Tree Count using Deep Learning and Drone Imagery

UMER ZEB KHAN

DEPARTMENT OF COMPUTER SCIENCE

DR. JUNAID AKHTAR

Abstract

A method is proposed for the automatic classification and count of trees using RGB images obtained via a commercial Unmanned Aerial Vehicle, a Ryze Tello quadcopter drone. The dataset used for the training of the Deep Convolutional Neural Network is curated using there different kind of images: Aerial imagery with field of view of 180 degrees (Parallel to the tree), Street view imagery (Perpendicular to the tree) and Drone captured imagery with field of view of 87 degrees (Diagonal to the tree).

Two solutions, one local and the other cloud based, are formulated where each yields an average detection of 77 % and 93 %, respectively. Publicly available packages for deep learning like tensorflow, darknet, opencv and pytorch are used to construct an intelligent vision system capable of localizing and counting trees from aerial imagery. Pertinent to mention is the lack of expensive sensing devices in our implementation, a stark shift from many of the previous studies along with demonstrable detection results with excellent potential for scalability of the proposed solution to work with hyper-spectral sensory data.

Acknowledgment

First of all, I am grateful to Almighty Allah, who helped me persevere and persists in the face of hardships and turmoils. Followed by my dedicated group of companions and friends who pushed me literally to achieve what I so easily would have given up on. Without you, I would not be here, thank you.

This prose shall remain incomplete without expressing my gratitude to Dr. Junaid Akhtar for his unconventional yet charismatic and insightful mentor-ship throughout the course of this year long project. I am humbled, and shall remain forever indebted.

A faithful thanks to my parents who deserve acclaim for putting up with my tantrums, and whose sincere prayers and well wishes has propelled me to achieve great things. They shall always remain my shimmering light.

Last but not the least, to all the marvelous individuals that remain anonymous on stackoverflow and other internet communities working hard and out of passion to help advance the scientific cause and the people who write tutorials, and make exhaustive youtube videos to make knowledge sharing easy. It is truly impossible to progress without the intellect and genius of such contributors. Thank you very much, whoever you are and wherever you are.

Contents

List of Figures

Chapter 1

Introduction

Pakistan is one of the most prone countries to climate change and global warming, as assessed by several environmental agencies throughout the world. Leaders and people in position of authority usually announce large scale plantation drives. The move certainly earns them the goodwill of many people but independent verification is difficult due to many factors, mostly due to ineffective plantation techniques such as spraying seeds over large tracts of land from aerial vehicles i.e. helicopters, and lack of post plantation care of trees for their fragile years of initial growth along with the tree's mapping.

Prime Minister Imran Khan has been elected in 2018 whereby one of his electoral promises has been to plant 10 billion trees. In a recent development, the premier's advisor of climate change, Amin Aslam has invited third parties to enhance transparency in the Clean Green Pakistan initiative by evaluating and monitoring the mass plantation (The Express Tribune, 2019).

The problem statement is: detection of trees in order to determine their count using a drone equipped with camera. The objective of this project is to facilitate the tracking of trees over a period of time and confirm the veracity of claims made by philanthropists. The motive is not to verify the claims of mortals, but rather to devise a mechanism to seamlessly monitor tree plantations for their utmost significance as a valuable part they play in our ecosystem and in order to fight the very real and potent threat of global warming.

1.1 Acronyms

CNN → Convolutional Neural Network
R-CNN → Region-based Convolutional Neural Network
RoI → Region of Interest
IoU → Intersection over union
FOV → Field of Vision
LIDAR → Light Detection and Ranging
NIR → Near Infra Red spectroscopy
RGB → Red Green Blue
MS COCO → Microsoft's Common Objects in Context
PASCAL VOC → PASCAL's Visual Object Classes
IR → Infra Red Light
X-ray → An electromagnetic wave
UV → Ultraviolet Light

MP → Mega Pixels
HD → High Definition
GPU → Graphics Processing Unit
API → Application Programming Interface
ML → Machine Learning
HSL → Hue, Saturation, Lightness
RPN → Region proposal network
ReLU → Rectified Linear Unit
GPS → Global Positioning System

1.2 Document's Flow

The next segment of the report sheds light on various similar technologies and techniques being employed by scientists and engineers to solve applied computer vision problems and how they converge with the defined scope of my project. Every single paragraph is written to feature one main idea and a transition to the next is a logical progression.

Literature review is closely followed by requirements and analysis section which explains in depth the prerequisites of the project and a methodological analysis of it. The fourth chapter consists of the project's proposed design, actual technological implementation and testing strategy employed to evaluate its performance.

The fifth chapter includes a thorough discussion on the obtained results of our tree count problem and a critical analysis of it inclusive of visual and critical aspects. The last chapter is the conclusion of this report and my attempt to highlight achievements and future work. The end consists of a bibliography in Harvard referencing style.

Chapter 2

Literature Review

This chapter deals with the literature consulted in the pursuit of my own agenda for counting trees. An overview is given regarding the imaging techniques available in the market to obtain different types of parameters and how they can be processed. Moreover the crucial impact of the data used for model training and the varying categories of prepared datasets influence on detection and accuracy is debated.

The current state of the art work done within the domain of computer vision from aerial imagery can be divided primarily on the basis of their input: LIDAR, NIR and/or RGB. The former two are remote sensing technologies that provide radiometric and geometric information regarding the Earth's surface (Yang, L. et al., 2009) while the latter provides limited information depending on the algorithm used and the manipulation of data set.

