Hawk: Object Detection in Aerial Imagery

Mayank Vachher (mvachher@sfu.ca) and Anna Mkrtchyan (amkrtchy@sfu.ca)

1 Introduction

Disasters in the south pacific are an unfortunate reality, and their consequences can be devastating for the local population, who heavily rely on the local agriculture as their primary food source. When disasters strike, it is essential to assess the damage done as soon as possible and secure undamaged food sources such as coconut, banana and other trees. Another important task is to evaluate road conditions so the aid can be distributed in the most efficient way. Currently, aerial imagery taken post-disaster for the damage assessment is analyzed manually, which can take up to days.

WeRobotics partners with the OpenAerialMap and the World Banks UAVs for Disaster Resilience Program in the attempt to significantly accelerate the analysis of aerial imagery before and after major humanitarian disasters. Their Open AI Challenge: Aerial Imagery of South Pacific Islands [1] is a competition for participating teams to develop machine learning classifiers to automate the analysis of the aerial imagery. As a part of the competition, high accuracy classifiers ($> 80\%$) are required for detection of coconut, banana, mango and papaya trees as well as road type and conditions. As training data, we are given a single high resolution (about 8cm/pixel) aerial image that covers a $50km^2$ region of the Kingdom of Tonga, the snapshot taken in October 2017. Along with the image we are given shapefiles, that identify geometric locations and classes of the objects of interests (i.e. coconut, banana tree etc.). The data is manually labeled by the group of volunteers from Humanitarian OpenStreetMap Team. Details of the competition can be found in [2].

Aerial imagery detection has numerous applications and has been extensively studied before [3–5]. These works use convolutional neural networks of various architectures (discussed in detail in Section 2) to detect objects on aerial imagery. However, the majority of the work is done on vehicle and road detection. In this project, we apply similar approaches to develop a framework which locates and detects coconut and banana trees. This is by no means a trivial task. Challenges include 1) difficulty in differentiating generic shapes that different objects in the aerial images might have, 2) small objects residing in densely crowded regions in the aerial images, and finally 3) the absence of well-annotated datasets of objects in aerial scenes.

The report is structured as follows. We introduce our methodology and data pipeline in Section 2, describe out results in Section 3, and discuss our findings and future work in Section 4.

2 Methodology

2.1 Literature Review and Model Selection

Before we can proceed to build our data processing pipeline we need to choose the approach. In this section, we give a brief overview of the popular state-of-the-art models for object detection and justify our model selection. The appropriate choice of the model is critical, and careful consideration of pros and cons of all the approaches is necessary to achieve the best performance. We base our review on [6].

There are several approaches developed to tackle object detection problem. Region-based convolutional networks (R-CNN) were among the first proposed models [7]. First, selective search method was applied to detect region proposals from the original image. Region proposals formed the feature vector that was fed to 1) multiple classifiers to produce probability for each class and to 2) regressor to refine bounding boxes of regions of a proposal. Fast R-CNN [8], and Faster R-CNN [9] speed up the detection process by first applying deep CNNs to the image and extracting feature map and replacing selective search by Region Proposal Network (RPN) to generate region proposals, predict bounding boxes and classes of the objects. The extension of Faster R-CNN, Mask R-CNN [10] adds a parallel branch to object detection to predict object masks with very little overhead. Mask R-CNN outperformed state-of-the-art models in the 2017 COCO challenge in segmentation, bounding-box detection, and object detection.

The models above approach object detection as a classification problem and build a pipeline where regions with objects in them are proposed and then those regions are sent to classification and regression heads. Another set of models attempt to predict bounding boxes and class probabilities in a single network in a single evaluation. In You Only Look Once (YOLO) [11] model the input image is subdivided into a grid with each grid predicting given number of bounding boxes. It is passed through a CNN once and the output layer contains predictions for each grid with corresponding class probabilities. Finally, Single Shot Detector (SSD) [12] have a similar architecture as YOLO but uses extra feature layers from different feature maps to refine the bounding box prediction.

