Distributed Wild Bird Surveillance and Recognition System

Design Document

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

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

Team E-Mail: sdmay19-10@iastate.edu
Team Website: http://sdmay19-10.sd.ece.iastate.edu
Last Revised: December 2, 2018 | Version 2

Table of Contents

List of Definitions

GPU - Graphics Processing Unit
FPS - Frames Per Second
HW - HardWare
SW - SoftWare
VM - Virtual Machine
ML - Machine Learning
CNN - Convolutional Neural Network
RESTful - REpresentational State Transfer (API)
Real-Time - Processing data as it is received
GCP - Google Cloud Platform
POST - an HTTP request to accept data, most often for storage

1 Introduction

1.1 Acknowledgement

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. SMAY19-10 would also like to thank our client, Dr. Joseph Zambreno, for the financial resources provided and his continued support of this project. We would also like to thank Craig Rupp, our advisor, for his guidance as we develop this project.

1.2 Problem and Project 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. Differences of this type remain constant for many of the birds across North America; because of this, bird identification can be a complex task.

To provide a solution to this problem, Dr. Zambreno has proposed creating a self-contained system that can detect and classify bird species in realtime in a consumer setting. The current consumer products available are limited to nature cameras that are constrained by insufficient video streaming and photo capture capabilities, which causes them to be unable to classify birds.

As a result of this project, we will have delivered a well-documented HW/SW product to our client that will successfully identify North American birds in real time, as well as store and stream high-quality images and video for remote access.

1.3 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 will have to include some type of enclosure to prevent the damage of our project.

1.4 Intended Users and Uses

The primary user of this product will be our client, Dr. Joseph Zambreno. The primary end-use of this product will 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 physical product will be necessary unless he would like to change the position of the product within his backyard.

1.5 Assumptions and Limitations

Assumptions:

- The operating environment will not be in extremely hazardous weather or conditions.
    - Normal weather conditions include temperatures ranging from 40°-80° 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.

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 TX1.
- 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.

1.6 Expected End Product and Deliverables

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

The hardware system will facilitate data streaming and bird detection in real-time. This system will primarily consist of an NVIDIA Jetson TX1 and a 4k camera. The Jetson will be responsible for managing necessary hardware drivers for the camera and running the bird detection & classification system, as well as uploading media to the storage system. The camera will capture 4k photographs on-demand, as well as stream high-quality video taken from the client's yard in real-time. The system will be robust enough to withstand Iowa weather and provide a stable camera mount for a clear video stream.

The detection and classification system will be running on the Jetson board. This system will leverage a pre-trained convolutional neural network (CNN) to locate and identify birds in the field of view. The detection and classification will run in real-time and be able to identify all of iowa's native bird species. After detection, the camera will take a photo of the field of view.

The storage system will encompass cloud backup storage as well as local storage. These stored videos will be able to be streamed to a web client on request. In addition, the system will be able to stream live video from the bird feeder. The video playback and live streaming will be at least 1080p quality or higher at 30 fps.

The final product will deliver a frontend web client and notification system. The frontend client will allow functionality to view previously identified birds as images and short video clips, receive notification upon new identification, provide a live video stream to the feeder, allow filtering for which birds are identified, and show statistics about bird identifications.

In summary, this project will accomplish:

- Creation of a hardware box that can:
    - Store and take pictures
    - Identify and classify birds in real time
    - Stream 1080p, 30fps video and store 4k images to a web client
    - Withstand Iowa weather and prevent hardware damage
- Creation of a web client that can:
    - Display 4k images
    - Display a 1080p, 30fps video stream
    - Notify users of bird detection events
    - Display and allow filtering of bird statistics/notifications
- Creation of a detection system that can:
    - Detect and classify running faster than 3fps
    - Classification with >90% accuracy

2 Specifications and Analysis

2.1 Proposed Design

Proposed System

We propose using 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.

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 Jetson Platform through the Camera Serial Interface (CSI). Once the frame is on the board, we will apply image pre-processing to reduce the image size for more efficient computing. Then, the image will be fed through our neural network architecture, which may be one or more neural networks, for detection and classification. Afterward, we apply a image post-processing algorithm. From the board, these frames will be sent to the cloud for storage over the client's home network.

