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

DESIGN DOCUMENT

Dec1709 – Autonomous Learning Vehicle Integrating Neural Network

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

Revised: 3/10/2017
Version 1

Contents

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. This will require the team to identify the software and hardware requirements capable of performing these tasks. Computer vision tools can then be used to detect objects within a picture or video stream. Machine learning and neural networks will then be incorporated to perform teaching operations for identifying different objects or distinguishing between similar objects. Furthermore, part of the project will consist of determining which machine learning algorithms are particularly suitable for our object recognition task. 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 (such as a landing strip) or objects in the air, such as 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. Teaching the system to distinguish between objects would be the next immediate goal. With these goals achieved we can work towards our final goal of implementing all of this with our own neural network.

If time allows, we could then also look at:

- Detecting landing strips versus other paved features such as large parking lots or roadways.
- Detecting another drone which could lead to detecting its movements and setting autopilot to follow it.
- Auto Course correction when detecting objects when flying to avoid collisions.

- Object identification confidence, projecting the system's confidence on what it perceives an identified object to be.

2 Deliverables

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

- Phase I: Detect 'X' in an image
- Phase II: Detect 'X' with background noise in an image
- Phase III:
    - Detect multiple objects within an image, possibly incorporating GPS or Google Maps to help facilitate object detection based on what stationary objects/landmarks may be in the area (airport, lake, specific buildings, etc.).
    - Identify objects even if objects have several different appearances.
- Phase IV: After detecting an object, report to control possible object identifications with associated confidence levels.
- Phase V: Identify an object that may present itself in several different positions
- 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
- Phase VII: Tracking movements of identified objects
- Phase VIII: Beyond tracking movements, track fellow system in air and follow the system via autopilot.
- Possible Bonus Features
    - Course Correction: Identify objects in the system's path and communicate with control. Control can advise whether to continue the pre-selected course or course correct. Possibly incorporate GPS or Google Maps to help facilitate this feature.
    - 360º 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.
    - Feature Recognition: Ability to recognize various features present on objects (colors, symbols, unique traits, etc.)
    - Thermal Imaging or Night Vision: Object detection and identification through thermal imaging or night vision cameras.

3 Design

3.1 SYSTEM SPECIFICATIONS

Design, train, and deploy deep neural network based embedded board for use as a customizable Unmanned Aircraft System (UAS) mission processor. The project will use open-source frameworks and NVIDIA's latest GPU-accelerated deep learning platform to identify objects and fuse sensor data from one or more cameras and potentially other sensors, such as LIDAR.

Initially, use NVIDIA tutorials to implement some type of visual object recognition or similar task and use a mockup environment in the lab. (Toy airplanes on sticks, etc.) As confidence grows, Rockwell Collins can potentially provide some video footage from one of our drone flights this spring or summer. The specific objective of the visual system can be negotiated; examples might be runway detection, detection of another nearby aircraft, etc.

3.1.1 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 req.
- 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.

3.1.2 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).

3.1.3 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.

3.2 PROPOSED DESIGN/METHOD

The design for this project is broken down into four phases: *Preprocessing, Learning, Evaluation,* and *Prediction* (see *Figure 1* 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.

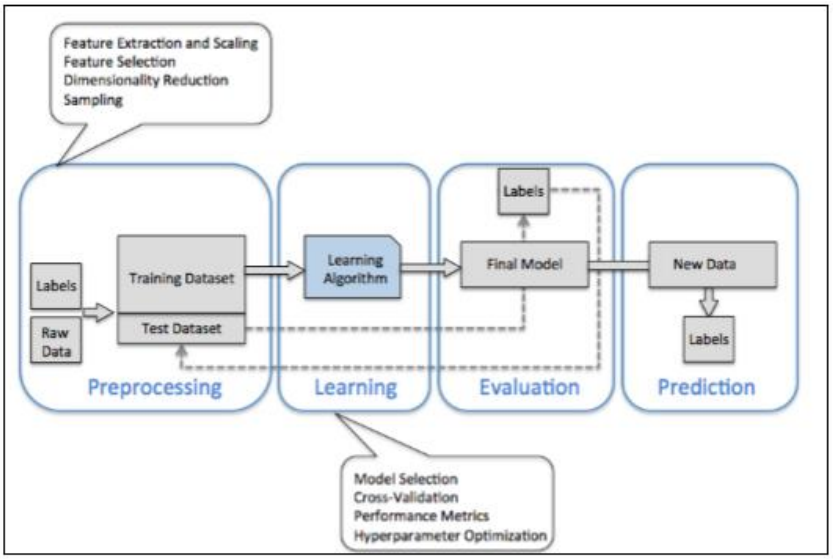

Figure 1: Design Process Diagram

3.3 DESIGN ANALYSIS

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, 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 TensorFlow to be able to do object detection quickly using neural networks.

