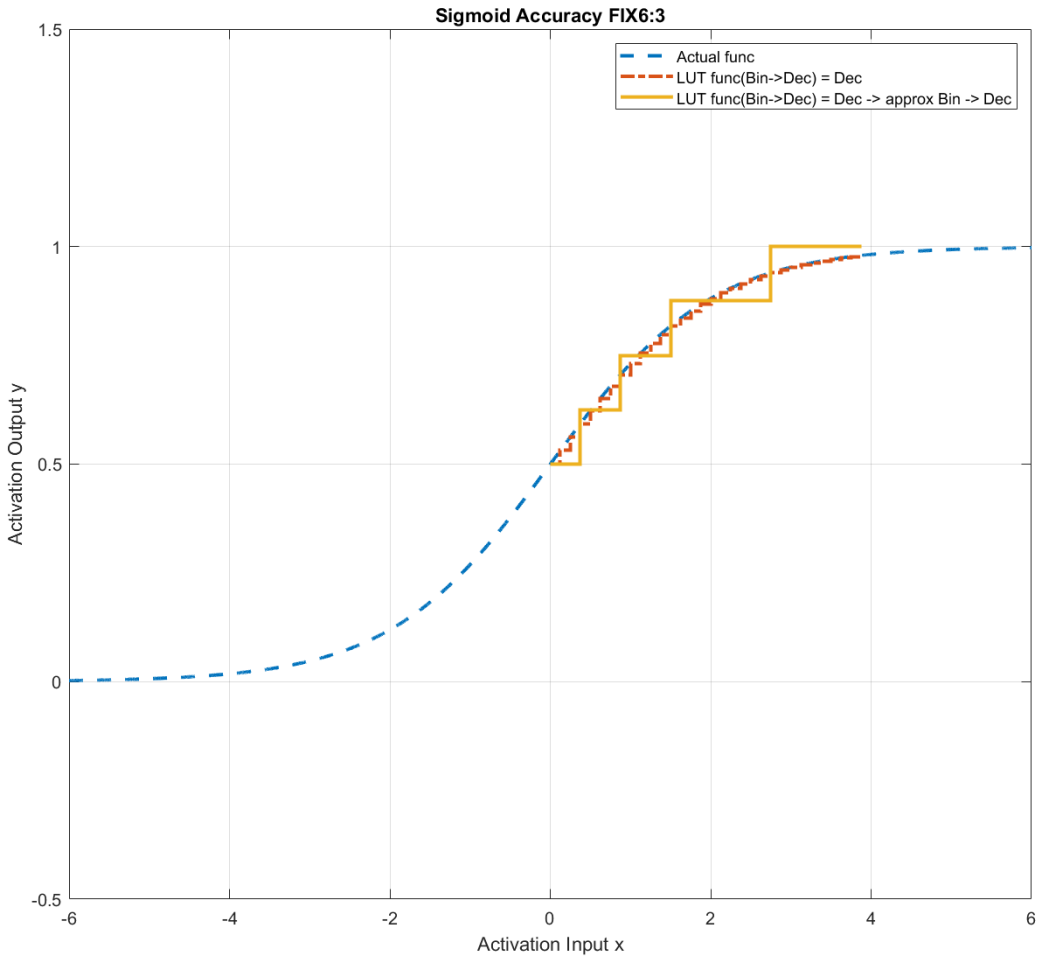

Figure 27 - FP12 Sigmoid in FIX_6_3 Notation

In both graphs, the blue dashed line is the actual sigmoid function calculated in Matlab's floating-point double accuracy. The red semi-dotted-dashed line is the sigmoid function derived from the entire binary range available in a FIX_6_3 number. The red line is meant to simulate all possible floating-point values that would result from an input bus in FIX_6_3 accuracy. Lastly, the solid yellow line represents the closes output representation of the sigmoid function with a FIX_6_3 input and its closes approximate FIX_6_3 output. The yellow line displays the actual accuracy performance of the sigmoid function on an FPGA.

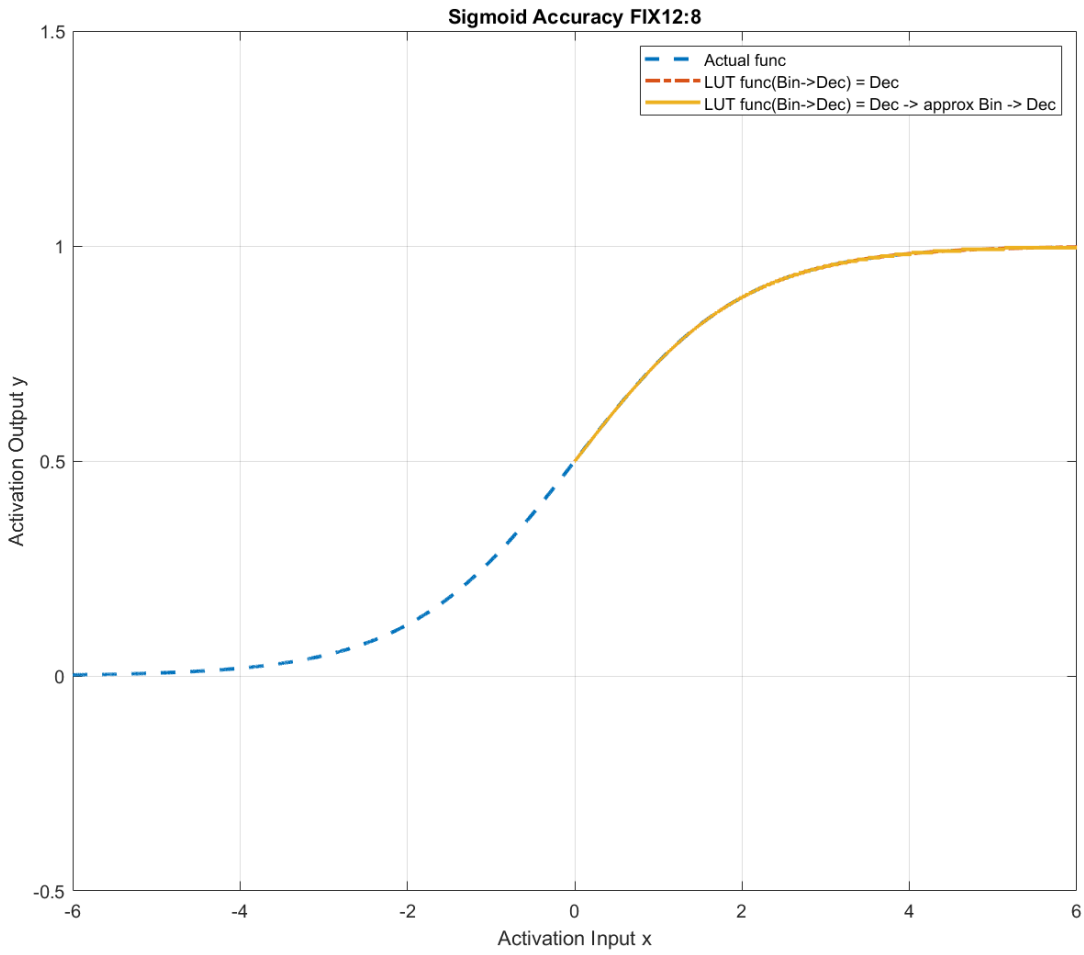

Figure 28 - FP12 Sigmoid in FIX_12_8 Notation

The calculation method for the above plot is the same as described in the description for figure 27, except the accuracy available to the right of the radix point is increased by 5-bits, and an additional bit has been added to the left of the radix point. Experimentation of different FIX notations occurred while modeling the Sigmoid, Softmax, Tanh, and ReLu functions, but FIX_12_8 provided the best overall accuracy performance in comparison to the actual sigmoid function calculation shown in the blue line in figure 28.

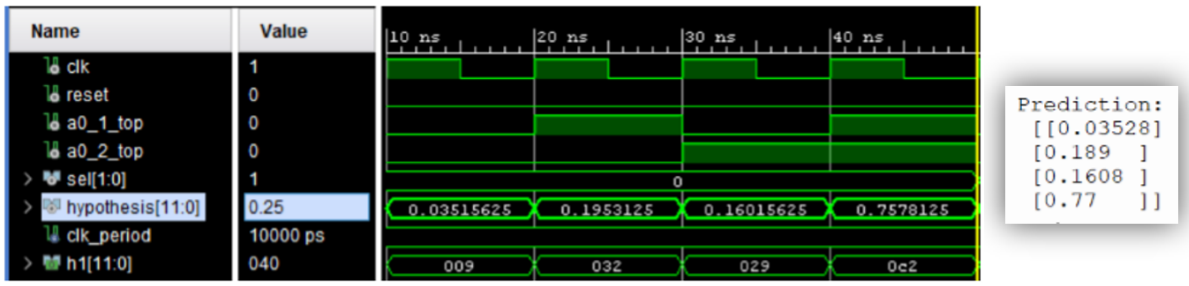

Figure 29 - Neural Network FPGA FP12 Accuracy

Figure 29 compares the simulation results on the left from a VHDL based neural network implemented with a sign-magnitude FIX_12_8 bus to the last predicted results displayed to the Python console window when the AND gate model was trained in Keras. The structure of the trained model used to implement the AND gate can be referenced in figure 30. The simulation shows the precision afforded by the FP12 notation. The resulting accuracy shown in figure 29 supports the team's choice in choosing FP12 as the optimal notation of preference because it offers more flexibility to move the radix point to the right or left by two bits while maintaining accurate results. It is important to note that fixed-point formats higher than FP12 did show values closer in precision to the Keras final prediction results at the expense of logic cell usage inside the FPGA. Bus lengths of 10-bits in an FP10 structure did produce similar accuracy to FP12 and may also provide enough precision to satisfy the needs of image recognition applications.

