Distributed Wild Bird Surveillance and Recognition System

Final Report

Team Number: SDMAY19-10
Advisor and Client: Dr. Joseph Zambreno

Pierce Adajar - Repository Wrangler and Machine Learning Lead
Francisco Arreola - Meeting Scribe and Infrastructure Lead
Claudia Athens - Team Communications Lead and Systems Integration Lead
Ben Simon - Meeting Facilitator and User Interaction Lead

Team Email: sdmay19-10@iastate.edu
Team Website: http://sdmay19-10.sd.ece.iastate.edu
Revised: April 30, 2019

Table of Contents

Acknowledgements

SDMAY19-10 would like to thank the Department of Electrical and Computer Engineering for their continued support of student's professional and technical growth through the administration of this course. SDMAY19-10 would also like to thank our client and advisor, Dr. Joseph Zambreno, for the financial resources provided and his continued support and guidance of this project.

I. Project Statement

1. Problem Statement

Bird identification is a more complicated topic than one would think. As an example, can you identify the species of the four following birds?

Fig. 1: Four North American Birds[1]

In the clockwise order from the top left, the birds are a male cardinal, female cardinal, another female cardinal, and a juvenile cardinal. Four seemingly different birds, but they all happen to be the same species. Now imagine these differences happening for all the birds across North America. Because of this, even for a hobbyist, bird identification can be challenging.

The task at hand to provide a solution to this problem is to create a self-contained system that can detect and classify bird species in realtime in our client's backyard. The current consumer products available are limited to nature cameras that are constrained by basic video streaming and photo capture capabilities that are not able to classify birds.

To do this, we will use an embedded GPU platform with a convolutional neural network for real time object detection and classification. Our model will be trained and tested with two already available North American bird datasets: NABirds V1 and Caltech-UCSD Birds 200. As a result of this project, we will have delivered a well-documented hardware/software product to our client that will successfully identify North American birds in real time.

2. Operational Environment

The primary operational environment will be in our client's backyard in Iowa. Because of this, it will have to withstand Iowa's normal weather year-round. It will be exposed to dusty, windy, and wet conditions with a large temperature range. Our design includes a weatherproof enclosure to prevent the damage of our project.

3. Intended Users and Uses

The primary user of this product will be our client, Dr. Joseph Zambreno. The intended end use of this product will mostly be between the client and our web interface which will allow the client to view the video stream, the captured photos, and the birds classified. Little to no interaction between the client and the actual product will be necessary unless he would like to change the position of the product within his backyard.

4. Assumptions and Limitations

4.1 Assumptions

- The operating environment will not be in extremely hazardous weather or conditions.
    - Normal weather conditions include temperatures ranging from 40^{o}-80^{o} F, and precipitation not exceeding 10 millimeters an hour.
- The end product will only be used in the client's backyard while stationary.

- The client will have access to the Internet in order for the product to access the web interface.
- The system will have access to the client's Wi-Fi and power.
- Only one user will view the video steam at a time.
- The system will only need to classify birds commonly found in North America.
- The system will only need to detect and classify birds in well-lit conditions.

4.2 Limitations

- The datasets we will be training our model on are professional-level image quality.
- The system is limited to the computing power of the NVIDIA Jetson TX2.
- The case of the system must be small enough to be attached on a deck railing.
- The system is constrained by the client's bandwidth.
- The project will have a budget of $1500.

II. Project Deliverables and Specifications

The four core deliverables for the end product are the hardware system, the detection and classification system, the data streaming/storage system, and the frontend and user notification system.

1. Deliverables and Functional Requirements

1.1 Hardware

The hardware will facilitate data streaming and bird detection in real time. This system will consist of an embedded processing board, a 4k camera, and internet connection. The 4k camera will stream real time video from a bird feeder located in the client's yard. The system will be robust enough to withstand normal Iowa weather and provide a stable camera mount for a clear video stream.

1.2 Detection and Classification

The detection and classification systems will be running on the embedded processing board. The detection and classification will run in real time and be able to identify all of Iowa's native bird species.

1.3 Data Streaming and Storage

Once the detection and classification system has identified a bird, the hardware system will send an image of the bird to the cloud storage system. These stored images will be able to be served to a web client on request. In addition, the system will be able to stream live video from the bird feeder. The live video feed will be served at a 1080p quality.

1.4 User Notifications and Frontend

The final product will deliver a frontend web client and notification system. The frontend client will allow our client to view previously identified birds as images and receive a notification upon new identification. It will also provide a live video stream of the bird feeder.

2. Non-Functional Requirements

The non-functional requirements revolve around the ease of use of the system for the client. The client should have no issues setting up the camera and moving the system's enclosure. The client should be able to seamlessly use the web interface provided to

watch the streamed video, browse the captured photos and videos, and configure the notification system. In addition, it should also be under the project's budget of $1500.

III. Previous Work / Literature Review

1. Existing Solutions

Currently, there are no commercial solution that accomplish the task at hand. There are a variety of wildlife cameras available on the market but, most existing solutions only take pictures when it detects motion in front of the camera. They do not attempt to detect actual objects in the frame, so they will be prone to false positives. Additionally, they do not attempt to perform any sort of classification for the objects in the picture.

We were able to find a similar hobbyist project by Kirk Kaiser[2]. On his blog, Kaiser describes a bird detection and classification solution based around an Nvidia Jetson TX2 board. Kaiser runs machine learning models for object detection on the TX2, and whenever the models detect a bird, the camera saves 240 frames from the camera to an attached SSD. Where Kaiser's solution lacks is classification - the models Kaiser is using are very broad, and will only apply the label "bird" to any birds in the image. We are looking for a solution to not only detect birds in an image, but also apply a label based on its species. Additionally, we aim to move storage requirements to a more accessible medium such as cloud based storage, rather than an onboard SSD.

2. Relevant Literature

There have been other attempts to use deep neural network architectures to identify animals in images from wildlife cameras. We found one instance where Norouzzadeh et al.[3] used a multistage pipeline to detect and classify 48 species in the snapshot Serengeti dataset. The researchers were able to achieve a very high detection and classification accuracy, above 99% top 5, but it should be noted they used a very deep architecture consisting of 152 layers. We believe this architecture to be far too deep for our embedded application, but can use other ideas from them. We can use their idea of a two stage pipeline, one stage for detection and another for classification. This enables us to use a more expensive classification model, and just have it run less often.

The most relevant literature to our project is based around the classification solutions. Recently, there have been many breakthroughs in the field of image processing and classification. We researched a variety of architectures such as SqueezeNet[4] and AlexNet[5], and ultimately went with a SqueezeNet-like architecture due to its simplicity and focus on performance in embedded applications.

IV. System Design

The system is designed to be a self-contained embedded platform that is equipped with several key features: online connectivity, high quality optics, and an onboard recognition system. A sample block diagram of the system can be seen below.

