Automated Timber Recognition System

K.H.P.W.L De Silva

169309H

Master of Science in Information Technology

Faculty of Information Technology

University of Moratuwa

Sri Lanka

February 2019

Automated Timber Recognition System

K.H.P.W.L De Silva

169309H

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfilment of Degree of Master of Science in Information Technology.

February 2019

Declaration

I declare that this is my own work and this thesis does not incorporate without acknowledgment any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgment is made in the text.

Also, I at this moment grant to the University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or another medium. I retain the right to use this content in whole or part in future works (such as articles or books).

………………………………. ………………………

K.H.P.W.L De Silva

(Signature of the candidate) Date:

The above candidate has carried out research for the master's thesis under my supervision.

……………………………….. ………………….

Dr. C. D Gamage Date

Signature of the Supervisor

………………………………… …………………..

Mr.Saminda Premaratne Date

Signature of the Co-Supervisor

Acknowledgment

I would like to take this opportunity to express my gratitude and a profound feeling of admiration for my project supervisors. Many thanks go to all those who helped me in this work. My special thanks go to the University of Moratuwa for giving an opportunity to carry out this research project.

I would like to gratefully acknowledge to Dr. Chandana Gamage, Senior Lecturer, Department of Computer Science and Engineering, University of Moratuwa, supervisor of the project, for sharing the experiences and expertise with the project matters throughout the whole duration of this research. I would like to extend my heartfelt gratitude to Mr. Saminda Premaratne, Senior Lecturer, Department of Information Technology, University of Moratuwa, co-supervisor of the project for his continuous guidance and support throughout the research.

I would also like to thank Galle Bussa Timber Corporation, Head Office of Timber Corporation Battaramulla, Thalalla Timber Corporation, Rathmalana Timber Corporation, Ampara Timber Corporation, Saw Mills Moratuwa, Sampath Saw Mill Matara, and Lal de Silva Construction, Matara for supporting us with image data collection and knowledge on different timber species which made this research a success and I like to thank to my friends ,academic and non-academic staff member in department of computer science grateful support for success my research.

I would like to extend my thanks to the team of photographers Mr. Disitha Nethsara, Mr. Thilina Gamage, Mr. Shashi Nilan Amarasinghe and Mr. Sandaruwan Perera for their great support and effort to take 17000 photos with different timber spices at different locations.

I must express my very profound gratitude to my parents, sister, and aunties for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Copyright statement

I hereby grant the University of Moratuwa the right to archive and to make available my thesis or dissertation in whole or part in the University Libraries in all forms of media, subject to the provisions of the current copyright act of Sri Lanka. I retain all proprietary rights, such as patent rights. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Abstract

More than 2,000 different types of wood species can be found from a tropical rain-forest. Generally, in Sri Lanka, out of these 2000, only about 200 varieties are being used by the timber industry today. Trees and its products have been used by society for thousands of years. Timber plays a significant role in many aspects of today's world. As timber is the only considerable building material that is grown, we have a natural inclination that building in timber is good for the environment. Nowadays, the main timber consumers are building constructors, timber fabricators and furniture manufacturers where the need for recognition of timber species is essential. A programmed timber identification system has still not been well established mainly due to the absence of research in this specific area and the difficulty in gaining a wood database. Such a system is highly needed by various industries and people. However, timber identification is an area which is difficult to accomplish easily to meet the market demand. In this study, we present an effective methodology for solving the problem of timber recognition. The proposed system is an automated timber recognition system based on image processing and machine learning. The proposed system is designed to categorize different indigenous timber for Sri Lanka according to the type of wood images we acquire locally. The image processing technique is developed using newly established image processing libraries and texture of timber structures to analyze images. The gray-scale co-occurrence matrix technique and the k- Nearest Neighbor algorithm have been used to extract features and train the data respectively for classification purposes. The proposed system can deliver timber identification within a short period of time, unlike the macroscopic timber detection by removing the necessity for human recognition.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

List of Abbreviations

GLCM	Gray Level Co-Occurrence Matrix
BGLAM	Basic Grey Level Aura Matrix
SPPD	Statistical Properties of Pores Distribution
BGLAM	Basic Grey Level Aura Matrix
KNN	k- Nearest Neighbor
LDA	Linear Discriminant Analysis
Sp-DNDR	Sub-pattern based Discriminative Non-Linear Reduction
MLBP	Multi-Layer neural network Based on the popular back Propagation
ANN	Artificial Neural Network
UD	Undamaged
MD	Moderate Damage
SD	Severe Damage
BP	Back Propagation
ASM	Angular Second Moment
RGB	Red, Green, Blue
SIFT	Scale Invariant Feature Transform
SURF	Speeded up robust features
HSV	Hue, Saturation, and Value
HSB	short for Hue, Saturation and Brightness

Chapter 1
Introduction

1.1 Introduction

While the standard of living conditions is rising, and the world population is growing, the demand for timber has been ever-increasing for many uses including furniture and building constructions. Timber has been a very important resource in the life of humans from ancient times. It is necessary to provide produced timber as wooden beams or planks for furniture or home construction. In these circumstances, various kinds of wood are found and identified as a switch for different uses in different environments. It is important to identify the type of wood accurately as it has a major impact on usability and cost. The price and the quality of timber depend on the distinguishable characteristics and the unique features each type of wood has[1]. To use the timber for a particular intended task, it is necessary to recognize the type of wood, especially in the timber industry.

Another advantage of recognizing timber is to verify on fake timber as some timber sellers lean towards to combine different types of wood materials with growing their income margin. Table 1.1 showing the types of timbers. Many countries have banned the export wood species such as the *Gonystylus bancanus* which mainly supply by Malaysia, due to pressure from the environmental associated non-government organizations. It is a protected species as it is included in the list of critically endangered plant species that is rapidly heading to be wiped out. Due to its demands, people are illegally exporting the *Gonystylus bancanus*. Good background knowledge of timber and correct identification of timber types will help to track the illegal trades and cheating occurring timber in business. It is essential to distinguish the timber accurately as those are utilized for specific appropriate uses. Erroneously identified wood can cause an erratic impact as some timber species are being used as building material in the construction field. In Sri Lanka, the traditional methodologies are still used to identify the wood species and it necessitates broad knowledge and experience. The uniqueness of the tree in the wood can be simplified by exploring the leaves, flowers, and fruits. Though, once the tree is cut down and gets the timber out from it, the identification of the type of tree is difficult. The aging process of the wood object, stains, wax, varnish, and other finishing treatments can further complicate the recognition of the wood. Wood has distinct characteristics that are not easy to identify. Some timber species look similar which is making it difficult to determine the correct identity. [2] Therefore, with the proposed methodology we are expecting to address the eager necessity of the automatic timber identification system which uses low-cost equipment for the recognition of timber species based on the images of wood pieces.

Table 1. 1 : List of timber names with botanical name and family Statement

No	**Common Name**	**Botanical Name**	**Family**
1	Acasia	*Acasia melanaoxylon*	Leguminosae
2	Albizia	*Paraserianthus falcataria*	Leguminosae
3	Alastonia	*Alastonia macrophylla*	Apocynaccac
4	Atamba	*Mangifera Zeylancia*	Anarcardaccae
5	Balu	*Shorea ciliata*	Ditpterucarpaceae
6	beddidel	*Artocarpus nobilis*	Moraceae
7	Bintangor	*Callophylum spp*	Guttiferac
8	bitis	*Madhuca utilis*	Sapotaccae
9	Bluegum	*Eucalyptus globulus*	Myrtaceae
10	Burutha	*Chloroxylon swietenia*	Rutaceae
11	Citridora	*Euaclyptus citridora*	Myrtaceae
12	Cypress	*Cypress spp*	Cupressaccae
13	Dawata	*Carallia brachiata*	Rhizophoraccae
14	Domba	*Calophyllum inophyllum*	Clusiaceae
15	Ebony	*Diospryros ebenum*	Ebanaceae
16	Ehela	*Cassia fistula*	Leguminosae
17	Gammalu	*Pterocarpus marsupium*	Fabaccae
18	Ginisapu	*Michelia Champaca*	Magnoliaceae
19	Garndis	*Eucalyptus grandis*	Myrtaccae
20	Halmilla	*Berrya cordifolia*	Tilliaccae
21	Hora	*Dipterocarpus zeylancius*	Dipterocarpaccac
22	Ilulanhik	*Chukrasia tabularis*	Meliaceae
23	Jak	*Artocarpus heterophyllus*	Moraccae
24	Kandis	*Garcinia spp*	Guttiferae
25	Kapur	*Dryobalanops spp*	Dipterocarpaceae
26	Khaya	*Khaya senengalensis*	Meliaccac
27	Kaluwara	Diospyros ebenum Koenig	Ebenaceae
28	Kempus	*Koompassia malaccensis*	Leguminosae
29	Keranji	*Dialium spp*	Leguminosae
30	Kcruing	*Dipterocarpus spp*	Dipterocarpaceae
31	Ketakela	*Bridelia retusa*	Euphorbiaceae
32	Kolon	*Adina cordifolia*	Rubiaccae
33	Kon	*Schleichera oleosa*	Sapindaceae
34	Kumbuk	*Terminlia arjuna*	Combretaceae
35	Liyan	*Homalium zeylanicum*	Flacourtiaccae
36	Lunumidella	*Melia dubia*	Meliaceae
37	Madan	*Syzygium cumini*	Myrtaccae
38	Madatiya	*Adenanthera pavonina*	Leguminosae
39	Mahogany	*Swietenia macrophylla*	Meliaceae
40	Malaboda	*Myristica dactyloides*	Miristicaccae
41	Mango	*Mangifera indica*	Anacardiaceae
42	Margosa	*Azadirachta indica*	Meliaceae

43	Meranti	*Shorea uliginosa*	Dipterocarpaceae
44	Mi	*Madhuca longifolia*	Sapotaccae
45	Micro	*Eucalyptus microcoris*	Myrtaccae
46	Milla	*Vitex altissima*	Verbinaceae
47	Na	*Mesua ferrea*	Clusiaceae
48	Nawada	*Shorea stipularis*	Dipterocarpaceae
49	Nedun	*Pericopsis mooniana*	Leguminosae
50	Neralu	*Elaeodendron glaucum*	Celastraccae
51	Oak	*Quercus spp*	Fagaccae
52	Palu	*Manilkara hexandra*	Sapotaccae
53	Paramara	*Samanea saman*	Leguminosae
54	Pihimbiya	*Filicium decipiens*	Sapindaceae
55	Pilularis	*Eucalyptus pilularis*	Myrtaccae
56	Pinus	*Pinus caribaea*	Pinaceae
57	Red Balau	*Shorea kunstleri*	Dipterocarpaceae
58	Redgum	*Eucalyptus brasiliensis*	Myrtaccae
59	Rukattana	*Alastonia scholaris*	Apocynaceae
60	Rubber	*Hevea brasiliensis*	Euphorbiaceae
61	Sabukku	*Grevillea robusta*	Proteaceae
62	Satin	*Chloroxylon swietenia*	Rutaceae
63	Suriyamara	*Albizia odoratissima*	Leguminosae
64	Sapu	*Michelia champaca*	Magnoliaceae
65	Teak	*Tectona grandis*	Verbinaceae
66	Timbiri	*Diospyrous malabarcica*	Ebenaceae
67	Toona	*Toona ciliate*	Meliaceae
68	Tualang	*Koompassia excelsa*	Leguminosae
69	Walsapu	*Michelia nilagirica*	Magnoliaceae
70	Walkohomba	*Melia azedarach*	Meliaceae
71	Welang	*Pterospermum suberifolium*	Sterculiaccae

Woods have distinct characteristics which are difficult in identification. Some timber types look similar which makes it difficult to determine the correct identity. The aging process of the wood object, stains, wax, varnish, and other finishing treatments can further complicate the recognition of the wood type. On a density scale, the densest are Calamander, Ebony, Satinwood, then Nedun, followed by Jak and Teak. Good background knowledge on timber and correct identification of timber types will help to track the illegal trades and cheating occurring in the timber business. Therefore, the proposed research problem is the accurate identification of wood from digital images.

1.2 Importance of the automated timber recognition system

- Timber recognition is a highly focused and interesting task as much as the economic value of the timber is concerned.
- There isn’t any recognizing system is developed earlier.
- If any user does not know the wood types, then merchants can cheat on users by giving them wrong timber types.
- No conflict between similar types.

1.3 Aim and Objectives

The goal of this study is to design and build a timber identification system in Android platform which could identify the type of wood from the enormous data sets through visual analysis by using images of the different types of timber taken from the timber cooperation and timber mills.

The main objectives of the research can be categorized as,

- Build a mobile application which correctly identifies the timber
- Use a database of pre-defined data sets and identify the type of timber through a template matching approach.
- Build a machine learning based system that can learn by itself to identify the type of timber that is not in the reference data set which can be used to improve the reference data set.

Achieving the above objectives will allow the system to identify the correct type of timber and avoid duplicate timber selection. The layman will be benefited from this and will be able to distinguish the timber accurately and use them for their appropriate use.

1.4 Types of Indigenous Wood Used in the Study

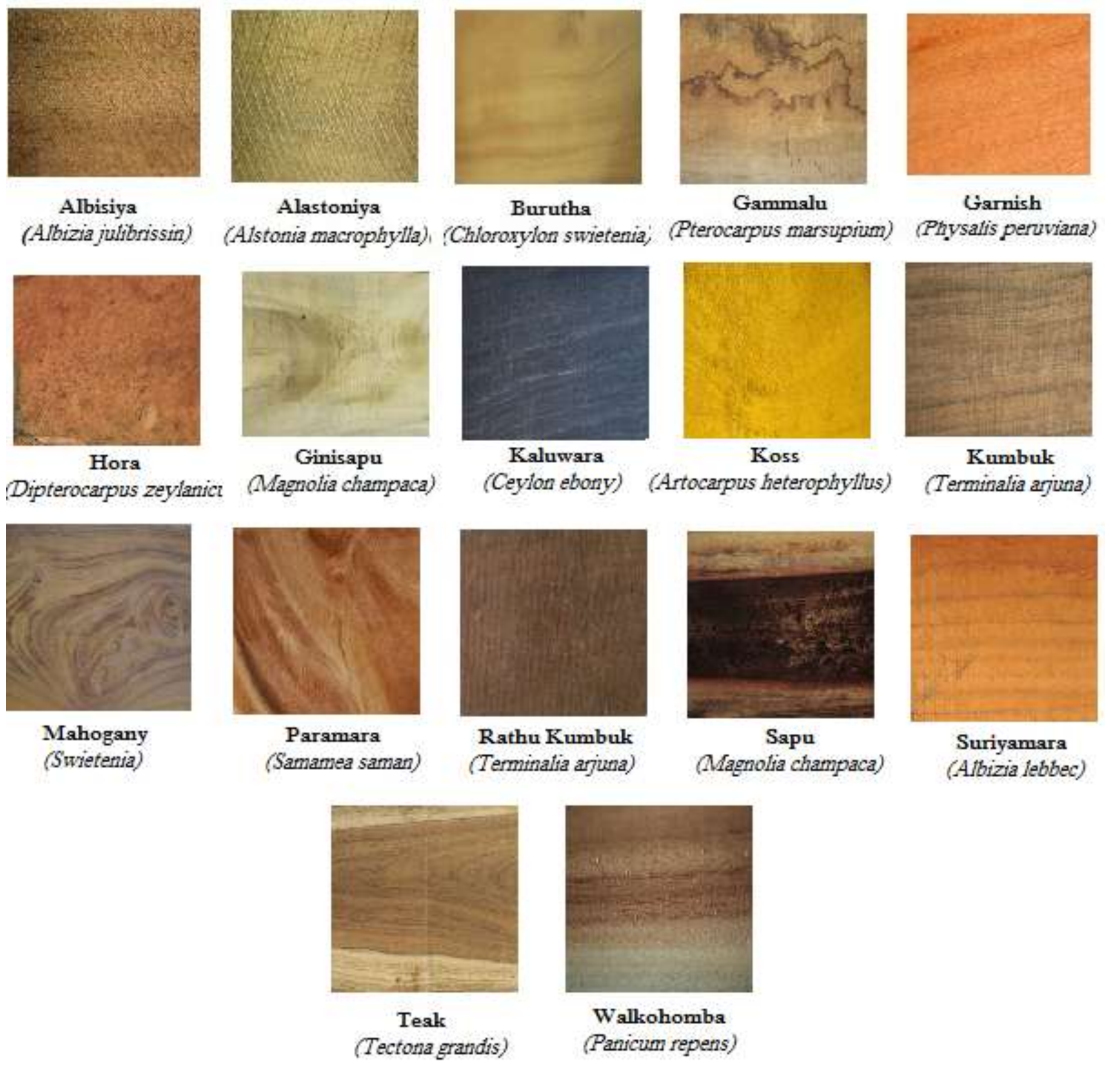


Figure 1. 1: Digital images of different types of wood species

1.5 Summary

This document explains an automated system of identification of wood by using digital images. Despite the complications and similarities of wood species, our focus is to distinguish timber types using image processing techniques and machine learning. Chapter 2 of the document is on the literature review of wood recognition systems and techniques used in timber identification. Chapter 3 covers the methodology which we have used in this study of timber recognition. The methodology is mainly developed upon image processing techniques and machine learning. Chapter 4 illustrates and describes the results we got from testing the proposed system in detail. The final chapter is on the conclusions we gathered by analyzing and experimenting with the proposed methods to identify wood species.

Chapter 2
Literature review

2.1 Literature Review

Many studies have been carried out to identify wood species using image processing. The number of frauds increasing each year and deforestation became major reasons to implement such systems to avoid fault identification and fraud transactions. To find a solution, many works focused on identifying wood species accurately. Most researches have been carried out to get macroscopic level images and analyze the features to determine a specific wood species. The approach of using macroscopic images of wood has been resulting in accurate answers for identifying wood species. Even though, the strategy seems to be working fine; there are many drawbacks of the practical usage of the method. Taking macroscopic images every time, a wood species need to be identified is time- consuming, expensive, and extensive. Hence, we use digital wood surface images to classify timber species which is challenging with accuracy measures.

The approach of using digital wood surface images is easy to use by any given user at any time. The solution of identifying wood species accurately through the wood surface image is developed mainly to be used by the timber sellers and buyers who are involved with such issues of fraud detection.

2.2 Texture Recognition

In wood species classification, the most considered aspect is texture recognition. Gabor filters are one example of texture classification introduced by Dennis Gabor. Gabor filter is a linear function where it analyses whether there is any specific frequency content in the image in specific directions in a localized region around the point or region of analysis [3]. Even though Gabor Filters is used for many applications the most efficient texture classification is GLCM (Grey-Level Co-occurrence Matrix) which was introduced by Haralick et al. 1973 [3]. Authors have proved that GLCM performs better than Gabor Filters in texture classification by comparing and combining the features of Gabor Filters and GLCM for a wood classification study [4].

2.3 Support Vector Machine (SVM)

The tree species identification is done by using high-resolution satellite imagery data with 0.5m spatial resolution Worldview -3. Some trees are highly valued in the market [5]. Tropical forest is also stored with a stock of carbon which below and above ground biomass. Tree species identification control forest cut, critical timber logged. This helps to reduce global warming and effect on the carbon cycle. Tree classification is time-consuming and limited access in the site survey. Remote sensing technology identified tree species. This study scope is limited to carry out to identify dominant tree species of Ayer Hitam Forest Reserve Puchong for biomass values. The method is used as Support Vector Machine (SVM) for this classification. Forest managers control the forest ecosystem to avoid climate change. Remote sensing techniques are a more effective and conventional method. Quantitative methods are used to analyze the research problem. Quantitative data along with statistical methods analyze the research study with particular statistical models. Statistical findings provided particular figures and solutions to investigate the study.

