Using a Lightweight Convolutional Neural Network in Order to Detect Forest Trees from Aerial Imagery

Shai Aarons
CSC2005Z Research Report
University of Cape Town
Supervisor: Patrick Marais

Abstract— **In this research paper we attempted to use a lightweight and compact convolutional neural network in order to delineate and detect forest trees from aerial imagery. Our convolutional network is a pipelined version of MobileNets combined with YOLOV3. We used height data as well as RGB elements as input for the neural network. Training was performed with a small input of around 750 images. After successful training of the neural network we found that our mobile and lightweight convolutional neural network is able to achieve detection in cases where the trees are well seperated.**

I. Introduction

Computer vision combined with the ever-increasing trend towards machine learning applied in areas such as farming, biology or botany can lead to myriad benefits, such as increased profits or a deeper understanding of eco-systems. In South Africa companies such as Aerobotics [1] are using machine learning in order to maximize crop management efficiency by locating problematic trees instead of having farmers and workers manually search for these trees. This research evaluates how a lightweight convolutional neural network can be used for the task of delineating trees from images of canopies or forests. Detecting trees from images is a difficult task due factors such as the variability in tree sizes and types. When the trees are closely bundled or grouped near one another, this presents difficulty in finding where a tree begins or ends. The difficulty is in telling if there is only one tree with more than one branch or if there is more than one tree. To deal with the task of detecting and segmenting forests or orchards, we investigated compact convolutional neural networks in order to discover if it is model that would work best for this particular problem and the speed and performance demands that it requires. Both RGB images and height data are used as inputs to the convolutional neural network. This data is obtained from images and data acquired by flying drones over areas of regions with apricot trees. The goal is to train the neural network within a controlled and limited amount of training time. In this project we have used supervised machine learning (specifically through a convolutional neural network). Much previous work in the field of detection and segmentation from aerial imagery has used classical computer vision approaches that use no form of artificial intelligence. More specifically, we use a lightweight and mobile convolutional neural network in the form of a pipe-lined version of YOLOV3 [2] and Mobilenet [3]. The rest of the paper is organized as follows: we first discuss related works and the various attempts at solving similar problems. Then, our methodology is described in detail. The methodology is followed by our results where we discuss how successfully our implementation performed the task of detecting trees. The conclusions are then presented.

II. Related Works

A. Tree Counting and Classical Approaches

Much work has been done up to now on the process of using a Convolutional Neural Network for counting individual trees from RGB imagery: here, the number of trees in a photo taken from drone or satellite imagery are merely tallied. An example of this is from the work of Cheang et al. [4] whereby a system is proposed for using supervised learning for counting and localizing

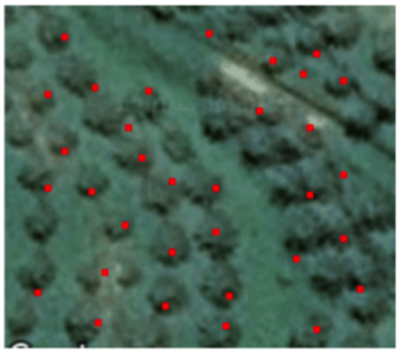


Fig. 1. Counting treetops using deep learning [4]

palm trees in high-resolution, panchromatic satellite imagery. This work differs from ours in that we attempt to detect and outline whole trees with a bounding box whereas Cheang et al. [4] only finds the tree crowns of a specific type of tree (Palm Tree) in order to count the number of trees in an image. An example of tree counting is shown in Figure 1. While much work has been done on counting trees, this project seeks to offer a segmentation feature for a specific type of tree.

Furthermore, much previous work in image segmentation and detection from aerial imagery has used more classical computer vision approaches. This can be seen in the work done by Wang et al. [5] where the established approach of using methods such edge detection, watersheds etc. have been used for individual tree-crown delineation and treetop detection in High-Spatial-Resolution aerial imagery.