Recent advances in computer vision are attributed to the availability of large and small datasets alike, allowing researchers and scientists to not only tweak their algorithms and refine their models but to make innovations possible by the availability of high quality datasets.. Multiple years of hard work have resulted in exhaustive data sets for training and validation, seven years to be precise in the case of Pascal VOC (Everingham, M. et al., 2009) consisting of 20 object classes of more than 11, 000 images with a little more than 27, 000 region of interest annotated images and about 7000 segmentations.

Another comprehensive dataset available is ImageNet (Deng, J. et al., 2009), by far the biggest and largest dataset, consisting of 2200 objects within 15 million images, with average images in the range of 500-1000 belonging to each class of available objects, based on the hierarchical structure of WordNet inclusive of only the nouns (Fellbaum, C., 1998). The dataset has been used extensively in the field of image processing and computer vision due to its scalability, abundance of labelled images and hierarchical structure.

OverFeat is an intelligent feature extractor developed by (Sermanet, P., 2013) using their best performing model that provides an integrated framework using convolutional neural networks to classify, detect and localize images for an annual competition called the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). A subset of ImageNet with 1000 object and roughly the same number of images in each class is used for the competition. All the three tasks are fundamental in computer vision and consequently recognition of interest regions that are marked using bounding boxes, all of these processes taking place simultaneously using just a single convolutional network and a common feature learning base.

Any mention of datasets shall remain inconclusive without the name of MS COCO that has 91 categories of object with about 328, 000 images with 2.5 million instances (Lin, T.-Y. et al., 2014). The state of the art dataset put the agenda of object recognition in the broader context of scene comprehension, something a child does effortlessly but still poses a huge challenge for computers. Due to the versa-

tility of the data set, it has been used commonly to solve classical computer vision problems such as mentioned in (Mottaghi, R. et al., 2014) that the context is usually an important cue to weigh in before making a prediction regarding the object.

Applications of image procession in the field of agriculture are many including fruit grading, plant health, plant count, weed detection and imaging techniques (Vibhute, A. & Bodhe, S.K., 2012). Imaging techniques denote the type of images obtained for analysis, using the change in the wavelength of light as a primary parameter, such as IR imaging, Radio imaging, X-ray imaging, UV imaging, imaging in the visual band ect (Gonzalez, R.C. & Woods, R.E., 1993). Electromagnetic radiations are employed to infer geo-biophysical characteristics and surface features of the earth for identification, the process is known as Remote Sensing. Further applications are determination of water stress that helps in irrigation, management of floods, plant disease monitoring, environment assessment, mapping of manmade and natural plantation etc (Hänsch, R., Schulz, K. & Sörgel, U., 2018).

Palm trees are an important economic crop of Malaysia. A deep learning based solution is proposed for detecting palm trees, their count, estimating their yield and enhancing their productivity (Li, W. et al., 2016). Traditional methods for crown detection of trees, namely, template matching, local maximum filter and artificial neural network have also been implemented to draw a comparison between pre-existing models and the custom deep convolutional neural network. It's no surprise, a deep learning model specifically tailored to detect and count palm trees fared the best.

Most relevant to my area of interest is the work of Yang, L. et al. that proposes a mechanism for the detection of trees from aerial imagery. A pixel level classifier to applied to an image to distinguish between tree and non-tree regions, being further refined by a partitioning algorithm. Once refining is done, a clean image segmentation is obtained for trees and non-trees followed by template matching. In the end, greedy selection is done for the location of individual tree crowns (Yang, L. et al., 2009). The proposed solution works on pure images only while delivering satisfiable performance on large scale tree plantations.

Chapter 3

Requirements and Analysis

3.1 Overview

For a moment, just think about how a kid of four intuitively knows the deformed entity on the sofa is not a tiger but rather a cat as evident from Fig. 3.1, similarly the small object on the table is a fork, not an elephant's tail. The reason has to do with their remarkable ability to put thing within the right context after having seen or/and interacted with it previously. That forms the basic premise of a neural network, where they are taught or either left to learn themselves, from given data.

CNNs learn just like toddlers, where experience if positive strengthens a specific neural pathway, while negative feedback diminishes it. The learning is directly dependent on the labeled data provided to test and train the neural network. Our problem requires trees to be detected from spatially rich drone imagery in RGB spectrum.

Figure 3.1: Tiger or Cat?

3.2 Analysis

There are several steps in the process pipeline before you arrive at a system capable of counting tree from a given image. The process begin with gathering the right type of data, in our case, images captured by a 5 MP camera fitted on Ryze quadcopter drone. The second step is to label the region of interest, trees, in each of the image manually using Labelmg as annotation tool. The third step is the division of the

data into two chunks, conventionally named, training and testing data. Both the data chunks are passed to the CNN for training and testing, both of which are resource intensive and time consuming processes. The last step includes giving images to the model to detect and count trees.

3.3 Data and Hardware requirements

Garbage in, Garbage out: nowhere is this more important that teaching a model to detect a specific object hence the need for high quality, versatile and soundly labeled data. So a customized dataset is prepared for our problem using a drone equipped with a 5 MP HD camera. In our case, the drone is a quadcopter, Ryze Tello. Additionally, training of a CNN requires a powerful, dedicated GPU equipped system with an active Internet connection and enough space to accommodate large datasets and model configuration files.

3.4 Methodologies

3.4.1 Computer Vision

Computer vision, in its simplest form is, the ability of a computer to "see" and process visual data. More formally, it is an interdisciplinary field of study aimed at understanding why the human visual system works and ultimately replicating it. It is the process of extracting, analyzing and understanding visual information present in images or videos (Rosenfeld, A. & Kak, A.C). A digital image can be defined as a continuous sample of light radiations reflected from different objects onto a rectangular array of pixels. Digital signal processing can be applied to extract information and generate meaning out of these numbers (Szeliski, R., 2010).