The crucial task of wood recognition focused on the anatomical features of wood and physical properties of wood [6]. Traditionally wood was recognized human experts on different characteristics of wood such as texture, color, and structure. There are wood species which unable to identify for experienced scientists. The study is used to recognize wood species through multidimensional texture analysis. Multidimensional signals help to identify wood images from periodic spatially evolving characteristics. Vertical and horizontal image patches use with linear dynamical systems as the method for the study. Then images concatenated in histograms. Support Vector Machines (SVM) classifier as the histogram representation of wood species. This method and data set are named as WOOD AUTH. There are more than 4200 wood images about radial, cross, tangential sections about wood structure. The study considered 12 wood species in the Green territory. These images contained high classification rates of wood cross sections when compared to tangential and radial sections. There are some misclassifications that existed between anatomically different wood species. This is happening due to not ideal shooting conditions carried out by non-professional photographers. Microscopic images and macroscopic images extend the proposed methodology. Mathematical models and statistical methods used to investigate the study. The graphical presentation is used to conduct the study.

2.4 Gray-Level Co-Occurrence Matrix (GLCM)

There are 14 features in GLCM. Out of the 14 features, five features were selected for the proposed method for wood classification: contrast, correlation, energy, entropy, and homogeneity. Apart from the features mentioned above, the GLCM itself can be taken as a feature after it is down-sampled to a raw GLCM [7]. A GLCM is generated by cumulating the total numbers of gray pixel pairs from an image. A GLCM will be generated by defining a spatial distance (distances between neighboring cell-pairs on the image) and an orientation. For each wood image, features are extracted from 4 different orientations like horizontal (0^o), vertical (90^o), and diagonals (45^o, 135^o) as shown in Figure 2.1. The features are extracted from both training data and test data. The extracted features from trained and test images are then co-related for image classification.

In this study, we have used 3 features for wood texture classification. They are energy, Angular Second Moment (ASM), and homogeneity.

Energy is the orderliness of an image. Energy is 1 for a consistent image.

$$Energy = \sum_{i,j=0}^{N-1} \left(P_{ij}\right)^2$$

The angular second moment is the uniformity and also this is a measurement of local homogeneity.

Homogeneity is the most commonly used measure that increases with less contrast in the pixel.

$$Homogeneity = \sum_{i,j=0}^{N-1} \frac{P_{ij}}{1+(i-j)^2}$$

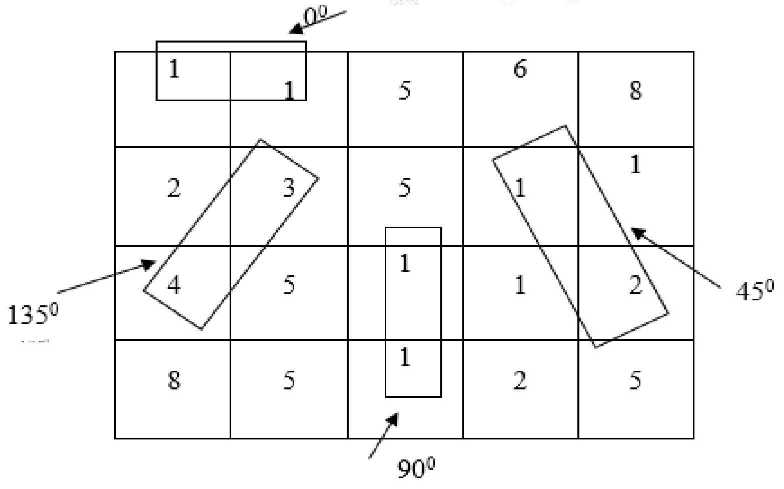


Figure 2. 1: Four orientations of GLCM

Correlation texture measures the linear dependency of grey levels of neighboring pixels in an image. The correlation is a measure of association between two images to find the portions that match according to the measure of correlation. The correlation is calculated as the last step to determine the class of the test image.

2.5 Gabor Filters

The Gabor filters are also identified as Gabor Wavelets. The theory behind Gabor Filter is stimulated by mammalian simple cortical cells [4].

A reviewed research on Gabor filters and grey-level co-occurrence matrices in texture classification used a method of merging both texture classification methods point out above to achieve higher accuracy [4]. The authors have compacted the dimensionality of the Gabor filter feature vector by using PCA. Features from GLCM and Gabor filters are taken into one feature vector without any modification to original values and then classified using k- Nearest Neighbor Algorithm. This blend of features has value-added accuracy rather than taking one feature extractor into consideration.

Many studies carried out after the introduction of the texture classification techniques over the years. An efficient hardware implementation for fingerprint image enhancement using anisotropic Gaussian filter studied by Tariq Mahmood Khanat et al. The author used real-time image techniques for faster implementation of the fingerprint enhancement [8]. The fingerprint filtering techniques are very expensive for hardware implementations. Therefore, they want to use an effective method to implement the image enhancement and decided to use a modified anisotropic Gaussian filter. Every person’s fingerprint is unique and immutable, consisting of parallel ridges and furrows. Sometimes one ridge may split into two (ridge bifurcation) and sometimes they do terminate and continue no more (ridge endings). These local ridge singularities (deviations from normal parallel behavior), also called as minutiae points which are unique to each person are used to identify individuals.

2.6 Classification Approaches

Classification approaches always deal with pre-processing techniques. In image processing, there are several popular pre-processing techniques such as edge detection, RGB to Gray, re-size images, image enhancement, sharpening and contrasting. Edge detection is the most used efficient pre-processing technique in image processing where many types of research are based on. We are discussing two main edge detection approaches in this paper namely: Scale-Invariant Feature Transform (SIFT) and Speeded-Up Robust Features (SURF). Many studies have used SIFT to pre-process the images in order to reduce the expense to train qualified identifiers and to increase the accuracy measurements. Once, the pre-processed images are obtained, the images are then fed into a machine learning algorithm.

One such study is wood species recognition on SIFT key point histogram [9]. The authors have used the SIFT algorithm to extract key points from wood cross section images. After that, k-means and k-means++ algorithms are used to cluster the images. After obtaining the clustering results, a SIFT key points histogram is calculated for each wood image. Furthermore, classification models, including Artificial Neural Networks (ANN), Support Vector Machine (SVM) and K-Nearest Neighbor (KNN) are used to verify the performance of the method. The use of SIFT key points histograms proved to be more successful than using GLCM and Local Binary Pattern (LBP).

Another study is timber defect detection using Local Binary Pattern (LBP), Scale-Invariant Feature Transform (SIFT), and Support Vector Machines (SVM) [10]. First, the authors create a dictionary based on the bag-of-words approach in the training step. The dictionary is generated either using LBP and SURF features alone or as a combination of both features. Then, an image processing pipeline which associates contrast, enhancement, entropy maximization, and image filtering is used to detect the potential defect regions. After that, the authors use SVM classifier to detect knots and cracks. The proposed algorithm is evaluated on two different datasets which have knots and cracks as ground truth. The experimental results show a precision of around 0.9.

A Tropical wood species recognition system based on multi-feature extractors and classifiers is introduced as an automated wood recognition system to classify tropical wood species by Marzuki Khalid et al. [1]. The wood features are extracted based on two feature extractors: Basic Grey Level Aura Matrix (BGLAM) technique and statistical properties of pores distribution (SPPD) technique. The study involves the comparison of the system with and without pre-classification using KNN classifier and LDA classifier. The methodology of the system consists of image acquisition, image pre-processing, feature extraction, clustering and dimension reduction in the pre-classification stage and finally the classification. The k- means clustering and dimension reduction happens in the pre-processing stage have reduced the computational time required by the system while k- Nearest Neighbor and Linear Discriminant Analysis have been used to calculate accuracy. As it is not efficient to deal with high dimensional data, the dimensionalities of the input images were reduced as a key processing step before the classification. The sub-patterns within the dataset were used for the recognition of image classes. So the proposed solution has used nonlinear reduction based on different patterns within data samples which is called Sp-DNDR. This approach has shown a higher accuracy in image recognition with high scalability in practical usage.

Another research is carried out in using Gabor Filters as Image Multiplier for Tropical Wood Species Recognition System [11]. As wood images do not have many distinguishable features for unique identification, the Gabor filter was used within this research to derive more features within the pre-processing step. Then the features were extracted based on GLCM approach and Gabor filters by generating more features to increase the accuracy. A multilayer neural network was used for the classification and obtained a high accuracy as a result of an increasing amount of features. In the Computer Vision field to help prevent misclassifying of wood types in the timber industry, a new application of intelligent wood species recognition based on Local Binary Pattern is studied[6]. Wood recognition is done by capturing images of the wood samples and by observing their characteristics. In modified anisotropic Gaussian filter is efficiently adopted in hardware by decomposing the filter into two orthogonal Gaussians and an oriented line Gaussian. Architecture is developed for dynamically controlling the orientation of the line Gaussian filter. To further improve the performance of the filter, the input image is homogenized by local image normalization. Test results show an improved speed for its hardware architecture while maintaining reasonable enhancement benchmarks.

Research of rotational invariant wood species recognition by Jing Yi Tou et al. is describing a system that can identify wood species by wood species verification and rotational invariant recognition [12]. The main objective is to recognize wood species is by using GLCM but having a verification process along with it make the solution much more efficient. The wood texture images which are at the macroscopic level are used in this study. Apart from the GLCM features, raw GLCMs were taken as features for the dataset. Four GLCMs were constructed for training data. For test samples, GLCMs were taken from eight different orientations rather than rotating the images in order to come up with a rotational invariant algorithm. Another objective is the wood species verification where the algorithm checks whether the tested sample is from the same species with the samples in the templates. The energy value is calculated from the train set to get a threshold value. Then the energy value is calculated from the test dataset. The energy value computed from the test data will be compared with the threshold value and test sample will be verified against the template. Three types of experiments were done in this study by using images of 576x768 pixels, 512x512 pixels, and 256x256 pixels. This is to test whether the area size matter on calculating the accuracy. Results show that the 576x768 had more accuracy but other two experiments got low results compared to the first one. This proved that a computer vision algorithm needs an image wide enough to cover the sufficient characteristics in texture. The second, third, and fourth species gave high accuracies but lower for the rest. This happens due to the distinct textures in the two different samples that are belonging to the same species confusing the classifier. Therefore, it is vital to have more samples from the same species in order to train and collect good models. The wood verification and rotational invariant technique had more impact on the accuracy of the data. Texture samples from each species were collected and verified. The GLCMs in test samples were taken in 8 different orientations to be matched with the trained data.

Data fusion techniques are complex recognition system but performance is high. This technique helps to tree species recognition [13]. Information extracts from leaves and barks. The study provides results to increase the fusion system through the power of discrimination and better detection results only on leaves. The two fusion system architectures are identified and compared during the study. These two fusion strategies recognize tree species through barks and leaves. Sub classifiers of leaves perform well and bring more information when compared to barks sub-classifiers. The system is properly managing confusion introduced through barks. The first strategy is shown high performance when compared to the second strategy. This method is good to adapt to the system to quality recognition of species. Species identify leaves. That information might not use to reduce barks confusion. When barks are identified information, which those will not use to leaves. Algorithms built in bagging are used as classifiers. Numerical data was used to analyze the study. Complete data analysis was done with the quantitative data analysis method. Scatter plots and other diagrams used to carry out the study. Statistical models are also supported by the study.

2.7 Artificial Neural Networks

The research of automated recognition of wood damages using Artificial Neural Network(ANN) has been studied by Zhao Dong where they have implemented a solution to identify wood damages of three types using an artificial neural network [14]. The objective is to find wood damages based on the level of damage namely: Undamaged, Moderate Damage, and Severe Damage. The damage on the wood was imposed by humans using a specific experimental method using special instruments. Finally, an Artificial Neural Network (ANN) is used to do the training of damaged and undamaged samples. There are three types of damaged wood cases such as Undamaged (UD), Moderate Damage (MD), and Severe Damage (SD) [1]. Training dataset had 60 sets of data having 20 sets per damaged wood group. Rest of the 30 sets were test data where each group had 10 datasets. The node number of the input layer and the output layer are 8 and 2 respectively. The node number of the hidden layer is 10. Therefore, the network structure is 8X10X2. If the expected value of the network output is [0 0] in the allowed error range, the undamaged case appeared (UD). If the expected value of the network output is [0 1] in the allowed error range, the Moderate-Damage case appeared (MD). If the expected value of the network output is [1 0] in the allowed error range, the severe-damage case appeared (MD). The 60 groups of data were trained by the neural network for 3000 times and the error is less than 0.01. In the results from those objectives are the data was trained to verify the non-linear ability of Back Propagation (BP) network and the network output agrees well with the expected result.   First, the verification of the node of the input, hidden, and output layers were defined. The node number of the hidden layer was verified to 10 in this study. This was determined through experience by executing the algorithm by increasing the node number. The node number of the input layer is determined by the eigenvector number made up of training cases. As there are three types of wood damages, the node number of the output layer is 2. Therefore, a network is constructed to 8x10x2. The training data will be fed to the network and trained for 3000 times. The error given by the network was less than 0.01 by providing an efficient mechanism of identifying wood damages. We hope to use a similar type of neural network in order to identify the type of timber. The paper described a clear view on how data can be trained using the neural network.

Toro M at el find a long-term problem to match sonar images with high accuracy because it is hard to model sonar images due to reflections, viewpoint dependence and noise [15]. Autonomous underwater vehicles require sonar image matching capabilities for simultaneous localization and mapping, tracking and objective detection/recognition. Convolutional Neural Networks matches train to match labeled sonar data. Then it is done after pre-process of particular matching or non- matching of pairs as the second step. 0.91 areas cover the ROC curve for CNN with the 39k training pair's data set. As of under a binary classification matching decision which covers 0.89 are for another CNN as the output matching score. Under the classical key point matching methods like SURF, SIFT, AKAZE, and ORB obtain 0.61 to 0.68 AUC. Support Vector Matching gets 0.66 AUC and Random Forecast Classifier is an alternative learning method which obtains 0.79 AUC. Therefore, it is clear that 2 Chan CNN class network match FLS image patching with high accuracy over 0.91 AUC. However, classical key point matching is only done with low accuracy, but it is only better than random chance. These results explained to get appropriate results for small training data set like 39k samples. It is viable to increase the data and objects with high variability. These kinds of data sets are possible to capture from a small water tank with household garbage objects under laboratory conditions. This method is limited by the small number of objects and images. Quantitative data is used to conduct this study as well. Quantitative data use to carry out particular statistical methods to investigate the study.

The Convolutional Neural Networks (CNN) have been trending past few years due to its efficient feature extraction capability and the requirement of relatively little pre-processing. One such example is forest species recognition using deep CNN.

Researchers have investigated the use of CNN for two forest species datasets; one with macroscopic images and the other with microscopic images authors have achieved around 97% of accuracy for the microscopic image dataset using the deep CNN [16].

Another research is a methodology to identify wood samples of seven commercial timber species using CNN [17]. Authors have created a system with images of wood species and a portable microscope attached to a personal computer. The images were divided into patches and grouped into training, validation and test sets. The convolutional neural network consists of four layers: two convolutional layers with max pooling and two fully connected layers at the output. The results seem to be higher with 94% accuracy.

The people adaptation on advanced technologies that artificial intelligence and neural networks play a major role in fields of automobiles, medicine, military, aeronautical science, and many more. Taxonomic research plants are carried out in botanical science field very little [4]. The manual identification and classification of plant species are not much accurate. However, long term results based on the same matter is not much accurate. Advanced Plant Identification System (APIS) is an intelligent system. Photographs of leaves of tree species help to identify more accurate results of leaves over time. APIS is a web-based application with the server-side application and client-side applications. Web service used to communicate with each other. Server-side application use to feature extraction process and image recognition in this method. APIS is a processing and neural network technique which removes noise and normalizes leaf area [18]. This reduces unwanted and scales the leaf image. Capture the image of a leaf on a white background. Then upload the captured leaf image to APIS system through client application and upload of the image. Then this converts into a byte array and sends to the server. Then byte array image coverts to the original image. Then three spate clone images are used in three different ways. APIS accuracy is 95%.

In this research process, different feature sets extracted, combined and evaluated in the framework [19]. Two approaches focus on the framework is multiple feature sets and image segmentation. Macroscopic and microscopic images help to increase the word recognition rates from 74.58% - 95.68% and 68.69% - 88.90%. It is important to explore appropriate local estimation and window sizes to estimate LPQ descriptor. The result found from this study is tested better than a database. This method is important than Gaussian functions and classical uniform. The strategy developed on Blackman function. Further, the study identified the advantageous of the results obtained from the study. The database of the study is connected with multiple features sets framework. A deeper analysis of the method is required for local estimation. Multiple feature vectors combine in hierarchical approach through employing of quadtree based approach. Graphs and statistics are analyzed with the study to carry out the study. Numerical analysis was used to analyze the study.

Artificial Neural Networks (ANN) is a deep neural network which has multilayers. This is one of the powerful tools as for the literature [20]. Artificial Neural Networks (ANN) can handle a huge amount of data. This interested in identifying deeper hidden layers and surpassing classical method performance which usually uses in pattern recognition. Further, CNN is an important layer within the network and time-consuming. Performance of the network also depends on the number of levels. Convolutional Neural Network (CNN) is the most popular deep neural network. The mathematical linear operation uses to identify convolution among matrixes. Convolutional Neural Network has multiple layers covering non-linearity layers, fully connected layers, convolutional layer, and pooling layer. The conventional layer has parameters and fully connected layers do not have parameters. Convolutional Neural Network had shown excellent performance during machine learning problems. Image data provided a large image classification, natural language processing, computer vision to achieve amazing results. CNN is a powerful tool for machine learning on video recognition, face detection, and image and voice recognition.

The convolutional neural network is identified as a mathematic model [21]. This used to identify close recognition. This model can be developed with the Apparel classification and clothing image dataset. Augmentation method expands the dataset. The productivity of the process can be developed in stable and efficiently with the neural network. Type and color are used as two parameters for clothing identification. Computer vision technology identified the complete solution for clothing identification and recognition. Computational experiments are used under low efficiency with the key points of computer descriptors and detection of SIFT and SURF. Then experimentations carried out to evaluate the neural network model developed through the convolutional neural network — training data set to use to apply under the augmentation method and increase the accuracy of recognition. Clothing recognition algorithm accuracy is around 84%. Quantitative data are collected and analyzed under necessary models to accurately analyze the research problem in a meaningful way. Necessary graphical presentations are also used appropriately.

Tree species can be identified automatically through convolutional neural networks (CNN) [13]. Tree leaves can be analyzed through identification of multi-dimensional features of shape, color and leaf vein signatures. Convolutional neural networks integrate multi-dimensional leaf features accurately and identify tree species. Rotation invariance is achieved by additional preprocessing steps. Robustness identifies and increases the preprocessing steps of tree species identification. Leaf snap database evaluates through convolutional neural networks and satisfying performance. Leaf classification task is filled effectively with the experimental results. Leaf snap database contained 184 kinds of trees. Every picture shows a light background on a single leaf. These pictures are taken under different conditions to vary the background. Different samples are taken out from the leaf snap database. 30 samples are taken from each species from the database. This classification is difficult with the process. In order to validate the study, 25 kinds of tree species are taken from the database. More samples generate with the zoom of original leaf image and translation of the original leaf image. 2358 samples have experimented in total. Reliable classification results can be easily achieved with the preprocessing process. Quantitative data is collected to investigate the study. CNN model is carried out statistically with quantitative data.