B. Using Unsupervised Learning

Some previous research has asked the question as to whether supervised learning, unsupervised learning or a mixture of both should be used to tackle our task. Weinstein et al. [6] combined supervised learning with unsupervised learning in order to detect forest trees from aerial imagery. Here, a semi-supervised Convolutional Neural Network is used to detect individual tree crowns in RGB imagery. This method uses existing Light Detection and Ranging (LIDAR) models to create an unsupervised learning model, which is then pipe-lined through to a supervised neural network that uses RGB imagery as its input. This process is shown in Figure 2.

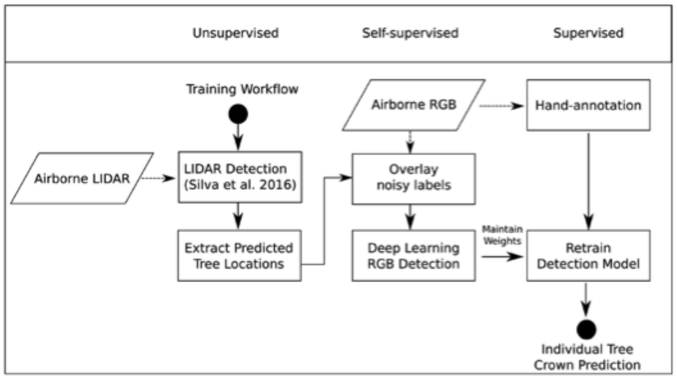


Fig. 2. Semi-Supervised Learning Process for Segmenting Trees [6]

This research project will only used supervised learning to tackle the task of isolating individual tree crowns. Our research differs also in the sense that we will use RGB layers as well as Digital Elevation Model (DEM) height data as inputs for the convolutional neural network.

C. Using a Modern Object Detection Algorithm

Both Weinstein et al. [6] and Cheang et al. [4] use a classical sliding window approach for achieving their respective tree counting projects. Furthermore, Li et al. [7] used LeNet (GoogLeNet [8]), to create a Convolutional Neural Network in order to detect and count Oil Palm Trees from High-Resolution Remote Sensing Images. LeNet [8] uses a sliding window approach for its neural network. This model uses the 'You Only Look Once' (YOLO V3 [2]) state-of-the-art, real-time object detection system. YOLO V3 does not use the classical sliding window approach.

D. Closely Bundled Trees

Our project seeks to locate and outline individual trees in areas and regions where trees are closely crowded together. As Li et al. [7] points out in their study: "Previous palm tree or tree crown detection studies have focused on detecting trees that are not very crowded and have achieved good detection results for their study areas." Closely bundled trees are problematic in the sense that it is a cumbersome task to evaluate where trees begin and end. There are instances where trees are

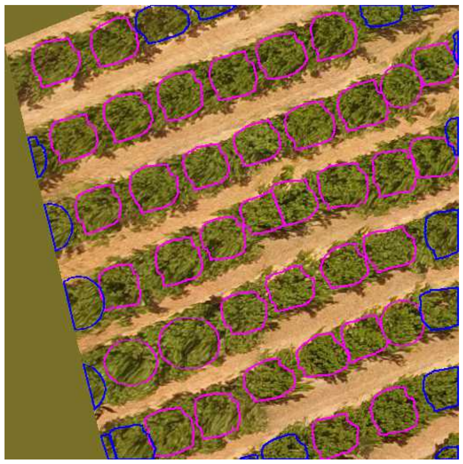


Fig. 3. Closely Bundled Trees

closely bundled such that their crowns overlap with one another - from an aerial image it is difficult to determine if these are single or multiple trees. Furthermore, a tree might have split branches so that from above it appears as if there is more than one tree crown while it is, in fact, rooted from the same trunk and thus a single tree. Our project attempts to alleviate these issues of closely bundles trees as depicted in Figure 3.