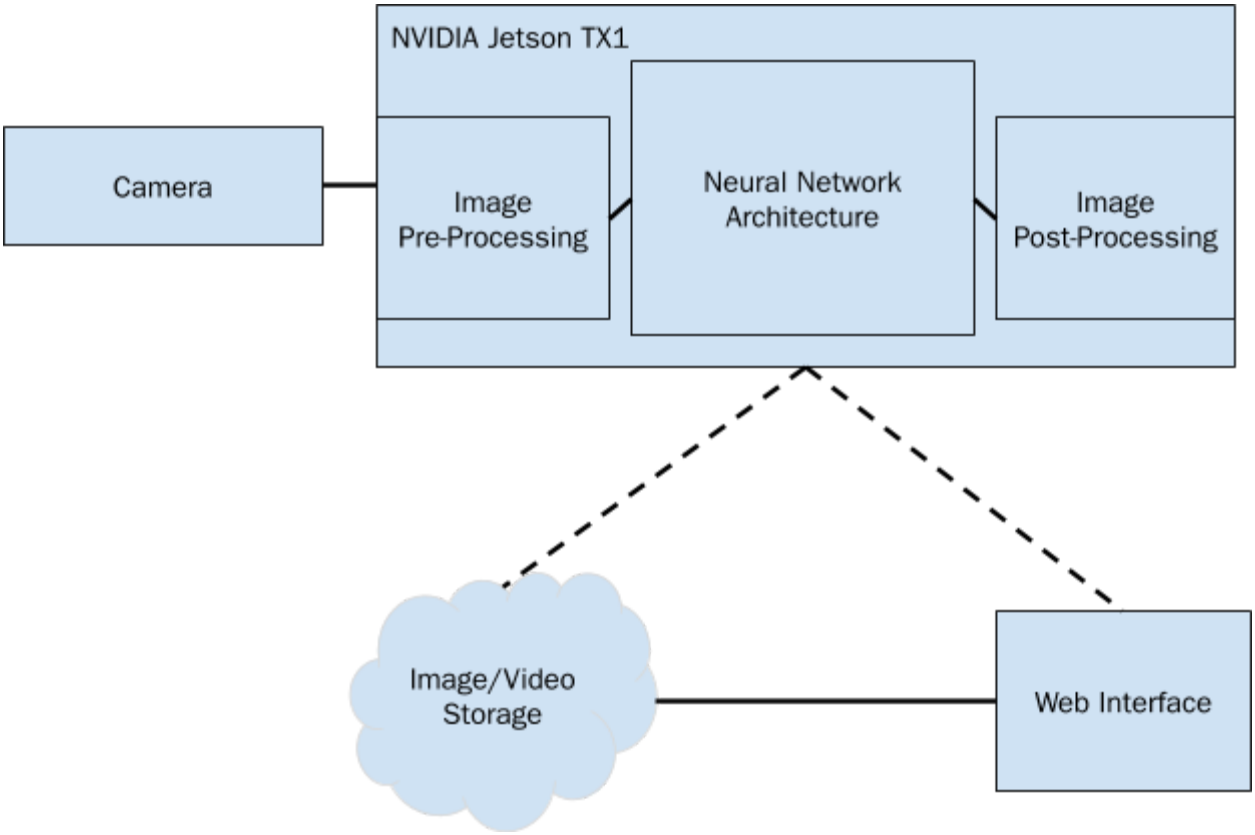

Fig. 2: System Block Diagram

1. Hardware/Base Station

The base station facilitates real-time bird detection and video streaming in our clients' backyard. The system needs to be powerful enough to run the detection and classification systems but also small enough to fit in a weather-resistant enclosure. To accomplish this, the hardware shall consist of a small form-factor embedded processing board equipped with a Graphics Processing Unit (GPU). The enclosure conforms to the IP55 standard[6] to ensure protection from harsh Iowan weather. Lastly, the board needs to transmit birds and the detected birds and live stream to the web interface. To this

end, the board shall be equipped with high-speed wireless communication capabilities and an Internet connection.

2. Detection and Classification

Detection and classification is the subsystem that encompasses both the decision process of which frames to consider for classification, as well as our machine learning model for classifying images. A constraint of this system is the high computational cost of machine learning models. To facilitate real time processing, a less expensive motion detection algorithm selects which frames to classify. This prevents unnecessary classification on frames without birds, as well as providing a simple way to downscale frames for faster processing.

3. Data Streaming and Storage

We leverage cloud technologies to run our storage system. Making use of cloud providers removed a lot of development overhead related to data maintenance and distribution, and allocated more team time on the more novel portions of this project. For video streaming, the project uses our own server to distribution of video. This was decided due to the large cost associated with streaming data on cloud platforms. Additionally, our scaling needs for video streaming are minimal, reducing our need for a self-scaling solution in the cloud.

4. User Notifications and Frontend

This system covers the client’s direct interaction with our system. It consists of a website that displays the gathered data in a visually pleasing manner. This website enables our end user to receive notifications on bird detection events and allow the user to view the video stream from the base station.

