Distributed Wild Bird Surveillance and Recognition System

Design Document

Team Number: SDMAY19-10
Client: Dr. Joe 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: October 7, 2018 | Version 1

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

GPU - Graphics Processing Unit
HW - HardWare
SW - SoftWare
VM - Virtual Machine
ML - Machine Learning
CNN - Convolutional Neural Network
RESTful - REpresentational State Transfer (API)
POST - an HTTP request to accept data, most often for storage
QED - Quod Erat Demonstratum (*that which was set out to be demonstrated*)

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. Joe 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

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 HW/SW product to our client that will successfully identify North American birds in real time.

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. Joe 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.

1.5 Assumptions and Limitations

Assumptions:

- The operating environment will not be in hazardous weather or conditions.
- The end product will only be used in the client's backyard.
- The system will have access to the client's Wi-Fi and power.
- The system will have access to some cloud interface.
- The client will have access to the Internet to access the web interface.
- 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 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 wild bird project has several key end product deliverables by the end of the second semester. The core deliverables are the hardware system, the detection/classification system, the data streaming/storage system, and the frontend/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 camera will stream real time video from a bird feeder located in the clients yard. 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 sourced hardware. This system will utilize a neural network architectures 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.

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.

2. Specifications and Analysis

2.1 Proposed Design

Include any/all possible methods of approach to solving the problem:

– Discuss what you have done so far – what have you tried/implemented/tested, etc?

– We want to know what you have done

• Approach methods should be inclusive of **functional and non-functional requirements** of the project, which can be repeated or just referred to in this section

If your project is relevant to any **standards** (e.g. IEEE standards, NIST standards) discuss the applicability of those standards here

Proposed System / System / Solutions

We propose using a self-contained embedded platform that is equipped with several key features: online connectivity, internal storage for images and videos, 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 Platform through the Camera Serial Interface. Once the frame is on the board, we will apply some image pre-processing to reduce the image size for more efficient computing. Then, the image will be fed through our neural network architecture. This may be one or more neural networks for detection and classification. After, we will 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.