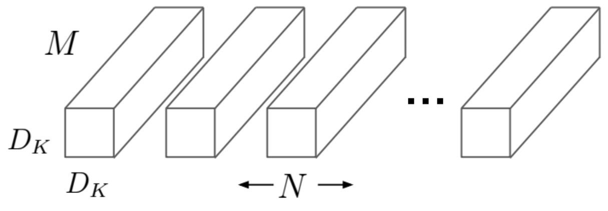
E. MobileNet

MobileNet [3] is a convolutional neural network built for use in mobile phones, robotics, self-driving cars or augmented reality applications. It was built to deal with the continually increasing demand for machine learning use cases within consumer or business electronics. MobileNets are built with a “’streamlined architecture that uses depthwise separable convolutions to build light-weight deep neural networks.” Mobilenet uses smaller and more factorized convolutions which transform regular convolutions into a depthwise convolution and a type of convolution called a pointwise convolution. Each convolution node combines the above inputs into outputs for the next layer all within one step. This is depicted in Figure 4.

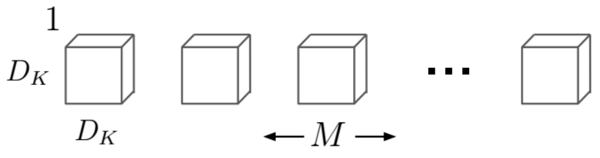
This structure uses many of the aforementioned convolutional layers, combined with the Rectified Linear Unit (ReLu) [9] activation function which provides the best results. There is a final softmax layer which performs the classification itself. This structure uses the ’ADAM’ [10] method for optomized calculation of the loss function during training, which has been shown to provide improved performance. The full structure of MobileNet is shown in Figure 5.

F. You Only Look Once V3

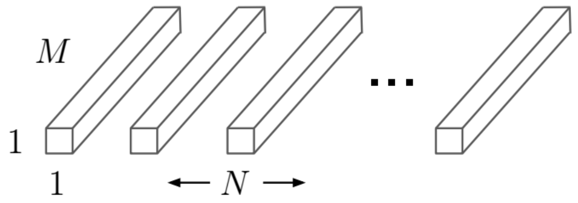
You Only Look Once (YOLO) Version 3 [2] is used in this project for object detection. YOLO is a modern object detection system. The neural network is applied to the whole image, instead of applying a model to various iterations of scales and locations within individual images. An image is split into smaller sub-images and a probability is applied to each of these individual regions. This yields an overall understanding and perspective of an image instead of only classifying the main feature of an image, which is a beneficial aid for the object detection process. YOLO uses a package ’Darknet’ [2] for displaying the bounding boxes on images.



(a) Standard Convolution Filters

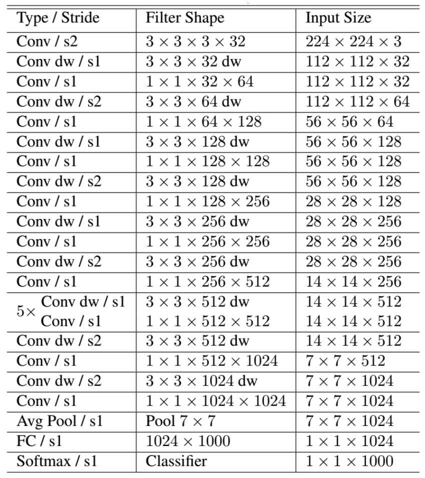


(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Fig. 4. MobileNet



Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
5× Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Fig. 5. MobileNet Structure

III. Methodology

A. Overview

In this research, the goal is to perform object detection with tree images and data. We used a pre-trained object detection model (YOLO V3), converted and coupled with the backbone of MobileNet which serves as the convolutional neural network itself.

B. Pipe-Lined Convolutional Neural Network

The model we chose to use for this project, developed by Adam Yang [11], is pipe-lined version of YOLO V3 and MobileNet. This model implements important features from both. This means that the MobileNet structure is used as the activation layers as the hidden layers, whilst YOLO feeds the input into the network and performs output operations. This hybrid model allowed for increased speed and performance during training.

