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
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.6 Mobile Split</td>
<td>49</td>
</tr>
<tr>
<td>A.7 Mobile Classify</td>
<td>51</td>
</tr>
<tr>
<td>A.8 Turi Train</td>
<td>53</td>
</tr>
<tr>
<td>A.9 Turi Auto Pick</td>
<td>55</td>
</tr>
<tr>
<td>A.10 Naïve Bayes CPP</td>
<td>58</td>
</tr>
<tr>
<td>A.11 Naïve Bayes Classifier</td>
<td>66</td>
</tr>
<tr>
<td>A.12 GS Method</td>
<td>69</td>
</tr>
<tr>
<td>B. TensorFlow Object Detection API Documentation</td>
<td>73</td>
</tr>
<tr>
<td>C. Pix4D Data</td>
<td>81</td>
</tr>
<tr>
<td>D. Results</td>
<td>92</td>
</tr>
<tr>
<td>D.1 Turi Create CNN</td>
<td>92</td>
</tr>
<tr>
<td>D.2 Turi Create Other</td>
<td>108</td>
</tr>
<tr>
<td>D.3 Naïve Bayesian</td>
<td>119</td>
</tr>
</tbody>
</table>
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)](image-url)
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](image)

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)](image)

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)](image)
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).

![Convolutional Neural Network](image)

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](image)

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.
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:

![sUAS Control View](image)

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.

<table>
<thead>
<tr>
<th></th>
<th>Road</th>
<th>Not Road</th>
<th>Total</th>
<th>Recognition (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Road</td>
<td>17</td>
<td>3</td>
<td>20</td>
<td>85.00</td>
</tr>
<tr>
<td>Not Road</td>
<td>3</td>
<td>77</td>
<td>80</td>
<td>96.25</td>
</tr>
<tr>
<td>Total</td>
<td>20</td>
<td>80</td>
<td>100</td>
<td>94.00</td>
</tr>
</tbody>
</table>

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.

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.

<table>
<thead>
<tr>
<th>Direction</th>
<th>Label</th>
<th># of images used to train</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>L</td>
<td>100</td>
</tr>
<tr>
<td>Left Straight</td>
<td>LS</td>
<td>100</td>
</tr>
<tr>
<td>Straight</td>
<td>S</td>
<td>100</td>
</tr>
<tr>
<td>Right Straight</td>
<td>RS</td>
<td>100</td>
</tr>
<tr>
<td>Right</td>
<td>R</td>
<td>100</td>
</tr>
<tr>
<td>Ask User</td>
<td>AU</td>
<td>100</td>
</tr>
<tr>
<td>No road in image</td>
<td>NR</td>
<td>100</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Regression</td>
<td>37.00</td>
</tr>
<tr>
<td>K-Nearest Neighbor</td>
<td>50.00</td>
</tr>
<tr>
<td>Boosted Trees</td>
<td>47.00</td>
</tr>
<tr>
<td>Random Forest</td>
<td>56.00</td>
</tr>
</tbody>
</table>

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 $0.67 \times 0.67 \times 0.67 = 0.3$ 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


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


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


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

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
    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)

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))
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
    }
}
/* 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{
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]
    }
}

override func prepare(for segue: UIStoryboardSegue, sender: Any!) {
    if segue.identifier == "goToFlight") {
        let svc = segue.destination as!
        DefaultLayoutCustomizationViewController
        svc.userPickDict = self.dictPassed
    }
}
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 ****/
```swift
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"
    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 {


44
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: "%0.3f",value)) "
                    /*i = i + 1
                    if (i % 5 == 0) {
                        outputString += "\n"
                    }*/
                }
            }
            self.errorLabel.text = outputString
        }
    })
}

self.errorLabel.text = "refresh error"
return
@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()
    }
}
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 UIImage as a param, 
// classifies those images and outputs an array of dictionaries 
// containing strings (labels) and doubles whose 
// value is 0 - 1 (being the confidence rating of the 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  TuriTrain

# 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))
# 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: \n%.2f\% %s") % metrics['accuracy']
print("Confusion Matrix: \\
" % 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.\n"""
)
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 trainClassifier =
    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, trainClassifier);
    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
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
for (int i = 0; i < 7; i++)
{
    for (int j = 0; j < 7; j++)
    {
        cout << confuMatrix[i][j] << " \t";
        outputFile << confuMatrix[i][j] << " \t";
    }
    cout << endl;
    outputFile << endl;
}
```cpp
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;
}

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)
  {
```

62
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);
    }
}
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 classTupkeCounts[classCount] = {};
auto means = new double[dimensionCount][classCount];
zero(means, dimensionCount);
// Put per-class sums of attribute values in means and build classTupkeCounts
for (int tuple = 0; tuple < tupleCount; tuple++) {
    classTupkeCounts[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] /= classTupkeCounts[clazz];
    }
priorProbabilities[clazz] = double(classTupkeCounts[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] /= classTupkeCounts[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

```cpp
#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++)
```cpp
{ 
    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 environments

Create an environment:
   cd into the environment folder
   > cd environments
   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

close object detection api repository

> 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
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:

```python
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.')
```

```bash
> cd 'wherever\objectDetect
> python xml_to_csv.py
```

In 'generate_tfrecord.py':

```python
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,
```

```bash
> 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_detection
start training

> cd 'wherever\models\research\object_detection'

