A.L.V.I.N.N.

AUTONOMOUS LEARNING VEHICLE INTEGRATING NEURAL NETWORKS

PROJECT PLAN

Dec1709

Rockwell Collins – Josh Bertram

Dr. Jones & Dr. Zambreno

Bijan Choobineh – Team Leader
Darren Davis – Team Leader
Tracy La Van – Communicator
Jesse Luedtke – Key Concept Holder
David Schott – Key Concept Holder
Robert Stemig – Webmaster

dec1709@iastate.edu
http://dec1709.sd.ece.iastate.edu/

Revised: Mar. 31, 2017 / Version 2

Contents

1 INTRODUCTION 2
1.1 Project Statement 2
1.2 Purpose 2
1.3 Goals 2
2 DELIVERABLES 4
3 DESIGN 5
3.1 Previous Work/Literature 5
3.2 Proposed System Block Diagram 6
3.3 Assessment of Proposed Methods 7
3.3.1 Hardware 7
3.3.2 Software 7
3.3.3 Experiments 8
3.4 Validation 11
4 PROJECT REQUIREMENTS/SPECIFICATIONS 13
4.1 Functional 13
4.2 Non-Functional 13
4.3 Standards 13
5 CHALLENGES 15
6 TIMELINE 16
6.1 First Semester 16
6.2 Second Semester 16
7 CONCLUSIONS 18
8 REFERENCES 19

1 INTRODUCTION

1.1 PROJECT STATEMENT

This project is in collaboration with Rockwell Collins to use computer vision, machine learning, and neural networks to have a drone detect an airport runway or objects in the air. The project has two teams assigned to it and our team will focus on detecting objects in the air. Classifications of the objects that could be detected in the air include airplanes, helicopters, drones, birds, and stationary items such as buildings. This will require the team to learn about computer vision, machine learning, and neural networks to identify and implement the software and hardware requirements capable of performing these tasks. Furthermore, part of the project will consist of determining which machine learning algorithms are particularly suitable for our object recognition task.

Computer vision tools will be used to process images from a picture or video stream to detect key features of objects. Machine learning and neural networks will then be incorporated to perform teaching operations for identifying different objects or distinguishing between similar objects based on those key features. Since this problem is interdisciplinary in nature, we will also have to investigate to what extent traditional computer vision techniques could be incorporated into our solution.

1.2 PURPOSE

Rockwell Collins is an aviation electronics company dealing with military and commercial aircraft. Rockwell Collins could use this technology to introduce new products that could have multiple uses for their customers. Although we are starting simple with basic object detection this could develop into technology that the military could use to survey an area before sending troops in, detect enemy forces, identify bombing locations, or drop locations for aid packages. Commercial uses could be finding hotspots in forest fires, locating remote wreckage sites, finding lost hikers, or finding survivors of some disaster. The diversity of these different use-cases shows the extensibility, power, and usefulness of our project.

1.3 GOALS

Rockwell Collins' primary goal of this project is to detect objects on the ground (landing strip) or objects in the air (other drones). To accomplish these goals, we need to set some smaller goals to help us succeed. First, we have an educational goal. This goal is to learn about the topics covered in this project from computer vision to neural networks. Upon acquisition of this knowledge, we can then accomplish our next goal of deciding what software, technology stack, and equipment to use.

Once we have decided on the software to use we can start working with it and learning how to use it to proceed to the goal of object detection. To accomplish this goal, we will start with detecting a simple object (the letter X) on a plain background. We can then increase the difficulty of detecting an object by adding in background, similar objects, and multiples of the same object. The techniques we learn for detecting a simple object will then be applied to detecting the more complicated airborne objects.

Teaching the system to distinguish between objects would be the next immediate goal. Again, we will start with a simple object to learn the techniques needed to accomplish the goal. Once effective techniques have been established we will move to training with the airborne objects. With these goals achieved we can work towards our final goal of implementing all of this with our own neural network.

