Automated Feature Detection of Aerial Imagery

Christian Taruc, Michael Chiang, Patrick Vidican, Zhenhui Jiang

Abstract

World Bank is looking for a team to help identify, classify, and quantify crops and infrastructure in the South Pacific. Our data comes from several high resolution UAV photographs of the Kingdom of Tonga. For supervision in classification, we obtain class labels in from shapefiles provided by World Bank. We achieved ~87% accuracy for classification using Convolutional Neural Networks, ~73% accuracy with Linear Support Vector Machines, ~68% with Logistic Regression.

Introduction

Due to their geographic location and topology, island nations like Tonga are often struck by devastating natural disasters. Cyclones and floods during the rainy season or droughts during the sunny season can cripple the local food supply. Having an automated way to assess the damage can help aid groups such as the World Bank be more effective in dispatching first responders to the area.

These trees and their locations can then be compared before and after major disasters to better understand just how much local agriculture and, in response, food security has been affected. This can directly inform and accelerate subsequent relief efforts. Our data is aerial imagery collected in October 2017. These UAV photos comprise of four Areas of Interest (AOIs) in the Kingdom of Tonga. There were 13475 trees, 10315 coconut, 2729 banana, 261 mango, and 89 papaya. Based on the biased sample numbers, we down-sampled the coconut trees and performed binary classification (coconut vs non-coconut).

Data and Preprocessing

The World Bank provided teams with georeferenced aerial imagery of the AOIs in addition to class labels from training data in the form of shapefiles. To actually help solve the World Bank problem, we needed to run our model on each image they gave and return a count of each tree type and their respective locations. Given the constraints of the course, we were able to come up with a sort of minimum viable product as a step in the right direction towards solving the greater challenge at hand. To build models that can effectively recognize coconut trees among various other kinds of tropical vegetation, we needed a large set of tree pictures from the air. A quick Google image search reveals that getting such pictures from a source outside our own data was infeasible. Additionally, the data was in a less-than-ideal format, making even the simplification of the problem harder than expected.

Tree types did not have a consistent naming scheme in the shapefile provided to us. Some trees (of the same type) would be labeled by both their genus and species, just one, or none at all, each with arbitrarily different spellings and punctuation. The solution was to comb through each unique identifying string in the relevant field and find commonality in them that could be used to form an all-encompassing query on the vector layer.

In the shapefile, each tree's location was only identified by a single coordinate pair, as opposed to a bounding box that would give us a basis for clipping. The library we used for clipping the aerial photograph (GDAL) required we give it four coordinate pairs corresponding to each corner of the bounding square that we want to clip. Therefore, to get at least 4 meters of clearance on each tree (accounting for varying sizes and growth angles) we had to pick an offset that generalized well to each tree in the given image. The offset had to be in degrees from the

prime meridian and not in meters as we had hoped. The solution was to manually create bounding boxes in ArcMap as examples. The offsets values in ArcMap were used in the feature extraction (tree-clipping.ipynb).

The thought process for the right way to do the feature extraction within the scope of this course was as follows: For the given aerial photograph, iterate through each feature referenced by its corresponding shapefile. If that feature is a tree, save a ~100x100 pixel clipping of the given aerial photograph at that feature's coordinates. This way, we are able to build up a collection of tree images in the correct context (lighting, resolution, point of view) on which we can perform a simple binary classification.

Methods and Results

Logistic Regression

Logistic Regression is a binary classification model that categorizes an object into 0 and 1 which in this case represent Not-Coconut and Coconut. The model graphs the points and draws a S-shaped curve that best fits the graphed data points. Logistic Regression was chosen to be the base model.

The model split the trees into the classes: Coconut and Not-Coconut. It took too long to run because the data set had over 6000 images and so a sample was taken of the original data set to have 2400 images. They were split in the following ways: 261 Mango trees, 850 Banana trees, 89 Papaya trees and 1200 Coconut trees. Half the dataset was coconut and the other half were other trees. The images were stored into a list as arrays of RGB values representing each pixel. Then they were converted to grayscale. This reduced processing time because all the images are green colored trees, the color is not as important in classifying the images. The array

was converted to a Spark RDD and fed into the model. The accuracy rate was 68%, not satisfying our goal of having an over 80% accuracy. This was expected as the image data of pixels are probably scattered with 4 primary groups forming and do not form a S shaped curve.