Dictating the bus structure required for the implementation in the AISS is the number and maximum value of the bits allotted to image intensity in the images that the neural network trained on because the weights become adjusted to the magnitude of these values. Since the team chose to train the models on 4-bit color accurate PNGs scaled to an equivalent value of 0 to 255 for simplistic viewing in Matlab and on the PC for demo purposes, the neural network must be retrained for an FPGA implementation with RGB colors that range from 0 to 16 instead of scaling them to a range of 0 to 255. In other words, the FP structure must be set up to handle the imagery specifications that the neural network experienced during training. The following two areas should be considered when choosing a fixed notation bus:

1. Integer range going into the neural network.
2. Placement of pixel values relative to the radix point.

For example, when creating a fixed-point bus based on an integer range, an FPGA image recognition network that uses a sign-magnitude FP12 bus would require a FIX_12_7 notation to allow for 4-bits to the left of the radix point to be available for the 4-bit, 0 to 16, color values of an image.

Likewise, further optimizations are available when considering the placement of *uint8* pixel values coming out of the camera to fit the data around the number of bits available to the right of the radix point. The positioning of bits inside a *std_logic_vector* representing a fixed notation vector is critical in optimizing bus widths for pixel data lengths that cannot fit left of the radix point but can to the right.

For example, in a sign-magnitude FIX_8_4 structured bus, the network's designer may want to compress the training data from the original 0 to 255 8-bit RGB images to 4-bit 0 to 16 values and then shift the 4-bit integer color space to the right of the radix point for a color representation in levels of 2^{-4} to take advantage of the 4-bits available to the right of the radix point. This second approach allows for the most efficient use of bus space with no additional image processing overhead in comparison to the integer only consideration method. The VHDL code used to handle the image processing that would compress the color depth of the image for the neural networks bus was only theorized in this project. Nonetheless, bus design plays a critical role in the VHDL code generation process.

5. Neural Network Computing Architectures

To better understand the team's approach to generating a neural network's computing architecture on an FPGA, the neural network shown in figure 30 provides an example.

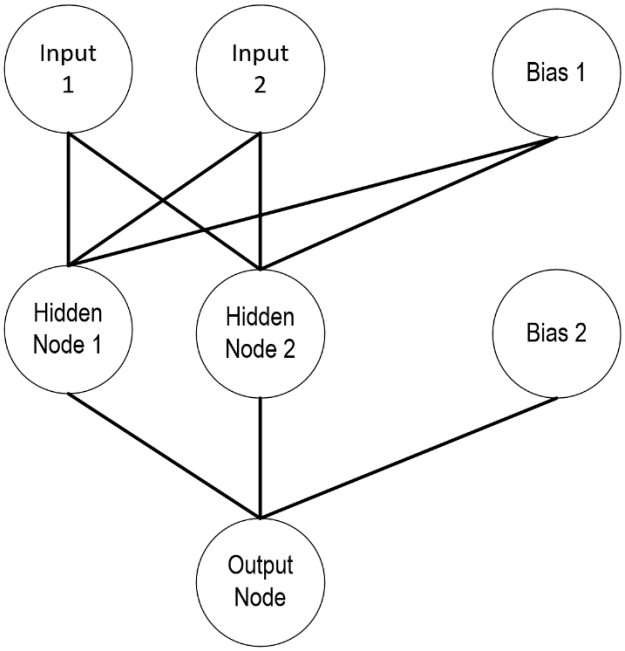

Figure 30 - Logic Gate Neural Network

Figure 30 displays is a shallow neural network that implements a logic gate like XOR, OR, AND, and others. The structural hierarchy of the system contains a single output node with two input nodes representing either a 0 or 1 value. Implementation of the above model ran successfully on both a Digilent Nexys 4 DDR3 and Avnet Zedboard FPGA using external interfaceable switches to control the two input values and LEDs to show the output node's fixed notation results. Both figure 31 and figure 32 discuss different methods to implement the above model.

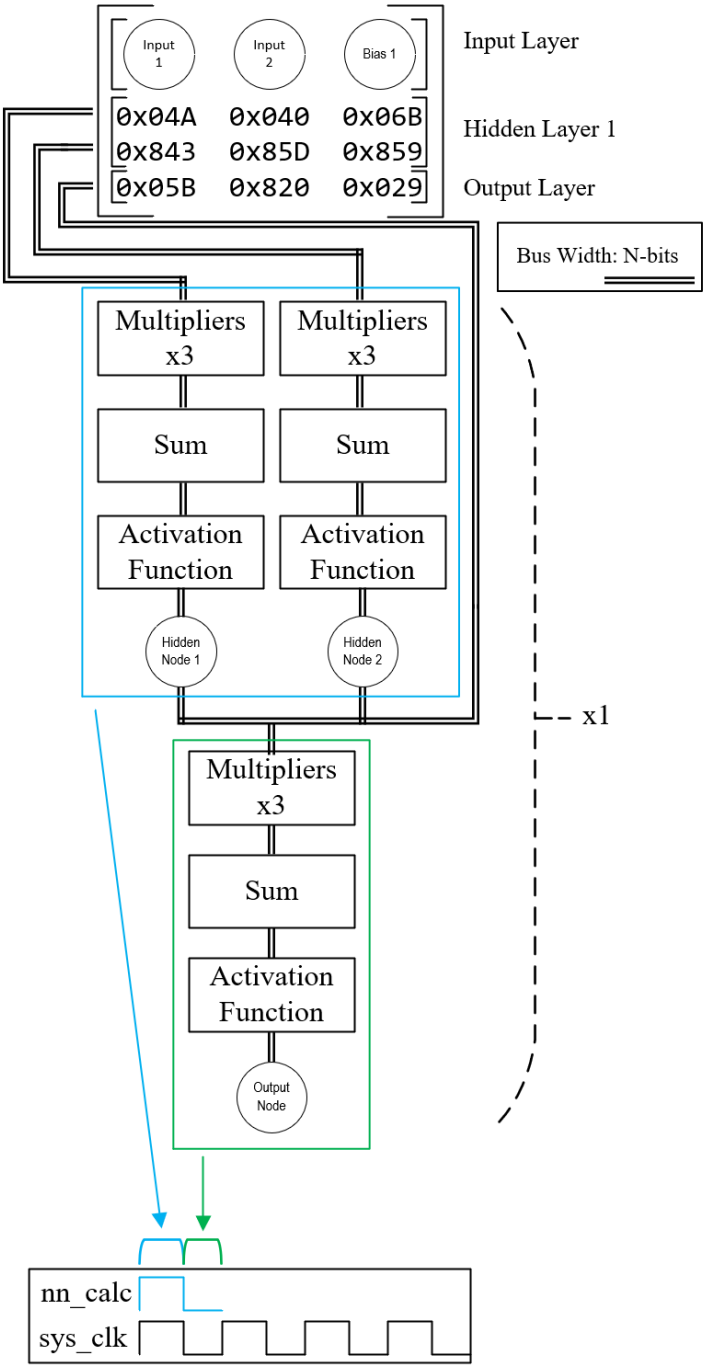

Figure 31 - Current Network Generation Architecture

Running the network from figure 30 through the Matlab conversion script builds the VHDL code in a manner, which creates the single-clock cycle inferencing network shown in figure 31. This approach results in a neural network that operates in the way that the brain does, but is extremely inefficient in

terms of logic cell usage on an FPGA. The architecture of this VHDL model will not calculate a value in one clock cycle and then finish the calculation in another. Although, this architecture results in incredibly quick inferencing, it cannot be realistically implemented for deep neural networks because current FPGA chips do not contain enough logic and DSP cells. Even if this architecture was possible, it would not be energy efficient in an embedded product or needed to perform millions of inferences per second for edge based inferencing tasks.