The author Zhang at el self - organizing feature map neural network helps to identify the feature extraction of leaf including shape and texture features through 2 D movement [22]. Self- organizing feature map has advantages of low complexity of learning, the simple structure of the leaf and mapping topology. These benefits solve the high dimensional input vector, multi-class pattern recognition and large quantity training data. First, in this method, 2 D movements help to extract the shape features of the leaf. Then statistical movements and discrete wavelet transform used to capture the texture information of enervation. Then SOM neural network identifies the plant species. 2-D moments invariants are extracted and continued the study with quantitative methods. Statistical calculations made possible findings in the study. Quantitative data provided a clear numerical analysis in the study. The shape of the leaf feature is taken from the quantitative method. Particular statistical models used to investigate the study. A sample is collected from full-grown leaves from the forest. Leaf image captured from CCD Camera or scanner. Leaf images are put in a vertical direction. RGB color image is considered as leaf output. Graphs charts are also used to present the results in a meaningful way.

2.8 Image Processing

Automatic counting system of logs helped to avoid manual operating and time consuming and tedious issues and inaccuracies [2]. Production efficiency of log yards can be improved with the counting of logs. Digital image processing on log counting is discussed in the paper. Images of the logs produced with CCD cameras. Then logs separate from its background through Otsu thresholding. Difficulties arise with the various reasons for contrast quality, overlapping logs and touching. The distance transform is adapted to logs. As a result of that binary transform, images changed to one rank grey image. Then conglutinate logs of the picture divided to erosion arithmetic. Core points scan to count the logs. Counting can be done inaccurately with the loeroges that are circle images [23]. The steps of digital image processing are image collection, image smooth, thresholding, distance transform, Conforming the Core Dimension and Counting by Contour Tracking Method. Quantitative method is used to carry out the study in a meaningful way. Detail statistical calculations helped to carry out the calculations. Cumulative probabilities, variance, and etc. use to carry out the study. Quantitative data is gathered to analyze the study with graphs, charts and possible statistical methods.

2.9 Summary

The literature of image processing and computer vision has spread over a wide area and many new techniques are being introduced. Texture recognition is mainly done by GLCM or Gabor filters where SIFT and SURF methods are being used for edge-detection in images. Apart from the techniques mentioned above, many machine learning algorithms are combined to classify and identify images with supervised learning. Deep learning is a new trend which is showing tremendous growth in image processing. There are special neural network designs especially built for image recognition such as convolutional neural networks.

Chapter 3
Methodology

3.1 Methodology

This chapter describes the proposed methodology of using digital images of wood to identify wood species. Image processing techniques and machine learning are the core components of developing the timber recognition system. There are many databases of wood found online such as CAIRO etc. These databases of wood consist of the macroscopic level of timber structures. The limitation of identifying timber species using macroscopic images is a limited number of images in databases due to time-consuming, expensive, and extensive research carried out to create macroscopic structures of wood. The proposed system is using digital images which can be acquired in a faster and effective way. Figure 3.1 below shows the methodology of the proposed system in wood species classification.

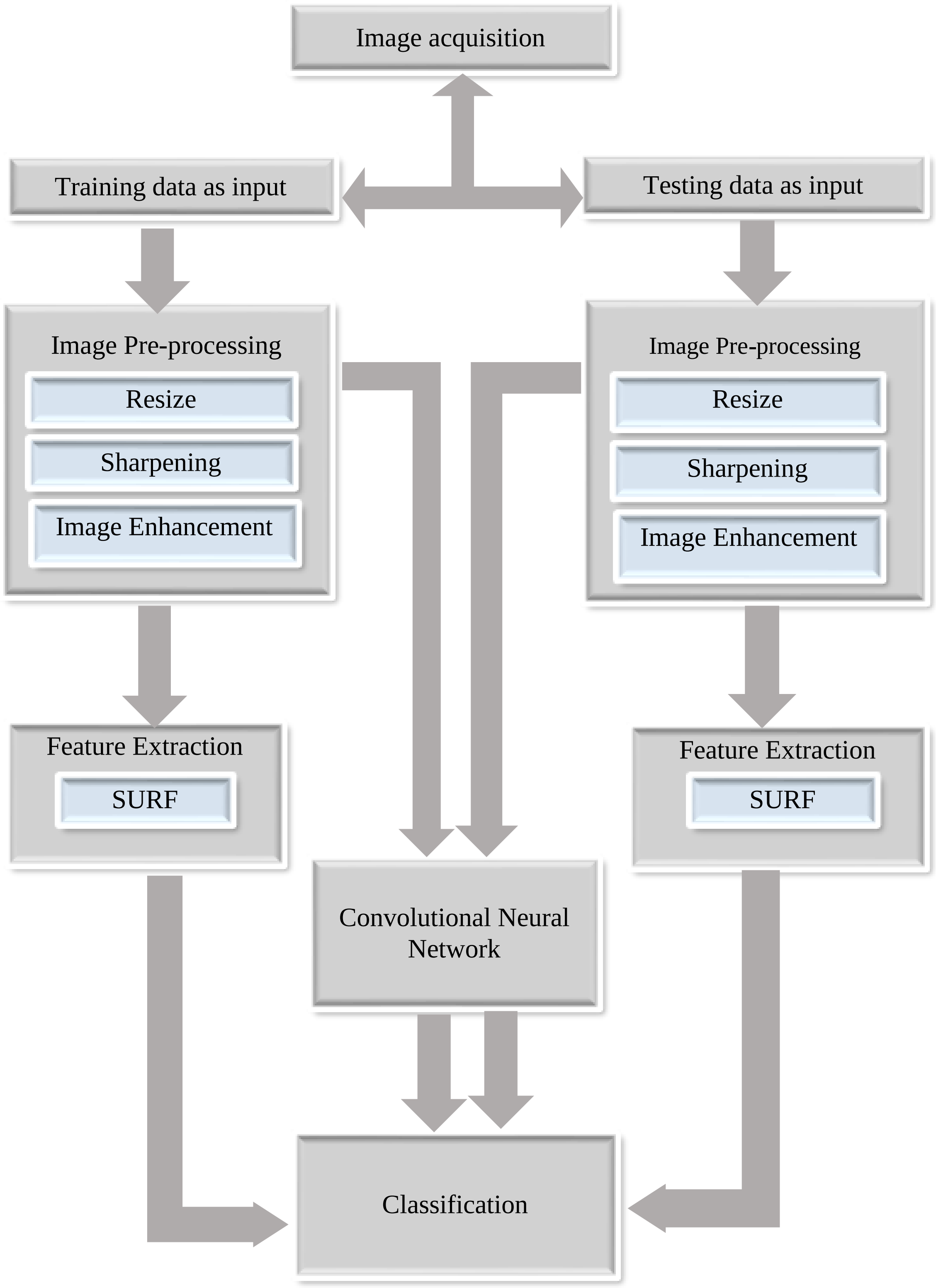


Figure 3. 1: Steps of the timber recognition in the proposed system

3.1.1 Image Acquisition

Collection of digital wood species images using NIKON- DSLR (D 90), lenses used: NIKON, in the range of 18-125 mm image capturing. Data were collected from the timber corporations in Matara Thalalla, Rathmalana and Bussa in Sri Lanka. Each dataset has several images of a specific type of a tree. The total dataset consists of 500 digital images.

3.1.2 Training and Testing Datasets

The training data and testing data split into 12260 and 5254 respectively. We are especially focused on the texture and the color of wood pieces to identify the timber species. We have gathered digital images from 17 different indigenous trees of Sri Lanka as described in table 3.1.

Table 3. 1: Indigenous timber species experimented in the system

Name of Species	Botanical Name
Teak	*Tectona grandis*
Ebony	*Diospyros ebenum*
Jak	*Artocarpus heterophyllus*
Gammalu	*Pterocarpus marsupium*
Mahogany	*Swetenia Macrophylla*
Suriyamara	*Albizia odoratissima*
Kumbuk	*Terminalia arjuna*
Paramara	*Samamea saman*
Suriyamara	*Albizia odoratissima*
Garnish	*Eucalyptus grandis*
Walkohomba	*Melia azedarach*
Albisiya	*Paraserianthus falcataria*
Sapu	*Michelia champaca*
GiniSapu	*Michelia Champaca*
Hora	*Dipterocarpus zeylancius*
Kaluwara	*Diospyros ebenum Koenig*

Table 3. 2: The quantity of timber images types and training, testing images data set

Timber Name	# of images	Training	Testing
Albisiya	896	625	271
Attoniya	1000	700	300
Burutha	1098	770	328
Gammalu	498	350	150
Garnishh	780	540	240
Ginisapu	578	406	172
Hora	904	632	272
Kumbuk	878	618	260
Koss	1361	956	405
Kaluwara	906	635	271
Paramara	787	550	237
Mahogani	2668	1867	801
Rathu kumbuk	33	23	10
Sapu	1058	740	318
Suriyamara	1188	831	357
Tekka	1754	1227	527
Walkohomba	1127	788	339
Total	**17514**	**12260**	**5254**

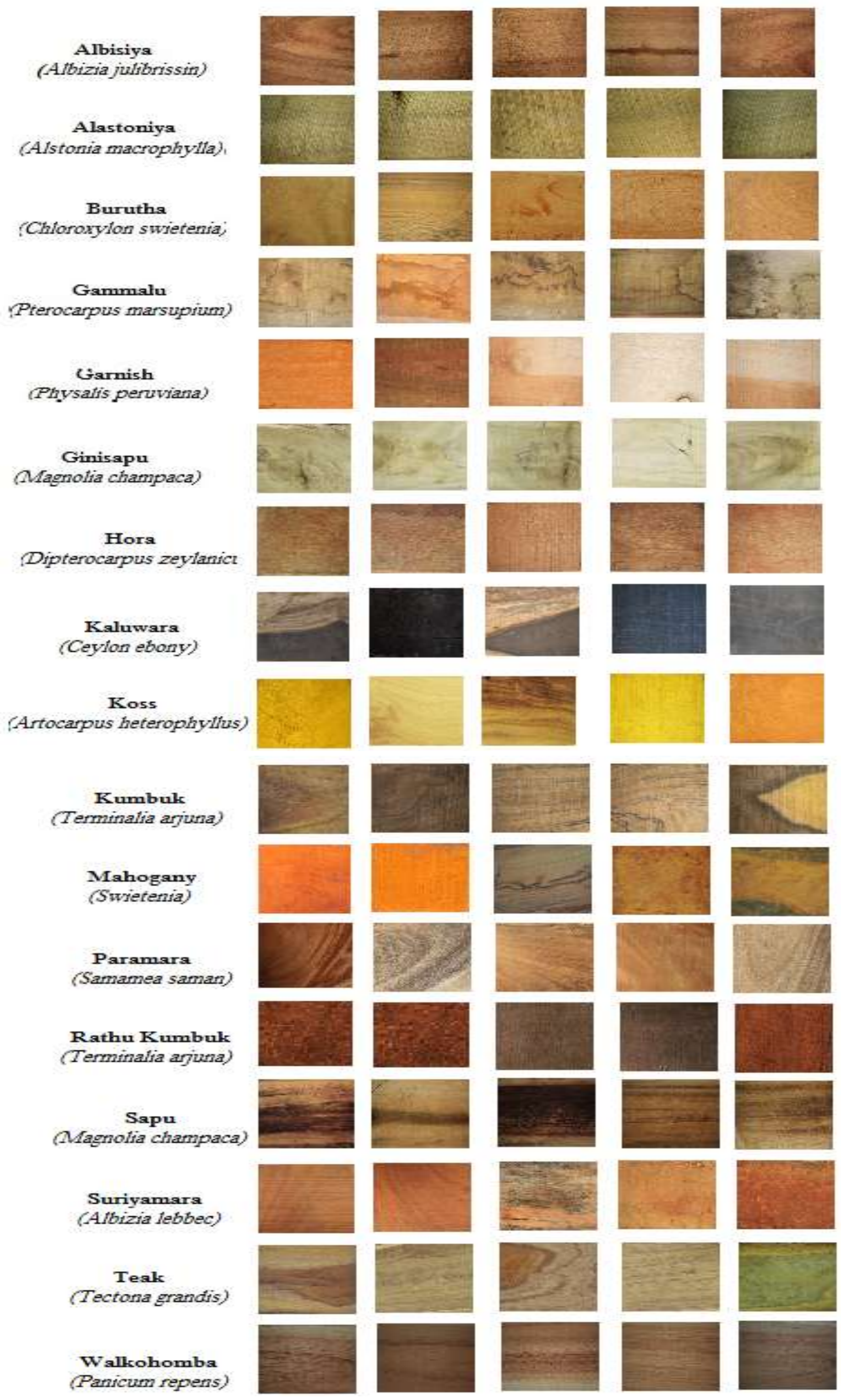


Figure 3. 2: List of data set in 17 classes with different images of each classes

As the initial step, we have used the GLCM to extract features and k- Nearest Neighbor Classification Algorithm to classify the wood species. The features we extracted were; homogeneity, Angular Second Moment (ASM), and entropy. Before sending the images to the GLCM, we pre-processed all training and testing images using edge detection techniques, RGB to Gray, image enhancement, and re-size. The results obtained from the pre-processing techniques are illustrated in figure 3.2.

3.1.3 Image Pre-processing

The first step of the proposed solution is image pre-processing. Image pre-processing techniques were applied to the digital images we gathered for the wood database. There are several preprocessing techniques we tried: resize image enhancement such as sharpening and contrast, RGB to Gray, and edge detection. The numbers of images are shown here in figure 3.3.

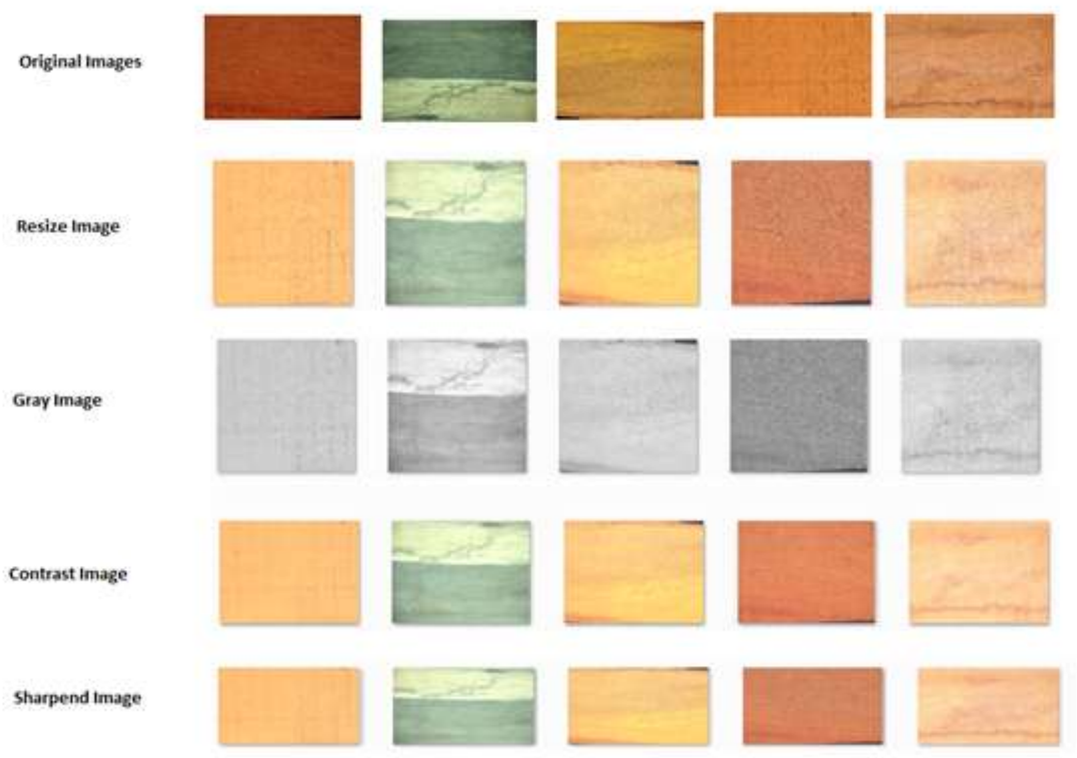


Figure 3. 3: Pre-processed images (Original, Resize, Gray, Contrast, and Sharpened)

3.1.4 Edge Detection

Finding the boundaries of objects within images is called Edge detection which is an image processing technique. It is the method of detection the discontinuity in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.

We used two edge detection methods;

The first method was SIFT (scale-invariant feature transform) where the algorithm takes images to define key points and descriptors. The key points are points of interest in an image whereas descriptors provide a unique and robust description of a key point.

The second method is an enhanced method of SIFT namely SURF (Speeded Up Robust Features).

In figure 3.4 to figure 3.6 shown the SIFT and SURF methods. Authors of SURF proved that SIFT is more robust against image transformations than SIFT [7].

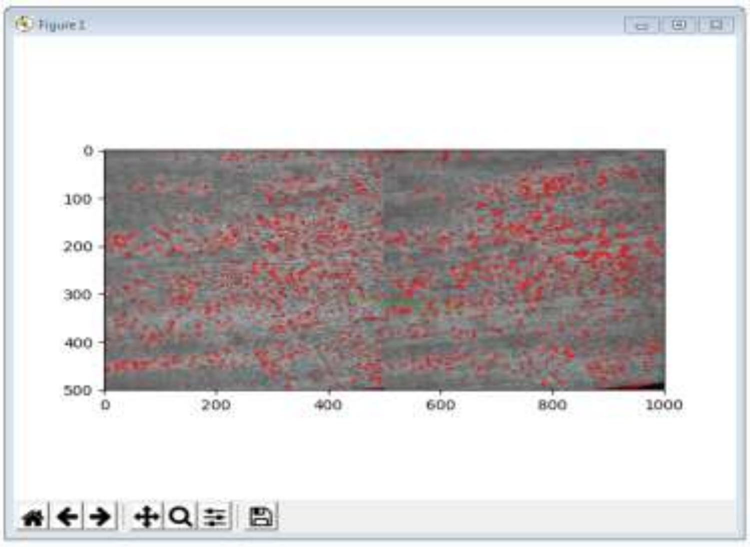


Figure 3. 4: Test two images using SIFT (Gammalu and Gammalu)

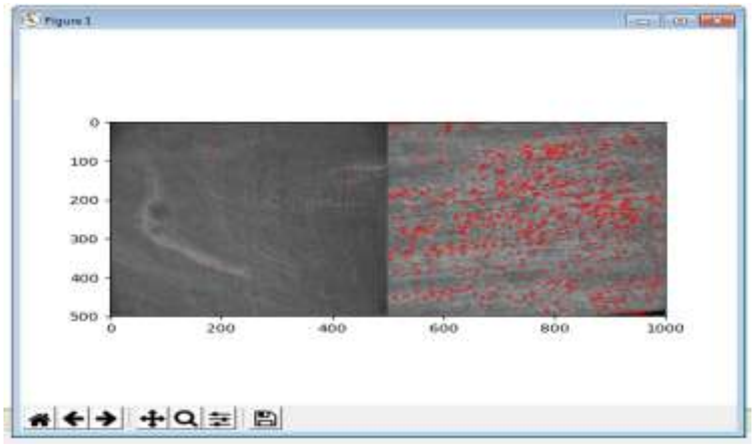


Figure 3. 5: Test two images using SIFT (Ebony and Gammalu)

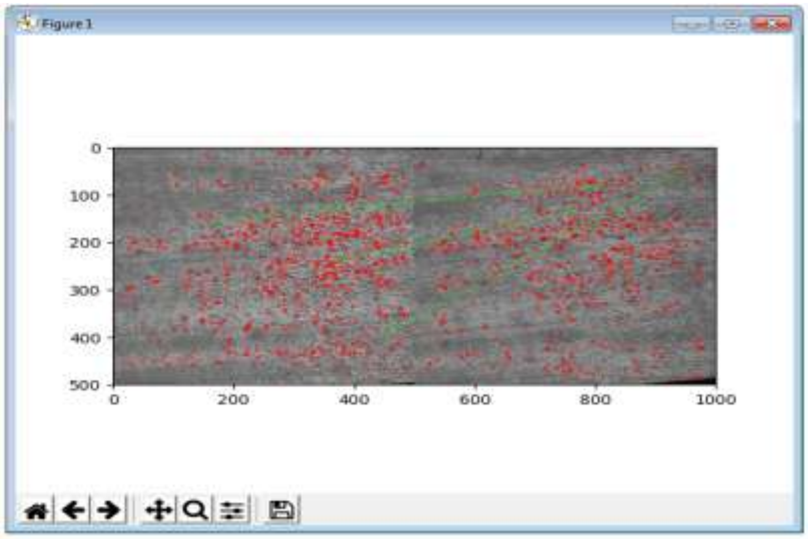


Figure 3. 6: Test two images using SURF (Gammalu and Gammalu)

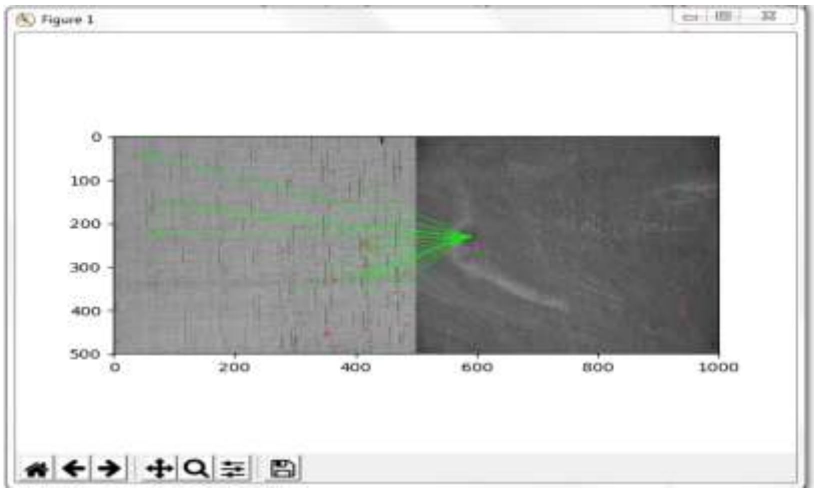


Figure 3. 7: Test two images using SURF (Garnish and Ebony)

When we compared same species, SIFT figure 3.4 and 3.5 shows the same number of key points in the training and test images but when we compared two different species, SIFT learns the difference by showing no relevance with the least number of key points. The same results figure 3.6 and 3.7 were generated by SURF method as well.

