TREE DETECTION AND CLASSIFICATION THROUGH AERIAL IMAGERY USING DEEP NEURAL NETWORKS

By

PC Naiha Mubashir
NC Mubashir Ilyas
NC S.M Umer Latif
NC Muhammad Haris

Submitted to the Faculty of Electrical Engineering,

Military College of Signals, National University of Sciences and Technology, Islamabad

in partial fulfillment for the requirements of a B.E. Degree in

Electrical(Telecom) Engineering

MAY 2019

ABSTRACT

Automatic detection and classification of trees by using remotely analyzed information have been a dream of the many scientists, and land use administrators. The motivation for this problem comes from pollen tree excavation issue, automated 3D town modeling, urban planning and forestation, within which such information is employed to come up with the models.

Here, we offer an automatic methodology for individual tree detection and classification through aerial imagery using unmanned aerial vehicles (UAV), which is a rapidly evolving, cost effective and economical technology.

Firstly, the model is trained for the purpose of tree detection per image pixel by assigning a {tree, non-tree} label to each pixel in an aerial image. Afterwards, the output is refined into clean segmented image based upon which, we implement pattern matching to locate the separable tree crowns, which are then classified on the basis of tree species type with our algorithm.

We have verified the algorithm on many gathered aerial pictures across varied zones of a district and have confirmed excellent quality results with a good scalability of our proposed methodology. In contrast, most of formerly done work used costly hardware like multispectral images for tree detection and classification. Thus, our proposed technique has the potential to classify individual trees in an exceedingly cost-effective manner. This will be a usable tool for several forest researchers, managements, and also for the concerned government bodies to detect and excavate pollen trees, to fight with this seasonal pollen allergy war.

CERTIFICATE FOR CORRECTNESS AND APPROVAL

This officially states that the thesis work contained in this report

"Tree Detection and Classification through Aerial Imagery using Deep Neural Networks"

Is carried out by:

Naiha Mubashir, Mubashir Ilyas, S.M Umer Latif and Muhammad Haris

under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelors of Electrical (Telecomm) Engineering from National University of Sciences and Technology (NUST).

Approved By:

_____

Dr. Adil Masood Siddiqui

EE Department

Military College of Signals, NUST

DATED: May, 2019

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent

To our Faculty, without whose unflinching support and cooperation,

a work of this magnitude would not have been possible.

And our Parents for their support.

ACKNOWLEDGEMENTS

We would like to thank Allah Almighty for His incessant blessings which have been bestowed upon us. Whatever we have achieved, we owe it to Him, in totality.

We are also thankful to our families for their continuous moral support which makes us what we are.

We are extremely grateful to our project supervisor Dr. Adil Masood Siddiqui and our co-supervisor Dr. Hasnat Khurshid from MCS, who in addition to providing us with valuable technical help and guidance also provided us moral support and encouraged us throughout the development of the project.

We are highly thankful to all of our teachers and staff of MCS who supported and guided us throughout our course work. Their knowledge, guidance and training enabled us to carry out this whole work.

Finally, we are grateful to the faculty of Electrical(Telecommunication) Department of the Military College of Signals, NUST.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

UAV	Unmanned Aerial Vehicle
NCS	Neural Compute Stick
LIDAR	Light Detection and Ranging
CNN	Convolutional Neural Network
MATLAB	Matrix Laboratory

INTRODUCTION

1 INTRODUCTION

1.1 Overview

Tree Detection and Classification through Aerial imagery with the help of Unmanned Aerial Vehicle (UAV) is an innovative idea put forward for the proper detection as well as the classification of trees within a defined region. In this prototype, aerial imagery of a well-defined region is gathered with the help of drone for the purpose of data set formation, which is used for the training of pixel-level classifier for assigning a {tree, non-tree} label tag to every individual pixel within the aerial image. The trained model is then advanced by a deep neural network based algorithm which is further used to implement pattern matching to locate the separable tree crowns, which are then classified on the basis of tree types with our algorithm, based on the platform of deep neural networks.

1.2 Problem Statement

Tree detection and species classification is an extensive challenge. Since tree and plants are possessed with high entropy. The detection technique and classification procedures were all primitively based upon the satellite imagery, for instance, LiDAR, Hyperspectral, Multispectral, etc. which are all quite expensive and their accuracies are not relatively satisfactory. The available dataset and in practice machine learning frameworks are complex and convoluted, which in the result are not producing significant or effective outcomes.

Furthermore, with the rapidly increasing deforestation of woodland and enhancing global warming has a virulent effect on the environment of the whole world which is the consequence of non-monitoring condition of the forest, urban and rural areas, or even the generated reports of these areas are not being maintained to prevent all these factors that are affecting the environment.

Lastly, efficient 3D modeling and map-making of cities are not that feasible or achievable, if satellite imagery is considered.

1.3 Approach

Our, proposition is a framework and an enterprise that acquire the automated approach to distinct tree detection and classification with the help of optical-aerial imagery using unmanned aerial vehicles (UAV), which is a rapidly evolving and cost-effective technology, and with our developed and designed deep learning algorithms which help us to perform all these tasks with great and comprehensive accuracy and precision.

Additionally, we allow to observe and analyze the environment of the forests, urban or rural areas by monitoring and generating a well-structured report for forest-woodland department and city development for management authorities, which subsequently stops the factors of forest illegal chopping, cutting and smuggling etc.

1.4 Scope

The project finds its scope in the forestation department and all other government bodies working for the reduction of illegal tree cutting, smuggling and global warming. Considering the innovation side of this prototype, not just the forestation departments but also the bodies working for the pollen tree excavation projects like CDA (Child Development Association) and PMD (Pakistan Meteorological Department) can take benefits from this cost effective and rapid tree detection algorithm.

1.5 Aim & Objectives

1.5.1 Research Objectives

- To attain an automated methodology for the detection and classification of trees in a region from aerial imagery obtained by UAV (unmanned aerial vehicle)
- To design and formulate a neural network framework with the help of structural algorithm of machine learning and implication of pixel-level classifier and utilize it for objects that have high entropy, for instance, trees, grass, river, etc. that are natural objects.

- To attain an automated methodology for the detection and classification of trees in a region from aerial imagery obtained by UAV (unmanned aerial vehicle)
- To design and formulate a neural network framework with the help of structural algorithm of machine learning and implication of pixel-level classifier and utilize it for objects that have high entropy, for instance, trees, grass, river, etc. that are natural objects.

1.5.2 Academic Objectives

- Working in the field of Image Processing
- Programming Skills (Practice on Python language)
- Aerial data acquisition with the help of DJI Phantom 4 pro
- Deep Neural Network platform
- Machine Learning for classification of trees

1.5.3 Commercial Objectives

- Designing of mobile application and online website for generalized density based tree map report generation.

1.5.4 Other Objectives

- To boost the work of forestation and meteorological departments.

With this project, we wish to integrate our academic knowledge with practicality to achieve further understanding and polish our skills in all the fields as mentioned above.

1.5.5 Organization

This document is divided into five main sections, including:

- The first section of the thesis lays the abstract which describes the main details of our research idea, followed by the introduction section which specifies the problem statement, approach, scope, and objectives.
- The second section summarizes the literature about the various resources read online regarding the project and the previous research on the topic.
- The third section emphasizes on the design and development part which illustrates the flow diagrams of different steps involved in the project as well as the description of main modules.
- The fourth section is the analysis and evaluation part which gives the detail of results obtained from deep neural network algorithms.
- The fifth section compromises of the future work, further improvements and points out the additional developments which can be made to enhance the scope of the project.
- In Appendix A we have added the Synopsis document of the project.
- Appendix B contains the code for tree detection
- And Appendix C contains the code for tree classification.

LITERATURE REVIEW

2 LITERATURE REVIEW

The literature available for this project is explained below:

2.1 Background Study

2.1.1 Tree Detection

Individual detection of the tree crown using remotely recognized data plays a large role in the monitoring and city planning purposes. There are many methods generated in the past for the purpose of tree crown detection. The research into tree detection with the help of digital and aerial imagery dates back to the mid of the 1980s. One of the initial examples of work on tree detection was the use of the Vision Expert System that was designed and developed by Pinz in the year 1991. He was able to detect and locate the tree crown's center by examining the local brightness maxima in the acquired imagery. Later on, other algorithms such as region growing were also designed and introduced for the tree detection and crown radius size measurements purpose.

In current years, Light Detection and Ranging (LiDAR) and Hyperspectral imagery information have appeared as the prime source for the detection of trees with the basic assumption that the top of tree crown is positioned at the point with the extreme radiometric values which decreases by moving nearest to the border of tree crown.

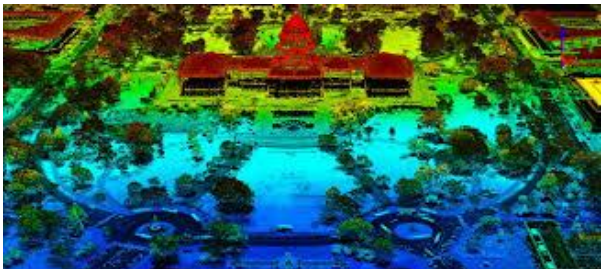

Figure 2-1 LiDAR Imagery Data

Figure 2-2 Hyperspectral Imagery Data Acquisition

In the image domain, typical algorithms for tree crown detection includes local maximum filtering, image binarization or thresholding, and template matching.