C. Dataset

The dataset used in this project was provided by Aerobotics. The dataset includes images of Apricot Trees in a wide variety of shapes and forms. The dataset included images of trees that are closely bundled as well as trees that are sparse and far apart.

D. Data Preprocessing

The data for training, validating and testing the model was acquired by flying a drone over various apricot tree canopies. The drones capture RGB imagery as well as near infrared data, Normalized Difference Vegetation Index (NDVI) data and Digital Elevation Model (DEM) data. This information was stored in a hierarchical data format, via HDF5 [12]. In each image, the trees were manually outlined and annotated - here a binary mask was created where a 1 is stored if there is a tree, and 0 if there is not. This binary mask is also stored in the h5 directory. A diagram of the entire preprocessing is shown in Figure 6.

1) Layers of Transformed Image: From the HDF5 data structures, we need to extract three layers of input data for the neural network. The reason we only extract three layers is because the pre-built model [11] (discussed above) only allows for a three layer input - typically RGB imagery. We extract the following three layers to compile a new transformed image:

- The Digital Elevation Model data (DEM data)
- The Green layer from the RGB image
- A luminance layer (Y from YIQ), converted from the RGB image

2) Creating the Masked Image: The set of trees from the newly transformed image created above are then extracted to an entirely separate image. This was done using the binary mask stored with each associated image. The binary mask was matched against the transformed image and wherever there is a 1, the new image would store the same pixel as the transformed image.

3) Individual Trees: From these masked images and the transformed images, the individual trees from each image were selected with bounding boxes using the VOC annotation format as required by the model. For this, we used the software 'LabelImg' developed by Tzutalin [13]. The locations of each tree within each image were stored in a text file that would be used by YOLO during training.

E. Keras

All of the infrastructure used for our model was coded with the Keras [14] framework on Python. Keras is widely considered one of the most high-level, modular and easy-to-use frameworks for developing complex neural networks [15]. Due to

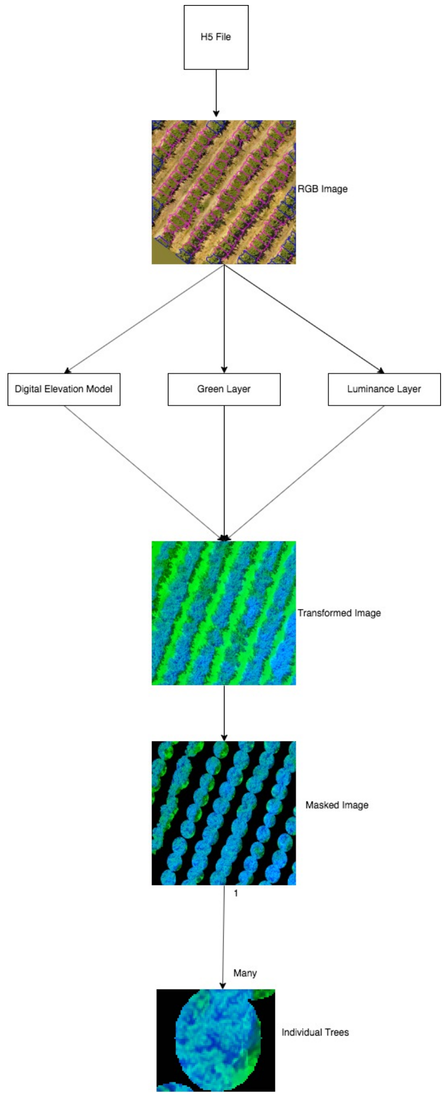


Fig. 6. Data Preprocessing

the compact and mobile nature of this research, Keras seems a fitting match for our solution. Keras uses Tensorflow as its backend which provides fitting handling operations for both CPU and GPU usage.

F. Experimentation

1) Training Environment: All training, validating and testing were performed without using a Graphical Processing Unit. The training was performed on a Macbook Pro 2015 model with only a 4 core 2.7 GHz Intel Core i5 processor and only 8GB of memory. The reason for this was attempting to demonstrate the portability and mobile nature of our model. Each iteration of training roughly averaged in around 30 hours per iteration, whereas the detection/prediction module is almost instantaneous.