2 DELIVERABLES

To meet the goals outlined in our proposal, the following deliverables are necessary:

- Phase I: Education and determining software/hardware to use
    - Learn about computer vision, neural networks, and machine learning
    - Determine hardware to use (boards)
    - Determine software based on capabilities and ability to integrate with hardware
- Phase II: Detecting a simple object (the letter 'X')
    - Image processing - show image, gray scale, threshold, ORB feature detection, homography, and object matching (detect an 'X').
    - Detect 'X' with background noise in an image
    - Detect multiple X's in an image (with & without background noise)
- Phase III: Extending Phase I by detecting objects other than X's
    - Detect airborne objects and begin training/machine learning
    - Incorporate cascade classifier training using OpenCV. This prioritizes two sample image groups, positive and negative, where negative images are manually created with non-object images and positive images are generated via opvencv_createsamples utility
- Phase IV: Detect and classify basic images such as X's or handwritten digits with a simple neural network.
    - Training a network with images of X's and handwritten digits
    - Discover the limitations of simple neural networks
    - After detecting an object, report to control possible object identifications with associated confidence levels.
- Phase V: Detect and classify complex images with more advanced and deeper neural network models such as Convolutional Neural Networks.
    - Complex images contain many objects in different locations and orientations
    - Detect multiple objects within an image
    - Identify objects even if objects have several different appearances
- Phase VI: After successfully being able to identify objects, begin feeding the system back to back images building up to real-time image capture and video feed
- Possible Bonus Features
    - Tracking movements of identified objects
    - Feature Recognition: Ability to recognize various features present on objects (colors, symbols, unique traits, etc.)
    - Multiple cameras
        - 360◦ View: Like Nissan's 360◦ view technology around their vehicles, have a 360◦ view of what is around the system laterally and what is above and/or below the system.
        - Thermal Imaging or Night Vision
    - Beyond tracking movements, track fellow system in air and follow the system via autopilot.

3 DESIGN

3.1 PREVIOUS WORK/LITERATURE

Below are several resources/research that has previously been done on our topic. Besides those listed below, machine learning work has been done before on image detection and movement. For instance, Google has Google Draw Neural Network. We are currently still in our educational stage and learning about these resources.

Deep learning has been a very popular area of research so there have been many reputable papers published over the years. Additionally, a specific type of neural network called a Convolutional Neural Network has become popular recently and many more papers on the subject have been published.

- Nielsen, Michael. Neural Networks and Deep Learning
  - http://neuralnetworksanddeeplearning.com
  - More of a tutorial than research. This is an online book explaining neural networks and solving a common classification problem.

- Lecunn, Y., Bottou, L., Bengio, Y., Haffner, P. (1998) Gradient Based Learning Applied to Document Recognition
  - http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
  - One of the first papers demonstrating the idea of Convolutional Neural Networks
  - At the time, computers lacked the computation needed to train these networks. Thus, they did not become popular until around a decade later.

- Krizhevsky, A., Sutskever, I., Hinton, G. E., (2012) ImageNet Classification with Deep Convolutional Neural Networks
  - http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
  - Recent paper on image classification using convolutional neural networks (CNN)
  - The first CNN to win the ImageNet Large-Scale Visual Recognition Challenge

- Simonyan, K., Zisserman, A. (2015) Very Deep Convolutional Networks for Large-Scale Image Recognition
  - https://arxiv.org/pdf/1409.1556v6.pdf
  - This deep network, also known as VGG, was designed such that it was simpler and deeper than previous models. This model also changed some of the hyperparameters specific to CNN's such as the kernel size and stride. VGG altered these parameters such that more information was processed for each image. The downside to this approach is training becomes much slower.

The above references are incredibly advantageous as they will be able to provide the team a theoretical understanding of neural networks and machine learning, since this is a new topic for the team. While it is mostly advantageous, a shortcoming that they do have is the level of abstraction in terms of their discussion on machine learning/neural networks. As the team is still

new to the concept of machine learning, certain advanced algorithms for very specialized neural networks and such should be considered carefully as to not get bogged down with details.

3.2 PROPOSED SYSTEM BLOCK DIAGRAM

Figure 1 below is a very high-level overview of our system. It is read in top-down fashion, beginning with input from a visual feed sourced from a drone camera attached to our embedded board. The visual feed is then pre-processed using OpenCV by utilizing suitable blurring and smoothing techniques, upon which feature extraction is performed in a way that is compliant with our object analyzer or classifier. This next stage of the pipeline is the trickiest part since object detection is an active research area overlapping fields such as machine learning or computer vision. Here our neural network will analyze the extracted features and provide data about any identified objects in the image. Initially, this will constitute of just the name of the object. Depending on success rate, we can choose to add additional descriptors such as color or type of object.