2.1.2 Tree Classification

Spatially categorical information on tree species arrangement offers valuable data for environment preservationists as well as for metropolitan and forest managers and is required very frequently over large altitudinal ranges. Studies that included remote sensing data for the classification of the tree species and their mapping purpose reaches back to a time span of numerous years. Our research shows that the quantity of studies concentrating on the classification of tree species has raised over the last few years and various approaches have been offered for numerous types of sensors. However, these researches were only based on the data acquired with the LiDAR sensors or with the help of multispectral and hyperspectral imaging techniques.

All these methodologies for tree classification focused only on the following criteria:

- The algorithm must be able to differentiate between at least two tree species.
- The research must not focus on the broader forest tree variety.
- The algorithm must mainly consider the presence/absence of the tree species within any geographical region.

2.2 Existing Literature

[1] TREE DETECTION FROM AERIAL IMAGERY (Lin Yang, Xiaqing Wu, Emil Praun, Xiaoxu Ma).

This paper gives the overview of the generalized methodology used for tree detection.

The whole methodology is divided into two phases. During the first phase, a pixel level classifier is trained for assigning label of {tree and non-tree} regions within the aerial image. This assignment is done on the basis of the features considered for tree detection within the acquired image. Afterwards, a segmented image is generated which his passed on to the second stage of the methodology.

During the second phase, a set of templates are considered for the purpose of correlation with the output segmented images of the first stage to locate and the tree crowns. A correlation score is generated on the basis of these outputs. The images with score above the correlation score are considered whereas the ones with lower score are discarded.

This method as compared to the previous ones, requires only RGB channels of the aerial imagery for the detection of tree with up to 90 % of precision level. Also, the training procedure for this methodology is open for any type of features and data type thus, this method can be easily integrated with the previously existing methods to boost up their performance levels.

[2] Automatic classification of trees using a UAV onboard camera

and deep learning (Masanori Onishi, Takeshi Ise)

This paper proposes an approach for the classification of tree species on the basis of remotely sensed data that is acquired from aerial imagery with the help of Unmanned Aerial Vehicle (UAV). With the help of UAV and deep learning algorithms, a system is constructed for the automatic classification of tree on the basis of their species type.

In this method, the UAV imagery is segmented into distinct tree crowns on which deep learning algorithms are applied as a result of which, 7 tree types with an accuracy of about 89% have been achieved. This performance is notable because of using easily available digital RGB images and publicly available package for deep learning. In contrast, most of previous studies used expensive hardware such as multispectral imagers to improve performance. In the matter of spatial scale, this method of using a UAV can be limited more than previous method using airborne. But low-cost and easy-to-use feature of UAVs can enable the periodic monitoring. Thus, machine vision system will be a cost-effective and usable tool for forest remote sensing.

DESIGN AND DEVELOPMENT

3 DESIGN AND DEVELOPMENT

3.1 Preliminary Design

The details for the design are given as below:

3.1.1 Technical Specification

The project consists of the following modules:

3.1.1.1 Hardware

- DJI Phantom 4 pro
- Intel Movidius

3.1.1.1.1 DJI Phantom 4 pro

The DJI phantom 4 pro is an intelligent and easy to operate drone with attached gimbal that has cutting-edge 4K camera. The gimbal provides 3-axis stabilization. The drone camera provides high resolution aerial images that encompasses great details of crown size information. The drone was used to acquire dataset.

Figure 3-1 DJI Phantom 4 pro

3.1.1.1.2 Intel Movidius Neural Compute Stick (NCS)

The intel NCS is an Intel movidius visual processing unit embedded on a USB. It is a low-powered chip bringing visual intelligence in thousands of devices. The stick was used to boost neural computations and calculations, increasing overall performance.

Figure 3-2 Intel Movidius NCS

3.1.1.2 Software

- DJI Go 4
- Pix4d Mapping
- Python

3.1.1.2.1 DJI Go 4

It is a mobile device application made for DJI drones by DJI manufacturer. The drone is controlled using this application. The mobile device is mounted on drone's remote controller. All the operations that a drone performs are done through this application.

Figure 3-3 Interface of DJI Go 4

3.1.1.2.2 Pix4d Mapping

It is a mobile application for automated flight control of drone and automated image capturing by giving it the proper constraints and location. Imagery can be taken of a defined area with drone following the desired path just by drawing path lines on app, measured from a fixed desired altitude. The captured images are then processed into desktop version of pix4d for image stitching and generating ortho-maps.

Figure 3-4 Mobile application and PC interface of pix4d

3.1.1.2.3 Python IDE

Python is an interpreted, advanced, general-purpose programming language. It is best for developing machine learning algorithms and working on neural networks. It makes these tasks easier as it has powerful programming modules such as tensorflow, numpy, scikit, opencv.

3.2 Design Requirements and Specifications

Tree detection and classification is a state-of-the-art platform that uses deep neural networks to process aerial images of an area to detect the number of trees in that area and then classify the tree species accordingly.

The project uses drone for dataset collection, which is then inputed to the neural network models. The dataset was divided in 10:90 ratios for testing and training of created models.

In our project mainly 4 python modules were used which are,

- Tensorflow (Keras)
- Numpy
- Opencv
- Pandas

3.3 Methodology

The methodology is divided into 5 blocks which are described in detail as follows:

3.3.1 Google Earth Imagery

With the help of **GOOGLE EARTH PRO**, we have obtained the whole satellite imagery of our campus and other areas and have marked all the trees present within by using the **GOOGLE LENS** and with the slight knowledge about tree types.

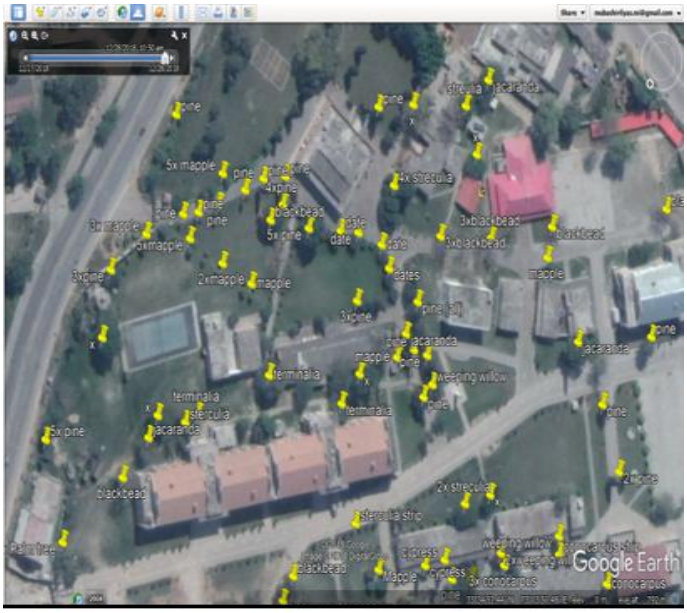

Figure 3-5 Google Earth Imagery

3.3.2 Ground Resolution

For this task we took the imagery of our laptop with known dimensions at different height levels and then by using MATLAB commands, we have calculated the number of pixels that covered the Laptop area and obtained our results as shown in table 3-1.

Table 3-1 Ground Resolution

Height of Drone (Feet)	Width (Pixels)	Length (Pixels)	Pixel Per Inch
30	100	150	101
50	61	93	38
70	40	63	17
100	27	41	7

Dimensions of laptop: 9.9 x 15 inches

3.3.3 Data Set Acquisition

For the purpose of data set formation, we have covered aerial imagery with the help of our drone of various areas of Rawalpindi and Islamabad which are mentioned as follows;

- Military College of Signals
- Orchids of Sargodha District
- Ankara Park, Islamabad
- Graveyard H-8, Islamabad
- F-9, Islamabad

Some of the aerial imagery data acquired with our air drone is shown here;

