Detecting Coconut Trees from Aerial Photographs for Disaster Relief

Amber Thomas, Tynan Challenor
Department of Computer Science
Stanford University
`alt3@stanford.edu, tynan@stanford.edu`

Abstract

Natural disasters in the tropics can threaten food security [1]. As a result, the World Bank issued a challenge to the global AI community to develop computer vision algorithms to count and locate standing food trees, like coconut trees, from aerial photographs [2]. With an RCNN using Resnet101 base classifier pre-trained on the kitti object detection dataset and tuned using an in situ aerial photograph taken over the Island of Tonga, we trained a detection model that returned 94% AP upon the dev set using the PASCAL VOC evaluation metric [15], [16]. However the test set–from the same distribution as far as we could tell–could not get above 71.32% AP.

1 Introduction

Tropical Storm Gita hit the islands of Tonga on February 11^{th} and 12^{th} of this year, providing another reminder that innovative disaster relief is a necessity [1]. Food trees, such as banana, coconut, and papaya are blown over in such storms, leading to food shortages [2]. As a result, the World Bank, WeRobotics, and OpenAerialMap have challenged the global AI community to develop computer vision algorithms for reading aerial photographs taken after natural disasters in tropical areas. The goal is to identify and return the location of standing food trees, like coconut trees, captured in aerial photographs to assess relative food security.

We trained a faster R-CNN with resnet101 base classifier on a single aerial photograph taken over the Islands of Tonga to detect coconut trees. The image was 25,006 by 17,761 pixels with a spatial resolution of 9cm and covered a region approximately 50km^2. The input for training consisted of 300x300 subdivisions of the original image with ground-truth coconut trees identified. When tested on new images, the model generated bounding boxes around the center of predicted coconut trees as well as its relative confidence in the predictions. We assessed the model's performance with the PASCAL VOC average precision metric [4].

2 Related work

Some object detection algorithms use only one step to examine an image, predict bounding boxes and classify objects. The Single Shot Detection (SSD) algorithm uses a slightly abbreviated convolutional neural network in which the final classification layers are replaced with more convolutional layers. These layers produce multiple feature maps of different sizes [3]. The algorithm predicts the offset and the class probabilities for each bounding box, discarding those boxes with an IoU (intersection over union) score less than 0.5 [3]. The SSD was an attempt to improve upon detection accuracy while gaining the speed offered by integrating the region propsal and classification step [3]. Similarly, the *You Only Look Once* (YOLO) algorithm also detects and classifies objects with a single pass [5]. It divides an image into an S x S grid and outputs multiple bounding boxes per cell. Accompanying each box is the relative confidence that the model has detected an object; a grid cell is only responsible for detection if the object's center lies inside [5].

A second approach to object detection and classification is to break the algorithm into two components: region proposal and then detection and classification [6]. While such networks often take longer to train, this approach has proven the most reliable at detecting objects [7]. The R-CNN (R stands for regions) architecture utilizes a region-proposal network (RPN) to direct the detection network's focus [6]. The original R-CNN identified 2000 possible regions, passed each one through a CNN for feature extraction, and finally classified the outputs with a support vector machine (SVM) [7],[8]. It also used linear regression to assess the proposed bounding boxes [7],[8]. Iterating upon the R-CNN was the Fast R-CNN [9]. This model ran the image through a CNN prior to identifying regions of interest (ROI's) and replaced the SVM with a softmax classifier to accommodate multiple possible classes [9]. The final version to date is the Faster R-CNN. While not as speedy as an SSD, the Faster R-CNN network improved upon its predecessors by sharing convolutional layers between the RPN and the object detection network [7].

Computer vision has proven a boon to problem-solving in disciplines from medicine to sustainability and transfer learning has proven fundamental to training very deep networks without a lot of data. Jean et al. 2016 trained a model to predict poverty levels using nighttime light expenditure captured from satellite images taken over Nigeria, Tanzania, Uganda, Malawi, and Rwanda [10]. Pre-training on the ImageNet dataset and then learning a mapping from day-time satelite images to night-time light output provided the group a model that could be tuned with a relative paucity of ground-truth survey data [10].