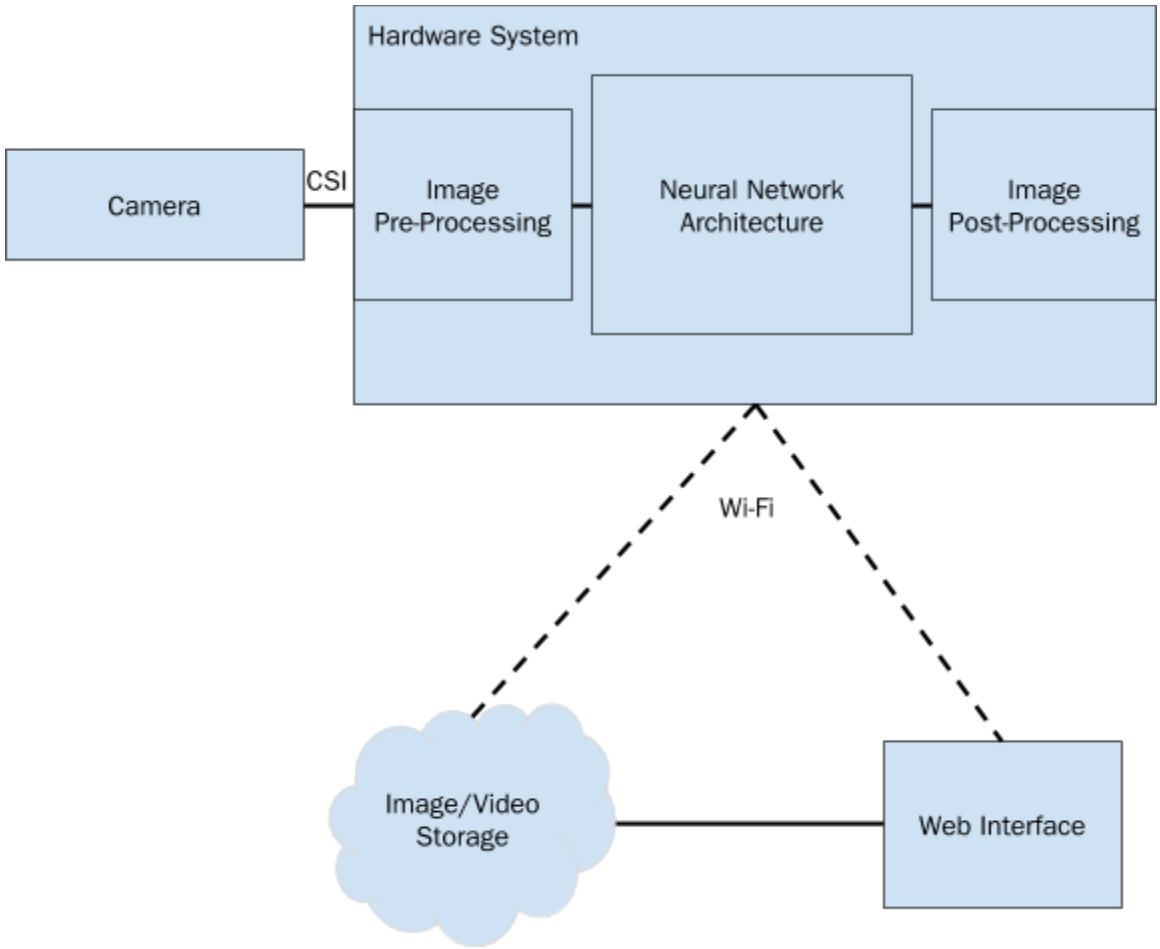


Fig. 2: System Block Diagram

Hardware

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 board with the camera. On top of that, the board is also compatible with GStreamer-1.0 for video recording and network streaming. We are also utilize a sd card and reader to store images captured from the board. So far, we have setup the Jetson TX1, flashed it with the correct software, setup the camera, and have sent an image to the cloud storage system.

Detection and Classification

We are considering using a cascading-style neural network architecture. The first neural network will be solely for bird detection. The second will be a convolutional neural network for the bird classification. We are splitting these two up because it is much faster to detect a bird than to classify one. We are currently investigating different neural network architectures to determine which one will give us the high accuracy rates we need, at the processing rates we desire. So far, Squeezenet's architecture[2] has been promising in our preliminary analysis, providing a good balance between processing speed on Jetson platforms[3] and accuracy on complex datasets.

Data Streaming and Storage

Our storage solution will be based in the cloud using Google Cloud Platform (GCP). The Jetson board will connect to an endpoint for a cloud application and stream picture data as well as classification/detection data to the cloud service. The cloud service will backup the data, and make it available to the frontend.