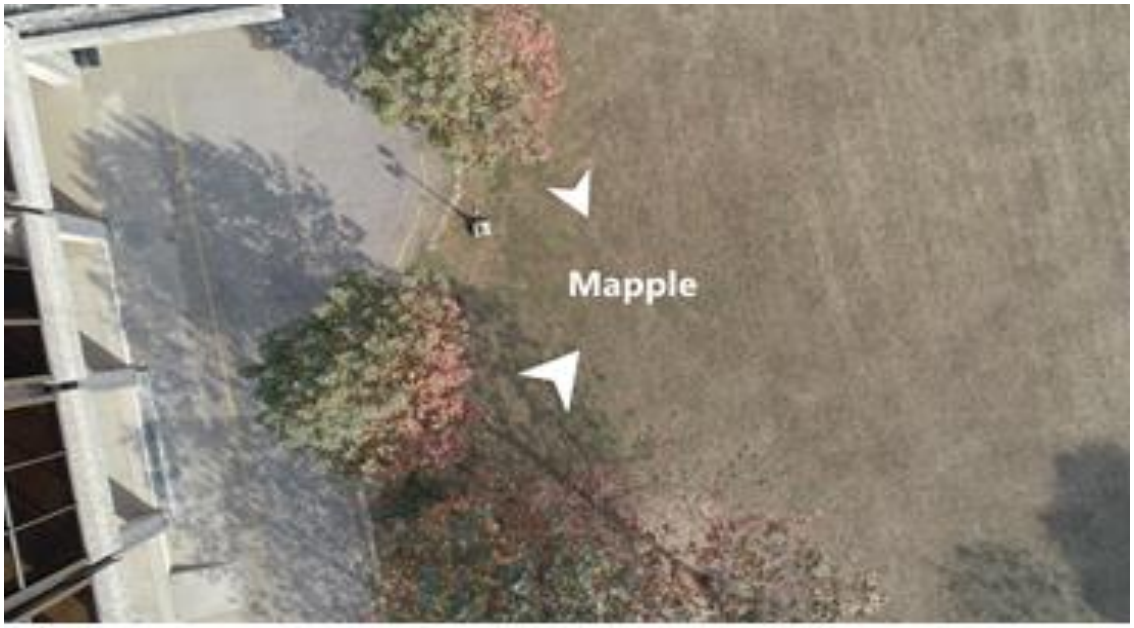

Figure 3-6 Data Set Imagery no. 01

Figure 3-7 Data Set Imagery no.02

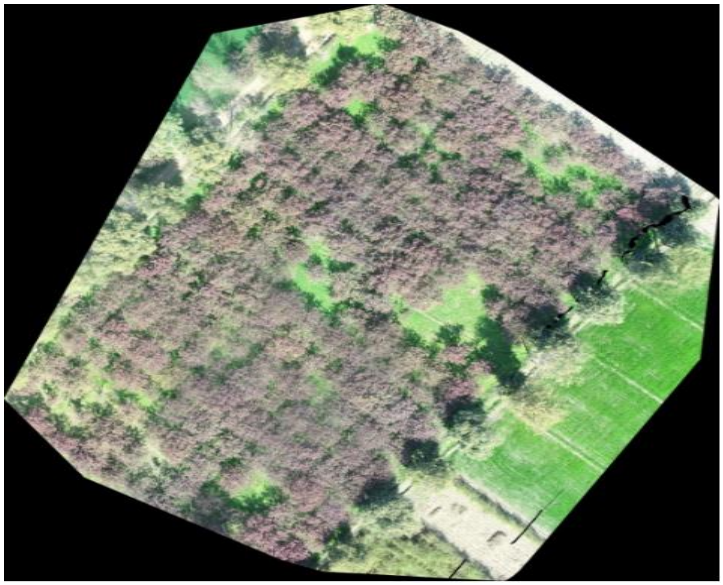

Figure 3-8 Data Set Imagery of Orchids

3.3.4 Pix4D mapping Software

After the task of aerial imagery data acquisition, we move towards the step of image stitching. We have used pix4d app for pre-processing as well as mapping and stitching of the imagery taken. Also its android app was used for automated flights of drone and taking automated imagery of the orchids by giving it the proper constraints and location.

By giving the area to be covered, the height and then by randomly selecting the lines of the drone flight by adjusting them with our fingers according to our requirement we fed the imagery data to the pix4d mapper windows software for the image stitching and creation of ortho-maps.

Figure 3-9 Image Stitching with PIX4D software

3.3.5 Tree Detection

The following section will elaborate the 4th block of our model i.e. used codes and algorithm along with the employed techniques throughout the framework designing:

- In the detection part, first of all we run the feature extractor code that will extract each pixel's value from the whole image and will save it in form of numpy array of file format with extension of ".npy". The saved array named with "feature.npy" will be call upon every time for it utilization in the forthcoming designed algorithm.
- Secondly, we run the label accumulator code which then extract the labels content from the provided ground truth of the image and saved it in corresponding position of the pixels in another array of "label.npy". This array contains the whole pixel's information of whether it represents the "non-tree" or "tree" part of the aerial image.
- Now, we run the model development and training code which is combined together in a single code file. This file will develop the training parameters and the structure of the deep learning model. The parameters will be trained on the basis of the input pixel's values from feature vector as well as their corresponding labels from the label vector. Both the vectors, feature and label are called by this code file. This code will also define the optimizer, loss function, metrics, batch size, epoch, model approach, activation function and all the values that are essential to design the deep neural network on complete foundation. This code file also saves the model-architecture along with its weight and graphs-flow of trained parameters in ".h5py" format.
- The model will be saved as "model.h5py", in an appropriate format which will then be utilized by prediction code file for making predictable output on the basis of input features of the image file, which on the basis of foundation weights and trained parameters makes the inference. The provided inference of the prediction code file will then be viewed as an output image in the form of {one-zero} values that can be analyzed in the form of an array.

- The weights and the accuracy of the model can be further enhanced by running the file code which is designed for model training. This code file will then be re-optimized by the weights and training parameters, without changing the architecture of the model.
- Multiple features are tested upon the provision of the same corresponding labels. The model which is tested and checked by accuracy parameters is then managed to be deployed on the untested and unlabeled data version, which provides the prediction score on the basis of the model training and architectural design of the network.
- The metrics and precision factor is set as "accuracy", which will then be checked upon the completion of each epoch. The memory utilization of the model training and development will take about the 10 percentage of the total data to fully organize the network. The memory utilization is adjusted by changing the parameters and the number of layers

3.3.5.1 Features for Tree Detection

We have used the following features with given specifications as below:

3.3.5.1.1 Spectral Features

- Red, Green, Blue color features which are of the same size as the original image.
- Hue, Value and Saturation frames which are also same as size of the image.
- Light frame from the "LAB"-format along with A (combination of green and red frames) and B (combination of blue and yellow) textural frames.
- Likewise, we used about 9 spectral features for the model development and all are calculated for each image file.

3.3.5.1.2 Spatial Features

The texture feature is uses the filter type of "Gabor-form" with different window size, standard deviations (S.D) and angles, while offset and other parameters are of default value.

- Texture_1: with window of (9, 9) and S.D of 3.5 and angle of pi/14
- Texture_2: with window of (9, 9) and S.D of 4.0 and angle of pi/24
- Texture_3: with window of (13, 13) and S.D of 5.0 and angle of pi/20
- Texture_4: with window of (17, 17) and S.D of 4.0 and angle of pi/16
- Texture_5: with window of (17, 17) and S.D of 3.5 and angle of pi/10

All the textures are generated and observed after the hit and trial version of experiment to better visualize the texture-analysis.

The spatial and spectral features will be extracted per pixel with the help of our algorithm and a vectorized feature vector matrix will be generated on the basis of the features selected.

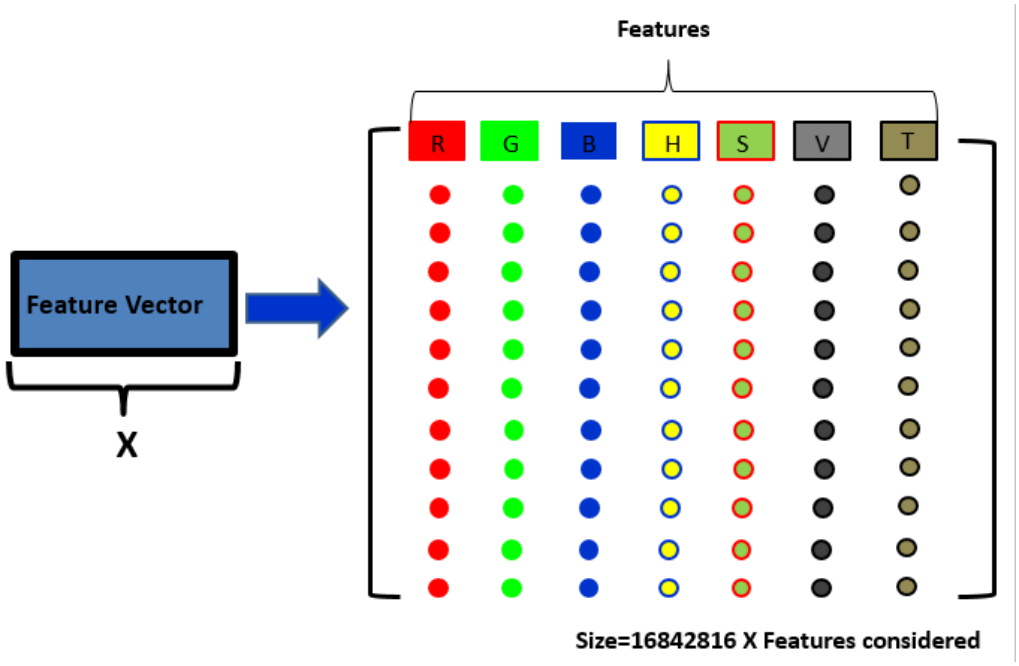

Figure 3-10 Feature Vector

3.3.5.2 Labels for Tree Detection

Ground truth marking is done by making small circles of equal radii on all the tree crowns. Afterwards, the labels are simply extracted from the created ground truths and saved in array file of appropriate format to recall on its usage.