With the deep learning revolution, the domain of computer vision has progressed exponentially and can now boast about object detection and scene comprehension from videos and images, alike. In order to understand an image, three primary levels of abstraction are utilized to make sense of the sensory data: at the bottom includes the basic components of an image i.e. edges, texture elements or regions; a higher level of abstraction is inclusive of boundaries, surfaces and volumes; while the highest level of comprehension includes objects, scenes, or events recognition (Liu, Z. et al., 2018).

3.4.2 Neural Networks

A neural network mimics the human brain (J. DeMuro, 2018). The neural network learns via an algorithm by incorporating new data. A neuron also known as a node or unit and is the basic unit of computation in a neural network. These neurons are then connected into a large mesh networks. The nodes receive inputs from an external source or from some other nodes and compute an output. All these nodes have some unique weights (w), which is assigned based on its relative significance to other inputs (Agatonovic-Kustrin, S. and Beresford, R. 2000). The nodes apply a function to the weighted sum.

So x_1 and x_2 are the two inputs, where w_1 and w_2 are the associated weights to the two inputs respectively. Additionally, another input is 1 with weight b, known as the bias. The bias provides a trainable constant value to every normal input that the node receives. This bias function value allows moving the activation function to the left or right. The above f function is a non-linear activation function. An activation function introduces non-linearity into the output of a node because most of the real world problems are non-linear (Agatonovic-Kustrin, S. and Beresford, R. 2000). In practice, there are several activation functions e.g. sigmoid, tanh and ReLU.

Any neural network can be thought of as a network of "neurons" which is organized in layers and those layers are defined as follows:

- Input Layer → It takes predictors/inputs and attaches coefficients to them which are called "Weights".
- Hidden Layer → It makes the neural network non-linear and improves the performance. Neural network can be linear with no hidden layer.
- Output Layer → It outputs the results generated from weighted predictors passed through hidden layers, if any.

A Rectified linear unit (ReLU) takes a real valued input and threshold it with zero, replacing every negative value with zero. This operation has been used after every convolution operation. As convolution is a linear operation but most of the real world problems are non-linear, so for non-linearity, ReLU is introduced. For a processing task in deep learning, neural networks cannot be programmed directly like other algorithms. Neural networks use some strategies to learn information e.g. supervised learning, unsupervised learning and reinforced learning (Agatonovic-Kustrin, S. and Beresford, R. 2000).

Neural Networks' behavior is determined by its neurons' particular transfer functions, by learning rule and architecture. Transfer function, also called Activation functions, defines the output of given input or set of inputs to a node of neurons is shown in Fig. 3.2.

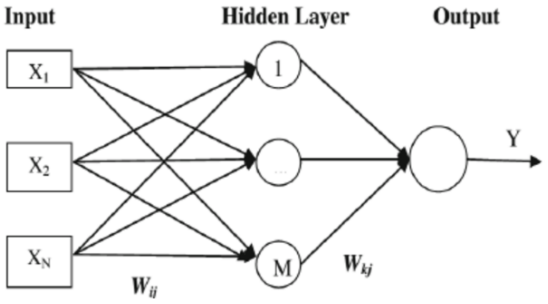


Figure 3.2: Basic Functioning of Neural Network.

Applications of the neural networks can be summarized into classification or pattern recognition, prediction and modeling.

3.4.3 Data Augmentation

Training of a machine learning model demands the availability of labeled data critically. Often times, while training a neural network, the dataset available is not of desired quality, or is too small in size. This relative scarcity of data results in a neural network with a very low accuracy and may result in over-fitting. To alleviate this problem, the most effective approach is data augmentation which is the technique of artificially increasing the size of a dataset by applying transformations such as rotation, flipping, cropping, or adding noise/filter to images. This technique is proven to be quite effective in training neural networks with limited data. For example, if we have a neural network trained on a dataset containing 100 images, it will not be as efficient as a neural network trained on a dataset containing ten times as much images (Asperti & Mastronardo 2017).

Image labeling or annotation is the process of defining regions in an image and giving them a textual label for creating ground truth. Image annotation tools usually take the input as RGB images. However modern widely available sensors such as Microsoft Kinect can use depth images in addition to RGB images (Pordel and Hellstrom, 2015). One of the many magnificent features of the humans' visualization is to create rich description of a scene within a glance at an image. They can tell what are the objects and their associated attributes in the image. For example, an image can be defined as "containing a person, sitting in a shiny red car wearing a white shirt". Moreover, humans can effortlessly describe each object in the image. During the past few decades, one of the fundamental objectives of computer vision research has been to replicate this ability, resulting in-depth studies of computer vision problems including detection of objects in a scene by describing them using their attributes and creating the ground truth (Felzenszwalb, 2014), (Everingham et al., 2007), (Farhadi et al., 2009), (Kulkarni et al., 2013), (Lampert, Nickisch and Harmeling, 2014).

Chapter 4

Design Implementation and Testing

The development process is primarily divided into three different parts. The first part involves capturing and annotating the data. Followed by training of a deep CNN classifier to detect trees. Last but not the least, video captured by the drone is processed to extract the frames and stitch them together to form a 360 degrees panoramic image. The aim is to use computer vision techniques and machine learning to teach a neural network to detect and count trees in a given image.

4.1 Technologies Used

Tensorflow is an open source deep learning library developed by Google and is widely used for training and constructing machine learning models. In comparison with Keras, it allows for a lower level abstraction for implementation of neural networks. Tensorboard is used to analyze, evaluate, numerically compute and debug tensorflow graphs (Metz et al., 2018). .

Python is a high-level programming language created by Guido Van Rossum, python is released in 1991 (Michael Dawson., 2009). Python can be used for an extensive variety of applications. The latest version of python is python 3.7, used in this project.