For video streaming, the cloud service will establish a GStreamer connection between the board and front end. Establishing this connection between the board and the user will help reduce overheard from cloud processing, and reduce cloud costs. The user can then choose if they would like to save certain clips from the stream in our cloud storage solution. We are currently investigating how the streaming pipeline will be constructed.

User Notifications and Frontend

The frontend solution will be implemented also using GCP. The website will allow users to view previously identified birds, a live stream of the bird feeder, and change notification settings. Each previously identified bird will have an associated video or image, and statical information about the detection/classification. The user can select which birds to be notified for and modify the threshold at which to notify. The notification system will be implemented using the Web Notification API. Clients will use a web browser to subscribe to notification from the bird watching application. We are currently designing the user interface and specifications for the web interface.

2.2 Design Analysis

2.2.1 Hardware Analysis

So far, we have primarily worked with the NVIDIA Jetson TX1 board - this board was selected because of its onboard GPU, which we can use to run our ML model very efficiently. We have currently installed the necessary software components onto the board, including camera drivers received from the manufacturer as well as Tensorflow and associated libraries. The camera

hardware operates within desired specifications, and is able to take pictures and video ranging from 640x480px at 120fps to 4k (4096x2160px at 28fps or 4192x3120px at 20fps).

In general, this approach has worked moderately well so far - we did experience some issues installing the necessary software on the board (due to setup issues), but these shouldn't be a problem if we decided to go with another hardware device in the future, since we've already found the cause and remedied them. Additionally, we ran into issues using existing image detection frameworks which are computation-heavy with the Jetson - YOLO v3 is unstable and unreliable, even with images preloaded on the board. However, YOLO v2 (which is less resource-intensive) runs without issue.

Moving forward, we'll need to closely monitor our capabilities to run the software we need in "real-time". If we find the compute capabilities of the TX1 lacking, we may need to upgrade to a TX2 model (which is roughly twice as fast, from our analysis).

Hardware Troubleshooting

- When installing software on the Jetson, our setup in the lab was:
    - Jetson connected to Internet via Ethernet to a Network Switch
    - Virtual Machine connected to Internet via virtual Ethernet from host
    - Host connected to Internet via Ethernet to a Network Switch
    - Jetson connected to Host via USB, forwarded to VM
- We did have issues with using a VM - it was noticeably slower than a native installation of Linux. We recommend using a native installation if available.
- Our VM initially was configured to use USB 1.0 - the installation only works when using USB 2.0
- We did have an issue related to our VM's filesystem - we aren't quite sure what caused this, but the entire root filesystem locked itself to read-only. We had to reinstall a new VM, and would recommend being aware that this may occur.
- YOLO v3 is killed partway through network construction. We believe this is due to insufficient memory / incorrect configuration, and troubleshooting is ongoing.

2.2.2 Machine Learning Analysis

So far, we have tried running a relatively shallow architecture (2 convolution layers, 2 maxpool layers in a conv-maxpool-conv-maxpool configuration), which gave us approximately 10% accuracy (better than random, but still poor). Because this was such a simple architecture, we didn't necessarily expect a high accuracy.

We think, from doing preliminary market research and testing with the Jetson, that we will be able to run much more sophisticated architectures without running into issues, so our next logical step is to try a deeper network. We have selected SqueezeNet's architecture[2] as a promising basis -- the entire model should be less than 1MB of storage, and while being a nontrivial architecture (30 layers), the model is relatively cheap to compute due to many layers being 1x1 convolutions. Our modified architecture (*Appendix A*) is a bit more lightweight, with some layers being resized based on our input images.

Moving forward, we may find that we can achieve >90% accuracy only when using the top 3 predictors of the network - this may change some of the front-end and infrastructure storage slightly, but overall shouldn't be a major issue.

2.2.3 Infrastructure Analysis

Currently, we are planning on building a cloud-based service to handle the storage and retrieval of stored images, as well as handling the communication between the board and the the web interface. We believe that a cloud based solution will provide us with a scalable solution to storage. Additionally, one of the stretch goals for this project was to build the solution into a distributed network of cameras. With our cloud-based approach, we can easily add more base stations the the system to allow the client to focus on multiple points in their backyard.

Based off of our currently market research, we plan on deploying our cloud solution on the Google Cloud Platform (GCP). We believe the majority of our cost will come long-term image storage, and in this GCP provides the cheapest storage options.