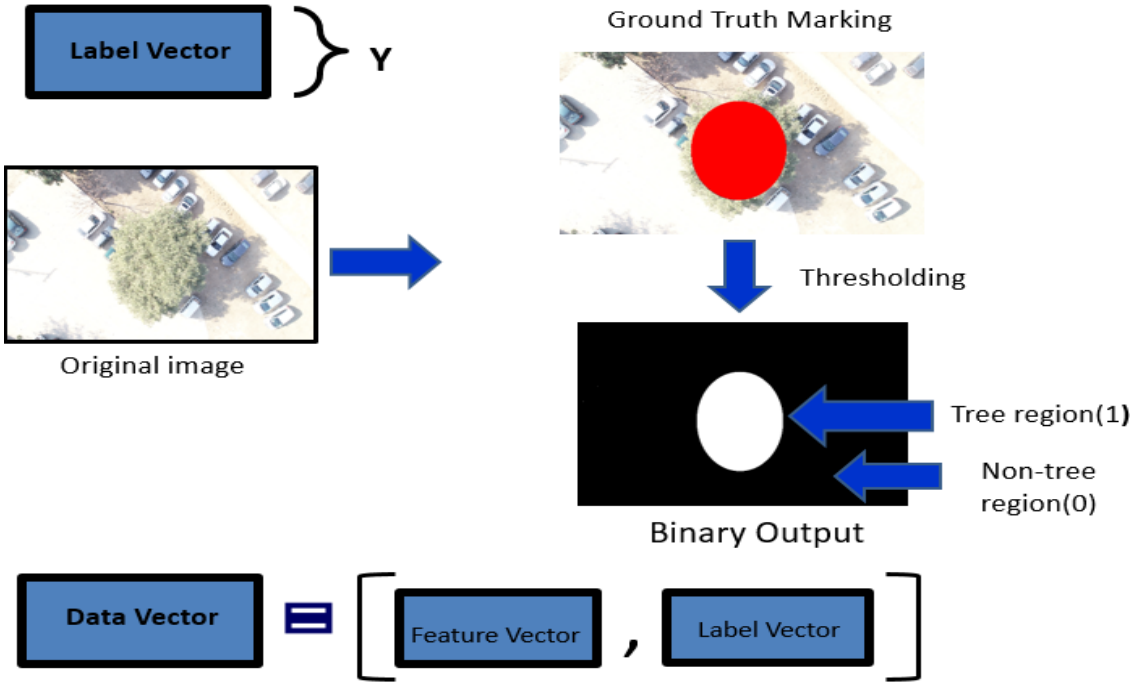

Figure 3-11 Label Vector Formation

3.3.5.3 Model Description

The model is composed of the following parameters as described below:

- Model = Sequential (Keras library implementation)
- 3 layered model
- 1st layer with 100 hidden units
- 2nd layer with 50 hidden units
- 3rd layer with 50 hidden units
- 4th layer with 20 hidden units
- 5th layer with 20 hidden units
- And last layer contains only contain one neuron

- The initial layering contains of “Relu” function as activation function
- While the last layer contains “Sigmoid” function as activation function
- Model compilation contain loss, optimizer and measuring metrics
- Loss = binary cross-entropy
- Optimizer = Adam
- Metrics= Accuracy
- Batch size of 100000
- Evaluation is also performed on same measuring metrics

The detection is performed by the division of the dataset in ratio of 90 percent for training & development while for testing & validating with 10 percent of the data. This division of our given dataset will lead to achieve the accuracy of approx. 91 percent on multiple testing versions.

3.3.6 Tree Classification

This is the final block of our model where the tree classification on the basis of species is completed.

- In the classification portion, we run the binary-image extractor code which will take a binary image from the ground truth of the provided image (whole orthomosaic-map image). After that it will multiply it smartly with original map image that will leave all the sections of the image with zero value except the part that contains trees that are already marked in the ground truth making procedure. The product image will then be saved for further processing.
- Secondly, we will run the circular tree image extractor that will crop out the image in the form of circles from the previous product of binary and original image and then save each cropped image with the name of it’s appropriate class and gives it a suitable range number uniquely attached to it. This circular cropping procedure will make a comprehensive dataset for classification purpose in forthcoming method.

- After that, we run the code file of label extractor which will then get the labels from the cropped images of the classified dataset. This will then be saved in the form of an array that will then be further utilized for model training.
- Now, we will run the model development and training code which is combined together in a single code file and given in the Appendix C of this documentation. This file will develop the training parameters and the structure of the convolutional neural model. These parameters will be trained and optimized on the basis of the input image from the classified dataset as well as their corresponding labels from the label vector. This code will also define the optimizer, loss function, metrics, batch size, epoch, model approach, activation function and all the values that are essential to design the convolutional neural network on complete foundation. This code file also saves the model-architecture along with its weight and graphs-flow of trained parameters in ".h5py" format.
- The model file will be saved as "model.h5py" in an appropriate format, which will then be utilized by prediction code file for making predictable output on the basis of inputted image files, which on the foundation weights and trained parameters makes the inference. The provided inference of the prediction code file will then be viewed as individual prediction vector that will be comprised of some probability score and after maximizing the vector we get the desired output value defining the classification of each provided category in the dataset.
- Multiple images of the same dataset are tested upon the provision of the same corresponding labels. The model which is tested and checked by accuracy parameter is then managed to deploy on the untested and unlabeled data version which will provide the prediction score on the basis of the model training and architectural design of the network.
- The metrics and precision factor is set as "accuracy", which then be checked upon the completion of each epoch.

3.3.6.1 Convolutional Neural Network (CNN)

The provided CNN model for the purpose of tree species classification, has following characteristics:

- Model = Convolutional Sequential Model (Keras library)
- Convolutional2D 3-layered model with 1 dense layer
- Image reshaping and resizing is performed for data fitting according to model
- Input shape of 256 x 256 x 1 (grey scaled)
- Batch size of 100
- Kernel size = 3 x 3
- Filters = 40
- Dropout = 0.2
- Pool-size = 2 (Only for Max pooling)
- 1st layer with input size of 256 x 256
- 1st Max pooling2D with pool-size
- 2nd layer with input size of previous output size
- 2nd Max pooling2D with pool-size
- 3rd layer with further downed size input from previous output
- Flattening layer
- Dense layer contains neurons of equal labels of classes
- The initial layering contains of “Relu” function
- While the last layer contains “Softmax” function
- Model compilation contain loss, optimizer and measuring metrics
- Loss = Categorical cross-entropy

- Epoch = 10
- Optimizer = Adam
- Metrics= Accuracy

We still require a large number of the same aerial imagery dataset to avoid the model overfitting. Furthermore, the model generalization task is remained to achieve for better assessment of the previously unexposed dataset tiles to the trained model. Model overfitting causes the weights of the model shifted to the undesired and unsuitable point of accuracy. We still need to gather some large set of aerial imagery with various categories to train the model in unbiased manner. The provided and acquired dataset is not able to fulfill the very demand of comprehensive model training. Though we finally managed to get about the accuracy approx. of 96 % for two classes of tree species at most.

3.4 Block Diagram

This block diagram shows the complete methodology proposed for the purpose of tree detection and classification.