Up to this point, TensorFlow appears to be a great choice to model, train, and run a neural network for the following reasons.

1. TensorFlow gives us the ability to model complex neural networks as computation graphs which define how data flows from one neuron to the next.
2. We can easily define how our input and output data is structured
3. Most importantly, TensorFlow is a highly-optimized library for training and running neural networks on GPUs or chips with CUDA compatibility

We started with a classic neural network vision problem which was classifying handwritten digits given by the MNIST data set. See *Figure 2* for an example of the images used for handwritten digits.

Figure 2: MNIST sample images

A typical naive approach for this specific problem that we used to model our neural network looked something like *Figure 3*.

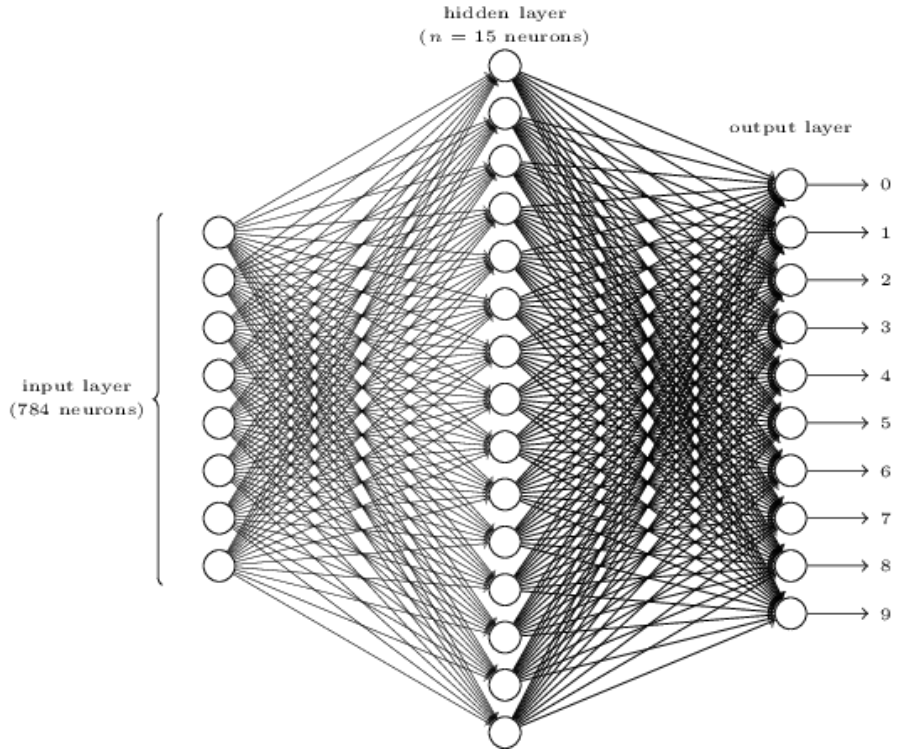

Figure 3: Naive MNIST image classifier

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

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.

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 FaceRecognizer 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 4* shows an example of someone being recognized.