3.1.5 RBG to Gray

The method is developed on by taking an average of three colors. Since it is an RGB image, which means you are getting the desired gray-scale image by adding R, G and B and then dividing the answer by 3.

It is done in this way; Grayscale = (R + G + B / 3)

3.1.6 Image Enhancement

- Sharpening: It is used to subtract the blurred images copies from the original image to detect any edges.
- Contrast: This increases at the edges and the effect of the sharpening is applied to the original image.

3.1.7 Hue, Saturation, and Value (HSV)

In graphics and paint programs the RGB color model is often used. When using this color model after the color is specified, white or black colors are added to make easy adjustments. HSV may also be called HSB (short for Hue, Saturation and Brightness). In our proposed methodology, we are finding the mean and standard deviation of the HSV image. Mean can be considered as the average color value in the image. Standard deviation is the square root of the variance of the distribution.

3.1.8 Image Resizing

When resizing or distorting the image from one-pixel grid to another, Image interpolation occurs. When you are in need of increasing or decreasing the total number of pixels, Image resizing is necessary. When correcting for lens distortion or rotating an image Remapping can occur. Zooming refers to increase in the number of pixels, so that when you zoom an image, you will see more detail.

3.1.9 Feature Extraction

After image pre-processing, feature extraction takes place. Features were extracted from digital wood images by using GLCM (Gray-Level Co-Occurrence Matrix). GLCM is a popular texture classification technique which calculates the co-occurrence of pixel pairs within an image according to the spatial distance between the pixels and four orientations of pixels. In GLCM, there are 14 features that can be extracted and among the 14 features, 5 features are considered essential for wood classification: contrast, correlation, energy, entropy, and homogeneity. Apart from the features mentioned above, the GLCM itself can be taken as a feature after it is down-sampled to a raw GLCM [8]. A GLCM is generated by cumulating the total numbers of gray pixel pairs from an image. A GLCM will be generated by defining a spatial distance (distances between neighboring cell-pairs on the image) and an orientation. For each wood image, features are extracted from 4 different orientations like horizontal (0o), vertical (90o), and diagonals (45o, 135o) as shown in Figure 3.8. The features are extracted from both training data and test data. The extracted features from trained and test images are then co-related for image classification.

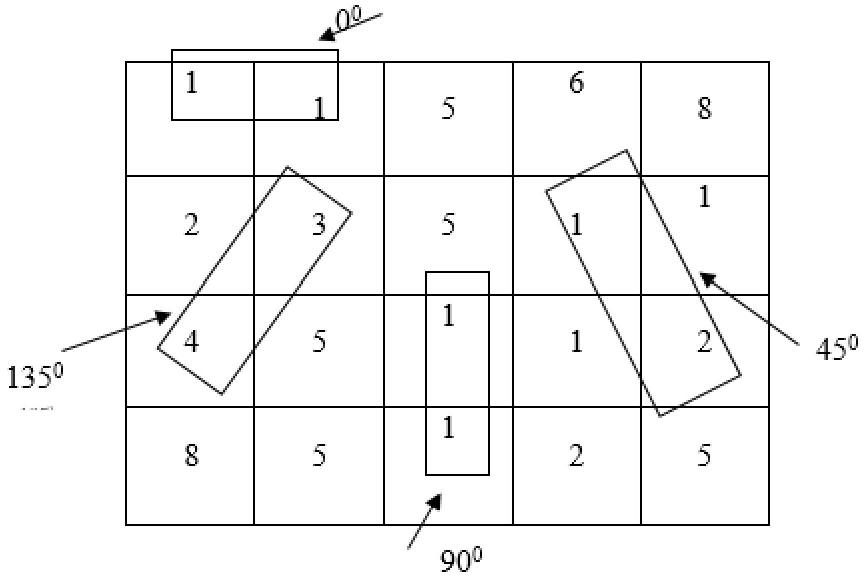


Figure 3. 8: Four orientation of GLCM

In this study, we have used 3 features for wood texture classification. They are energy, Angular Second Moment (ASM), and homogeneity.

a) Energy is the orderliness of an image. Energy is 1 for a consistent image.

$$Energy = \sum_{i,j=0}^{N-1} \left(P_{ij}\right)^2$$

b) The angular second moment is the uniformity and also this is a measurement of local homogeneity.

c) Homogeneity is the most commonly used measure that increases with less contrast in the pixel.

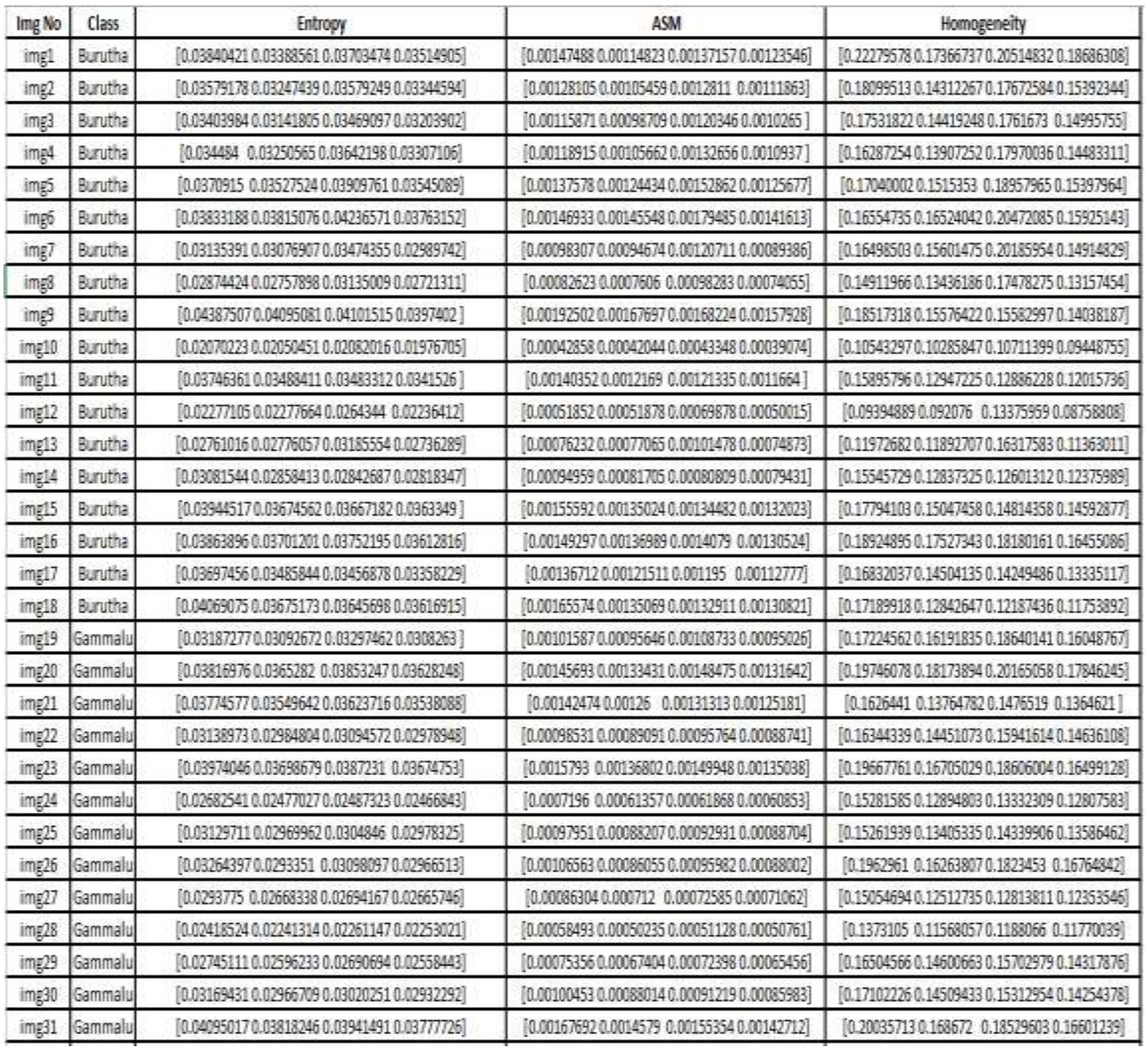
$$Homogeneity = \sum_{i,j=0}^{N-1} \frac{P_{ij}}{1+\left(i-j\right)^2}$$

a) Correlation texture measures the linear dependency of grey levels of neighboring pixels in an image. The correlation is a measure of association between two images to find the portions that match according to the measure of correlation. The correlation is calculated as the last step to determine the type

$$P(i,j) = P_d(i,j) / \sum_{i,j=0}^{N-1} P_d(i,j)$$

Of timber species of the test image. For the implemented system, if the value of the correlation is greater than 0.95, we can determine the test image is identified as one of the trained images. The formula we used in this study is the Pearson product-moment correlation as follows: The features we extracted were; homogeneity, Angular Second Moment (ASM), and entropy. Table 3.3 below shows the results obtained by the feature extraction of GLCM.

Table 3. 3: Entropy, ASM, and Homogeneity results obtained by GLCM



Img No	Class	Entropy	ASM	Homogeneity
img1	Burutha	[0.03840421 0.03388561 0.03703474 0.03514905]	[0.00147488 0.00114823 0.00137157 0.00123546]	[0.22279578 0.17366737 0.20514832 0.18686308]
img2	Burutha	[0.03579178 0.03247439 0.03579249 0.03344594]	[0.00128105 0.00105459 0.0012811 0.00111863]	[0.18099513 0.14312267 0.17672584 0.15392344]
img3	Burutha	[0.03403984 0.03141805 0.03469097 0.03203902]	[0.00115871 0.00098709 0.00120346 0.0010265]	[0.17531822 0.14419248 0.1761673 0.14995755]
img4	Burutha	[0.034484 0.03250565 0.03642198 0.03307106]	[0.00118915 0.00105662 0.00132656 0.0010937]	[0.16287254 0.13907252 0.17970036 0.14483311]
img5	Burutha	[0.0370915 0.03527524 0.03909761 0.03545089]	[0.00137578 0.00124434 0.00152862 0.00125677]	[0.17040002 0.1515353 0.18957965 0.15397964]
img6	Burutha	[0.03833188 0.03815076 0.04236571 0.03763152]	[0.00146933 0.00145548 0.00179485 0.00141613]	[0.16554735 0.16524042 0.20472085 0.15925143]
img7	Burutha	[0.03135391 0.03076907 0.03474355 0.02989742]	[0.00098307 0.00094674 0.00120711 0.00089386]	[0.16498503 0.15601475 0.20185954 0.14914829]
img8	Burutha	[0.02874424 0.02757898 0.03135009 0.02721311]	[0.00082623 0.0007606 0.00098283 0.00074055]	[0.14911966 0.13436186 0.17478275 0.13157454]
img9	Burutha	[0.04387507 0.04095081 0.04101515 0.0397402]	[0.00192502 0.00167697 0.00168224 0.00157928]	[0.18517318 0.15576422 0.15582997 0.14038187]
img10	Burutha	[0.02070223 0.02050451 0.02082016 0.01976705]	[0.00042858 0.00042044 0.00043348 0.00039074]	[0.10543297 0.10285847 0.10711399 0.09448755]
img11	Burutha	[0.03746361 0.03488411 0.03483312 0.0341526]	[0.00140352 0.0012169 0.00121335 0.0011664]	[0.15895796 0.12947225 0.12886228 0.12015736]
img12	Burutha	[0.02277105 0.02277664 0.0264344 0.02236412]	[0.00051852 0.00051878 0.00069878 0.00050015]	[0.09394889 0.092076 0.13375959 0.08758808]
img13	Burutha	[0.02761016 0.02776057 0.03185554 0.02736289]	[0.00076232 0.00077065 0.00101478 0.00074873]	[0.11972682 0.11892707 0.16317583 0.11363011]
img14	Burutha	[0.03081544 0.02858413 0.02842687 0.02818347]	[0.00094959 0.00081705 0.00080809 0.00079431]	[0.15545729 0.12837325 0.12601312 0.12375989]
img15	Burutha	[0.03944517 0.03674562 0.03667182 0.0363349]	[0.00155592 0.00135024 0.00134482 0.00132023]	[0.17794103 0.15047458 0.14814358 0.14592877]
img16	Burutha	[0.03863896 0.03701201 0.03752195 0.03612816]	[0.00149297 0.00136989 0.0014079 0.00130524]	[0.18924895 0.17527343 0.18180161 0.16455086]
img17	Burutha	[0.03697456 0.03485844 0.03456878 0.03358229]	[0.00136712 0.00121511 0.001195 0.00112777]	[0.16832037 0.14504135 0.14249486 0.13335117]
img18	Burutha	[0.04069075 0.03675173 0.03645698 0.03616915]	[0.00165574 0.00135069 0.00132911 0.00130821]	[0.17189918 0.12842647 0.12187436 0.11753892]
img19	Gammalu	[0.03187277 0.03092672 0.03297462 0.0308263]	[0.00101587 0.00095646 0.00108733 0.00095026]	[0.17224562 0.16191835 0.18640141 0.16048767]
img20	Gammalu	[0.03816976 0.0365282 0.03853247 0.03628248]	[0.00145693 0.00133431 0.00148475 0.00131642]	[0.19746078 0.18173894 0.20165058 0.17846245]
img21	Gammalu	[0.03774577 0.03549642 0.03623716 0.03538088]	[0.00142474 0.00126 0.00131313 0.00125181]	[0.1626441 0.13764782 0.1476519 0.1364621]
img22	Gammalu	[0.03138973 0.02984804 0.03094572 0.02978948]	[0.00098531 0.00089091 0.00095764 0.00088741]	[0.16344339 0.14451073 0.15941614 0.14636108]
img23	Gammalu	[0.03974046 0.03698679 0.0387231 0.03674753]	[0.0015793 0.00136802 0.00149948 0.00135038]	[0.19667761 0.16705029 0.18606004 0.16499128]
img24	Gammalu	[0.02682541 0.02477027 0.02487323 0.02466843]	[0.0007196 0.00061357 0.00061868 0.00060853]	[0.15281585 0.12894803 0.13332309 0.12807583]
img25	Gammalu	[0.03129711 0.02969962 0.0304846 0.02978325]	[0.00097951 0.00088207 0.00092931 0.00088704]	[0.15261939 0.13405335 0.14339906 0.13586462]
img26	Gammalu	[0.03264397 0.0293351 0.03098097 0.02966513]	[0.00106563 0.00086055 0.00095982 0.00088002]	[0.1962961 0.16263807 0.1823453 0.16764842]
img27	Gammalu	[0.0293775 0.02668338 0.02694167 0.02665746]	[0.00086304 0.000712 0.00072585 0.00071062]	[0.15054694 0.12512735 0.12813811 0.12353546]
img28	Gammalu	[0.02418524 0.02241314 0.02261147 0.02253021]	[0.00058493 0.00050235 0.00051128 0.00050761]	[0.1373105 0.11568057 0.1188066 0.11770039]
img29	Gammalu	[0.02745111 0.02596233 0.02690694 0.02558443]	[0.00075356 0.00067404 0.00072398 0.00065456]	[0.16504566 0.14600663 0.15702979 0.14317876]
img30	Gammalu	[0.03169431 0.02966709 0.03020251 0.02932292]	[0.00100453 0.00088014 0.00091219 0.00085983]	[0.17102226 0.14509433 0.15312954 0.14254378]
img31	Gammalu	[0.04095017 0.03818246 0.03941491 0.03777726]	[0.00167692 0.0014579 0.00155354 0.00142712]	[0.20035713 0.168672 0.18529603 0.16601239]

After pre-processing and feature extraction, the images are taken for testing. After the GLCM feature extraction, we set the correlation to 0.95. If the value of the correlation is greater than 0.95, we can determine the test image is identified as one of the trained images. For example, we used the wood species; Jak (*Artocapus heterophyllus*) compared with four other species namely: Teak (*tectona grandis*), Paramara (*Samamea saman*), Kaluwara (*Diospyros ebenum*), and Burutha (*Chloroxylon swietenia*). Table 3.4 and Table 3.5 below shows the results of comparison.

Table 3. 4: Distinguishing different species by GLCM feature extraction Cont...

