NORTHWEST NAZARENE UNIVERSITY

Autonomous Drone Project

THESIS

Submitted to the Department of Mathematics and Computer Science

in partial fulfillment of the requirements

for the degree of

BACHELOR OF SCIENCE

Casey Lewis

2018

THESIS

Submitted to the Department of Mathematics and Computer Science

in partial fulfillment of the requirements

for the degree of

BACHELOR OF SCIENCE

Casey Lewis

2018

Autonomous Drone Project

Author: _____

Casey Lewis

Approved: _____

Dale Hamilton Ph.D., Department of Mathematics and Computer Science, Faculty Advisor

Approved: _____

Stephen Riley Ph.D., School of Theology & Christian Ministry, Second Reader

Approved: _____

Barry L. Myers, Ph.D., Chair, Department of Mathematics & Computer Science

ABSTRACT

Small Unmanned Aircraft System Classifying Linear Features Mid-Flight with a Mobile Device.

LEWIS, CASEY (Department of Mathematics and Computer Science), HAMILTON, DR. DALE (Department of Mathematics and Computer Science)

The use of small Unmanned Aircraft Systems (sUAS) in real world applications are rising. This is most likely due to the ability of an sUAS to access areas quicker, safer and easier than a human. This rising demand of the sUAS therefore promotes opportunity for computer scientists to create programs that can help organizations complete tasks with less effort than before. This project was created to help complete the task of flying. Specifically, this project focused on whether an sUAS could autonomously fly a road in a forested environment. At first, the need to answer if a computer system could identify a road was paramount. Next, since mobile devices are used to control the sUAS, the ability to classify an image while maintaining control of the sUAS had to be determined. Finally, image classification is simply not enough, the output of the classification of the image must have such information as to help ascertain direction. The results are promising. The sUAS can classify images while providing continuous control of the sUAS to the user and the images are classified in such a way as to provide adequate information in determining direction. Future work includes having the sUAS decide direction.

ACKNOWLEDGEMENTS

Without the support and dedication of my wife Jennifer I would not have made it through college, let alone this project. She has always been and will continue to be the love of my life. I also want to thank Dale Hamilton Ph.D. and Jason Colwell Ph.D. for their continued help in problem solving and bringing unique concepts to this project.

Table of Contents:

Title Page i

Signature Page ii

Abstract iii

Acknowledgements iv

Table of Contents v

List of Figures vii

Introduction

- Background 1

Body

- Classification 2
  - Using SVM 3
  - Using CNN 4
- Design and Implementation 8
  - Turi Create 12
- Direction 14

Conclusion

- Future Work 18
- Conclusion 18

References 21

Appendices 24

- A. Code 24
  - A.1 CNN 24
  - A.2 Split and Classify 33
  - A.3 App Home View 38
  - A.4 sUAS Control View 41
  - A.5 Retry Manager 48

A.6 Mobile Split 49
A.7 Mobile Classify 51
A.8 Turi Train 53
A.9 Turi Auto Pick 55
A.10 Naïve Bayes CPP 58
A.11 Naïve Bayes Classifier 66
A.12 GS Method 69
B. TensorFlow Object Detection API Documentation 73
C. Pix4D Data 81
D. Results 92
D.1 Turi Create CNN 92
D.2 Turi Create Other 108
D.3 Naïve Bayesian 119

List of Figures

Figure 1. Support Vector Machine (Hamilton et al., 2018) 3

Figure 2. Road images classified SVM 4

Figure 3. Convolution (Drawing a convolution with Tikz) 5

Figure 4. Max Pooling (Max-pooling, 2018) 5

Figure 5. Convolutional Neural Network (Google Developer, 2018) 6

Figure 6. TensorFlow Object Detection API results 7

Figure 7. Image – Grid representation 8

Figure 8. Application Home View 10

Figure 9. sUAS Control View 10

Figure 10. Confusion Matrix for Road Classification 14

Figure 11. Straight (a) Ask User (b) Left (c) Right (d) 15

Figure 12. Direction Labels 16

Figure 13. Directional Classifiers 16

Background

Using technology to simplify our lives is not a new idea. This is seen from inventions such as the wheel, the printing press and Ford's assembly line. While not all inventions change the world like that of the printing press or the assembly line, many smaller technological advances have made tasks easier and more efficient. The Autonomous Drone project was created to make flying easier and more efficient. As flying can be technical. The hoped result of a more efficient flight is a greater preservation of battery life and thus longer flight time.

The idea for the Autonomous Drone (AD) project came from NNU's Fire Monitoring and Assessment Platform (FireMAP) research team. Currently, FireMAP utilizes small unmanned aircraft systems (sUAS) in areas from postfire mapping to archaeology. While the AD project is designed to be applicable in both areas, the first version of the AD project is primarily concerned with archaeology. The goal for the sUAS in archaeology is to fly above an old rail grade, which is a railroad where the tracks and ties have been removed, and then fly an area for some specified distance on each side of the rail grade. The imagery will then be used to help locate artifacts that may have been left by past settlements that lived close to rail roads. The first version of the AD project should operate as follows:

1. Take an image at a height of 120 meters
2. Classify the image (looking for a road)
3. Determine direction
4. Travel that direction for 100 meters
5. Repeat 3 times then return home

To accomplish the feat of having an sUAS fly by itself is a major challenge. First, a way for a computer system to adequately classify a road, had to be found. In the context of the AD project, a road is a paved or dirt surface between eight to twelve feet in width with definitive edges. The reason a road needs to be classified instead of a rail grade is because a road is generally more defined than a rail grade and this will provide proof of concept. Second, any program created in the hopes of flying a sUAS needs to interact with the software controlling the sUAS. Third, after the computer system has identified the road, the system must then use that information to determine direction. Lastly, and most importantly, all of this had to be done from a mobile device which contains less computing power than a desktop.

Classification

Finding a way for a computer system to classify a road was the priority for the Autonomous Drone (AD) project. Fortunately, the FireMAP research team had faced a similar situation and used machine learning algorithms to extract data from images to generate information, which was what the AD project needed to do. Thus, using machine learning seemed the logical place to start.

Machine learning can be broadly separated into two categories: supervised and unsupervised learning. Supervised learning consists mainly of classification and numerical predictions while unsupervised learning has long been synonymous with clustering (Han, 2012) but has evolved into reinforcement learning and evolutionary computation. One of the main differences between supervised and unsupervised

learning is that in supervised learning, the data is labeled and there are known inputs and desired outputs. This means that when training a model, which is an algorithm that is constructed to predict class labels, the data going into the model while training is labeled. For example, to train a model to recognize images with roads, the input would be images of road labeled as road and images without a road in them labeled as not road.

Using SVM

The FireMAP team used supervised learning techniques to classify imagery of burned areas. This meant the team used algorithms to see if an image contained a burned area of land and to what extent that area was burned. If it worked for finding burned areas it seemed reasonable that the same algorithms would work for finding a road. To find burned areas and determine their extent, the FireMAP team used a support vector machine (SVM). An SVM "transforms training data into a higher dimension, where it finds a hyperplane that separates the data by class" (Han, 2012, p. 393), an illustration of the SVM is in Figure 1:

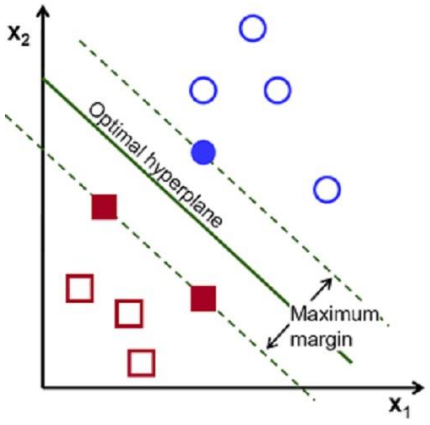


Figure 1. Support Vector Machine (Hamilton et al., 2018)

The FireMAP team had been using the SVM with great success in pixel-based classification of images to measure burn extent (Hamilton et al., 2018). Simply, pixel-based classification is looking at each pixel in an image and classifying those pixels. The AD project tried to do the same while looking for roads. Figure 2 shows a sample image of the results.

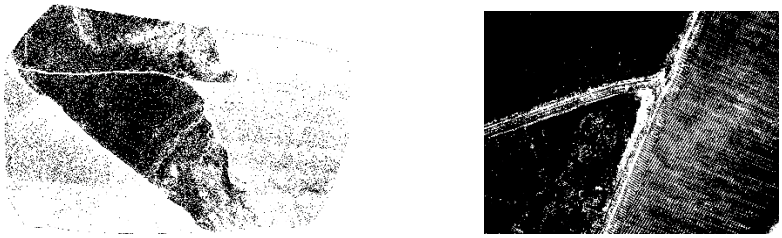


Figure 2. Road images classified SVM

The images in Figure 2 display the output of the SVM where the light areas are supposed to be the road and the dark areas are not road. While the SVM was able to classify the road correctly, the SVM also showed large areas of not road that it labeled as road. Clearly, the image of the road on the left side of Figure 2 would be very hard for a sUAS to follow as large areas of dirt are classified as road.

Using CNN

Since the SVM with pixel-based classification would not work for the AD project, the search for finding a road within an image continued. Seeing the success in machine learning algorithms with imagery, the choice to look at other classifiers seemed the best option. The next classifier studied was the Convolutional Neural Network (CNN) which works by convoluting an image. Convolution is the process of taking a feature map, such as an image, performing “element-wise multiplication” on a subset of that feature map,

and then returning a single value to be the input into a new feature map (Google Developers, 2018). This is shown Figure 3:

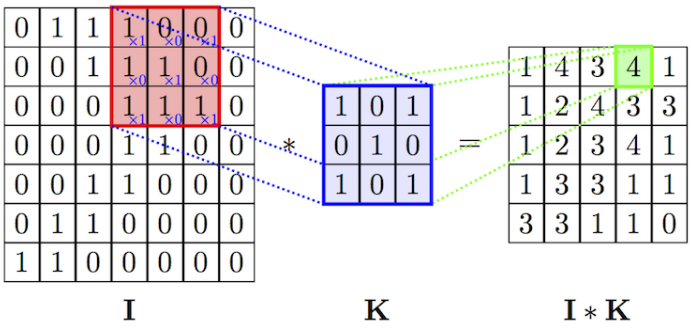


Figure 3. Convolution (Drawing a convolution with Tikz)

The amount of times a convolution can be performed, or the number of convolution layers a CNN has, is called its depth. After convolution, non-linearity is introduced to the model and then pooling is performed. Pooling is like convolution in that the feature map has an algorithm that is performed on a subset of itself to return a single value. Max pooling, a common algorithm used for pooling, retrieves the max number from the subset of the feature map and makes a new feature map. The main difference between convolution and max pooling, besides the algorithm used, is how the filter (the subset of the feature map) moves along the feature map (Google Developers, 2018). Max pooling is visually demonstrated in Figure 4:

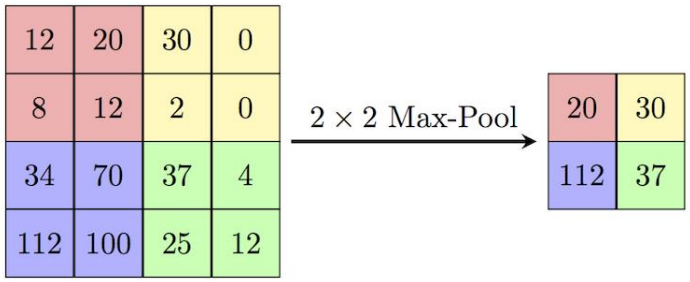


Figure 4. Max Pooling (Max-pooling, 2018)

After a series of convolution and pooling layers, the resulting feature map is used as an input layer for a fully connected neural network (See Figure 5).

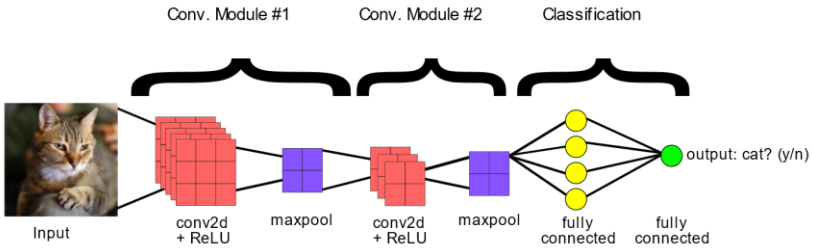


Figure 5. Convolutional Neural Network (Google Developer, 2018)

The first CNN used in the AD project was TensorFlow's CNN in which the number of convolutional layers and pooling layers are determined by the programmer (See Appendix A.1 for CNN code). After much time spent learning about best practices in training neural networks and applying that learning, such as selecting batch size, understanding gradient descent and how steps size (or learning rate) affect the neural networks ability to train and how quickly it does so, the CNN had tremendous success identifying images with roads in them. While being able to distinguish images with roads in them was a success, this, on its own, would not be enough to decide direction. At the time, knowing where the road was in an image seemed like the next step in telling a sUAS which direction to travel.

Finding an object in an image is a little different than determining if an object is in an image. Determining if an object is in an image is classification, it is a yes or no problem. Object detection, on the other hand, is classification and then also locating that object within the image. Fortunately, TensorFlow (TF) has an open source object

detection API (application program interface) for locating objects in an image. However, TF's object detection API is still fairly new and implementing it took a lot of work (See Appendix B.1 for documentation on how to install TF's object detection API within a python virtual environment in Windows 10). After much effort, the Autonomous Drone project was able to make TF's object detection API detect roads within an image and outline the road with a bounding box:

Figure 6. TensorFlow Object Detection API results

Unfortunately, there is a problem with using object detection. For example, the right image in Figure 6 would show the same bounding box if the road was coming up from the bottom center of the image and going to the top right of the image at a 45-degree angle. This is a problem because drawing a box around the road shows that object detection does not help determine direction any better than classification, as the same bounding box could mean the sUAS should go straight or right. Although object detection proved to be of little use in determining direction, the Autonomous Drone project did have one final idea. The AD project decided to create a grid of images from the original image.

A grid of images is created by slicing an image into 25 smaller images, classifying those images with a 0 being 'not road' and a 1 being 'road', and then putting those

results into an array (See Appendix A.2 for code). The belief is that the array will hold enough information to help the small Unmanned Aircraft System determine direction. Here is a visual representation of how the contents of the grid would represent the original image:

0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0

Figure 7. Image – Grid representation

The grid itself would be represented as an array in the program. This array can then be used to help train another classifier or any other method to help determine direction.

Design and Implementation

After determining that an image can be classified in such a way as to provide information for direction decision making, the next step is to interact with the small unmanned aircraft system (sUAS). The FireMAP research team uses DJI (Dà-Jiāng Innovations) drones, such as the Phantom 4 Pro and Phantom 4, to complete projects. The DJI drones the FireMAP team use are controlled by mobile technology. Since the FireMAP research team is using an Apple mobile device to control the sUAS, the program to determine direction is written for iOS (i Operating System) devices in the Swift language. The feat here is being able to control the sUAS, take an image with the

sUAS, have the iOS device retrieve the image and then classify the image on that device all while maintaining control of the sUAS.

Figuring out how to control the sUAS quickly became the priority. Fortunately, the desire to create applications for a drone has been noticed and DJI has made a software development kit (SDK) that can be downloaded. This SDK acts as base software for creating unique programs that interact with their drones. Also, DJI has extensive documentation for how "to give any developer with iOS or Android experience the knowledge and understanding required to create world changing applications using DJI's technology" (DJI, 2018). However, the documentation for iOS is written in an older language called Objective-C while the iOS community has moved on to Swift. Even though translating from one unknown language to another was difficult, reading *Mastering Swift 4* by Jon Hoffman and utilizing Stack Overflow's community to help answer questions was extremely beneficial.

After knowing the sUAS can be controlled from an iPhone or iPad, the design of the application became important. Knowing this program had to eventually be able to autonomously fly wildland fire containment lines and rail grades, the first action the user of the program should take is specifying whether the user is about to fly a postfire area or rail grade. Also, terrain can be important to know because future versions of the application can utilize different algorithms for different terrains. These features dictated the home page of the project which is shown in Figure 8.

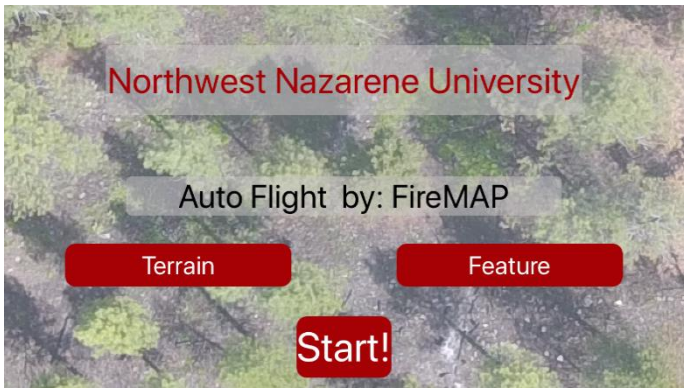


Figure 8. Application Home View

Once the home page was created (See Appendix A.3 for home page code), the next page, or view as it is called within the iOS development, is the view where the functions are that control the sUAS. This view looks like:

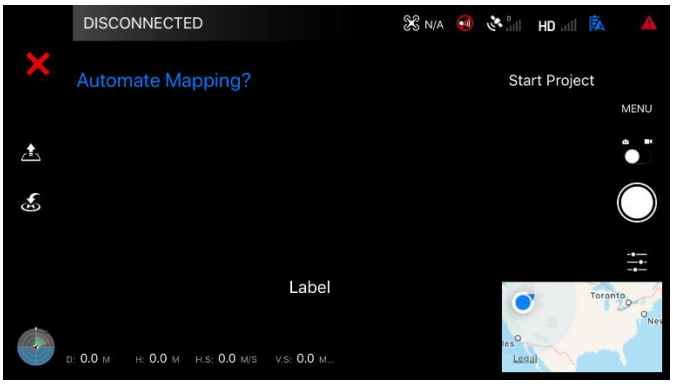


Figure 9. sUAS Control View

The large black area that consumes the majority of the view is where the camera feed from the drone is shown. The 'Start Project' button at the top right of the sUAS Control View gives the ability to have the sUAS take an image about every 13 meters the sUAS flies horizontally. One of the current applications the FireMAP research team uses is the Pix4DCapture app. The Pix4DCapture app also has the ability to set how often an image is taken per distance traveled. While the Pix4DApp can be set to one image taken per 1

meter traveled this is not truly the case. On one flight (See Appendix C for details), the Pix4DApp took an image every 17.79 meters when it was set to take one image per meter. Part of this is due to the ability of the sUAS. The DJI Phantom 4 cannot take an image while it is storing the image it just took. This means the process of taking images quickly is largely limited by the speed of the SD card and file format (DJI, n.d). Another way to speed up the ability of the sUAS to take a picture is to continuously attempt to retrieve an image right after the sUAS took an image. This forces the sUAS to take a picture as soon as the sUAS is ready. The "Retry Manager" (Miller, 2018) is implemented in the program (See Appendix A.5 for code) to take pictures quickly with the sUAS. This result is one example of how programming the functions and not using pre-existing applications can improve speed of an action (See Appendix A.4 for sUAS Control View Code).

The "Start Project" button is needed for mapping and the "Automate Mapping?" button at the top left of the sUAS Control View internally utilizes the "Start Project" functionality. The "Automate Mapping?" button, when pushed, calls code that has the sUAS take an image and then send the image to the mobile device. At that point, the mobile device slices the image into 25 smaller images, classifies those images and stores the classification results into an array (See Appendix A.4, A.6 and A.7 for the "Automate Mapping?" code). Although a CNN in TensorFlow was able to be utilized on a desktop computer, that same CNN could not be used on a mobile device.

Turi Create

Not being able to use the same CNN built on the desktop meant discovering how to create a CNN that would work on a mobile device. While TensorFlow documentation does have a guide to *Building TensorFlow on iOS* using TensorFlow Mobile, TensorFlow expects to deprecate TensorFlow Mobile in early 2019 in lieu of their TensorFlow lite application (TensorFlow, 2018). Unfortunately, the documentation on using TensorFlow lite in iOS development is very scarce and "TensorFlow Lite currently supports (only) a subset of TensorFlow operators" (TensorFlow Lite, 2018). This means that the CNN created on the desktop with TensorFlow could not be implemented in a mobile environment without major modifications.

Although the model created with TensorFlow to find a road within an image could not be used in the mobile environment, this did not mean a different, simpler CNN would not work. There were two choices at time of this research effort: create a CNN from scratch or find another CNN that could be used. While researching how to implement a CNN in a mobile environment, Turi Create was discovered. Turi Create was founded to help mobile developers find an easier way to deploy machine learning algorithms in their applications. Turi Create was acquired by Apple, which then focused on making Turi Create easier for iOS developers to implement machine learning (ML) algorithms that were easily exported to CoreML (Pham, 2018), CoreML is the framework Apple provides to integrate ML algorithms into an application.

Turi Create allows a programmer to take advantage of a desktop's computing capabilities while training and then creates a trained CoreML model that can be inserted

directly into an iOS application (See Appendix A.8 and A.2 for code to create a trained CoreML model). As is common with the implementation of any machine learning classifier, the hardest part about creating a model for the iOS device was not trying to use Turi Create but training an accurate model. Training a model for mobile is more difficult than using TensorFlow on a desktop. This is because a CNN cannot be as complex in a mobile device as it can be in a desktop environment as there is little or no support for some operations (TensorFlow Lite, 2018).