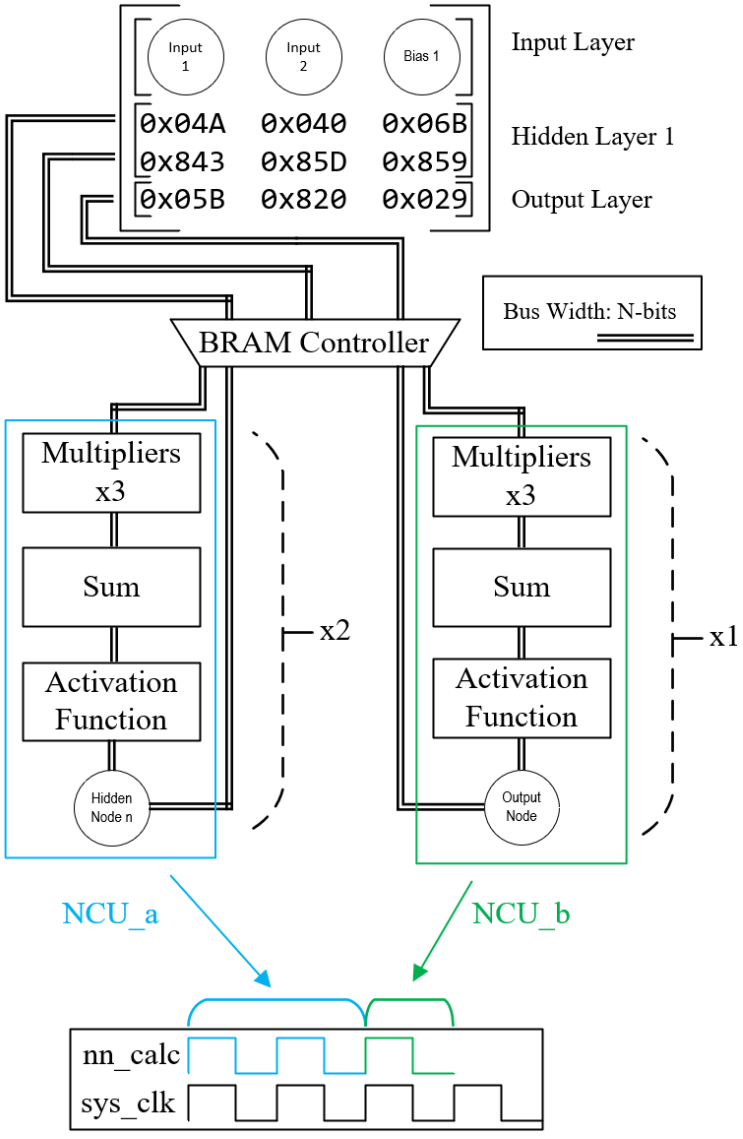

Figure 32 - Efficient Network Generation Architecture

Although not implemented in this project, the approach to generating the neural compute unit (NCU) architecture theorized in figure 32 provides a solution to optimize the issues plaguing figure 31. This second architecture applies the same fundamentals of its predecessor, but focuses on controlling the number of computations per clock cycle using neural compute units. This concept decreases the need for long computational routes within the FPGAs fabric of logic cells; thereby, increasing the operating clock frequency closer to its peak value. Both neural compute units, *NCU_a* and *NCU_b*, operate at different times depending on the number of nodes to calculate and the number of DSP cells available. The number of DSP cells available on an FPGA must be provided to the Matlab script by the user from an FPGA's datasheet. The NCU's are controlled by the BRAM Controller that selectively chooses the number of weights multiplied by a node or nodes per clock cycle. The compute units differ depending on the shape and size of the matrices that each must calculate. The overall goal is to create a standard reusable neural compute unit unique to each model and layer size to conserve on logic cell usage.

The difference between the approach in this capstone project and other architectures like Nvidia's Tensor Processing Units is the customizability to optimize a model to the end-hardware for edge-computing applications. For example, consider the following design parameters:

- an FPGA has 2,000 DSP cells that can multiply two 16-bit values per cell
- the FPGA is operating on a single dense layer of 500 nodes in the AISS model
- the bus width is 16-bits of some FIX_16_X notation
- the sigmoid activation function is required, which uses an optimized LUT that takes advantage of sigmoid's y-axis symmetry.

Computing a single node in the next hidden layer of 500 nodes each would require 501 multiplications, 501 additions, and one possible subtraction for the activation function if the input value is negative. Thus, the calculation of almost two nodes occurs in a clock cycle. The whole layer of 500 nodes with a ReLu activation function can be computed in under 251 clock cycles or 3.35 [us] with a 75 [Mhz] system clock. Although the architecture in figure 32 is depicted in a configuration to solve the neural network model from figure 30, the NCU configuration offers the theoretical flexibility to increase the computational throughput of an FPGA to handle large dense layers required for image recognition and deep learning applications.

D. Artificial Intelligence System

The artificial intelligence portion of this project plays a significant role in the sorting process because without image recognition, capturing fish would be an indiscriminate operation. This section covers the software used for inferencing, plots of training accuracy, and GPU inferencing performance. While small neural network models were generated to run on an FPGA, large image recognition models would not fit with the current hardware and software written for this project. Thus, a GPU was used to achieve a working demo of the artificial intelligence system by the end of the 2019 Spring semester. The model shown in figure 17 is the same structure used for the final build of the project. The resulting neural network used three convolutional layers and two dense layers with 500 nodes each.

1. Software

Initially, the machine learning API Keras was used to implement the model in figure 17, but Keras could not fit the model onto the 6 Gb of VRAM inside a GTX 1060 graphics card. Thus, the team switched to using Matlab's deep learning toolbox to train the neural network.

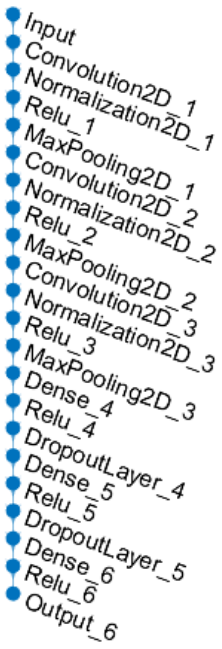

Figure 33 - Embedded Neural Network Hierarchy

Figure 33 offers a hierarchical view that displays the order of the layers inside the embedded system from figure 17 in descending order from input to output. The Stochastic Gradient Descent Method (SGDM) is

the cost function used during the neural network's training to minimize errors. While training, image data was sent from the database into the model using the Matlab function: *imageDataStore*. This function uses pointers to tell the model the location of the training data on the computer. However, before training can commence the *imageDataStore*, must be customized to randomize the data and decide the percentages of images in the database that will be training data or verification data. Unlike other machine learning APIs, Matlab only uses training and verification data. Matlab uses a portion of training data as testing data.

Setting up the layers of the model, as shown above in the embedded hierarchical plot, were accomplished using the following four function sets of code (purple variables may change in the additional layers not shown):

1. Input Layer

```
imageInputLayer( [ 150, 200, 3] ,'Name', 'Input');
```

2. Convolutional 2D Layer

```
convolution2dLayer(11, 64, 'Padding', 'same', 'Name', 'Convolution2D_1')
batchNormalizationLayer('Name', 'Normalization2D_1')
reluLayer('Name','Relu_1')
maxPooling2dLayer(3,'Stride',3, 'Name', 'MaxPooling2D_1')
```

3. Dense Layer

```
fullyConnectedLayer(1000, 'Name', 'Dense_4')
reluLayer('Name', 'Relu_4')
dropoutLayer(DropOutRate,'Name', 'DropoutLayer_4')
```

4. Output Layer

```
fullyConnectedLayer(Classes, 'Name','Dense_6')
softmaxLayer('Name', 'Relu_6')
classificationLayer('Name','Output_6')];
```

Beginning training starts by using the Matlab function:

```
NN2 = trainNetwork( <data pointer>, object_layers, objects_training_options).
```

Exporting the model after training as .mat file and a .onnx file:

```
save(Model_MatPath, 'NN2');
exportONNXNetwork(NN2,  ModelPath);
```

Note, '*NN2*' is the trained model containing the layer types and weights. For training to continue at a later date, Matlab's *load(Model_MatPath*) must be used to import the .mat file. Trials by the team to import an *.onnx* file for continued training were unsuccessful, but continuing training from a *.mat* file did not cause any issues.

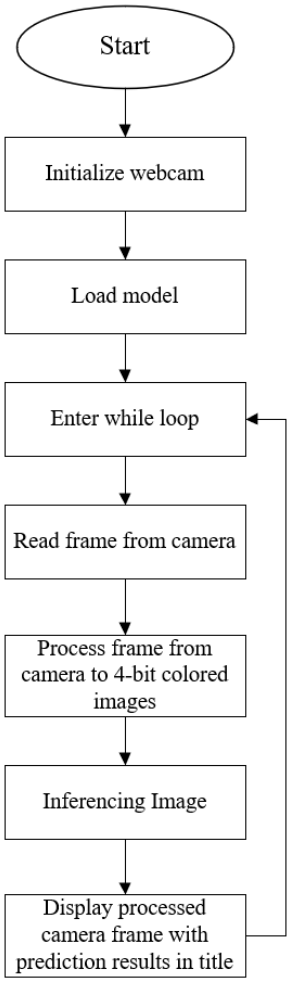

Figure 34 - ML Vision Flowchart

The flowchart in the above figure shows the method implemented to test the neural network with live imaging data coming from the laptop's webcam. Inferencing occurs by loading the model's .mat file from training and using the function:

```
[pred, confMatrix] = classify(NN2, frame_s)
```

This function returns the top class name as a string and a confidence matrix, which consists of one row and 17 columns of floating point prediction results.
The image processing pipeline used to create the 150 x 200 x 3 input size and adjust color intensity levels to the specifications that the neural network trained on is shown in the following lines of code:

```
%------------------------------------ Get Foreground
frame = snapshot(cam);

%------------------------------------ Process Image
%frame_s = imresize(frame, [150, 200]);
frame_s = frame(95:244,70:269,:);

frame_s(:,:,1) = gray2ind(frame_s(:,:,1), 16);
frame_s(:,:,2) = gray2ind(frame_s(:,:,2), 16);
frame_s(:,:,3) = gray2ind(frame_s(:,:,3), 16);
frame_s = frame_s.*(255/15);
```

First, the newest frame is accessed from the camera's frame buffer; then the image is cropped to the input size from the camera's native 400 x 400 x 3 resolution to 150 x 200 x 3. The reason that the image is cropped instead of resized is to remove the possibility of image distortion impeding prediction accuracy when using Matlab's *imresize* function. Secondly, cropping provided a more accurate distance from the lens to the subject as used during the filming of the training database. Next, each R, G, and B frames are separated and converted to 16 total image intensity levels. Lastly, the picture is scaled back to the full *uint8* image range to match the training data format.