V. System Implementation

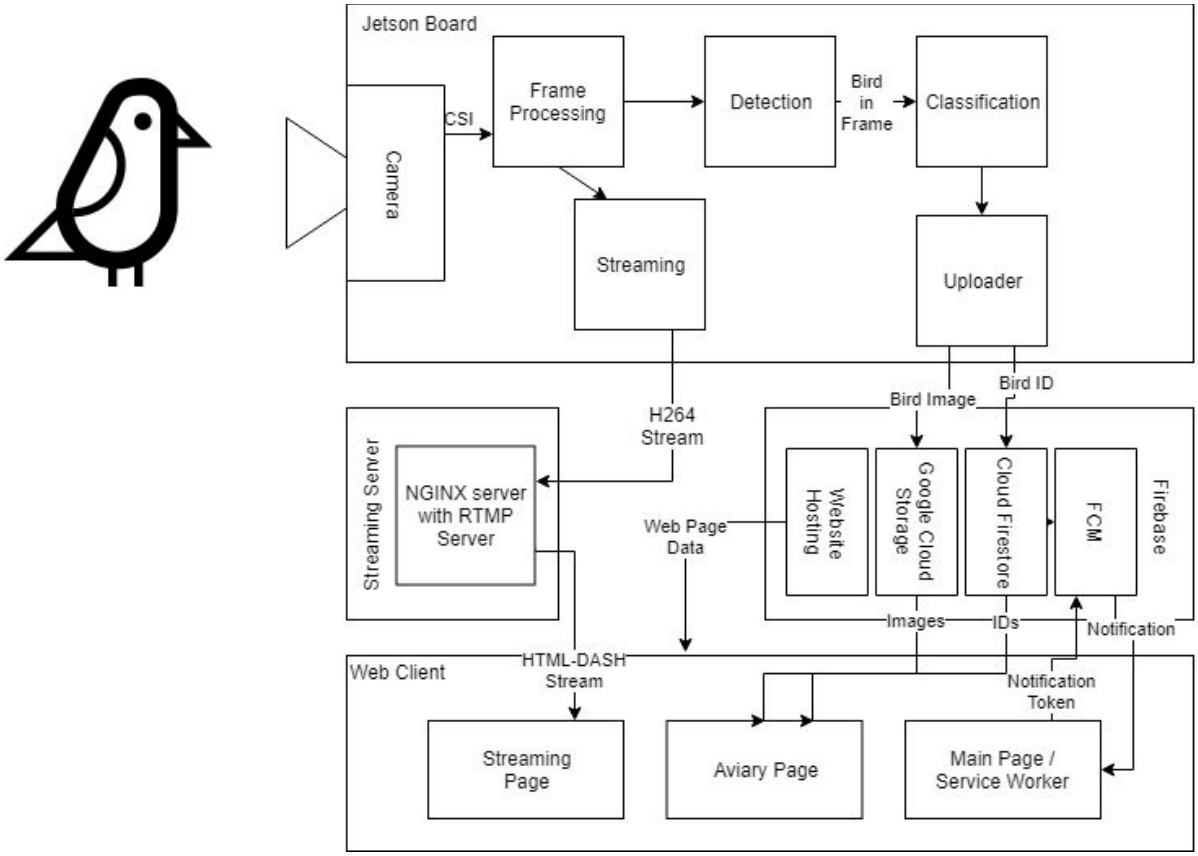

Fig. 3: System Architecture

The system begins with a 4K image capture or stream from the e-CAM131_CUTX2 from E-Con Systems. The camera interfaces with the NVIDIA Platform through the Camera Serial Interface (CSI). Once the frame is on the board, we apply some image pre-processing to reduce the image size for more efficient computing. Then, the image is fed through our neural network architecture. From the board, these frames and classifications are sent over the client's home network to the cloud for storage. Additionally, the board transmits a constant video stream to a secondary server. This server handles processing the stream into a web deliverable video stream. Finally, our user interface handles notifying the end user of bird detections and allows the user to view the collected data.

1. Hardware

The hardware system begins with the NVIDIA Jetson TX2. We will leverage the board's high performance embedded GPU to run our neural network.

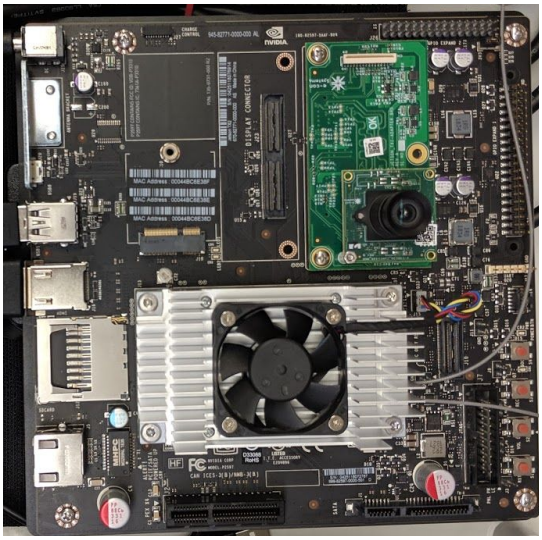

Fig. 4: NVIDIA Jetson TX2 Developer Board

For the camera, E-Con Systems provides a Linux camera driver (V4L2) for the 13.0 MP MIPI CSI-2 camera. This will allow us to interface the camera from the board. The board is also compatible with GStreamer-1.0 for video recording and network streaming to the web client.

2. Detection and Classification

We make use of a two-step system which splits the detection and classification processes into discrete actions. Our detection is done using a background subtraction process which detects motion. The algorithm used is a CUDA-accelerated Gaussian mixture-based algorithm. By using a background subtractor, we can also extract bounding box data which will enable us to reduce the scope of information passed to our classification process and ultimately speed up processing.

Our classification process will use a convolutional neural network to identify the species of the birds in the frame. This neural network is built, trained, and evaluated using TensorFlow. The architecture of our neural network is based on the architecture of SqueezeNet[4], making extensive use of smaller convolutions as a form of dimensionality reduction. We have modified the architecture to use LeakyReLU[7] instead of the

described ReLU[8] to minimize vanishing gradients. To initialize the network parameters, we will use the Xavier initialization strategy[9], and to calculate the loss, we will use the softmax cross-entropy function. This is because our task is to label images with a single truth value, and we already have a dataset of pre-labeled images. Once the classification is done, this subsystem will trigger the uploading of combined classification and image data.

3. Data Streaming and Storage

Our storage solution is based in the cloud. We chose to use Google's Firebase product suite for all of our cloud infrastructure. Firebase was a simple solution to work with and covered all of our needs, from image and classification storage to website hosting. Additionally, Firebase includes a cloud messaging feature that powers our notification delivery. We make use of Cloud Functions to handle our server side logic. A serverless model for our backend better fits our event driven system and reduces the overall cost of our cloud system.

For video streaming we make use of an NginX server hosted at Iowa State University. This server receives the H264 video stream from the jetson and converts it to a MPEG DASH stream. This enables us to use existing Javascript libraries to display our video. Due to the high volume of data in a 1080p videostream, we moved this process from a cloud virtual machine, to a local solution.

4. User Notifications and Frontend

The user frontend will be a standard html website. Our cloud system will notify the frontend when new data has been uploaded to our storage system and will subsequently query and display the new data. Our website shows the stored information inside of individual panels, which hold the image along with stored classification data. The delivery of updates is handled by Firebase Cloud Messaging. Our cloud backend will publish bird detection events, and from there a background service worker on the frontend will receive these updates and display a web push notification to the user.

5. Interfaces

The first interface is between the camera and the processing board. The camera uses the CSI port on the Jetson. This allows the Jetson to capture both high-quality images and video feeds.

The second interface is between the Jetson and our cloud interface. Here, combined images and classification data are sent to the data storage service. This interface is fulfilled using the Google Firebase framework, which lends itself well to our Google Cloud backend.

The third interface is between the Jetson board and the stream server. Here the Jetson sends RTMP stream packets for later conversion to MPEG DASH.

The fourth interface is from the data storage service and the web client. Over this interface, the website, images, and notifications are transferred to the client.

The last interface is the connection between the stream server and webclient. After the server convert the packets to the MPEG DASH format, the client connects to the html server. This is how the web client displays the stream for the user.

VI. Assessment of Proposed Solution

1. Hardware

1.1 Strengths

- E-Con Systems cameras are supported by the Nvidia Jetson platform and have high frame rates at a 4k video resolution.
- The Nvidia Jetson board is widely used in embedded solutions for image recognition, and is a good form factor for our outdoor application.