> 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))
<table>
<thead>
<tr>
<th>Picture Name</th>
<th>Lat</th>
<th>Long</th>
<th>Altitude</th>
<th>Distance From Last Pic (meter)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DJI_0801.JPG</td>
<td>43.93856228</td>
<td>-115.9714383</td>
<td>1486.114014</td>
<td></td>
</tr>
<tr>
<td>DJI_0802.JPG</td>
<td>43.93858978</td>
<td>-115.9714326</td>
<td>1486.313965</td>
<td>3.09</td>
</tr>
<tr>
<td>DJI_0803.JPG</td>
<td>43.93872422</td>
<td>-115.9714416</td>
<td>1486.313965</td>
<td>14.97</td>
</tr>
<tr>
<td>DJI_0804.JPG</td>
<td>43.93889028</td>
<td>-115.9714519</td>
<td>1486.41394</td>
<td>18.48</td>
</tr>
<tr>
<td>DJI_0805.JPG</td>
<td>43.93908031</td>
<td>-115.971462</td>
<td>1486.41394</td>
<td>21.15</td>
</tr>
<tr>
<td>DJI_0806.JPG</td>
<td>43.93927228</td>
<td>-115.9714711</td>
<td>1486.41394</td>
<td>21.36</td>
</tr>
<tr>
<td>DJI_0807.JPG</td>
<td>43.93946653</td>
<td>-115.9714741</td>
<td>1486.313965</td>
<td>21.60</td>
</tr>
<tr>
<td>DJI_0808.JPG</td>
<td>43.93975419</td>
<td>-115.9714729</td>
<td>1486.313965</td>
<td>31.99</td>
</tr>
<tr>
<td>DJI_0809.JPG</td>
<td>43.93994389</td>
<td>-115.9714711</td>
<td>1486.514038</td>
<td>21.09</td>
</tr>
<tr>
<td>DJI_0810.JPG</td>
<td>43.94012772</td>
<td>-115.9714705</td>
<td>1486.514038</td>
<td>20.44</td>
</tr>
<tr>
<td>DJI_0811.JPG</td>
<td>43.94031578</td>
<td>-115.971467</td>
<td>1486.41394</td>
<td>20.91</td>
</tr>
<tr>
<td>DJI_0812.JPG</td>
<td>43.94050819</td>
<td>-115.9714612</td>
<td>1486.313965</td>
<td>21.40</td>
</tr>
<tr>
<td>DJI_0813.JPG</td>
<td>43.94079875</td>
<td>-115.9714514</td>
<td>1486.213989</td>
<td>32.32</td>
</tr>
<tr>
<td>DJI_0814.JPG</td>
<td>43.94099481</td>
<td>-115.9714399</td>
<td>1486.313965</td>
<td>21.82</td>
</tr>
<tr>
<td>DJI_0815.JPG</td>
<td>43.94118689</td>
<td>-115.9714177</td>
<td>1486.313965</td>
<td>21.43</td>
</tr>
<tr>
<td>DJI_0816.JPG</td>
<td>43.94130853</td>
<td>-115.9712984</td>
<td>1486.213989</td>
<td>16.56</td>
</tr>
<tr>
<td>DJI_0817.JPG</td>
<td>43.94132069</td>
<td>-115.9711058</td>
<td>1486.114014</td>
<td>15.49</td>
</tr>
<tr>
<td>DJI_0818.JPG</td>
<td>43.94128175</td>
<td>-115.9708601</td>
<td>1486.213989</td>
<td>20.14</td>
</tr>
<tr>
<td>DJI_0819.JPG</td>
<td>43.94121411</td>
<td>-115.970461</td>
<td>1486.313965</td>
<td>32.82</td>
</tr>
<tr>
<td>DJI_0820.JPG</td>
<td>43.94110964</td>
<td>-115.9702274</td>
<td>1486.213989</td>
<td>22.02</td>
</tr>
<tr>
<td>DJI_0821.JPG</td>
<td>43.94098131</td>
<td>-115.9700166</td>
<td>1486.114014</td>
<td>22.10</td>
</tr>
<tr>
<td>DJI_0822.JPG</td>
<td>43.94080414</td>
<td>-115.9699323</td>
<td>1486.014038</td>
<td>20.82</td>
</tr>
<tr>
<td>DJI_0823.JPG</td>
<td>43.9406015</td>
<td>-115.9699434</td>
<td>1486.014038</td>
<td>22.55</td>
</tr>
<tr>
<td>DJI_0824.JPG</td>
<td>43.94029433</td>
<td>-115.9699912</td>
<td>1486.213989</td>
<td>34.37</td>
</tr>
<tr>
<td>DJI_0825.JPG</td>
<td>43.94009114</td>
<td>-115.9700197</td>
<td>1486.213989</td>
<td>22.71</td>
</tr>
<tr>
<td>DJI_0826.JPG</td>
<td>43.93989564</td>
<td>-115.9700394</td>
<td>1486.213989</td>
<td>21.80</td>
</tr>
<tr>
<td>DJI_0827.JPG</td>
<td>43.93969794</td>
<td>-115.9700464</td>
<td>1486.213989</td>
<td>21.99</td>
</tr>
<tr>
<td>DJI_0828.JPG</td>
<td>43.93950475</td>
<td>-115.9700475</td>
<td>1486.213989</td>
<td>21.48</td>
</tr>
<tr>
<td>DJI_0829.JPG</td>
<td>43.93921861</td>
<td>-115.9700461</td>
<td>1486.213989</td>
<td>31.82</td>
</tr>
<tr>
<td>DJI_0830.JPG</td>
<td>43.93902678</td>
<td>-115.9700373</td>
<td>1486.313965</td>
<td>21.34</td>
</tr>
<tr>
<td>DJI_0831.JPG</td>
<td>43.93883694</td>
<td>-115.9700276</td>
<td>1486.213989</td>
<td>21.12</td>
</tr>
<tr>
<td>DJI_0832.JPG</td>
<td>43.93864511</td>
<td>-115.9700211</td>
<td>1486.313965</td>
<td>21.34</td>
</tr>
<tr>
<td>DJI_0833.JPG</td>
<td>43.93845617</td>
<td>-115.9700166</td>
<td>1486.213989</td>
<td>21.01</td>
</tr>
<tr>
<td>DJI_0834.JPG</td>
<td>43.93826183</td>
<td>-115.9700166</td>
<td>1486.213989</td>
<td>21.61</td>
</tr>
<tr>
<td>DJI_0835.JPG</td>
<td>43.93797778</td>
<td>-115.9700043</td>
<td>1486.213989</td>
<td>31.60</td>
</tr>
<tr>
<td>DJI_0836.JPG</td>
<td>43.93784994</td>
<td>-115.9698949</td>
<td>1486.313965</td>
<td>16.70</td>
</tr>
</tbody>
</table>
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 23.20
DJI_0131.JPG 43.94019214 -115.9755226 1487.223022 22.90