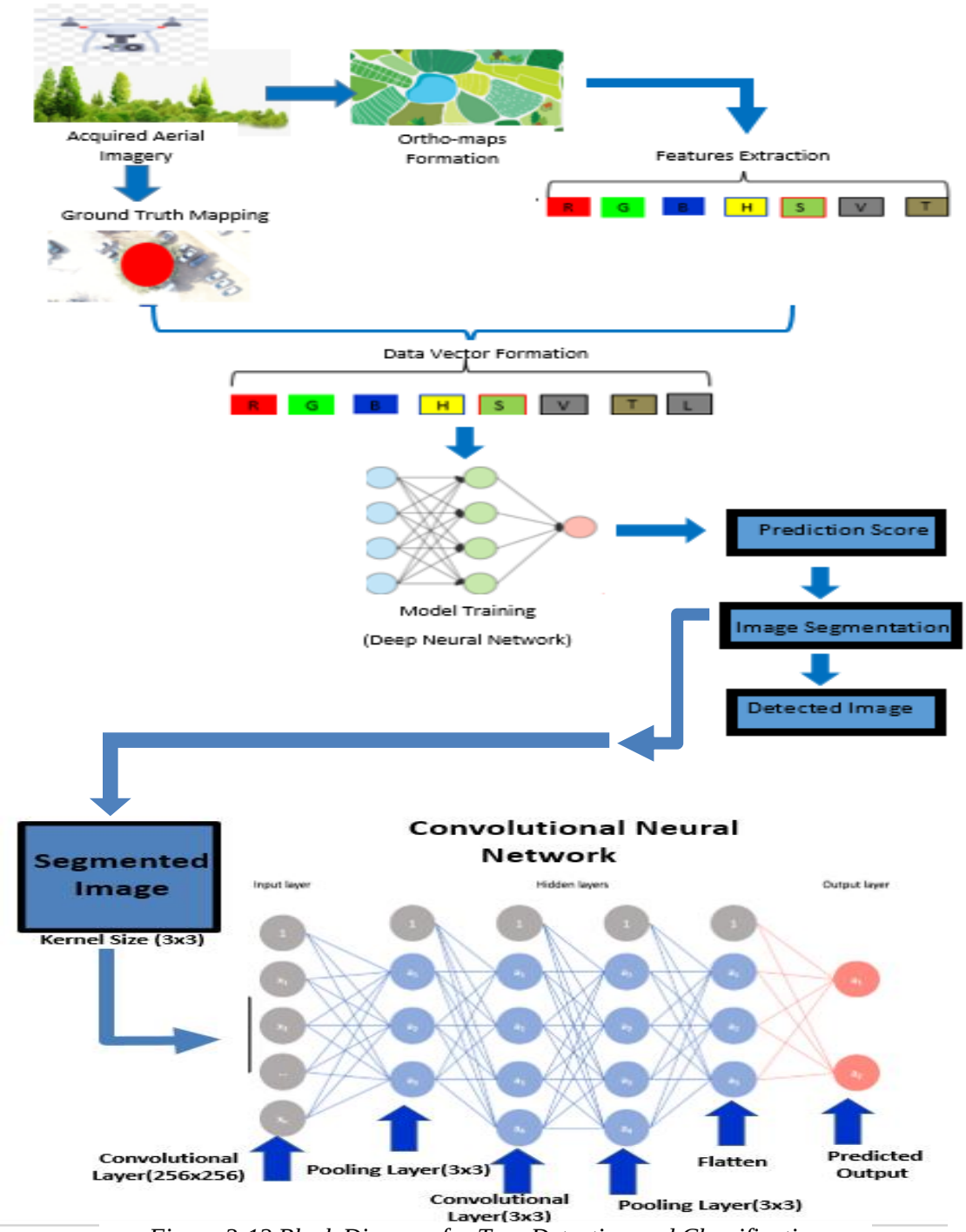

Figure 3-12 Block Diagram for Tree Detection and Classification

ANALYSIS AND EVALUATION

4 PROJECT ANALYSIS AND EVALUATION

Six types of tree species were considered for the analysis and evolution purpose named as follows:

- Paper Mulberry
- Pine tree
- Guava
- Orange
- Maple
- Pine

4.1 Feature Extraction Results

We have extracted 7 types of features per each pixel for the purpose of tree detection.