Linear Support Vector Machines (Linear SVM)

Linear SVM is a supervised learning model that effectively performs binary classification by finding the "maximum margin hyperplane". To obtain the hyperplane, the model takes the pixel arrays taken from each image and graphs it. The data points form two groups based on similar features. The points in Class One that are closest to Class Two and the points in Class Two that are closest to Class One are the Support Vector Machines. These SVMs form a hyperplane that covers the maximum distance between the two classes.

The data was taken from the RDD created for the Logistic Regression model. The accuracy was 73%, which did not satisfy the goal of over 80%.

Convolutional Neural Networks

The building block of the CNN is the convolutional layer. This layer runs a sliding window through the image and at each step convolves (a dot product) between the image and several filters. This step produces a new volume where we have increased the depth.

A pooling layer downsamples the volume along the spatial dimensions by a certain factor. This reduces the size of the representation and the number of parameters which makes the model more efficient and prevents overfitting. The most common type of pooling layer is a max-pooling layer, which takes the max from each downsampled block.

Our model stacks up Convolutional Layers, followed by Max Pooling layers. We also include Dropout to avoid overfitting. Dropout is a regularization technique that randomly sets a subset of

the hidden layers to zero. This forces the model to learn new features as the dropped subset cannot influence the retained features. Finally, we add a fully connected layer followed by a sigmoid activation function. This gives us our prediction probabilities and a final label.
Given below is the model structure:

```
model = Sequential()
model.add(Conv2D(32, 3, 3, activation='relu', input_shape=(101,100,3)))
model.add(MaxPooling2D(pool_size = (2,2)))

model.add(Conv2D(32, 3, 3, activation='relu', input_shape=(101,100,3)))
model.add(MaxPooling2D(pool_size = (2,2)))

model.add(Conv2D(64, 3, 3, activation='relu', input_shape=(101,100,3)))
model.add(MaxPooling2D(pool_size = (2,2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))
```

In the above code, we use 3 convolutional layers and 1 fully-connected layer. Lines 2 and 4 adds convolutional layers with 32 filters / kernels with a window size of 3×3. In line 6, we add a layer with 64 filters. In lines 3,5, and 7 we add max pooling layers with window size 2×2. In line 9, we add a dropout layer with a dropout ratio of 0.5. In the final lines, we add the sigmoid function which spits out the final probability prediction.

For training we fit the model on 80% of the data. As a binary classifier, we use binary cross entropy loss and the stochastic gradient descent optimizer to train the network. We run it over 10 epochs. Finally we test the model on the final 20% of the data.

Accuracy steadily increased between epochs. There is a large jump from 40% accuracy all the way to over double at 80% accuracy. This might be due to overfitting, however we

believe we accounted for overfitting through regularization methods. Overall, this is our champion model and the one that satisfies World Bank's metrics of 80%<.

Challenges

Data preprocessing was very difficult. We were given only aerial imagery and a shapefile. Locating trees, extracting features and transforming features from the picture to arrays that we can use for training, were all tasks we have not done before. Thus, we not only had to understand Keras deep learning models but also the GIS framework for full feature extraction. Since we had a huge amount of features, all the models we trained took a very long time. Every picture we extracted had 101 * 100 pixels with three RGB channels. With over 6000 trees, we had to wait hours to fit the model. Restricted to the lowest tier CPU on Domino, we were not able to utilize the full data set for the Logistic Regression and Linear SVM models.

Object detection was attempted, but there were large, overarching problems that created limitations in creating such model. We wanted to use the 'YOLO' framework, for our problem because the model is very fast and allows for contextual information to aid in avoiding false positives because of the way that the model is structured. However, environment and time constraints did not allow us to create this model.

Conclusions

In this project, we found location of each tree and preprocessed each tree by extracting features and transforming features from the picture to arrays. Due to the biased sample size, we down-sampled coconut trees and trained about 6000 trees including about 3000 coconut trees. We fitted CNN model which had more 87% accuracy and linear SVM with about 73% accuracy. We would like to try one or two more models such as random forest to compare the models. At

last, we will try to fit a model that can quantify the trees (likely through the YOLO method) and actually complete the World Bank Challenge.