DJI_0132.JPG 43.94030867 -115.9757401 1487.422974 22.23
DJI_0133.JPG 43.94056656 -115.9762986 1487.422974 31.89
DJI_0134.JPG 43.94067617 -115.9765199 1487.422974 21.50
DJI_0135.JPG 43.94078478 -115.9767371 1487.422974 21.18
DJI_0136.JPG 43.94082697 -115.9768436 1487.123047 9.73
DJI_0137.JPG 43.94082675 -115.9768518 1487.02949 0.65
DJI_0138.JPG 43.94084122 -115.9768735 1487.322974 2.37
DJI_0139.JPG 43.94090422 -115.9769931 1487.322974 11.86
DJI_0140.JPG 43.94100128 -115.9771808 1487.322974 18.51
DJI_0141.JPG 43.94116819 -115.9775131 1487.322974 32.44
DJI_0142.JPG 43.94121961 -115.9777657 1487.422974 21.02
DJI_0143.JPG 43.94121994 -115.9779791 1487.123047 17.09
DJI_0144.JPG 43.94121333 -115.9780154 1487.223022 3.00
DJI_0145.JPG 43.94120286 -115.9780578 1487.422974 3.59
DJI_0146.JPG 43.94113397 -115.9781631 1487.422974 11.39
DJI_0147.JPG 43.94098953 -115.9782139 1487.422974 16.57
DJI_0148.JPG 43.94080622 -115.978204 1487.422974 20.40
DJI_0149.JPG 43.94062486 -115.9781195 1487.422974 21.27
DJI_0150.JPG 43.94040875 -115.9778604 1487.522949 31.75
DJI_0151.JPG 43.94030253 -115.9776326 1487.522949 21.73
DJI_0152.JPG 43.94019606 -115.9773993 1487.422974 22.12
DJI_0153.JPG 43.94008592 -115.977165 1487.422974 22.40
DJI_0154.JPG 43.93997022 -115.9769283 1487.322998 22.91
DJI_0155.JPG 43.93979656 -115.9765734 1487.223022 34.35
DJI_0156.JPG 43.93967994 -115.9763376 1487.322998 22.91
DJI_0157.JPG 43.93956644 -115.9761072 1487.322998 22.35
DJI_0158.JPG 43.93945061 -115.9758741 1487.322998 22.68
DJI_0159.JPG 43.93933422 -115.9756428 1487.223022 22.59
DJI_0160.JPG 43.93923594 -115.9753993 1487.223022 22.36
DJI_0161.JPG 43.93913747 -115.9750094 1487.322998 33.08
DJI_0162.JPG 43.93907933 -115.974746 1487.422974 22.06
DJI_0163.JPG 43.939025 -115.974477 1487.422974 22.37
DJI_0164.JPG 43.93897278 -115.9742136 1487.322998 21.87
DJI_0165.JPG 43.93894036 -115.97409 1487.223022 10.54
DJI_0166.JPG 43.93893575 -115.9740837 1487.322998 0.72
<table>
<thead>
<tr>
<th>filename</th>
<th>latitude</th>
<th>longitude</th>
<th>altitude</th>
<th>temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>DJI_0209.JPG</td>
<td>43.94103144</td>
<td>-115.9762433</td>
<td>1486.922974</td>
<td>21.64</td>
</tr>
<tr>
<td>DJI_0210.JPG</td>
<td>43.94093483</td>
<td>-115.9760076</td>
<td>1487.022949</td>
<td>21.72</td>
</tr>
<tr>
<td>DJI_0211.JPG</td>
<td>43.94083728</td>
<td>-115.9757703</td>
<td>1486.922974</td>
<td>21.88</td>
</tr>
<tr>
<td>DJI_0212.JPG</td>
<td>43.94071081</td>
<td>-115.9753921</td>
<td>1486.922974</td>
<td>33.39</td>
</tr>
<tr>
<td>DJI_0213.JPG</td>
<td>43.94062569</td>
<td>-115.9751376</td>
<td>1486.822998</td>
<td>22.47</td>
</tr>
<tr>
<td>DJI_0214.JPG</td>
<td>43.94053806</td>
<td>-115.9748821</td>
<td>1486.822998</td>
<td>22.66</td>
</tr>
<tr>
<td>DJI_0215.JPG</td>
<td>43.94045014</td>
<td>-115.9746317</td>
<td>1486.822998</td>
<td>22.31</td>
</tr>
<tr>
<td>DJI_0216.JPG</td>
<td>43.94031836</td>
<td>-115.9742673</td>
<td>1486.922974</td>
<td>32.65</td>
</tr>
<tr>
<td>DJI_0217.JPG</td>
<td>43.94022161</td>
<td>-115.9740366</td>
<td>1487.022949</td>
<td>21.38</td>
</tr>
<tr>
<td>DJI_0218.JPG</td>
<td>43.94011875</td>
<td>-115.9738096</td>
<td>1487.022949</td>
<td>21.47</td>
</tr>
<tr>
<td>DJI_0219.JPG</td>
<td>43.94001319</td>
<td>-115.9735806</td>
<td>1487.022949</td>
<td>21.77</td>
</tr>
<tr>
<td>DJI_0220.JPG</td>
<td>43.93990906</td>
<td>-115.9733508</td>
<td>1487.022949</td>
<td>21.74</td>
</tr>
<tr>
<td>DJI_0221.JPG</td>
<td>43.93980356</td>
<td>-115.9731204</td>
<td>1486.922974</td>
<td>21.86</td>
</tr>
<tr>
<td>DJI_0222.JPG</td>
<td>43.93964764</td>
<td>-115.9727657</td>
<td>1486.922974</td>
<td>33.28</td>
</tr>
<tr>
<td>DJI_0223.JPG</td>
<td>43.93952725</td>
<td>-115.9725458</td>
<td>1486.922974</td>
<td>22.12</td>
</tr>
<tr>
<td>DJI_0224.JPG</td>
<td>43.93937133</td>
<td>-115.9723819</td>
<td>1486.922974</td>
<td>21.75</td>
</tr>
<tr>
<td>DJI_0225.JPG</td>
<td>43.93919372</td>
<td>-115.9722604</td>
<td>1486.922974</td>
<td>22.02</td>
</tr>
<tr>
<td>DJI_0226.JPG</td>
<td>43.93901464</td>
<td>-115.9721417</td>
<td>1486.922974</td>
<td>22.06</td>
</tr>
<tr>
<td>DJI_0227.JPG</td>
<td>43.93884764</td>
<td>-115.9719956</td>
<td>1486.922974</td>
<td>21.95</td>
</tr>
<tr>
<td>DJI_0228.JPG</td>
<td>43.93869819</td>
<td>-115.9718301</td>
<td>1486.822998</td>
<td>21.26</td>
</tr>
</tbody>
</table>