The team did find changes to the webcam's white balance necessary by forcing the camera into a manual white balance mode to keep the webcam from attempting to correct and normalize the blue levels in the

video. This change allows for the camera data to more closely representing the training data; thereby, increasing the model's accuracy.

2. Training Accuracy

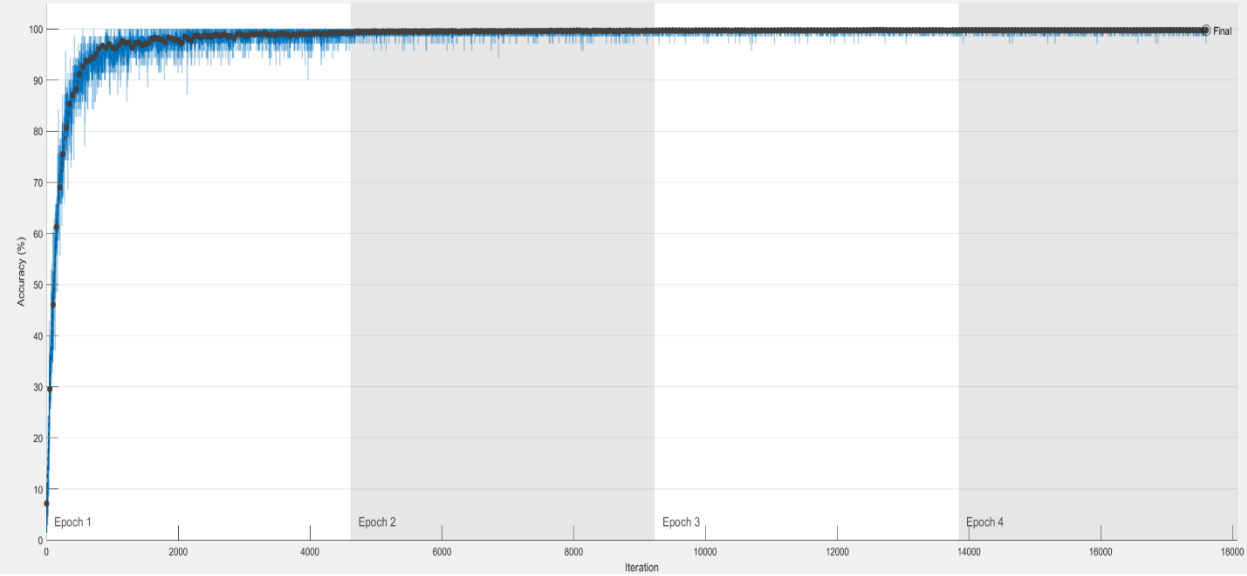

Figure 35 - Training Graph

Figure 35 shows the accuracy graph plotted during the training of the final model used in the demonstration to the faculty and IAB members. This model is learning on 17 of the 20 classes. After training for three days, the neural network stopped improving in verification accuracy, and the patience limit invoked the model to quit. A patience limit is a design attribute setting that causes a training model to stop learning once the accuracy stops increasing after a set number of iterations, which is 100 in the case of the above network.

Results	
Validation accuracy:	99.75%
Training finished:	Met validation criterion
Training Time	
Start time:	05-Mar-2019 15:25:13
Elapsed time:	4172 min 39 sec
Training Cycle	
Epoch:	4 of 10
Iteration:	17600 of 46140
Iterations per epoch:	4614
Maximum iterations:	46140
Validation	
Frequency:	50 iterations
Patience:	100
Other Information	
Hardware resource:	Single GPU
Learning rate schedule:	Piecewise
Learning rate:	0.00125

Figure 36 - Training Parameters and Results

Figure 36 contains statistics and parameters associated with the plot in figure 35. The model's accuracy performed exceptionally well with a 99.75% validation accuracy at the time of quitting. This model contains 17 of the 20 classes with CD09, CD03, and ST05 left out. The model went through 17,600 iterations consisting of batches with 100 images loaded to the GPU at one time. The network had access to 16,000 pictures per lure or 70% of the available training data based on the class with the least amount of images. Every 50 iterations the network was set up to test itself against validation data. The learn rate started at 0.01, and every epoch, after the first, the learning rate was dropped by half of its previous value.

3. Model Performance

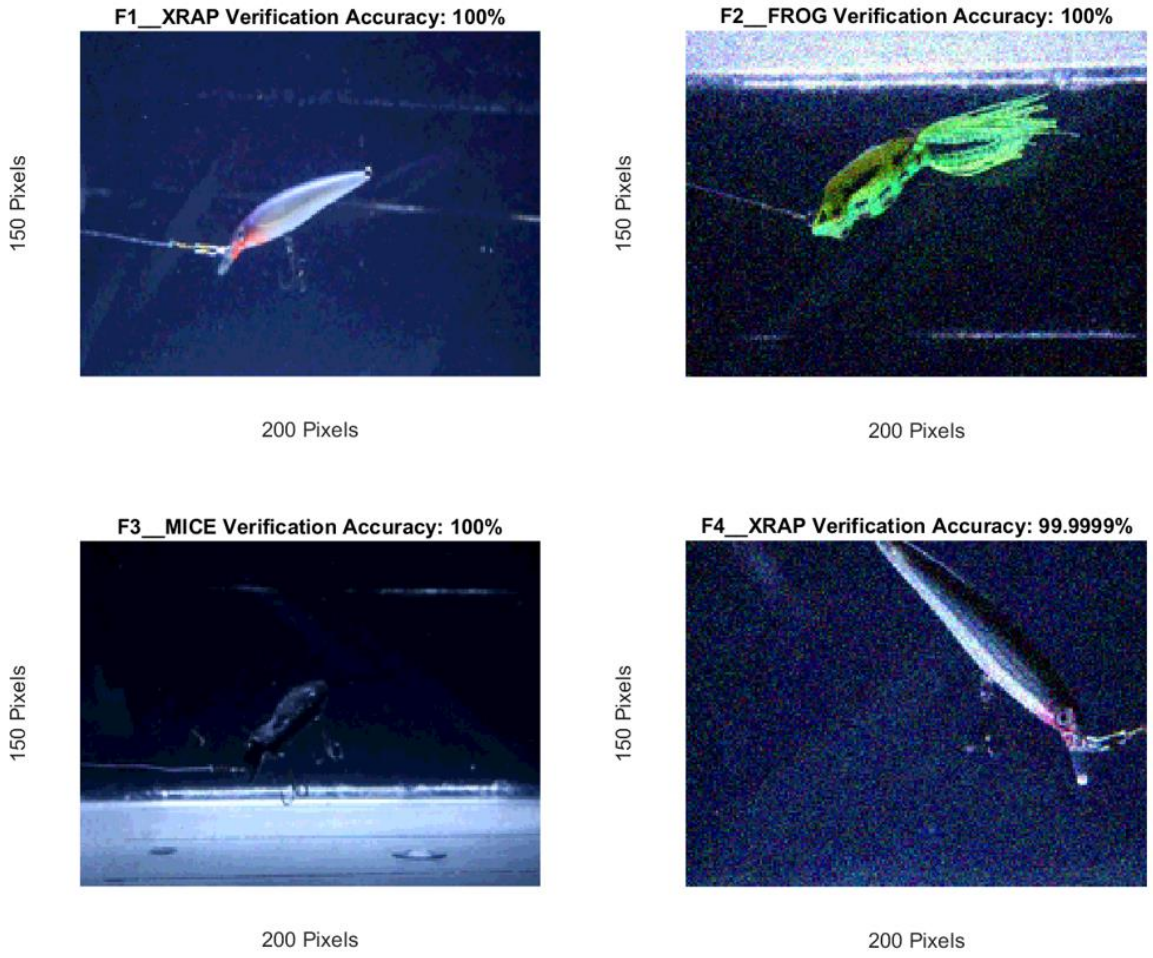

Figure 37 - Testing Verification Accuracy

The above picture contains images from the first four classes that the neural network saw during training. One should expect to see accuracy that correlates to the values observed at the end of the training plot during the fourth epoch in figure 35. Thus, the team can verify the model's successful training due to the percentage and correct classification result shown in the title of each image in figure 37.