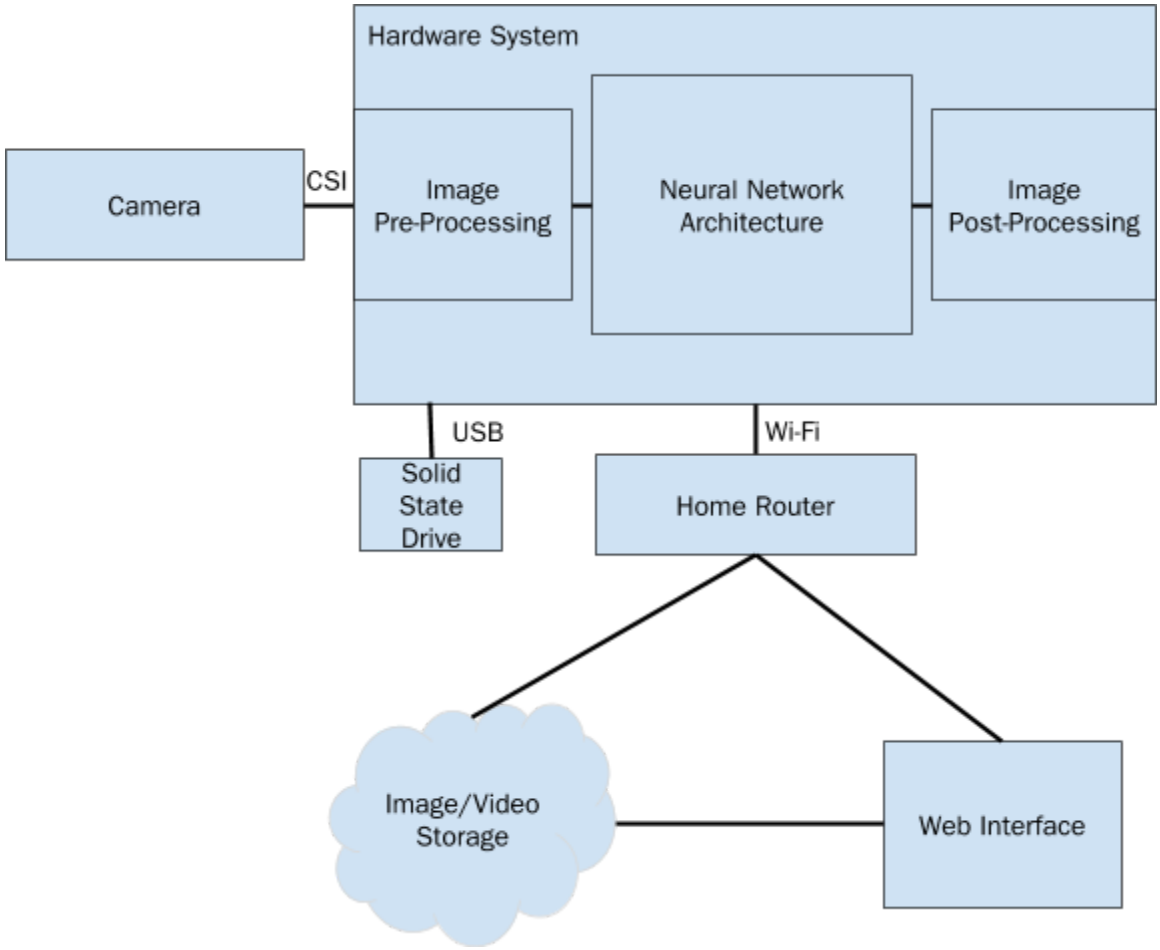

Fig. 2: Block Diagram

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 having a solid state drive connected to the board over USB to have easy access to the captured images from local storage. So

far, we have setup the Jetson TX1, flashed it with the correct software, and currently working on interfacing the camera with the board.

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 bird accuracy rates we need.

Data Streaming and Storage

Out storage solution will be based in the cloud. 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 connection between the board and front end. Establishing a direct 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 working through which cloud provider we would like to use and investigating how we will pull the data off the board.

User Notifications and Frontend

The frontend solution will be implemented using AWS. 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 an RSS feed. Clients will use a RSS client 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

- Discuss what you did so far
- Did it work? Why or why not?
- What are your observations, thoughts, and ideas to modify or continue?
- If you have key results they may be included here or in the separate "Results" section

-Highlight the **strengths, weakness**, and your observations made on the proposed solution.

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, and are waiting on receiving camera drivers from the manufacturer. We have, however, received the camera - just not tested its functionality.

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.

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). Additionally, we may need to play around with the software drivers to get the camera to work effectively.

Hardware Troubleshooting

- When installing software on the Jetson, our setup in the lab was:
    - Jetson connected to Internet via Ethernet → Network Switch
    - Virtual Machine connected to Internet via virtual Ethernet from host
    - Host connected to Internet via Ethernet → 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.

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 as a promising one -- the entire model should be less than 1MB of storage, and while being a relatively deep architecture (100+ layers), the model is actually cheap to compute due to many layers being 1x1 convolutions.

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. We believe the majority of our cost will come long-term image storage, and in this GCP provides the cheapest storage options. We still need to do further research into our video streaming solution, but that may affect our cloud cost calculation.

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.

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, 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 amount of time period. Lastly, is the settings page. Here the client will be able to set his preferences for notifications 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

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or a software library

Although the tooling is usually significantly different, the testing process is typically quite similar regardless of CprE, EE, or SE themed project:

1. Define the needed types of tests
2. Define the individual items to be tested
3. Define, design, and develop the actual test cases
4. Determine the anticipated test results for each test case
5. Perform the actual tests
6. Evaluate the actual test results
7. Make the necessary changes to the product being tested
8. Perform any necessary retesting
9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you've determined.

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-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. The pictures will need to make their way to heuristic service which will increase our classification accuracy. 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 receive 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

– Indicate any hardware and/or software used in the testing phase

– Provide brief, simple introductions for each to explain the usefulness of each

3.2.1 Functional Testing

- Examples include unit, integration, system, acceptance testing

Unit Testing

Explaining that we will well define unit test for later revisions…. blah

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 weights 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), we have 5794 test images. We can evaluate the model's performance via the formula:

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

Integration Testing

Each link
Board -> Cloud (Images + Classification)
Board -> UI (Stream)
Cloud <-> UI (Web UI)
ML on Board (Real time Bird Stuff)

System Testing

Web detection system (Cam>NN>Cloud>WebUI || Cam>Board>WebUI)

Acceptance Testing

Zambreno

3.2.2 Non-Functional Testing

Testing for performance, security, usability, compatibility

Zambreno takes this and doesn't hate it

3.3 Process

Unit testing - Integration testing - system testing - acceptance testing

Our testing process will follow a fairly straight forward 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 service. So whenever we touch that interacts with these contracts, we need to test the integration between the two components. 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.

3.4 Results

No results as of this version.

Best project NA

QED

4 Closing Material

4.1 Conclusion

In summary, the majority of our work thus far has been exploratory, with the Jetson TX-1 and toy ML architectures. Soon, we will have more concrete implementations for the cloud infrastructure as we acquire hosting services, as well as web services.

Our goals for the future 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. We believe that the best course of action to these goals involves the Jetson board (due to its onboard GPU for efficient image processing and video encoding, as well as various other secondary features such as built-in network connectivity), a Convolutional Neural Network (due to their ability to learn feature maps from 2d imagery effectively), and the Google Cloud Platform (due to its competitive pricing model).

4.2 References

This will likely be different than in project plan, since these will be technical references versus related work / market survey references. Do professional citation style(ex. IEEE).

4.3 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar that does not directly pertain to the problem but helps support it, include that here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc. PCB testing issues etc. Software bugs etc.