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 |
+--------------+-----------------+-------+
| road         | notRoad         | 3     |
| notRoad      | notRoad         | 48    |
| notRoad      | road            | 12    |
| road         | road            | 37    |
+------------------------------------------+
<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>67</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>13</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>20</td>
</tr>
</tbody>
</table>

Accuracy for imagesToClassify3:
0.94

Confusion Matrix for imagesToClassify3:

**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:
<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>road</td>
<td>notRoad</td>
<td>9</td>
</tr>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>52</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>8</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>31</td>
</tr>
</tbody>
</table>

Accuracy for imagesToClassify2:
0.72

Confusion Matrix for imagesToClassify2:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>road</td>
<td>notRoad</td>
<td>1</td>
</tr>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>53</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>27</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>19</td>
</tr>
</tbody>
</table>

Accuracy for imagesToClassify3:
0.89

Confusion Matrix for imagesToClassify3:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
</table>
**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:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>target_label</td>
<td>predicted_label</td>
<td>count</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>road</td>
<td>notRoad</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>notRoad</td>
<td>notRoad</td>
<td>71</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>notRoad</td>
<td>road</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>road</td>
<td>road</td>
<td>18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Accuracy for imagesToClassify2:

0.8

Confusion Matrix for imagesToClassify2:
<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>60</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>20</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>20</td>
</tr>
</tbody>
</table>

Accuracy for imagesToClassify3:
0.88

Confusion Matrix for imagesToClassify3:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>road</td>
<td>notRoad</td>
<td>4</td>
</tr>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>70</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>8</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>18</td>
</tr>
</tbody>
</table>

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

Accuracies: 0.85, 0.81, 0.85

avg = 0.83666

Accuracy for imagesToClassify1:
Confusion Matrix for imagesToClassify1:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>road</td>
<td>notRoad</td>
<td>5</td>
</tr>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>50</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>10</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>35</td>
</tr>
</tbody>
</table>

Accuracy for imagesToClassify2:

0.81

Confusion Matrix for imagesToClassify2:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>61</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>19</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>20</td>
</tr>
</tbody>
</table>

Accuracy for imagesToClassify3:

0.85

Confusion Matrix for imagesToClassify3:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
</table>
**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:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>road</td>
<td>notRoad</td>
<td>4</td>
</tr>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>67</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>11</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>18</td>
</tr>
</tbody>
</table>

Total: 98
Accuracy for imagesToClassify2:
0.7

Confusion Matrix for imagesToClassify2:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>50</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>30</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>20</td>
</tr>
</tbody>
</table>

Accuracy for imagesToClassify3:
0.86

Confusion Matrix for imagesToClassify3:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>road</td>
<td>notRoad</td>
<td>1</td>
</tr>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>65</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>13</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>21</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>72</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>6</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>22</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>road</td>
<td>notRoad</td>
<td>5</td>
</tr>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>53</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>7</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>35</td>
</tr>
</tbody>
</table>
**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

---

**Accuracy for imagesToClassify2:**

0.86

**Confusion Matrix for imagesToClassify2:**

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>road</td>
<td>notRoad</td>
<td>1</td>
</tr>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>67</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>13</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>19</td>
</tr>
</tbody>
</table>