3 Dataset and Features

Our training data was provided by WeRobotics and OpenAerialMap as a 25,006 by 17,761 pixel aerial photograph with spatial resolution of 9cm [11], [12]. Accompanying the photograph were geo-referenced points specifying coconut tree locations. We padded and divided the images into 300 by 300 segments for a total of 5040 smaller pictures and 11,300 labeled coconut trees. After random shuffling, we extracted 74% (3,832) training images and 12% (604 images) for each of the dev and test sets. We augmented the data by horizontally filipng half the training images (chosen randomly) and adding them to the data set. During the first round of testing and analysis we were having difficulty getting our detection loss below our benchmark. After goint through our dev set to get a sense of where errors were coming from we noticed a high incidence of unlabeled trees in the dataset provided by WeRobotics and OpenAerialMap as well as a number of mistakenly labeled banana trees. After a round of data cleanup we correctly labeled more than a thousand additional coconut trees.

4 Methods

We trained a Faster R-CNN network using a resnet101 base classifier. The Faster R-CNN has two components: the region proposal network (RPN) and an object detection and classification network [7]. In the RPN, images pass through a CNN generating a feature map. A small CNN slides over this feature map, shrinking the dimensionality of the output [7]. Each subsequent feature then passes through two fully-connected layers: one for classification and another for regression upon the proposed bounding boxes [7]. Bounding boxes are generated for k different proposals per location of the sliding CNN [7]. Proposed anchor boxes are given a positive or negative label predicting whether they enclose an object. To assign a a positive label the box must either have the highest IoU of the k boxes in that region or have an IoU > 0.7; there can thus be multiple positively labeled anchor boxes for a single location [7].

The loss function is as follows:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cla}}\sum_i L_{cla}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*))$$

With i being the index of a certain anchor box, p_i the probability that i's anchor box bounds an object, p_i^* the binary label (1 for positive, 0 for negative), the classification loss (L_{clas}) is binary log loss:

$$-(p_i^* \log(p_i) + (i - p_i^*) \log(1 - p_i^*))$$

The predicted bounding box, t_i, is a vector containing the box's four parameterized coordinates and t_i^* is the vector for the ground-truth. The loss for the bounding boxes is only considered when $p_i^* = 1$, therefore bounding boxes with no predicted object do not contribute. The loss for bounding box prediction is a smooth L_1 loss defined below [7], [9].

If $| \; x \; |< 1$:

$$smooth_{L1}(x) = 0.5x^2$$

Otherwise:

$$smooth_{L1}(x) = | \; x \; | - 0.5$$

The base classifier for the model we used was a deep residual network, resnet101. Deep networks have suffered from vanishing and exploding gradient problems, whereby information from the loss function is multiplied by many numbers less than one or far greater than one, preventing meaningful gradients from back-propagating to earlier layers [13]. Vanishing and exploding gradients can be addressed by normalizing the input or adding normalizing layers to the network; harder to target is accuracy degradation. As networks deepen, accuracy can saturate and even decrease [13]. Theoretically an optimally trained deep network should perform no worse than its shallower counterpart because deep layers can learn an identity mapping of the earlier ones; however, this has proved hard to do in practice [13]. In a residual network, deep layers are tasked with fitting a residual mapping. For example, if the underlying mapping is $H(x)$ we redefine it as $F(x) = H(x) - x$ and solve for $H(x) = F(x) + x$. This is accomplished via *shortcut connections* whereby the output of a previous layer both enters the subsequent one and skips to a layer further down the network:

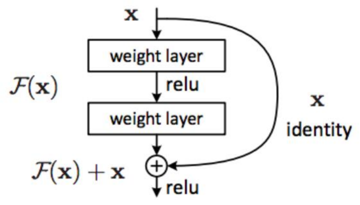

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Diagram from He et al.: Deep Residual Learning for Image Recognition.

Deep residual networks leverage the accuracy that comes from learning a more complicated function without sacrificing the training loss of deep networks [13].

To increase training efficiency we used transfer learning with the Faster R-CNN. The network was pre-trained using the kitti dataset from the Karlsruhe Institute of Technology [14]. The dataset was collected with high resolution gray-scale and color video footage from cameras mounted on a Volkswagen station wagon driving along rural roads and on highways [14]. The dataset provides a candid, messier representation of objects, which works well for our task of detecting coconut trees from images taken after natural disasters.