To achieve the same accuracies in a mobile environment as a desktop environment, more images must be used to train the model. Although using more images can cause a model to take longer to train, Turi Create does try to improve runtime performance through parallelization by automatically utilizing Mac graphics processing units (GPU) and using pre-trained image classifiers. GPU's, over a central processing unit (CPU), are frequently used for machine learning tasks as a GPU can perform many more specific computations simultaneously than a CPU, since a GPU has many more cores than a CPU. Similarly, training on a desktop with better, more powerful hardware, compared to a mobile device, will decrease the training time of a CNN dramatically. Turi Create also uses pre-trained image classifiers to help speed up results by "not needing to adjust hyper-parameters, faster training, and better performance even in cases where you don't have enough data to create a convention deep learning model" (Apple, 2018).

After many trials and continuous cleaning and manipulation of images, the Autonomous Drone project created a model from 12,000 images, 6,000 labeled

“notRoad” and 6,000 labeled “road”, that produced an accuracy of .94 or 94% in determining whether an image had a road in it (See Appendix D.1 for the results and image count of each trained model). The classifiers were tested on three different sets of images which the classifiers had not seen during training. Each classifier produced an accuracy from each set of images and the average of those results became the classifiers overall accuracy. See Figure 10 for the confusion matrix of the model that had an average accuracy of .94.

	Road	Not Road	Total	Recognition (%)
Road	17	3	20	85.00
Not Road	3	77	80	96.25
Total	20	80	100	94.00

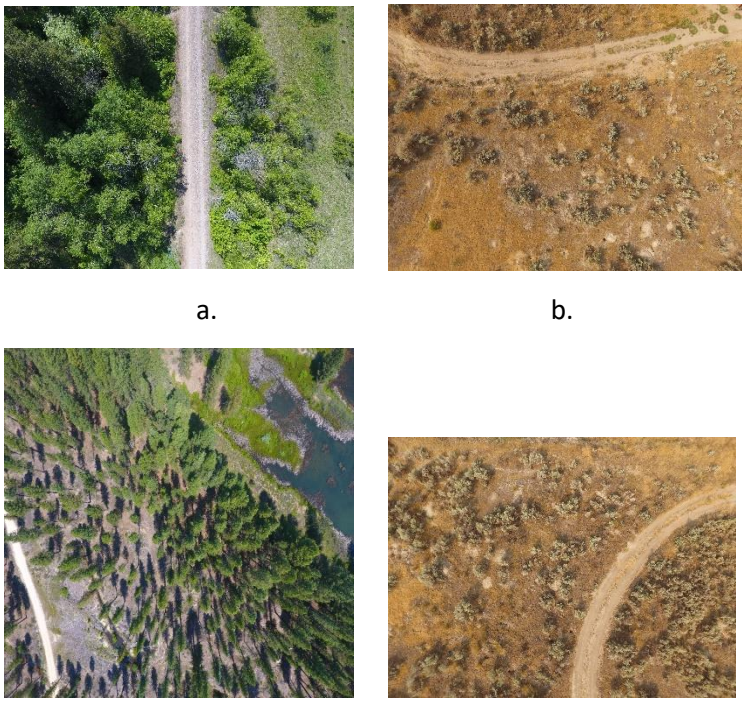
Figure 10. Confusion Matrix for Road Classification

The confusion matrix in Figure 10, which was modified output from Turi Create, shows the classifier correctly classified “road” when the image contained a road 17/20 times and classified “notRoad” when the image did not contain a road 77/80 times.

Direction

After training the Turi Create convolutional neural network (CNN), Turi Create generated a trained CoreML model automatically. This CoreML model was easily integrated into the iOS application (See Appendix A.7 to see CoreML model in code). Turi Create also comes with other classifiers. Since a classifier was able to find a road in an image, maybe a classifier could determine direction as well. Multiple classifiers were then used in trying to choose a direction. The information fed into each classifier for

training were the same, the array of results from the CNN image classifier and a label. The label was chosen by a person looking at the image and picking a result from the following scenarios.



a. b.

c. d.

Figure 11. Straight (a) Ask User (b) Left (c) Right (d)

If the road originated from the bottom of the image and traveled to the left (See Figure 11 c), top left corner, straight (Figure 11 a), top right corner or right (Figure 11 d), then a corresponding direction was assigned. However, if the road did not originate from the bottom of the image, which would happen if there was no road in the image or in an image like that in Figure 11 b, a “notRoad” label and an “ask user” label would be assigned to the image respectively. An image would have an ask user label in the event the sUAS had just started the “AutoMapping?” procedure and had not already been following a road.

After understanding what label to give each image, the team labeled a total of 700 images in which to train a classifier to determine direction. Each label had 100 images (see figure 12 for table). Keeping balanced classes was a decision made to not give a bias to any class. After the 700 images were labeled, each image was cut into 25 smaller images, classified, and the classification results and label were put into an array.

Direction	Label	# of images used to train
Left	L	100
Left Straight	LS	100
Straight	S	100
Right Straight	RS	100
Right	R	100
Ask User	AU	100
No road in image	NR	100

Figure 12. Direction Labels

The array for each image were then used to train multiple classifiers built by Turi Create (See Appendix A.9 for the code to run different classifiers in Turi Create) in order to determine the direction of the road. These directional classifiers and their accuracies can be seen in Figure 13.

Classifier	**Accuracy (%)**
Logistic Regression	37.00
K-Nearest Neighbor	50.00
Boosted Trees	47.00
Random Forest	56.00

Figure 13. Directional Classifiers

Obviously, an accuracy of 0.56 on determining direction is not optimal. Basically, at 0.56 accuracy, it is a coin flip guess on whether the classifier can accurately predict direction

(See Appendix D.2 for more detailed results). However, there was one more classifier the AD project wanted to use and it was not in the arsenal of Turi Create.

In the fall of 2018, Casey Lewis, Jacob Winters and Keaton Woodcook created a Naïve Bayesian (n-Bayes) classifier (See Appendix A.10 and A.11 for n-Bayes code) that utilizes the Gram-Schmidt method to help with the Naïve Bayesian's independence assumption between features (Lewis, Winters, Woodcook, 2017). This was the final classifier the AD project used to determine direction (See Appendix A.12 for Gram-Schmidt code). At first, the results were not promising with an accuracy of about 0.55. However, after further studying the images, it became clear that a sUAS could still find a road if the sUAS was off in determining direction by 45 degrees. This is because the drone has a field of view of 180 degrees and any direction the drone travels, the sUAS will still see where the sUAS would have gone had it chosen a direction 45 degrees from its current position.

Using this directional classification to help determine the direction the sUAS should fly, the predicted decisions were assigned a weight. If the algorithm accurately predicted the correct label (left, right, straight and so on) then that prediction would get a weight of 1. If the algorithm's prediction was 45 degrees off, as in picking left when the prediction should have been left-straight, then that prediction would get a weight of 0.5 and all remaining predictions would get a weight of 0. This weighted approach improved accuracy to 0.67 (See Appendix D.3 for n-Bayes results and confusion matrix). A weighted approach to 90 degrees was also performed. Though, it was ultimately

decided that if the sUAS was 90 degrees off the correct direction the sUAS would not be able to find the road again.

Future Work

Even with a weighted approach to accuracy, determining direction with 0.67 accuracy is not ideal. Actually, an accuracy of 0.67 is worse than ideal. Having 0.67 accuracy means that the drone would have a 0.67 * 0.67 * 0.67 = 0.3 or 30% chance of correctly identifying the direction to travel three times in a row. Obviously, the classifiers used thus far are not going to work. However, the Autonomous Drone project has identified two more avenues to pursue in the future for determining direction. One is to create an Artificial Neural Network (NN) from scratch. Although, with the lack of success from previous classifiers in determining direction and the amount of time it would take to build, the NN will be completed as last resort. The other avenue to pursue is the maximum likelihood estimation (MLE) method proposed by Jason Colwell, Ph.D, which "is the procedure of finding the value of one or more parameters for a given statistic which makes the known likelihood distribution a maximum "(Weisstein, 2018). Basically, the MLE method tries to find direction based on the maximum likelihood given the observations. Observations, in this case, are the 0's (not road) and 1's (road) in the array and their position within the array.

Conclusion

Overall, this project was extremely fun and difficult. While machine learning and artificial intelligence are not new concepts, many of the tools, especially when it comes to neural networks, are. This presented many challenges in how to use the tools to help

accomplish the goal of the Autonomous Drone project. Of course, using the tools were only a small fraction of the project, training the networks were time consuming and took a lot of research and learning to do correctly. Also, determining direction was and still is a challenge.

To get this far on the project, nearly every class taken from Northwest Nazarene University has helped. A couple of the math courses I relied on heavily in this project were Linear Algebra (an example was utilizing the Gram-Schmidt method) and Probability and Statistics (as is evident in the Naïve Bayesian). I also leaned on the knowledge I gained from my computer science classes. Data Mining/Machine Learning was obviously a huge influence on the Autonomous Drone project. Also, Data Structures and Operating Systems proved extremely beneficial. In fact, one of the most crucial design implementations were solved with what I learned in Operating systems.

This design implementation is seen in Appendix A.7 under the function 'classifyImages'. In 'classifyImages' there is a 'DispatchQueue.global().async' method that allows me to give priority to the MainQueue, which contains the foreground processes, but still control the order in which the images are classified. This is extremely important because the foreground process, which in this case is the part of the app that controls the drone, should not be hindered by the classification of images. Meaning, I should still be able to control the sUAS while classifying images. This is only one of the many ways my degree has helped me with this project.

While the Autonomous Drone project was not fully completed, a large amount of the project and research was. With further development, the AD project could generate good publicity for the department as Machine Learning and Artificial Intelligence is currently a hot topic. Also, having a machine control the flight of a sUAS could improve efficiency and increase area flown per battery. This is because a computer can ensure accurate, consistent mapping techniques, while a human is less likely to map an area efficiently.

References

Apple. (2018, June 04). How Does this Work? Retrieved November 24, 2018, from https://github.com/apple/turicreate/blob/master/userguide/image_classifier/how-it-works.md

DJI. (2018). Mobile SDK. Retrieved November 24, 2018, from https://developer.dji.com/mobile-sdk/documentation/introduction/index.html

DJI. (n.d.). DJI Android Mobile SDK Reference. Retrieved November 24, 2018, from https://developer.dji.com/iframe/mobile-sdk-doc/android/reference/dji/sdk/Camera/DJICameraSettingsDef.CameraPhotoIntervalParam.html

Drawing a convolution with Tikz. (n.d.). Retrieved November 21, 2018, from https://tex.stackexchange.com/questions/437007/drawing-a-convolution-with-tikz

Google Developers | ML Practicum: Image Classification | Machine Learning Practica. (Last Updated 2018, October 1). Retrieved November 21, 2018, from https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks

Hamilton, D; Pacheco*, R; Myers, B; Peltzer, B*, (In Press) “kNN vs. SVM: a Comparison of Algorithms”, *Fire Continuum Conference: Preparing for the Future of Wildland Fire;* May 21-24, 2018; Missoula, MT

Han, J., Kamber, M., & Pei, J. (2012). *Data mining: Concepts and techniques*. Waltham: Morgan Kaufmann.

Hoffman, J. (2017). *Mastering Swift 4: An in-depth and comprehensive guide to modern programming techniques with Swift / Jon Hoffman*. Birmingham, UK: Packt Publishing.

Lewis, C; Winters, J; Woodcook, K. (2017) Fixing the Naivety of the Naïve Bayesian Classifier.

Max-pooling / Pooling. (Last edited 2018, February 27). Retrieved November 21, 2018, from https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

Miller, J. (2018, June 27). DJI SDK Get last image. Retrieved December 8, 2018, from https://stackoverflow.com/questions/50957589/dji-sdk-get-last-image

Pham, K. (2018, May 23). Machine learning in iOS: Turi Create and CoreML – Flawless App Stories – Medium. Retrieved November 24, 2018, from https://medium.com/flawless-app-stories/machine-learning-in-ios-turi-create-and-coreml-5ddce0dc8e26

TensorFlow. (2018, November 13). Building TensorFlow on iOS | TensorFlow Lite | TensorFlow. Retrieved November 24, 2018, from https://www.tensorflow.org/lite/tfmobile/ios_build

TensorFlow Lite. (2018, November 13). How to use custom operators | TensorFlow Lite | TensorFlow. Retrieved November 24, 2018, from

https://www.tensorflow.org/lite/custom_operators

Weisstein, E. W. (2018). Maximum Likelihood. Retrieved November 25, 2018, from http://mathworld.wolfram.com/MaximumLikelihood.html

Appendices

A. Code

A.1 CNN Code

```
#  Copyright 2018 The TensorFlow Authors. All Rights Reserved.
#
#  Licensed under the Apache License, Version 2.0 (the "License");
#  you may not use this file except in compliance with the License.
#  You may obtain a copy of the License at
#
#   http://www.apache.org/licenses/LICENSE-2.0
#
#  Unless required by applicable law or agreed to in writing, software
#  distributed under the License is distributed on an "AS IS" BASIS,
#  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#  See the License for the specific language governing permissions and
#  limitations under the License.



# This CNN is created to differentiate between railgrades and not railgrades
# using images.
# Adapted by Casey Lewis on 06/11/2018 from https://www.tensorflow.org/tutorials/layers,
# https://github.com/tensorflow/models/blob/master/samples/core/get_started/custom_estimator.py,
# knowledge extracted from Google's TensorFlow documentation and
# Blake Johanson's existing code.


import tensorflow as tf
import os
import argparse
import random


def cnn_model_fn(features, labels, mode, params):
    """Model function for CNN."""

    ##########################################################################################
```

```
    # Input Layer for creating convolutional and pooling layers for two-dimensional         #
    #   image data expect input tensors to have a shape of [batch_size, image_height,       #
    #   image_width, channels] by default.                                                  #
    # To convert our input feature map to this shape, we can use the                        #
    #   tf.reshape() function. the second arg in this function call, the array,             #
    #   holds four values:                                                                  #
    #       ele[0] = batch size, a batch size of -1 means 'this dimension should be         #
    #           dynamically computed based on the number of input values in                 #
    #           "features["x"], holding the size of all other dimensions constant'.         #
    #           Normally the batch size is the 'size of the subset of examples used         #
    #           when performing gradient descent during training'                           #
    #                                                                                       #
    #       ele[1] is the image height in pixels                                            #
    #       ele[2] is the image width in pixels                                             #
    #       ele[3] is the number of channels, i.e. 3 would be for RGB                       #
    #                                                                                       #
    # To note:                                                                              #
    #   Each layer accepts a tensor as input and returns a transformed tensor as output     #
    #   making it easy to connect one layer to another.                                     #
    #########################################################################################

     input_layer = tf.reshape(features["x"], [-1, params['im_h'],
params['im_w'], params['channels']])

    #########################################################################################
    # Convolutional Layer #1
    #   Arg 1, input:
    #       see block comment above
    #
    #   Arg 2, filters:
    #       amount of filters applied to the input layer, also creates the
    #       same amount of features
    #
    #   Arg 3, kernel_size:
    #       5x5 pixel size of filter
    #       TIP: If filter height and width have the same value, you can instead specify a
    #           single integer for kernel_size—e.g., kernel_size=5.
```

```
#
#   Arg 4, padding:
#       'same' specifies output tensor to have the same height and width
#       as the input tensor
#
#   Arg 5, activation:
#       using Rectified Linear Unit to create a non-linear function
#
# Input layer tensors change based on arguments, filters arg will be the channel input
#   of the next layer (also depends on padding...)
#########################################################################################


 # Input Tensor Shape: [batch_size, 300, 400, 3]
 # Output Tensor Shape: [batch_size, 300, 400, 64]
 conv1 = tf.layers.conv2d(
     inputs=input_layer,
     filters=64,
     kernel_size=[10, 10],
     padding="same",
     activation=tf.nn.relu)

#########################################################################################
# Pooling Layer #1
#   Arg 1, input:
#       see 1st block comment above
#
#   Arg 2, pool size:
#       size of pooling (max-pooling in this case) filter
#       TIP: If both dimensions have the same value, you can
#           instead specify a single integer (e.g., pool_size=2).
#
#   Arg 3, strides:
#       the distance, in pixels, separating each extracted tile
#
# Input layer tensors change based on arguments, pool arg will change the height and
#   width of the next layer (height = input height / pool_size height)
#########################################################################################
```

```
    # Input Tensor Shape: [batch_size, 300, 400, 64]
    # Output Tensor Shape: [batch_size, 60, 80, 64]
    pool1 = tf.layers.max_pooling2d(inputs=conv1, pool_size=[5, 5], strides=5,
padding="same")

    # Convolutional Layer #2
    # Input Tensor Shape: [batch_size, 60, 80, 64]
    # Output Tensor Shape: [batch_size, 60, 80, 128]
    conv2 = tf.layers.conv2d(
        inputs=pool1,
        filters=128,
        kernel_size=[10, 10],
        padding="same",
        activation=tf.nn.relu)

    # Pooling Layer #2
    # Input Tensor Shape: [batch_size, 60, 80, 128]
    # Output Tensor Shape: [batch_size, 12, 16, 128]
    pool2 = tf.layers.max_pooling2d(inputs=conv2, pool_size=[5, 5], strides=5,
padding="same")

   ##########################################################################################

   # Dense Layer

   # First, a dense layer (a normal hidden layer for a neural net) accepts an input

   #   format of [batch_size, features]. Currently, the input format is [batch_size,

   #   height, width, channels], so we reshape to change the format with tf.reshape()

   # Next, we use tf.layers.dense, unit is amount of neurons in the

   #   layer and activation is an activation function.

   # Finally, dropout performs a type of generalization. The rate arg specifies the

   #   dropout rate, which is the percentage of elements that will be randomly

   #   dropped during training.

   ##########################################################################################


    # Input Tensor Shape: [batch_size, 12, 16, 128]
    # Output Tensor Shape: [batch_size, 12 * 16 * 128]
    pool2_flat = tf.reshape(pool2, [-1, 12 * 16 * 128])

    # Input Tensor Shape: [batch_size, 7 * 7 * 64]
    # Output Tensor Shape: [batch_size, 1024]
    dense = tf.layers.dense(inputs=pool2_flat, units=1024,
activation=tf.nn.relu)

    # Add dropout operation; (1 - rate) probability that element will be kept

    dropout = tf.layers.dropout(
        inputs=dense, rate=0.63, training=mode == tf.estimator.ModeKeys.TRAIN)
```

```
    ########################################################################################

    # Logits Layer is a dense layer that returns the raw values of our predictions,

    #   one neuron for each target class

    ########################################################################################


    # Input Tensor Shape: [batch_size, 1024]
    # Output Tensor Shape: [batch_size, 10]
    logits = tf.layers.dense(inputs=dropout, units=2)

    predictions = {
        # Generate predictions (for PREDICT and EVAL mode)
        "classes": tf.argmax(input=logits, axis=1),
        # Add `softmax_tensor` to the graph. It is used for PREDICT and by the
        # `logging_hook`.
        "probabilities": tf.nn.softmax(logits, name="softmax_tensor")
    }

    if mode == tf.estimator.ModeKeys.PREDICT:
        return tf.estimator.EstimatorSpec(mode=mode, predictions=predictions)

    # Calculate Loss (for both TRAIN and EVAL modes)
    loss = tf.losses.sparse_softmax_cross_entropy(labels=labels,
logits=logits)

    # Configure the Training Op (for TRAIN mode)
    if mode == tf.estimator.ModeKeys.TRAIN:
        optimizer = tf.train.AdamOptimizer(learning_rate=0.001)
        #optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001)
        train_op = optimizer.minimize(
            loss=loss,
            global_step=tf.train.get_global_step())
        return tf.estimator.EstimatorSpec(mode=mode, loss=loss,
train_op=train_op)

    # Add evaluation metrics (for EVAL mode)
    eval_metric_ops = {
        "accuracy": tf.metrics.accuracy(
            labels=labels, predictions=predictions["classes"])}
    return tf.estimator.EstimatorSpec(
        mode=mode, loss=loss, eval_metric_ops=eval_metric_ops)

def _parse_function(filename, label, img_h, img_w):
    """Reads an image from a file, decodes it into a dense tensor, and resizes
it
    to a fixed shape."""
    image_string = tf.read_file(filename)
    # read in the jpeg and create features, channels = 3 for rgb
    image_decoded = tf.image.decode_jpeg(image_string, channels=3)
    image_resized = tf.image.resize_images(image_decoded, [img_h, img_w])
    image_reshaped = tf.reshape(image_resized, [-1, img_h, img_w, 3])
    label = tf.reshape(label, [1])
    return image_reshaped, label
```

```
def input_fn(filenames, labels, image_h, image_w, batch_size):
    """
    An input function for training
    Filenames but become
        features after this function
    """
    # Convert the inputs to a Dataset.
    dataset = tf.data.Dataset.from_tensor_slices((filenames, labels))
    dataset = dataset.map(lambda x, y: _parse_function(x, y, image_h,
image_w))

    dataset = dataset.batch(batch_size)

    # Return the read end of the pipeline.
    iterator = dataset.make_one_shot_iterator()
    features, labels = iterator.get_next()
    return {'x': features}, labels


    # dataset = dataset.map(lambda filename, label: _parse_function(filenames,
labels, img_h=28, img_w=28))
    # Shuffle and batch the examples.
    # dataset = dataset.shuffle(1000).batch(batch_size)




def find_filepaths_and_labels(file_dir, label_name):
    """
    file_dir is the directory that holds images
    label_name is the name of the images, which should be saved as the label.
    """

    # two lists, one consisting of image filepaths and the other labels, both
are returned.
    filenames = []
    labels = []
    # iterate through images in the file and add them to filenames and labels
list
    for root, dirs, files in os.walk(file_dir):
        for filename in files:
            filenames.append(os.path.join(root, filename))
            # get the name of the image up to the 'space', ex: trail (20).jpeg
            #   gets trail, tail would be 1 and everything else 0
            label = 1 if (str(filename.split(' ')[0]) == label_name) else 0
            #print(filename, label)
            labels.append(label)

    # shuffles both lists together to keep labels with filenames
    combined = list(zip(filenames, labels))
    random.shuffle(combined)
    filenames[:], labels[:] = zip(*combined)

    return filenames, labels
```

```
def main(data_dir, class_label, num_of_eval,
         img_height, img_width, batch_size, steps):

    channels = 3

    # for logging
    tf.logging.set_verbosity(tf.logging.INFO)

    # get filepaths and labels
    filenames, labels = find_filepaths_and_labels(data_dir, class_label)

    # make train and test
    # get the last n data to the end of the list
    eval_filenames = filenames[-num_of_eval:]
    eval_labels = labels[-num_of_eval:]
    # get all data up to n away from end of the list
    filenames = filenames[:-num_of_eval]
    labels = labels[:-num_of_eval]

    # make filenames and labels a tensorflow constant
    filenames = tf.constant(filenames)
    labels = tf.constant(labels, dtype=tf.int32)

    # create feature column, basically 3d matrix with shape of image and
channels
    #my_feature_columns = [tf.feature_column.numeric_column("x",
shape=[img_height, img_width, channels], dtype=tf.float32)]


    # Create the Estimator
    classifier = tf.estimator.Estimator(
        model_fn=cnn_model_fn,
        params={
            'im_h': img_height,
            'im_w': img_width,
            'channels': channels
        },
        model_dir="./my_models/convnet_model")

    # Set up logging for predictions
    # Log the values in the "Softmax" tensor with label "probabilities"
    tensors_to_log = {"probabilities": "softmax_tensor"}
    logging_hook = tf.train.LoggingTensorHook(
        tensors=tensors_to_log, every_n_iter=1)

    classifier.train(
        input_fn=lambda: input_fn(filenames, labels, img_height, img_width,
batch_size),
        steps=steps,
        hooks=[logging_hook]
    )

    eval_results = classifier.evaluate(
        input_fn=lambda: input_fn(eval_filenames, eval_labels, img_height,
img_width, batch_size),)
```

```
    print(eval_results)

    #p_results = classifier.predict(
    #    input_fn=lambda: input_fn(eval_filenames, eval_labels, img_height,
img_width, batch_size)
    #)
    #print(p_results)



if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "-d",
        "--data-dir",
        type=str,
        required=True,
        help="""\
        The directory where all images are stored. The directory should
        contain two directories, one for 'label' and one for 'not label'.
Label can
        be anything you are trying to classify, i.e. trail, railgrade, car,
etc.\
        """
    )
    parser.add_argument(
        "-l",
        "--class-label",
        type=str,
        required=True,
        help="""\
            This is the name of the image up to the space. For example, if
image is saved
            as 'trail (10).jpeg' this arg would need to be 'trail'. This arg
            acts as the label.\
            """
    )
    parser.add_argument(
        "-b",
        "--batch-size",
        type=int,
        required=True,
        help="This arg sets the batch size."
    )
    parser.add_argument(
        "-ne",
        "--num-of-eval",
        type=int,
        required=True,
        help="The amount of images to be used for evaluation (testing)."
    )
    parser.add_argument(
        "-ih",
        "--img-height",
        type=int,
```

```
    required=True,
    help="The height (in pixels) of the image to resize to."
)
parser.add_argument(
    "-iw",
    "--img-width",
    type=int,
    required=True,
    help="The width (in pixels) of the image to resize to."
)
parser.add_argument(
    "-s",
    "--steps",
    type=int,
    required=True,
    help="Amount of steps."
)
args = parser.parse_args()

if args.num_of_eval < 1:
    raise ValueError("-ne must be greater than 1.")

main(**vars(args))
```