Type of Timber	Albisiya	Attoniya	Burutha	Gammalu	Garnishh	Ginisapu	Hora	Kumbuk
Albisiya	0.999921	0.554976	0.99484	0.869748	0.54374	0.920099	0.75604	0.871190
Attoniya	0.515320	0.997131	0.47707	0.040774	0.99726	0.692499	0.94860	0.036642
Burutha	0.992725	0.516024	0.96887	0.899080	0.51150	0.933190	0.72758	0.897759
Gammalu	0.861531	0.081220	0.89296	0.987152	0.07948	0.717237	0.34675	0.999374
Garnishh	0.544083	0.997622	0.51309	0.085240	0.86677	0.732641	0.95970	0.079400
Ginisapu	0.917514	0.724272	0.93323	0.721237	0.73548	0.973737	0.87537	0.712239
Hora	0.743874	0.968435	0.71615	0.335080	0.96595	0.865740	0.97216	0.331268
Kumbuk	0.859668	0.071119	0.88893	0.999611	0.06764	0.705130	0.33712	0.991388
Koss	0.978659	0.398062	0.99051	0.948377	0.39239	0.883680	0.63014	0.948064
Kaluwara	0.861857	0.634280	0.89866	0.733048	0.65735	0.982190	0.79439	0.719300
Paramara	0.861857	0.361432	0.57427	0.510141	0.41561	0.741320	0.47873	0.484342
Mahogani	0.654125	0.152220	0.72108	0.929492	0.13380	0.572560	0.10904	0.921305
Sapu	0.710732	0.124340	0.76574	0.963223	0.11427	0.589403	0.14412	0.958508
Suriyamara	0.633096	0.248740	0.68763	0.934869	0.24149	0.478813	0.01959	0.932001
Tekka	0.694980	0.095320	0.75927	0.944363	0.07764	0.618496	0.16646	0.936213
Walkohomba	0.717595	0.114650	0.77210	0.965575	0.10473	0.597024	0.15385	0.960877

Table 3. 5: Distinguishing different species by GLCM feature extraction

Type of Timber	Koss	Kaluwara	Paramara	Mahogani	Sapu	Suriyamara	Thekka	Walkohomba
Albisiya	0.982376	0.861467	0.496118	0.661660	0.72103	0.638545	0.704643	0.731360
Attoniya	0.359600	0.602469	0.348137	0.185728	0.16194	0.294316	0.126156	0.145903
Burutha	0.991970	0.895422	0.574237	0.726812	0.77396	0.691413	0.766864	0.783753
Gammalu	0.942450	0.726550	0.506550	0.930207	0.96678	0.938097	0.945208	0.969811
Garnishh	0.397250	0.653234	0.413760	0.126707	0.10858	0.244327	0.067283	0.092467
Ginisapu	0.885138	0.979866	0.737817	0.578405	0.59499	0.477244	0.626812	0.607898
Hora	0.619930	0.781871	0.472526	0.097674	0.13211	0.001139	0.158533	0.148178
Kumbuk	0.940110	0.710937	0.482876	0.924647	0.96423	0.938111	0.939288	0.967090
Koss	0.391108	0.854360	0.544970	0.794826	0.84332	0.776289	0.828560	0.851302
Kaluwara	0.858660	0.992776	0.848550	0.659898	0.65320	0.538927	0.701478	0.664755
Paramara	0.545015	0.856391	0.854839	0.639216	0.56262	0.467178	0.658180	0.569593
Mahogani	0.788699	0.651404	0.621908	0.999722	0.99236	0.981996	0.997582	0.991648
Sapu	0.834404	0.645561	0.552704	0.990414	0.98244	0.991048	0.991286	0.999698
Suriyamara	0.770428	0.538148	0.463258	0.978491	0.98988	0.683342	0.972125	0.987568
Tekka	0.821126	0.690836	0.640543	0.998476	0.99395	0.976257	0.973594	0.994170
Walkohomba	0.839781	0.651999	0.555464	0.989886	0.99998	0.989845	0.991376	0.747388

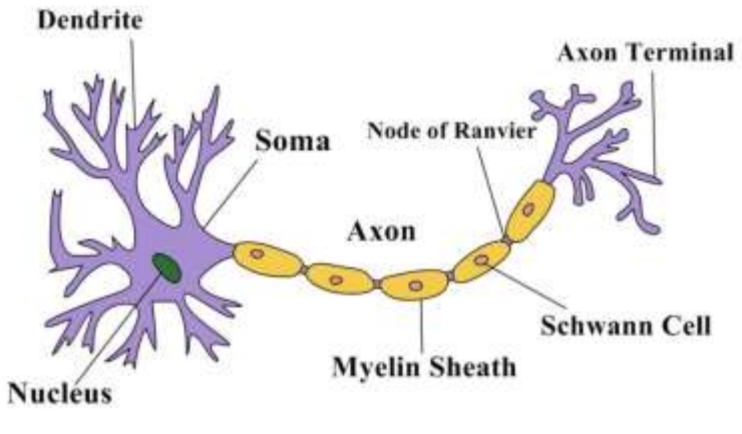
By observing the results in table 3.4, we can determine that results are accurate for distinguishing only some species. That is mainly because the GLCM is analyzing the digital wood surface images rather than the images at a macroscopic or microscopic level. The method is mainly detecting the color of the wood images and when there are two species with likely colors; the method fails to distinguish between some species.

As a solution, we analyzed more on the alternatives to be taken as the wood species need to identify accurately to meet the objectives of our research. A contemporary method which is efficient in image processing are neural networks.

3.1.10 Artificial Neural Networks in Image Processing

The concept of artificial neural networks is based on the human nervous system. The human nervous system is the main system which transmits signals in a human body. At the cellular level, the nervous system consists of trillions of special types of cells called neurons. These neurons have a special structure which enables signal transmitting.

Figure 3.9 shown a neuron or a nerve cell is different from other cells in the human body in many ways. The main difference and the unique role of a neuron is to transmit signals to communicate with other cells through synapses. The structure of a



neuron is illustrated.

Figure 3. 9: Human nervous system

A neuron is mainly consisting of a nucleus, axon, and axon terminal. The axon is responsible for making synaptic contacts. These axons are extended through the body bundled together which we call nerves. Neurons transmit signals through a membrane-to-membrane junction called synapses. These junctions are placed in between two neurons which transmit signals electronically or chemically. The structure of synapses is illustrated below in figure 3.10.

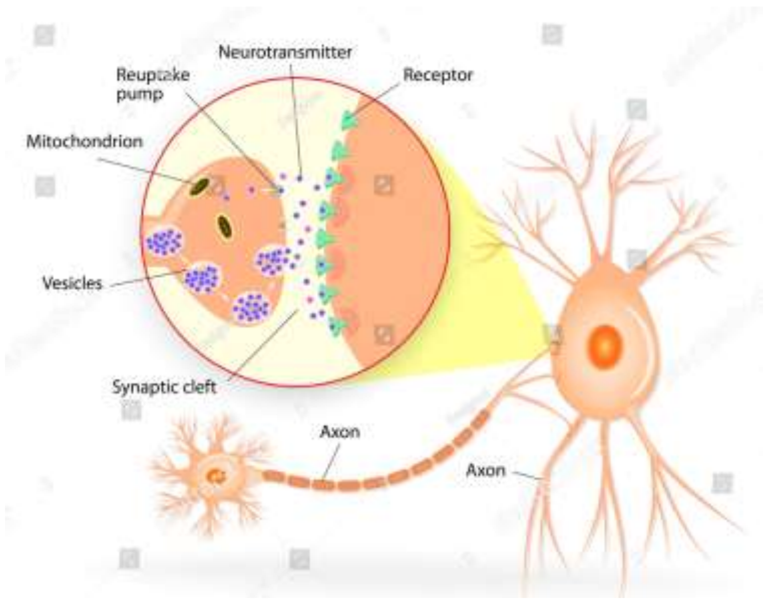


Figure 3. 10: The Human Nervous System: An Anatomical Viewpoint

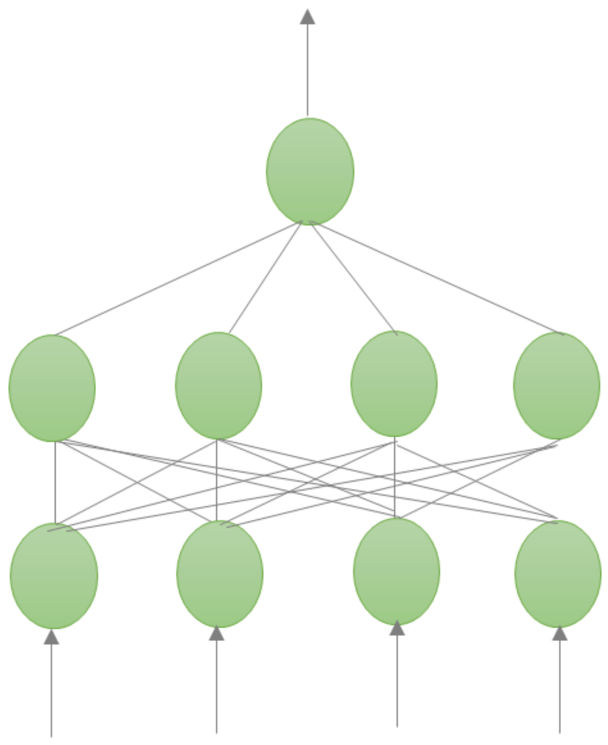


Figure 3. 11: Basic structure of the ANN

Similarly, an artificial neural network is also designed to transmit signals or data from one end to another end. The neural networks are computational models built upon the concept of neurons of the brain. One neuron takes an input, then process and transfer the output to the following neuron. Figure 3.12 below shows the architecture of a conventional neural network. Usually, a neural network consists of 3 types of layers: an input layer, hidden layer, and the output layer.

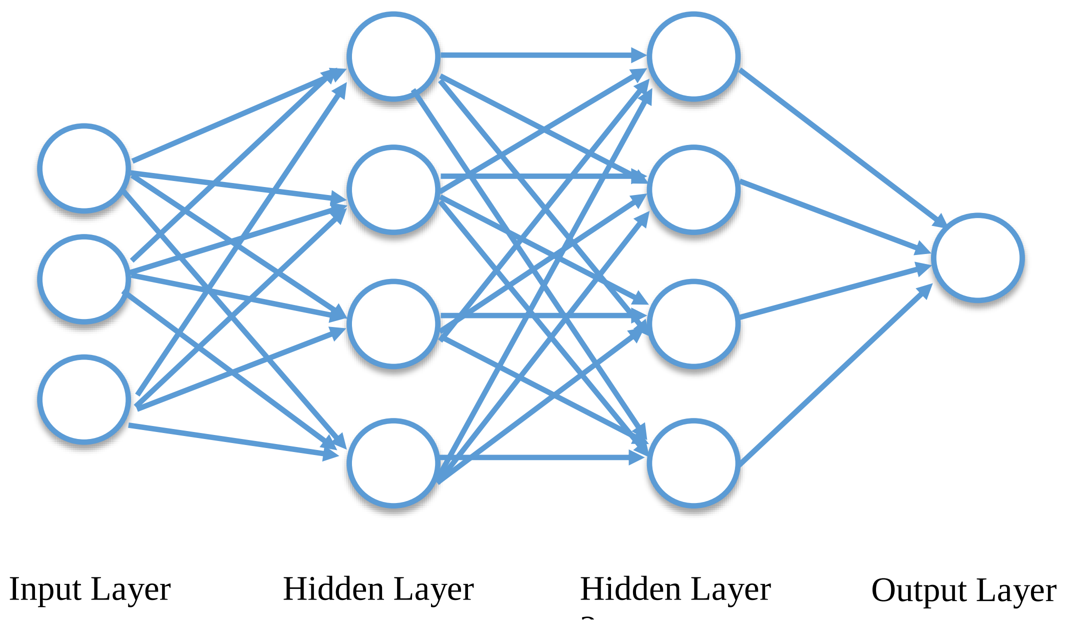


Figure 3. 12: Convolutional neural network with layers

There are a few components of an artificial neural network. They are neurons or nodes, weights, and activation functions. A structure of an artificial neuron is illustrated in figure 3.13.

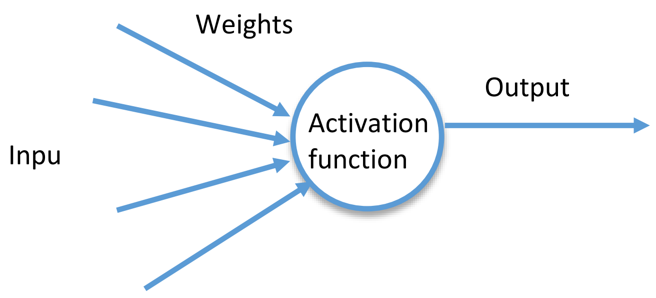


Figure 3. 13: Artificial Neuron

The nodes can be seen as computational units which take input or obtain the output. The connections between nodes determine the information flow. The information flow can be direct towards one side or bidirectional. In the human nervous system, the signals are passed through the nerves and if strong signals are received (surpass a certain threshold), a neuron is activated and transfer the signal through the axon; then to the synapses and then to another neuron. The artificial neuron also behaves in a similar way where input nodes are taking the signal at first. Then, some weights have been assigned to the connections where the input is getting multiplied by the weight value ignorer to trigger the activation function. These weight values can be negative as well. Therefore, the computation of neurons is dependent on the weights. Furthermore, certain algorithms are used to adjust and determine the weights. By adjusting the weight values, we can obtain the desired output for certain inputs. The adjustment of weights in an artificial neural network is called training.

There are different topologies introduced for artificial neural networks which are being used in many critical fields of analyzing and prediction. In this paper, we are mainly focusing on Artificial Neural Networks (ANN) in image processing. In image processing, a certain image data set should be sent into the neural network to teach the computer to recognize the images and classify them into relevant categories. By labeling the images, the computer will recognize the patterns in a particular image and start building its own cognition. As computers are able to read numbers rather than interpreting images, machine learning uses two ways of converting the images to numbers: using greyscale and using RGB values.

The input layer of a neural network gets inputs in different forms. The hidden layers are to transform the inputs process data. The feature extraction is also done in hidden layers which is an important step in identifying an image. Then, the output layer will produce the desired output. Each node of the neural network is given weight during the training process. At first, some random values of weights are given to each node. Also, each layer has a bias attached to influence the output. The bias and weights are used to calculate the activation function which is used to determine the output of a neural network.

3.1.11 Feedforward Neural Networks

This is the simplest form of an artificial neural network. In feedforward ANN, data travels in one direction. The data passes through inputs nodes and exit from output nodes. Feedforward neural networks may or may not have hidden layers. The sum of the weights and the products of inputs are calculated to obtain the output. The output is considered if it surpasses a threshold; usually, the threshold is set to zero and when the neuron fires with an activated output; usually 1. If the neuron does not fire, the deactivated value is emitted which is usually set to -1. The applications of feedforward neural networks can be found in classification problems such as computer vision and speech recognition.

The backpropagation neural network is a type of feedforward neural network which is used in layered feedforward neural networks. The signals are sent forward and there may be one or more hidden layers to the network. This is a supervised learning algorithm where we provide examples of the inputs and outputs we want the network to compute and the error is calculated. The error calculated is the difference of the expected and the actual output. In backpropagation, the error is reduced until the network learns the training data.

3.1.12 Radial Basis Function Neural Networks

This neural network is focused on the distance. The distance of a point with respect to the center is considered in calculating the output. There are two layers in RBF networks. In the first layer, the features are combined with the RB function in the inner layer. The output of these features is taken into consideration while computing the same output in the next time-step which is basically a memory.

The distance calculation happens from the center to a point in the plane similar to a radius of a circle hence the network is named as radial basis function neural network. Any distance measurement can be used in this domain and widely used measurement is Euclidean distance. The output depends on the radius of the circle where if the point is located in or around the radius, the likelihood of the new point begins classified into that class is high. The applications of such neural networks are used in power restoration systems.

3.1.13 Recurrent Neural Networks

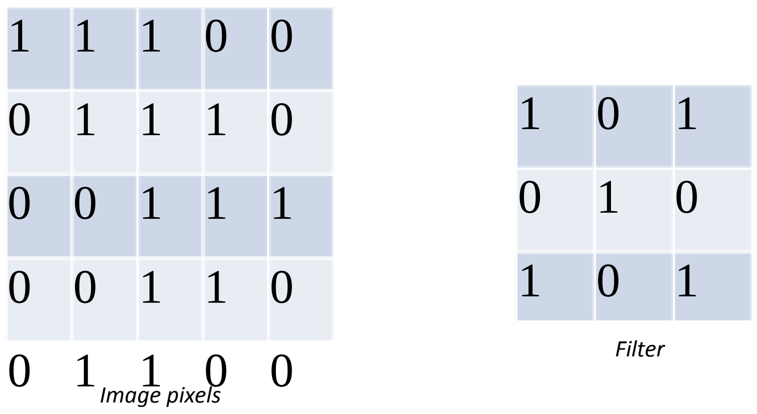
In recurrent neural networks, the output of the network is taken back as the input again to help in predicting the outcome of the layer. The first layer is designed as a feedforward neural network with the product of the sum of the weights and features. Each neuron of the network will remember some information it had in the previous time. This makes the neurons act like memory cells. The backpropagation is used to calculate the error and work towards the right prediction. This type of neural networks is widely used in the text to speech approaches.

3.1.14 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) belong to deep neural networks. CNNs are widely used for image analyzing. The major advantage of using a CNN is it requires little pre-processing. This is because a CNN learns filters which are hand-engineered by other algorithms. In a convolutional neural network, the images are divided into pieces or tiles rather than converting the image into numbers which also means a CNN accepts pixel values. Each tile is analyzed to identify a certain object wherever the object is in the image. Especially, in a convolutional neural network there are four types of layers:

Convolutional layer, ReLU Layer, Pooling Layer, and Fully Connected Layer. A filter matrix is used by the convolutional layer over the array of image pixels and performs convolution operation to obtain a convolved feature map. As shown in figure 3.15, the filter matrix can be slide through the image to compute the convolution operation.

3.1.15 Modular Neural Network



1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image pixels

1	0	1
0	1	0
1	0	1

Filter

Figure 3. 14: Modular Neural Network

Modular Neural Network has shown figure 3.14 is a collection of networks where each network acts independently. The inputs of each network are different from others where each network is responsible of accomplishing a subtask. The idea behind a modular neural network is to divide a large task into smaller components and reducing the complexity of a particular task. This approach will decrease the number of connections and increase computational speed.

3.1.16 Convolutional Neural Networks in Timber Recognition

Due to the powerful performance in the field of image processing and the requirement of little pre-processing relative to other image classification algorithms as CNNs use filers to learn and apply them to images. Therefore, we chose the CNNs to analyze the digital images of wood surfaces to classify wood species.

In our study, we have used a 2D feed-forward convolutional neural network for classifying test images. The CNN is consisting of 2 convolutional layers and one pool layer. The convolutional layers are in the size of 3 and 2 respectively. A convolutional layer does most of the computational process. Convolutional layer factors are consist of a set of filters. Let’s take a typical filter of one convolutional layer. The layer size will be 5x5x3 (5 pixels for width and height, and 3 because the images have a depth of 3). With the help of the filters, to produce a 2-dimensional activation map, we slide each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. Eventually, the network will learn filters that activate when they see some type of visual feature. Now, we have an entire set of filters in each convolutional layer and each of them will produce a separate 2-dimensional activation map. We will stack these activation maps along the depth dimension and produce the output volume.

The pooling layers are used for downsampling. The most common pool setting is max-pooling with 2x2 receptive fields which we have used in our research. This setting of downsampling reduces 75% of the activations in the input image. Figure 3.15 shows the architecture of the CNN which we have used in our study.