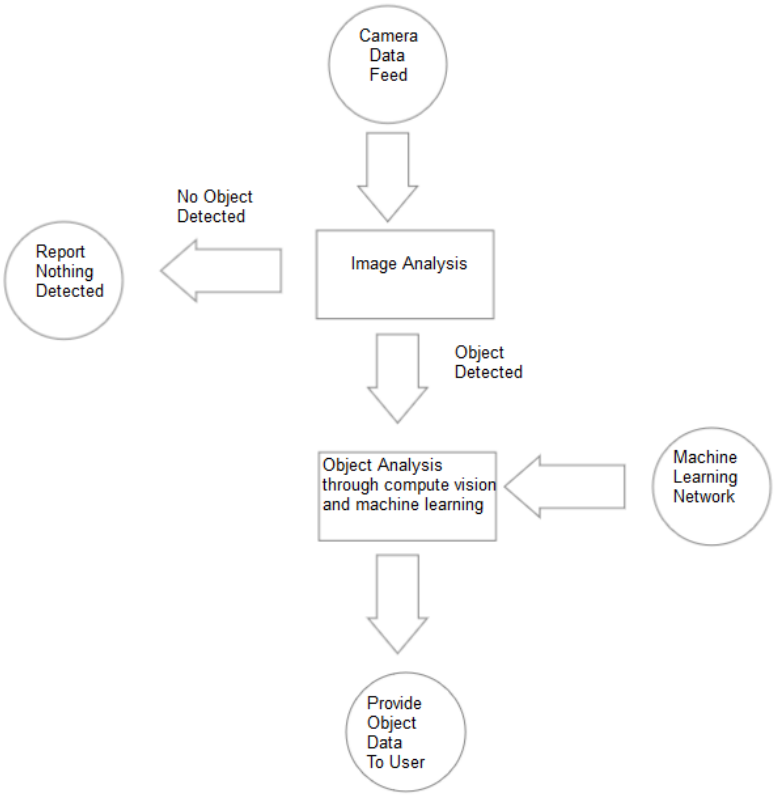

Figure 1: A.L.V.I.N.N Block Diagram

The design for the machine learning network part of our project is broken down into four phases: Preprocessing, Learning, Evaluation, and Prediction (see Figure 2 below). In the Preprocessing phase, raw images and streams of video data will be sent into our image processing unit for filtering and detection of features from which we can create data sets for training and testing. From the training data sets, the information is transferred to the second phase of the project, Learning. In this phase, the algorithm will take the incoming labeled data and use machine

learning to train a model used to make predictions on the test data. With the model trained, we move to the Evaluation phase. In this phase, we evaluate how well the trained model performs on the test data. If the evaluation is satisfactory we move on to its final phase, Prediction. Using the neural network, our algorithm will then predict on what it thinks it sees in new unseen data. We can then inform the user about any found objects and their relative associated information.

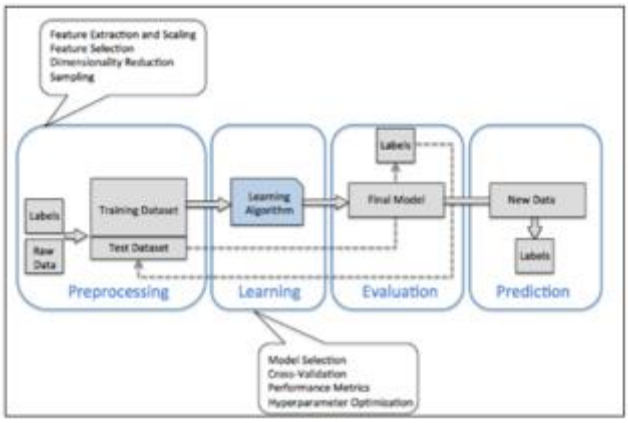

Figure 2 Machine Learning Process Diagram

3.3 ASSESSMENT OF PROPOSED METHODS

3.3.1 HARDWARE

The NVIDIA boards that we will be using can be purchased as a kit incorporating I/O, HDMI, USB, WI-FI, and a camera interface. NVIDIA also offers CUDA, an application programming interface, to deploy the neural network architecture onto the GPU. Through these predefined interfaces, we should be able to do everything required for this project. Other possible interfaces, outside the scope of this project, would include the onboard GPS and autopilot system Rockwell has running on the drone.

3.3.2 SOFTWARE

Python was the programming language of choice for our libraries because more team members have used it than C++. For those team members who have not used either, Python is said to be the easier of the two to learn.

MATLAB is a programming environment and language all in one. Used by engineers and scientist from visualizing data to develop and running algorithms in the areas of robotics, signal processing, and image processing. MATLAB presents itself as a viable option because it has toolboxes for computer vision, machine learning, and neural networks.