1.2 Weaknesses

- E-Con Systems cameras are not widely used in the consumer market and not many examples of implementation are around.
- The Jetson does not have a very powerful processor in comparison to the larger desktop models. As such, any non-GPU operations could be a bottleneck in the system.

2. Detection and Classification

2.1 Strengths

- Convolutional neural networks are widely used today for image recognition, and are highly expandable to different image domains.

2.2 Weaknesses

- Most of the available training datasets consist of high quality images of birds and only birds, whereas the images our system will be using will likely contain more non-bird objects, leading to lower classification accuracy.
- Previous academic work with the datasets we are using are sparse.

3. Data Streaming and Storage

3.1 Strengths

- Cloud storage will be expandable and will not have to work about hardware limitation or hardware reliability for storage and the web interface.

- Moving video streaming from the cloud to a locally managed backend greatly reduces the cost of the whole system.
- Making use of existing cloud solutions reduces development overhead.

3.2 Weaknesses

- 4k images take up a lot of storage and could use a significant amount of our client's internet bandwidth.
- Firebase Cloud messaging is slow and increases notification delivery time.

3.3 Trade-Offs

- We are sacrificing some video resolution and frame rate during video streaming to ensure that the stream will not be choppy or delayed.

4. User Notifications and Frontend

4.1 Strengths

- A cloud-based frontend will provide a clean interface for notifications and viewing of birds.
- Limiting the number of images available on a single page makes our user interface more responsive.

4.2 Weaknesses

- The website will be hosted in the cloud and will have recurring costs.
- Limiting the number of images available makes it more difficult for the user to view older images.

VII. Project Timeline

The following two Gantt charts break down our project across both semesters. We will also discuss our project versus actual milestones and in the case they did not align, some insight into why.

1. First Semester Goals

	Task Title	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13	WEEK 14	WEEK 15
0	**Project Introduction**															
	Project Research															
1	**Machine Learning**															
1.1	Research Architecture															
1.2	Implement Architecture															
1.3	Test Accurary of Architecture															
1.4	Data Pipeline															
1.5	Dataset Processing															
1.6	Model Integration															
1.7	Hardware Testing															
2	**User Interface**															
2.1	Website Skeleton															
2.2	Displaying Images															
2.3	Displaying Stream															
2.4	Use Case Testing															
2.5	Notification System															
2.6	Setting and Filtering															
3	**Infastructure**															
3.1	Configuring Cloud Storage															
3.2	Uploading Images From Board															
3.3	Monthly Images Purging System															
3.4	Streaming Passthrough															
4	**Jetson Board/Camera**															
4.1	Order Camera															
4.2	Configure/Flash Jetson															
4.3	Configure Camera and Flash Drivers															
4.4	Capture and Store Images															
4.5	Object Detection (OpenCV and YOLO)															
4.6	Capture and Store Video															
4.7	Hardware Testing															

Fig. 5: Projected First Semester Gantt Chart

1.1 Projected

After the first semester, our overall goal was to have a preliminary prototype available for the client to take home over winter break. This entailed the following:

- The Jetson board and the camera interfacing properly with image and video capture and storage. There also needed to be some basic bird detection with YOLO[10] so that the board knows when to capture an image.

- The cloud storage will be configured so there is somewhere to send the captured images and videos of birds. The connection between the cloud and the web interface will also be configured so that images and video taken can be viewed by the client.
- Finally, we will also have a clear convolutional neural network architecture established. We hope to have a 90% accuracy on training images and 60% accuracy on real data from the board. We do not plan on having the neural network running on the board for winter break, but these metrics will be tested later on images captured from the board over break.

1.2 Actual

After the first semester, our overall progress did not match our original expectations. We were not able to get the full functionality of the prototype done. Our progress can be summed up with the following:

- We did have the Jetson board and the camera interfacing with image and video capture and storage. We did not yet have YOLO$_{[10]}$ running on the board for object detection.
- We did have the cloud storage configured such that the client could view a captured image on the website. We did not however get to the point where we could view a stream on the website.
- We defined a clear machine learning model architecture, but we were not able to implement the full architecture in the first semester. This did not allow us to meet our accuracy goal.

1.3 Discrepancies

The discrepancies between the projected and actual first semester results can be explained by the following.

- We experimented with many object detection frameworks and even different ways to integrate YOLO$_{[10]}$ into our project. Ultimately our priorities at the end of the first semester were more aligned with documentation and the final presentation, so we did not have time to complete the YOLO$_{[10]}$ integration.
- Our hardware development platform had a higher learning curve than we had expected. As such, we did not get the GStreamer camera video streaming done til second semester.
- Our accuracy goal for the first semester was too lofty. We were too optimistic in what we could accomplish. Realistically, modern research teams are able to achieve around an accuracy of 85 percent in the top one classifier. They are able to achieve this because they have over 100 layers in their models. We on the

other hand, always had the plan of using one fifth or one sixth of that since we are limited to an embedded platform. We were only able to create a toy model that had a higher accuracy than random, and we started our full model implementation.

2. Second Semester Goals

	Task Title	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13	WEEK 14
0	**Project Introduction**														
	Project Research														
1	**Machine Learning**														
1.1	Deployment Testing														
1.2	Model Training														
1.3	Model Testing														
1.4	Model Rework														
1.5	Pre-Processing Segmentation														
1.6	Sub-Image Classification														
1.7	SqueezeDet Modifications														
2	**User Interface**														
2.1	Stream Page														
2.2	Storage Page														
2.3	User Notifications From Log File														
2.4	Web Notifications														
2.5	Settings Page														
3	**Infastructure**														
3.1	Design Database														
3.2	Data Pipeline Between UI/Board														
3.3	Notification Processing														
3.4	Store Bird Tags and Statistics														
3.5	Notification Filtering														
4	**Jetson Board/Camera**														
4.1	Camera Controls														
4.2	Research Hardware Enclosures														
4.3	Purchase Hardware Enclosure														
4.4	Enclosure Modifications														
4.5	Debug Logging														
4.6	Bird Logging														
5	**Testing**														
5.1	Client Evaluation														
5.2	Integration Testing														
5.3	System Testing														
5.4	Acceptance Testing														

Fig. 6: Projected Second Semester Gantt Chart

2.1 Projected

After the second semester, our overall goal was to have a fully-integrated system available for the client. We wanted to focus entirely on implementing and refining our plan from the first semester. This entailed the following:

- A fully implemented convolutional neural network with a top three accuracy goal of 85 percent.
- A fully-fleshed out website with a clean user interface. The website was to be divided into stream, storage, and settings pages. Push notifications were also to be served to the user upon a bird classification.
- Infrastructure-wise, we wanted a designed database that could effectively store bird images with tags and statistics. In addition, it was to filter notifications based on the user's preferences.
- The hardware system was to be contained in a weatherproofed and modified electronics enclosure. The board was also supposed to have an extensive debugging and bird logging system.
- For testing, we wanted several client evaluations throughout the semester with an extensive range of integration, system, and acceptance-level testing.