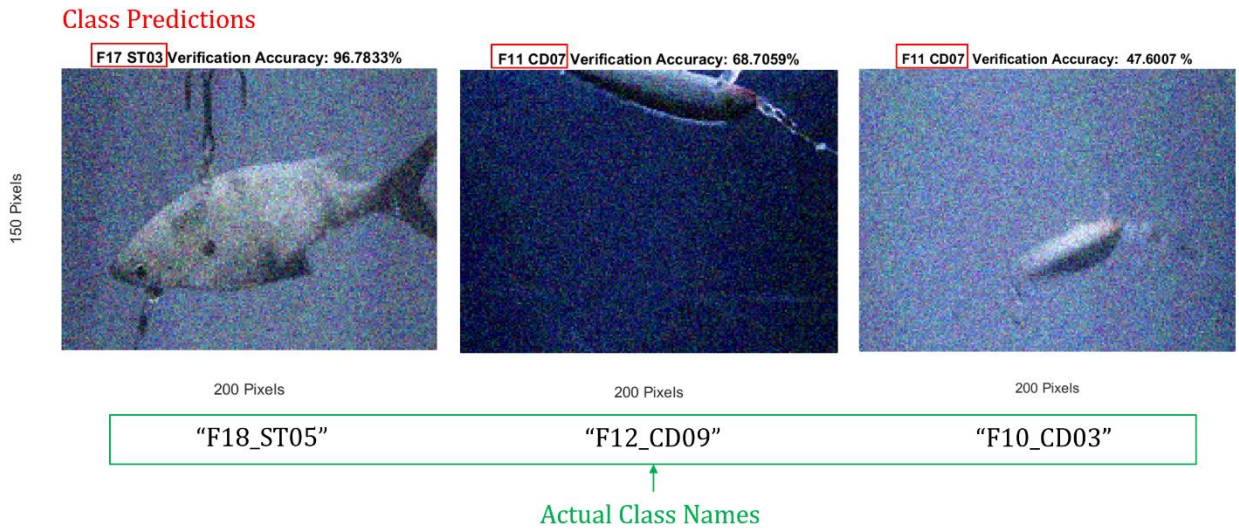

Figure 38 - Testing Three Unknown Classes

Figure 38 displays the inferencing performance of the trained model on the three classes that were left out to test the network's accuracy when forced to inference on a class that is larger than one that existed in the training database. In other words, this test simulates the possibility of a larger species of fish entering the AISS than was known during training. The green box at the bottom of the figure displays the actual class name of these larger classes if they had existed in the batched database; whereas, the red box is the approximated class of the smaller copy of the lure that the network accessed during training.

Left Image:

The neural network was extraordinarily accurate and correct in classifying this lure as an ST03. This image was selected for this test because it included an up-down affine transformation, Gaussian noise, and the imaging environment contains poor water clarity. The team considers this prediction to be successful.

Middle Image:

The network was accurate enough to classify this image correctly as a CD07. This image was included for this test because it displays the model's performance on a picture where the subject is partially in view and all image processing operations were applied. The team considers this prediction to be accurate regarding the difficulty of the subject's position.

Right Image:

The neural network did not select the best class, which is the CD01 class. Although, the model was accurate enough to predict that this lure is a member of the CD family. The top 2 predictions for this picture were CD07 then CD01; therefore, if the goal was to sort out all members of the CD family, then the team would consider this inference a success. The image was selected for testing because the noise levels are so extreme that a human could not effectively classify, which CD lure is in the frame.

Overall, the model averaged 82% accuracy on all unknown classes and exceeded 95% inferencing accuracy on training data. The AI system successfully exceeded the design objective to inference subjects above 50%.

VII. Discussion

To be clear, a problem such as the Asian Carp in the Mississippi River Inter-basin, in the team's opinion, has been approached using ineffective methods. The issue cannot be fixed with a physical barrier, and complete elimination of an invasive aquatic species is not financially feasible. Thus, the problem boils down to a population control issue. Secondly, altering the DNA to control a species is not yet well understood and could be incredibly dangerous. Since the recent discovery of Asian Carp DNA found in the DNA of local fish populations in the Great Lakes, science suggests that Asian Carp are already in the Great Lakes [3]. Therefore, the AISS team believes that the epidemic is a population control problem and an approach utilizing machine learning, and image processing is the only plausible solution to solve the epidemic.

Research by the team into possible renewable based power systems for the aquatic identification and sorting system would suggest that a hydro-generator is the best solution for this project, but such systems are prone to harming marine life. Although, new technology by General Electric (GE) are coming into the marketplace with turbine solutions that are capable of preventing injury to wildlife. If a fish farm had access to a high rate flow of water, hydro-generators offer an excellent stationary source of power. Ultimately, the downside to hydrological systems is that turbines are not a mobile solution and require a specific operating environment; therefore, capturing energy from the sun with solar panel arrays offers the overall best solution for the AISS.

VIII. Future Direction

1. Sorting System
    a. Develop a low power sensing system to detect an aquatic object nearing the front of the AISS
    b. Develop a long-range communication system to alert FDNR officials nearby that the AISS needs to be serviced or have fish removed
2. Database
    a. Increase the size of the database
    b. Increase the number of classes in the database
    c. Capture images of the lures in the simulation tank under various lighting conditions
3. Imaging
    a. Utilize different cameras to capture data at the same time
    b. Use different imaging methods like heat or high-frequency image cameras
    c. Use the same camera for inferencing as will be used for capturing images for the database
4. Image Processing
    a. Design an image processing pipeline for counting the number of lures in an image
    b. Estimate lures dimensions and weight using image processing to help the AISS know when the system is full
5. Neural Network
    a. Test the accuracy of a neural network to identify lures only from their outline
    b. Implement the neural network on other embedded hardware like an iPhone or an ARM processor
    c. Experiment with neural networks utilizing FP8 for low latency applications
    d. Design a neural network that can inference data from the AISS using a multi-camera system to increase accuracy
    e. Expand the VHDL neural network code generation software to support larger models and more advanced layers
6. Power System
    a. Get the parts to implement the solar array and power controller for the AISS
    b. Test the designed power systems ability to support high-current draw scenarios from the hoisting motors retracting the underwater sorting system towards the surface
    c. Design a data monitoring system to analyze the systems battery life and power generation capabilities

IX. Project Management

A. *Division of Labor*

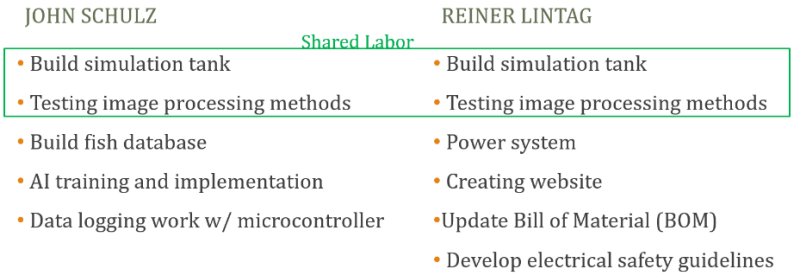

Figure 39 - Division of Labor

Figure 39 shows the division of labor shared between Reiner Lintag and John Schulz.

B. *Project Schedule*

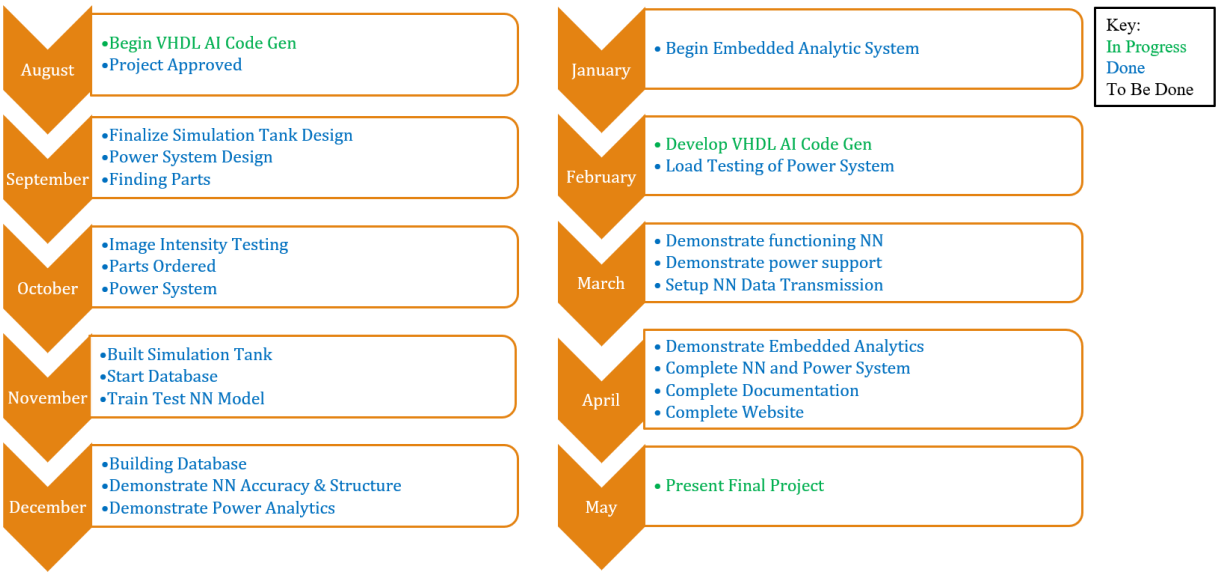

Figure 40 - Project Schedule

Figure 40 contains a high-level schedule followed during this senior capstone project. Appendix I includes a detailed list of milestones.