As for the implementation, we plan on building it up as a RESTful service. The board will submit POST requests containing the images and metadata. From there the data will be passed along to the storage layer to be placed into blob storage for future retrieval. Additionally, the storage layer will create an entry in the database for the image. The database will include a key to the image in blob storage along with columns for all of our metadata. Separating our image storage from the database should help keep the size of our tables down and improve our access times for queries. Additionally, storing the metadata in the database will make the filtering process much simpler.

Along with this, we will be using a GStreamer pipeline to manage the data on the board. This pipeline will include scheduling for the detection and classification as necessary, as well as submitting the aforementioned POST requests. Currently, we have streamed images locally through the pipeline, but are still working on streaming live video and sending media to the cloud via this pipeline.

2.2.4 Front End Analysis

The front end is currently in its design phase. Here, each page is being given a general layout. Later, the details of each page will be thought of and drawn out. In parallel to this, we are utilizing bootstraps template webpages to create a proof of concept for each page. This ensures that we can deliver a final product for the front end.

As of today, the front end is planned to have five web pages a splash page, a stream page, an archive page, a stats page, and a settings page. The splash page is the “about us” informational page. Here we will list our names and explain what the website is for. The next two pages, stream and archive, are the front end of the classification and detection system. As the naming suggests, the stream page will have a live stream of the bird feeder. It will show on a sidebar what the system detects in real-time. The archive page is where images of previously-recognized birds will be located. Here, a single bird can be selected, which will show more detailed information about its detection. Next is the stats page. Here, the site will be able to show reports and graphs about the birds detected. One idea for this page is for it to display pie charts of detected birds over some time period. Lastly, on the settings page, the client will be able to set his preferences for notifications (such as which species to receive notifications about) and configure any details about the website.

In parallel to the layout, we have created a prototype web page to learn about Bootstrap development. Bootstrap is a front end web framework which gives templates for web elements. We are hoping to utilize it to organize the full front end of our web page. From our current research, Bootstrap will be able to organize and beautify the front end, but more work will be needed to implement streaming.

3 Testing and Implementation

To ensure our project meets our client's needs in all technical regards, we will be implementing and evaluating various types of tests. These tests will cover 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 some of these one-time and on-going tests, we will need specialized tools for both hardware and software testing, described in Section 3.2.

3.1 Interface Specifications

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.

3.2 Hardware and software testing tools

Hardware

We plan on ensuring our box adheres to the IP43 standard. These tests require the use of a table and a water jet capable of displacing 10 liters of water per minute.

Software

We plan on writing our own code to test the software components of our system. The majority of this will be accomplished using unit testing tools for the appropriate languages: PyUnit for Python, Catch for C++, and Jasmine for JavaScript. Most of these testing frameworks also provide integration testing features. Finally, we will use Selenium Webdriver for testing our user interfaces.

PyUnit (also known as unittest) is the built-in unit testing library in Python. It provides a user-friendly set of unit testing functions that will help automate the testing process.

Catch is a unit testing framework C++. Similar to PyUnit, it provides unit testing functions that will help automate the testing process. The main advantage of Catch over other C++ testing frameworks such as GoogleTest or the BOOST test library is that it is easy to set up and requires very few dependencies.

Jasmine is a popular unit testing framework for JavaScript frontends. It is often paired with Angular, a common JavaScript framework, but will work with any of the popular frameworks as well. Tests are very simple to write, and will help us automate the unit testing process. Additionally, Jasmine provides very extensive Integration testing features.

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.

3.3 Functional Testing

Unit Testing

Unit tests will be written on a file by file basis. We will write simple functionality tests for the functions in our files. Additionally, we will write more thorough interface tests for the overall components of our project.

ML 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 current dataset (CalTech-UCSD Birds 200-2011[4]), we have 11788 test images. We can evaluate the model's performance via the formula:

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

Hardware Enclosure Testing

We will test that the box adheres to the IP43 standard. The tests for this certification are two fold. First we must ensure there are no holes in the enclosure larger than 1mm. This should limit access into the box and prevent wildlife from tampering with the system. Second, we must test for waterproofing. These tests consist of placing the box on a table and spraying the enclosure with a

spray nozzle for period of 5 minutes at various angles. After spraying the box, we will ensure the inside of the enclosure is dry.

Interface Unit 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 can it displays at least one image. We will also test the notification system. The unit tests will post a notification to the users. The testing framework will then validate that it has received the notification. Lastly, the navigation bar will be tested on each page. We will ensure that each link loads to ensure that no links are broken.

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.

System Testing

Our systems 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 picture 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. 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, and the web interface displaying the detected birds.

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 according to the systems testing procedure for 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.