OpenCV is an open source library including of the most robust implementations of both machine learning and computer vision algorithms (Baggio, 2012). With interfaces for different programing languages and operating systems the library was designed for computational efficiency aiming at real time computer vision systems. As the library goals to deliver support for real time computer vision systems, advanced data storage and transfer protocols have been added to allow easy management of image data.

NanoNets is machine learning API for developers which requires 1/10th of data and no machine learning expertise to train a model. Upload the data, wait for the confirmation email and get a model you can query over their easy to use cloud API. Often companies do not have enough data to train a machine learning model on their own using state of the art algorithms as well as don't have enough data scientists to work on those problems (Anon, The Machine Learning API). NanoNets solves both the problems for interested individuals and companies while providing three ways to integrate the model in your problem definition: API, Docker or mobile SDK.

4.2 Problem approach

Two solution are proposed and implemented for the problem of counting trees. Both the approaches used are machine learning based to classify trees, but the way in which they differ is their deploy-

ment. Namely, a local machine based approach and a cloud hosted machine learning API, NanoNets, is adopted.

The local machine based approach uses transfer learning to retrain a Resnet50 CNN model to detect and classify trees. The cloud based approach follows the the same process pipeline of retraining an existing model as highlighted in the Fig. 4.1 Additionally, the challenge of counting trees remain in accurately identifying them in an high resolution, abundant noise and excess information aerial image captured by a low budget unmanned aerial vehicle.

Figure 4.1: Steps involved in transfer learning.

4.3 Implementation

4.3.1 Dataset preparation

For my problem of tree detection, I curated a dataset of 1260 images where two types of images, Street view and Aerial view, were retrieved from Pasadena Urban Trees dataset (Wegner, J.D. et al., 2016). The

images can be classified on their FOV: Parallel to the ground, Perpendicular to the ground and Oblique to the ground. Type of data augmentation techniques used to enhance the existing data set are done using geometric transformation, noise and weather scenarios. The annotation of the images is done manually using a tool called lablemg which stores the coordinates of the bounding boxes in the format of y_min, y_max, x_min, x_max in an xml format file.

Augmentation happens in two major ways. The first one is the rotation along the horizontal axis, vertical axis and on both the axis using OpenCV's cv2.flip function and the stored dimensions of the image in the xml file generated after labelling. The second way is the use of filters, in my case, 7 types of filters are used i.e. Motion blur, blur, rain, salt & pepper, snow, shady and sunny.

Salt & Pepper filter is applied by randomly selecting pixels in a source image and adding white (255) and black (0) dots on the image. It results in an artificially created salt and pepper noise on the image, changing its features. The rain filter is applied by using cv2.line function of OpenCV which draws on images by taking as input the source image, drawing coordinates, drawing size, and drawing color stated the drawing coordinates are selected randomly. The image is converted from RGB to HSL to reduce the value of the 'light' channel and converted back to RGB. This results in a darkened image. Motion blur happens when a kernel convolves on the source image using cv2.filter2d function of OpenCV.

Orignal image is shown in Fig. 4.2 followed by augmented images as evident from Fig. 4.3, Fig. 4.4, Fig. 4.5, Fig. 4.6, Fig. 4.7, Fig. 4.8, Fig. 4.9, Fig. 4.10, Fig. 4.11 and Fig. 4.12 respectively.

Figure 4.2: Original Image.

4.3.2 Tree Detection and Count

Figure 4.3: Image Augmentation Blur.

Figure 4.4: Image Augmentation Motion Blur.

Figure 4.5: Image Augmentation Horizontal Flipped.

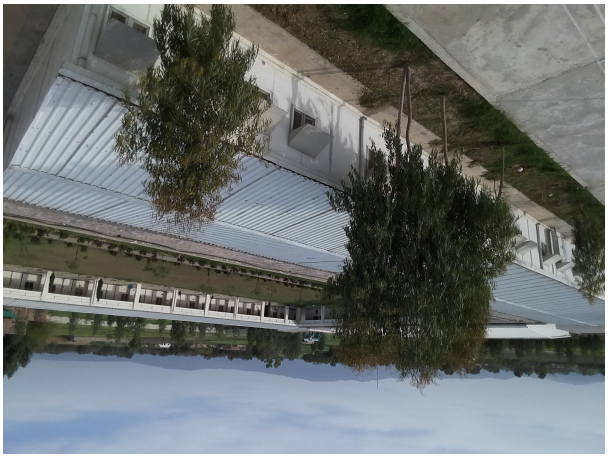


Figure 4.6: Image Augmentation Vertical Flipped.

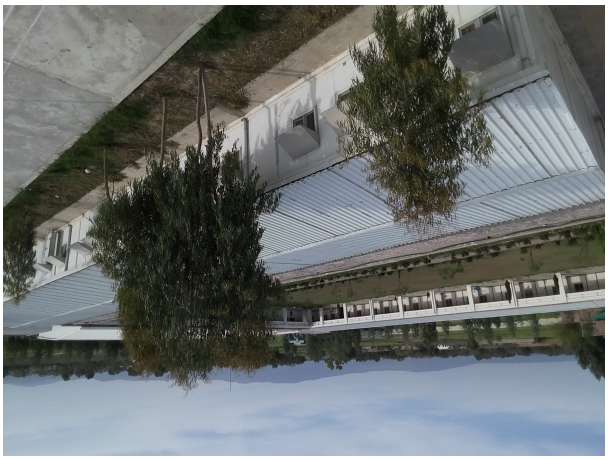


Figure 4.7: Image Augmentation Horizontal and Vertical Flipped.

Figure 4.8: Image Augmentation Sunny.

Figure 4.9: Image Augmentation Shady.

Figure 4.10: Image Augmentation Snow.