2) Training: We performed transfer learning for the training of our model. The model uses the YOLO V3 pre-learned weights. This uses the MS COCO datataset [16] which is already trained to detect 80 different classes. We decided to freeze the first 20 layers of the neural network whilst performing the transfer learning. We found that a larger epoch performed better for our dataset so after experimentation we decided to use an epoch size of 115 and batch size of 32 during the final and latest iteration of training. The convolutional neural network was trained using 750 images, 600 for training and 150 for validation. The input data was shuffled and randomized. Furthermore, batch normalization was performed on the dataset which helped with the image being in a different form to the typical RGB imagery training data.

IV. Results and Discussion

A. Overview

Upon completion of experimenting and tuning the hyper-parameters of our model, such as epoch size, learning rate, batch size and the amount of training images, the model is tested based on the following factors:

- How the CNN model performs for trees are well separated.
- How the CNN model performs for trees that are closely bundled together.
- The number of trees detected as compared to the ground-truth.

In the context of our research, we are particularly concerned with the ability to determine the bounds of trees within an image, rather than the confidence probability of a bounding box surrounding a tree. This is why there is a low threshold whereby if the neural network predicts that something is a tree with a probability of 30% or higher, it is accepted as a tree. We are more concerned with how the neural network predicts the shapes and bounding boxes surrounding the trees that it has detected. This will be discussed in the following sections below.

B. Well Separated Trees

For trees that are easily distinguishable by the human eye, the model performs well. This is most likely due to the fact that the convolutional neural network had to use hand-annotated images with bounding boxes containing the items of individual trees hence the images could only be annotated using samples whereby an untrained human could easily differentiate the trees. Thus, the network was trained using images of trees that are relatively separate and apart from one another. Here, by inspection, the bounding boxes created by the output of our model surrounded the trees accurately and could account for the variability in tree size and shapes. Examples of this can be seen in Figure 7.

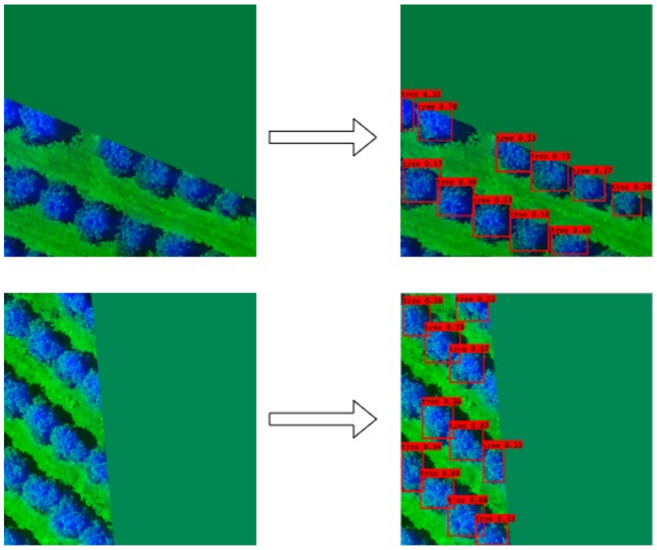


Fig. 7. Model Output For Well Separated Trees

Upon a further inspection, the trees for these type of images are surrounded by highly accurate bounding boxes produced by YOLO. The bounding boxes accurately account for trees that are clearly in the image as well as trees that are cropped in the periphery of the image such as in Figure 8. The model also performs well in that it neglects shadows created by the trees in the images. This is also most likely due to the accuracy of the annotations produced for the training images.