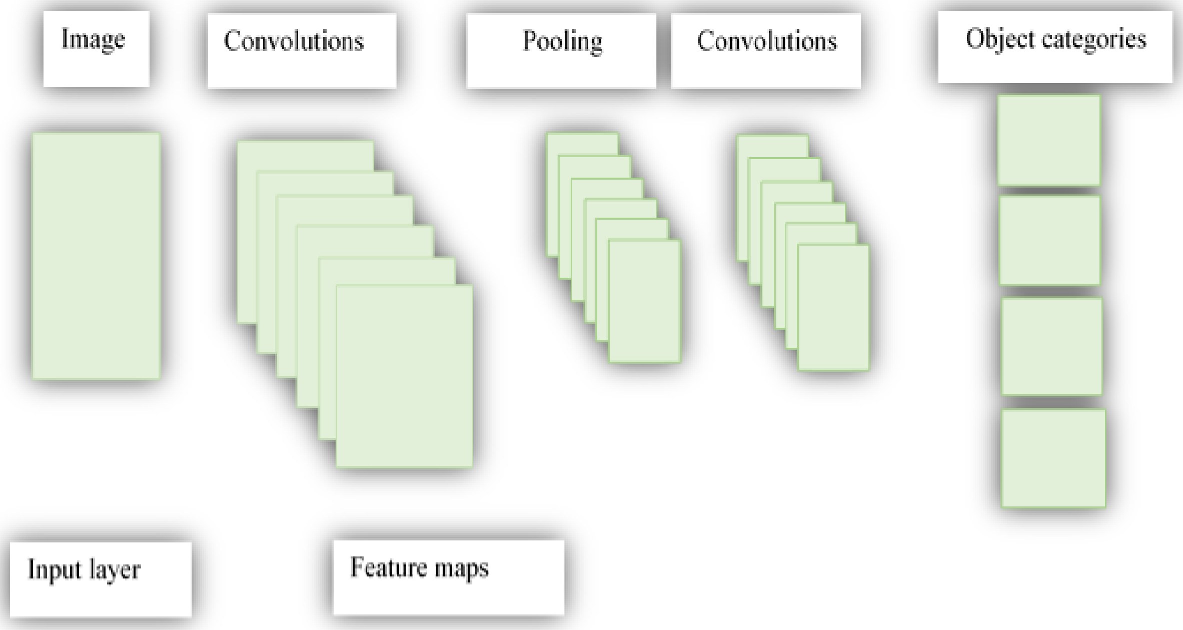


Figure 3. 15: Layers in CNN

3.1.17 Summary

The method we used in this study is the use of a convolutional neural network to recognize timber species using digital images of wood surfaces. We have tested the GLCM technique with the same dataset of timber but failed to get accurate results for some species as the color of wood surfaces are somewhat similar and GLCM is mainly focusing on the texture of wood. To distinguish between species more accurately, we have chosen a deep learning technique which is specially designed for image recognition. The convolutional neural network is using filters to analyze an image hence require little pre-processing. The rest of the document will explain the data, results, and the conclusions we gathered by analyzing the wood images using a CNN.

Chapter 4

Outcome of the timber recognition system

4.1 Introduction

This chapter presents the technologies to be used with the proposed solution and to discuss the various approaches to address the main problem statement. The automated timber recognition system is capable of identifying different timber types which are input by the users as captured images. This would be implemented as a mobile application which provides users with the certain functionalities to get informative feedbacks on different wood types and identifying timer species. The proposed system will be presented by highlighting hypothesis, input, output, process, users and the features of the Automated Timber recognition System. We hypothesize that the issue of not having a proper mechanism to recognize various timber types in an accurate manner irrespective of different obstacles and defects of the wood species. This process of correctly identifying the timber type is a major issue when the trees are cut done into timber boards, seasoned and varnished. The visual based system needs to recognize the base timber by eliminating all the above-mentioned obstacles.

The proposed system is to be implemented as a mobile application. The input to the application will be the images captured via the camera of the mobile phone. Once the images are sent through the proposed mobile application, the color and the texture of the timber is sent through the convolution neural network model to identify the type of the timber from the images of the timber available in the database. For the identification of timber species, we are going to evaluate different approaches such as GLCM feature with artificial neural network, SIFT & SURF for image recognition, convolutional neural network and select the best possible approach for the classification.

4.3 Input

For the implementation of the application, the images of different types of timbers were captured to build the model and extract other details. The collected data set includes approximately more than 500 images for one timber specification and there are 17 timber types altogether. All the images were captured within the same context and using the same equipment and other dependencies. The following table 4.1 shows the quantities and names of the collected timer types.

Table 4. 1: Total number of images in each timber classes

Timber Name	# of images
Albisiya	896
Attoniya	1000
Burutha	1098
Gammalu	498
Garnishh	780
Ginisapu	578
Hora	904
Kumbuk	878
Koss	1361
Kaluwara	906
Paramara	787
Mahogani	2668
Rathu kumbuk	33
Sapu	1058
Suriyamara	1188
Tekka	1754
Walkohomba	1127

4.4 Output

The main output of this research is a mobile application which is capable of recognizing timber types by classifying the input images. The captured images are recognized using the classification model and the identified types of the species are displayed to the user. The users are also capable of viewing details of available timber types along with wood images. In summary, the output of the timber recognition system includes the main components of Timber classifier, Mobile UI, Back end database.

4.5 Process

The captured images of several timber types were used to build the classifier. The images were sent through several preprocessing steps prior to the classification and convolutional neural network was used for the classification. So, the timber classification module acts as a mediator between mobile UI and the back end of the application. The system database keeps the information and captured images of the various timber types for users to view. So, the final outcome will be developed as a mobile application to recognize different wood species which make users to easily identify timber types.

4.6 Users

The identification of timber types can be useful in several primary and secondary industrial users of wood, government agencies, and museums, as well as to users' educational and scientific fields. This system can be used by any user who is willing to identify the different wood types.

4.7 Features

Apart from the main functionalities of the prosed solution, the system includes other nonfunctional features to improve the quality of the application. The system is developed in to several key modules such as user interface, timber classification and back end of the system. The system also has other functionalities such as

- Higher performance
- Low Cost
- Expandability
- Independence of operators

4.8 Summary

This chapter discussed the main approaches for the proposed timber recognition application including the main hypothesis of the research, input, output, users, and processes and features of the system. The research approach was defined by addressing the main drawbacks of timber recognition methods today. So, the automated timber recognition system provides an accurate mechanism to recognize various timber types in an accurate manner.

Chapter 5

Design and implementation

5.1 Introduction

In this chapter, we discuss implementation details about the system and module mentions. This presents the software and algorithms used in each module with the output.

5.2 Overall implementation

An overall solution of implementing the mobile application is to build it in Android application running on any mobile device. The implementation of the mobile application used the training model to train the data and identify the test data.

The automated timber recognition system maintains a separated set of interfaces to input timber images, recognize timber types and view the details of the different timber species. A database was also included within the system to keep the sample images for various timber types and other relevant details of the timbers such as its scientific name. The top-level architecture of the system is shown below.

5.2.1 Top level architecture

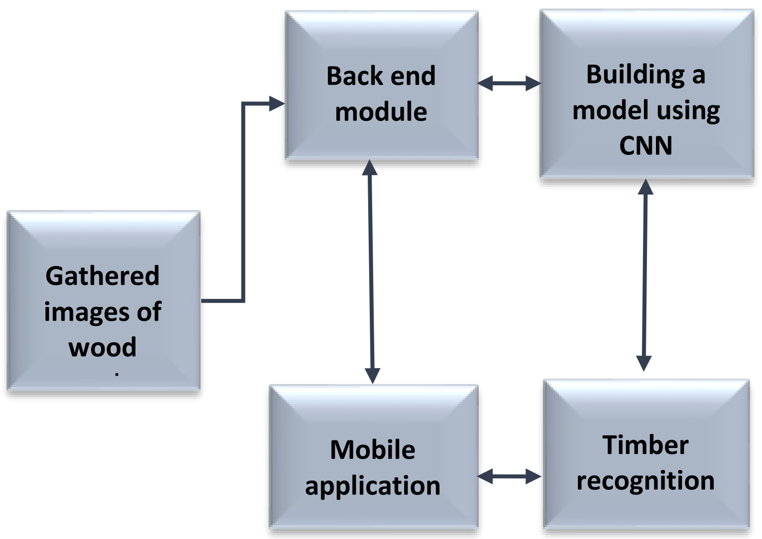


Figure 5. 1: High - level architecture for the proposed solution

As shown in the above figure, the data was collected from different sawmills as images and keep the captured data along with other details for each type of timber. The preprocessed data was used to build the timber classifier and the created model was evaluated and tested for the accuracy and different other factors. As the core module of the system is the timber classification, it acts as a mediator between application front end and back end to classify the various timber species by responding according to the received requests.

5.2.2 Timber classification

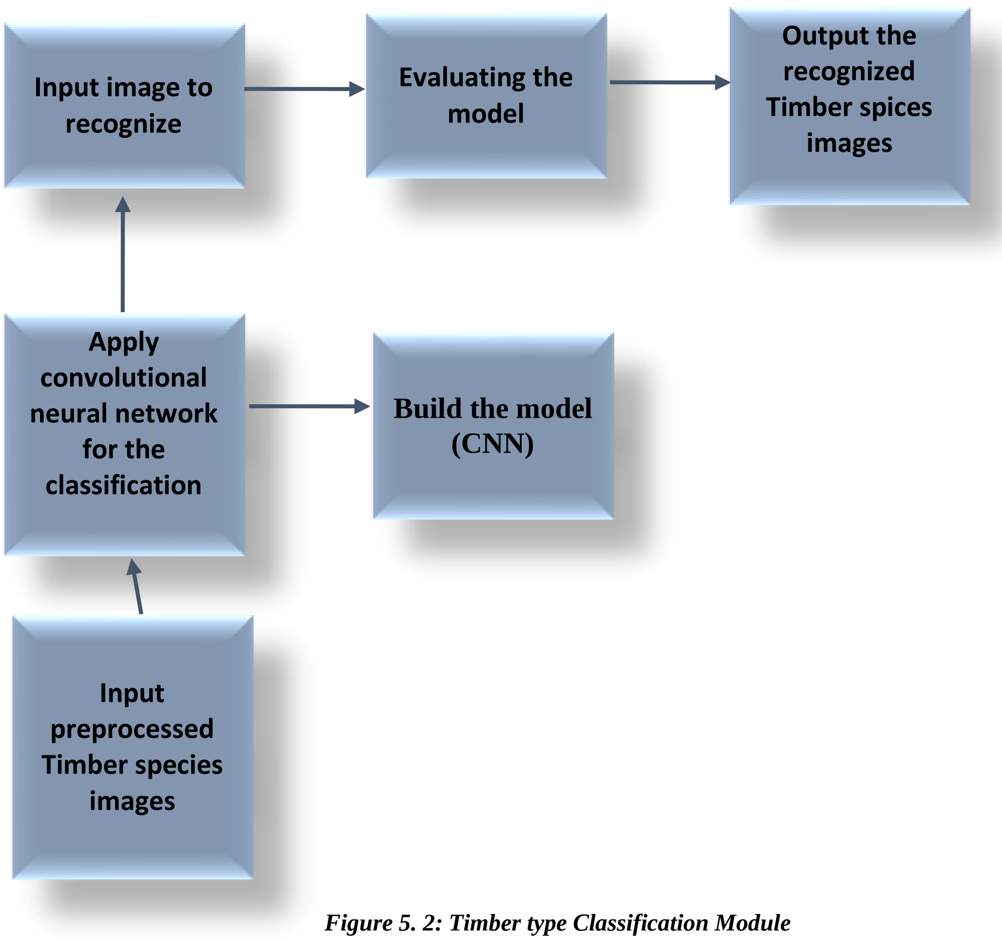

Input image to recognize
Evaluating the model
Output the recognized Timber spices images
Apply convolutional neural network for the classification
Build the model (CNN)
Input preprocessed Timber species images

Figure 5. 2: Timber type Classification Module

As the key module of the system, the timber classification performs various stages in classification such as preprocessing, feature selection, classification and finally the evaluating the results of the outcome. The following figure shows the main phases of the timber classification which is used in the proposed system to differentiate various timber types. Based on the request comes from the front end, the model can classify the input image, or the system is also capable of displaying details of various timber types according to the system request. When a user needs to view the figures or details of different timber types, the information can be derived from the database and display to the user. If the user needs to identify a wood type, they can capture the image using a mobile camera by selecting the scan timber option. Then the timber classification module will evaluate the model results from the classifier to recognize the timber type and display the final outcome to the user. In general, the system provides proper timber identification mechanism by classifying the input image.

5.2.3 Data collection

In the system used 17000 data for 70% training and 30% for testing. Chapter 3 We discussed more details about the data collection. Figure 3.4 show the number of classes we added recognition. We created a new data set for this research. There are some limitations also we followed to create the data sets. Ex: size of the image, lighting, quality of the image etc. Then we preprocess the data and used it into the created model.

5.2.4 Implementation of user interfaces

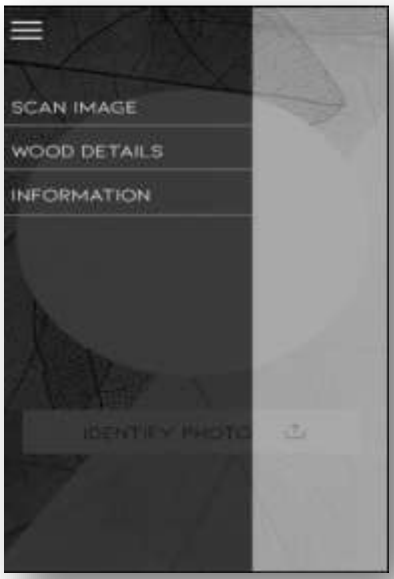


Figure 5. 3: Mobile application interface1

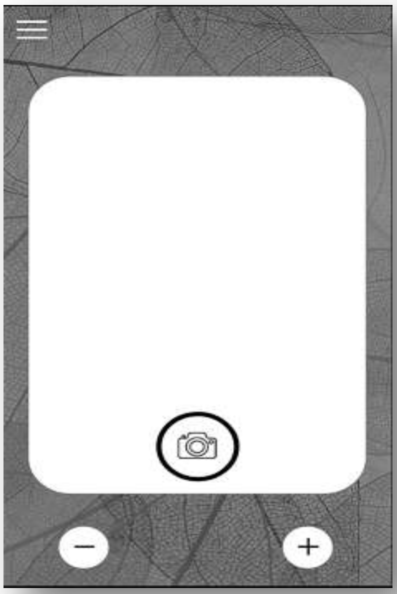


Figure 5.3. a: Mobile application interface 2

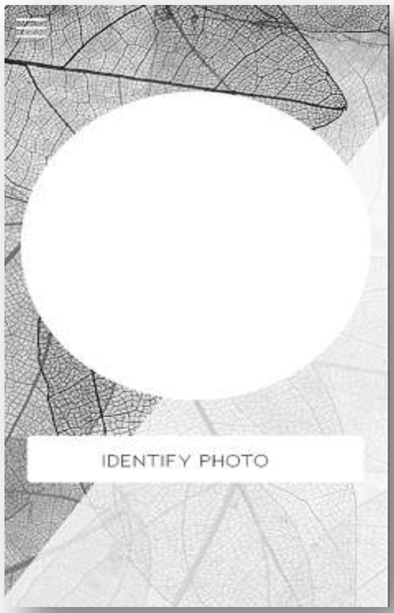


Figure 5.3. b: Mobile application interface3

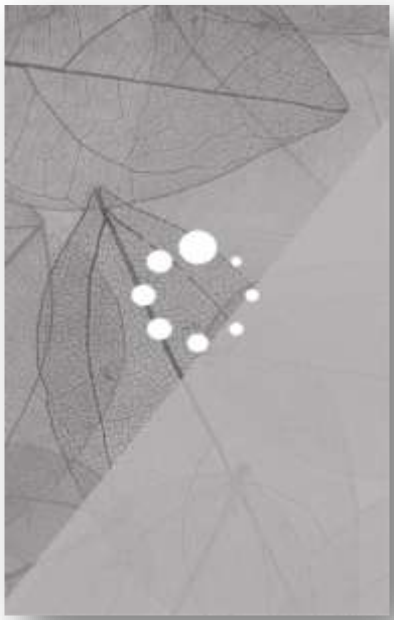


Figure 5.3. c: Mobile application interface4

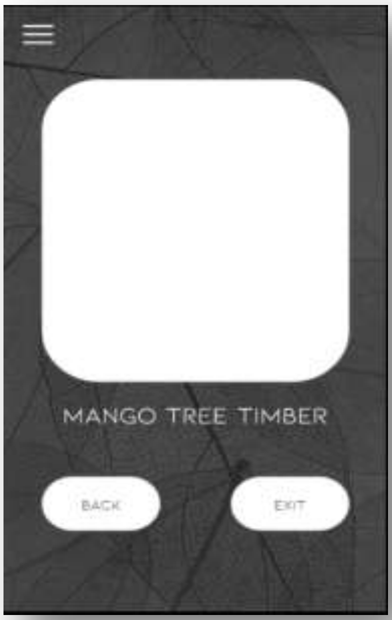


Figure 5.3. d: Mobile application interfaces5

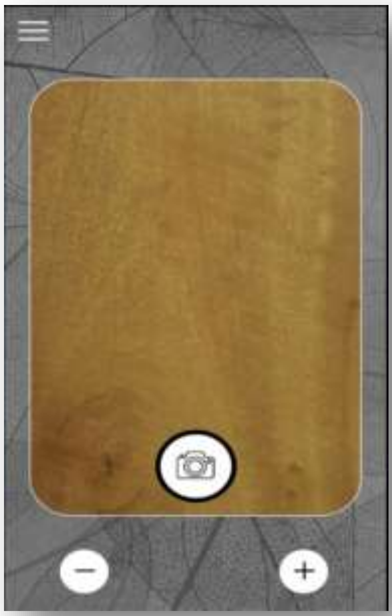


Figure 5.3. e: Mobile application interface6

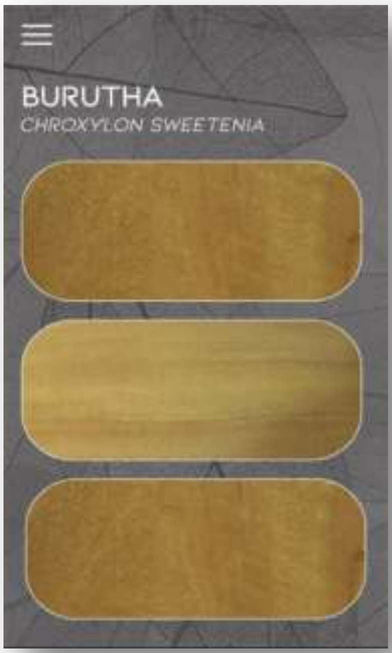


Figure 5.3. f: Mobile application interface7

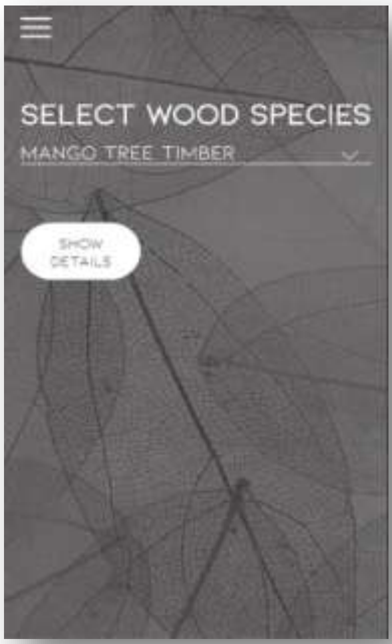


Figure 5.3. g: Mobile application interface8

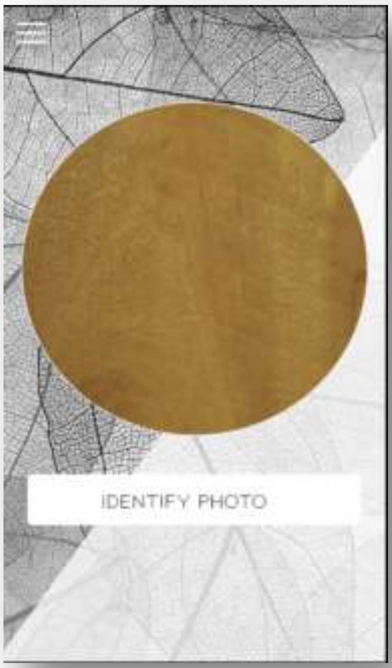


Figure 5.3. h: Mobile application9

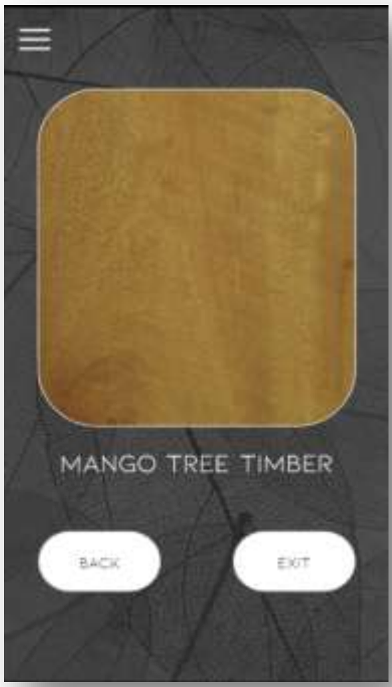


Figure 5.3. i: Mobile application 10

Figure 5.3 illustrates the function of the Android application. Figure 5.3.a is the enable of the camera to take photos and send it into Figure 5.3.b. This interface shows the image and has a button click to identify the image. Figure 5.3.c shows the loading image and Figure 5.3.d shows the type of image if anyone wants to go back or cancel the image identification. When we click the back button and go forward into Figure 5.3.e to select wood details, it shows Figure 5.3.f and button view of the details. Figure 5.3.g and Figure 5.3.i shows the sample piece images and shows the different images of the same category.