2.2 Actual

After the second semester, our overall progress did not match our original expectations. While we were not able to get the full functionality of the project done, our implementation was in line with the spirit of the idea. Our progress can be summed up with the following:

- We finished implementing our full convolutional neural network model architecture and integrated it onto our board. Our top one accuracy was 34 percent.
- We integrated a CUDA-accelerated background subtractor to accurately detect moving objects in our camera's field of view.
- We completed our video encoding and streaming subsystem from the GStreamer pipeline on the board to our NginX server as part of our cloud infrastructure. This was then displayed on the website at 10 frames per second at 1080p resolution.
- We also fleshed out our cloud infrastructure. We configured all of the necessary services on Firebase. Additionally, we created a serverless runtime to manage the intake of data from the base station and deliver notifications to our client. Finally, we established a connection between our two backend systems, both cloud and NginX server, and the frontend.
- On our frontend, also we built up our entire website. This website is able to connect to our two backends and properly display the stored images as well as the video stream from the base station. This website conforms to web standards

for video including MPEG DASH and dynamically loading new images from our cloud storage as they are uploaded from the Jetson.
- Our hardware system is also enclosed in a weatherproof IP55-rated[6] enclosure to protect against hardware damage.

2.3 Discrepancies

The discrepancies between the projected and actual second semester results can be explained by the following.

- We did not meet our accuracy benchmarks with our neural network. Again, this may be a limitation of our 26 layer design, but is likely stemming from severe network overfitting. We also had TensorFlow-related memory leak issues that were hard to track down.
- Our project plan did not account for all the unforeseen challenges that we had to overcome in our development of this project. These put us off track on our expected goals.
- Our project uses a wide variety of software stacks that have to work seamlessly together. This is not a simple task; in particular, we had trouble integrating our OpenCV and TensorFlow code because of dependency issues.
- We introduced a number of bugs into our system while in the system integration phase. These bugs were hard to track down and to figure out the component(s) causing the issues. These put us off track on our goals.

VIII. Challenges: Risks / Feasibility

1. Feasibility and Challenges

First and foremost, we needed to find a way to efficiently classify birds. Currently, machine learning models are often run in resource rich environments. For our application, we will needed our models to run in a resource limited embedded environment. We dealt with this issue by heavily preprocessing the input to our neural network by cropping images down to areas of interest and resizing the images. This allowed us to train a model for smaller images reducing the overall footprint of our network. These changes negatively impacted the accuracy of the model, but were necessary in order to meet some of our timing constraints.

Additionally, there exists a risk that we will overfit our dataset during model training. We worked around this issue by saving checkpoints in our training process so we could

quickly restart the process. Along with that, we put work into augmenting our dataset using bird images from other datasets available online.

A risk exists that optimizations in one subsystem will result in a loss in functionality or performance in another. We will have to take great care to ensure that we consider the health of the entire system in every change we make. To accomplish this we will develop various integration tests to test the health of the overall system. These tests should function as a means of ensuring that changes are localized, and do not have unforeseen consequences in other subsystems.

The next challenge after identification will be transmitting the images from the board over to the user. Given the high quality optics that are being used, the board will need to dedicate a significant amount of time to video encoding and data storage. This will require a balance of processing resources dedicated to video streaming/storage and bird detection/classification. Resource optimization problems like this can be realistically solved if we dedicate enough time to testing our streaming and classification systems.

We addressed this issue by making use of the video processing software GStreamer. GStreamer does a great job of leveraging system resources for video encoding and greatly expediting the processing tasks. Additionally, we parallelized our program on the board by creating multiple processes for detection, classification, and uploading. By parallelizing our computations, we can fully utilize the hardware at our disposal and ensure a high video processing rate.

2. Security Risks

There are two primary targets from a security standpoint. First, there is the risk of an attacker gaining control over the physical hardware. To combat this, we have limited communications to the board so that information only leaves the board. By disallowing external input, we minimize the risk of external access; to fully compromise the Jetson, an attacker would have to gain access to the user profile present on the board. The second risk would be an attacker compromising one of our two server backends. On one front, we offload our security on to our cloud provider: we leverage their existing infrastructure to secure those communication channels. On our NginX server, we mitigate our risks by limiting who is allowed to publish video streams, and by only opening up access to the necessary directories on the server.

3. Lessons Learned

The main lesson we learned was how much work has to go into the integration efforts of a project. One of our biggest roadblocks throughout the project was that many components would work fine independently, but once they were be connected there would be some issues. For example, when we integrated our Tensorflow model into our prototype, the TensorFlow build for the Jetson ran on an older version of libpng than the version that OpenCV required. Because we did not incorporate adequate scheduling for these types of setbacks, our project timelines had to be pushed back. It also helps emphasize the need for better preliminary assessment of what can be realistically be accomplished in a timeframe, especially when accounting for uncertainty.

Another lesson we learned was the importance of communication and properly defining the interfaces between subsystems of the project. Formalizing small things like naming conventions and protocols between subsystems can be tedious, but the lack of standardization can create bugs that are difficult to find. This can also be easily addressed by thoroughly considering every aspect of the design during the planning phase, and it is something we really emphasized throughout the second semester.

IX. Standards

We had to take several standards into consideration when exploring our options on case design and purchase. As an outdoor product, this project must be able to withstand some amount of dust and moisture exposure.

The IP code, for ingress protection, is an international standard that measures “Degrees of protection against the penetration of solid bodies, the penetration of water, and the access of personnel to live parts.[6]” IP codes are formatted as IP##, where the first numeral measures its protection against dust and the second numeral measures protection against moisture. It would be appropriate for the enclosure to have an IP code of IP55. This corresponds to “Dust protected” and “Protected against water jets in all directions.[6]”

For video streaming, we make use of a wide variety of protocols and standards. First, for streaming data from the Jetson to the NginX server, we make use of H264 video encoding. This minimizes the data footprint of our video stream. Additionally, we make use of the Real Time Messaging Protocol (RTMP). RTMP is used to stream the H264 video stream from the Jetson to our NginX server. Finally, our NginX server makes use of the MPEG DASH protocol. This protocol was designed for streaming video to an HTML5 website, and it greatly simplifies our code base by allowing us to use existing Javascript libraries to display the stream.

As part of this project, we also had to adhere to different professional standards like the IEEE Code of Ethics. We did “commit ourselves to the highest ethical and professional conduct[11]” in accordance with the IEEE Code of Ethics. In particular, one principle from is of extreme importance to our project: “to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations.[11]” Throughout this project, we had to grow our technical skills and be realistic with our ourselves, client, and advisor about what we can and did get done.