---

**Accuracy for imagesToClassify3:**

0.95

**Confusion Matrix for imagesToClassify3:**

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>73</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>5</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>22</td>
</tr>
</tbody>
</table>
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    |
```
Accuracy for imagesToClassify3:
0.96

Confusion Matrix for imagesToClassify3:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>74</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>4</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>22</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>road</td>
<td>notRoad</td>
<td>4</td>
</tr>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>55</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>5</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>36</td>
</tr>
</tbody>
</table>

Accuracy for imagesToClassify2:
0.89

Confusion Matrix for imagesToClassify2:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>road</td>
<td>notRoad</td>
<td>3</td>
</tr>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>72</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>8</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>17</td>
</tr>
</tbody>
</table>

Accuracy for imagesToClassify3:
0.91

Confusion Matrix for imagesToClassify3:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>road</td>
<td>notRoad</td>
<td>2</td>
</tr>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>71</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>7</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>road</td>
<td>notRoad</td>
<td>5</td>
</tr>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>54</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>6</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>35</td>
</tr>
</tbody>
</table>

Accuracy for imagesToClassify2:

0.94

Confusion Matrix for imagesToClassify2:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>target_label</td>
<td>predicted_label</td>
<td>count</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
<td>-------</td>
</tr>
<tr>
<td>road</td>
<td>notRoad</td>
<td>1</td>
</tr>
<tr>
<td>notRoad</td>
<td>road</td>
<td>1</td>
</tr>
<tr>
<td>notRoad</td>
<td>notRoad</td>
<td>77</td>
</tr>
<tr>
<td>road</td>
<td>road</td>
<td>21</td>
</tr>
</tbody>
</table>

Accuracy for imagesToClassify3:

0.98

Confusion Matrix for imagesToClassify3:
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

-----------------------------------------------

+++

<p>|</p>
<table>
<thead>
<tr>
<th>Iteration</th>
<th>Passes</th>
<th>Elapsed Time</th>
<th>Training-accuracy</th>
<th>Validation-accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1.033285</td>
<td>0.476510</td>
<td>0.363636</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1.043136</td>
<td>0.481544</td>
<td>0.333333</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1.054729</td>
<td>0.483221</td>
<td>0.303030</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1.064228</td>
<td>0.483221</td>
<td>0.303030</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>1.073549</td>
<td>0.483221</td>
<td>0.303030</td>
</tr>
</tbody>
</table>

SUCCESS: Optimal solution found.

Accuracy : \m0.374301675978

Confusion Matrix :

+---------------------+
| target_label | predicted_label | count |
+---------------------+
| NR          | F               | 4     |
| RF          | R               | 4     |
K-NN with max neighbors 5:
Starting ball tree nearest neighbors model training.

<table>
<thead>
<tr>
<th>Tree level</th>
<th>Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>957us</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query points</th>
<th>% Complete.</th>
<th>Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>656us</td>
</tr>
<tr>
<td>Done</td>
<td></td>
<td>6.744ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>class</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>0.4</td>
</tr>
<tr>
<td>AU</td>
<td>0.6</td>
</tr>
<tr>
<td>AU</td>
<td>0.6</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
[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.

<table>
<thead>
<tr>
<th>Query points</th>
<th>% Complete.</th>
<th>Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>915us</td>
</tr>
<tr>
<td>Done</td>
<td></td>
<td>7.08ms</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>680us</td>
</tr>
<tr>
<td>Done</td>
<td></td>
<td>6.559ms</td>
</tr>
</tbody>
</table>

{'accuracy': 0.5061728395061729}

Boosted trees classifier:

----------------------------------------
Number of examples : 627
Number of classes  : 7
Number of feature columns : 25
Number of unpacked features : 25

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Elapsed Time</th>
<th>Training-accuracy</th>
<th>Validation-accuracy</th>
<th>Training-log_loss</th>
<th>Validation-log_loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.061954</td>
<td>0.947368</td>
<td>0.424242</td>
<td>1.365759</td>
<td>1.727189</td>
</tr>
<tr>
<td>2</td>
<td>0.121393</td>
<td>0.984051</td>
<td>0.393939</td>
<td>1.015885</td>
<td>1.623551</td>
</tr>
<tr>
<td>3</td>
<td>0.185309</td>
<td>0.993620</td>
<td>0.424242</td>
<td>0.782035</td>
<td>1.504347</td>
</tr>
<tr>
<td>4</td>
<td>0.246836</td>
<td>0.998405</td>
<td>0.454545</td>
<td>0.610150</td>
<td>1.440168</td>
</tr>
<tr>
<td>5</td>
<td>0.308832</td>
<td>0.998405</td>
<td>0.484848</td>
<td>0.482705</td>
<td>1.393229</td>
</tr>
<tr>
<td>6</td>
<td>0.368625</td>
<td>1.000000</td>
<td>0.454545</td>
<td>0.387270</td>
<td>1.383487</td>
</tr>
<tr>
<td>11</td>
<td>0.649838</td>
<td>1.000000</td>
<td>0.454545</td>
<td>0.154015</td>
<td>1.289829</td>
</tr>
<tr>
<td>25</td>
<td>1.320851</td>
<td>1.000000</td>
<td>0.606061</td>
<td>0.035172</td>
<td>1.260491</td>
</tr>
<tr>
<td>50</td>
<td>2.200966</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.012643</td>
<td>1.316766</td>
</tr>
<tr>
<td>51</td>
<td>2.230356</td>
<td>1.000000</td>
<td>0.606061</td>
<td>0.012342</td>
<td>1.325593</td>
</tr>
<tr>
<td>75</td>
<td>2.886977</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.007841</td>
<td>1.383569</td>
</tr>
<tr>
<td>100</td>
<td>3.472501</td>
<td>1.000000</td>
<td>0.606061</td>
<td>0.005827</td>
<td>1.441169</td>
</tr>
<tr>
<td>101</td>
<td>3.495365</td>