A.2 Split and Classify Code

```
# Create script to resize a larger image to (960 x 540)
#   which is the size of the dji sdk preview
# Then cut image into 5x5, classify those images and
#   stick result into list then save the list as txt
#
# To see params in command line run:
#   $ python splitAndClassify.py -h
#
# Command line params:
#     "-id" which is the image directory and is required
#     "-dd" which is the destination directory to store the images and list as
txt file, default is 'results'
#   "-md" which is the directory where the model is stored

import turicreate as tc
import cv2 as cv
import os
import argparse
from getFilepaths import get_filepaths
import numpy as np


def get_dest_dir(image_dir, dest_dir):
    """
    This function checks to see if user supplied a destination
    directory and if not makes one, then returns the directory
    """
    # check if user supplied dest directory
    if dest_dir == None:
        # get path to image_dir
        dest_dir = os.path.dirname(os.path.abspath(image_dir))
        # get newImages directory in directory with images
        dest_dir = os.path.join(dest_dir, r'results')

        # make directory if doesn't already exist
        if not os.path.exists(dest_dir):
            os.makedirs(dest_dir)
            dest_dir = dest_dir

    return dest_dir

def main(image_dir, dest_dir, model_dir):
    """
    main gets the filepaths then crops the images
    and saves them to a directory as jpegs
    """
    #### CREATE COMMAND LINE PARAMS FOR THESE ####
    # images to cut across
    aImg = 5
    # images to cut down
    dImg = 5
```

```
    # resized image width
    rImgW = 960
    # resized image height
    rImgH = 540

    # total images to make
    ttlImg = aImg * dImg

    # get filepaths of images in image_dir
    imgFilepaths = get_filepaths(image_dir)

    # get destination directory
    dest_dir = get_dest_dir(image_dir, dest_dir)

    # iterators
    # using 1001 as starting with 1 saves images as 1,10-19,2,20-29,...
    imgSection = 1001
    imgSectW = 1
    imgSectH = 1

    # dictionary key is the image name, the values are the cropped pieces of
that image
    imageDict = {}

    for imgFile in imgFilepaths:
        # resize image
        img = cv.imread(imgFile)
        # interpolation default is cv.INTER_LINEAR
        image = cv.resize(img, (rImgW, rImgH))

        # get filename from imgFile
        imgFilename = os.path.splitext(os.path.basename(imgFile))[0]
        if "LF" in imgFilename:
            print(imgFilename + " in LF")
        elif "RF" in imgFilename:
            print(imgFilename + " in RF")
        elif "AU" in imgFilename:
            print(imgFilename + " in AU")
        elif "NR" in imgFilename:
            print(imgFilename + " in NR")
        elif "F" in imgFilename:
            print(imgFilename + " in F")
        elif "R" in imgFilename:
            print(imgFilename + " in R")
        elif "L" in imgFilename:
            print(imgFilename + " in L")

        # rename resized image and save
        #cv.imwrite(os.path.join(dest_dir, str(imgFilename) + ".jpg"), image)

        # set pixels to origin
        xPixel = 0
        yPixel = 0
        for sections in range(0, ttlImg):
            # add 1/x of width and height to current pixel
```

```
            nextX = int(xPixel + ((rImgW / aImg)))
            nextY = int(yPixel + ((rImgH / dImg)))

            #print("y: ", yPixel, nextY)
            #print("x: ", xPixel, nextX)

            crop_image = image[yPixel:nextY, xPixel:nextX]

            # move xPixel to new location
            xPixel = nextX

            # check if gone across image, if so increment yPixel and set
xPixel to 0
            if imgSectW == aImg:
                imgSectW = 1
                xPixel = 0
                yPixel = nextY
            else:
                imgSectW = imgSectW + 1


            # save cropped image in temp folder
            temp_dir = os.path.join(dest_dir, "temp" + imgFilename)

            # make directory if doesn't already exist
            if not os.path.exists(temp_dir):
                os.makedirs(temp_dir)

            cv.imwrite(os.path.join(temp_dir, str(imgFilename) + "_" +
str(imgSection) + ".jpg"), crop_image)

            # add tempfolder to dict
            imageDict[imgFilename] = temp_dir

            # increment section
            imgSection = imgSection + 1

        # reset imgSection
        imgSection = 1001


    # load model
    model = tc.load_model(model_dir)

    # predicted class dict of lists (0's and 1's)
    classDict = {}
    valueDict = {}

    for key in imageDict:

        data = tc.image_analysis.load_images(imageDict[key], with_path=True)
        data['predictions'] = model.predict(data)
        predict = model.classify(data)

        valueDict[key] = []
```

```
        for s in predict:
            #print(s)
            next = 0

            for k, v in s.items():
                #print("key: ",k," value: ", v)
                if next == 1:
                    valueDict[key].append("%.2f" % float(v))
                elif next == 2:
                    valueDict[key].append("%.2f" % (1 - float(v)))

                if v == 'road':
                    next = 1
                elif v == 'notRoad':
                    next = 2


        #data.print_rows(num_rows=200, num_columns=3, max_column_width=1500)

        #top = model.predict_topk(data, output_type='probability', k=2)
        #print("topk: ", model.predict_topk(data, output_type='probability',
k=2))
        #print("classify: ", model.classify(data))
        #print("predict: ", model.predict(data))

        classDict[key] = []

        # save predictions as one(road) or zero(notRoad)
        # as I used 1001 for imgSection number, the data['predictions'] SFrame
is
        # now ordered from left top section of the image going right then down
        # to the right bottom of the image
        for prediction in data['predictions']:
            #print(prediction)
            if prediction == "road":
                classDict[key].append(1)
            else:
                classDict[key].append(0)

    #save dictionary
    np.save(os.path.join(dest_dir, "labels"), classDict) # 0 or 1
    np.save(os.path.join(dest_dir, "values"), valueDict) # actual value (e.g.
.85

    # Load
    read_dictionary = np.load(os.path.join(dest_dir, "labels.npy")).item()
    read_dictionary2 = np.load(os.path.join(dest_dir, "values.npy")).item()
    print(read_dictionary)
    print(read_dictionary2)

    # delete temp folders
    #for tempFolder in range(1, imgNumber):
    #    os.remove(os.path.join(dest_dir, "temp" + str(tempFolder)))
```

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser()

    parser.add_argument(
        "-id",
        "--image-dir",
        type=str,
        required=True,
        help="REQUIRED. The directory where all images are stored."
    )
    parser.add_argument(
        "-dd",
        "--dest-dir",
        type=str,
        required=False,
        help="""\
             OPTIONAL. The directory where the newly created images will go.
             The default is the arg for '--image-dir' or '-id' + 'newImages'.\
             """
    )
    parser.add_argument(
        "-md",
        "--model-dir",
        type=str,
        required=True,
        help="REQUIRED. The directory where the model is stored."
    )

    args = parser.parse_args()

    main(**vars(args))
```

A.3 App Home View

```
//  MainViewController.swift
//  AutoFlight
//
//  Created by Casey Lewis on 9/15/18.
//  Copyright © 2018 Casey Lewis. All rights reserved.
//

import UIKit
import DJISDK

class MyMainViewController: UIViewController, UIPickerViewDataSource,
UIPickerViewDelegate {

    var dictPassed: [String:String] = ["Terrain":"None", "Feature":"None"]

    @IBOutlet weak var nnuLabel: UILabel!
    @IBOutlet weak var fireMapLabel: UILabel!

    @IBOutlet weak var terrainButton: UIButton!
    @IBOutlet weak var featureButton: UIButton!
    @IBOutlet weak var startButton: UIButton!

    @IBOutlet weak var terrDropDown: UIPickerView!
    @IBOutlet weak var featDropDown: UIPickerView!

    @IBOutlet weak var backgroundImage: UIImageView!

    // on load
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        // have image auto fit
        self.backgroundImage.contentMode = .scaleAspectFill

        // hide dropdown boxes
        self.terrDropDown.isHidden = true
        self.terrDropDown.delegate = self
        self.terrDropDown.dataSource = self
        self.featDropDown.isHidden = true
        self.featDropDown.delegate = self
        self.featDropDown.dataSource = self

        // round the corner of the background labels
        self.nnuLabel.layer.cornerRadius = 5
        self.fireMapLabel.layer.cornerRadius = 5
        self.terrainButton.layer.cornerRadius = 10
        self.featureButton.layer.cornerRadius = 10
        self.startButton.layer.cornerRadius = 10

    }
```

```
    /*********************************START DROPDOWN*********************/

    /* Terrain DropDown */
    let terrainOptions = ["Rangeland", "Forested"]

    @IBAction func terrainButtonPushed(_ sender: UIButton) {
        /* when Terrain button is pushed, have user select terrain and change
name of button to match*/

        if self.terrDropDown.isHidden{
            self.terrDropDown.isHidden = false
        }else{
            self.terrDropDown.isHidden = true
        }
    }

    /* Feature DropDown */
    let featureOptions = ["Railgrade", "Post-Fire"]

    @IBAction func featureButtonPushed(_ sender: UIButton) {
        /* when feature button is pushed, have user select feature and change
name of button to match*/

        if self.featDropDown.isHidden{
            self.featDropDown.isHidden = false
        }else{
            self.featDropDown.isHidden = true
        }
    }



    /* Functions for Terrain and Feature Dropdown */
    public func numberOfComponents(in pickerView: UIPickerView) -> Int{
        return 1
    }

    public func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent
component: Int) -> Int{

        // return number of items in the dropdown box
        if pickerView == self.terrDropDown{
            return self.terrainOptions.count
        } else {
            return self.featureOptions.count
        }
    }

    func pickerView(_ pickerView: UIPickerView, titleForRow row: Int,
forComponent component: Int) -> String? {

        // show list in dropdown box
        if pickerView == self.terrDropDown{
```

```
            return self.terrainOptions[row]
        } else {
            return self.featureOptions[row]
        }
    }

    func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int,
inComponent component: Int) {

        // on selection of dropdown row, put selection as name of box and into
dictionary
        if pickerView == self.terrDropDown{
            // change title of button to selection
            self.terrainButton.setTitle(self.terrainOptions[row], for:
.normal)
            self.terrDropDown.isHidden = true
            // put selection in dict
            self.dictPassed["Terrain"] = self.terrainOptions[row]
        } else {
            // change title of button to selection
            self.featureButton.setTitle(self.featureOptions[row], for:
.normal)
            self.featDropDown.isHidden = true
            // put selection in dict
            self.dictPassed["Feature"] = self.featureOptions[row]
        }
    }

    /*********************************END DROPDOWN**********************/


    @IBAction func startButtonPushed(_ sender: UIButton) {
        /* When go button is pushed go to default screen*/
        self.performSegue(withIdentifier: "goToFlight", sender: nil)
    }


    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    // This function passes data from one scene to another
    override func prepare (for segue: UIStoryboardSegue, sender: Any!) {
        if (segue.identifier == "goToFlight") {
            let svc = segue.destination as!
DefaultLayoutCustomizationViewController
            svc.userPickDict = self.dictPassed
        }
    }

}
```

A.4 sUAS Control View

```
//  DefaultLayoutCustomizationViewController.swift
//  Started from: UXSDK Sample, Copyright © 2016 DJI. All rights reserved.
//
//  Created by Casey Lewis on 9/14/18.
//  Copyright © 2018 Casey Lewis. All rights reserved.
//


import UIKit
import DJIUXSDK

// We subclass the DUXRootViewController to inherit all its behavior.
class DefaultLayoutCustomizationViewController: DUXDefaultLayoutViewController
{

    var userPickDict: [String:String] = [:]     // a dict that comes from
another scene
    var projectStarted: Bool = false
    var autoMapStarted: Bool = false

    // buttons
    @IBOutlet weak var startProject: UIButton!  // the button to start the
project
    @IBOutlet weak var autoMap: UIButton!       // the automate mapping button

    // labels
    @IBOutlet weak var errorLabel: UILabel!


    // tests
    @IBOutlet weak var testView: UIImageView!


    /****** On Load *****/
    override func viewDidLoad() {

        super.viewDidLoad()

        // check the choices from user from previous scene
        self.checkDict()
    }
    /***** End On Load *****/


    override var preferredStatusBarStyle: UIStatusBarStyle {
        return .lightContent;
    }


    // when user selects 'x' button, return to previous scene
    @IBAction func close () {
```

```
            self.dismiss(animated: true) {

            }
        }


        /***** Check user input from previous scene *****/
        func checkDict(){

            let terrain = self.userPickDict["Terrain"]
            let feature = self.userPickDict["Feature"]

            // if user did not specify terrain or feature
            if (terrain == "None") || (feature == "None"){

                // disable and hide the automate mapping button
                self.autoMap.isUserInteractionEnabled = false
                self.autoMap.alpha = 0.0
            }
        }


        /***** Take a picture *****/
        func takePic(camera: DJICamera){

            // change mode of camera to shoot photo mode
            camera.setMode( .shootPhoto, withCompletion: {(error) in

                if error != nil {
                    self.errorLabel.text = ("\(error!.localizedDescription)")
                }
                else {
                    // take picture when project starts
                    camera.startShootPhoto(completion: { (error) in

                        if (error != nil) {
                            print("Shoot photo error: \(error.debugDescription)")
                        }
                    })// end of camera shoot block
                }
            }) // end set mode block
        }
    /***** Automate Mapping *****/
        @IBAction func autoMapPushed(_ sender: UIButton) {

            if self.autoMapStarted{

                // change button title
                self.autoMap.setTitle("Automate Mapping?", for: .normal)

            } else {

                /***** Get location *****/
```

```
            var droneLoc: CLLocation? = nil

            guard let locationKey = DJIFlightControllerKey(param:
DJIFlightControllerParamAircraftLocation) else {
                self.errorLabel.text = "Couldn't create the key"
                return
            }

            guard let keyManager = DJISDKManager.keyManager() else {
                self.errorLabel.text = "Couldn't get the keyManager"
                // This will happen if not registered
                return
            }

            if let locationValue = keyManager.getValueFor(locationKey) {
                let location = locationValue.value as! CLLocation
                // set droneLoc to current location
                droneLoc = location
            }

            print("Drone \(String(describing: droneLoc?.altitude))")

            // if not able to get location or if drone is below 100 meters
            //if (droneLoc == nil) || ((droneLoc?.altitude)! < Double(100)) {
            //    self.errorLabel.text = "Drone must be above 100 meters"
            //    return
            //}


            /******* Start Project (eventually) *****/

            // if project has been started, stop the project
            if self.projectStarted {
                self.startProjectPushed(self.startProject)
            }

            /****** Setup Camera *****/
            // get current product
            guard let drone = DJISDKManager.product() else {
                self.errorLabel.text = "Product is connected but
DJISDKManager.product is nil when attempting to download media"
                return
            }

            // Get camera on drone
            guard  let camera: DJICamera = drone.camera else {
                self.errorLabel.text = "Unable to detect Camera in
initDownload()"
                return
            }

            print("Successfully detected the camera")


            // change button title
```

```
            self.autoMap.setTitle("Auto Map Started...", for: .normal)


            // take picture when project starts
            self.takePic(camera: camera)

            // get last pic
            //var imageHolder: UIImage
            self.getLastPic(camera: camera)


            ////////////// MAY NOT NEED THIS ////////
            // delay one second to wait for camera
            /*DispatchQueue.main.asyncAfter(deadline: .now() + 5) { // change
1 to desired number of seconds

                // get last pic
                self.getLastPic(camera: camera)
            }*/

        }// end of if else
        self.autoMapStarted = !self.autoMapStarted
    }



/***** Get Last Picture *****/
    func getLastPic(camera: DJICamera) {
        // check if we can download images with the product
        if !camera.isMediaDownloadModeSupported() {
            self.errorLabel.text = "Product does not support media download
mode"
            return
        }

        let retry = RetryManager() // Run the same block multiple times if the
command has an error
        retry.runBlock(withRetries: 5) {
            // switch camera mode to allow for media downloads
            camera.setMode( .mediaDownload, withCompletion: {(error) in

                if error != nil {
                    self.errorLabel.text = ("\(error!.localizedDescription)")
                }
                else {

                    // get the media manager from the drone to gain access to
the files
                    let manager = camera.mediaManager!


                    manager.refreshFileList(of:
DJICameraStorageLocation.sdCard, withCompletion:  { (error) in

                        if error != nil {
```

```
                            self.errorLabel.text = "refresh error"
                            return
                        }
                        else {
                            // stop retrying
                            retry.stop()

                            // get list of files
                            guard let files = manager.sdCardFileListSnapshot()
else {
                                self.errorLabel.text = "No files to download"
                                return
                            }

                            let lastImage = files.last

                            lastImage?.fetchPreview(completion: {(error) in
                                if (error != nil){
                                    self.errorLabel.text = "fetchPreview
error: \(String(describing: error))"
                                }else{
                                    //self.testView.image = lastImage?.preview

                                    // if image retrieval was success
                                    let droneImage = lastImage!.preview

                                    // crop image into 5 x 5
                                    let imgArr = cropImage(imageHolder:
droneImage!, aImg: 5, dImg: 5)

                                    // classify images
                                    let probArr = classifyImages(imageArr:
imgArr!)