YOLO and SSD are faster to train and predict but typically have lower accuracy. This trait makes them suitable for real-time detection tasks. However, for this challenge, we are concerned more about the accuracy than speed. Thus, we choose Mask R-CNN as our model. The choice was made based on two considerations 1) The model outperformed state-of-the-art models in the most recent COCO challenge for object detection and 2) the implementation of the model on Tensorflow with pre-trained weights which was easy to understand and modify to fit our needs was available on the GitHub [13].

2.2 Data Science Pipeline

After choosing the appropriate approach as discussed in the previous section, we proceed to build out data pipeline. It is summarized in Figure 1.

Briefly, we first convert the training data we are given for the challenge to a set of images suitable for training and testing the model. Since we start with pre-trained weights, there is no need to train the entire neural network. We train the last layers (classification/bounding boxes heads) and choose the configuration with the lowest validation error. Finally, we evaluate the performance of the model on the previously unseen test dataset. In the subsection below we discuss each of these stages in more detail

2.2.1 Data Processing

As we described in the Introduction section, we are given a single high-resolution image along with a shapefile - a GIS compatible file that stores geometric locations and attributes of geometric features, such as points, lines, and polygons. Our goal is to combine and convert these two data sources into training, validation, and test sets suitable for our object detection model. GDAL is an open source library to handle raster and vector geospatial data formats. Fiona is an extremely popular and sought after python library for reading and writing geospatial data files. We utilize

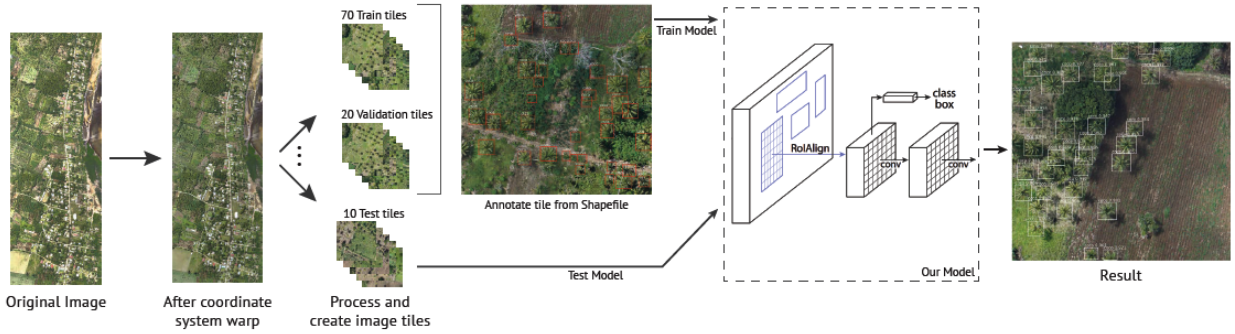

Figure 1: Data Science Pipeline. (*From left to right*) We adjust geodata of the input image to match the coordinate system used in shapefiles. Original image is divided into tiles that serve as training and testing sets. Images are annotated based on the labels provided in shapefiles, which are then fed to the pre-trained model. Last layers of the model are re-trained on our own data after which model is ready for predictions.

the power of Fiona to read the shapefile into a JSON format where we are inherently provided with a variety of geographical data and is readily available for data processing. Then we extract the locations of the objects of interests, such as coconut trees by checking the tags provided for each GeoJSON object in the shapefile. The locations are given in latitudinal and longitudinal coordinate system. In order to map these locations onto the high-resolution image, we convert the original image into the Latitude-Longitude coordinate system with the help of GDAL tools. We further map the latitude and longitude coordinates extracted from the given shapefile into the image pixels by utilising the geographical metadata in the aerial image. By the end of this procedure, we have an image along with the pixels where trees of interests (coconuts) are located. There are roughly 10314 coconut trees in the original image.

The R-CNN neural network takes as an input annotated images of about 1000x1000 pixels. We subdivide the original image into tiles of the size 1000x1000. We take 100 such tiles and manually annotate each tile by locating the trees and drawing rectangles around them. It is a tedious process but we believe properly annotated dataset is key for training a model with high prediction accuracy. We use VGG Image Annotator to quickly annotate 100 images. Each annotation has a JSON format and simply stores the positions of all rectangles as well as their tags. We have roughly between 60 to 70 objects in each of the tiles, so we conclude that the training data we have is substantial. An example of an image annotated with VGG Image Annotator is shown in Figure 2(A).