X. Test Plan

To ensure our project meets our client's needs in all technical regards, we will be implementing and evaluating various types of testing. These tests cover both functional- and non-functional use-cases, as well as ensuring that the interfaces between project components meet specifications throughout the iteration process. To carry out these tests, we need specialized tools for both hardware and software testing.
user to browse this data. Finally, we must pass along data from the live video stream coming off of the Jetson board along to our user interface.

1. Hardware and Software Testing Tools

1.1 Hardware

We will ensure our box adheres to the IP55 standard[6]. These tests require the use of a table and a water jet capable of displacing 10 liters of water per minute.

1.2 Software

To ease the testing process, we will use Selenium Webdriver for testing our user interfaces, and TensorFlow to test our machine learning model.

Selenium WebDriver is a program for testing web pages. It is a means of automating user interactions from the perspective of a user. It will boot up a web browser, go to the specified urls, and attempt to complete the defined actions.

TensorFlow provides high-level APIs doing various mathematical calculations quickly and effectively in a machine learning context. Here, we will use it for evaluating our model's performance over a predetermined dataset, as well as streamlining the processing of that data.

2. Functional Testing

2.1 Unit Testing

Component-level unit tests will be written on a feature by feature basis. We will write simple functionality tests for the functions of our components, along with more rigorous interface tests for the integration of these components.

2.1.1 Hardware Enclosure Testing

Based on the manufacturer's claim, the enclosure we purchased has an IP55 certification. We will assume this is valid since the tests to verify IP standards are not realistic outside of an industrial grade testing environment. We will simply verify that the slight modification we made to the top window of the enclosure does not affect the waterproof integrity of the enclosure.

2.1.2 Machine Learning Model Testing

When we test the ML model, our testing procedure is fairly standard:

1. Give the model an image as input, with the appropriate truth label attached
2. Compare the model's predictor confidences to the ground-truth label in (1)
3. Add this result to a table of results
4. Repeat steps 1-3 for each image in the test dataset.

For our dataset (CalTech-UCSD Birds 200), we have 5794 test images. We can evaluate the model's performance via the formula:

$$\frac{\#\ correct\ predictions}{\#\ total\ images}$$

TensorFlow provides the ability to evaluate this process over a single large batch of images, which is used during both training and evaluation.

2.1.3 Stream Testing

We must test that the stream meets several criteria: that the hardware platform is able to post camera data to the online server successfully; that the web interface is able to read and view these datas as they are updated; and that the entire process maintains a high quality of video. To do so, we will test both parts of the streaming separately - first, verifying that the server sees data uploaded to it from the Jetson, and second, that when using the web interface, these data points are able to be updated and accessed in real-time. When testing that the quality of video remains acceptable, we will compare the output from the web interface stream to the displayed on-board video feed, checking that the image quality remains similar and no quintessential features have been lost.

2.1.4 Database Testing

When testing the Cloud-based database, we will need to ensure that our on-board software can upload large 4k images in a reasonable timeframe - at least as quickly as we save them, so we don't build a backlog to be processed. This will be done by

monitoring the database while also launching “upload” jobs on the Jetson, checking if the data to be uploaded is seen before the next job is launched. We also must verify that users who can view the webpage can also view the images stored on the database, but do not have the ability to edit the data stored in any way - this is a matter of verifying the Web Interface displays the images correctly, and properly restricting access to our database via Google Firebase.

2.1.5 Web Interface Testing

Using Selenium WebDriver, we will test the functionality of the web interface. Each web page will be tested to ensure that it acts according to design: in the stream webpage, we will test that the page opens with a streaming element on located in the center of the page. For the bird information page, we will ensure that the cloud links loads and that it displays the appropriate images. We will also test the notification system by having the unit tests post a notification to the users, after which the testing framework will validate that it has received the notification. Lastly, the navigation bar will be tested on each page to ensure the client can easily navigate the interface, verifying that each link loads to ensure that no links are broken.

2.2 Interface Testing

After verifying that the components all perform their designed functions, we must verify that the specified contracts for all interfaces are abided by. To do so, we will test each interface by running or simulating the relevant components.

The first interface is the interface between the camera and the Jetson board. We will not have to explicitly test this interface since all of the software to control the camera is provided by the camera manufacturer. That being said will still have to keep this interface in mind as a possible source of error in out other tests.

Second, we will need to test the interfacing between the Jetson board and our cloud services. This pipeline will be our only method to pull detection and classification data off of the board, so it is of the utmost importance that this interface be reliable. We will be transmitting pictures of birds along with classification data over this link. Additionally, we plan on implementing camera controls in the user interface, so there will be a backwards link to send those commands.

Our next interface consists of the various software interfaces in our cloud backend. Here we will need to make sure that pictures sent by the board are properly delivered to the various backend services. From there, data will be sent to the storage layer, where we

will store pictures and metadata in our database. At the same time, our notification system should index the data and send out a notification of the event to our client. All of these services are very time-critical and we will have to test to ensure they both run efficiently, and reliably.

Penultimately, we have an interface for the board software to interact with the neural network architecture, which primarily consists of calculating detection & classification for images. This interface will need to be monitored to ensure that the neural net is operating correctly, and can be used to extract the classification data mentioned above.

Finally, we will have to test the interface between our backend services and our user interface. There will be various data streams from the backend to the user frontend. First, there will be a notification system that must quickly and efficiently receive a bird sighting event, and pass on this event to our user front end. Next, we must be able to pass along the images and metadata we have stored along to our webpage to allow the user to browse this data. Finally, we must pass along data from the live video stream coming off of the Jetson board along to our user interface.

2.3 Integration Testing

Our integration testing consists of testing the communication links and interactions between adjacent components in our system. This set of links includes:

1. Jetson Board to Cloud backend
    a. This link is focused on sending captured images along with the output of the classifier. Additionally, this link will carry the video stream from the camera in order to pass it on to the user interface.
2. Cloud to Client (Website)
    a. This link is focused on serving our webpage to the end user as well as send the large image files from our storage layer.
3. ML Model to Board
    a. We plan on working our model into our video processing pipeline. As such it will be constantly receiving video frames, and, on detection, output the classification data as well as the single frame for transmission to the storage layer.

To test these links we plan on writing unit tests in each component to test its relevant link. For example, the Jetson board will test its connection to the cloud by attempting to upload an image to the backend, then query the backend for images. If the test image is

not in the query result, then the test will fail. This style of testing will be implemented on each component of our system.

2.4 System Testing

Our system testing facilitates checking the system's performance from end to end. In our tests, we are looking for the following properties:

- Birds detected and classified in real time (faster than 3fps)
- User is notified of the birds
- Web client displays detected birds' stats accurately
- Hardware system functions unimpeded in Iowa's weather

To test these properties, we will run a set of supervised tests both inside a controlled test environment as well as at the client's deployment site. During the tests, we will be monitoring the hardware to ensure that each of the above properties are satisfied. Additionally, we will be recording the video produced from the hardware's camera to use in re-creating any scenarios where the system fails.