                                    var outputString = ""

                                    // iterator
                                    //var i = 1

                                    // print out results
                                    for k in probArr{
                                        for (key, value) in k{
                                            outputString += "\(key)
\(String(format: "%.3f",value)) "

                                            /*i = i + 1
                                            if (i % 5 == 0){
                                                outputString += "\n"
                                            }*/
                                        }
                                    }
                                    self.errorLabel.text = outputString
                                }
                            })

                        }
```

```
                    }) // end of file-refresh block
                } // end of if else
            }) // end of camera setMode block
        } // end of retry manager
    }


    /***** Start Project (taking photos every meter) *****/
    @IBAction func startProjectPushed(_ sender: UIButton) {

        let locationKey = DJIFlightControllerKey(param:
DJIFlightControllerParamAircraftLocation)

        guard let keyMngr = DJISDKManager.keyManager() else {
            print("Key Manager is nil")
            return
        }

        if self.projectStarted {
            // At anytime, you may stop listening to a key or to all keys for
a given listener
            keyMngr.stopListening(on: locationKey!, ofListener: self)

            // change button title
            self.startProject.setTitle("Start Project", for: .normal)

        } else {

            /****** setup camera *****/
            // get current product
            guard let drone = DJISDKManager.product() else {
                print("Product is connected but DJISDKManager.product is nil
when attempting to download media")
                return
            }

            // Get camera on drone
            guard  let camera: DJICamera = drone.camera else {
                print("Unable to detect Camera in initDownload()")
                return
            }

            print("Successfully detected the camera")

            // take picture when project starts
            self.takePic(camera: camera)

            /***** start listener for location changes ******/
            keyMngr.startListeningForChanges(on: locationKey!, withListener:
self, andUpdate: { (oldValue: DJIKeyedValue?, newValue: DJIKeyedValue?) in

                if(newValue != nil){

                    // get previous location
```

```
                    guard let oldLocation = oldValue?.value as? CLLocation
else {
                        print("Can't get old Location...")
                        return
                    }

                    // get new location
                    guard let location = newValue?.value as? CLLocation else {
                        print("Can't get Location...")
                        return
                    }

                    // distance traveled = diff from old location and new
location
                    let distance = location.distance(from: oldLocation)

                    // if distance is > one meter, take picture
                    if ((distance) > 1){
                        // take picture when project starts
                        camera.startShootPhoto(completion: { (error) in

                            if (error != nil) {
                                print("Shoot photo error:
\(error.debugDescription)")
                            }

                        }) // end camera
                    }// end dist if
                }else{

                    print("NewValue == nil")

                }// end if else

            })// end listener

            // change button title
            self.startProject.setTitle("Stop Project", for: .normal)
        }
        // change project started bool
        self.projectStarted = !self.projectStarted
    }


}
```

A.5 Retry Manager

```
//  RetryManager.swift
//
//  Created by Justin Miller on 4/12/18.
import Foundation

class RetryManager: NSObject {

    private let queue = DispatchQueue(label: "YOUR_LABEL_HERE")
    private let semaphore = DispatchSemaphore(value: 0)
    private var errorToCheck:NSError? = NSError()
    override init() {

    }

    open func runBlock(withRetries retry: Int, _ block: @escaping () -> Void)
{
        queue.async {
            var counter = 1
            while self.errorToCheck != nil {
                self.dispatchNow(block)
                self.semaphore.wait(timeout: .now() + 2)
                if counter >= retry {
                    break
                }
                counter += 1
            }
        }
    }

    open func proceed() {
        self.semaphore.signal()
    }

    open func stop() {
        errorToCheck = nil
    }

    fileprivate func dispatchNow(_ block: ()->()) {
        block()
    }
}
```

A.6 Mobile Split

```
//  cropImage.swift
//  AutoFlight
//
//  Created by Casey Lewis on 11/17/18.
//  Copyright © 2018 Casey Lewis. All rights reserved.
//

import Foundation
import UIKit


func cropImage(imageHolder: UIImage, aImg: Int, dImg: Int) -> Array<UIImage>?{
    // aImg is # images to cut across
    // dImg is # images to cut down

    // drones preview image width
    let rImgW = 960
    // drones preview image height
    let rImgH = 540
    // total images to make
    let ttlImg = aImg * dImg

    // set pixels to origin
    var xPixel = 0
    var yPixel = 0

    // image array
    var croppedImages: Array<UIImage> = []

    // iterators
    var imgSectW = 1

    for _ in 1...ttlImg{
        // so cropped image becomes whole again
        let imageCropped = imageHolder

        // add 1/x of width and height to current pixel
        let nextX = Int(xPixel + ((rImgW / aImg)))
        let nextY = Int(yPixel + ((rImgH / dImg)))

        let crop = CGRect(x: xPixel, y: yPixel, width: nextX, height: nextY)

        // move xPixel to new location
        xPixel = nextX

        //check if gone across image, if so increment yPixel and set xPixel to
0
        if imgSectW == aImg{
            imgSectW = 1
            xPixel = 0
            yPixel = nextY
        }else{
            imgSectW = imgSectW + 1
```

```
        }

        // Perform cropping in Core Graphics
        guard let cutImageRef: CGImage =
imageCropped.cgImage?.cropping(to:crop) else{
            return nil
        }

        // Return image to UIImage
        let croppedImage: UIImage = UIImage(cgImage: cutImageRef)

        croppedImages.append(croppedImage)
    }

    return croppedImages
}
```

A.7 Mobile Classify

```
//  classifyImages.swift
//  AutoFlight
//
//  This function accepts an array of UIImages as an param,
//  classifies those images and outputs an array of dictionaries
//  containing strings (labels) and doubles whose
//  value is 0 - 1 (being the confidence rating ofthe label)
//
//  Created by Casey Lewis on 11/17/18.
//  Copyright © 2018 Casey Lewis. All rights reserved.
//

import Foundation
import UIKit
import CoreML
import Vision
import ImageIO

func classifyImages(imageArr: Array<UIImage>) -> Array<[String:Double]>{

    // array of dicts
    var dictArr: Array<[String:Double]> = []

    // Load the ML model through its generated class
    guard let model = try? VNCoreMLModel(for: roadOC().model) else {
        fatalError("can't load Road CNN model")
    }

    // classify each image in the imageArr and place the results in the
dictArr
    for img in imageArr{
        dictArr.append(classifyImage(img: img, model: model))
    }

    return dictArr
}


func classifyImage(img:UIImage, model: VNCoreMLModel) -> [String:Double]{

    // DispatchGroup idea from
https://stackoverflow.com/questions/42484281/waiting-until-the-task-finishes
    let group2 = DispatchGroup()
    group2.enter()

    // empty dict
    var dict: [String: Double] = ["":0.0]


    // Create request for Vision Core ML model loaded (this is called in the
loop)
```

```
        let request = VNCoreMLRequest(model: model) { request, error in
            guard let results = request.results as? [VNClassificationObservation],
                let topResult = results.first else {
                    fatalError("unexpected result type from VNCoreMLRequest")
            }

            // using dispatch queue allows process to run in background
            // avoid deadlocks by not using .main queue here
            DispatchQueue.global().async {
                dict = [topResult.identifier: Double(topResult.confidence)]
                group2.leave()
            }

        }

        // Convert UIImage to CIImage to pass to the image request handler
        guard let ciImage = CIImage(image: img) else {
            fatalError("couldn't convert UIImage to CIImage")
        }
        // Run the Core ML classifier on global dispatch queue
        let handler = VNImageRequestHandler(ciImage: ciImage)
        DispatchQueue.global(qos: .userInteractive).async {
            do {
                try handler.perform([request])
            } catch {
                print(error)
            }
        }

        // waits until enter and leave are balanced
        group2.wait()

        return dict

    }
```

A.8 Turi Train

```
# TuriCreate put images into SFrame
# then train model
# (got most of this from
https://apple.github.io/turicreate/docs/userguide/image_classifier/)
#
# To see params in command line run:
#   $ python turiTrain.py -h
#
# Command line params:
#     "-id" which is the image directory and is required
#     "-dd" which is the destination directory to store sFrame and model and
is required
#   "-mn" which is the model name and is required

import turicreate as tc
import argparse
import os

def main(image_dir, dest_dir, model_name):

    # Load images (Note: you can ignore 'Not a JPEG file' errors)
    data = tc.image_analysis.load_images(image_dir, with_path=True)

    # From the path-name, create a label column
    data['label'] = data['path'].apply(lambda path: 'notRoad' if '/notRoad' in
path else 'road')


    # Save the data for future use
    data.save(os.path.join(dest_dir, model_name + '.sframe'))

    # Explore interactively
    data.explore()

    # Load the data
    data =  tc.SFrame(os.path.join(dest_dir, model_name + '.sframe'))

    # Make a train-test split
    train_data, test_data = data.random_split(0.8)

    # Create the model
    model = tc.image_classifier.create(train_data, target='label',
max_iterations=1000)

    # Save predictions to an SArray
    predictions = model.predict(test_data)

    # Evaluate the model and print results
    metrics = model.evaluate(test_data)
    print("Accuracy : \m%s" % metrics['accuracy'])
    print("Confusion Matrix : \n%s" % metrics['confusion_matrix'])
```

```
    # Save the model for later use in Turi Create
    model.save(os.path.join(dest_dir, model_name + '.model'))

    # Export for use in Core ML
    model.export_coreml(os.path.join(dest_dir, model_name +'.mlmodel'))


if __name__ == "__main__":
    parser = argparse.ArgumentParser()

    parser.add_argument(
        "-id",
        "--image-dir",
        type=str,
        required=True,
        help="REQUIRED. The directory where all images are stored."
    )
    parser.add_argument(
        "-dd",
        "--dest-dir",
        type=str,
        required=True,
        help="""\
             REQUIRED. The directory where the newly created sFrame will go.
             Also, the model will go here.\
             """
    )
    parser.add_argument(
        "-mn",
        "--model-name",
        type=str,
        required=True,
        help="REQUIRED. The name to save the model as."
    )

    args = parser.parse_args()

    main(**vars(args))
```

A.9 Turi Auto Pick

```
# TuriCreate puts numpy arrays into SFrame
# then train model
# (got most of this from
https://apple.github.io/turicreate/docs/userguide/image_classifier/)
#
# To see params in command line run:
#   $ python turiTrainAuto.py -h
#
# Command line params:
#      "-af" which is the file that holds numpy array and is required
#      "-dd" which is the destination directory to store sFrame and model and is
required
#   "-mn" which is the model name and is required

import turicreate as tc
import argparse
import os
import numpy as np

def main(array_file, dest_dir, model_name):

    # Load numpy array
    values = np.load(os.path.join(array_file)).item()
    #print(values)


    for d in range(0,25):
        exec 'column%s = []' %(d)

    labels = []
    #column = []
    for k, v in values.iteritems():
        key = ''.join([i for i in k if not i.isdigit()]) # remove ints from string
        #print(key, v)
        labels.append(key)
        #column.append(v)
        for i in range(0,25):
            exec 'column%s.append(float(v[%s]))' %(i, i)


    # create sframe
    data = tc.SFrame({'c0':column0, 'c1':column1, 'c2':column2, 'c3':column3,
'c4':column4,
        'c5':column5, 'c6':column6, 'c7':column7, 'c8':column8, 'c9':column9,
'c10':column10,
        'c11':column11, 'c12':column12, 'c13':column13, 'c14':column14,
'c15':column15, 'c16':column16,
        'c17':column17, 'c18':column18, 'c19':column19, 'c20':column20,
'c21':column21, 'c22':column22,
        'c23':column23, 'c24':column24,'labels':labels})

    # Save the sframe for future use
    data.save(os.path.join(dest_dir, model_name + '.sframe'))
```

```
    # Explore interactively
    #data.explore()

    # Make a train-test split
    train_data, test_data = data.random_split(0.8)

    '''
    numeric_features = ['c0','c1','c2','c3','c4',
                    'c5','c6','c7','c8','c9',
                    'c10','c11','c12','c13',
                    'c14','c15','c16','c17',
                    'c18','c19','c20','c21',
                    'c22','c23','c24']
    '''

    # Make a train-test split
    train_data, test_data = data.random_split(0.8)

    # Create a model automatically based on your data.
    model = tc.classifier.create(train_data, target='labels',
                                 features = ['c0','c1','c2','c3','c4',
                                             'c5','c6','c7','c8','c9',
                                             'c10','c11','c12','c13',
                                             'c14','c15','c16','c17',
                                             'c18','c19','c20','c21',
                                             'c22','c23','c24'])



    # Save predictions to an SArray
    predictions = model.predict(test_data)

    # Evaluate the model and print results
    metrics = model.evaluate(test_data)
    print("Accuracy : \m%s" % metrics['accuracy'])
    print("Confusion Matrix : \n%s" % metrics['confusion_matrix'])

    # Save the model for later use in Turi Create
    model.save(os.path.join(dest_dir, model_name + '.model'))

    # Export for use in Core ML
    model.export_coreml(os.path.join(dest_dir, model_name +'.mlmodel'))

if __name__ == "__main__":
    parser = argparse.ArgumentParser()

    parser.add_argument(
        "-af",
        "--array-file",
        type=str,
        required=True,
        help="REQUIRED. The file where all numpy arrays are stored."
    )
    parser.add_argument(
        "-dd",
        "--dest-dir",
        type=str,
        required=True,
```

```
    help="""\
          REQUIRED. The directory where the newly created sFrame will go.
          Also, the model will go here.\
          """
)
parser.add_argument(
    "-mn",
    "--model-name",
    type=str,
    required=True,
    help="REQUIRED. The name to save the model as."
)

args = parser.parse_args()

main(**vars(args))
```

A.10 Naïve Bayes CPP

```
// BayesForAutoFlight.cpp : Defines the entry point for the console
application.
//
//
// .exe filepathToArrayWLabels.txt useGSMBool epsilon
// Need to create command line param for amt of class labels


#include <fstream>
#include <iostream>
#include <string>
#include <vector>
#include "Classifier.h"
#include "GramSchmidtAngle.h"

using namespace std;

void putFileIntoVectors(ifstream &inFile, vector<uint8_t>
&classificationTrain, vector<vector< double >> &trainValues,
	vector<uint8_t> &classificationTest, vector<vector< double >>
&testValues);

int main(int argc, const char *argv[])
{
	if (argc == 1)
	{
		cerr << "Error: Did not include file path." << endl;
		exit(-1);
	}
	else if (argc > 4) // too many command line arg
	{
		cerr << "Error: Entered too many command line arguments." <<
endl;
		exit(-1);
	}

	ifstream inputFile;

	inputFile.open(argv[1]);



	// if the file opened
	if (inputFile)
	{
		/* creating vectors for data and labels*/
		// A vector that holds the classification of the tuples
		vector<uint8_t> classTrain;
		vector<uint8_t> classTest;
		// Vector of vectors...
		vector<vector< double >> trainValues(9);
```

```
		// Vector of vectors...
		vector<vector< double >> testValues(9);
		vector< double > testTuple(0);

		bool chansSelected[25];

		/* for accuracies */
		int right = 0;
		int wrong = 0;
		int total = 0;
		int confuMatrix[7][7] = { 0 };

		putFileIntoVectors(inputFile, classTrain, trainValues, classTest,
testValues);

		int attributesInCol = trainValues[0].size();

		bool isGSM = (argc > 2 && string(argv[2]) == "true");

		if (isGSM)
		{
			int epsilon = atoi(argv[3]);
			cout << "epsilon " << epsilon << endl;
			GramSchmidtAngle<double, uint8_t> gsa;
			std::vector<bool> columnsDeleted =
gsa.performGramSchmidtAngleMethod(trainValues, attributesInCol,
trainValues.size(), classTrain, epsilon);

			for (int i = 0; i < columnsDeleted.size(); i++)
			{
				cout << columnsDeleted[i] << endl;
				chansSelected[i] = !columnsDeleted[i];
			}
		}
		else
		{
			for (int i = 0; i < 9; i++)
			{
				chansSelected[i] = 1;
			}
		}

		int chanCount = 0;

		for (int i = 8; i >= 0; i--) {
			if (!chansSelected[i]) {
				trainValues.erase(trainValues.begin() + i);
				testValues.erase(testValues.begin() + i);
			}
		}
		// train naive bayes
		auto trainClassifer =
priorProbabilitiesMeansAndVariances<uint8_t(7)>(trainValues, classTrain);

		size_t dimensionCount = testValues.size();
```

```
		size_t tupleCount = testValues[0].size();

		// Put per-class sums of attribute values in means and build
classTupleCounts
		for (int tuple = 0; tuple < tupleCount; tuple++) {
			for (int dimension = 0; dimension < dimensionCount;
dimension++) {
				//cout << testValues[dimension][tuple] << ", ";
				// put values into tuple
				testTuple.push_back(testValues[dimension][tuple]);
			}

			// classify tuple
			auto classLabel =
classifyGivenStats<uint8_t(7)>(testTuple, trainClassifer);


			confuMatrix[classLabel][classTest[tuple]]++;

			(classTest[tuple] != classLabel) ? wrong++ : right++;
			total++;

			/*
			if ((classTest[tuple] - classLabel) > 4)
			{
				cout << "tuple: ";
				for (int i = 0; i < testTuple.size(); i++)
				{
					cout << testTuple[i] << ", ";
				}
				// print label of test tuple
				cout << "\nlabel: " <<
static_cast<int>(classTest[tuple]) << endl;
				// print classified tuple
				cout << " Classified Label: " <<
static_cast<int>(classLabel) << endl;
			}
			*/
			testTuple.clear();
		}

		int count = 0;
		double weightedAmt = 0;
		double matrixValue = 0;
		int count2 = 0;
		double weightedAmt2 = 0;
		ofstream outputFile;
		// get the name of the file (minus the .txt)
		string outFilename(argv[1]);
		outFilename.erase(outFilename.length() - 4);
		outFilename += (isGSM) ? "_" + string(argv[3]) + "_GSM" : "";
		outFilename += "output.txt";

		//cout << endl << outFilename << endl << endl;
```

```
		outputFile.open(outFilename);

		cout << "wrong: " << wrong << endl;
		outputFile << "wrong: " << wrong << endl;
		cout << "right: " << right << endl;
		outputFile << "right: " << right << endl;
		cout << "accuracy: " << static_cast<double>(right) / total << endl;
		outputFile << "accuracy: " << static_cast<double>(right) / total << endl;


		// print confusion matrix and save to file
		cout << endl << "L\t" << "LF\t" << "F\t" << "RF\t" << "R\t" << "AU\t" << "NR\t" << endl;
		cout << "-------------------------------------------------------" << endl;
		for (int i = 0; i < 7; i++)
		{
			for (int j = 0; j < 7; j++)
			{
				cout << confuMatrix[i][j] << "\t";
				outputFile << confuMatrix[i][j] << "\t";
				matrixValue = ((1 - (abs(i - j) / static_cast<double>(4))) * confuMatrix[i][j]);
				weightedAmt += (matrixValue > 0) ? matrixValue : 0;
				count += confuMatrix[i][j];
			}
			cout << endl;
			outputFile << endl;
		}
		cout << "this weighted Amount is 1 .75 .5 0 0 0 0(from the center diagonal" << endl;
		outputFile << "this weighted Amount is 1 .75 .5 0 0 0 0(from the center diagonal" << endl;
		cout << "conMat weighted Amt: " << weightedAmt << endl;
		outputFile << "conMat weighted Amt: " << weightedAmt << endl;
		cout << "count: " << count << endl;
		outputFile << "count: " << count << endl;
		cout << "new Accuracy: " << weightedAmt / static_cast<double>(count) << endl;
		outputFile << "new Accuracy: " << weightedAmt / static_cast<double>(count) << endl;


		// print confusion matrix
		cout << endl << "L\t" << "LF\t" << "F\t" << "RF\t" << "R\t" << "AU\t" << "NR\t" << endl;
		cout << "-------------------------------------------------------" << endl;
		for (int i = 0; i < 7; i++)
		{
			for (int j = 0; j < 7; j++)
			{
				cout << confuMatrix[i][j] << "\t";
```

```
                        outputFile << confuMatrix[i][j] << "\t";
                        matrixValue = ((1 - (abs(i - j) /
static_cast<double>(2))) * confuMatrix[i][j]);
                        weightedAmt2 += (matrixValue > 0) ? matrixValue : 0;
                        count2 += confuMatrix[i][j];
                  }
                  cout << endl;
                  outputFile << endl;
            }
            cout << "this weighted Amount is 1 .5 0 0 0 0 0(from the center
diagonal" << endl;
            outputFile << "this weighted Amount is 1 .5 0 0 0 0 0(from the
center diagonal" << endl;
            cout << "conMat weighted Amt: " << weightedAmt2 << endl;
            outputFile << "conMat weighted Amt: " << weightedAmt2 << endl;
            cout << "count: " << count2 << endl;
            outputFile << "count: " << count2 << endl;
            cout << "new Accuracy: " << weightedAmt2 /
static_cast<double>(count2);
            outputFile << "new Accuracy: " << weightedAmt2 /
static_cast<double>(count2);

            outputFile.close();
      }

      //cin.get();

    return 0;
}


// put the contents of the filie into vectors
void putFileIntoVectors(ifstream &inFile, vector<uint8_t>
&classificationTrain, vector<vector< double >> &trainValues,
      vector<uint8_t> &classificationTest, vector<vector< double >>
&testValues)
{
      int bufferIndex,
            valueIndex,
            verseNumber = 0,
            ssLength = 0,
            tuples = 1;
      string someString, label;
      bool labelFound = false,
            tupleNumTen = false;