OpenCV is an open source computer vision library that can be used with many of today's popular programming languages. The functions in OpenCV are specifically written towards computer

vision with focus on areas such as 2D and 3D features, motion tracking, learning, and artificial neural networks.

OpenCV focus is computer vision thus it has more functions to work with for this project. OpenCV is open source versus a hefty price tag for MATLAB. Although we have access to MATLAB, outside of school this may not be an option for us to continue with what we have learned. The final factor in choosing OpenCV is it will work on our hardware where MATLAB wont.

TensorFlow is another open source library which focuses more on machine learning and neural networks. It is only available for C++ or Python programming languages and we will be using Python. There is already an interface to use TensorFlow with the NVIDIA TK1 board that we will be using.

Café is a framework that can be used for machine learning and neural networks. After consulting different popular technologies and frameworks, we decided to use the Caffe deep learning framework thanks to its expressive architecture, extensible code base, and speed optimizations. Furthermore, another important reason for why we chose Caffe is that other popular frameworks such as Google's Tensorflow are simply not supported by the Nvidia Jetson TX1 embedded board we plan to deploy our trained model to.

3.3.3 EXPERIMENTS

3.3.3.a NEURAL NETWORKS

With the application revolving around neural networks and computer vision, focus for the project was split between learning about neural network frameworks such as TensorFlow or Caffe, as well as learning how computer vision works with the OpenCV framework. So far, we have practiced using neural network frameworks and OpenCV regarding handwriting analysis and facial recognition respectively. However, this consisted of purely experimenting with pre-existing code found. The next steps in the project involve detection of images with cluttered backgrounds, and being able to send OpenCV data to our neural network to be able to perform object detection quickly.

Our first tests with neural networks was with the Caffe framework. We started with a classic neural network vision problem which was classifying handwritten digits given by the MNIST data set. See *Figure 3* for an example of the images used for handwritten digits.

Figure 3 MNIST Sample Images

A typical naive approach for this specific problem that we used to model our neural network looked something like *Figure 4*.

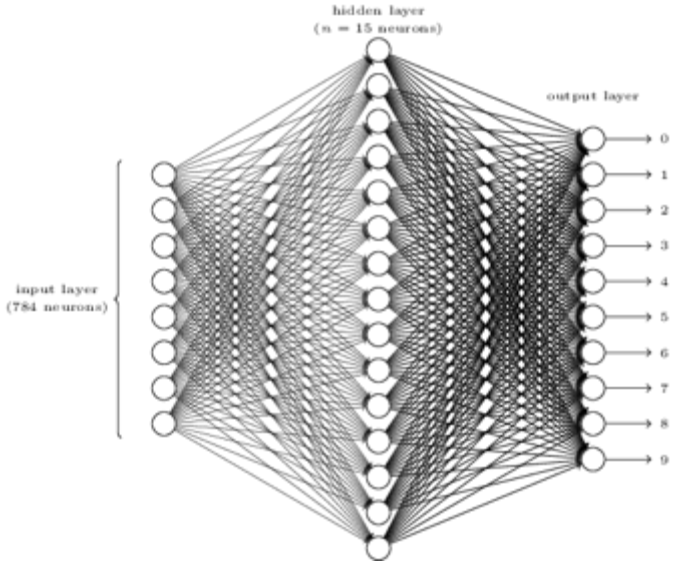

Figure 4 Naive MNIST Image Classifier

The neural network above depicts an input layer of 784 neurons where each neuron corresponds to a pixel in the images of handwritten digits (28 x 28 pixel images). A hidden layer(s) essentially compresses and generalizes the information from the input neurons. Finally, each neuron in the output layer outputs the confidence that input image matches its class (numbers 0 through 9).

After training this network with the MNIST dataset we could correctly classify digits 93% of the time. While this may sound impressive, it is very bad when comparing our results to other solutions that solve the same problem. This approach for processing images is naive for the following reasons.