5 Experiments/Results/Discussion

We trained both an SSD model and a Faster R-CNN using a pre-train model and tuned the weights to our data set. Initially we tried an SSD model with an InceptionV2 base network. We had hoped to maintain some of the speed of YOLOv2, but reap the benefits of higher accuracies reported by the SSD. After tuning we were unable to get the mAP above 78.12%. We suspected that the releatvie size of our trees and the way that they blended into the background was making it difficult for the Single Shot Detector to locate them. Benchmark accuracies from Liu et al. 2016 led us towards

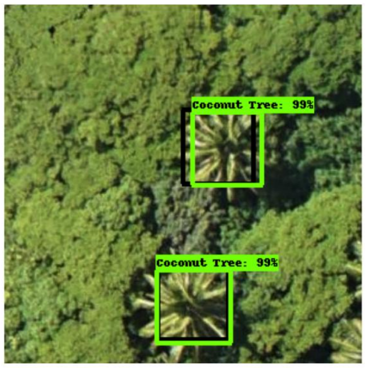
Faster R-CNN, which has superior performance when compared to SSD for small to medium objects. In addition, we used several concepts introduced in class when setting up hyperparameters:

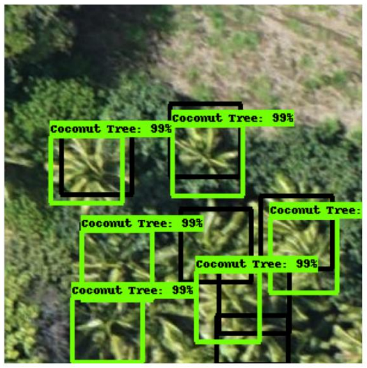
Model Attribute	Specification
Initialization	Xavier
IOU Threshold	0.6
Optimizer	Momentum
Starting α	0.0001

5.1 Results

α	Min loss in 7k iterations	Test mAP
0.00005	0.053	43.49%
0.0001	2.42 x 10^{-4}	71.32%
0.0002	$1.30x10^{-4}$	69.10%

At 4.8K iterations of training on the model initialized with $\alpha = 0.0001$, we returned a dev set mAP of 94.38%

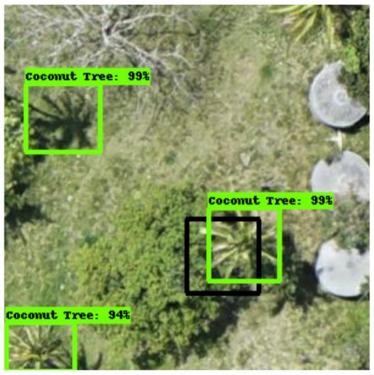


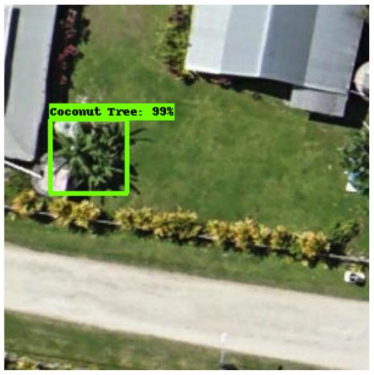


Images 1 and 2: On the left: two well-bounded coconut trees captured in the canopy. On the right: Coherently captured trees in a mass. Green boxes are predictions, black boxes are ground-truth. Some of the ground-truth boxes are mislabeled spaces between trees, such as the middle black box in the right-hand image.

5.2 Error Analysis

The model stumbled most frequently in mis-categorizing banana trees for coconut trees. Less often it was also confused by a shadow cast by a coconut tree. Error analysis in earlier iterations of the project also led to finding how poorly the data was labeled in certain sectors, which informed us of the need to clean the data set.



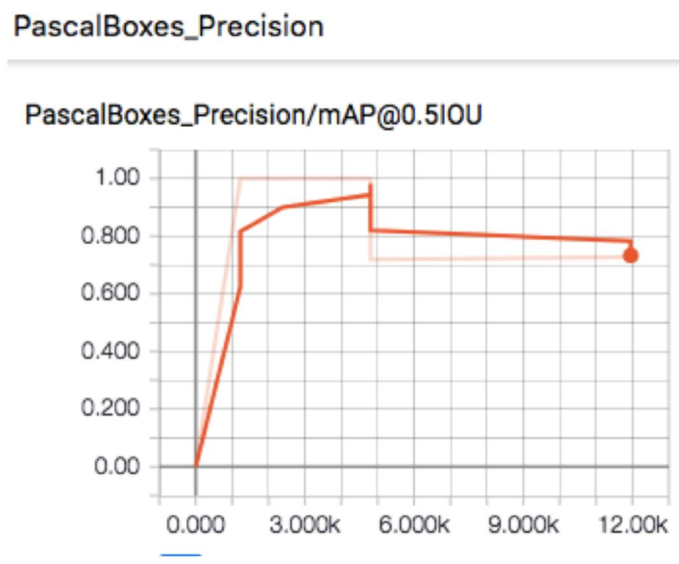


Images 3 and 4 On the left: An incorrectly classified coconut tree shadow. On the right: Banana trees that the classifier believes are coconut trees.