X. Conclusion

The first year of the Aquatic Identification and Sorting System (AISS) Senior Capstone project brought together a diverse array of electrical and computer engineering backgrounds from neural networks to power electronics to design the system presented in this report. Engineering concentrations within the project are concluded under the following areas: sorting system, simulation container, imaging, image processing, database creation, neural networks, and power electronics.

Sorting System

The sorting system in its entirety was not able to be fully implemented, due to time constraints. However, critical components in the system were completed like the neural network, ATmega128a servo control logic, and the image processing architecture. Also, the team determined that solar power and a battery support system was the best renewable power source for the AISS.

Simulation Container

The simulation container built by the team was successful in creating a reproducible environment for data collection, power system testing, and checking the neural networks inferencing performance; however, two main improvements could be made to the container. First, the team would mount the camera in a fixed location inside the tank to remove external glare, which would allow for increased filming consistency and inferencing performance. The second improvement would be to use a multicolor lighting solution to increase data augmentation, which would help the neural network perform better in different external lighting conditions.

These conclusions are also recommendations because the team struggled with getting the neural network to accurately perform on lures inside the simulation container with no blue hue from the water. Additionally, the distance of lens to the subject inside the tank, and the unavoidable presence of outside glare affected inferencing accuracy when the container was tested in the University Jobst Hall and not at John's house where the data collection took place.

Imaging

The approach taken by the team to recording video at 720p @ 60 frames-per-second proved to be successful in producing an extensive database of pictures that captured the lures without any noticeable motion blur. The imaging technique of recording only at night allowed for the precise imaging of the data,

but in real-world testing, this method resulted in a database that was not diverse enough in color temperatures. If the images did not have a heavily blue tint from the simulations containers blue lighting, the neural network would fail to accurately classifying the 17 trained lures. The concluded that the same camera used for database creation should be used for inferencing. Lastly, avoid the enabling of the camera's automatic white balance in all imaging systems is vital during data collection and inferencing because this image processing feature is difficult to adjust in the neural networks post-processing pipeline.

Image Processing

The image processing pipeline used to compress the number of RGB colors sent to the neural network proved to be a successful and innovative solution to compressing imaging data. Inferencing accuracy on a GPU did not appear to be affected by 4-bit imagery, while latency did decrease. The team concludes that the approach of using 4-bit color-accurate imagery should also reduce inferencing latency in embedded neural network hardware, like on an FPGA or ARM processor.

Database Creation

The teams approach to organizing and maintaining an extensive database using a defined database file hierarchy and the naming system allowed for regular updates to the database to occur with ease. Therefore, the restructuring of the approximately half terabyte database did not occur with the periodical acquisition of new data. Secondly, the use of Matlab's Parallel Computing Toolbox allowed for the addition and processing of new videos into the database to be ready in the training, testing, and verification folders by the neural network within minutes instead of hours.

Neural Networks

The embedded neural network computed on a GPU exceeded the team's accuracy objective of over 50% when tested on the three completely unknown classes. The three unknown lures in the database averaged 82% accuracy. The team's data proves that small and light-weight neural networks can perform well in the areas of latency, accuracy, and inferencing on low color intensity images. Testing done on small dense layer networks on FPGAs were successful in approaching different techniques and mathematical methods to optimize a model's layers and weights around inferencing predictions in different fixed-point accuracies.

Power Electronics

Although the team was unable to access some requested power electronic parts, research conducted supports solar power as the best renewable energy source for the AISS to operate off-grid for extended deployments.

Overall, the process of designing and engineering the AISS project will have a broad impact on the success and survival of native species and will be paramount in its effect on the aquaculture industry. Further development and research on the subject of this project will help Bradley University be at the forefront of solving a local and global problem facing the long-term sustainability of aquatic populations.

Lastly, the team would like to thank the valuable support from the Bradley University Department of Electrical Engineering faculty and staff for aiding in the research conducted during the course of the aquatic identification and sorting system project. Special thanks and appreciation goes to our professor and project advisor Dr. Mohammad Imtiaz for providing helpful guidance throughout the course of the team's senior capstone project.

Appendix I – Milestones

- 8/30/2018 – Senior capstone officially approved by faculty
- 9/12/2018 – First meeting with our advisor, Dr. Imtiaz
- 9/23/2018 – Finalize design and hardware components
- 10/5/2018 – Start testing image intensity levels for neural network
- 10/11/2018 – Submit order form for parts
- 10/19/2018 – Received simulation tank parts
- 10/22/2018 – Finish image intensity testing
- 10/27/2018 – Begin building image databases for the neural network.
- 10/29/2018 – Submit a rough outline of neural network structure
- 11/09/2018 – Finalize stats of image database
- 12/4/2018 – Demonstrate neural network: accuracy & structure
- 12/4/2018 – Demonstrate system power usage
- Winter Break
- 1/30/2019 – Neural network on an FPGA
- 2/05/2019 – Transition neural network to GPU instead of FPGA
- 3/13/2019 – Received DC IOT servo motors
- 3/18/2019 – Safety Management Guidelines and Emergency Contingency Plan
- Spring Break
- 4/01/2019 – Demonstrate UART controlled motor actuation system
- 4/04/2019 – Senior project draft submission
- 4/10/2019 – Demonstrate the power delivery system
- 4/11/2019 – Senior project conference abstract submission
- 4/16/2019 – Expo poster judgement
- 4/17/2019 – Expo poster award ceremony
- 4/18/2019 – IAB poster
- 4/30/2019 – Final presentation practice
- 5/04/2019 – Final presentation
- 5/06/2019 – Finalize deliverable on the project's web page
- 5/07/2019 – Submit all deliverables

Appendix II – Parts and Materials

- Tank – Cambro 12”x18”x9” Clear Food Grade Storage Box
- Lid – Cambro Multipurpose 12”x18” Clear Lid
- Pump – Jebaco DCP-2500, Max: 23 [W]
- LED Light Bar – Mingdak Max: 4 [W]
- Lures – Rapala, Bass Pro Shop, The Worm, and other manufacturers. Total Quantity: 20
- FPGA – Avnet Zedboard with a Xilinx ZYNQ 7000 chipset
- Camera – Canon 70D DSLR with 21 M-Pixel Sensor
- Lens – 35-75 mm with STM (silent-stepper motor) and OS (optical stabilization)
- Background Cladding – Hefty Black Trash Bags
- Servo Motors – Analog Servo by HuiDa RC International, INC.

Appendix III – Lure Datasheet

Table 1 - Lure Datasheet

Figures	Specifications	Database
	NN ID: F1_XRAP Manufacturer: Rapala Family: X-RAP Name: X-RAP Xtreme Action Slashbait Model No.: XR08 PG Color: Purple Gold Size: L:89 mm, W:13 mm, H:15 mm Pipe Out: 14”, Horizontal Mount	Recorded Time: 24 min, 09 sec No. Split: Clear 11607 PNGs Cloudy L1 2905 PNGs Cloudy L2 3012 PNGs Total: 17524
	NN ID: F2_FROG Manufacturer: LiveTarget Family: N/A Name: Hollow Body Model No.: Unknown Color: Green Yellow Size: L:125 mm, W:29 mm, H:22 mm Pipe Out: 17”, Horizontal Mount	Recorded Time: 24 min, 39 sec No. Split: Clear 11751 PNGs Cloudy L1 3003 PNGs Cloudy L2 3000 PNGs Total: 17754 PNGs
	NN ID: F3_MICE Manufacturer: Heddon Meadows Family: N/A Name: Vintage Mouse Model No. N/A Color: Black Size: L:111 mm, W: 22 mm, H: 21 mm Pipe Out: 17”, Horizontal Mount	Recorded Time: 28 min, 56 sec No. Split: Clear 15087 PNGs Cloudy L1 3008 PNGs Cloudy L2 2940 PNGs Total: 21035 PNGs
	NN ID: F4_XRAP Manufacturer: Rapala Family: X-RAP Name: X-RAP Xtreme Action Slashbait Model No.: XR08 S Color: Silver Size: L:89 mm, W:13 mm, H:15 mm Pipe Out: 14”, Horizontal Mount	Recorded Time: 26 min, 01 sec No. Split: Clear 12927 PNGs Cloudy L1 2904 PNGs Cloudy L2 3030 PNGs Total: 18861 PNGs