<td>1.000000</td>
<td>0.606061</td>
<td>0.005773</td>
<td>1.442580</td>
</tr>
<tr>
<td>125</td>
<td>4.001885</td>
<td>1.000000</td>
<td>0.636364</td>
<td>0.004721</td>
<td>1.483739</td>
</tr>
<tr>
<td>150</td>
<td>4.491855</td>
<td>1.000000</td>
<td>0.636364</td>
<td>0.004011</td>
<td>1.517363</td>
</tr>
<tr>
<td>175</td>
<td>4.946271</td>
<td>1.000000</td>
<td>0.606061</td>
<td>0.003519</td>
<td>1.538021</td>
</tr>
<tr>
<td>200</td>
<td>5.383961</td>
<td>1.000000</td>
<td>0.606061</td>
<td>0.003164</td>
<td>1.564999</td>
</tr>
<tr>
<td>225</td>
<td>5.799371</td>
<td>1.000000</td>
<td>0.606061</td>
<td>0.002878</td>
<td>1.581720</td>
</tr>
<tr>
<td>250</td>
<td>6.203940</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.002660</td>
<td>1.597127</td>
</tr>
<tr>
<td>275</td>
<td>6.603608</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.002485</td>
<td>1.608908</td>
</tr>
<tr>
<td>300</td>
<td>6.986063</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.002341</td>
<td>1.621525</td>
</tr>
<tr>
<td>325</td>
<td>7.358658</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.002218</td>
<td>1.635224</td>
</tr>
<tr>
<td>350</td>
<td>7.720665</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.002113</td>
<td>1.647486</td>
</tr>
<tr>
<td>375</td>
<td>8.070499</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.002020</td>
<td>1.658406</td>
</tr>
<tr>
<td>400</td>
<td>8.418456</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001939</td>
<td>1.666826</td>
</tr>
<tr>
<td>425</td>
<td>8.760573</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001869</td>
<td>1.673186</td>
</tr>
<tr>
<td>1175</td>
<td>17.150369</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001101</td>
<td>1.753039</td>
</tr>
<tr>
<td>1200</td>
<td>17.403714</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001091</td>
<td>1.753427</td>
</tr>
<tr>
<td>1225</td>
<td>17.652991</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001082</td>
<td>1.755194</td>
</tr>
<tr>
<td>1250</td>
<td>17.899065</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001073</td>
<td>1.754683</td>
</tr>
<tr>
<td>1275</td>
<td>18.146625</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001065</td>
<td>1.756154</td>
</tr>
<tr>
<td>1300</td>
<td>18.392806</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001056</td>
<td>1.757219</td>
</tr>
<tr>
<td>1325</td>
<td>18.634988</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001049</td>
<td>1.758028</td>
</tr>
<tr>
<td>1350</td>
<td>18.880148</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001042</td>
<td>1.759759</td>
</tr>
<tr>
<td>1375</td>
<td>19.124286</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001035</td>
<td>1.761247</td>
</tr>
<tr>
<td>1400</td>
<td>19.367452</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001028</td>
<td>1.763095</td>
</tr>
<tr>
<td>1425</td>
<td>19.608575</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001021</td>
<td>1.764941</td>
</tr>
<tr>
<td>1450</td>
<td>19.851569</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001014</td>
<td>1.767218</td>
</tr>
<tr>
<td>1475</td>
<td>20.093274</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001008</td>
<td>1.768246</td>
</tr>
<tr>
<td>1500</td>
<td>20.334845</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.001002</td>
<td>1.768659</td>
</tr>
<tr>
<td>1525</td>
<td>20.568953</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000996</td>
<td>1.770970</td>
</tr>
<tr>
<td>1550</td>
<td>20.805348</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000990</td>
<td>1.773814</td>
</tr>
<tr>
<td>1575</td>
<td>21.040191</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000985</td>
<td>1.776399</td>
</tr>
<tr>
<td>1600</td>
<td>21.274824</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000980</td>
<td>1.779022</td>
</tr>
<tr>
<td>1625</td>
<td>21.507000</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000975</td>
<td>1.781187</td>
</tr>
<tr>
<td>1650</td>
<td>21.717132</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000970</td>
<td>1.782989</td>
</tr>
<tr>
<td>1675</td>
<td>21.932951</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000965</td>
<td>1.786066</td>
</tr>
<tr>
<td>1700</td>
<td>22.153734</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000961</td>
<td>1.788303</td>
</tr>
<tr>
<td>1725</td>
<td>22.380398</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000956</td>
<td>1.790236</td>
</tr>
<tr>
<td>1750</td>
<td>22.601821</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000952</td>
<td>1.791834</td>
</tr>
<tr>
<td>1775</td>
<td>22.807907</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000948</td>
<td>1.793008</td>
</tr>
<tr>
<td>1800</td>
<td>23.009734</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000944</td>
<td>1.793883</td>
</tr>
<tr>
<td>1825</td>
<td>23.221458</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000940</td>
<td>1.795937</td>
</tr>
<tr>
<td>1850</td>
<td>23.441011</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000936</td>
<td>1.796873</td>
</tr>
<tr>
<td>1875</td>
<td>23.660857</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000932</td>
<td>1.798477</td>
</tr>
<tr>
<td>1900</td>
<td>23.876100</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000929</td>
<td>1.800473</td>
</tr>
<tr>
<td>1925</td>
<td>24.090995</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000925</td>
<td>1.801744</td>
</tr>
<tr>
<td>Year</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1950</td>
<td>24.306340</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000922</td>
<td>1.802770</td>
</tr>
<tr>
<td>1975</td>
<td>24.521544</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000918</td>
<td>1.804650</td>
</tr>
<tr>
<td>2000</td>
<td>24.734381</td>
<td>1.000000</td>
<td>0.575758</td>
<td>0.000915</td>
<td>1.806006</td>
</tr>
</tbody>
</table>

Accuracy: \m0.472972972973

Confusion Matrix:

<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>LF</td>
<td>3</td>
</tr>
<tr>
<td>LF</td>
<td>NR</td>
<td>1</td>
</tr>
<tr>
<td>RF</td>
<td>R</td>
<td>2</td>
</tr>
<tr>
<td>LF</td>
<td>AU</td>
<td>3</td>
</tr>
<tr>
<td>L</td>
<td>RF</td>
<td>1</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>9</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>17</td>