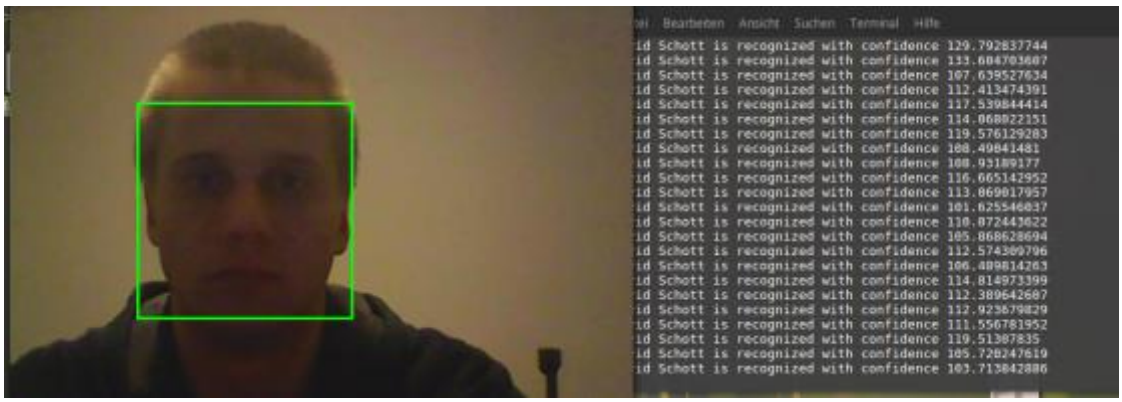

Figure 4: Facial recognition of David Schott

The code and documentation of our command-line interface for adding new faces (taking screenshots using a webcam), training the recognizer, and launching it has been uploaded to our group's GitLab repository.

4 Testing/Development

4.1 INTERFACE SPECIFICATIONS

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

4.2 HARDWARE/SOFTWARE

Development and testing will both be done using the software mentioned earlier. These include MATLAB, OpenCV, and TensorFlow. We will be using Python as the programming language to work with the OpenCV and TensorFlow libraries. 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.

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.

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 boards that we will be using.

Python was the programming language of choice for OpenCV and TensorFlow 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.

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.

The focus for hardware has been towards NVIDIA's TK1 and TX1 boards, as requested by the client. These boards are designed and built around machine learning and neural

networks. Although the GPU is the focus, the boards can be purchased as a kit with I/O, WI-FI, and a camera interface.

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.

4.3 PROCESS

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.
- 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.

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

5 Results

As of right now, the project has not reached a testing phase and has no significant results. As mentioned in Section 3.3, there have been promising results with the classic neural network vision problem which was classifying handwritten digits given by the MNIST data set and with performing detection of faces using OpenCV's FaceRecognizer module. More investigation into both tasks is needed before we can determine how these technologies can help us with our project.

6 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 TensorFlow 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 TensorFlow 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 TensorFlow. 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.

7 References

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

Machine Learning

- http://diydrones.com/profiles/blogs/using-tensorflow-to-allow-a-robot-to-identify-objects
- http://theopenacademy.com/content/machine-learning
- https://491dec1709.slack.com/files/daschott/F46274YA1/learning.zip
- Good lightweight article to read for not treating machine learning like magic:
    - http://www.pyimagesearch.com/2014/06/09/get-deep-learning-bandwagon-get-perspective/

Neural Networks

- Chapter 26 of the free online book. Good overview of using the simplest of Neural Networks. One can download a pdf of each chapter.
    - http://www.dspguide.com/pdfbook.htm
- Online book covering the basics of neural networks with a simple computer vision problem.
    - http://neuralnetworksanddeeplearning.com
- A very basic overview of the concept behind Convolutional Neural Networks and why they are used to classify images.
    - https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks

Computer Vision

- Slide 2 and 3 of the attached PDF gives a high-level overview of that Traditional Computer Vision pipe-line that is heavy on the Feature Extraction stage and light on the pre-processing and Classification stage.
    - https://491dec1709.slack.com/files/tlavan/F49KXLFH8/rcl_021317_murad.pdf
- Somewhat light article that shows a nice example of combining “Traditional” with Deep Learning Computer vision techies. (cool demo near the end: using the Jimmy Fallon show as an input))
    - https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78#.7732t9p7a
- Slide 9: gives a nice overview of a generic classical Computer Vision pipeline
    - http://developer.amd.com/wordpress/media/2013/06/2162_final.pdf
- Overview of traditional approaches for Computer Vision:
    - http://www.learnopencv.com/image-recognition-and-object-detection-part1/
- Near the end of this slide set, gives the topics covered which does a reasonable job of covering topics useful for doing computer vision.
    - http://www.cs.cmu.edu/~16385/spring15/lectures/Lecture1.pdf

Technologies

- TensorFlow Video
  - https://www.youtube.com/watch?v=APmF6qE3Vjc
- Intro to Matlab
  - https://491dec1709.slack.com/files/tlavan/F4663EHSR/ee225_matlabintro_2015.pdf
- OpenCV
  - http://www.learnopencv.com/

8 Appendices

Below are supporting documents and links associated with the project.

PowerPoints

- Our “marketing pitch” for our project.
  - ALVINN_Pitch
- Our first approach at what we think our component structure will look like
  - ALVINN_Component_Structure

TX1 and TK1 Boards (Specs & Info):

- http://www.nvidia.com/object/jetson-tx1-module.html
- http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html