Figure 4-1 Original Imagery

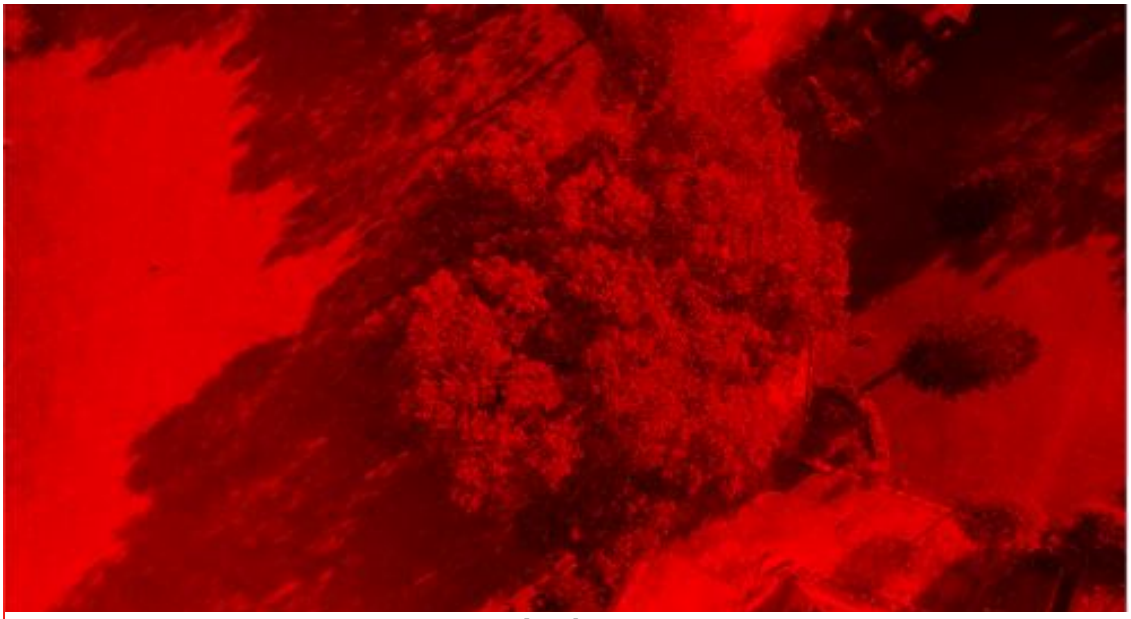

Figure 4-2 Red Color Feature Image

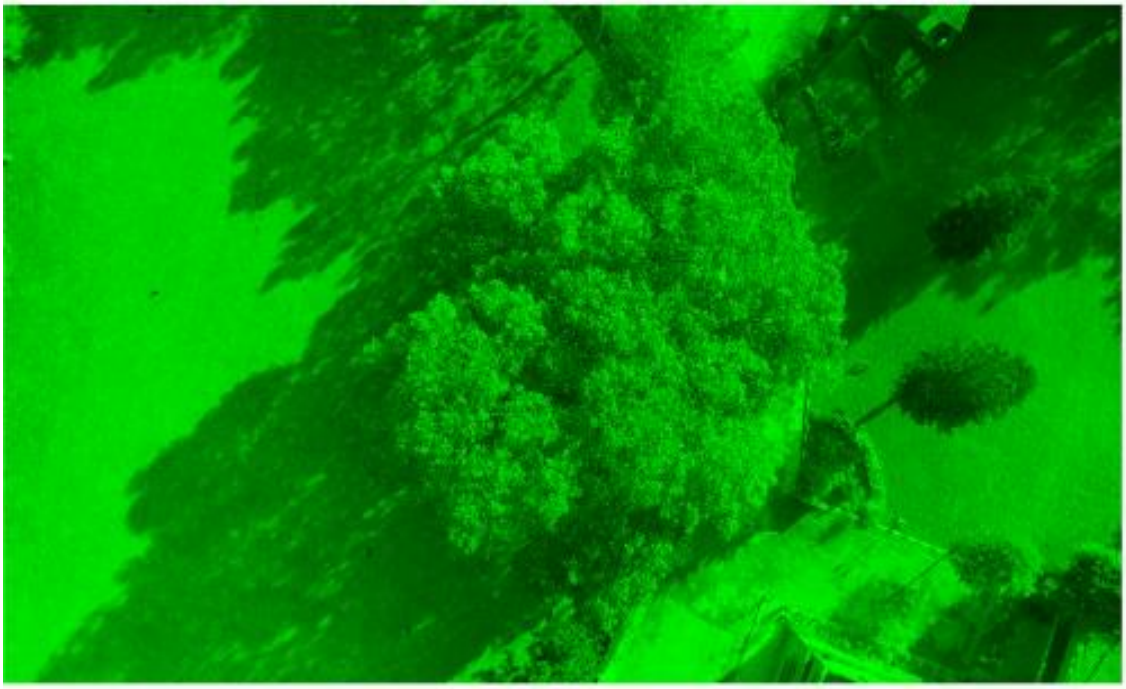

Figure 4-3 Green Color Feature Image

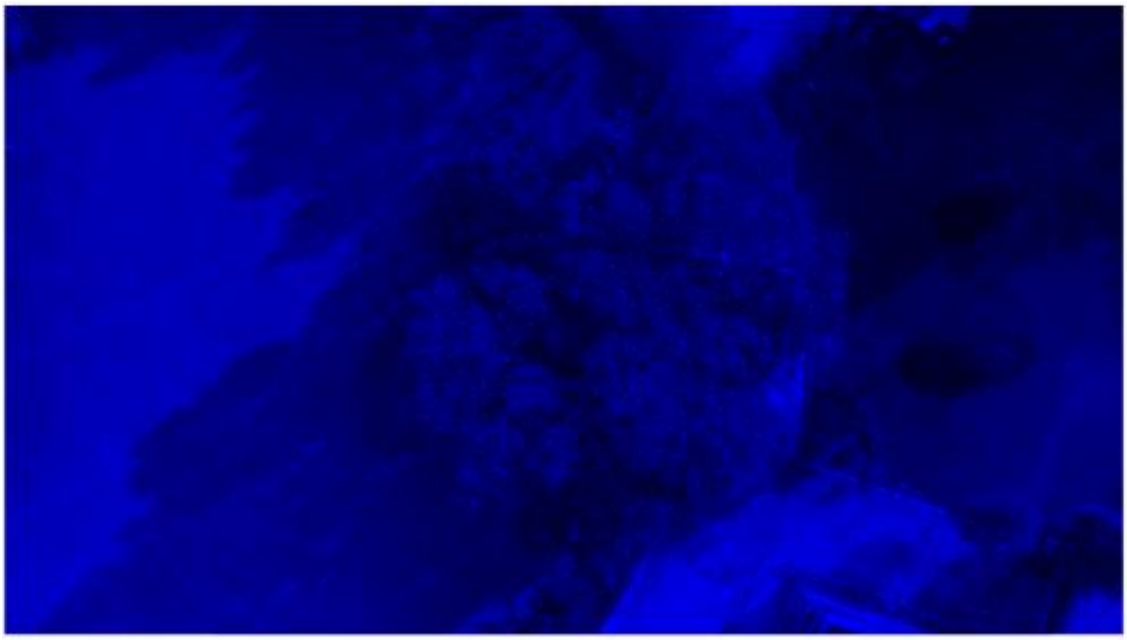

Figure 4-4 Blue Color Feature Image

Figure 4-5 Hue

Figure 4-6 Saturation

Figure 4-7 Value

Figure 4-8 LAB

Figure 4-9 Illumination

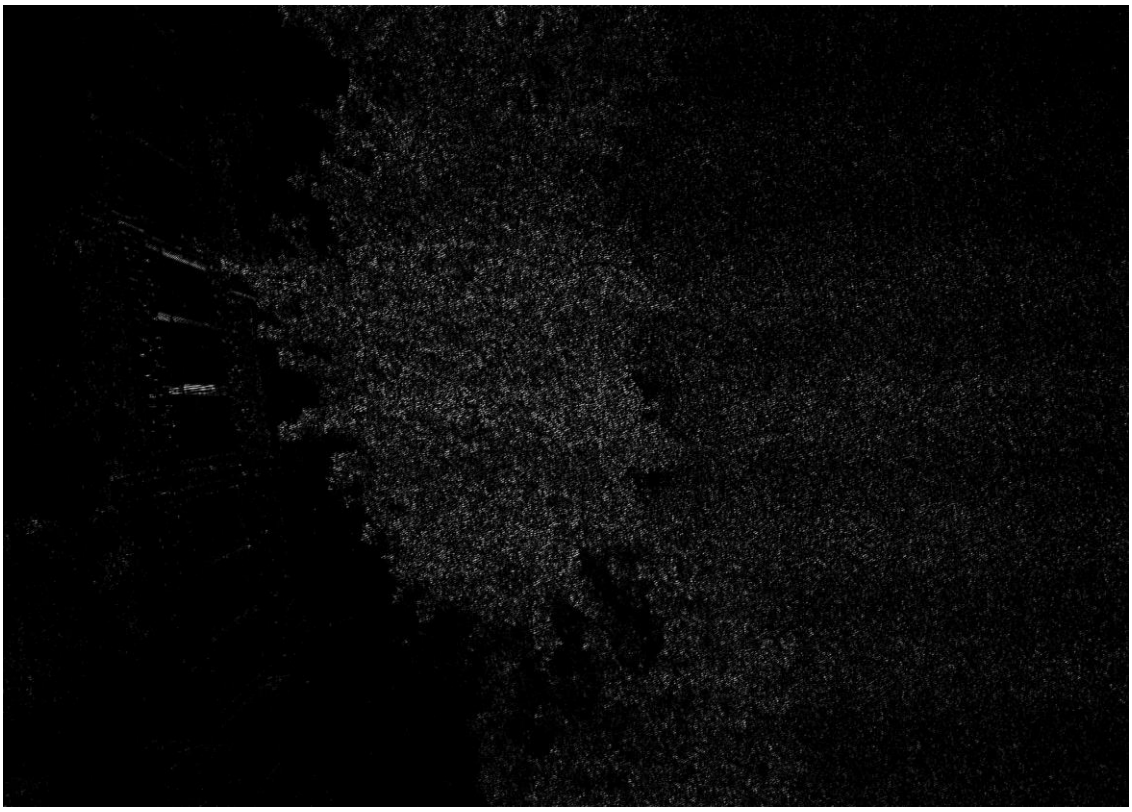

Figure 4-10 Textural Feature

4.2 Results obtained from Ground Truth Generation

For the purpose of label vector formation, we have marked the tree crowns with single colored circle of size equal to the tree crown and then multiplied it with the original image for the formation of binary image. These binary labelled vector with the feature vectors altogether makes the data vector which is passed through the model for the purpose of tree detection.

Figure 4-11 Ground Truth Masking

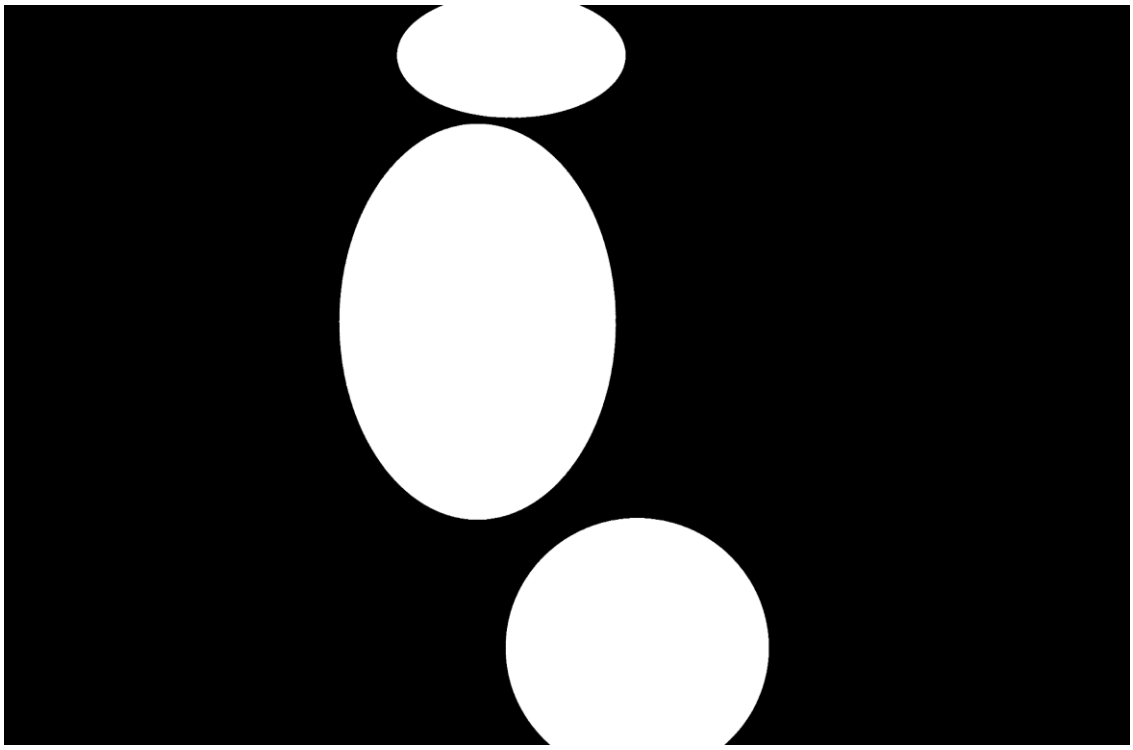

Figure 4-12 Binary Image

4.3 Predicted Output of Tree Detection Model

The data vector is fed to the deep neural network model to get the predicted output image. In this image only the tree area is defined in the form of binary 1 and all the other parts are marked as binary 0. Thus in this way the trees are detected.

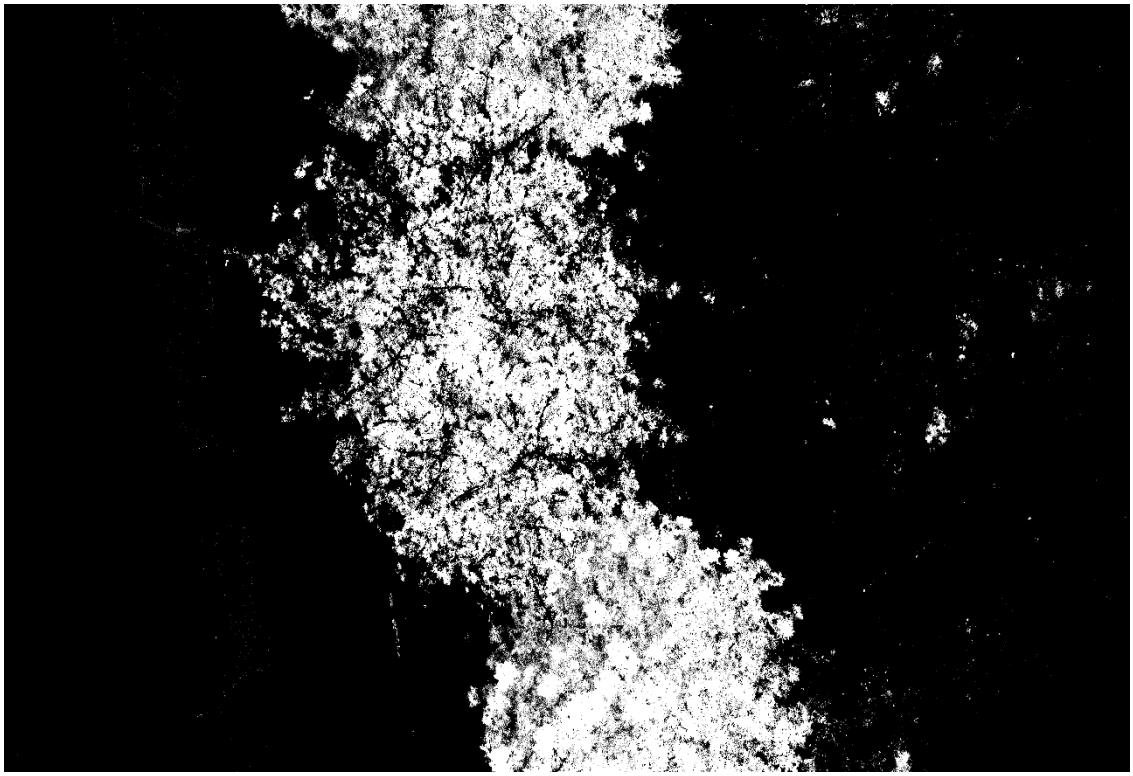

Figure 4-13 Predicted Output

4.4 Predicted Output of Tree Classification Model

The segmented images from the previous feature extraction model are then fed to the block of classification model as discussed previously in chapter 3, the output of which is saved in the form of excel sheet as shown:

	A	B	C	D	E	F
1		0				
2	0	guava				
3	1	guava				
4	2	guava				
5	3	guava				
6	4	orange				
7	5	guava				
8	6	guava				
9	7	guava				
10	8	guava				
11	9	guava				
12	10	guava				
13	11	orange				

Figure 4-14 Predicted Output of Tree Classification Model

CONCLUSION

5 CONCLUSION

5.1 Overview

We propose an automatic and well-defined approach with the help of data acquired from aerial imagery for the prime reason of tree detection and classification. In this model, aerial imagery of a well-defined region is gathered with the help of drone for the formation of data set, which is used for the training of pixel-level classifier for assigning a {tree, non-tree} tag to every individual pixel within the aerial image. The trained model is then advanced by a deep neural network based algorithm which is further used to implement pattern matching to locate the separable tree crowns, which are then classified on the basis of tree types with our algorithm based on the platform of deep neural networks.