5.2.5 Implementation of timber recognition system

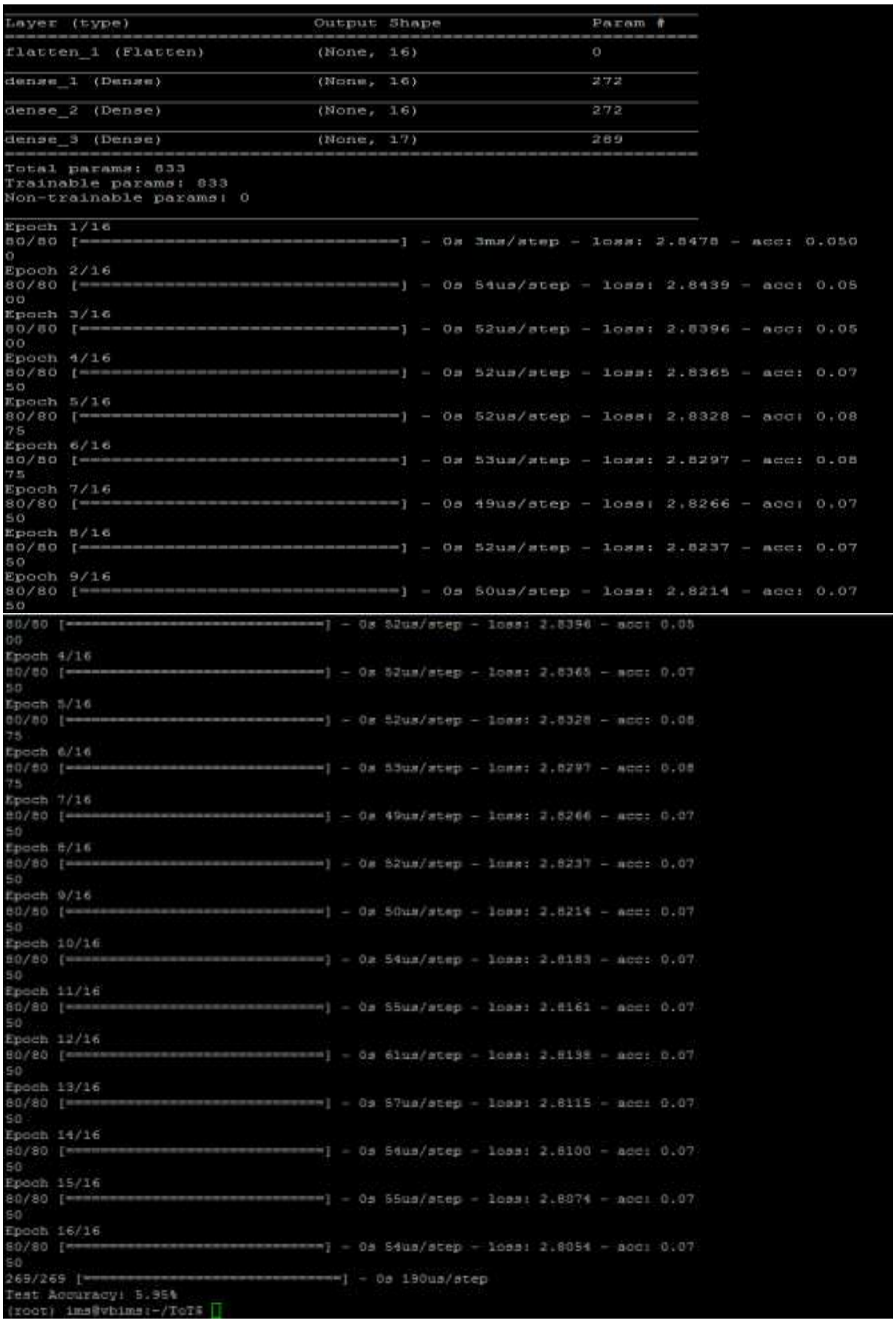
5.2.5.1 Gray-Level Co-Occurrence Matrix (GLCM)

Gray-Level Co-Occurrence Matrix statistically analysis the texture by considering the spatial relationship of each pixel. The texture of an image is distinguished by calculating the occurrence of pairs of pixels with a certain value in a defined spatial relationship. Then the statistical measurements are extracted from the GLCM matrix such as contrast, correlation. Energy and homogeneity. The contrast measures the local variations in the GLCM matrix and correlation measures the joint probability of defined pixel pairs. The energy gives the sum of a squared element in the GLCM and homogeneity evaluates how far the elements are within the GLCM to the GLCM diagonal.

To derive the GLCM features, sharpen of the input images were improved by adjusting the brightness and contrast of the images. Then the images were converted to grayscale and HSV images to obtain certain properties after resizing them into appropriate dimensions. After generating the grey matrix, the relevant properties were defined as energy, ASM (Angular Second Moment or Energy) and homogeneity to extract GLCM features. The average hue values for each image were also generated and normalized to include to the matrix. So, the matrix for each image contains energy, ASM, homogeneity and average hue of images.

GLCM features were extracted for all the images in several wood types and created an Artificial Neural Network (ANN) model with the obtained features of the training data set. The test accuracies and predictions were obtained by considering the most probable label for the test image based on extracted features. The accuracies and the predictions of the generated model based on GLCM statistics are as follows.

The GLCM feature extraction points and train the key points in images and find accuracy. The accuracy was 5% which is not sufficient for this research. Therefore, we used 12260 training images and testing 5254 images.



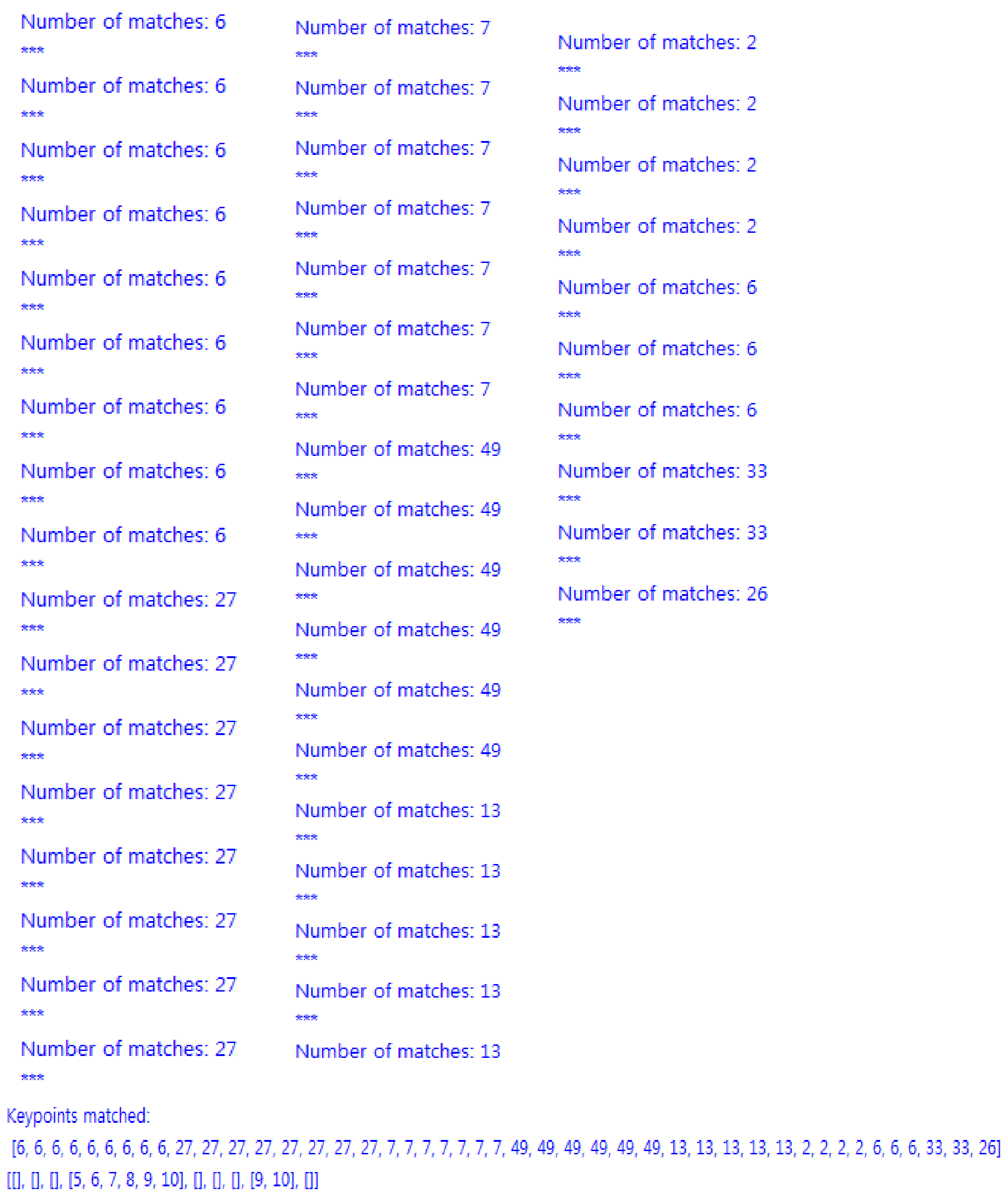
```
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 16)                0
_________________________________________________________________
dense_1 (Dense)              (None, 16)                272
_________________________________________________________________
dense_2 (Dense)              (None, 16)                272
_________________________________________________________________
dense_3 (Dense)              (None, 17)                289
=================================================================
Total params: 833
Trainable params: 833
Non-trainable params: 0
_________________________________________________________________
Epoch 1/16
80/80 [==============================] - 0s 3ms/step - loss: 2.8478 - acc: 0.050
0
Epoch 2/16
80/80 [==============================] - 0s 54us/step - loss: 2.8439 - acc: 0.05
00
Epoch 3/16
80/80 [==============================] - 0s 52us/step - loss: 2.8396 - acc: 0.05
00
Epoch 4/16
80/80 [==============================] - 0s 52us/step - loss: 2.8365 - acc: 0.07
50
Epoch 5/16
80/80 [==============================] - 0s 52us/step - loss: 2.8328 - acc: 0.08
75
Epoch 6/16
80/80 [==============================] - 0s 53us/step - loss: 2.8297 - acc: 0.08
75
Epoch 7/16
80/80 [==============================] - 0s 49us/step - loss: 2.8266 - acc: 0.07
50
Epoch 8/16
80/80 [==============================] - 0s 52us/step - loss: 2.8237 - acc: 0.07
50
Epoch 9/16
80/80 [==============================] - 0s 50us/step - loss: 2.8214 - acc: 0.07
50
```

```
80/80 [==============================] - 0s 52us/step - loss: 2.8396 - acc: 0.05
00
Epoch 4/16
80/80 [==============================] - 0s 52us/step - loss: 2.8365 - acc: 0.07
50
Epoch 5/16
80/80 [==============================] - 0s 52us/step - loss: 2.8328 - acc: 0.08
75
Epoch 6/16
80/80 [==============================] - 0s 53us/step - loss: 2.8297 - acc: 0.08
75
Epoch 7/16
80/80 [==============================] - 0s 49us/step - loss: 2.8266 - acc: 0.07
50
Epoch 8/16
80/80 [==============================] - 0s 52us/step - loss: 2.8237 - acc: 0.07
50
Epoch 9/16
80/80 [==============================] - 0s 50us/step - loss: 2.8214 - acc: 0.07
50
```

```
Epoch 10/16
80/80 [==============================] - 0s 54us/step - loss: 2.8183 - acc: 0.07
50
Epoch 11/16
80/80 [==============================] - 0s 55us/step - loss: 2.8161 - acc: 0.07
50
Epoch 12/16
80/80 [==============================] - 0s 61us/step - loss: 2.8138 - acc: 0.07
50
Epoch 13/16
80/80 [==============================] - 0s 57us/step - loss: 2.8115 - acc: 0.07
50
Epoch 14/16
80/80 [==============================] - 0s 54us/step - loss: 2.8100 - acc: 0.07
50
Epoch 15/16
80/80 [==============================] - 0s 55us/step - loss: 2.8074 - acc: 0.07
50
Epoch 16/16
80/80 [==============================] - 0s 54us/step - loss: 2.8054 - acc: 0.07
50
269/269 [==============================] - 0s 190us/step
Test Accuracy: 5.95%
(root) ims@vbims:~/IoT$
```

Figure 5. 4: Results of GLCM extract the feature using ANN

5.2.5.2 SIFT & SURF



```
Number of matches: 6
***
Number of matches: 6
***
Number of matches: 6
***
Number of matches: 6
***
Number of matches: 6
***
Number of matches: 6
***
Number of matches: 6
***
Number of matches: 6
***
Number of matches: 6
***
Number of matches: 27
***
Number of matches: 27
***
Number of matches: 27
***
Number of matches: 27
***
Number of matches: 27
***
Number of matches: 27
***
Number of matches: 27
***
Number of matches: 27
***
Number of matches: 7
***
Number of matches: 7
***
Number of matches: 7
***
Number of matches: 7
***
Number of matches: 7
***
Number of matches: 7
***
Number of matches: 7
***
Number of matches: 49
***
Number of matches: 49
***
Number of matches: 49
***
Number of matches: 49
***
Number of matches: 49
***
Number of matches: 49
***
Number of matches: 13
***
Number of matches: 13
***
Number of matches: 13
***
Number of matches: 13
***
Number of matches: 13
Number of matches: 2
***
Number of matches: 2
***
Number of matches: 2
***
Number of matches: 2
***
Number of matches: 6
***
Number of matches: 6
***
Number of matches: 6
***
Number of matches: 33
***
Number of matches: 33
***
Number of matches: 26
***
Keypoints matched:
 [6, 6, 6, 6, 6, 6, 6, 6, 6, 27, 27, 27, 27, 27, 27, 27, 27, 7, 7, 7, 7, 7, 7, 7, 49, 49, 49, 49, 49, 49, 13, 13, 13, 13, 13, 2, 2, 2, 2, 6, 6, 6, 33, 33, 26]
[[], [], [], [5, 6, 7, 8, 9, 10], [], [], [], [9, 10], []]
```

Figure 5. 5 : Identifying the key points using SURF

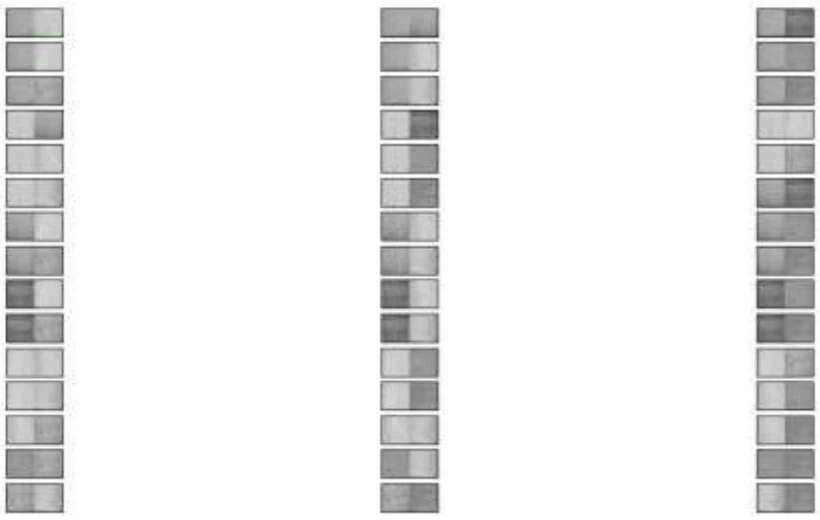


Figure 5. 6: Compare each image matching the key points.

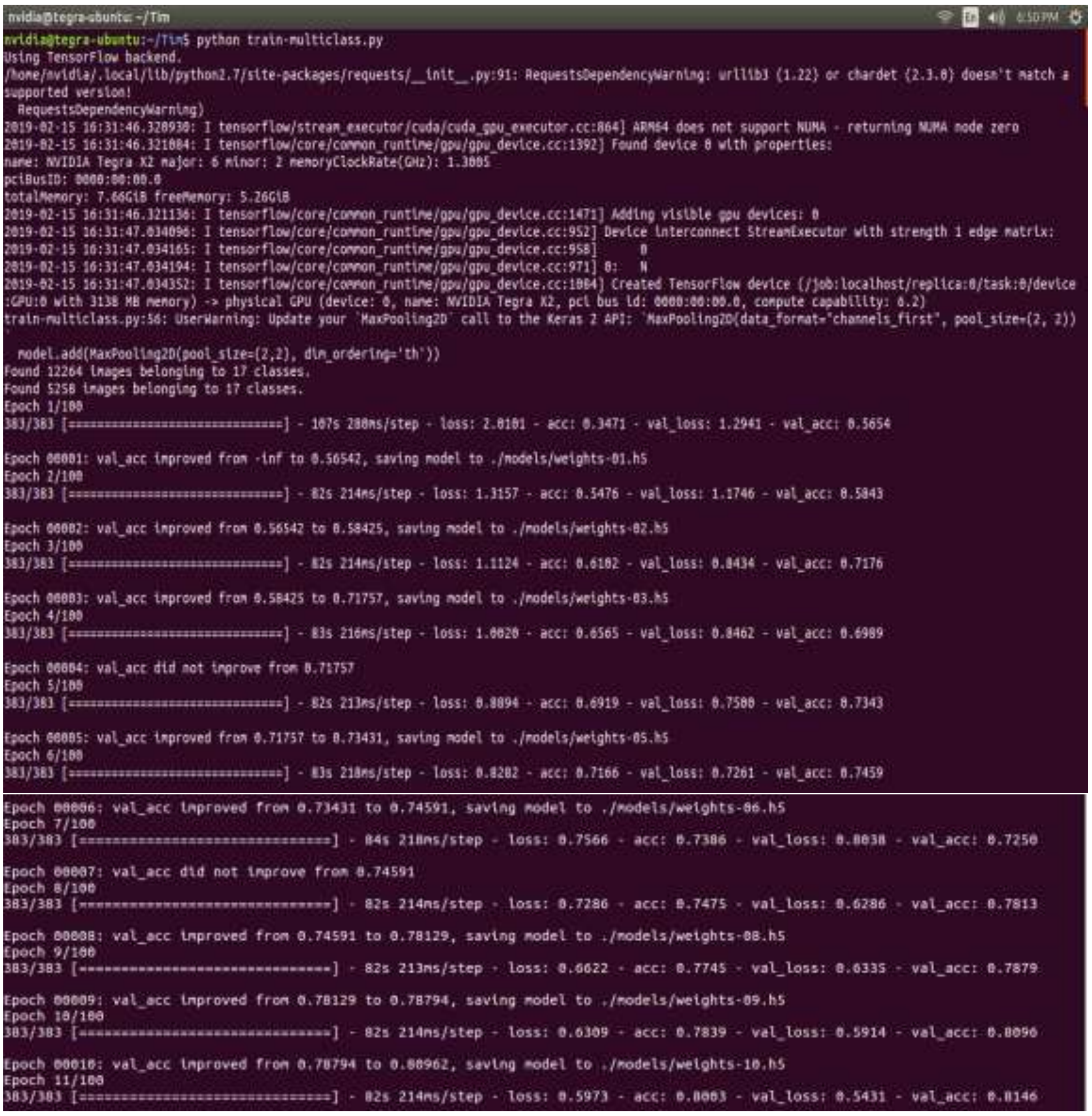
Scale-Invariant Feature Transform (SIFT) was used as one of the key methods to identify local features of the wood species images. This was implemented by transforming images into scale-invariant coordinates in local features. This generates a considerable number of features, which covers the image densely in complete locations of the image. The first step was to identify key locations in scale spaces by finding the local maxima or minima using the difference of Gaussian function. Then a feature vector was generated to describe the local regions of the images. The achieved invariances to local variation were called SIFT keys and they were used in identifying object modules using k nearest-neighbor approach.