Figure 4.11: Image Augmentation Rain.

Figure 4.12: Image Augmentation Salt and Pepper.

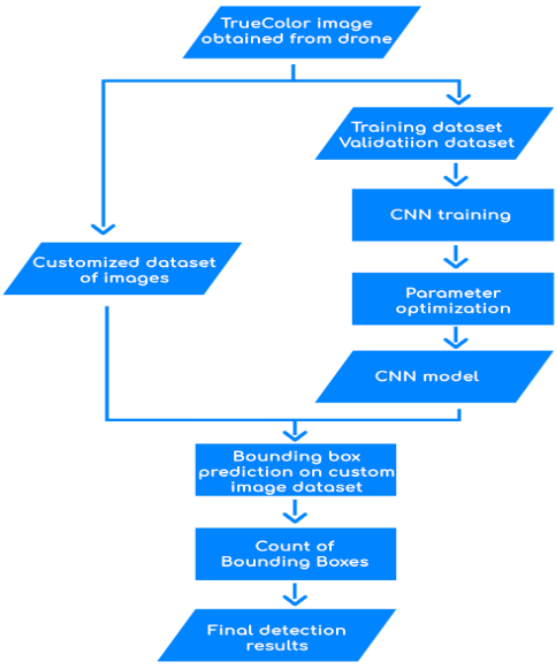


Figure 4.13: Flowchart for Tree detection and Count.

A model called Faster RCNN Resnet 50 (He, K. et al., 2016) trained on MS COCO dataset is re-purposed to detect trees. Faster RCNN is made up of three distinct networks: feature network, region proposals network and detection network with total network layers of 50. Transfer learning is used to tweak with the last layer of the model, where it's trained on my own dataset. The training/testing data is divided in the ratio of 1100 images for training while 126 images for testing. In order to test the performance of our neural network, a new set of images are passed which were neither used to train nor test the model where an average detection 77 % is achieved. Architecture of Resnet is detailed in Fig. 4.14

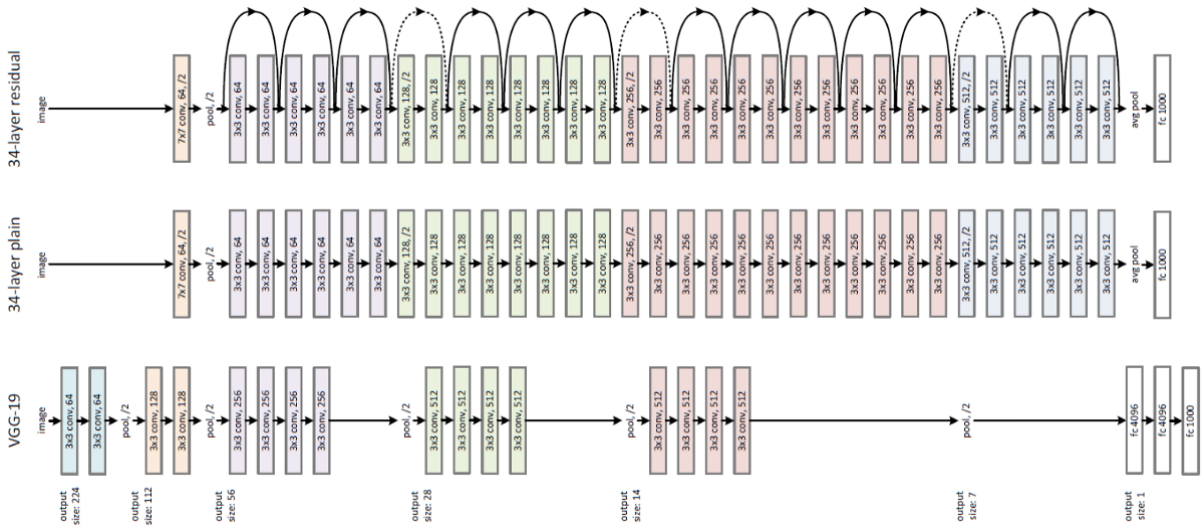


Figure 4.14: Resnet architecture.

Additionally, a cloud based machine learning API NanoNets is used to train another detector due to its easy integration and high level control. However previously annotated data cannot be used since NanoNets requires data to be labeled manually after uploading on their sever. The detection accuracy remains quite high at 93 % even after being trained on a mere 150 images with 300 annotations. Tree detection from the classifier results in an image of detected trees which are then counted by a python script that is hard-coded to count the number of bounding boxes produced in a given image by reading the distinct color of the bounding boxes. Result is shown in Fig. 4.15. Fig. 4.16 is attached which shows the integration code for the API.

4.3.3 Image Stitching

The Fig. 4.17 shows in detail the steps followed to stitch two images together (Brown, M. & Lowe, D.G., 2006). The captured frames of the video are extracted and processed to identify and track some relevant points in the image. Based on the difference between points in different angles, the program can estimate with relatively good accuracy the distances between all those points, the relative position of the camera and also, geometric shapes.

Photogrammetry is the science of getting measures based on photos. A camera image is just a projection of 3D information in a 2D plane (Nagamura, 2018). The limitation is, in the process of generating the stitched image, one dimension of information, namely, the depth is lost. To make up for it, an image from another angle or point of view is used to take measures by comparing the points

Figure 4.15: Cloud count of Trees.

```
#pip install requests
import requests, json
from requests.auth import HTTPBasicAuth
data = {
    "urls": ["https://s3-us-west-2.amazonaws.com/nanonets/replace_me.jpg"], \
    "modelId":"deb5387f-00a6-4255-bf0f-ecfc936a5bc6"
}

headers = {
    'accept': 'application/x-www-form-urlencoded'
}


url = "https://app.nanonets.com/api/v2/ImageCategorization/LabelUrls/"
r = requests.post(url, headers=headers, data=data,
auth=HTTPBasicAuth('mIFM8VsZcjOeMwuW2gTAom_LUoYI5vxo', ''))
print r.content
```

Figure 4.16: Nanonets integration code.

observed. Stereoscopic vision works in this way, as we have two eyes producing 2D information, in slight different positions, and which is enough for our brain to reconstruct the depth information. Image stitching process as outlined by Brown, M. & Lowe, D.G. is shown in Fig. 4.19.