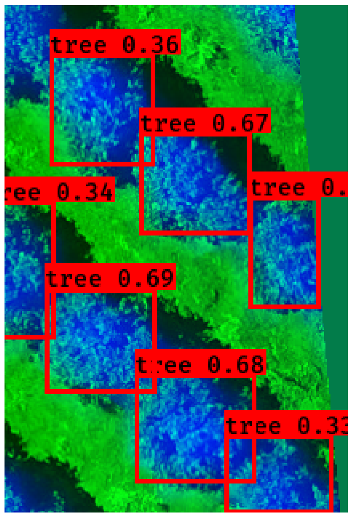


Fig. 8. Accurate Bounding Boxes

C. Closely Bundled Trees

For trees that are not easily distinguishable by the human eye, the model performs poorly. This is most likely due to the fact, once again, that the convolutional neural network had to use hand-annotated images with bounding boxes only containing individual trees hence the images could only be annotated using samples where an untrained person could easily differentiate the trees. With these images of closely bundled trees, the network struggles to produce any output, predictions or bounding boxes. For these type of images, in the rare case, the network is able to produce one or two bounding box predictions. There are specific instances of images whereby the tree sizes (shown in the DEM data) are in a very similar range - here the network is unable to produce any output as it is unable to determine where the closely bundled trees begin and end. Examples of this can be seen in Figure 9.

The tree images that our model finds difficulty with are typically images that present difficulty for humans in determining where a tree begins or ends. From our dataset, these images are in some cases blurred and smudged which presents difficulty for both humans and a neural network in delineating individual trees. Examples of these types of images are shown in Figure 10 and Figure 11. Thus, our model should not be used for applications or use cases whereby the trees are so closely bundled to the point that they are indistinguishable.

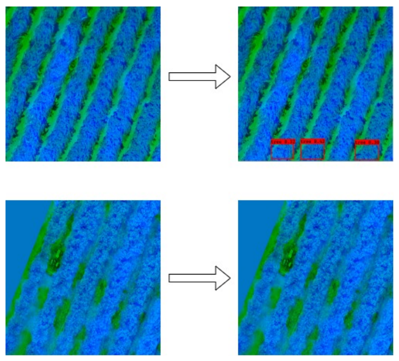


Fig. 9. Closely Bundled Trees

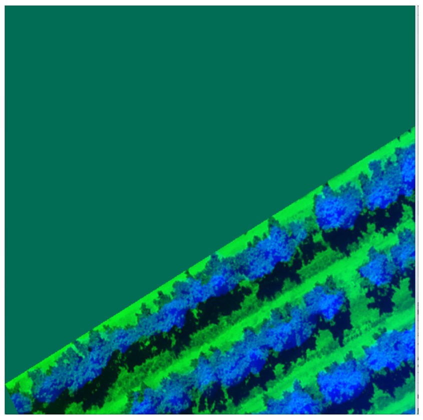


Fig. 10. Closely Bundled Trees Example

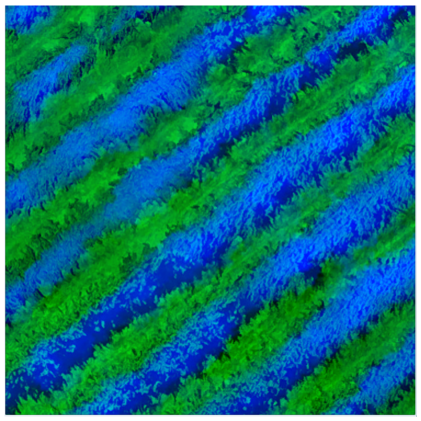


Fig. 11. Smudged Trees Example

D. Overall Accuracy

In order to determine how accurate our model is, a comparison needs to be made of how many trees are detected by our CNN model and the number of trees that are in the ground-truth. In order to do this a set of 50 test images were randomly selected from the Aerobotics dataset. For each image we manually recorded how many trees are in the ground truth. We then used the model to perform predictions for each image. We neglect any false positives for the detection when counting the number of trees detected. We further neglect any other items that may have been detected during prediction. To acquire a detection accuracy percentage the following formula was used:

$$\text{Detection Accuracy} = \frac{\text{Number of Trees Detected by the Model}}{\text{Number of Trees in the Ground Truth}}$$

By performing this manual comparison our model achieved a tree count detection accuracy percentage of 60.1%. This percentage can be compared to the convolution neural networked developed by Li et al. [7] who achieved an overall accuracy percentage of 85.1%. While the accuracy of the model by Li et al. [7] is significantly higher than our own, they trained their neural network with over 7200 training images whereas we only trained our model with almost 10 times less images. This information is depicted in table 1 below.