1. Only very low resolution images can be evaluated (28 x 28 pixels)
2. Can only correctly classify digits that are in the same relative position as the training images (cannot classify digits in different locations)
3. Has trouble classifying rotated digits
4. Poor accuracy when compared to better solutions
5. This type of model becomes infeasible when processing higher resolution images with millions of pixels
6. Potentially many more issues we are not aware of now

After finishing our work with traditional feed-forward neural networks and understanding the limitations they suffer from, we will adopt Convolutional Neural Networks which specialize in image classification. Some of the major problems with the previous design was that as our images became more complex, the network size increases exponentially. Most importantly, when a feed forward neural network processes an image, it has no sense of spatial locality, it sees the image in one dimension when it really is a two-dimensional input. CNNs solve both major problems by

introducing image convolution with kernels and by subsampling images down further. Here is an example of a simple CNN that also classifies MNIST handwritten digits:

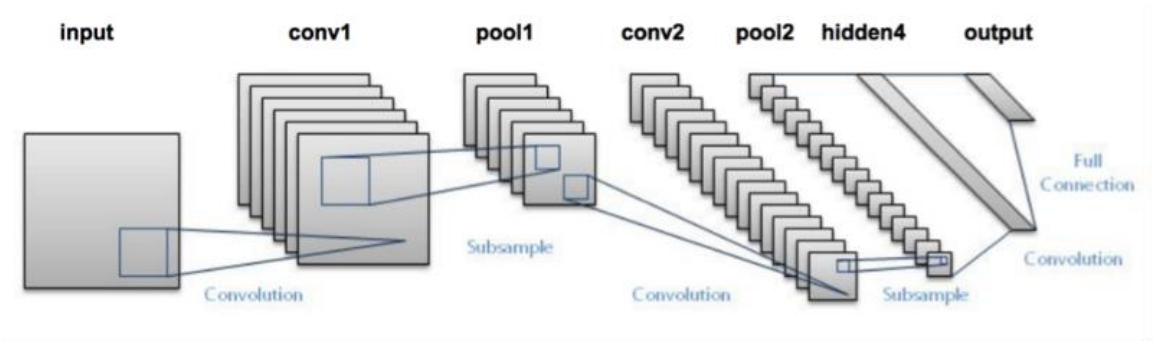

Figure 5 Lenet Convolutional Neural Network

The first step of Lenet is convoluting the input image with 6 different trained filters. These filters extract 6 different feature maps that are used in subsequent layers of the network. The input images are of size 32 x 32 pixels while the 6 filters are of size 5 x 5. Each kernel will pass over the entire image while convoluting subsections of the given image. Here is a quick example of a kernel with its trained weights over a specific section of the input image.

$$Kernel\,1=\begin{bmatrix}0&0&4&0&0\\0&0&4&0&0\\0&0&5&0&0\\0&0&4&0&0\\0&0&4&0&0\end{bmatrix}\quad Subsection\ of\ image=\begin{bmatrix}0&34&191&0&0\\0&25&196&0&0\\0&20&192&0&0\\0&0&195&10&0\\0&0&180&5&0\end{bmatrix}$$

Figure 6 Kernel with Trained Weights

Convoluting these two matrices gives a result of 801(a high positive result means the image section is like a straight vertical line). Passing this kernel over an entire image extracts the image's feature map for this specific kernel which looks for vertical lines. The resulting feature map is compressed into a smaller feature map using max pooling.

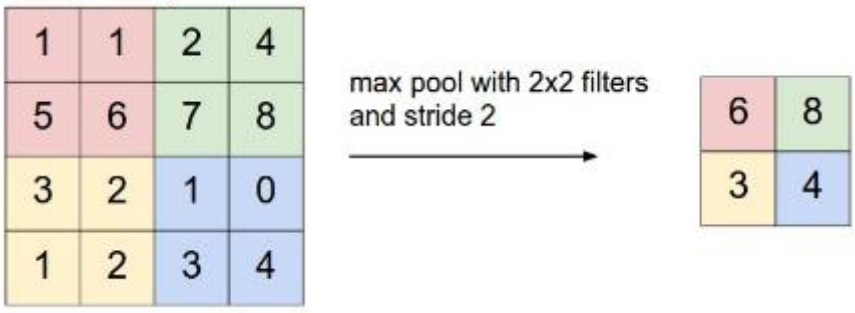

Figure 7 Max Pooling

These steps are repeated until the processed data is passed into a basic fully connected layer like the Caffe MNIST example above.

These reasons are precisely why we are using tools such as OpenCV to pre-process images before they are fed into our network. For example, we could find areas of interest in a high-resolution image to reduce the number of pixels that the network needs to process, effectively reducing the resolution of an image. Additionally, we plan to consider Convolutional Neural Networks which specialize in image classification.

3.3.3.b COMPUTER VISION

Furthermore, we explored another basic object recognition use-case using the OpenCV library. Specifically, we investigated how to perform detection of faces using OpenCV's Face Recognizer module. We began by feeding simple images containing faces to our program, and converting these into grayscale. Following up on that, we extended our program with the capability to distinguish between faces belonging to different people (by constructing a labelled training dataset). Upon consolidation of this feature, we then went on by feeding a video stream from the webcam to our facial recognition software, and visually drawing a square green box around recognized faces, as well as displaying a name on the command-line, should a familiar face have been encountered. *Figure 6* shows an example of someone being recognized.