The model works on pure imageries and is competent for the detection of trees and their classification on the basis of specie types in enormous sizes. On large data sets, we are able to train the classifier on only 1.0 % of the data while achieving more than 90.0 % accuracy for our designed model.

5.2 Objectives Achieved/Achievements

- Detection of trees and its crown size estimation through pixel classifier
- Multiple feature extraction of the trees (Spatial and Spectral features)
- Classification of various tree species
- Pollen trees determination
- Minimum labor requirement

5.3 Limitations

There are certain limitations associated with the project:

- Accuracy issue due to time constraints for data acquisition.
- Proper crown size detection is not possible with a small amount of gathered aerial imagery.
- The requirement of heavy servers for efficient data processing.
- The success of this whole project depends on the permission of government concerned bodies.

5.4 Future Research

In future we will work on the generation of proper generalized density map based pollen tree reports for the concerned government departments, so that with the help of them, complete pollen tree excavation could be performed.

We will present a whole process in which pollen trees will be excavated with the help of our developed team and in return environment friendly trees will be planted so to fight with the seasonal pollen allergy issues and to help in the reduction of global warming.

FUTURE WORK

6 FUTURE WORK

Tree detection and its classification on the basis of tree species is the basic goal and foundation of this project, but the project has some additional goals which can be achieved through further explorations and practical work.

- Better classification can be done by adding more tree species in the acquired data set.
- Formulation of city infrastructure and 3D city modeling
- Enhancement in the map versions.
- Aerially generated map of the whole globe pointing all the trees and their types.
- Density-based map and generalized tree report generation
- This proposed solution can also be used for the determination of pollen trees and their proper excavation, to fight with the seasonal pollen allergy war.

BIBLIOGRAPHY

7 BIBLIOGRAPHY

[1] A. Rosebrock, "Keras Tutorial: How to get started with Keras, Deep Learning, and Python," 10 September 2018. [Online]. Available: https://www.pyimagesearch.com/2018/09/10/keras-tutorial-how-to-get-started-with-keras-deep-learning-and-python/.

[2] A. Rosebrock, "Real-time object detection with deep learning and OpenCV," 18 September 2017. [Online]. Available: https://www.pyimagesearch.com/2017/09/18/real-time-object-detection-with-deep-learning-and-opencv/.

[3] Tompee, "FF-CNN-Binary-Classification," 13 October 2017. [Online]. Available: https://github.com/tompee26/FF-CNN-Binary-Classification/blob/master/.

[4] Alexandrejaguar, "Scikit-image: Image processing in Python," 1 June 2011. [Online]. Available: https://github.com/scikit-image/scikit-image.

[5] L. J. Q. Y. Ke, " A review of methods for automatic individual tree-crown," International Journal of Remote Sensing, vol. 32, pp. 4725-4747, 2009.

[6] X. W. E. P. X. M. L. Yang, "Tree Detection from Aerial Imagery," Seattle,Washington, 2009.

[7] T. I. M. Onishi, "Automatic classification of trees using a UAV onboard camera and deep learning," 2018.

[8] S. E. Franklin, "Pixel- and object-based multispectral classification of forest tree," Unmanned Vehicle Systems, vol. 6, pp. 195-211, 2018.

APPENDIX

8 APPENDICES

8.1 Appendix A

Synopsis

Extended Title: Tree Detection And Classification Through Aerial Imagery Using Deep Neural Network
Brief Description / Abstract: Automatic detection and classification of trees by using remotely analyzed information have been a dream of the many scientists, and land use administrators. The motivation for this problem comes from pollen tree excavation issue, automated 3D town modeling, urban planning and forestation, within which such information is employed to come up with the models. Here, we offer an automatic methodology for individual tree detection and classification through aerial imagery using unmanned aerial vehicles (UAV), which is a rapidly evolving cost effective and economical technology.
Scope of Work and Deliverable: The project finds its scope in the forestation department and all other government bodies working for the reduction of illegal tree cutting, smuggling and global warming. Considering the innovation side of this prototype, not just the forestation departments but also the bodies working for the pollen tree excavation projects like CDA (Child Development Association) and PMD (Pakistan Meteorological Department) can take benefits from this cost effective and rapid tree detection algorithm.

Objectives of the project:

- Classification of Tree species.
- To determine the density of the type of trees in forest that are on extinction level.
- To provide surveillance through aerial imagery of illegal cutting and smuggling of trees.
- Automated 3D city modeling and urban planning.

Applications:

- Generalized Report Formation
- Pollen Tree Excavation
- 3D City Modeling
- Enhancement in Map Versions
- Forestry and Forest Management
- Wildlife Departments
- A Precise Analysis and Recognition of Vegetation Types and Agricultural Areas

Group Members:

- **Naiha Mubashir (CGPA 3.96) (Syndicate leader)**
- **Mubashir Ilyas (CGPA 2.74)**
- **S.M Umer Latif (CGPA 2.30)**
- **Muhammad Haris (CGPA 3.93)**

Academic Objectives:

Image Processing, Deep learning Algorithms

Hardware Requirements:

UAV(Drone),Intel Movidius Neural Compute Stick

Approval Status

Supervisor Name & Signature

HoD Signature **----------**

R&D SC Record Status **File #_____ Coordinator Signature_____**

8.2 APPENDIX B

Code for Tree Detection Algorithm

- **Feature Extraction and Accumulation**

```
import cv2
import numpy as np
import os, glob
import pandas as pd


path_npy = str(r'/home/darakht/Desktop/detection/dataset/npy/')
path_image = str(r'/home/darakht/Desktop/detection/dataset/im.form/image/')


direc = os.listdir(path_image)
n = 0
for f in direc:
    n = n + 1


def gab(kernel, sd, angle):
    g_kernel = cv2.getGaborKernel(kernel, sd , angle, 1.0, 1.0, 0)
    gabimg = image.copy()
    gabimg = cv2.cvtColor(gabimg, cv2.COLOR_BGR2GRAY)
    tex = cv2.filter2D(gabimg, cv2.CV_8UC3, g_kernel)
```

```
    return tex


for i in range(n):
    i = i + 1
    image = cv2.imread(path_image + r'/image' + str(i)+ r'.png')
    size = image.shape[0]*image.shape[1]

    bimg, gimg, rimg = cv2.split(image)

    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    Himg, Simg, Vimg = cv2.split(hsv)

    lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
    Limg, Aimg, Bimg = cv2.split(lab)

    b = bimg.reshape(size)
    g = gimg.reshape(size)
    r = rimg.reshape(size)
    H = Himg.reshape(size)
    S = Simg.reshape(size)
    V = Vimg.reshape(size)
    L = Himg.reshape(size)
    A = Aimg.reshape(size)
```

```
B = Bimg.reshape(size)
tex1 = gab((9, 9), 3.5, np.pi/14)
tex2 = gab((9, 9), 4.0, np.pi/24)
tex3 = gab((13, 13), 5.0, np.pi/20)
tex4 = gab((17, 17), 4.0, np.pi/16)
tex5 = gab((17, 17), 3.5, np.pi/10)
tex1 = tex1.reshape(size)
tex2 = tex2.reshape(size)
tex3 = tex3.reshape(size)
tex4 = tex4.reshape(size)
tex5 = tex5.reshape(size)

features = np.array([b, g, r, H, S, V, L, A, B, tex1, tex2, tex3, tex4, tex5])
features = features.T
features = features.astype('float32')

#pd.DataFrame(features).to_csv(path_npy + r"/features.csv")
np.save(path_npy + r'/features/features' + str(i) + r'.npy', features)
```