</tr>
<tr>
<td>LF</td>
<td>RF</td>
<td>1</td>
</tr>
<tr>
<td>RF</td>
<td>L</td>
<td>3</td>
</tr>
<tr>
<td>R</td>
<td>F</td>
<td>1</td>
</tr>
</tbody>
</table>

[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
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.058713</td>
<td>0.828099</td>
<td>0.437500</td>
<td>0.784255</td>
<td>1.584066</td>
</tr>
<tr>
<td>2</td>
<td>0.112848</td>
<td>0.925620</td>
<td>0.468750</td>
<td>0.705202</td>
<td>1.488792</td>
</tr>
<tr>
<td>3</td>
<td>0.164260</td>
<td>0.937190</td>
<td>0.406250</td>
<td>0.705566</td>
<td>1.455549</td>
</tr>
<tr>
<td>4</td>
<td>0.215903</td>
<td>0.973554</td>
<td>0.468750</td>
<td>0.699122</td>
<td>1.505193</td>
</tr>
<tr>
<td>5</td>
<td>0.267798</td>
<td>0.971901</td>
<td>0.437500</td>
<td>0.691860</td>
<td>1.544475</td>
</tr>
<tr>
<td>6</td>
<td>0.317276</td>
<td>0.978512</td>
<td>0.343750</td>
<td>0.680686</td>
<td>1.540793</td>
</tr>
<tr>
<td>11</td>
<td>0.566280</td>
<td>0.990083</td>
<td>0.500000</td>
<td>0.659777</td>
<td>1.494686</td>
</tr>
<tr>
<td>25</td>
<td>1.250689</td>
<td>0.990083</td>
<td>0.437500</td>
<td>0.659777</td>
<td>1.494686</td>
</tr>
<tr>
<td>50</td>
<td>2.475490</td>
<td>0.988430</td>
<td>0.500000</td>
<td>0.654955</td>
<td>1.452508</td>
</tr>
<tr>
<td>51</td>
<td>2.524968</td>
<td>0.986777</td>
<td>0.500000</td>
<td>0.654888</td>
<td>1.449999</td>
</tr>
<tr>
<td>75</td>
<td>3.671921</td>
<td>0.986777</td>
<td>0.531250</td>
<td>0.653380</td>
<td>1.464449</td>
</tr>
<tr>
<td>100</td>
<td>4.905395</td>
<td>0.986777</td>
<td>0.531250</td>
<td>0.654525</td>
<td>1.458547</td>
</tr>
<tr>
<td>101</td>
<td>4.957311</td>
<td>0.986777</td>
<td>0.531250</td>
<td>0.654513</td>
<td>1.459158</td>
</tr>
<tr>
<td>125</td>
<td>6.148598</td>
<td>0.985124</td>
<td>0.531250</td>
<td>0.654641</td>
<td>1.463410</td>
</tr>
<tr>
<td>150</td>
<td>7.394932</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.653584</td>
<td>1.463583</td>
</tr>
<tr>
<td>175</td>
<td>8.629890</td>
<td>0.985124</td>
<td>0.593750</td>
<td>0.652266</td>
<td>1.464887</td>
</tr>
<tr>
<td>200</td>
<td>9.826513</td>
<td>0.986777</td>
<td>0.562500</td>
<td>0.651749</td>
<td>1.464130</td>
</tr>
<tr>
<td>225</td>
<td>11.070833</td>
<td>0.986777</td>
<td>0.562500</td>
<td>0.651246</td>
<td>1.469562</td>
</tr>
<tr>
<td>250</td>
<td>12.307491</td>
<td>0.986777</td>
<td>0.531250</td>
<td>0.651044</td>
<td>1.474489</td>
</tr>
<tr>
<td>275</td>
<td>13.543307</td>
<td>0.986777</td>
<td>0.531250</td>
<td>0.650680</td>
<td>1.474231</td>
</tr>
<tr>
<td>300</td>
<td>14.774302</td>
<td>0.986777</td>
<td>0.531250</td>
<td>0.651105</td>
<td>1.473009</td>
</tr>
<tr>
<td>325</td>
<td>16.011706</td>
<td>0.986777</td>
<td>0.531250</td>
<td>0.650745</td>
<td>1.471707</td>
</tr>
<tr>
<td>350</td>
<td>17.252563</td>
<td>0.985124</td>
<td>0.531250</td>
<td>0.650717</td>
<td>1.472394</td>
</tr>
<tr>
<td>375</td>
<td>18.483729</td>
<td>0.985124</td>
<td>0.531250</td>
<td>0.650435</td>
<td>1.470567</td>
</tr>
<tr>
<td>400</td>
<td>19.723438</td>
<td>0.985124</td>
<td>0.531250</td>
<td>0.650402</td>
<td>1.469777</td>
</tr>
<tr>
<td>425</td>
<td>20.945622</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650508</td>
<td>1.465642</td>
</tr>
<tr>
<td>450</td>
<td>22.180371</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650782</td>
<td>1.467142</td>
</tr>
<tr>
<td>475</td>
<td>23.409597</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650859</td>
<td>1.465050</td>
</tr>
<tr>
<td>500</td>
<td>24.646304</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650919</td>
<td>1.464268</td>
</tr>
<tr>
<td>501</td>
<td>24.696165</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650933</td>
<td>1.464105</td>
</tr>
<tr>
<td>525</td>
<td>25.896282</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.651089</td>
<td>1.462599</td>
</tr>
<tr>
<td>550</td>
<td>27.127370</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650914</td>
<td>1.462708</td>
</tr>
<tr>
<td>575</td>
<td>28.358254</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650868</td>
<td>1.461556</td>
</tr>
<tr>
<td>600</td>
<td>29.592859</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650953</td>
<td>1.461739</td>
</tr>
<tr>
<td>625</td>
<td>30.831760</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.651061</td>
<td>1.461140</td>
</tr>
<tr>
<td>650</td>
<td>32.011633</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650868</td>
<td>1.458550</td>
</tr>
<tr>
<td>675</td>
<td>33.242192</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650999</td>
<td>1.458966</td>
</tr>
<tr>
<td>700</td>
<td>34.463159</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.651061</td>
<td>1.457966</td>
</tr>
<tr>
<td>725</td>
<td>35.696096</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.651000</td>
<td>1.456534</td>
</tr>
<tr>
<td>750</td>
<td>36.889265</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650768</td>
<td>1.456302</td>
</tr>
<tr>
<td>775</td>
<td>38.135262</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650790</td>
<td>1.456854</td>
</tr>
<tr>
<td>800</td>
<td>39.425860</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650941</td>
<td>1.456730</td>
</tr>
<tr>
<td>825</td>
<td>40.673967</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.651068</td>
<td>1.458916</td>
</tr>
<tr>
<td>850</td>
<td>41.908425</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650863</td>
<td>1.458667</td>
</tr>
<tr>
<td>875</td>
<td>43.143179</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650760</td>
<td>1.457046</td>
</tr>
<tr>
<td>900</td>
<td>44.379394</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650671</td>
<td>1.457002</td>
</tr>
<tr>
<td>925</td>
<td>45.611772</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650647</td>
<td>1.456884</td>
</tr>
<tr>
<td>950</td>
<td>46.849196</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650476</td>
<td>1.456250</td>
</tr>
<tr>
<td>975</td>
<td>48.090197</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650427</td>
<td>1.455723</td>
</tr>
<tr>
<td>1000</td>
<td>49.330637</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650386</td>
<td>1.456912</td>
</tr>
<tr>
<td>1025</td>
<td>50.564925</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650351</td>
<td>1.456776</td>
</tr>
<tr>
<td>1050</td>