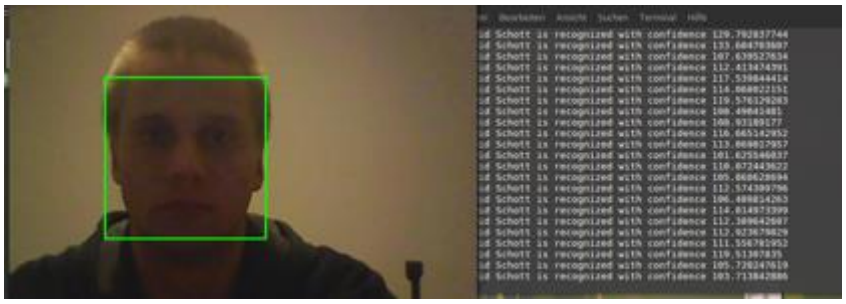

Figure 8 Facial Recognition of David Schott

3.4 VALIDATION

Development and testing will both be done using built-in tools that OpenCV and Caffe provides. These are a starting point and as we move along in the project we may discover these may not work or there are better alternatives.

For the variety of tasks that the application should be able to perform, there will be a series of test suites to be able to validate whether they adhere to the client's specification. The following is a listing of a few validation tests to be used for the application:

- Provide the application with a series of images of aircraft/other objects with a cluttered background and test whether the system can reliably detect the target image. A low error

rate for many images is the goal along with a high confidence level for each classified object in an image.

- Provide the application with a video of an object moving with a cluttered background behind it and test whether the system can reliably detect the moving object.
- Automated cross-validation techniques with labelled images to obtain an averaged model prediction accuracy.

For the above validation tests and many more, the success of the tests will be based on how consistently accurate the application can detect an object (either moving or stationary) through a large amount of trials (preferably in an automated way).

Additional software that may be used for testing and demonstration purposes include a flight simulator and a GUI. A flight simulator will allow us to have a controlled stream of data into the system. A GUI could be used to display system results such as identification of detected objects or prediction accuracy.

Additional hardware for this project would be in the form of a camera. A camera could be used to import either still pictures or to stream video into the system. With the numerous photos, available on the internet and the possible use of a flight simulator, we may not need to incorporate an actual camera.

4 PROJECT REQUIREMENTS/SPECIFICATIONS

4.1 FUNCTIONAL

The following list includes the functional requirements for the project:

- Drone/System must be able to process a single picture, string of pictures, and/or continuously moving video.
- Drone must be able to detect stationary objects apart from the background.
- Drone must be able to detect moving objects apart from the background (and possibly other objects).

4.2 NON-FUNCTIONAL

The following list includes the non-functional requirements for the project:

- Time nonfunctional requirement for data processing; i.e. we want to run within a certain time limit such that it doesn't take 5 minutes to process an image and figure out an image was spotted 5 minutes ago,
- Space/Memory nonfunctional requirement. Since the client would like to have this application run on a drone which is an embedded system with limited memory/storage, a non-functional requirement would be to be able to implement this without any large memory/space requirements.
- Output information format. I.e. what is the format in which an end user will see the data from our application. Thus, a non-functional requirement would be to have the data be outputted in an elastic way such that future users could easily integrate and use the data in other systems.
- Arguably, our bonus features could be considered as non-functional potentially. This is hard to say based on lack of knowledge on scale of bonus features but for example it could be argued that having a GUI for example would be a non-functional requirement.
- It should be able to identify certain objects within a certain threshold of accuracy. I.e. we may say that we want to have the application detect an airplane 80 percent of the time for example.

4.3 STANDARDS

When working on this project, the team will be adhering to several standards. First, all code that will be written will standardized and documented, making it easy to maintain, understand, and debug if necessary. Some IEEE standards which will be more applicable than others would be the IEEE floating point standardization, test methodology standardization, as well as code standardization. For testing and coding, these standards are incredibly important. Since our application will be using a lot of floating point precision and arithmetic, it is important that the IEEE standard for floating points is adhered to, such that it will be able to be transferred to the client's system easier. In the project, there will not be any practices which would be considered as unethical by organizations such as IEEE or ABET.