- **Labels Accumulation**

```
import cv2
import numpy as np
import os, glob
```

```
import pandas as pd

path_npy = str(r'/home/darakht/Desktop/detection/dataset/npy/')
path_truth = str(r'/home/darakht/Desktop/detection/dataset/im.form/truth/')

lower= np.array([160,100,100]) #red
upper= np.array([179,255,255])

direc = os.listdir(path_truth)
n = 0
for f in direc:
    n = n + 1

for i in range(n):
    i = i + 1
    truth = cv2.imread(path_truth + r'/truth' + str(i)+ r'.png')
    size = truth.shape[0]*truth.shape[1]

    hsv = cv2.cvtColor(truth, cv2.COLOR_BGR2HSV)
    binn = cv2.inRange(hsv, lower, upper)
    cv2.imwrite(r'/home/darakht/Desktop/range' + str(i) +r'.png', binn)

    labels = binn.reshape(size)
```

```
    labels = labels.astype('float32') / 255.0
    #pd.DataFrame(labels).to_csv(path_npy + r"/labels.csv")

    np.save(path_npy + r'/labels/labels' + str(i)+ r'.npy', labels)
```

- **Model Generation**

```
    from __future__ import absolute_import
    from __future__ import print_function

    from keras.models import Sequential
    from keras.layers import Dense, Activation, Dropout
    from keras.layers import Conv2D, MaxPooling2D, Flatten
    import numpy as np
    import cv2
    import h5py
    import os,glob

    path_features = str(r'/home/darakht/Desktop/detection/dataset/npy/features/')
    path_labels = str(r'/home/darakht/Desktop/detection/dataset/npy/labels/')
    path_model = str(r'/home/darakht/Desktop/detection/models/')

    hidden_units1 = 100
    hidden_units2 = 50
```

```
hidden_units3 = 50
hidden_units4 = 20
hidden_units5 = 10

model = Sequential()

model.add(Dense(hidden_units1, input_dim=14))
model.add(Activation('relu'))

model.add(Dense(hidden_units2))
model.add(Activation('relu'))

model.add(Dense(hidden_units3))
model.add(Activation('relu'))

model.add(Dense(hidden_units4))
model.add(Activation('relu'))

model.add(Dense(hidden_units5))
model.add(Activation('relu'))

model.add(Dense(1))
model.add(Activation('sigmoid'))
```

```
model.summary()
model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])

direc = os.listdir(path_features)
n = 0
for f in direc:
    n = n + 1

for i in range(1, n):
    i = i + 1
    f = np.load(path_features + r'/features'+ str(i) + r'.npy')
    l = np.load(path_labels + r'/labels'+ str(i) + r'.npy')
    model.fit(f, l, epochs=1, batch_size=100000)
    #scores = model.evaluate(f, l)
    #print("\n", r'----------------------------', "\n", r'Evaluation for ' + str(i) + r' instance:'
,
 "\n" ,r'----------------------------')
    #print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
    model.save(path_model + r'/modeldnn.h5')

f = np.load(path_features + r'/features4.npy')
l = np.load(path_labels + r'/labels4.npy')
```

```
scores = model.evaluate(f, l)

print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

- **Prediction Algorithm**

```
 version --2.2

#######################################################################


from keras.models import load_model

import numpy as np

import cv2

import h5py


path_features = str(r'/home/darakht/Desktop/detection/dataset/npy/features/')

path_model = str(r'/home/darakht/Desktop/detection/models/')


model = load_model(path_model + r'/modeldnn.h5')

x = np.load(path_features + r'/features5.npy')


p = model.predict(x)


p = p * 255

p = p.reshape([3078, 5472])

print(p)
```

```
np.save(path_model + r'/pred5.npy', p)

cv2.imwrite(path_model + r'/pred5.png', p)
```

8.3 APPENDIX C

Code for Tree Detection Algorithm

- **Binary Classification**

```
import numpy as np
import cv2
import math

image_path = str(r'<<provide path--binary-true multiplied image>>')
exp = cv2.imread(image_path)
gray = cv2.imread(image_path, 0)

height = exp.shape[0]
width = exp.shape[1]
mask = np.zeros((height,width), np.uint8)

gray_blur = cv2.medianBlur(gray, 13)
gray_lap = cv2.Laplacian(gray_blur, cv2.CV_8UC1, ksize=5)
circles = cv2.HoughCircles(gray_lap, cv2.HOUGH_GRADIENT,

1, 180, param1=50,param2=30, minRadius=90, maxRadius=120)
print(circles)
num = len(circles[0,:])
```

```
def draw_circles(img, circles):
    cimg = cv2.cvtColor(img,cv2.COLOR_GRAY2BGR)
    for i in circles[0,:]:
        cv2.circle(cimg,(i[0],i[1]),i[2],(0,255,0),2)
    return cimg

cimg = draw_circles(gray, circles)

for i, j in zip(circles[0,:], range(num)):
    cv2.circle(mask,(i[0],i[1]),i[2],(255,255,255),thickness=-3)
    masked_data = cv2.bitwise_and(exp, exp, mask=mask)
    x = i[0] - i[2]
    y = i[1] - i[2]
    x = math.ceil(x)
    y = math.ceil(y)
    w = math.ceil(i[2]*2)
    h = math.ceil(i[2]*2)
    crop = masked_data[y:y+h,x:x+w]
    print(x,y,w,h)
    s = 256
    k = abs(s - h)
    l = abs(s - w)
```

```
    a = np.zeros((s ,s ,3), np.uint8)
    a = np.pad(crop, ((0, k), (0, l), (0, 0)), 'constant')
    print(a.shape)
    cv2.imwrite(r'<<provide path>>//orange_' + str(j) + r'.png', a)

cv2.namedWindow('detected circles', cv2.WINDOW_NORMAL)
cv2.imshow('detected circles',cimg)
cv2.waitKey(0)
```

- **Label Data Accumulator**

```
import numpy as np
import cv2
import os, glob
import pandas as pd

path =  str(r'<<provide path>>')
direc = os.listdir(path)
c = 2 # no of classes
i = 0
data = np.zeros([1,256*256])

for f in direc:
    i = i + 1
```

```
labels = np.zeros([i, c])


def datastack(fy):
    img = cv2.imread(path + str(r'/') + fy, 0)
    x = np.reshape(img,[1, 256*256])
    return x


for f in direc:


    title, ext = os.path.splitext(os.path.basename(f))
    title = title.split(r'_')
    n = int(title[1])


    if title[0] == 'orange':
        dx = datastack(f)
        data = np.vstack((data, dx))
        labels[n,0] = 1          # '0' means for orange


    if title[0] == 'guava':
        dx = datastack(f)
        data = np.vstack((data, dx))
        labels[n,1] = 1          # '1' means for guava
```

```
data = np.delete(data, 0, 0) # iterative deletion (first axes)
data = data.astype('float32')
labels = labels.astype('float32')
path1 =  str(r'<<provide path>>')
np.save(path1 + r'/labels.npy', labels)
np.save(path1 + r'/data.npy', data)
```

- **Binary Multiplication**

```
import numpy as np
import cv2

path = str(r'<<provide path>>')
img = cv2.imread(path + r'\org-truth.png')

lower= np.array([170,100,100])
upper= np.array([179,255,255])

hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
bin0 = cv2.inRange(hsv, lower, upper)

cv2.imwrite(path + r'\bin.png', bin0)

b = cv2.imread(path + r'\bin.png')
```

```
x = cv2.imread(path + r'\org.jpg')


b = b / 255.0
z = x * b
z = z.astype(np.uint8)


cv2.imwrite(path + r'\binx.png', z)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- **Convolutional Neural Network Model**

```
#######################################################
## Model
#######################################################


from __future__ import absolute_import
from __future__ import division
from __future__ import print_function


import numpy as np
from keras.models import Sequential
from keras.layers import Activation, Dense, Dropout
```

```
from keras.layers import Conv2D, MaxPooling2D, Flatten
from keras.utils import to_categorical, plot_model
import h5py

path = str(r'<<provide path>>')
x = np.load(path + r'/data.npy')
y = np.load(path + r'/labels.npy')
num_labels = 2
par_in_size = 256
x = np.reshape(x, [-1, par_in_size, par_in_size, 1])
x = x.astype('float32') / 255
input_shape = (par_in_size, par_in_size, 1)

batch_size = 100
kernel_size = 3
pool_size = 2
filters = 40
dropout = 0.2
model = Sequential()
model.add(Conv2D(filters=filters,
                 kernel_size=kernel_size,
                 activation='relu',
                 input_shape=input_shape))
```

```
model.add(MaxPooling2D(pool_size))
model.add(Conv2D(filters=filters,
                 kernel_size=kernel_size,
                 activation='relu'))
model.add(MaxPooling2D(pool_size))
model.add(Conv2D(filters=filters,
                 kernel_size=kernel_size,
                 activation='relu'))
model.add(Flatten())
model.add(Dropout(dropout))
model.add(Dense(num_labels))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.fit(x, y, epochs=10, batch_size=batch_size)
loss, acc = model.evaluate(x, y, batch_size=batch_size)
print("\nTest accuracy: %.1f%%" % (100.0 * acc))
model.save(path + r'/modelccn.h5')
```

- **Output Prediction Model**

```
from __future__ import absolute_import

from __future__ import division

from __future__ import print_function


from keras.models import load_model

import numpy as np

import cv2

import h5py


path = str(r'<<provide path>>')

model = load_model(path + r'/modelcnn.h5')

data = np.load(path + r'/data.npy')

n = 2 # objects

X = np.reshape(X, [-1, 256, 256, 1])

for i in range(n):

    X = data[i,:]

    p = model.predict(X)

    print(r'ans:',p)

    print(r'labeled: ', np.argmax(p))
```