Evaluation Index	**Our Model**	**Li et al. Model**
Number of correctly detected trees	358	1397
Number of trees in the ground truth	591	1642
Accuracy	60.1%	85.1%

The results could be improved with a higher number of training image; however we attempted to demonstrate a convolutional neural network that is constrained by limited computing ability. Thus from the results produced, our model could be sufficient for mobile and portable use cases in agriculture and farming where a higher accuracy percentage is not needed.

Moreover, our model only accepts a three-layer input image that is typically associated with RGB imagery. This constrains the ability to use other forms of information as input, such as normalized difference vegetation index data (NDVI) or near infrared data that the drones used in our research captured. Due to our network being lightweight in nature, this information had to be discarded. However, despite this lack of useful information, an accuracy of over 60% could be seen as sufficient for a neural network operating under these constraints.

V. Conclusion and Future Work

We use a pipe-lined convolutional neural network that implemented parts of MobileNets and YOLOV3 in order to detect Apricot trees from drone imagery. We used both RGB data as well as Digital Elevation Model data as inputs for the network by performing multiple transformations on these images. We found that our mobile and compact network had an overall detection accuracy of 60.1% when predicting tree bounding boxes on the images from our dataset.

Our model performed well when the images used for testing are of trees that are not closely bundled together. When the trees are relatively separate and are distinguishable by the human eye, the neural network is able to make accurate predictions whilst detecting trees and the bounding boxes are mostly of the correct shape. Thus our convolutional neural network is able to achieve detection for this use case.

However, we found that our model should not be used in cases where there are trees that are close to one another and indistinguishable based on aerial imagery. The network performs badly for these images because it was trained using images of individual and distinguishable trees.

In conclusion, our convolutional neural network model should be used in applications that will not rely heavily on accuracy. Examples of these use cases could be crop farming or even tree-count surveying for regions where there are a limited number of distinguishable and separate trees.

A possible improvement on our model could be the inclusion of a larger number of training images from the dataset. By having more training images the neural network would be able to provide more accurate detection and counts of trees. A further improvement would be allowing for more than three layers of input into the network - this way we could include all of RGB, DEM, NDVI, near infrared and CI data into the prediction and training processes.

References

[1] G. Steyn, "Watch: Aerobotics uses artificial intelligence to make farming easier," 10 2018.

[2] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[3] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[4] E. K. Cheang, T. K. Cheang, and Y. H. Tay, "Using convolutional neural networks to count palm trees in satellite images," *CoRR*, vol. abs/1701.06462, 2017.

[5] L. Wang, P. Gong, and G. S. Biging, "Individual tree-crown delineation and treetop detection in high-spatial-resolution aerial imagery," *Photogrammetric Engineering & Remote Sensing*, vol. 70, no. 3, pp. 351–357, 2004.

[6] B. G. Weinstein, S. Marconi, S. Bohlman, A. Zare, and E. White, "Individual tree-crown detection in rgb imagery using semi-supervised deep learning neural networks," *Remote Sensing*, vol. 11, no. 11, p. 1309, 2019.

[7] W. Li, R. Dong, H. Fu, and L. Yu, "Large-scale oil palm tree detection from high-resolution satellite images using two-stage convolutional neural networks," *Remote Sensing*, vol. 11, no. 1, p. 11, 2019.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[9] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[11] A. Yang 2018.

[12] The HDF Group, "Hierarchical data format version 5," 2000-2010.

[13] Tzutalin, "Labelimg," 2015.

[14] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[15] J. Brownlee, "Your first deep learning project in python with keras step-by-step," 7 2019.

[16] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.