3.4 Non-Functional Testing

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

- 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.

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. Directly after 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 copies to record of the cost of each part to ensure the cost is less than $1500.

3.5 Process

Our testing process will follow a fairly straightforward approach. Unit testing will be run throughout the development life cycle. They will be continuously run when code is in development and we will run our tests before commiting code to the repo. This will prevent us from pushing broken code to the repo and will ensure that any future bugs arise from the code we are writing rather than from previous code.

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 test the integration between the two components to ensure. This way we assure that all promises made by the contract are upheld.

Our final set of testing is system and acceptance testing. This testing will consist of full end-to-end system testing. 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.

3.6 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 and run on the Jetson.

Our training process has previously used cross-entropy softmax[5] as our loss-minimization algorithm. Moving forward, we are instead using the Adam algorithm[6], as it has been shown to help models converge faster.

3.7 Results and Challenges

We have successfully connected the camera to the Jetson board, and used it to capture images and video. This required setting up the camera drivers on the Jetson. We have configured a database in Google Cloud Platform. We are able to upload images from the board to the cloud backend, and then serve them up on a mock user interface.

We are still continuing to work on our classification model. We began with a toy architecture based on 2 convolutional layers and 2 maxpool layers in an alternating configuration. This resulted in ~10% accuracy in testing (note that a random selection gives 0.5% accuracy). This is will act as a benchmark for our future architectures.

Implementation Issues and Challenges

We are currently facing issues with the Jetson platform. It was difficult to configure at first, but now we are running into issues with third party models, such as YOLO v3[7], running. We were hoping to use an established model like YOLO[7][8][9] for object detection, so we will need to work out these issues moving forward. Additionally, have faced some issues with training our classifier. The process can be slow since we do not have access to better hardware. We are hoping to use a university machine, as well as streamline our testing pipeline in order to speed up the testing process.

4 Closing Material

4.1 Conclusion

In summary, our implementation thus far has included development across the four main areas of our project: hardware, detection and classification, data streaming and storage, and user notifications and frontend. The Jetson board and E-Con Systems camera are configured properly with the JetPack SDK and drivers. Basic image and video capture are available and a third-party object-detection algorithm, YOLO[7][8][9], is running on the Jetson. Saved image files are successfully uploaded to our Google Cloud infrastructure. The web interface shows the images from the cloud platform. A machine learning architecture has been established and the model is in the process of training. The combination of all this progress makes a basic functional prototype of the design for our client.

Our goals for next semester primarily revolve around “end-to-end” functionality: taking high-quality imagery of birds; being able to detect & classify birds accurately; efficiently transmitting, cataloging, and storing the imagery and appropriate labels; and easily accessing that stored information via a functional web-based user interface. There will also be a significant amount of testing conducted next semester to ensure the quality of our product. The combination of all these efforts will lead to a well-documented HW/SW product to our client that meets their needs to successfully identify North American birds in real time.

4.2 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] 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.

[3] O. Shykhkerimov and I. Mishchenko, "Real-time Object Detection with Convolutional Neural Networks," in *AI Ukraine*, 29-Sep-2018.

[4] Wah C., Branson S., Welinder P., Perona P., Belongie S. "The Caltech-UCSD Birds-200-2011 Dataset." Computation & Neural Systems Technical Report, CNS-TR-2011-001

[5] G. Hinton, N. Srivastava, and K. Swersky, "Lecture 4c; Another diversion: The softmax output function," in *Neural Networks for Machine Learning*, 25-Oct-2018.

[6] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization." Ithaca, New York, 30-Jan-2017. [v9]

[7] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement." Ithaca, New York, 08-Apr-2018.