	NN ID: F5_XDEP Manufacturer: Rapala Family: X-RAP Name: X-RAP Deep Diving Slashbait Model No. XRD08 PG Color: Purple Gold Size: L:107 mm, W:13 mm, H:15 mm Pipe Out: 14”, Horizontal Mount	Recorded Time: 26 min, 21 sec No. Split: Clear 13200 PNGs Cloudy L1 2892 PNGs Cloudy L2 3101 PNGs Total: 19193 PNGs
	NN ID: F6_FINE Manufacturer: Rapala Family: Finesse Name: Ultra Light Minnow Finesse Sinking Model No.: ULM06 CH Color: Chrome Size: L:67 mm, W:10 mm, H:13 mm Pipe Out: 17”, Angle Mount	Recorded Time: 24 min, 52 sec No. Split: Clear 10416 PNGs Cloudy L1 3660 PNGs Cloudy L2 3828 PNGs Total: 17904 PNGs
	NN ID: F7_MINO Manufacturer: Family: Unknown Name: Unknown Model No.: Unknown Color: Chrome Size: L:63 mm, W:7 mm, H:9 mm Pipe Out: 14”, Horizontal Mount	Recorded Time: 25 min, 10 sec No. Split: Clear 11544 PNGs Cloudy L1 2904 PNGs Cloudy L2 3480 PNGs Total: 17928 PNGs
	NN ID: F8_BPSX Manufacturer: Bass Pro Shop Family: XPS Name: XPS Pro Series Shad Model No.: DXBPM35-08 Color: Silver Size: L:123mm, W: 13mm, H:17 mm Pipe Out: 14”, Horizontal Mount	Recorded Time: 21 min, 16 sec No. Split: Clear 10848 PNGs Cloudy L1 2892 PNGs Cloudy L2 3504 PNGs Total: 17244 PNGs

	NN ID: F9_CD01 Manufacturer: Rapala Family: Countdown Name: Countdown Sinking Model No.: CD01 S Color: Silver Size: L:33 mm, W:9 mm, H:10 mm Pipe Out: 17", Angle Mount	Recorded Time: 22 min, 32 sec No. Split: Clear 10428 PNGs Cloudy L1 2916 PNGs Cloudy L2 2880 PNGs Total: 16224 PNGs
	NN ID: F10_CD03 Manufacturer: Rapala Family: Countdown Name: Countdown Sinking Model No.: CD03 S Color: Silver Size: L:44 mm, W:9mm, H:12 mm Pipe Out: 17", Angle Mount	Recorded Time: 23 min, 23 sec No. Split: Clear 10500 PNGs Cloudy L1 3432 PNGs Cloudy L2 2904 PNGs Total: 16836 PNGs
	NN ID: F11_CD07 Manufacturer: Rapala Family: Countdown Name: Countdown Sinking Model No: CD07 S Color: Silver Size: L:73 mm, W:13 mm, H:17 mm Pipe Out: 17", Angle Mount	Recorded Time: 23 min, 01 sec No. Split: Clear 10368 PNGs Cloudy L1 3299 PNGs Cloudy L2 2905 PNGs Total: 16560 PNGs
	NN ID: F12_CD09 Manufacturer: Rapala Family: Countdown Name: Countdown Sinking Model No.: CD09 S Color: Silver Size: L:91 mm, W:14 mm, H:19 mm Pipe Out: 17", Angle Mount	Recorded Time: No. Split: Clear 11532 PNGs Cloudy L1 2952 PNGs Cloudy L2 3108 PNGs Total: 17592 PNGs

	NN ID: F13_JERK Manufacturer: Rapala Family: Jerkbait Name: Shadow Rap Jerkbait Model No.: SDR11 S Color: Silver Size: L: 112 mm, W:12 mm, H:18 mm Pipe Out: 14", Horizontal Mount	Recorded Time: 28 min, 03 sec No. Split: Clear 14292 PNGs Cloudy L1 2892 PNGs Cloudy L2 3006 PNGs Total: 20190 PNGs
	NN ID: F14_FLOT Manufacturer: Rapala Family: Original Floating Name: Original Floating Live Rainbow Trout Model No.: F09 RTL Color: RTL Size: L:92 mm, W:11 mm, H:15 mm Pipe Out: 14", Horizontal Mount	Recorded Time: 25 min, 24 sec No. Split: Clear 12060 PNGs Cloudy L1 3108 PNGs Cloudy L2 3120 PNGs Total: 18288 PNGs
	NN ID: F15_WORM Manufacturer: The Worm Factory Family: The Worm Name: The Worm Scented 6" Model No.: 274/24 Color: Brown Size: L:152 mm, W:8 mm, H:8 mm Pipe Out: 17", Angle Mount	Recorded Time: 23 min, 11 sec No. Split: Clear 10632 PNGs Cloudy L1 3060 PNGs Cloudy L2 2988 PNGs Total: 16692 PNGs
	NN ID: F16_HUSK Manufacturer: Rapala Family: Huskey Name: Husky Jerk Firetiger Model No.: HJ08 FT Color: Firetiger Size: L:81 mm, W:11 mm, H:15 mm Pipe Out: 14", Horizontal Mount	Recorded Time: 24 min, 32 sec No. Split: Clear 11496 PNGs Cloudy L1 2904 PNGs Cloudy L2 3264 PNGs Total: 17664 PNGs

	NN ID: F17_ST03 Manufacturer: Storm Family: Live Name: Live Kickin' 03 Shad Model No.: LKSD03SD Color: Silver Size: L:85 mm, W:11 mm, H:24 mm Pipe Out: 14", Top Mount	Recorded Time: 24 min, 41 sec No. Split: Clear 11544 PNGs Cloudy L1 2916 PNGs Cloudy L2 3312 PNGs Total: 17772 PNGs
	NN ID: F18_ST05 Manufacturer: Storm Family: Live Name: Live Kickin' 05 Shad Model No. LKSD05SD Color: Silver Size: L:130 mm, W:15 mm, H:36 mm Pipe Out: 14", Top Mount	Recorded Time: 22 min, 51 sec No. Split: Clear 10668 PNGs Cloudy L1 2904 PNGs Cloudy L2 2880 PNGs Total: 16452 PNGs
	NN ID: F19_FROG Manufacturer: Strike King Family: Rage Tail Name: Rage Tail Toad Soft Bait Model No.: Unknown Color: Green Red Size: L:100 mm, W:23 mm, H:15 mm Pipe Out: 17", Angle Mount	Recorded Time: 23 min, 57 sec No. Split: Clear 11028 PNGs Cloudy L1 2976 PNGs Cloudy L2 3240 PNGs Total: 17244 PNGs
	NN ID: F20_BROK Manufacturer: Rapala Family: Unknown Model No.: Unknown Color: Silver Size: L:122 mm, W:14 mm, H:17 mm Pipe Out: 14", Horizontal Mount	Recorded Time: 25 min, 12 sec No. Split: Clear 11772 PNGs Cloudy L1 2892 PNGs Cloudy L2 3480 PNGs Total: 18144 PNGs

Appendix IV – Database Layout

Figure 41 contains a high-level overview of the naming structure given to images and movie files in the database. The relationship within the D:\ drive between the images and the Matlab and Keras training scripts provides a visual display of where the data exists when being loaded into the neural network.