A sample image is shown in Fig. 4.20 produced after stitching frames extracted from a drone video using OpenCV's cv2.Stitcher_create function where the code is shown in fig. 4.18. In order to extract frames from a video, the code is given in Fig. 4.21, notice how every 5th frame is extracted since the scene do not change significantly in consecutive frames.

4.4 Justification

This section deals with the rationale behind opting for the specific architecture, drone and technologies for our defined problem. Training a CNN from scratch is a labor intensive and computationally expensive task. Coupled with the unavailability of powerful, GPU equipped systems, transfer learning on a pretrained model was the only viable way to go while keeping in view the restrictions posed by the availability of a relatively small dataset. Necessity is the mother of invention, rightly so, gave birth to the alternate solution of cloud model as tinkering with models on a local system was a cumbersome process. Moreover, award winner of ILSVRC 2015 and MS COCO 2015 (Tsang, 2018) detection, segmentation, classification and localization, Resnet50 model is suitable for our problem as it is a deep neural network which has already learnt features that are relevant to our dilemma of tree detection with a proven

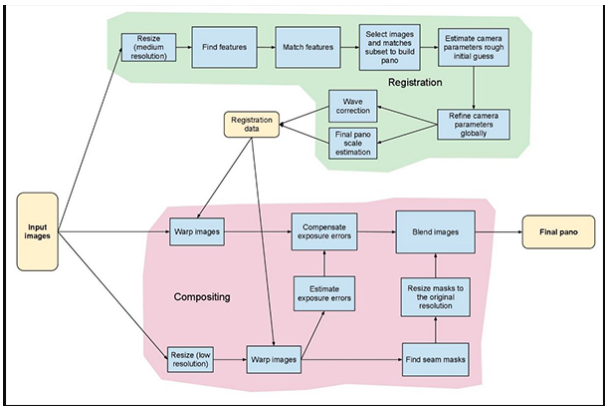


Figure 4.17: Image stitching pipeline.

performance.

Furthermore, to count the number of trees in a given area, the frequency of images to cover the entire area might be many hence a taxing task to separately pass the images to CNN for detection. In lieu of this, a video is made of the interested region from where the frames are extracted and stitched together to get estimated count in a single step. To enhance the accuracy of our model and avoid overfitting, data augmentation techniques have been applied to the dataset. The drone imagery collected is from Ryze Tello quadcopter, which happens to be a budget drone with a decent performance.

```
# USAGE
# python image_stitching.py --images images/scottsdale --
output output.png --crop 1

# import the necessary packages
from imutils import paths
import numpy as np
import argparse
import imutils
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--images", type=str, required=True,
	help="path to input directory of images to stitch")
ap.add_argument("-o", "--output", type=str, required=True,
	help="path to the output image")
ap.add_argument("-c", "--crop", type=int, default=0,
	help="whether to crop out largest rectangular region")
args = vars(ap.parse_args())

# grab the paths to the input images and initialize our images
list
print("[INFO] loading images...")
imagePaths = sorted(list(paths.list_images(args["images"])))
images = []

# loop over the image paths, load each one, and add them to
our
# images to stich list
for imagePath in imagePaths:
	image = cv2.imread(imagePath)
	images.append(image)

# initialize OpenCV's image sticher object and then perform
the image
# stitching
print("[INFO] stitching images...")
stitcher = cv2.createStitcher() if imutils.is_cv3() else
cv2.Stitcher_create()
(status, stitched) = stitcher.stitch(images)

# if the status is '0', then OpenCV successfully performed
image
```

```
# stitching
if status == 0:
	# check to see if we supposed to crop out the largest
rectangular
	# region from the stitched image
	if args["crop"] > 0:
		# create a 10 pixel border surrounding the
stitched image
		print("[INFO] cropping...")
		stitched = cv2.copyMakeBorder(stitched, 10, 10,
10, 10,
			cv2.BORDER_CONSTANT, (0, 0, 0))

		# convert the stitched image to grayscale and
threshold it
		# such that all pixels greater than zero are
set to 255
		# (foreground) while all others remain 0
(background)
		gray = cv2.cvtColor(stitched,
cv2.COLOR_BGR2GRAY)
		thresh = cv2.threshold(gray, 0, 255,
cv2.THRESH_BINARY)[1]

		# find all external contours in the threshold
image then find
		# the *largest* contour which will be the
contour/outline of
		# the stitched image
		cnts = cv2.findContours(thresh.copy(),
cv2.RETR_EXTERNAL,
			cv2.CHAIN_APPROX_SIMPLE)
		cnts = imutils.grab_contours(cnts)
		c = max(cnts, key=cv2.contourArea)

		# allocate memory for the mask which will
contain the
		# rectangular bounding box of the stitched
image region
		mask = np.zeros(thresh.shape, dtype="uint8")
		(x, y, w, h) = cv2.boundingRect(c)
		cv2.rectangle(mask, (x, y), (x + w, y + h),
255, -1)
```



```
		# create two copies of the mask: one to serve
as our actual
		# minimum rectangular region and another to
serve as a counter
		# for how many pixels need to be removed to
form the minimum
		# rectangular region
		minRect = mask.copy()
		sub = mask.copy()

		# keep looping until there are no non-zero
```