5 CHALLENGES

The immediately observable challenges we face in this project are knowledge or theory related. The machine learning field is unfamiliar territory to most of us, since ISU does not offer its machine learning class as an elective to computer engineers. The vastness of this field and the open-endedness of our client's expectations means we need to be careful before investing a lot of time in a technique or method that may not be scalable. Expanding on this, we also need to be aware of potential pitfalls since there is no "one-size-fits-all" approach to an image recognition problem. Consequently, clear and consistent communication with our project stakeholders (who are more knowledgeable on the machine learning field) is of utmost importance.

Another issue we are challenged by (particularly in the early planning stage) is organization. Our project consists of fifteen people total; two teams of six, two advisers, and one client. We have four different mailing lists and already experienced that our advisers may have conflicting expectations or visions.

6 TIMELINE

The timeline below shows the goal of the project. The phases listed correspond to the phases listed in *Section 2: Deliverables*. The first semester is heavily focused on education, choosing hardware to use and determining which software interfaces well with the hardware to accomplish the goals set forth for the project. The second semester is heavily focused on producing the deliverables outlined within each phase of the project.

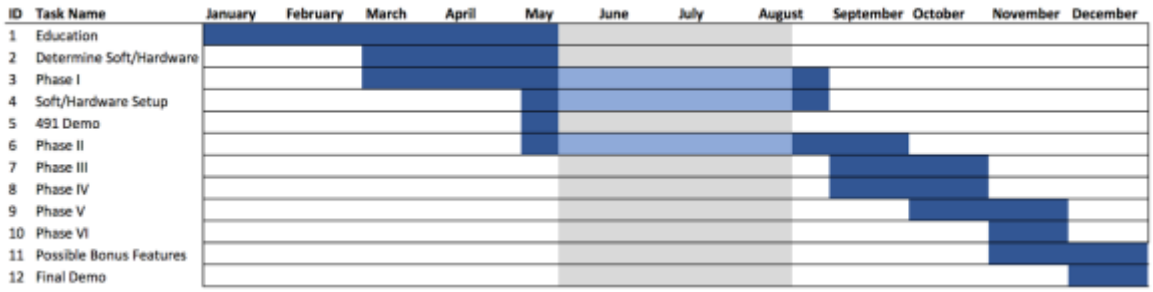

Figure 9: Proposed Timeline

6.1 FIRST SEMESTER

The proposed project breakdown for the first semester of senior design can be seen below. Most this semester will be spent on education on the topic, on the chosen hardware, and on software that will integrate with the hardware. Various phases may be lightly touched on through education but no serious programming will be accomplished until the second semester.

Week Beginning/Ending			Tasks	Assignment	Deliverables
February	13	19	Education	All Members	
	20	26			
March	27	5			
	6	12	Begin Setup & Coding	All Members	Design Document 1
	13	19	Begin Phase I	All Members	[Spring Break]
	20	26			
	27	2			Project Plan 2
April	3	9			
	10	16			
	17	23			Design Document 2 & Project Plan 3
	24	30			491 Demo
May	1	7	Begin Phase II		

Figure 10: First Semester Project Breakdown

6.2 SECOND SEMESTER

The proposed project breakdown for the second semester of senior design can be seen below. This semester is focused on using the education from first semester and applying it towards programming the software and hardware to produce the desired deliverables. Assignments for phases will be broken up and assigned to specific team members once the education phase from the first semester is complete. This will allow the team to judge the strengths of each member more accurately while assigning tasks. This will also allow the team to work on multiple phases at

once and extend the amount of time needed one a phase, if necessary. Bonus features will be attempted only if there is adequate time at the end of the semester; otherwise, these features will be included to move forward with the project in the future.

Week Beginning/Ending			Tasks	Assignment	Deliverables
August	21	27	Continue Phase II		Phase I
September	28	3	Begin Phase III		
	4	10			Phase II
	11	17			Phase III
	18	24	Begin Phase IV		
	25	1			
October	2	8			Phase IV
	9	15	Begin Phase V		
	16	22			
	23	29			Phase V
November	30	5	Begin Phase VI		
	6	12	Begin Prearing for Demo	All Members	
	13	19	Possible Bonus Features		Phase VI
	20	26	Final Demo Preparation	All Members	[Thanksgiving Break]
	27	3			492 Final Demo
December	4	10			

Figure 11: Second Semester Project Breakdown

7 CONCLUSIONS

The primary objective of our project is to appreciate what it is like to work on a larger project with a multidisciplinary engineering team. Although it may be advisable to “under promise” and then “over deliver” to the client, we believe setting aggressive goals for such an exciting project will be beneficial and a self-fulfilling prophecy (higher morale ultimately yielding higher productivity). With that in mind, we still want our goals to be realistic and we are actively working with our advisers to discuss the feasibility of phases III and beyond in more detail. This can be displayed with some of the scaling limitations of the methods of approach discussed in the previous section.