5.3 Discussion

For the majority of our model-building the larger Faster R-CNN network seemed more reliable at learning to detect and classify coconut trees. We were suprised at how well our model performed. During benchmark testing using logistic regression we had an accuracy of 74.5%. Even though we were splitting the image into small crops to make the problem purely one of classification and not detection, it was still far higher than expected. Additionally since our data set came from the same photograph split into pieces, it has the same lighting features, air quality, background types, etc between the train, dev and test sets. On top of this, object detection models are shown to have a higher performance when working with fewer categories, so only working with coconut trees likely helped boost our score.

However, our test set mAP was far lower than the dev set mAP we'd been seeing. We tried to recreate this score, but failed to. Below is the dev image that led us to believe that we were reaching 94% mAP:



Between 2000 and 4,800 iterations the model's mAP score climbed from 81% to 94% before declining sharply. We tried to replicate this score by retraining from scratch, but failed to achieve the same mAP results on the dev set, leading us to believe that we'd altered the code somewhere along the way.

6 Conclusion/Future Work

Performance of the model could be enhanced with some changes: the first is negatively weighting banana trees; the model may be learning the coconut tree's fronds and is therefore tricked by the banana tree's similar structure. We might also try to visualize the encodings of layers close to the output classification layer to see which patterns are activated in deep neurons. Further data cleaning may also be helpful. Even after extensive time spent cleaning there are still mislabeled and unlabeled trees in the dataset that are inevitably preventing the model from learning to its full capacity.

But given our low test set results and the un-reproducibility of our dev set milestone we cannot yet conclude whether Faster R-CNN is an effective model for our task.

7 Contributions

Amber handled much of the data cleaning and labeling, creating scripts to not only break up the aerial photograph, but also to add bounding boxes to the data for training.

Tynan set up the GPU's as well as got the model off the ground. Amber and Tynan worked together once the model was up with the design choices of training. Tynan also took the lead in crafting the poster and the writeup with support from Amber.

Both Amber and Tynan read extensively.

References

[1] A. Fritz, “Tropical Cyclone Gita is a monster Category 4, and it’s hammering Tonga,” Washington Post, 12-Feb-2018.

[2] Using AI For Good: A New Data Challenge To Use AI To Triage Natural Disaster Aerial Imagery,” Forbes. [Online]. Available: https://www.forbes.com/sites/kalevleetaru/2018/01/20/using-ai-for-good-a-new-data-challenge-to-use-ai-to-triage-natural-disaster-aerial-imagery/. [Accessed: 23-Feb-2018]

[3] W. Liu et al., “SSD: Single Shot MultiBox Detector,” arXiv:1512.02325 [cs], vol. 9905, pp. 21–37, 2016.

[4] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, The PASCAL Visual Object Classes (VOC) Challenge.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[7] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” arXiv:1506.01497 [cs], Jun. 2015.

[8] J. Xu, “Deep Learning for Object Detection: A Comprehensive Review,” Towards Data Science, 11-Sep-2017. [Online]. Available: https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9. [Accessed: 20-Mar-2018].

[9] R. Girshick, “Fast R-CNN,” arXiv:1504.08083 [cs], Apr. 2015.

[10] N. Jean, M. Burke, M. Xie, W. M. Davis, D. B. Lobell, and S. Ermon, “Combining satellite imagery and machine learning to predict poverty,” Science, vol. 353, no. 6301, pp. 790–794, Aug. 2016.

[11] https://werobotics.org/blog/2018/01/10/open-ai-challenge/.

[12] “OpenAerialMap,” OpenAerialMap Browser. [Online]. Available: https://browser.openaerialmap.org/. [Accessed: 22-Mar-2018].

[13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” arXiv:1512.03385 [cs], Dec. 2015.

[14] “The KITTI Vision Benchmark Suite.” [Online]. Available: http://www.cvlibs.net/datasets/kitti/. [Accessed: 23-Mar-2018].

[15] "Speed/accuracy trade-offs for modern convolutional object detectors.” Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Gudarrama S, Murphy K, CVPR 2017

[16] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, 2012, pp. 3354–3361.