It is important to note, that during the annotation we notice a lot of discrepancies, such as trees that were mislabeled (coconut trees labels as bananas, shadows labeled as trees), and some trees were not labeled at all. While we could correct these erroneous data, we did not change anything from the original dataset given to us such that we dont violate the conditions of the OpenAI challenge.

2.2.2 Deep Learning Architecture and Training

As we discussed in the Model Selection section, the core of our model is based on the Faster R-CNN implementation. The backbone CNN that extracts feature maps from the original image is ResNet101, a Deep Residual Neural Network with 101 layers [14]. Residual networks allow us to

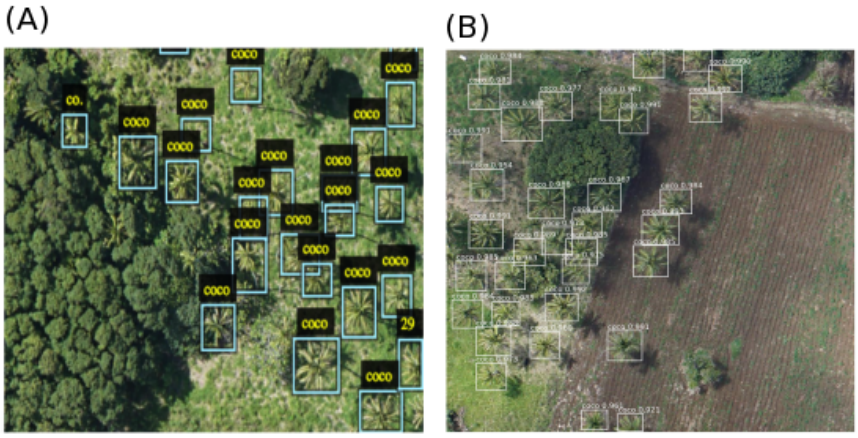

Figure 2: From training to prediction. (A) Training data after annotation with VGG annotator. (B) Bounding box predictions on a test file. All prediction confidences are above 90% for coconut trees. Trees that are not coconuts are not detected.

efficiently train very deep neural networks by introducing skip connections, where weights from earlier layers are copied into a deeper layer. It takes an image of 1000x1000x3 and outputs feature map of size 32x32x2048. These features are then passed to an RPN (Region Proposal Network) for training classification/regression of object classes and generation of bounding boxes.

We started the training process by downloading a model pre-trained on COCO dataset, one of the most commonly used datasets for object detection and segmentation [13]. Since we only aim to train the last layers of the model (referred as classification and regression heads) we do not change the configuration of earlier layers but change few RPN parameters. To speed up the training process while we aim to achieve high accuracy, we set the minimum Regions Proposal Confidence to 0.8, which means that only regions with more than 80% confidence of potentially containing an object are considered. Moreover, since in aerial images trees are expected to have roughly similar sizes and aspect ratios, anchor scales are set between 8 to 128. Learning rate is set to 0.001 with the weight decay of 0.0001. We split the available data (100 annotated tiles) into training/validation/test set of 70/20/10 images and conduct several experiments by varying the number of epochs, number of steps, and number of maximum possible ROIs (Regions Of Interest). We monitor the performance with the help of TensorBoard. In Figure 3 we show the results of the bounding box regression during the training (left) and validation (right).

Based on the validation scores of several experiments that we conducted, we choose the configuration having the best performance: train for 20 epochs, with 100 steps each, with the maximum number of ROIs being 100 (this is a reasonable choice since we got at most 70 trees per image during the annotation).