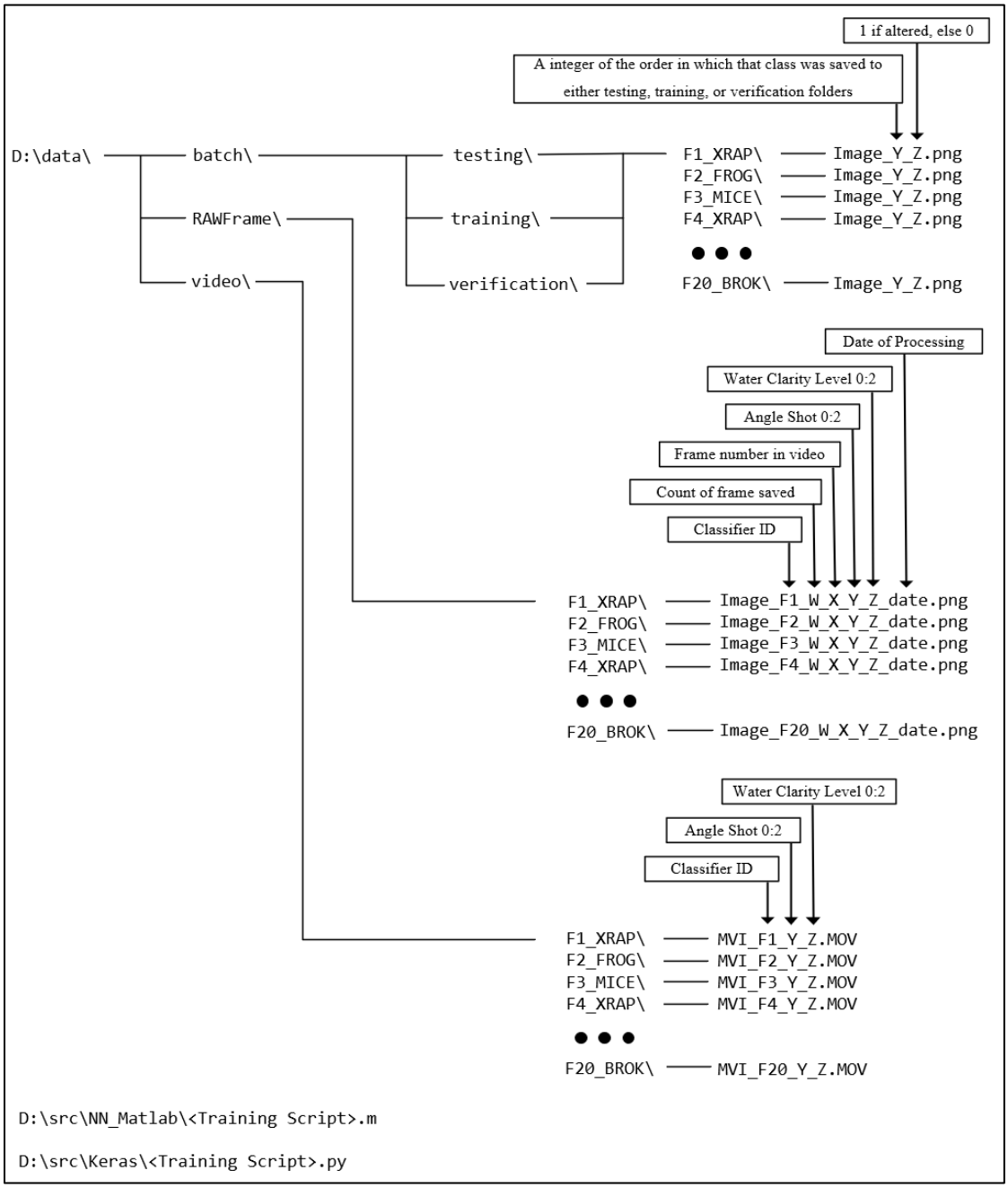

Figure 41 - Hierarchy Image and Training Script File Paths

Appendix V - Safety Management System Guidelines

C. Introduction

Instruction set will be applied to every team that takes the Aquatic Identification and Sorting System Project.

D. Objective

The objective of the Safety Management System is to provide a mechanism to measure and track the safety and performance of all team members working on the project.

E. Emergency Contact

Electrical Engineering Department
Bradley University,
1501 W Bradley Ave.
Peoria, IL 61625

F. Procedure

While working with lines that will be connected to the power grid, users must follow these rules before operating on a live system.

1. Check all equipment cords to see if any insulation is open. If equipment shows damage, please immediately stop and retrieve a replacement.
2. Wipe and dry the entire workspace area before setting up equipment.
3. Use a Ground Fault Circuit Interrupt (GFCI) extension cord adapter on all lines to protect the user from electrocution from an imbalanced incoming and outgoing current.
4. While the project holds water, both members will need to indicate where the splash zones are and move all electronics away from those zones to prevent damage and hazards.

5. Fires can occur while working, have a fire extinguisher close by to suppress a possible fire from starting or spreading in the case of an electrical fire occurring.
6. Wear proper personal protective equipment (PPE) when working with power tools and assembling the system.
    a. Safety Glasses (Z87) rated
    b. Working gloves (for power tools)
    c. Rubber gloves (for working around live wires)

G. Responsibilities

The advisor and all team members should be responsible for looking after each others safety and well being. Project responsibilities consist of setting up a proper working space and tracking all safety goals regularly at team meetings, which is critical to accomplishing the working environment envisioned by this document. Reporting and investigating all incidents is crucial to overall team safety. Reviewing the actions of the individuals in question that resulted in an investigation should be completed promptly. Safety performance updates are to be provided to Dr. Imtiaz at each meeting following project work that entails the exposure to live wires.

Appendix VI – Emergency Contingency Plan

H. Introduction

The purpose of this plan is to ensure each member of the team is provided a safe working environment. The emergency contingency plan has been made to provide an organized procedure and course of action to prepare and respond to minor or major natural and human-caused emergencies that threaten Bradley University and its students.

I. Objective

The SMS text message will be sent to all members and faculty to address an emergency that has affected the team.

A. Notification to Parents
B. Notification to the Electrical Engineering Department: Advisor and Chair
C. Notification to Bradley University Department of Human Resources

J. Procedure

Each member of the project will develop an emergency plan applicable to their needs. Emergencies will be assessed by an individual that is near the accident or event. Any time an emergency plan is implemented, the incident should be documented and include: date, event description, actions taken, and parts of the plan that were successfully implemented.

K. Definitions

A. **Emergency**. Any incident, human-caused or natural that requires responsive action to protect life and property.
B. **Event**. A planned non-emergency activity.
C. **Natural Emergencies**. Major fires, hurricanes, earthquakes, tornados, snow, and other severe weather.
D. **Human-Caused Emergencies**. Hazardous chemical releases, civil disorder, or riots.

L. Responsibilities

Advisor and team members must watch and provide help for an injured member. Faculty and members must know emergency numbers and emergency center locations. Reporting potential emergency situations to their perspective advisors.

References

[1] World Bank, "FISH TO 2030 Prospects for Fisheries and Aquaculture," Rep. no. 83177-GLB, Dec. 2013. [Online]. Available: http://www.fao.org/docrep/019/i3640e/i3640e.pdf, Accessed on: April 23, 2018.

[2] News Desk, "Keeping Asian carp out of the Great Lakes will cost billions and take decades," *PBS NewserHour Productions LLC.*, Jan. 6, 2014. [Online]. Available: https://www.pbs.org/newshour/science/keeping-asian-carp-out-of-the-great-lakes-will-cost-billions-and-take-decades, Accessed on: April 23, 2018.

[3] Vice News, "The Worst Fish in America: Asian Capocalypse," Sep 24, 2014. [Online]. Available: https://www.youtube.com/watch?v=YnZp1jtOhR0, Accessed on: April 23, 2018.

[4] S. Raje, "AI Acceleration," *presented at Xilinx XDF Silicon Valley 2018 Conf.*, Xilinx Inc., Oct. 10, 2018, San Jose, CA. [Online]. Available: https://www.youtube.com/watch?v=qWFREOeRiqo, Accessed on: Oct. 25, 2018.

[5] H. Gao, "A Walk-Through of AlexNet," Medium Corp., Aug. 7, 2017. [Online]. Available: https://medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637, Accessed on: Oct. 25, 2018.

[6] Y. Ayster, "Lecture: Deep Learning for Vision," *MIT Computer Vision* (Course 6.869: Advances in Computer Vision). Cambridge, MA., USA. [Online]. Available: http://6.869.csail.mit.edu/fa16/lecture/6.869-Lecture19-DeepLearning.pdf, Accessed on: Oct. 29, 2018.

[7] Power HD 3001HB Analog Servo, Ver. 1. HuiDa RC International Inc., Huizhou, Guangdong, China. [Online]. Available: https://www.pololu.com/file/0J728/HD-3001HB.pdf, Accessed on: Apr. 2, 2019.

[8] ATmega128a – Complete Datasheet, Rev. 815J. Atmel Corp., San Jose, California, USA, Sep. 2015, pp. 85-249. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8151-8-bit-AVR-ATmega128A_Datasheet.pdf, Accessed on: Apr. 2, 2019.

[9] K. Willis, "Using AI to Help Manage Canada's Invasive Species," *Univ. of Alberta*, Alberta, Canada, May 8, 2018. [Online]. Available: https://www.ualberta.ca/science/science-news/2018/may/ai-to-manage-canadas-invasive-species, Accessed on: Apr. 4, 2019.

[10] D. Kling, “Economist Develop Decision-Making Method for Lionfish Management,” *Oregon State University*, OSU Extension Catalog, Corvallis, Oregon, USA, Nov. 2017. [Online]. Available: https://extension.oregonstate.edu/news/economists-develop-decision-making-method-lionfish-management, Accessed on: Apr. 4, 2019.

[11] B. Lei and J. Marsman, “Species Around the World are in Trouble, But We Can Help,” *Microsoft*, Apr. 18, 2018. [Online]. Available: https://blogs.microsoft.com/green/2018/04/18/earth-week-biodiversity/, Accessed on: Apr. 4, 2019.

[12] P. Wright, “Turning in Dead Lionfish to Florida Wildlife Officials Could Make You $5,000 Richer,” *The Weather* Company, TWC Product and Technology LLC., Georgia, USA, June 22, 2018. [Online]. Available: https://weather.com/science/nature/news/2018-06-22-florida-bounty-dead-lionfish, Accessed on: Apr. 4, 2019.

[13] K. Sato, C. Young and D. Patterson, “AI & Machine Learning: An In-depth Look at Google’s First Tensor Processing Unit (TPU),” *Google*, May 12, 2017. [Online]. Available: https://cloud.google.com/blog/products/gcp/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu, Accessed: Apr. 5, 2019.

[14] D. Clark, “Top 16 Open Source Deep Learning Libraries and Platforms,” *KDnuggets*, Apr. 2018. [Online]. Available: https://www.kdnuggets.com/2018/04/top-16-open-source-deep-learning-libraries.html, Accessed: Apr. 5, 2019.

[15] “About: ONNX,” *Facebook*. [Online]. Available: https://onnx.ai/about, Accessed: Apr. 5, 2019.

[16] M. Sandru, “Hydroelectric Generator: How to Build a Small One,” *The Green Optimistic*. [Online]. Available: https://www.greenoptimistic.com/hydroelectric-generator/#.XMpwwHdFzE5, Accessed on: May 1, 2019.

[17] “HP-200,” *powkey*. [Online]. Available: http://www.ledpowkey.com/content/?235.html, Accessed on: May 1, 2019.