      // Read a line from the file
      getline(inFile, someString, '\n');

      /*while last read operation was
      successful, continue */
      while (inFile)
      {
```

```
		tupleNumTen = (tuples % 10 == 0);

		// get length of the string
		ssLength = someString.length();

		valueIndex = 0;
		bufferIndex = 0;

		for (int i = 0; i < ssLength; i++)
		{
			/* get everything on left side of colon (which is label)
*/
			if (someString[i] == ':')
			{
				labelFound = true;
			}
			if (!labelFound) // if colon hadn't been found, continue
			{
				continue;
			}
			else if(someString[i] == ':')
			{
				// get everything on left side of colon (which is
label)
				label = someString.substr(bufferIndex, i -
bufferIndex);
				bufferIndex = i + 1;
			}
			else if (someString[i] == ',') // if comma (don't worry
about end of string as last ele is a comma)
			{
				if (tupleNumTen)
				{
					// get everything on left side of comma
(which is value)

	testValues[valueIndex].push_back(stod(someString.substr(bufferIndex, i -
bufferIndex)));
				}
				else
				{
					// get everything on left side of comma
(which is value)

	trainValues[valueIndex].push_back(stod(someString.substr(bufferIndex, i
- bufferIndex)));
				}
				bufferIndex = i + 1;
				valueIndex++;
			}
			else // is a number, just skip
			{
				;
			}
```

```
}

if (tupleNumTen)
{
	// map label to int
	if (label == "L")
	{
		classificationTest.push_back(0);
	}
	else if (label == "LF")
	{
		classificationTest.push_back(1);
	}
	else if (label == "F")
	{
		classificationTest.push_back(2);
	}
	else if (label == "RF")
	{
		classificationTest.push_back(3);
	}
	else if (label == "R")
	{
		classificationTest.push_back(4);
	}
	else if (label == "AU")
	{
		classificationTest.push_back(5);
	}
	else if (label == "NR")
	{
		classificationTest.push_back(6);
	}
}
else
{
	// map label to int
	if (label == "L")
	{
		classificationTrain.push_back(0);
	}
	else if (label == "LF")
	{
		classificationTrain.push_back(1);
	}
	else if (label == "F")
	{
		classificationTrain.push_back(2);
	}
	else if (label == "RF")
	{
		classificationTrain.push_back(3);
	}
	else if (label == "R")
	{
```

```
                        classificationTrain.push_back(4);
                }
                else if (label == "AU")
                {
                        classificationTrain.push_back(5);
                }
                else if (label == "NR")
                {
                        classificationTrain.push_back(6);
                }
        }


        // Read next item
        getline(inFile, someString, '\n');
        tuples++;
    }
}
```

A.11 Naïve Bayes Classifier

```
#pragma once

#define _USE_MATH_DEFINES
#include <math.h>
#include <vector>
#include <tuple>
#include <array>

using namespace std;

typedef uint8_t u8;

// This function processes the data and makes the statistics needed for
classification
template<u8 classCount>
tuple<array<double, classCount>, double(*)[classCount], double(*)[classCount]>
priorProbabilitiesMeansAndVariances(vector<vector<double>>& data, vector<u8>&
classes);

// This function uses those stats to classify new tuples
template<u8 classCount>
u8 classifyGivenStats(vector<double>& thisTuple, tuple<array<double,
classCount>, double(*)[classCount], double(*)[classCount]>&
priorProbabilitiesMeansAndVariances);




double square(double x) {
	return x*x;
}

double normalDistribution(double x, double mean, double variance) {
	return 1 / sqrt(M_2_PI*variance) * exp(-square(x - mean) / (2 *
variance));
}

template<u8 classCount>
void zero(double arr[][classCount], size_t dimensionCount) {
	for (int i = 0; i < dimensionCount; i++) {
		for (int j = 0; j < classCount; j++) {
			arr[i][j] = 0;
		}
	}
}

// Combined for optimization
template<u8 classCount>
tuple<array<double, classCount>, double(*)[classCount], double(*)[classCount]>
priorProbabilitiesMeansAndVariances(vector<vector<double>>& data, vector<u8>&
classes) {
```

```
	size_t dimensionCount = data.size();
	size_t tupleCount = data[0].size();
	int classTupleCounts[classCount] = {};
	auto means = new double[dimensionCount][classCount];
	zero(means, dimensionCount);
	// Put per-class sums of attribute values in means and build
classTupleCounts
	for (int tuple = 0; tuple < tupleCount; tuple++) {
		classTupleCounts[classes[tuple]]++;
		for (int dimension = 0; dimension < dimensionCount; dimension++)
{
			means[dimension][classes[tuple]] +=
data[dimension][tuple];
		}
	}
	array<double, classCount> priorProbabilities = {};
	// Divide per-class sums of attribute values by per-class tuple count to
get means
	// Normalize class tuple counts to get prior probabilities
	for (int clazz = 0; clazz < classCount; clazz++) {
		for (int dimension = 0; dimension < dimensionCount; dimension++)
{
			means[dimension][clazz] /= classTupleCounts[clazz];
		}
		priorProbabilities[clazz] = double(classTupleCounts[clazz]) /
tupleCount;
	}
	auto variances = new double[dimensionCount][classCount];
	zero(variances, dimensionCount);
	// Calculate variances
	for (int tuple = 0; tuple < tupleCount; tuple++) {
		for (int dimension = 0; dimension < dimensionCount; dimension++)
{
			variances[dimension][classes[tuple]] +=
square(data[dimension][tuple] - means[dimension][classes[tuple]]);
		}
	}
	for (int clazz = 0; clazz < classCount; clazz++) {
		for (int dimension = 0; dimension < dimensionCount; dimension++)
{
			variances[dimension][clazz] /= classTupleCounts[clazz] -
1;
		}
	}
	return make_tuple(priorProbabilities, means, variances);
}

template<u8 classCount>
u8 classifyGivenStats(vector<double>& thisTuple, tuple<array<double,
classCount>, double(*)[classCount], double(*)[classCount]>&
priorProbabilitiesMeansAndVariances) {
	auto& priorProbabilities = get<0>(priorProbabilitiesMeansAndVariances);
	auto means = get<1>(priorProbabilitiesMeansAndVariances);
	auto variances = get<2>(priorProbabilitiesMeansAndVariances);
	size_t dimensionCount = thisTuple.size();
```

```
	u8 bestClass;
	double bestClassProbability = -INFINITY;
	for (u8 currentClass = 0; currentClass < classCount; currentClass++) {
		double currentClassProbability =
priorProbabilities[currentClass];
		for (int dimension = 0; dimension < dimensionCount; dimension++)
{
			currentClassProbability *=
normalDistribution(thisTuple[dimension], means[dimension][currentClass],
variances[dimension][currentClass]);
		}
		if (currentClassProbability > bestClassProbability) {
			bestClass = currentClass;
			bestClassProbability = currentClassProbability;
		}
	}
	return bestClass;
}
```

A.12 GS Method

```
#ifndef GRAMSCHMIDTANGLE_H
#define GRAMSCHMIDTANGLE_H

#include <vector>
#include <cmath>
#include "Entropy.h"

#include <iostream>

/*************************************************************************
***
*  The Gram Schmidt method is ran on the attribute columns to determine the
*
*  dependence of the columns. This dependence is measured by the cosine found
*
*  from the length of difference between vi and the sum of all the projections
*
*  of the vector, vi, onto EACH basis / length of vi
            *
*  Template is the type in matrix
                                  *
*  ASSUMPTIONS:
                                                *
*     1. All columns have the same number of attributes
                    *
*     2. All attributes are of a numeric data type, i.e. double, int, short,
      *
*           float...
                                                *
*     By Casey Lewis, 2017
                                          *
*************************************************************************
**/

template<class T, class T2>
class GramSchmidtAngle
{
public:
      GramSchmidtAngle() {} // default Constructor

      std::vector<bool> performGramSchmidtAngleMethod(const std::vector<
std::vector<T> > &matrix,
            const int &tuples, const int &columns, const std::vector< T2 >
&classes, const int epsilon)
      {

            // amount of columns in the original matrix (and thus all
vectors)
            int amountOfCol = columns;
            int amountOfTuples = tuples;
```

```
		/* find the entropy for all vectors to determine which vector to
make as
		basis1. This is because each vector is projected onto basis1,
want to start with
		the best. ENTROPY BINS THE DATA*/
		Entropy<T, T2> findEntropy;
		int startingVector =
findEntropy.findColumnWithHighestEntropy(matrix, amountOfTuples, amountOfCol,
classes);

		/* vectorOfProjectionsOfVi is the sum of all the projections of
		the vector, vi, onto EACH basis */
		std::vector<double> vectorOfProjectionsOfVi(amountOfTuples, 0);
		/* distanceDifference holds the difference between vi and the
		vectorOfProjectionsOfVi*/
		std::vector<double> distanceDifference(amountOfTuples);

		// This vector holds the columns to be deleted
		std::vector<bool> columnNumsDeleted(amountOfCol);

		// first vector, use the vector and its length as basis1
		// add first vector as basis
		matrixOfBases.push_back(matrix[startingVector]);

	matrixOfBasesLength.push_back(findLengthOfVector(matrixOfBases[0]));

		// add false to the columnNumsDeleted
		columnNumsDeleted[startingVector] = false;

		cout << "col" << startingVector << endl;

		// set amount of vectors in matrixOfBases
		int amountOfBases = 1;

		/* for each vector in matrix find the cosine of the angle
produced
		by the projection of the vector onto the bases */
		for (int i = (startingVector + 1) % amountOfCol; i !=
startingVector; i = (i + 1) % amountOfCol)
		{
			cout << "col" << i << endl;

			/* find the vector that is the sum of all the projections
of
			the vector, vi, onto EACH basis */
			for (int basis = 0; basis < amountOfBases; basis++)
			{
				projectOnto(matrix, i, basis,
vectorOfProjectionsOfVi, amountOfTuples);
			}

			/* find the vector that holds the difference between vi
and the
			vectorOfProjectionsOfVi*/
			for (int entry = 0; entry < amountOfTuples; entry++)
```

```
                    {
                          distanceDifference[entry] = (matrix[i][entry] -
vectorOfProjectionsOfVi[entry]);
                    }

                    double lengthOfdiff =
findLengthOfVector(distanceDifference);

                    /* find the cosine of theta, which is the length of the
difference between vi and the
                    vectorOfProjectionsOfVi (distanceDifference) divided by
the length of vi */
                    double cosTheta = cos(lengthOfdiff /
findLengthOfVector(matrix[i]));

                    cout << "cosTheta (abs val) " << abs(cosTheta * 100)/100
<< endl;

                    if (abs(cosTheta * 100) > epsilon)
                    {
                          // add true to the columnNumsDeleted
                          columnNumsDeleted[i] = true;
                    }
                    else // add the vectorDistanceDifference and its length as
the next basis
                    {
                          matrixOfBases.push_back(distanceDifference);
                          matrixOfBasesLength.push_back(lengthOfdiff);
                          // add false to the columnNumsDeleted
                          columnNumsDeleted[i] = false;
                          // increment amount of vectors in matrixOfBases
                          amountOfBases++;
                    }
             }
             return columnNumsDeleted;
      }

private:

      // matrixOfBases holds all the basis vectors
      std::vector< std::vector<double> > matrixOfBases;
      // matrixOfBasesLength holds all the lengths of the basis vectors
      std::vector<double> matrixOfBasesLength;

      /* the length of the column equals the square root of the sum
      of the square entries, sqrt(e1^2 + e2^2 +...+eN^2) */
      double findLengthOfVector(const std::vector<double> &aVector)
      {
             double sumOfSquares = 0;

             for (int j = 0; j < aVector.size(); j++)
             {
                    sumOfSquares += pow(aVector[j], 2.0);
             }
```

```
		return sqrt(sumOfSquares);
	}


	/* for the ith vector (or ith column of the matrix) find the projection
	of said column onto each basis */
	void projectOnto(const std::vector< std::vector<T> > &matrix, const int
&column, const int &basis,
		std::vector<double> &vectorOfProjections, int amtOfTuples)
	{
		double dotProductOfViAndB = 0;
		double scalar = 0;


		// projection of vi onto basis =  (vi dot bi) * bi
		for (int entry = 0; entry < amtOfTuples; entry++)
		{
			dotProductOfViAndB += (matrix[column][entry] *
matrixOfBases[basis][entry]);
		}

		scalar = (dotProductOfViAndB / pow(matrixOfBasesLength[basis],
2.0));

		// multiply the scalar by the basis
		for (int entry = 0; entry < amtOfTuples; entry++)
		{
			vectorOfProjections[entry] += (scalar *
matrixOfBases[basis][entry]);
		}
	}
};

#endif // !GRAMSCHMIDTANGLE_H
```

B. Documentation

B.1 TensorFlow Object Detection API Documentation

Get object detection API to work in windows 10- Tensorflow

By Casey Lewis

*** NOTE: every time you change an environment var, make sure to exit command prompt and open new one ***

*** '>' denotes command line

**** TO DO: in models\research\ can run >python setup.py and should make it so

you do not have to move folders into models\research\ ****

Install OpenCV

Download python 3.6.5 64 bit (tensorflow only works up to 3.6)

select add to PATH

Install Now

*** if wanting to install tensorflow-gpu then do 'install cuda' and 'install cuDNN' else skip ***

As of May 31, 2018, Tensorflow worked best with CUDA 9 and cuDNN 7

install cuda

go to https://developer.nvidia.com/cuda-toolkit-archive

select CUDA toolkit 9.0

download AND install Base Installer

THEN download and install Patch's in sequential order

add C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\bin

and C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\lib\x64 to PATH

Wouldn't hurt to restart computer

install cuDNN

go to https://developer.nvidia.com/cudnn

click Download cuDNN (may have to setup account)

select I agree to terms

select archive

select Download cuDNN v7.0.5 for CUDA 9.0 -> cuDNN v7.0.5 Library for Windows 10

from nvidia:

The following steps describe how to build a cuDNN dependent program. In the following sections:

your CUDA directory path is referred to as C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0

your cuDNN directory path is referred to as <installpath>

1. Navigate to your <installpath> directory containing cuDNN.
2. Unzip the cuDNN package.

   cudnn-9.0-windows7-x64-v7.zip

   or

   cudnn-9.0-windows10-x64-v7.zip
3. Copy the following files into the CUDA Toolkit directory.

   a) Copy <installpath>\cuda\bin\cudnn64_7.dll

      to

      C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\bin.

   b) Copy <installpath>\cuda\ include\cudnn.h

      to

      C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\include.

   c) Copy <installpath>\cuda\lib\x64\cudnn.lib

      to

      C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\lib\x64.

Set environment variables

> SET PATH=C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\bin;%PATH%

> SET PATH=C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\extras\CUPTI\libx64;%PATH%

> SET PATH=C:\tools\cuda\bin;%PATH%

Install virtualenv:

> pip install --upgrade virtualenv

Create an environment folder:

> cd Documents

> mkdir venvironments

Create an environment:

cd into the environment folder

> cd venvironments

create the environment

> mkdir target_directory

> virtualenv --system-site-packages -p python3 target_directory

Activate the environment:

To use the environment, we need to activate it

> target_directory\Scripts\activate

Within the virtual environment, you can use the command python instead of python3. Same goes with pip and pip3.

Deactivate the environment:

> deactivate

install tensorflow

> pip install --upgrade tensorflow

or

> pip install --upgrade tensorflow-gpu

install dependencies

> pip install numpy

> pip install Cython

> pip install pillow

> pip install lxml

> pip install jupyter

> pip install matplotlib

> pip install pandas

clone object detection api repositor

> cd "to wherever"

> git clone https://github.com/tensorflow/models.git

add PYTHONPATH to environment variables

open Environment Variables

under User Variables select New...

type PYTHONPATH for variable name

type path to models\research\ for variable value

type path to models\research\slim for variable value

select Ok

add %PYTHONPATH% to PATH

go to where you cloned models and create objectDetect folder

ex:

Documents

-> models

-> objectDetect

install protoc

go to https://github.com/google/protobuf/releases

Download protoc-3.4.0-win32.zip (the newer versions don't work in windows)

unzip

copy protoc.exe in protoc-3.4.0-win32 -> bin

open \models\research\

paste protoc.exe

> cd to 'wherever'\models\research\

> protoc object_detection/protos/*.proto --python_out=.

test if install worked

in models\research\

> python object_detection/builders/model_builder_test.py

**** If you are using tensorflow-gpu and get this error:

ImportError: DLL load failed: A dynamic link library (DLL) initialization routine failed.

Then go back a version in tensorflow by using:

>pip install --upgrade --ignore-installed tensorflow-gpu==1.5

****

create images

Download prebuilt binaries of labelImg from https://github.com/tzutalin/labelImg

This will create bounding boxes of images in pascal voc format

In objectDetect folder (created above), create directory 'images'

Put images to classify in images directory

When using labelImg save xml (with coordinates of bounding boxes) in images directory

Within images directory create train and test folder:

objectDetect

->images

->train

->test

COPY about 5-10% of images along with matching xml annotations into test and COPY the rest into train

change xml to tfrecords

Create 'data' directory in objectDetect

objectDetect

->images

->data

go to https://github.com/datitran/raccoon_dataset

Copy 'xml_to_csv.py' and 'generate_tfrecord.py' and put them in the objectDetect folder

In 'xml_to_csv.py' change main function to:

```
def main():
    for directory in ['train', 'test']:
        image_path = os.path.join(os.getcwd(), 'images/{}'.format(directory))
        xml_df = xml_to_csv(image_path)
        xml_df.to_csv('data/{}_labels.csv'.format(directory), index=None)
        print('Successfully converted xml to csv.')
```

> cd 'wherever'\objectDetect

> python xml_to_csv.py

In 'generate_tfrecord.py':

On line #29, change row_label == 'your label'

If multiple records, make elif row_label == 'your 2nd label' return 2

Return 0 is a placeholder, don't use it

Also, the command line command is in the comments at the top,

> cd 'wherever'\objectDetect

> python generate_tfrecord.py --csv_input=data/train_labels.csv --output_path=data/train.record

> python generate_tfrecord.py --csv_input=data/test_labels.csv --output_path=data/test.record

get model and config file

pre-trained models can be found at:

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

their corresponding config file can be found:

'wherever'\models\research\samples\configs

put the model and config file into 'wherever'\objectDetect

use 7-zip TWICE to extract model from .tar and .gz

In config file:

Under model funct change

num_classes: "your num of classes"

Under train_config change

fine_tune_checkpoint: "modelfolder/model.ckpt"

Under train_input_reader

input_path: "data/train.record"

label_map_path: "data/object-detection.pbtxt"

Under eval_input_reader

input_path: "data/test.record"

label_map_path: "data/object-detection.pbtxt"

Move config file into training directory

objectDetect

->training

->"whatever config file you picked"

In data folder create 'object-detection.pbtxt"

objectDetect

->data

->object-detection.pbtxt

Inside object-detection.pbtxt

item {

id: 1

name: "your label here"

}

move folders to api folder

from "wherever"\objectDetect

Copy directories:

data

"whatever model you picked"

images

training

Paste these into "wherever"\models\research\object_detction

start training

> cd 'wherever'\models\research\object_detction

> python train.py --logtostderr --train_dir=training\ --pipeline_config_path=training\"whatever config file you picked".config

*** if you get error:

ValueError: Tried to convert 't' to a tensor and failed.

Error: Argument must be a dense tensor: range(0, 3) - got shape [3], but wanted []

go into "wherever"\models\research\object_detection\utils\learning_schedules.py

and change:

rate_index = tf.reduce_max(tf.where(tf.greater_equal(global_step, boundaries),

range(num_boundaries),

[0] * num_boundaries))

into

rate_index = tf.reduce_max(tf.where(tf.greater_equal(global_step, boundaries),

list(range(num_boundaries)),

[0] * num_boundaries))

C. Pix4D Data

Picture Name	Lat	Long	Altitude	Distance From Last Pic (meter)
DJI_0801.JPG	43.93856228	-115.9714383	1486.114014	
DJI_0802.JPG	43.93858978	-115.9714326	1486.313965	3.09
DJI_0803.JPG	43.93872422	-115.9714416	1486.313965	14.97
DJI_0804.JPG	43.93889028	-115.9714519	1486.41394	18.48
DJI_0805.JPG	43.93908031	-115.971462	1486.41394	21.15
DJI_0806.JPG	43.93927228	-115.9714711	1486.41394	21.36
DJI_0807.JPG	43.93946653	-115.9714741	1486.313965	21.60
DJI_0808.JPG	43.93975419	-115.9714729	1486.313965	31.99
DJI_0809.JPG	43.93994389	-115.9714711	1486.514038	21.09
DJI_0810.JPG	43.94012772	-115.9714705	1486.514038	20.44
DJI_0811.JPG	43.94031578	-115.971467	1486.41394	20.91
DJI_0812.JPG	43.94050819	-115.9714612	1486.313965	21.40
DJI_0813.JPG	43.94079875	-115.9714514	1486.213989	32.32
DJI_0814.JPG	43.94099481	-115.9714399	1486.313965	21.82
DJI_0815.JPG	43.94118689	-115.9714177	1486.313965	21.43
DJI_0816.JPG	43.94130853	-115.9712984	1486.213989	16.56
DJI_0817.JPG	43.94132069	-115.9711058	1486.114014	15.49
DJI_0818.JPG	43.94128175	-115.9708601	1486.213989	20.14
DJI_0819.JPG	43.94121411	-115.970461	1486.313965	32.82
DJI_0820.JPG	43.94110964	-115.9702274	1486.213989	22.02
DJI_0821.JPG	43.94098131	-115.9700166	1486.114014	22.10
DJI_0822.JPG	43.94080414	-115.9699323	1486.014038	20.82
DJI_0823.JPG	43.9406015	-115.9699434	1486.014038	22.55
DJI_0824.JPG	43.94029433	-115.9699912	1486.213989	34.37
DJI_0825.JPG	43.94009114	-115.9700197	1486.213989	22.71
DJI_0826.JPG	43.93989564	-115.9700394	1486.213989	21.80
DJI_0827.JPG	43.93969794	-115.9700464	1486.213989	21.99
DJI_0828.JPG	43.93950475	-115.9700475	1486.213989	21.48
DJI_0829.JPG	43.93921861	-115.9700461	1486.213989	31.82
DJI_0830.JPG	43.93902678	-115.9700373	1486.313965	21.34
DJI_0831.JPG	43.93883694	-115.9700276	1486.213989	21.12
DJI_0832.JPG	43.93864511	-115.9700211	1486.313965	21.34
DJI_0833.JPG	43.93845617	-115.9700166	1486.213989	21.01
DJI_0834.JPG	43.93826183	-115.9700166	1486.213989	21.61
DJI_0835.JPG	43.93797778	-115.9700043	1486.213989	31.60
DJI_0836.JPG	43.93784994	-115.9698949	1486.313965	16.70

DJI_0837.JPG	43.93780556	-115.9697205	1486.313965	14.81
DJI_0838.JPG	43.93780283	-115.9694972	1486.514038	17.88
DJI_0839.JPG	43.93781064	-115.9692534	1486.514038	19.53
DJI_0840.JPG	43.93782417	-115.9688702	1486.514038	30.72
DJI_0841.JPG	43.93784453	-115.9686174	1486.514038	20.37
DJI_0842.JPG	43.93792839	-115.9684763	1486.41394	14.65
DJI_0843.JPG	43.93805242	-115.9684642	1486.313965	13.82
DJI_0844.JPG	43.93820919	-115.9684853	1486.41394	17.51
DJI_0845.JPG	43.93838264	-115.9684893	1486.41394	19.29
DJI_0846.JPG	43.93864817	-115.968509	1486.41394	29.57
DJI_0847.JPG	43.93882475	-115.9685297	1486.41394	19.70
DJI_0848.JPG	43.93900606	-115.9685383	1486.313965	20.17
DJI_0849.JPG	43.93918581	-115.9685162	1486.313965	20.07
DJI_0850.JPG	43.93936669	-115.9685014	1486.313965	20.15
DJI_0851.JPG	43.93963978	-115.9684968	1486.313965	30.37
DJI_0852.JPG	43.93982419	-115.9684919	1486.313965	20.51
DJI_0853.JPG	43.94001011	-115.9684874	1486.313965	20.68
DJI_0854.JPG	43.94019708	-115.9684831	1486.313965	20.79
DJI_0855.JPG	43.94038461	-115.9684816	1486.313965	20.85
DJI_0856.JPG	43.94057158	-115.9684807	1486.313965	20.79
DJI_0857.JPG	43.940761	-115.9684792	1486.213989	21.06
DJI_0858.JPG	43.94104169	-115.9684431	1486.213989	31.35
DJI_0859.JPG	43.94115553	-115.9682951	1486.114014	17.34
DJI_0860.JPG	43.94118253	-115.9680768	1486.114014	17.73
DJI_0861.JPG	43.94116767	-115.9678278	1486.213989	20.01
DJI_0862.JPG	43.94113886	-115.9675671	1486.313965	21.12
DJI_0863.JPG	43.94107736	-115.9671857	1486.213989	31.29
DJI_0864.JPG	43.94094631	-115.9670522	1486.014038	18.07
DJI_0865.JPG	43.94081428	-115.967042	1485.91394	14.70
DJI_0866.JPG	43.94078472	-115.9670596	1485.91394	3.58
DJI_0867.JPG	43.94075192	-115.9670895	1486.014038	4.36
DJI_0868.JPG	43.94072475	-115.9671151	1486.014038	3.65
DJI_0869.JPG	43.94070319	-115.9671383	1486.114014	3.03
DJI_0870.JPG	43.9406995	-115.9671458	1486.114014	0.72
DJI_0871.JPG	43.94068692	-115.9671499	1486.213989	1.44
DJI_0872.JPG	43.94066042	-115.9671549	1486.213989	2.97
DJI_0873.JPG	43.94061892	-115.9671602	1486.313965	4.63
DJI_0874.JPG	43.94061872	-115.9671631	1483.614014	0.23
DJI_0875.JPG	43.94061994	-115.9671592	1478.014038	0.33
DJI_0876.JPG	43.94061978	-115.96716	1472.114014	0.00
DJI_0877.JPG	43.94054625	-115.9676329	1456.813965	38.74

DJI_0878.JPG	43.94055797	-115.9677346	1471.514038	8.25
DJI_0879.JPG	43.94057897	-115.9678914	1474.91394	12.76
DJI_0880.JPG	43.94059908	-115.9681146	1483.014038	18.01
DJI_0881.JPG	43.94060928	-115.9683715	1485.713989	20.60
DJI_0882.JPG	43.94060814	-115.9686394	1486.313965	21.45
DJI_0883.JPG	43.94059694	-115.9690553	1486.313965	33.32
DJI_0884.JPG	43.94059581	-115.9693306	1486.213989	22.04
DJI_0885.JPG	43.940638	-115.9695924	1486.213989	21.48
DJI_0886.JPG	43.94070536	-115.9698473	1486.213989	21.73
DJI_0887.JPG	43.94077394	-115.970101	1486.313965	21.70
DJI_0888.JPG	43.94084261	-115.9703539	1486.213989	21.64
DJI_0889.JPG	43.94088908	-115.9707563	1486.313965	32.63
DJI_0890.JPG	43.94089097	-115.9710289	1486.313965	21.83
DJI_0891.JPG	43.94089192	-115.9713024	1486.213989	21.90
DJI_0892.JPG	43.94086786	-115.9715366	1486.213989	18.94
DJI_0893.JPG	43.94076578	-115.9716443	1486.213989	14.26
DJI_0894.JPG	43.94060836	-115.9716656	1486.313965	17.59
DJI_0895.JPG	43.94032931	-115.9716434	1486.313965	31.08
DJI_0896.JPG	43.94013906	-115.9716196	1486.313965	21.24
DJI_0897.JPG	43.939948	-115.9715961	1486.313965	21.33
DJI_0898.JPG	43.93975769	-115.9715733	1486.313965	21.24
DJI_0899.JPG	43.93956531	-115.9715504	1486.313965	21.47
DJI_0900.JPG	43.93937406	-115.9715286	1486.313965	21.34
DJI_0901.JPG	43.93908569	-115.9715016	1486.313965	32.14
DJI_0902.JPG	43.93889356	-115.9714877	1486.313965	21.39
DJI_0903.JPG	43.93870167	-115.9714773	1486.41394	21.35
DJI_0904.JPG	43.93850972	-115.9714672	1486.313965	21.36
DJI_0905.JPG	43.93831608	-115.9714541	1486.313965	21.56
DJI_0906.JPG	43.93812089	-115.9714413	1486.313965	21.73
DJI_0907.JPG	43.93793772	-115.9713853	1486.313965	20.85
DJI_0908.JPG	43.93779233	-115.9711327	1486.213989	25.89
DJI_0909.JPG	43.93778317	-115.9709129	1486.213989	17.63
DJI_0910.JPG	43.937808	-115.9706587	1486.313965	20.54
DJI_0911.JPG	43.93785017	-115.9704119	1486.114014	20.31
DJI_0912.JPG	43.93785856	-115.9703369	1486.014038	6.07
DJI_0913.JPG	43.93786836	-115.970283	1486.114014	4.45
DJI_0914.JPG	43.93788192	-115.9701229	1486.213989	12.91
DJI_0915.JPG	43.93789122	-115.969903	1486.313965	17.64
DJI_0916.JPG	43.93788436	-115.9696506	1486.41394	20.23
DJI_0917.JPG	43.93786836	-115.9693837	1486.41394	21.44
DJI_0918.JPG	43.93785847	-115.9691226	1486.41394	20.94

DJI_0919.JPG	43.93785867	-115.9687266	1486.514038	31.71
DJI_0920.JPG	43.937859	-115.9684588	1486.514038	21.44
DJI_0921.JPG	43.93785833	-115.9681919	1486.41394	21.37
DJI_0922.JPG	43.93787275	-115.967926	1486.41394	21.35
DJI_0923.JPG	43.93795825	-115.9677344	1486.313965	18.05
DJI_0924.JPG	43.93810411	-115.9676427	1486.313965	17.80
DJI_0925.JPG	43.9382525	-115.9676441	1486.114014	16.50
DJI_0926.JPG	43.93828478	-115.9678523	1486.114014	17.05
DJI_0927.JPG	43.93826881	-115.9680946	1486.114014	19.48
DJI_0928.JPG	43.93827061	-115.9683665	1486.114014	21.77
DJI_0929.JPG	43.93828211	-115.9686438	1486.114014	22.24
DJI_0930.JPG	43.93829694	-115.9689149	1486.213989	21.78
DJI_0931.JPG	43.93830975	-115.9691793	1486.213989	21.22
DJI_0932.JPG	43.93832786	-115.9694398	1486.313965	20.95
DJI_0933.JPG	43.9383325	-115.9698322	1486.313965	31.43
DJI_0934.JPG	43.93833222	-115.9700984	1486.313965	21.32
DJI_0935.JPG	43.93833322	-115.9703647	1486.313965	21.32
DJI_0936.JPG	43.93833314	-115.9706331	1486.313965	21.49
DJI_0937.JPG	43.93833186	-115.9708008	1486.114014	13.43
DJI_0938.JPG	43.93833067	-115.9709029	1486.114014	8.18
DJI_0939.JPG	43.93832775	-115.9710218	1486.213989	9.52
DJI_0940.JPG	43.93832922	-115.9711971	1486.213989	14.04
DJI_0941.JPG	43.93833486	-115.9712409	1486.114014	3.56
DJI_0942.JPG	43.93835647	-115.9713159	1486.114014	6.47
DJI_0943.JPG	43.93835631	-115.9713168	1483.614014	0.00
DJI_0944.JPG	43.93835683	-115.9713153	1474.91394	0.13
DJI_0945.JPG	43.93835681	-115.9713192	1468.91394	0.31
DJI_0946.JPG	43.93837403	-115.9714024	1461.813965	6.93
DJI_0947.JPG	43.93837986	-115.9714432	1455.014038	3.33
DJI_0948.JPG	43.93837883	-115.9714442	1449.014038	0.13
DJI_0949.JPG	43.93837911	-115.9714431	1443.114014	0.13
DJI_0950.JPG	43.93839206	-115.9714889	1440.614014	3.94
DJI_0951.JPG	43.93838194	-115.9715188	1440.614014	2.65
DJI_0952.JPG	43.93840472	-115.9715201	1440.713989	2.53
DJI_0953.JPG	43.93840792	-115.9714896	1439.313965	2.47
DJI_0954.JPG	43.93840769	-115.9714905	1437.213989	0.13
DJI_0955.JPG	43.93840642	-115.9714932	1431.813965	0.27
DJI_0956.JPG	43.93840939	-115.9714673	1425.514038	2.10
DJI_0957.JPG	43.93841033	-115.971458	1421.313965	0.75
DJI_0958.JPG	43.93840461	-115.9714661	1419.114014	0.90
DJI_0959.JPG	43.93842147	-115.9714655	1419.213989	1.87

DJI_0960.JPG	43.93843231	-115.9714512	1418.313965	1.66
DJI_0961.JPG	43.93843369	-115.9714546	1416.014038	0.33
DJI_0962.JPG	43.93843417	-115.9714549	1413.813965	0.00
DJI_0963.JPG	43.93843531	-115.9714577	1411.713989	0.27
DJI_0964.JPG	43.93843928	-115.9714658	1410.614014	0.79
DJI_0965.JPG	43.93844644	-115.9714619	1408.614014	0.86
DJI_0966.JPG	43.93845683	-115.9714408	1405.313965	2.04
DJI_0967.JPG	43.93846228	-115.9714384	1401.91394	0.64
DJI_0968.JPG	43.93846703	-115.9714306	1396.114014	0.82
DJI_0969.JPG	43.93847172	-115.9714212	1390.014038	0.92
DJI_0970.JPG	43.93848108	-115.971377	1487.170044	3.69
DJI_0971.JPG	43.93847797	-115.9714224	1487.069946	3.65
DJI_0972.JPG	43.93847889	-115.9715863	1487.27002	13.13
DJI_0973.JPG	43.93847178	-115.9718354	1487.369995	19.96
DJI_0974.JPG	43.93846181	-115.9721057	1487.27002	21.67
DJI_0975.JPG	43.93844444	-115.9723775	1487.069946	21.85
DJI_0976.JPG	43.93843164	-115.9726466	1487.170044	21.59
DJI_0977.JPG	43.93845656	-115.9730278	1487.170044	30.65
DJI_0978.JPG	43.93854917	-115.9731321	1486.869995	13.26
DJI_0979.JPG	43.93869867	-115.9731419	1487.170044	16.64
DJI_0980.JPG	43.93887725	-115.973133	1487.27002	19.87
DJI_0981.JPG	43.93906339	-115.9731234	1487.27002	20.71
DJI_0982.JPG	43.9392495	-115.9731129	1487.369995	20.71
DJI_0983.JPG	43.93953675	-115.9731	1487.27002	31.96
DJI_0984.JPG	43.93972964	-115.9730918	1487.27002	21.46
DJI_0985.JPG	43.93991958	-115.9730824	1487.27002	21.13
DJI_0986.JPG	43.94011133	-115.9730733	1487.27002	21.33
DJI_0987.JPG	43.9403075	-115.9730631	1487.27002	21.83
DJI_0988.JPG	43.94050556	-115.9730589	1487.170044	22.02
DJI_0989.JPG	43.94080517	-115.9730525	1487.170044	33.32
DJI_0990.JPG	43.94098742	-115.9730768	1487.069946	20.36
DJI_0991.JPG	43.94107819	-115.9731976	1487.069946	13.98
DJI_0992.JPG	43.94109992	-115.9733788	1487.170044	14.70
DJI_0993.JPG	43.94109339	-115.9736127	1487.170044	18.74
DJI_0994.JPG	43.94108414	-115.9738771	1487.069946	21.19
DJI_0995.JPG	43.94104564	-115.9742862	1486.969971	33.03
DJI_0996.JPG	43.94096544	-115.9745018	1486.969971	19.43
DJI_0997.JPG	43.94082897	-115.9745817	1487.069946	16.47
DJI_0998.JPG	43.94066161	-115.9745856	1487.170044	18.61
DJI_0999.JPG	43.94047736	-115.9745816	1487.170044	20.49
DJI_0001.JPG	43.94019133	-115.9745983	1487.170044	31.83

DJI_0002.JPG	43.94000042	-115.9746366	1487.069946	21.45
DJI_0003.JPG	43.93981053	-115.9746672	1487.069946	21.26
DJI_0004.JPG	43.93961892	-115.9746735	1487.069946	21.31
DJI_0005.JPG	43.93942392	-115.9746695	1487.069946	21.69
DJI_0006.JPG	43.93922478	-115.9746653	1487.069946	22.15
DJI_0007.JPG	43.93893469	-115.9746597	1487.069946	32.26
DJI_0008.JPG	43.93874439	-115.9746545	1487.069946	21.17
DJI_0009.JPG	43.93855114	-115.9746486	1487.069946	21.49
DJI_0010.JPG	43.93835853	-115.9746608	1486.969971	21.44
DJI_0011.JPG	43.93823683	-115.9747963	1486.869995	17.34
DJI_0012.JPG	43.93821497	-115.9750113	1486.869995	17.39
DJI_0013.JPG	43.93825508	-115.9754109	1486.869995	32.30
DJI_0014.JPG	43.93827344	-115.9756842	1486.969971	21.98
DJI_0015.JPG	43.93828503	-115.9759566	1486.969971	21.85
DJI_0016.JPG	43.93833886	-115.9761884	1486.869995	19.50
DJI_0017.JPG	43.93846614	-115.9762725	1486.77002	15.67
DJI_0018.JPG	43.93862903	-115.9762566	1487.069946	18.16
DJI_0019.JPG	43.93889892	-115.9761945	1487.27002	30.42
DJI_0020.JPG	43.93908444	-115.9761547	1487.369995	20.87
DJI_0021.JPG	43.93927142	-115.9761203	1487.469971	20.97
DJI_0022.JPG	43.93946142	-115.9761121	1487.469971	21.14
DJI_0023.JPG	43.93965828	-115.9761161	1487.369995	21.89
DJI_0024.JPG	43.93985447	-115.9761285	1487.369995	21.84
DJI_0025.JPG	43.94014642	-115.9761413	1487.369995	32.48
DJI_0026.JPG	43.94033778	-115.9761359	1487.369995	21.28
DJI_0027.JPG	43.94052594	-115.9761165	1487.369995	20.98
DJI_0028.JPG	43.940715	-115.9760901	1487.469971	21.13
DJI_0029.JPG	43.94089922	-115.9760634	1487.469971	20.60
DJI_0030.JPG	43.94108253	-115.9760383	1487.469971	20.48
DJI_0031.JPG	43.94120319	-115.9761203	1487.369995	14.94
DJI_0032.JPG	43.94124875	-115.9764171	1487.369995	24.30
DJI_0033.JPG	43.94125994	-115.9766686	1487.369995	20.17
DJI_0034.JPG	43.94127186	-115.9769273	1487.369995	20.76
DJI_0035.JPG	43.941284	-115.9771905	1487.369995	21.12
DJI_0036.JPG	43.94126883	-115.9774404	1487.369995	20.08
DJI_0037.JPG	43.94116361	-115.9775938	1487.27002	16.96
DJI_0038.JPG	43.94100142	-115.9776649	1487.27002	18.91
DJI_0039.JPG	43.94071994	-115.9777184	1487.27002	31.59
DJI_0040.JPG	43.94052689	-115.9777496	1487.27002	21.61
DJI_0041.JPG	43.94033378	-115.9777766	1487.27002	21.58
DJI_0042.JPG	43.94014111	-115.9778002	1487.27002	21.51

DJI_0043.JPG	43.93995053	-115.977807	1487.27002	21.20
DJI_0044.JPG	43.93975889	-115.9778086	1487.27002	21.31
DJI_0045.JPG	43.93946508	-115.9778108	1487.27002	32.67
DJI_0046.JPG	43.93927192	-115.9778026	1487.27002	21.49
DJI_0047.JPG	43.93908153	-115.9777932	1487.369995	21.18
DJI_0048.JPG	43.93889278	-115.9777896	1487.369995	20.99
DJI_0049.JPG	43.93872156	-115.9778569	1487.27002	19.79
DJI_0050.JPG	43.93857794	-115.9779989	1487.27002	19.60
DJI_0051.JPG	43.93843419	-115.9780402	1487.170044	16.32
DJI_0052.JPG	43.93843517	-115.9780434	1487.170044	0.28
DJI_0053.JPG	43.93842739	-115.9779961	1487.27002	3.89
DJI_0054.JPG	43.93843289	-115.9778405	1487.369995	12.47
DJI_0055.JPG	43.93848703	-115.9776387	1487.469971	17.25
DJI_0056.JPG	43.93857769	-115.9774277	1487.569946	19.67
DJI_0057.JPG	43.93866739	-115.9771994	1487.569946	20.83
DJI_0058.JPG	43.9386915	-115.9767999	1487.369995	32.10
DJI_0059.JPG	43.93866483	-115.9765307	1487.369995	21.75
DJI_0060.JPG	43.93865225	-115.976268	1487.369995	21.08
DJI_0061.JPG	43.93863428	-115.9760024	1487.369995	21.36
DJI_0062.JPG	43.93860681	-115.9757408	1487.369995	21.17
DJI_0063.JPG	43.93857797	-115.9754756	1487.369995	21.47
DJI_0064.JPG	43.938542	-115.9750692	1487.27002	32.78
DJI_0065.JPG	43.93850114	-115.9747989	1487.170044	22.11
DJI_0066.JPG	43.93843225	-115.9745422	1487.170044	21.94
DJI_0067.JPG	43.93833575	-115.9743041	1487.27002	21.88
DJI_0068.JPG	43.93823939	-115.9740776	1487.27002	21.06
DJI_0069.JPG	43.93819033	-115.9738389	1487.369995	19.88
DJI_0070.JPG	43.93816061	-115.9734503	1487.369995	31.29
DJI_0071.JPG	43.93815325	-115.9731813	1487.369995	21.55
DJI_0072.JPG	43.93816178	-115.9729163	1487.369995	21.24
DJI_0073.JPG	43.93817153	-115.9727883	1487.369995	10.31
DJI_0074.JPG	43.93817139	-115.9726571	1487.170044	10.51
DJI_0075.JPG	43.93819114	-115.9725288	1487.27002	10.50
DJI_0076.JPG	43.938219	-115.9723261	1487.369995	16.52
DJI_0077.JPG	43.93823669	-115.9721183	1487.170044	16.76
DJI_0078.JPG	43.93823186	-115.9721153	1487.170044	0.59
DJI_0079.JPG	43.93826008	-115.9720185	1487.369995	8.36
DJI_0080.JPG	43.93829919	-115.9718356	1487.369995	15.28
DJI_0081.JPG	43.93832508	-115.9716515	1487.170044	15.02
DJI_0082.JPG	43.93832036	-115.9716235	1487.170044	2.30
DJI_0083.JPG	43.93834739	-115.9715855	1487.27002	4.27

DJI_0084.JPG	43.93842317	-115.9714611	1485.77002	13.05
DJI_0085.JPG	43.93852069	-115.9712665	1478.27002	18.98
DJI_0086.JPG	43.93861686	-115.9710428	1468.069946	20.86
DJI_0087.JPG	43.93864247	-115.9709395	1459.869995	8.75
DJI_0088.JPG	43.93861389	-115.9709605	1450.569946	3.59
DJI_0089.JPG	43.93859078	-115.9710156	1443.670044	5.11
DJI_0090.JPG	43.93857522	-115.9710882	1436.27002	6.06
DJI_0091.JPG	43.93857686	-115.9711068	1430.369995	1.50
DJI_0092.JPG	43.93856011	-115.9711341	1424.369995	2.87
DJI_0093.JPG	43.93855417	-115.9711618	1419.670044	2.32
DJI_0094.JPG	43.93855167	-115.9711775	1415.569946	1.28
DJI_0095.JPG	43.93854958	-115.9711739	1414.170044	0.37
DJI_0096.JPG	43.93854922	-115.9711876	1413.069946	1.09
DJI_0097.JPG	43.93854911	-115.9711866	1412.069946	0.00
DJI_0098.JPG	43.93854731	-115.9712032	1409.869995	1.34
DJI_0099.JPG	43.93854611	-115.9712121	1408.369995	0.72
DJI_0100.JPG	43.93854719	-115.9712184	1406.869995	0.53
DJI_0101.JPG	43.93854322	-115.9712405	1404.469971	1.83
DJI_0102.JPG	43.93853961	-115.971272	1402.969971	2.55
DJI_0103.JPG	43.93853533	-115.971298	1400.369995	2.14
DJI_0104.JPG	43.93853889	-115.9713242	1397.469971	2.14
DJI_0105.JPG	43.93852769	-115.9713574	1393.469971	2.94
DJI_0106.JPG	43.93852303	-115.9713718	1390.569946	1.27
DJI_0107.JPG	43.93851972	-115.9713916	1387.77002	1.63
DJI_0108.JPG	43.9385155	-115.9714236	1384.77002	2.60
DJI_0109.JPG	43.93850753	-115.9714409	1381.869995	1.64
DJI_0110.JPG	43.93850406	-115.9714538	1379.069946	1.10
DJI_0111.JPG	43.93849975	-115.9714548	1375.170044	0.48
DJI_0112.JPG	43.93849928	-115.9714545	1372.569946	0.00
DJI_0113.JPG	43.93849942	-115.9714549	1370.469971	0.00
DJI_0114.JPG	43.93849939	-115.9714535	1369.670044	0.13
DJI_0117.JPG	43.93872139	-115.9723946	1458.123047	79.29
DJI_0118.JPG	43.93874947	-115.9724734	1467.922974	7.04
DJI_0119.JPG	43.93882028	-115.9726204	1477.922974	14.16
DJI_0120.JPG	43.93896381	-115.972931	1486.422974	29.55
DJI_0121.JPG	43.93906608	-115.9731608	1487.422974	21.63
DJI_0122.JPG	43.93916622	-115.9734007	1487.723022	22.21
DJI_0123.JPG	43.93925592	-115.973622	1486.922974	20.33
DJI_0124.JPG	43.93932214	-115.9738066	1487.422974	16.51
DJI_0125.JPG	43.93941097	-115.9740317	1487.223022	20.55
DJI_0126.JPG	43.93950886	-115.9742722	1487.223022	22.12

DJI_0127.JPG	43.93967636	-115.9746224	1487.123047	33.66
DJI_0128.JPG	43.93980261	-115.9748557	1487.123047	23.37
DJI_0129.JPG	43.93993219	-115.9750899	1487.123047	23.65
DJI_0130.JPG	43.94006344	-115.9753103	1487.123047	22.90
DJI_0131.JPG	43.94019214	-115.9755226	1487.223022	22.23
DJI_0132.JPG	43.94030867	-115.9757401	1487.422974	21.70
DJI_0133.JPG	43.94041278	-115.9759624	1487.522949	21.23
DJI_0134.JPG	43.94056656	-115.9762986	1487.422974	31.89
DJI_0135.JPG	43.94067617	-115.9765199	1487.422974	21.50
DJI_0136.JPG	43.94078478	-115.9767371	1487.223022	21.18
DJI_0137.JPG	43.94082697	-115.9768436	1487.123047	9.73
DJI_0138.JPG	43.94082675	-115.9768518	1487.022949	0.65
DJI_0139.JPG	43.94084122	-115.9768735	1487.322998	2.37
DJI_0140.JPG	43.94090422	-115.9769931	1487.322998	11.86
DJI_0141.JPG	43.94100128	-115.9771808	1487.322998	18.51
DJI_0142.JPG	43.94116819	-115.9775131	1487.422974	32.44
DJI_0143.JPG	43.94121961	-115.9777657	1487.422974	21.02
DJI_0144.JPG	43.94121994	-115.9779791	1487.123047	17.09
DJI_0145.JPG	43.94121333	-115.9780154	1487.223022	3.00
DJI_0146.JPG	43.94120286	-115.9780578	1487.422974	3.59
DJI_0147.JPG	43.94113397	-115.9781631	1487.422974	11.39
DJI_0148.JPG	43.94098953	-115.9782139	1487.422974	16.57
DJI_0149.JPG	43.94080622	-115.978204	1487.422974	20.40
DJI_0150.JPG	43.94062486	-115.9781195	1487.422974	21.27
DJI_0151.JPG	43.94040875	-115.9778604	1487.522949	31.75
DJI_0152.JPG	43.94030253	-115.9776326	1487.522949	21.73
DJI_0153.JPG	43.94019606	-115.9773993	1487.422974	22.12
DJI_0154.JPG	43.94008592	-115.977165	1487.422974	22.40
DJI_0155.JPG	43.93997022	-115.9769283	1487.322998	22.91
DJI_0156.JPG	43.93979656	-115.9765734	1487.223022	34.35
DJI_0157.JPG	43.93967994	-115.9763376	1487.322998	22.91
DJI_0158.JPG	43.93956644	-115.9761072	1487.322998	22.35
DJI_0159.JPG	43.93945061	-115.9758741	1487.322998	22.68
DJI_0160.JPG	43.93933422	-115.9756428	1487.223022	22.59
DJI_0161.JPG	43.93923594	-115.9753993	1487.223022	22.36
DJI_0162.JPG	43.93913747	-115.9750094	1487.322998	33.08
DJI_0163.JPG	43.93907933	-115.974746	1487.422974	22.06
DJI_0164.JPG	43.939025	-115.974477	1487.422974	22.37
DJI_0165.JPG	43.93897278	-115.9742136	1487.322998	21.87
DJI_0166.JPG	43.93894036	-115.97409	1487.223022	10.54
DJI_0167.JPG	43.93893575	-115.9740837	1487.322998	0.72

DJI_0168.JPG	43.93888853	-115.9741409	1487.322998	6.97
DJI_0169.JPG	43.93882275	-115.9744202	1487.322998	23.53
DJI_0170.JPG	43.93881981	-115.9746799	1487.422974	20.80
DJI_0171.JPG	43.93883711	-115.9749435	1487.422974	21.19
DJI_0172.JPG	43.93886078	-115.9752105	1487.422974	21.54
DJI_0173.JPG	43.93888967	-115.9754838	1487.422974	22.12
DJI_0174.JPG	43.93891636	-115.9757583	1487.322998	22.18
DJI_0175.JPG	43.93892528	-115.9761673	1487.322998	32.76
DJI_0176.JPG	43.93891792	-115.97644	1487.223022	21.85
DJI_0177.JPG	43.93890669	-115.9767197	1487.322998	22.42
DJI_0178.JPG	43.93889717	-115.9769989	1487.223022	22.38
DJI_0179.JPG	43.93888603	-115.9772755	1487.223022	22.18
DJI_0180.JPG	43.93887322	-115.9775597	1487.123047	22.80
DJI_0181.JPG	43.93886422	-115.977846	1487.123047	22.95
DJI_0182.JPG	43.9388575	-115.9782639	1487.223022	33.47
DJI_0183.JPG	43.93885367	-115.978536	1487.322998	21.79
DJI_0184.JPG	43.93887258	-115.9788037	1487.322998	21.54
DJI_0185.JPG	43.93896025	-115.9790274	1487.322998	20.39
DJI_0186.JPG	43.939197	-115.9791159	1487.322998	27.26
DJI_0187.JPG	43.93938994	-115.979079	1487.422974	21.66
DJI_0188.JPG	43.93958372	-115.9790456	1487.422974	21.71
DJI_0189.JPG	43.93977531	-115.9790598	1487.422974	21.33
DJI_0190.JPG	43.93996656	-115.9791091	1487.422974	21.63
DJI_0191.JPG	43.94026017	-115.9791639	1487.422974	32.94
DJI_0192.JPG	43.94044844	-115.9791223	1487.422974	21.20
DJI_0193.JPG	43.94063094	-115.9790552	1487.422974	20.99
DJI_0194.JPG	43.94081531	-115.9790123	1487.322998	20.79
DJI_0195.JPG	43.94100275	-115.9790234	1487.322998	20.86
DJI_0196.JPG	43.94118828	-115.9790527	1487.322998	20.76
DJI_0197.JPG	43.94126931	-115.9790606	1486.922974	9.03
DJI_0198.JPG	43.94131686	-115.9790384	1486.822998	5.58
DJI_0199.JPG	43.94127125	-115.9789573	1486.723022	8.25
DJI_0200.JPG	43.94118567	-115.9788089	1486.822998	15.22
DJI_0201.JPG	43.94114614	-115.9784672	1486.922974	27.71
DJI_0202.JPG	43.94116656	-115.9782098	1486.922974	20.73
DJI_0203.JPG	43.94119192	-115.9779442	1486.922974	21.45
DJI_0204.JPG	43.94121969	-115.9776746	1486.922974	21.81
DJI_0205.JPG	43.94124006	-115.9774022	1486.822998	21.93
DJI_0206.JPG	43.94124436	-115.977129	1486.922974	21.87
DJI_0207.JPG	43.94120464	-115.9767266	1486.822998	32.52
DJI_0208.JPG	43.94112489	-115.9764803	1486.922974	21.62

DJI_0209.JPG	43.94103144	-115.9762433	1486.922974	21.64
DJI_0210.JPG	43.94093483	-115.9760076	1487.022949	21.72
DJI_0211.JPG	43.94083728	-115.9757703	1486.922974	21.88
DJI_0212.JPG	43.94071081	-115.9753921	1486.922974	33.39
DJI_0213.JPG	43.94062569	-115.9751376	1486.822998	22.47
DJI_0214.JPG	43.94053806	-115.9748821	1486.822998	22.66
DJI_0215.JPG	43.94045014	-115.9746317	1486.822998	22.31
DJI_0216.JPG	43.94031836	-115.9742673	1486.922974	32.65
DJI_0217.JPG	43.94022161	-115.9740366	1487.022949	21.38
DJI_0218.JPG	43.94011875	-115.9738096	1487.022949	21.47
DJI_0219.JPG	43.94001319	-115.9735806	1487.022949	21.77
DJI_0220.JPG	43.93990906	-115.9733508	1487.022949	21.74
DJI_0221.JPG	43.93980356	-115.9731204	1486.922974	21.86
DJI_0222.JPG	43.93964764	-115.9727657	1486.922974	33.28
DJI_0223.JPG	43.93952725	-115.9725458	1486.922974	22.12
DJI_0224.JPG	43.93937133	-115.9723819	1486.922974	21.75
DJI_0225.JPG	43.93919372	-115.9722604	1486.922974	22.02
DJI_0226.JPG	43.93901464	-115.9721417	1486.922974	22.06
DJI_0227.JPG	43.93884764	-115.9719956	1486.922974	21.95
DJI_0228.JPG	43.93869819	-115.9718301	1486.822998	21.26
			Average:	17.79

D. Results

D.1 Turi Create CNN

*all accuracies come from same set of three eval tests

**trainOutput89 is a model built on 4000 images and had a .89 accuracy

Accuracies: 0.85, 0.87, 0.94

avg = 0.8866 about 0.89

Accuracy for imagesToClassify1:

0.85

Confusion Matrix for imagesToClassify1:

```
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|     road     |     notRoad     |   3   |
|   notRoad    |     notRoad     |   48  |
|   notRoad    |       road      |   12  |
|     road     |       road      |   37  |
+--------------+-----------------+-------+
```

Accuracy for imagesToClassify2:

0.87

Confusion Matrix for imagesToClassify2:

```
+--------------+-----------------+-------+
```

target_label	predicted_label	count
notRoad	notRoad	67
notRoad	road	13
road	road	20

Accuracy for imagesToClassify3:

0.94

Confusion Matrix for imagesToClassify3:

target_label	predicted_label	count
notRoad	notRoad	72
notRoad	road	6
road	road	22

**trainOutput81 is a model built on a couple hundred images, .81 accuracy

Accuracies: 0.83, 0.72, 0.89

avg = 0.81333 about 0.81

Accuracy for imagesToClassify1:

0.83

Confusion Matrix for imagesToClassify1:

```
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|     road     |     notRoad     |   9   |
|   notRoad    |     notRoad     |   52  |
|   notRoad    |       road      |   8   |
|     road     |       road      |   31  |
+--------------+-----------------+-------+
```

```
Accuracy for imagesToClassify2:
0.72
Confusion Matrix for imagesToClassify2:
```

```
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|     road     |     notRoad     |   1   |
|   notRoad    |     notRoad     |   53  |
|   notRoad    |       road      |   27  |
|     road     |       road      |   19  |
+--------------+-----------------+-------+
```

```
Accuracy for imagesToClassify3:
0.89
Confusion Matrix for imagesToClassify3:
```

```
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
```

```
|     road     |     notRoad     |   4   |
|   notRoad    |     notRoad     |   71  |
|   notRoad    |       road      |   7   |
|     road     |       road      |   18  |
+--------------+-----------------+-------+
```

**trainOutput82 about 100 more images added to 81, .82 accuracy

```
Accuracies: 0.79, 0.8, 0.88
    avg = .8233

Accuracy for imagesToClassify1:
0.79
Confusion Matrix for imagesToClassify1:
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|     road     |     notRoad     |   6   |
|   notRoad    |     notRoad     |   45  |
|   notRoad    |       road      |   15  |
|     road     |       road      |   34  |
+--------------+-----------------+-------+

Accuracy for imagesToClassify2:
0.8
Confusion Matrix for imagesToClassify2:
```

```
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|   notRoad    |     notRoad     |   60  |
|   notRoad    |       road      |   20  |
|     road     |       road      |   20  |
+--------------+-----------------+-------+

Accuracy for imagesToClassify3:
0.88
Confusion Matrix for imagesToClassify3:
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|     road     |     notRoad     |   4   |
|   notRoad    |     notRoad     |   70  |
|   notRoad    |       road      |   8   |
|     road     |       road      |   18  |
+--------------+-----------------+-------+
```

**trainOutput84 about 100 more images added to 82, 0.84 accuracy

```
Accuracies: 0.85, 0.81, 0.85
    avg = 0.83666

Accuracy for imagesToClassify1:
```

```
0.85
Confusion Matrix for imagesToClassify1:
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|     road     |     notRoad     |   5   |
|   notRoad    |     notRoad     |   50  |
|   notRoad    |       road      |   10  |
|     road     |       road      |   35  |
+--------------+-----------------+-------+

Accuracy for imagesToClassify2:
0.81
Confusion Matrix for imagesToClassify2:
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|   notRoad    |     notRoad     |   61  |
|   notRoad    |       road      |   19  |
|     road     |       road      |   20  |
+--------------+-----------------+-------+

Accuracy for imagesToClassify3:
0.85
Confusion Matrix for imagesToClassify3:
+--------------+-----------------+-------+
| target_label | predicted_label | count |
```

```
+--------------+-----------------+-------+
|     road     |     notRoad     |   4   |
|   notRoad    |     notRoad     |   67  |
|   notRoad    |       road      |   11  |
|     road     |       road      |   18  |
+--------------+-----------------+-------+
```

**trainOutput79 is built on the road images from 84 with more forested environment added, but all of the negatives were replaced with a different selection of data.

Accuracies: 0.81, 0.70, 0.86

avg = 0.79

Accuracy for imagesToClassify1:

0.81

Confusion Matrix for imagesToClassify1:

```
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|     road     |     notRoad     |   1   |
|   notRoad    |       road      |   18  |
|   notRoad    |     notRoad     |   42  |
|     road     |       road      |   39  |
+--------------+-----------------+-------+
```

Accuracy for imagesToClassify2:

0.7

Confusion Matrix for imagesToClassify2:

```
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|   notRoad    |     notRoad     |   50  |
|   notRoad    |       road      |   30  |
|     road     |       road      |   20  |
+--------------+-----------------+-------+
```

Accuracy for imagesToClassify3:

0.86

Confusion Matrix for imagesToClassify3:

```
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|     road     |     notRoad     |   1   |
|   notRoad    |     notRoad     |   65  |
|   notRoad    |       road      |   13  |
|     road     |       road      |   21  |
+--------------+-----------------+-------+
```

**trainOutputS89 Is built from a pseudo-random sample set of all flights and an random set of images taken from older images (which were taken before this year and what was used for trainOutput97, though less duplicates)

Accuracies: 0.83, 0.89, 0.94

avg = 0.8866 about 0.89

Accuracy for imagesToClassify1:

0.83

Confusion Matrix for imagesToClassify1:

target_label	predicted_label	count
road	notRoad	3
notRoad	notRoad	46
notRoad	road	14
road	road	37

Accuracy for imagesToClassify2:

0.89

Confusion Matrix for imagesToClassify2:

target_label	predicted_label	count
notRoad	notRoad	69
notRoad	road	11
road	road	20

Accuracy for imagesToClassify3:

0.94

Confusion Matrix for imagesToClassify3:

target_label	predicted_label	count
notRoad	notRoad	72
notRoad	road	6
road	road	22

**trainOutput90 Is the same as trainOutputS89 but I removed corners and ambiguous pos labels. This did increase accuracy

across the image sets and lowered False positives. 1511 positives and 1700 negatives

Accuracies: 0.88, 0.86, 0.95

avg = 0.89666 about 90

Accuracy for imagesToClassify1:

0.88

Confusion Matrix for imagesToClassify1:

target_label	predicted_label	count
road	notRoad	5
notRoad	notRoad	53
notRoad	road	7
road	road	35

```
+--------------+-----------------+-------+


Accuracy for imagesToClassify2:

0.86

Confusion Matrix for imagesToClassify2:

+--------------+-----------------+-------+

| target_label | predicted_label | count |

+--------------+-----------------+-------+

|     road     |     notRoad     |   1   |

|   notRoad    |     notRoad     |   67  |

|   notRoad    |       road      |   13  |

|     road     |       road      |   19  |

+--------------+-----------------+-------+


Accuracy for imagesToClassify3:

0.95

Confusion Matrix for imagesToClassify3:

+--------------+-----------------+-------+

| target_label | predicted_label | count |

+--------------+-----------------+-------+

|   notRoad    |     notRoad     |   73  |

|   notRoad    |       road      |   5   |

|     road     |       road      |   22  |

+--------------+-----------------+-------+
```

**trainOutput92 Is the same as trainOutput90 but I removed 4 pos images and added new 300 neg labels in hopes of lowering false positives. 1507 positives and 2000 negatives

```
Accuracies: 0.9, 0.89, 0.96

    avg = 0.91666 about .92


Accuracy for imagesToClassify1:

0.9

Confusion Matrix for imagesToClassify1:

+--------------+-----------------+-------+

| target_label | predicted_label | count |

+--------------+-----------------+-------+

|     road     |     notRoad     |   4   |

|   notRoad    |     notRoad     |   54  |

|   notRoad    |       road      |   6   |

|     road     |       road      |   36  |

+--------------+-----------------+-------+

[4 rows x 3 columns]


Accuracy for imagesToClassify2:

0.89

Confusion Matrix for imagesToClassify2:

+--------------+-----------------+-------+

| target_label | predicted_label | count |

+--------------+-----------------+-------+

|     road     |     notRoad     |   2   |

|   notRoad    |     notRoad     |   71  |

|   notRoad    |       road      |   9   |

|     road     |       road      |   18  |
```

```
+--------------+-----------------+-------+

[4 rows x 3 columns]


Accuracy for imagesToClassify3:

0.96

Confusion Matrix for imagesToClassify3:

+--------------+-----------------+-------+

| target_label | predicted_label | count |

+--------------+-----------------+-------+

|   notRoad    |     notRoad     |   74  |

|   notRoad    |       road      |   4   |

|     road     |       road      |   22  |

+--------------+-----------------+-------+
```

**trainOutputS90 Is the same as trainOutput90 but I removed 4 pos images and got a new sample of 2000 neg labels in hopes of lowering false positives. 1507 positives and 2000 negatives. This did a better job than trainOutput 90 with false positives

```
Accuracies: 0.91, 0.89, 0.91

    avg = 0.9033


Accuracy for imagesToClassify1:

0.91

Confusion Matrix for imagesToClassify1:

+--------------+-----------------+-------+

| target_label | predicted_label | count |

+--------------+-----------------+-------+
```

```
|     road     |     notRoad     |   4   |
|   notRoad    |     notRoad     |   55  |
|   notRoad    |       road      |   5   |
|     road     |       road      |   36  |
+--------------+-----------------+-------+
```

```
Accuracy for imagesToClassify2:
0.89
Confusion Matrix for imagesToClassify2:
```

```
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|     road     |     notRoad     |   3   |
|   notRoad    |     notRoad     |   72  |
|   notRoad    |       road      |   8   |
|     road     |       road      |   17  |
+--------------+-----------------+-------+
```

```
Accuracy for imagesToClassify3:
0.91
Confusion Matrix for imagesToClassify3:
```

```
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|     road     |     notRoad     |   2   |
|   notRoad    |     notRoad     |   71  |
|   notRoad    |       road      |   7   |
```

```
|     road     |       road      |   20  |
+--------------+-----------------+-------+
```

trainOutput94 I am using 6000 negatives produced from the 1500 positives. I am augmenting the 1500 positives so I will have 6000 negatives and 6000 positives. I will rotate the original image 90 degrees and flip both horizontally and vertically the original and rotated image. This seems to have worked fairly well as the false positives and false negatives are similar and low

```
Accuracies: 0.89, 0.94, 0.98

    avg = 0.93666 about .94


Accuracy for imagesToClassify1:

    0.89

Confusion Matrix for imagesToClassify1:

+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|     road     |     notRoad     |   5   |
|   notRoad    |     notRoad     |   54  |
|   notRoad    |       road      |   6   |
|     road     |       road      |   35  |
+--------------+-----------------+-------+


Accuracy for imagesToClassify2:

    0.94

Confusion Matrix for imagesToClassify2:

+--------------+-----------------+-------+
| target_label | predicted_label | count |
```

```
+--------------+-----------------+-------+
|     road     |     notRoad     |   3   |
|   notRoad    |     notRoad     |   77  |
|   notRoad    |       road      |   3   |
|     road     |       road      |   17  |
+--------------+-----------------+-------+
```

```
Accuracy for imagesToClassify3:
    0.98
Confusion Matrix for imagesToClassify3:
```

```
+--------------+-----------------+-------+
| target_label | predicted_label | count |
+--------------+-----------------+-------+
|     road     |     notRoad     |   1   |
|   notRoad    |       road      |   1   |
|   notRoad    |     notRoad     |   77  |
|     road     |       road      |   21  |
+--------------+-----------------+-------+
```

D.2 Turi Create Other

Results:

Logistic regression:

Number of examples : 596

Number of classes : 7

Number of feature columns : 25

Number of unpacked features : 25

Number of coefficients : 156

Starting Newton Method

+-----------+----------+--------------+------------------+----------------------+

| Iteration | Passes | Elapsed Time | Training-accuracy | Validation-accuracy |

+-----------+----------+--------------+------------------+----------------------+

| 1 | 2 | 1.033285 | 0.476510 | 0.363636 |

| 2 | 3 | 1.043136 | 0.481544 | 0.333333 |

| 3 | 4 | 1.054729 | 0.483221 | 0.303030 |

| 4 | 5 | 1.064228 | 0.483221 | 0.303030 |

| 5 | 6 | 1.073549 | 0.483221 | 0.303030 |

+-----------+----------+--------------+------------------+----------------------+

SUCCESS: Optimal solution found.

Accuracy : \m0.374301675978

Confusion Matrix :

+--------------+-----------------+-------+

| target_label | predicted_label | count |

+--------------+-----------------+-------+

| NR | F | 4 |

| RF | R | 4 |

| LF | NR | 4 |

| R | AU | 2 |

| NR | RF | 3 |

| LF | L | 7 |

| F | LF | 2 |

| RF | LF | 1 |

| L | RF | 1 |

| R | R | 8 |

+--------------+-----------------+-------+

[45 rows x 3 columns]

K-NN with max neighbors 5:

Starting ball tree nearest neighbors model training.

+------------+--------------+

| Tree level | Elapsed Time |

+------------+--------------+

| 0 | 957us |

+------------+--------------+

+--------------+-------------+--------------+

| Query points | % Complete. | Elapsed Time |

+--------------+-------------+--------------+

| 1 | 0.5 | 656us |

| Done | | 6.744ms |

+--------------+-------------+--------------+

+-------+-------------+

| class | probability |

+-------+-------------+

| F | 0.4 |

| AU | 0.6 |

| AU | 0.6 |

| AU | 0.4 |

| F | 0.8 |

| AU | 0.6 |

| RF | 0.6 |

| F | 0.8 |

| RF | 0.4 |

| R | 0.8 |

+-------+-------------+

[162 rows x 2 columns]

Note: Only the head of the SFrame is printed.

You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.

WARNING: Ignoring `roc_curve`. Not supported for multi-class classification.

+--------------+-------------+--------------+

| Query points | % Complete. | Elapsed Time |

+--------------+-------------+--------------+

| 1 | 0.5 | 915us |

| Done | | 7.08ms |

+--------------+-------------+--------------+

+--------------+-------------+--------------+

| Query points | % Complete. | Elapsed Time |

+--------------+-------------+--------------+

| 1 | 0.5 | 680us |

| Done | | 6.559ms |

+--------------+-------------+--------------+

{'accuracy': 0.5061728395061729}

Boosted trees classifier:

Number of examples : 627

Number of classes : 7

Number of feature columns : 25

Number of unpacked features : 25

Iteration	Elapsed Time	Training-accuracy	Validation-accuracy	Training-log_loss	Validation-log_loss
1	0.061954	0.947368	0.424242	1.365759	1.727189
2	0.121393	0.984051	0.393939	1.015885	1.623551
3	0.185309	0.993620	0.424242	0.782035	1.504347
4	0.246836	0.998405	0.454545	0.610150	1.440168
5	0.308832	0.998405	0.484848	0.482705	1.393229
6	0.368625	1.000000	0.454545	0.387270	1.383487
11	0.649838	1.000000	0.454545	0.154015	1.289829
25	1.320851	1.000000	0.606061	0.035172	1.260491
50	2.200966	1.000000	0.575758	0.012643	1.316766
51	2.230356	1.000000	0.606061	0.012342	1.325593
75	2.886977	1.000000	0.575758	0.007841	1.383569
100	3.472501	1.000000	0.606061	0.005827	1.441169
101	3.495365	1.000000	0.606061	0.005773	1.442580
125	4.001885	1.000000	0.636364	0.004721	1.483739
150	4.491855	1.000000	0.636364	0.004011	1.517363
175	4.946271	1.000000	0.606061	0.003519	1.538021
200	5.383961	1.000000	0.606061	0.003164	1.564999
225	5.799371	1.000000	0.606061	0.002878	1.581720
250	6.203940	1.000000	0.575758	0.002660	1.597127
275	6.603608	1.000000	0.575758	0.002485	1.608908
300	6.986063	1.000000	0.575758	0.002341	1.621525
325	7.358658	1.000000	0.575758	0.002218	1.635224
350	7.720665	1.000000	0.575758	0.002113	1.647486
375	8.070499	1.000000	0.575758	0.002020	1.658406
400	8.418456	1.000000	0.575758	0.001939	1.666826
425	8.760573	1.000000	0.575758	0.001869	1.673186

450	9.102748	1.000000	0.575758	0.001807	1.681062
475	9.434222	1.000000	0.575758	0.001750	1.690461
500	9.759489	1.000000	0.575758	0.001696	1.697165
501	9.772874	1.000000	0.575758	0.001694	1.697354
525	10.076421	1.000000	0.575758	0.001648	1.701890
550	10.401247	1.000000	0.575758	0.001604	1.701263
575	10.720698	1.000000	0.575758	0.001564	1.704800
600	11.025947	1.000000	0.575758	0.001527	1.706760
625	11.328159	1.000000	0.575758	0.001494	1.708677
650	11.626098	1.000000	0.575758	0.001463	1.712650
675	11.915011	1.000000	0.575758	0.001435	1.712307
700	12.195754	1.000000	0.575758	0.001408	1.712605
725	12.451276	1.000000	0.575758	0.001382	1.714562
750	12.715143	1.000000	0.575758	0.001358	1.715586
775	12.995917	1.000000	0.575758	0.001335	1.718567
800	13.270625	1.000000	0.575758	0.001315	1.721214
825	13.540179	1.000000	0.575758	0.001295	1.724288
850	13.808205	1.000000	0.575758	0.001277	1.725745
875	14.076067	1.000000	0.575758	0.001259	1.729455
900	14.343771	1.000000	0.575758	0.001241	1.733496
925	14.605201	1.000000	0.575758	0.001226	1.736035
950	14.867022	1.000000	0.575758	0.001211	1.739858
975	15.127365	1.000000	0.575758	0.001196	1.740721
1000	15.388919	1.000000	0.575758	0.001182	1.742404
1001	15.399481	1.000000	0.575758	0.001182	1.741656
1025	15.650155	1.000000	0.575758	0.001169	1.743046
1050	15.893652	1.000000	0.575758	0.001156	1.743451
1075	16.141433	1.000000	0.575758	0.001144	1.743293
1100	16.390559	1.000000	0.575758	0.001132	1.745422
1125	16.645027	1.000000	0.575758	0.001121	1.747193
1150	16.897446	1.000000	0.575758	0.001111	1.750098

1175	17.150369	1.000000	0.575758	0.001101	1.753039
1200	17.403714	1.000000	0.575758	0.001091	1.753427
1225	17.652991	1.000000	0.575758	0.001082	1.755194
1250	17.899065	1.000000	0.575758	0.001073	1.754683
1275	18.146625	1.000000	0.575758	0.001065	1.756154
1300	18.392806	1.000000	0.575758	0.001056	1.757219
1325	18.634988	1.000000	0.575758	0.001049	1.758028
1350	18.880148	1.000000	0.575758	0.001042	1.759759
1375	19.124286	1.000000	0.575758	0.001035	1.761247
1400	19.367452	1.000000	0.575758	0.001028	1.763095
1425	19.608575	1.000000	0.575758	0.001021	1.764941
1450	19.851569	1.000000	0.575758	0.001014	1.767218
1475	20.093274	1.000000	0.575758	0.001008	1.768246
1500	20.334845	1.000000	0.575758	0.001002	1.768659
1525	20.568953	1.000000	0.575758	0.000996	1.770970
1550	20.805348	1.000000	0.575758	0.000990	1.773814
1575	21.040191	1.000000	0.575758	0.000985	1.776399
1600	21.274824	1.000000	0.575758	0.000980	1.779022
1625	21.507000	1.000000	0.575758	0.000975	1.781187
1650	21.717132	1.000000	0.575758	0.000970	1.782989
1675	21.932951	1.000000	0.575758	0.000965	1.786066
1700	22.153734	1.000000	0.575758	0.000961	1.788303
1725	22.380398	1.000000	0.575758	0.000956	1.790236
1750	22.601821	1.000000	0.575758	0.000952	1.791834
1775	22.807907	1.000000	0.575758	0.000948	1.793008
1800	23.009734	1.000000	0.575758	0.000944	1.793883
1825	23.221458	1.000000	0.575758	0.000940	1.795937
1850	23.441011	1.000000	0.575758	0.000936	1.796873
1875	23.660857	1.000000	0.575758	0.000932	1.798477
1900	23.876100	1.000000	0.575758	0.000929	1.800473
1925	24.090995	1.000000	0.575758	0.000925	1.801744

| 1950 | 24.306340 | 1.000000 | 0.575758 | 0.000922 | 1.802770 |

| 1975 | 24.521544 | 1.000000 | 0.575758 | 0.000918 | 1.804650 |

| 2000 | 24.734381 | 1.000000 | 0.575758 | 0.000915 | 1.806006 |

+----------+-------------+-----------------+--------------------+-----------------+---------------------+

Accuracy : \m0.472972972973

Confusion Matrix :

+--------------+-----------------+-------+

| target_label | predicted_label | count |

+--------------+-----------------+-------+

| F | LF | 3 |

| LF | NR | 1 |

| RF | R | 2 |

| LF | AU | 3 |

| L | RF | 1 |

| R | R | 9 |

| F | F | 17 |

| LF | RF | 1 |

| RF | L | 3 |

| R | F | 1 |

+--------------+-----------------+-------+

[36 rows x 3 columns]

Random forest classifier:

Number of examples : 605

Number of classes : 7

Number of feature columns : 25

Number of unpacked features : 25

+----------+-------------+-----------------+--------------------+-----------------+---------------------+

| Iteration | Elapsed Time | Training-accuracy | Validation-accuracy | Training-log_loss | Validation-log_loss |

1	0.058713	0.828099	0.437500	0.784255	1.584066
2	0.112848	0.925620	0.468750	0.705202	1.488792
3	0.164260	0.937190	0.406250	0.705566	1.455549
4	0.215903	0.973554	0.468750	0.699122	1.505193
5	0.267798	0.971901	0.437500	0.691860	1.544475
6	0.317276	0.978512	0.343750	0.680686	1.540793
11	0.566280	0.990083	0.500000	0.664409	1.511799
25	1.250689	0.990083	0.437500	0.659777	1.494686
50	2.475490	0.988430	0.500000	0.654955	1.452508
51	2.524968	0.986777	0.500000	0.654888	1.449999
75	3.671921	0.986777	0.531250	0.653380	1.464449
100	4.905395	0.986777	0.531250	0.654525	1.458547
101	4.957311	0.986777	0.531250	0.654513	1.459158
125	6.148598	0.985124	0.531250	0.654641	1.463410
150	7.394932	0.985124	0.562500	0.653584	1.463583
175	8.629890	0.985124	0.593750	0.652266	1.464887
200	9.826513	0.986777	0.562500	0.651749	1.464130
225	11.070833	0.986777	0.562500	0.651246	1.469562
250	12.307491	0.986777	0.531250	0.651044	1.474489
275	13.543307	0.986777	0.531250	0.650680	1.474231
300	14.774302	0.986777	0.531250	0.651105	1.473009
325	16.011706	0.986777	0.531250	0.650745	1.471707
350	17.252563	0.985124	0.531250	0.650717	1.472394
375	18.483729	0.985124	0.531250	0.650435	1.470567
400	19.723438	0.985124	0.531250	0.650402	1.466977
425	20.945622	0.985124	0.562500	0.650508	1.465642
450	22.180371	0.985124	0.562500	0.650782	1.467142
475	23.409597	0.985124	0.562500	0.650859	1.465050
500	24.646304	0.985124	0.562500	0.650919	1.464268
501	24.696165	0.985124	0.562500	0.650933	1.464105

525	25.896282	0.985124	0.562500	0.651089	1.462599
550	27.127370	0.985124	0.562500	0.650914	1.462708
575	28.358254	0.985124	0.562500	0.650868	1.461556
600	29.592859	0.985124	0.562500	0.650953	1.461739
625	30.831760	0.985124	0.562500	0.651061	1.461140
650	32.011633	0.985124	0.562500	0.651141	1.458550
675	33.242192	0.985124	0.562500	0.650999	1.458966
700	34.463159	0.985124	0.562500	0.651224	1.457966
725	35.696096	0.985124	0.562500	0.651000	1.456534
750	36.889265	0.985124	0.562500	0.650768	1.456302
775	38.135262	0.985124	0.562500	0.650790	1.456854
800	39.425860	0.985124	0.562500	0.650941	1.456730
825	40.673967	0.985124	0.562500	0.651068	1.458916
850	41.908425	0.985124	0.562500	0.650863	1.458667
875	43.143179	0.985124	0.562500	0.650760	1.457046
900	44.379394	0.985124	0.562500	0.650671	1.457002
925	45.611772	0.985124	0.562500	0.650647	1.456884
950	46.849196	0.985124	0.562500	0.650476	1.456250
975	48.090197	0.985124	0.562500	0.650427	1.455723
1000	49.330637	0.985124	0.562500	0.650386	1.456912
1001	49.381336	0.985124	0.562500	0.650351	1.456776
1025	50.564925	0.985124	0.562500	0.650380	1.456156
1050	51.800937	0.985124	0.562500	0.650447	1.456965
1075	53.040241	0.985124	0.562500	0.650443	1.458007
1100	54.277947	0.985124	0.562500	0.650489	1.457774
1125	55.518080	0.985124	0.562500	0.650404	1.457202
1150	56.745447	0.985124	0.562500	0.650408	1.457990
1175	57.920416	0.985124	0.562500	0.650527	1.458698
1200	59.160483	0.985124	0.562500	0.650495	1.458192
1225	60.354893	0.985124	0.562500	0.650548	1.458292
1250	61.590631	0.985124	0.562500	0.650420	1.457749

1275	62.753303	0.985124	0.562500	0.650304	1.457173
1300	63.993493	0.985124	0.562500	0.650402	1.456854
1325	65.196012	0.985124	0.562500	0.650539	1.457009
1350	66.428554	0.985124	0.562500	0.650505	1.456863
1375	67.638553	0.985124	0.562500	0.650403	1.456399
1400	68.851376	0.985124	0.562500	0.650382	1.456680
1425	70.116222	0.985124	0.562500	0.650317	1.457428
1450	71.343629	0.985124	0.562500	0.650420	1.458323
1475	72.588915	0.985124	0.562500	0.650578	1.458501
1500	73.835285	0.985124	0.531250	0.650587	1.458779
1525	75.070740	0.985124	0.562500	0.650591	1.458245
1550	76.319315	0.985124	0.562500	0.650565	1.458382
1575	77.555052	0.985124	0.562500	0.650520	1.458086
1600	78.802081	0.985124	0.562500	0.650423	1.457679
1625	80.030131	0.985124	0.562500	0.650415	1.457603
1650	81.254296	0.985124	0.562500	0.650407	1.457697
1675	82.485882	0.985124	0.562500	0.650378	1.457572
1700	83.720258	0.985124	0.562500	0.650330	1.457837
1725	84.903269	0.985124	0.562500	0.650328	1.457584
1750	86.135472	0.985124	0.562500	0.650335	1.457258
1775	87.371512	0.985124	0.562500	0.650275	1.457091
1800	88.600510	0.985124	0.562500	0.650371	1.457680
1825	89.801200	0.985124	0.562500	0.650384	1.457521
1850	90.962386	0.985124	0.562500	0.650313	1.457055
1875	92.288540	0.985124	0.562500	0.650385	1.457145
1900	93.583934	0.985124	0.562500	0.650352	1.457971
1925	94.885882	0.985124	0.562500	0.650370	1.458339
1950	96.193078	0.985124	0.562500	0.650385	1.458882
1975	97.466232	0.985124	0.562500	0.650321	1.458941
2000	98.707760	0.985124	0.562500	0.650397	1.458878

Accuracy : \m0.561403508772

Confusion Matrix :

+--------------+-----------------+-------+

| target_label | predicted_label | count |

+--------------+-----------------+-------+

| L | F | 4 |

| R | LF | 1 |

| LF | AU | 1 |

| R | NR | 3 |

| RF | RF | 8 |

| AU | RF | 5 |

| F | AU | 1 |

| NR | F | 1 |

| F | LF | 5 |

| R | AU | 3 |

+--------------+-----------------+-------+

[41 rows x 3 columns]

D.3 Naïve Bayes

wrong: 35

right: 43

accuracy: 0.551282

45 degree Weighted Con Matrix:

L	SL	S	SR	R	AU	NR
4	2	0	0	1	0	0
2	2	2	1	1	0	0
1	1	10	3	1	1	0
1	0	1	5	2	0	0
0	0	0	0	3	2	1
2	2	1	3	1	9	0
0	0	0	0	0	3	10

this weighted Amount is 1 .5 0 0 0 0 0(from the center diagonal

conMat weighted Amt: 52.5

count: 78

new Accuracy: 0.673077

90 degree Weighted Con Matrix:

L	SL	S	SR	R	AU	NR
4	2	0	0	1	0	0
2	2	2	1	1	0	0
1	1	10	3	1	1	0
1	0	1	5	2	0	0
0	0	0	0	3	2	1
2	2	1	3	1	9	0
0	0	0	0	0	3	10

this weighted Amount is 1 .75 .5 0 0 0 0(from the center diagonal

conMat weighted Amt: 61.75

count: 78

new Accuracy: 0.791667