3 Results

We use weights of the model trained for 20 epochs as described in the section above, to predict coconut trees on the test set consisting of 10 images. Our prediction outcome on one of the images is shown in Figure 2(B). We see that all coconut trees were detected with quite high confidence

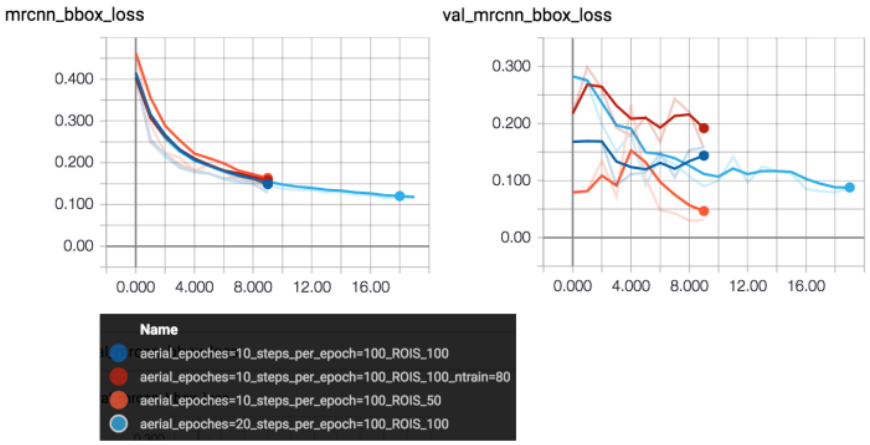

Figure 3: Training and validation monitored in TensorBoard. Based on the validation, model trained for 20 epochs with 100 steps per epoch has the best performance.

($> 90\%$). For a formal model evaluation, we choose the mean average precision (mAP) metric, as well as Precision and Recall for those 10 images. The mAP metric is a commonly used metric for object detection. It is simply a mean of average precision, where not only the number but also the order of correct predictions is evaluated. As for the challenge, we shift our focus to Precision and Recall since we are more concerned about the overall result rather than the order. Precision and recall are also metrics used in the challenge for evaluation purposes. We present all three metrics along with the F_1-measure in Table 1.

Table 1: Metrics of Model Performance

Metric	Mean Average Precision	Precision	Recall	F_1 Score
Value	85%	91.3%	97.1%	94.1%

4 Conclusion

Through this report, we present the work we have done for the Open AI Challenge: Aerial Imagery of South Pacific Islands. We developed a Faster R-CNN based model, which is trained on the provided dataset of a single high-resolution aerial image along with the shapefile that locates and tags various types of trees. At the moment, our model is trained to detect coconut trees with high accuracy ($> 91\%$ precision and $> 97\%$ recall). Our model can be easily extended to detect and classify other types of trees as well. We are currently training our model to detect and classify both coconut and banana trees. So far the detection of banana trees has been poor (0.3 confidence). This is mostly due to the fact that we have a large class imbalance, with 2000 banana trees in the original image and less than 100 banana trees in our annotated images. We are in the process of acquiring more training data to increase the accuracy of the model. In the future, we plan to classify other types of trees (mango, papaya), as well as road types and their conditions.

References

[1] https://werobotics.org/blog/2018/01/10/open-ai challenge/. Open ai challenge: Aerial imagery of south pacific islands.

[2] https://docs.google.com/document/d/16kKik2clGutKejU8uqZevNY6JALf4aVk2ELxLeRmsQ/edit. Open ai challenge: Details.

[3] Michael Ying Yang, Wentong Liao, Xinbo Li, and Bodo Rosenhahn. Vehicle detection in aerial images. *arXiv preprint arXiv:1801.07339*, 2018.

[4] Jennifer Carlet and Bernard Abayowa. Fast vehicle detection in aerial imagery. *arXiv preprint arXiv:1709.08666*, 2017.

[5] Belén Luque, Josep Ramon Morros Rubió, and Javier Ruiz Hidalgo. Spatio-temporal road detection from aerial imagery using cnns. In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Volume 4: VISAPP*, pages 493–500. SCITEPRESS, 2017.

[6] https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection c1f3d437b852. Comparison of obejct detection models.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2016.

[8] Ross Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.

[9] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[11] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[12] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[13] https://github.com/matterport/Mask_RCNN. Mask rcnn.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.