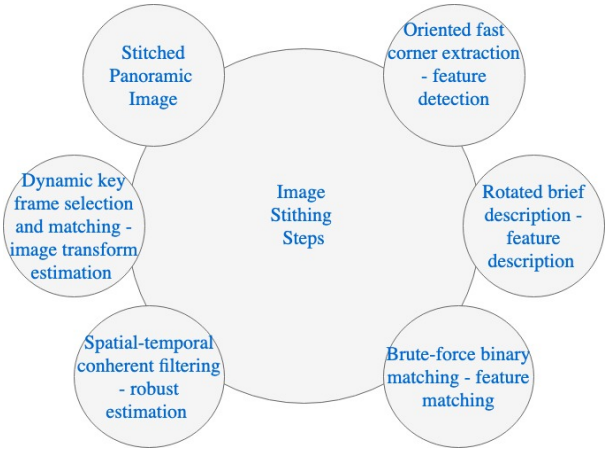


Figure 4.19: Image Stitching Process.

Figure 4.20: Image stitching example.

```
import cv2

vidcap = cv2.VideoCapture('E:\\Umer\\testvid.mp4')
success, image = vidcap.read()
count = 0
success = True
while success:
    print(count)
    success, image = vidcap.read()
    image = cv2.resize(image, (1280, 720), interpolation=cv2.INTER_AREA)
    if count % 5 == 0:
        cv2.imwrite("E:\\Umer\\Frames\\image%d.jpg" % count, image)  # save frame as JPEG
file
        print(count)
    if cv2.waitKey(10) == 27:
        break
    count += 1
```

Figure 4.21: Video frames code.

Chapter 5

Results and discussions

No project can be complete without its evaluation and a critical discussion of the results which is the main agenda of this chapter. Several images after tree detection are included and scenarios highlighted where improvements can be made. Moreover, keeping in mind the limitations encountered, a thoughtful analysis is given vis-a-vis the project's defined goals in earlier chapters. In order to calculate the efficiency of the model, a simple formula is used, where the accuracy is equivalent to number of detected trees divided by the ground truth value as; $Accuracy = \frac{true-detection}{ground-truth}$

Fig. 5.1 shows an image downloaded from Google to check the detection of trees in an image that the CNN has neither used for training nor testing. Since our model is specifically trained to count and detect trees from aerial imagery, I deliberately passed an image which was not aerial. The results have been surprising. It's pertinent to note how a bunch of trees are classified by a single bounding box ultimately being considered a single tree which results in an underestimation of the overall tree count of a given tact of land. My proposed solution is semantic segmentation of many trees classified as one, as it allows for greater precision however is a physically taxing task since it requires manual labeling of color line.

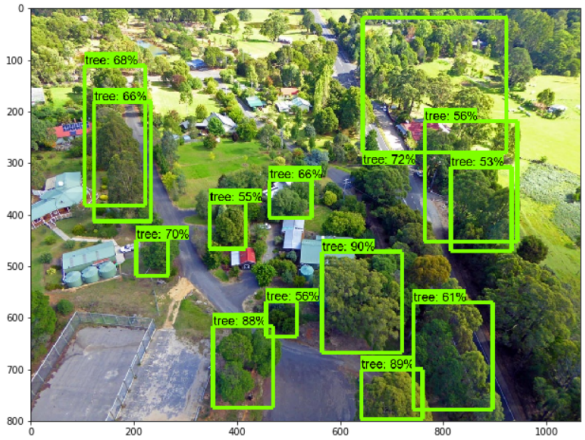


Figure 5.1: Result on image downloaded from Google.

Another test case is shown in Fig. 5.2, which is a street view image taken from a street camera. Excellent detection results are given, as partially visible tree on the left is also detected. However, trees hidden behind building were not detected even though their crowns are clearly visible. The plausible explanation is how the training and testing dataset influences further images detection since my dataset

contains trees labeled starting from the bark till their crown.

Figure 5.2: Result on image taken from a street camera.

As a final evidence, a self captured drone image is given in Fig. 5.3. You can notice how the small trees in the background are not detected. The reason is my starting assumption that any tree below the height of 6 feet shall not be counted and hence has not been labeled in my dataset. As a result, the solution lies in carefully re annotating the dataset to include small heighted trees in the training and testing to improve performance further.

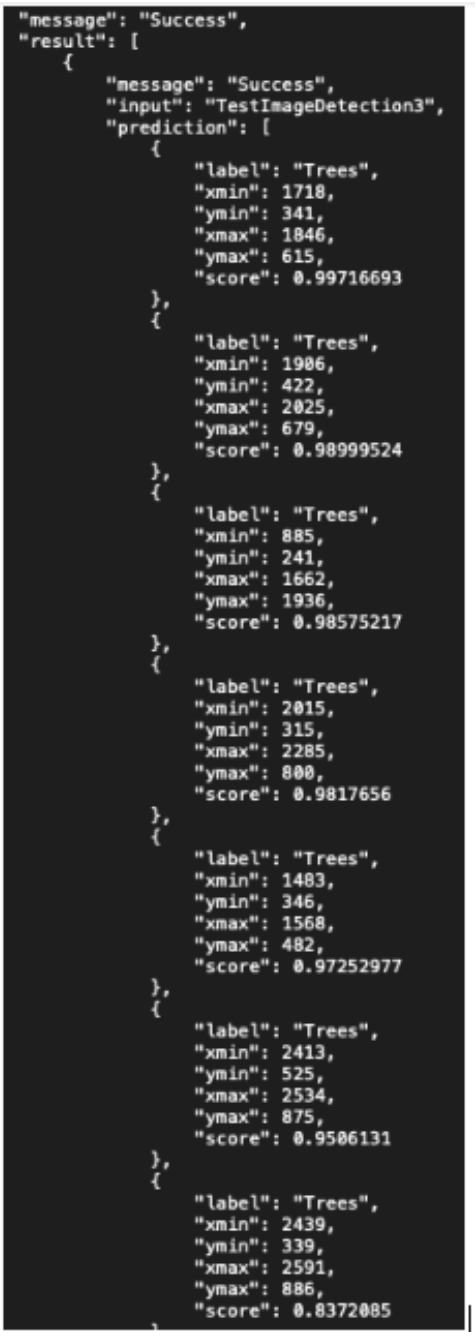
Figure 5.3: Result on self captured drone image.