<td>51.800937</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650380</td>
<td>1.456156</td>
</tr>
<tr>
<td>1075</td>
<td>53.040241</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650447</td>
<td>1.456965</td>
</tr>
<tr>
<td>1100</td>
<td>54.277947</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650443</td>
<td>1.458007</td>
</tr>
<tr>
<td>1125</td>
<td>55.518080</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650489</td>
<td>1.457774</td>
</tr>
<tr>
<td>1150</td>
<td>56.745447</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650404</td>
<td>1.457202</td>
</tr>
<tr>
<td>1175</td>
<td>57.920416</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650408</td>
<td>1.457990</td>
</tr>
<tr>
<td>1200</td>
<td>59.160483</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650527</td>
<td>1.458698</td>
</tr>
<tr>
<td>1225</td>
<td>60.354893</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650495</td>
<td>1.458192</td>
</tr>
<tr>
<td>1250</td>
<td>61.590631</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650420</td>
<td>1.457749</td>
</tr>
<tr>
<td>1275</td>
<td>62.753303</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650304</td>
<td>1.457173</td>
</tr>
<tr>
<td>1300</td>
<td>63.993493</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650402</td>
<td>1.456854</td>
</tr>
<tr>
<td>1325</td>
<td>65.196012</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650539</td>
<td>1.457009</td>
</tr>
<tr>
<td>1350</td>
<td>66.428554</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650505</td>
<td>1.456863</td>
</tr>
<tr>
<td>1375</td>
<td>67.638553</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650403</td>
<td>1.456399</td>
</tr>
<tr>
<td>1400</td>
<td>68.851376</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650382</td>
<td>1.456680</td>
</tr>
<tr>
<td>1425</td>
<td>70.116222</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650317</td>
<td>1.457428</td>
</tr>
<tr>
<td>1450</td>
<td>71.343629</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650389</td>
<td>1.456863</td>
</tr>
<tr>
<td>1475</td>
<td>72.588915</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650403</td>
<td>1.456399</td>
</tr>
<tr>
<td>1500</td>
<td>73.835285</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650382</td>
<td>1.456680</td>
</tr>
<tr>
<td>1525</td>
<td>75.070740</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650389</td>
<td>1.456863</td>
</tr>
<tr>
<td>1550</td>
<td>76.319315</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650382</td>
<td>1.456680</td>
</tr>
<tr>
<td>1575</td>
<td>77.555052</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650382</td>
<td>1.456680</td>
</tr>
<tr>
<td>1600</td>
<td>78.802081</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650423</td>
<td>1.457679</td>
</tr>
<tr>
<td>1625</td>
<td>80.030131</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650415</td>
<td>1.457603</td>
</tr>
<tr>
<td>1650</td>
<td>81.254296</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650407</td>
<td>1.457697</td>
</tr>
<tr>
<td>1675</td>
<td>82.485882</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650378</td>
<td>1.457572</td>
</tr>
<tr>
<td>1700</td>
<td>83.720258</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650300</td>
<td>1.457837</td>
</tr>
<tr>
<td>1725</td>
<td>84.903269</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650328</td>
<td>1.457584</td>
</tr>
<tr>
<td>1750</td>
<td>86.135472</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650335</td>
<td>1.457258</td>
</tr>
<tr>
<td>1775</td>
<td>87.371512</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650275</td>
<td>1.457091</td>
</tr>
<tr>
<td>1800</td>
<td>88.600510</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650371</td>
<td>1.457680</td>
</tr>
<tr>
<td>1825</td>
<td>89.801200</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650384</td>
<td>1.457521</td>
</tr>
<tr>
<td>1850</td>
<td>90.962386</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650313</td>
<td>1.457055</td>
</tr>
<tr>
<td>1875</td>
<td>92.288540</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650385</td>
<td>1.457145</td>
</tr>
<tr>
<td>1900</td>
<td>93.583934</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650352</td>
<td>1.457971</td>
</tr>
<tr>
<td>1925</td>
<td>94.885882</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650370</td>
<td>1.458339</td>
</tr>
<tr>
<td>1950</td>
<td>96.193078</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650385</td>
<td>1.458882</td>
</tr>
<tr>
<td>1975</td>
<td>97.466232</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650321</td>
<td>1.458941</td>
</tr>
<tr>
<td>2000</td>
<td>98.707760</td>
<td>0.985124</td>
<td>0.562500</td>
<td>0.650397</td>
<td>1.458878</td>
</tr>
</tbody>
</table>
Accuracy : \m0.561403508772

Confusion Matrix:

+--------------+-----------------+-------+
<table>
<thead>
<tr>
<th>target_label</th>
<th>predicted_label</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>F</td>
<td>4</td>
</tr>
<tr>
<td>R</td>
<td>LF</td>
<td>1</td>
</tr>
<tr>
<td>LF</td>
<td>AU</td>
<td>1</td>
</tr>
<tr>
<td>R</td>
<td>NR</td>
<td>3</td>
</tr>
<tr>
<td>RF</td>
<td>RF</td>
<td>8</td>
</tr>
<tr>
<td>AU</td>
<td>RF</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>AU</td>
<td>1</td>
</tr>
<tr>
<td>NR</td>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>LF</td>
<td>5</td>
</tr>
<tr>
<td>R</td>
<td>AU</td>
<td>3</td>
</tr>
</tbody>
</table>

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

[41 rows x 3 columns]
### D.3  Naïve Bayes

wrong: 35  
right: 43  
accuracy: 0.551282

45 degree Weighted Con Matrix:

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>SL</th>
<th>S</th>
<th>SR</th>
<th>R</th>
<th>AU</th>
<th>NR</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>9</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

this weighted Amount is 1.5 0 0 0 0 (from the center diagonal)

conMat weighted Amt: 52.5  
count: 78  
new Accuracy: 0.673077

90 degree Weighted Con Matrix:

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>SL</th>
<th>S</th>
<th>SR</th>
<th>R</th>
<th>AU</th>
<th>NR</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>9</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

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