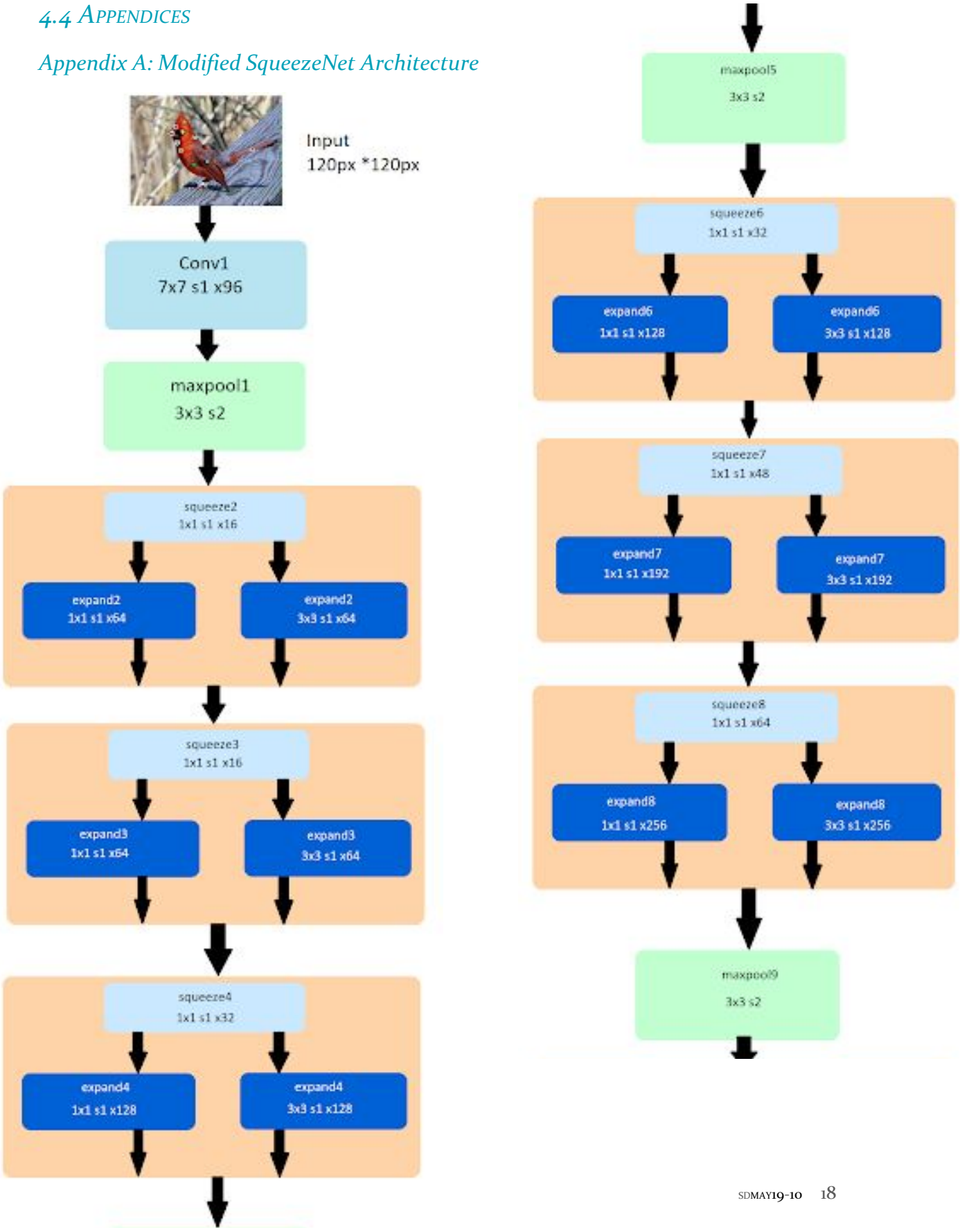
[8] J. Redmon, S. Divvala, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection." *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
[v5]

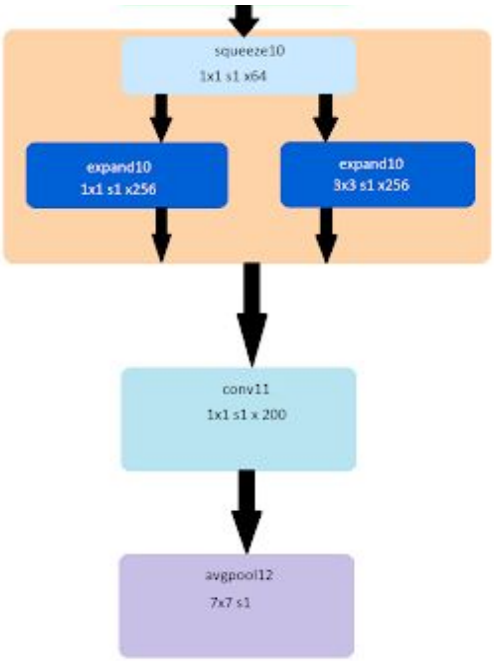
[9] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

4.3 FIGURES

4.4 Appendices

Appendix A: Modified SqueezeNet Architecture




squeeze10
1x1 s1 x64
expand10
1x1 s1 x256
expand10
3x3 s1 x256
conv11
1x1 s1 x 200
avgpool12
7x7 s1