To test in the testing environment, we will model the real world by printing images of birds. These pictures will be moved into the camera's field of view to simulate a bird flying into view. This will test the system's ability to detect and classify accurately. Separately, testing at the client's site will be a real world test, during which the tester will monitor the hardware and manually record any birds in the field of view. If a bird is in the field of view, the tester will record the time and check to see if the hardware detected and classified the bird. This real world test will verify the hardware system's resilience to Iowa's weather, as well as aiding in gathering data in realistic usage scenarios. A successful test in either environment will show the system processing at least 3 frames per second, the camera detecting the bird, notifications sent to the user, the web interface displaying images of the detected birds, and a viewable stream of the camera feed.

2.5 Acceptance Testing

Acceptance testing will be performed in two stages. Stage one will be usage testing and demonstration of the system in a test environment. Stage two will be real-world testing at the client's site.

During stage one, the client will be invited to a testing lab. Here, we will perform a systems test on printed images of birds. The client will be able to see the performance of the entire system. We will provide a list of client's system requirements. These are

the same requirements listed in the project plan. The client will be asked provide feedback on how well the system fulfilled each requirement. This feedback will be incorporated back into the design before the next stage.

Stage two is a test at the client's site. Here the system will be tested under real world conditions. The box will be exposed to outside weather and continuous use for 24 hours. After that testing, period the client will be asked to respond how well the system fulfilled each of the requirements.

3. Non-Functional Testing

Non-functional testing will be looking at empirical usability of the system, rather than measurable functionality. The non-functional requirements are:

- The client should have no issues setting up the camera and moving the system's enclosure.
- The client should be able to seamlessly use the web interface provided to watch the streamed video, browse the captured photos and videos, and configure the notification system.
- The total cost of the project should be under the project's budget of $1500.

3.1 Validation Testing

In order to validate our project design against these non-functional requirements, we will meet with our client to verify the project's adherence to and satisfaction of the desired requirements. Our design will be presented and demonstrated, along with progress updates since the last meeting. This testing will primarily focus on the client's preferences with regards to the usability and overall aesthetic of the system, specifically with the web interface and system setup.

To test ease of setup, the system will be installed and setup with the client. Afterwards, the client will be asked about the difficulty of the installation and if they believe they could install it themselves. If the answer is no, the client will be asked for feedback as to why and if they have any suggestion to improve the installation experience.

Testing of the web interface ease of use will be in sequence with acceptance testing. Immediately following completion of the acceptance testing, the client will be asked if they believe the system is easy to use. If the answer is no, the client will be asked to provide a reason why, and improvements will be made based on the feedback.

Cost will be determined as the semester continues. The cost of each part is logged with the Electronics and Technology Group (ETG). We keep record of the cost of each part to ensure the cost is less than $1500.

4. Process

Our testing process will follow a fairly straightforward approach. Unit and component testing will be run continuously throughout the development life cycle. This is so that code currently under development can be verified for completeness and correctness; we will run our tests before commiting code to the repository to prevent us from having broken code. This helps ensure that any future bugs arise from the code we are writing rather than from previous code, and can be tracked down and fixed efficiently.

Next, we will run Integration tests when we touch code that directly interacts with any of the interfaces we have defined above. The purpose of the interfaces is to function as a contract between two services, and whenever we touch components which interact with these contracts, we need to verify the contracts are still being upheld. This way, we can be confident that all promises made by the contract are always upheld, and each component can be developed isolated from issues with another.

Our final set of testing is system, acceptance, and validation testing. This testing will consist of full end-to-end system testing, partially with client presence. We will manually start the process by feeding the system a picture of a bird and test the user stories we have developed. This includes testing video streaming, user notifications, and image access on the user interface. These will be evaluated for performance metrics such as stream, notification, and user interface response time, as well as detection and classification time. These tests will be repeated by the client, who can then offer feedback from an end-user usability viewpoint.

5. Simulation and Modeling

A major component of our project is the ability to detect and classify birds. To this end, as mentioned above, we are using a CNN to determine appropriate labels from images. As part of creating a functional CNN, we will have to determine filter weights for each convolutional layer. To do so, we use loss-minimization techniques, where the loss is calculated based on processing an image whose label is predetermined, and comparing the network's predicted label to the known "ground truth" label. This process, commonly known as "training," is a form of simulation, as the input images are collected, labeled,

and organized beforehand, and running them through the network is done in a controlled laboratory environment, off of the Jetson board. Once the appropriate weights have been determined, they can be loaded onto the Jetson for use in classification calculation.

6. Testing Results

All components of our project satisfy their designed functionality as determined by our manual testing methods, although some of our initial ideal benchmarks are not met.

The hardware platform provides an adequate amount of processing power for a minimum viable product, is set up in a way that can continue to be improved upon, and meets our strict weatherproofing requirements. All of the software is set up and the system runs continuously without additional input.

The machine learning model has achieved a 99.5% classification accuracy on the training image set, but only a 34% accuracy on the test set after 300 training epochs. Figures 7 and 8 are the training and loss plotted over the training process.

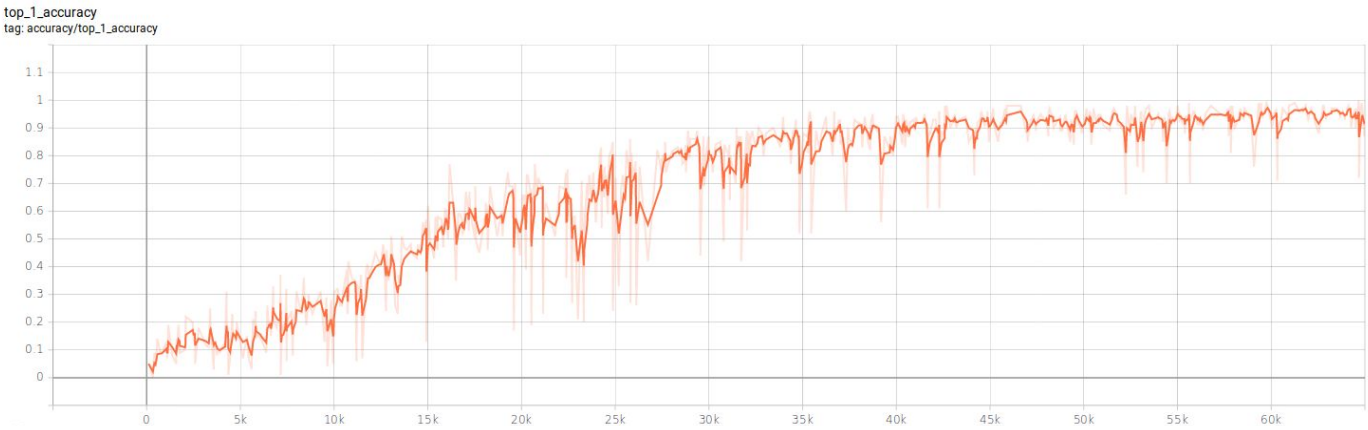

Fig. 7: Top 1 Accuracy, Training

The discrepancy between train and test accuracy is most certainly due to model overfitting - we did mitigate this somewhat by adding regularization into the model, but could be further improved with more regularization and the introduction of node dropout.