5.1 Critical discussion

So far, the proposed solution achieves reasonable detection results on images of varying types, including those from drone, street view and camera images. Certain scenarios exists where deviation from the expected results are witnessed however these can be contributed to limited dataset size, fewer training steps due to the lack of powerful machine, incomplete annotation of data and the similarity between new data with the data used for the training of preexisting model.

The project does indeed fulfill its requirements of tree detection and count using two different models. A rightful comparison between them exhibits the stark difference in accuracy and precision of 77 % and 93 %, respectively, which can be attributed mainly to the category of the models used. For instance, the CNN hosted on NanoNets specifically employs a model trained on hundred and thousand

of aerial imagery while Resnet50, although a very large model, is trained on images found in the MS COCO dataset, which are exhaustive but lack aerial imagery. Hence transfer learning is more effective in the case of cloud model. An evaluative criteria of the difference between actual and estimated count is used along with using images of different dimensions and types. During the course of implementation, the input was changed from an image to a video whereby the additional task of frames extraction and image stitching had to be accommodated in the project's execution pipeline. It was indeed challenging but a lot of help from the open source community and with some hard work, an end product capable of taking a video as input and giving count as an output has been successfully made. As an example, final detection results for Fig. 5.3 are exported in JSON format with the coordinates of bounding boxes and their prediction scores saved as shown in figure. Fig. 5.4.



```
"message": "Success",
"result": [
    {
        "message": "Success",
        "input": "TestImageDetection3",
        "prediction": [
            {
                "label": "Trees",
                "xmin": 1718,
                "ymin": 341,
                "xmax": 1846,
                "ymax": 615,
                "score": 0.99716693
            },
            {
                "label": "Trees",
                "xmin": 1906,
                "ymin": 422,
                "xmax": 2025,
                "ymax": 679,
                "score": 0.98999524
            },
            {
                "label": "Trees",
                "xmin": 885,
                "ymin": 241,
                "xmax": 1662,
                "ymax": 1936,
                "score": 0.98575217
            },
            {
                "label": "Trees",
                "xmin": 2015,
                "ymin": 315,
                "xmax": 2285,
                "ymax": 800,
                "score": 0.9817656
            },
            {
                "label": "Trees",
                "xmin": 1483,
                "ymin": 346,
                "xmax": 1568,
                "ymax": 482,
                "score": 0.97252977
            },
            {
                "label": "Trees",
                "xmin": 2413,
                "ymin": 525,
                "xmax": 2534,
                "ymax": 875,
                "score": 0.9506131
            },
            {
                "label": "Trees",
                "xmin": 2439,
                "ymin": 339,
                "xmax": 2591,
                "ymax": 886,
                "score": 0.8372085
```


Figure 5.4: JSON format output.

Chapter 6

Conclusion

Tree count, at the level of abstraction, solves the problem of counting which is fundamentally what as humans we have strived to do for abundance of objects. The project's objective of detection and counting trees are met using two different solution having variable accuracies. Bunch of trees being classified as a single tree can be solved using semantic segmentation while the model's inability to detect small plants and trees can be improved by enhancing dataset and retraining.

The process of counting trees has been automated and mass plantations can be monitored and counted with the ease of technology. Our drone has a flight time of 13 minutes, during which an area of one hectare can be recorded and the resulting image/video passed to determine tree count. The project started by including plant health in addition to plant count as objectives however the former was unimplemented due to lack of hyper-spectral sensory data. Moreover, flight automation of the drone was proposed but could not be made a part of end product as the budget drone used for tree detection lacked a GPS module which is necessary to program a drone to maneuver itself independently.

The project has excellent scalability in terms of working with different types of aerial imagery captured from varying angles and shall work readily with the introduction of hyper-spectral sensory input. In the future, an interactive user interface as a front end shall be developed along with the automation of flight planning provided with a drone equipped with a GPS module. Additionally, the provision of a GPS can be integrated with the JSON result obtained after detection to robotically tag the trees. Introduction of tree species classification can happen alongside the count to maintain an exhaustive catalog of trees on a real time map.

During the eight months allocated for this project, fundamentals of computer vision, application of linear algebra and new technologies such as Jupyter notebook, Microsoft's Azure Machine Learning Library, Google Colab and OpenCV were learnt and implemented which was a challenging task. Collection of a soundly labeled dataset was a daunting task, and we had to settle for a relatively small dataset due to time constraints. Imposition of deadlines and following through them was a challenge in itself since the hardest thing to do is be your own boss. However with hard work and dedication, I persevered and have produced a complete solution to the proposed problem.

In crux, if answers were simply known and mapping of intricate real world problems onto a binary scale seamless, I would not have taught a machine how to learn to solve a problem i.e. tree count. Instead I would have taught it a certain set of rules to abide by, which shall always result in a solution but as evident, that's not the case. My ability to identify and formulate a real world problem into small, dividable and manageable chunks of practical work that a computer can process shall remain invaluable for the rest of my life.

Bibliography