Considering that estimation work (at best) is required for a visual object recognition task we are trying to approach this project from a research perspective. Thus, to optimally achieve our goals, we will be in close contact with our advisers and the client to evaluate what steps are likely to lead to success. Time will play a pivotal role in the success of our project; therefore, we will try to run heavily computation tasks overnight using remote machines with specialized performance-critical hardware. Our goal is to complete phases I and II before the summer and then use the remaining time before the next semester to have the next set of goals pinned down for the next phases.

We started our project by deciding what software was available to us that would facilitate incorporating computer vision and machine learning into our project. Using the background experience from team members, previously coded examples, advisor expertise, and tutorials, the team has been able to work with the different software to see how it performed to make a final decision on what software to use. With this decision, out of the way, we can start working on the design process to reach our goals of detecting landing strips or other flying objects. The process we will be following consist of preprocessing images, machine learning through training, evaluating and evolving our model, and finally prediction.

By utilizing OpenCV for image processing and Caffe for machine learning we can begin to build a model that we will be able to train, test, and then use to predict various objects. OpenCV and Caffe have been used in numerous computer vision projects like the goal of our project, thus there is access to a plethora of information. Likewise, NVIDIA provides an interface for the chosen hardware, OpenCV, and Caffe. Using proven technologies will allow us time to concentrate on adapting the technologies to our project goals instead of spending time figuring out how to get the various technologies to communicate with each other.

8 REFERENCES

Listed below are references that have been used so far as part of the research and testing phases.

"Cascade Classifier Training." Cascade Classifier Training — OpenCV 2.4.13.2 Documentation. OpenCV Dev Team, 16 Dec. 2016. Web. 31 Mar. 2017. <http://docs.opencv.org/2.4.13.2/doc/user_guide/ug_traincascade.html>.

"Computer Technology Standards." IEEE Standards Association. IEEE, 2017. Web. 31 Mar. 2017. <http://standards.ieee.org/cgi-bin/lp_index?status=active&%3Bpg=40&%3Btype=standard&%3Bcoll=15>.

Deshpande, Adit. "A Beginner's Guide to Understanding Convolutional Neural Networks." A Beginner's Guide to Understanding Convolutional Neural Networks – Adit Deshpande – CS Undergrad at UCLA ('19). N.p., 20 July 2016. Web. 31 Mar. 2017. <https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/>.

Geitgey, Adam. "Machine Learning Is Fun! Part 4: Modern Face Recognition with Deep Learning." Medium. Medium Corporation, 24 July 2016. Web. 31 Mar. 2017. <https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78#.7732t9p7a>.

Mallick, Satya. "Home." Learn OpenCV. Big Vision LLC, 14 Nov. 2016. Web. 31 Mar. 2017. <http://www.learnopencv.com/image-recognition-and-object-detection-part1/>.

Nielsen, Michael A. "Neural Networks and Deep Learning." Neural Networks and Deep Learning. Determination Press, 01 Jan. 1970. Web. 31 Mar. 2017. <http://neuralnetworksanddeeplearning.com/>.

Ng, Andrew. "Machine Learning." Machine Learning | The Open Academy. Open Academy, n.d. Web. 31 Mar. 2017. <http://theopenacademy.com/content/machine-learning>. Ng, Andrew. "Machine Learning." Machine Learning | The Open Academy. Open Academy, n.d. Web. 31 Mar. 2017. <http://theopenacademy.com/content/machine-learning>.

Qureshi, Shehrzad. "Computer Vision Acceleration Using GPUs." Computer Vision Acceleration Using GPUs (n.d.): n. pag. AMD Developer. AMD, June 2011. Web. 31 Mar. 2017. <http://developer.amd.com/wordpress/media/2013/06/2162_final.pdf>.

Shimodaira, Hiroshi. "Learning and Data Note." Multi-Layer Neural Networks (2015): n. pag. Web. <https://491dec1709.slack.com/files/daschott/F46274YA1/learning.zip>.

Smith, Steven W. "Nueral Networks (and More!)." The Scientist and Engineer's Guide to Digital Signal Processing. San Diego, CA: California Technical Publ., 1999. 451-80. Print.