The video stream is able to be opened and viewed, but is not as responsive as initially hoped to be, only handling 10 frames per second at a quality of 1080p. A large portion of this is due to the immense amount of additional processing that must be done

alongside the streaming - the video processing and encoding and model evaluation notably taking a large portion of both the GPU and CPU.

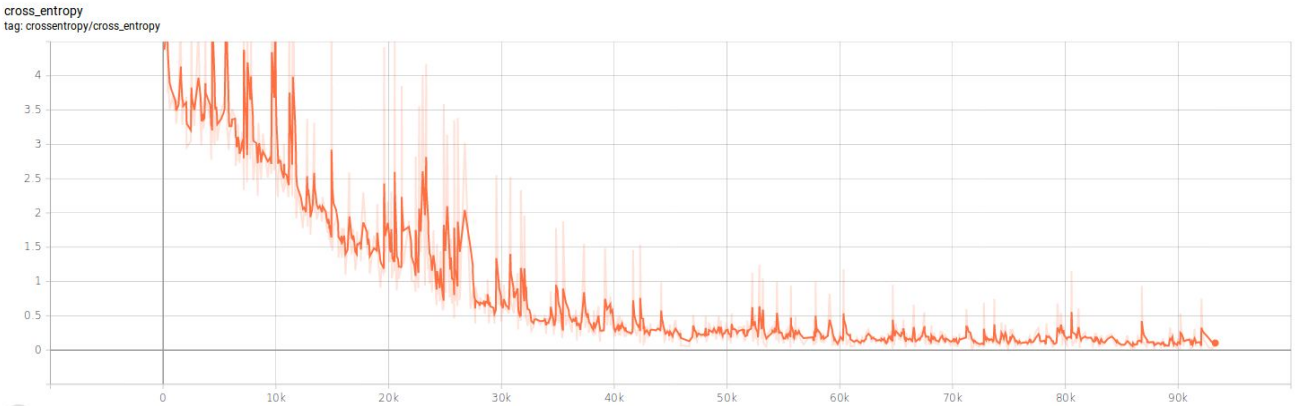

Fig. 8: Loss, Training

The database performs exceedingly well - data is able to be efficiently stored and retrieved, and system performance is unhindered by the database's performance. None of the launched jobs wait on the cloud database, and the web interface sees updates instantly when it requires them.

The web interface covers the minimum viable product laid out by our client successfully, being able to display and access the live video stream along with the labeled images uploaded when a bird is detected. Furthermore, these detection events post web notifications to users if enabled and are deployed in a reasonable timeframe to be able to see the detected bird on the stream. Navigation between the implemented pages works well, and the designed interface is functional. However, some additional features - such as board status logging and system interfacing via a remote terminal or graphical user interface - are beyond the scope of this minimum, and were not included in the delivered product. Additionally, the interface could stand to be more polished - during acceptance testing, the client notes that some features, such as the detected image displays, could be easier to use, and the visual design may still require work.

Integrating the entire system gives promising results - the entire system takes approximately 1 second to evaluate end-to-end, starting with a bird entering the frame and ending with the image being uploaded and displayed successfully.

In our non-functional deployment testing, we found that the requisite weatherproofing was satisfied by the enclosure developed. This enclosure also was able to house a

touchscreen through which the client is able to debug any setup issues, which contributes greatly to the overall ease of deployment.

XI. Conclusions

In summary, our project was to provide our client with an embedded wild bird classification system. We delivered a solution with moderate success with respect to the work done on object detection, bird classification, video streaming, website design, and overall system integration. Throughout this project, we grew as a team by facing and overcoming a number of design, implementation, and testing challenges. This reflects many of the challenges that are faced in the workplace when not only working in a team but also when developing a new product.

With respect to future work, given more time we would like to further improve our overall user experience. First off, we would like to improve the overall accuracy of our model. Next, we would flesh out our notification processing, including improving the speed of notifications and allow for filtering of notifications. As a whole we would like to improve user control of the overall system, allowing them to control the camera and fine tune various settings throughout our processing pipeline. Additionally, the system could be further updated to accommodate multiple cameras on the system. Currently, this could be accomplished by deploying multiple Jetson systems, but we would like to develop a truly distributed solution with multiple cameras feeding into a single Jetson processing platform or intercommunication between Jetson systems to avoid duplicate processing.

XII. Figures and Tables

The following figures were referenced and included in this document.

XIII. References

[1] “Northern Cardinal Identification, All About Birds, Cornell Lab of Ornithology,” *, All About Birds, Cornell Lab of Ornithology*. [Online]. Available: https://www.allaboutbirds.org/guide/Northern_Cardinal/id. [Accessed: 29-Sep-2018].

[2] K. Kaiser, “Designing A Deep Learning Bird Camera Platform,” *Make Art with Python*, 14-Feb-2018. [Online]. Available: https://www.makeartwithpython.com/blog/deep-learning-bird-camera/. [Accessed: 01-Dec-2018].

[3] M. S. Norouzzadeh, A. Nguyen, M. Kosmala, A. Swanson, M. S. Palmer, C. Packer, and J. Clune, “Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning,” Proceedings of the National Academy of Sciences, vol. 115, no. 25, Jun. 2018.

[4] Iandola, Forrest N, et al. “SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and <0.5MB Model Size.” *ArXiv*, CoRR, 4 Nov. 2016, arxiv.org/abs/1602.07360. Accessed 27 Oct. 2018.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[6] “Degrees of Protection Provided By Enclosures,” *Schneider Electric*. [Online]. Available: https://www.schneider-electric.com/resources/sites/SCHNEIDER_ELECTRIC/content/live/FAQS/176000/FA176928/en_US/Degrees of protection IP,IK, NEMA.pdf. [Accessed: 27-Sep-2018].

[7] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier Nonlinearities Improve Neural Network Acoustic Models,” 2013.

[8] V. Nair and G. E. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines,” *International Conference on Machine Learning*, 2010.

[9] X. Glorot, A. Bordes, and Y. Bengio, “Deep Sparse Rectifier Neural Networks,” *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, vol. 15, Apr. 2011.

[10] J. Redmon, S. Divvala, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection.” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. [v5]

[11] IEEE Code of Ethics. (2018). *IEEE Potentials*.