SURF or Speeded Up Robust Features uses integral images to generate key points and descriptors in an efficient manner. This mainly developed in two stages for key point detection and key point description. The integral images were based on Laplacian of Gaussian images with a box filter. The determinants of the Hessian matrix were used to detect the key points of an image. So, the SIRF built the scale space by using the same image size and varying the filters. So, SURF key points define the pixels that have average intensities which greatly differs from the immediate neighbors. The descriptor establishes the relation between the key points and the obtained neighboring key points. After both SURF descriptors and key points are calculated, they were compared using k nearest neighbor algorithm.

Both approaches were implemented for the recognition of wood species and SURF has given a considerable level of accuracy compared to the SIFT. Even though the SURF was accurate in wood recognition compared to SIFT, in this case, it did not show a high level of performance in wood recognition as the identified key points were not enough for the comparison. The figures of identified key points are in different wood species are shown below.

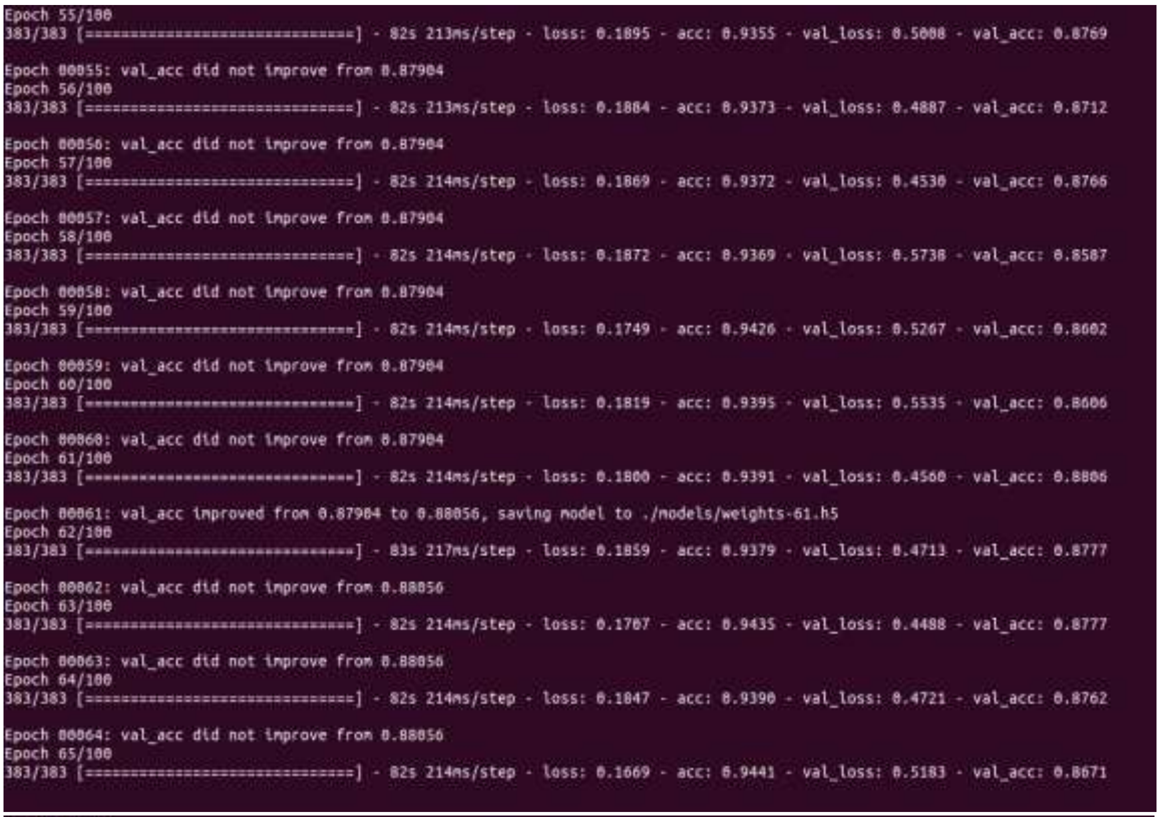
5.2.5.3 CNN training and testing

In timber recognition system needs some more valuable data set to training the model. It is a very difficult task. Because no more indigenous data set to find. Then we have to create with limitations. When we used the CNN, the major task is training the data sets that used in machine learning techniques to train the images. Using the all captured timber images data set and the training the model using CNN. In this task 17514 amount of data used in the training and testing data. The training data set is 12260 (70%) and 5254 (30%) for training including 17 timber classes.

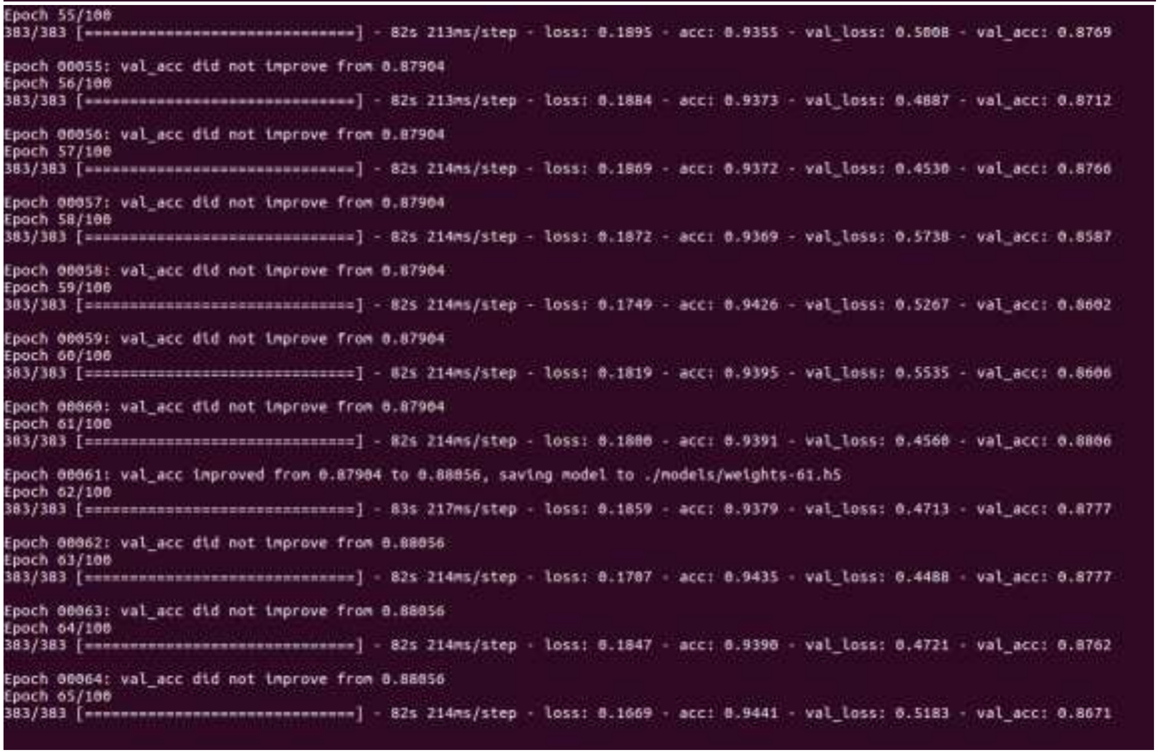


```
nvidia@tegra-ubuntu: ~/Tim
nvidia@tegra-ubuntu:~/Tim$ python train-multiclass.py
Using TensorFlow backend.
/home/nvidia/.local/lib/python2.7/site-packages/requests/__init__.py:91: RequestsDependencyWarning: urllib3 (1.22) or chardet (2.3.0) doesn't match a
supported version!
  RequestsDependencyWarning)
2019-02-15 16:31:46.320930: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:864] ARM64 does not support NUMA - returning NUMA node zero
2019-02-15 16:31:46.321084: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1392] Found device 0 with properties:
name: NVIDIA Tegra X2 major: 6 minor: 2 memoryClockRate(GHz): 1.3005
pciBusID: 0000:00:00.0
totalMemory: 7.66GiB freeMemory: 5.26GiB
2019-02-15 16:31:46.321136: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1471] Adding visible gpu devices: 0
2019-02-15 16:31:47.034096: I tensorflow/core/common_runtime/gpu/gpu_device.cc:952] Device interconnect StreamExecutor with strength 1 edge matrix:
2019-02-15 16:31:47.034165: I tensorflow/core/common_runtime/gpu/gpu_device.cc:958]      0
2019-02-15 16:31:47.034194: I tensorflow/core/common_runtime/gpu/gpu_device.cc:971] 0:   N
2019-02-15 16:31:47.034352: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1084] Created TensorFlow device (/job:localhost/replica:0/task:0/device
:GPU:0 with 3138 MB memory) -> physical GPU (device: 0, name: NVIDIA Tegra X2, pci bus id: 0000:00:00.0, compute capability: 6.2)
train-multiclass.py:56: UserWarning: Update your `MaxPooling2D` call to the Keras 2 API: `MaxPooling2D(data_format="channels_first", pool_size=(2, 2))
`
  model.add(MaxPooling2D(pool_size=(2,2), dim_ordering='th'))
Found 12264 images belonging to 17 classes.
Found 5258 images belonging to 17 classes.
Epoch 1/100
383/383 [==============================] - 107s 280ms/step - loss: 2.0101 - acc: 0.3471 - val_loss: 1.2941 - val_acc: 0.5654

Epoch 00001: val_acc improved from -inf to 0.56542, saving model to ./models/weights-01.h5
Epoch 2/100
383/383 [==============================] - 82s 214ms/step - loss: 1.3157 - acc: 0.5476 - val_loss: 1.1746 - val_acc: 0.5843

Epoch 00002: val_acc improved from 0.56542 to 0.58425, saving model to ./models/weights-02.h5
Epoch 3/100
383/383 [==============================] - 82s 214ms/step - loss: 1.1124 - acc: 0.6102 - val_loss: 0.8434 - val_acc: 0.7176

Epoch 00003: val_acc improved from 0.58425 to 0.71757, saving model to ./models/weights-03.h5
Epoch 4/100
383/383 [==============================] - 83s 216ms/step - loss: 1.0020 - acc: 0.6565 - val_loss: 0.8462 - val_acc: 0.6989

Epoch 00004: val_acc did not improve from 0.71757
Epoch 5/100
383/383 [==============================] - 82s 213ms/step - loss: 0.8894 - acc: 0.6919 - val_loss: 0.7580 - val_acc: 0.7343

Epoch 00005: val_acc improved from 0.71757 to 0.73431, saving model to ./models/weights-05.h5
Epoch 6/100
383/383 [==============================] - 83s 218ms/step - loss: 0.8282 - acc: 0.7166 - val_loss: 0.7261 - val_acc: 0.7459
Epoch 00006: val_acc improved from 0.73431 to 0.74591, saving model to ./models/weights-06.h5
Epoch 7/100
383/383 [==============================] - 84s 218ms/step - loss: 0.7566 - acc: 0.7386 - val_loss: 0.8038 - val_acc: 0.7250

Epoch 00007: val_acc did not improve from 0.74591
Epoch 8/100
383/383 [==============================] - 82s 214ms/step - loss: 0.7286 - acc: 0.7475 - val_loss: 0.6286 - val_acc: 0.7813

Epoch 00008: val_acc improved from 0.74591 to 0.78129, saving model to ./models/weights-08.h5
Epoch 9/100
383/383 [==============================] - 82s 213ms/step - loss: 0.6622 - acc: 0.7745 - val_loss: 0.6335 - val_acc: 0.7879

Epoch 00009: val_acc improved from 0.78129 to 0.78794, saving model to ./models/weights-09.h5
Epoch 10/100
383/383 [==============================] - 82s 214ms/step - loss: 0.6309 - acc: 0.7839 - val_loss: 0.5914 - val_acc: 0.8096

Epoch 00010: val_acc improved from 0.78794 to 0.80962, saving model to ./models/weights-10.h5
Epoch 11/100
383/383 [==============================] - 82s 214ms/step - loss: 0.5973 - acc: 0.8003 - val_loss: 0.5431 - val_acc: 0.8146
```

Figure 5. 7: CNN training data screen shots for 100 epoch

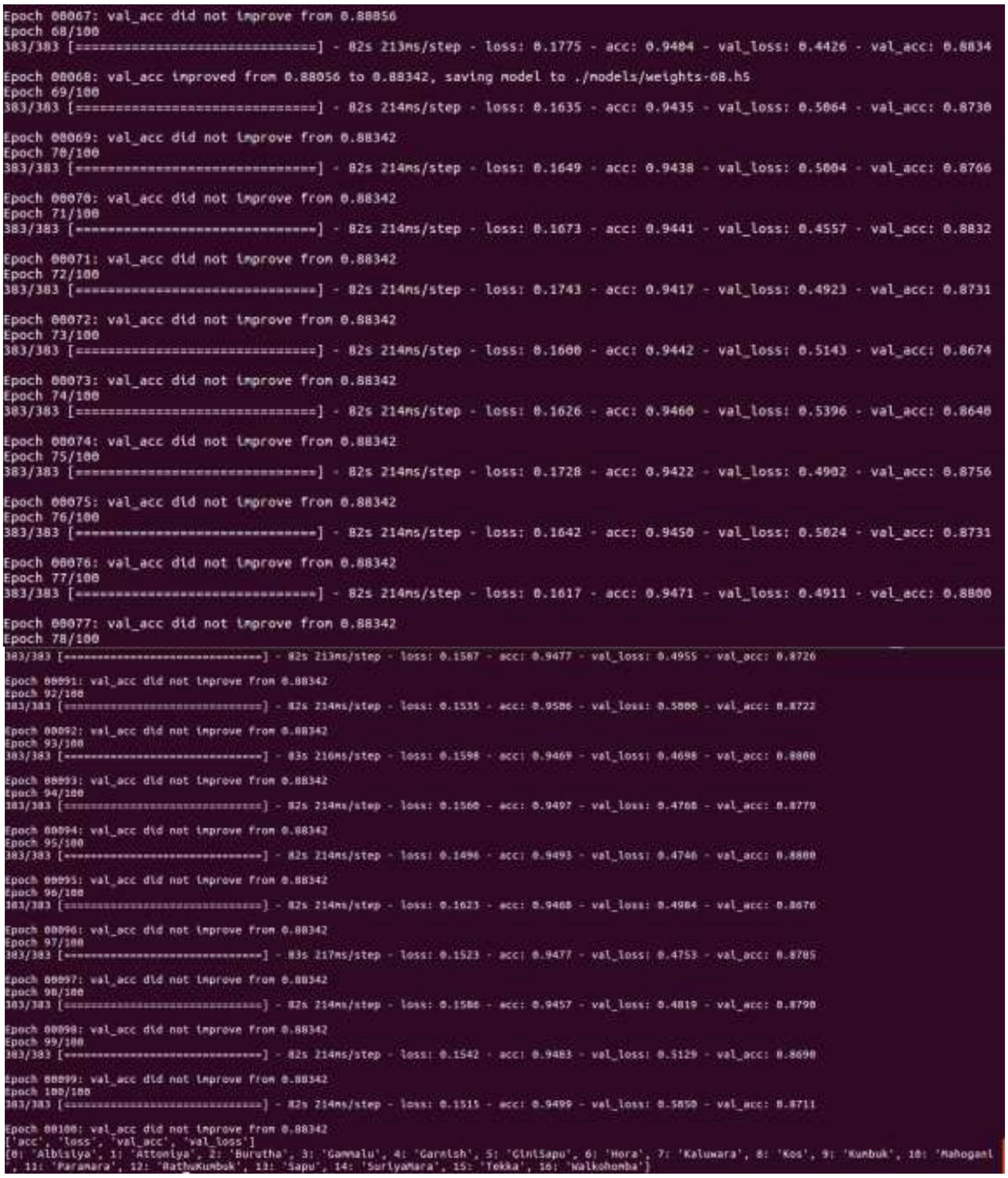


```
Epoch 55/100
383/383 [==============================] - 82s 213ms/step - loss: 0.1895 - acc: 0.9355 - val_loss: 0.5008 - val_acc: 0.8769

Epoch 00055: val_acc did not improve from 0.87904
Epoch 56/100
383/383 [==============================] - 82s 213ms/step - loss: 0.1884 - acc: 0.9373 - val_loss: 0.4887 - val_acc: 0.8712

Epoch 00056: val_acc did not improve from 0.87904
Epoch 57/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1869 - acc: 0.9372 - val_loss: 0.4530 - val_acc: 0.8766

Epoch 00057: val_acc did not improve from 0.87904
Epoch 58/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1872 - acc: 0.9369 - val_loss: 0.5738 - val_acc: 0.8587

Epoch 00058: val_acc did not improve from 0.87904
Epoch 59/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1749 - acc: 0.9426 - val_loss: 0.5267 - val_acc: 0.8602

Epoch 00059: val_acc did not improve from 0.87904
Epoch 60/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1819 - acc: 0.9395 - val_loss: 0.5535 - val_acc: 0.8606

Epoch 00060: val_acc did not improve from 0.87904
Epoch 61/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1800 - acc: 0.9391 - val_loss: 0.4560 - val_acc: 0.8806

Epoch 00061: val_acc improved from 0.87904 to 0.88056, saving model to ./models/weights-61.h5
Epoch 62/100
383/383 [==============================] - 83s 217ms/step - loss: 0.1859 - acc: 0.9379 - val_loss: 0.4713 - val_acc: 0.8777

Epoch 00062: val_acc did not improve from 0.88056
Epoch 63/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1707 - acc: 0.9435 - val_loss: 0.4488 - val_acc: 0.8777

Epoch 00063: val_acc did not improve from 0.88056
Epoch 64/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1847 - acc: 0.9390 - val_loss: 0.4721 - val_acc: 0.8762

Epoch 00064: val_acc did not improve from 0.88056
Epoch 65/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1669 - acc: 0.9441 - val_loss: 0.5183 - val_acc: 0.8671
```



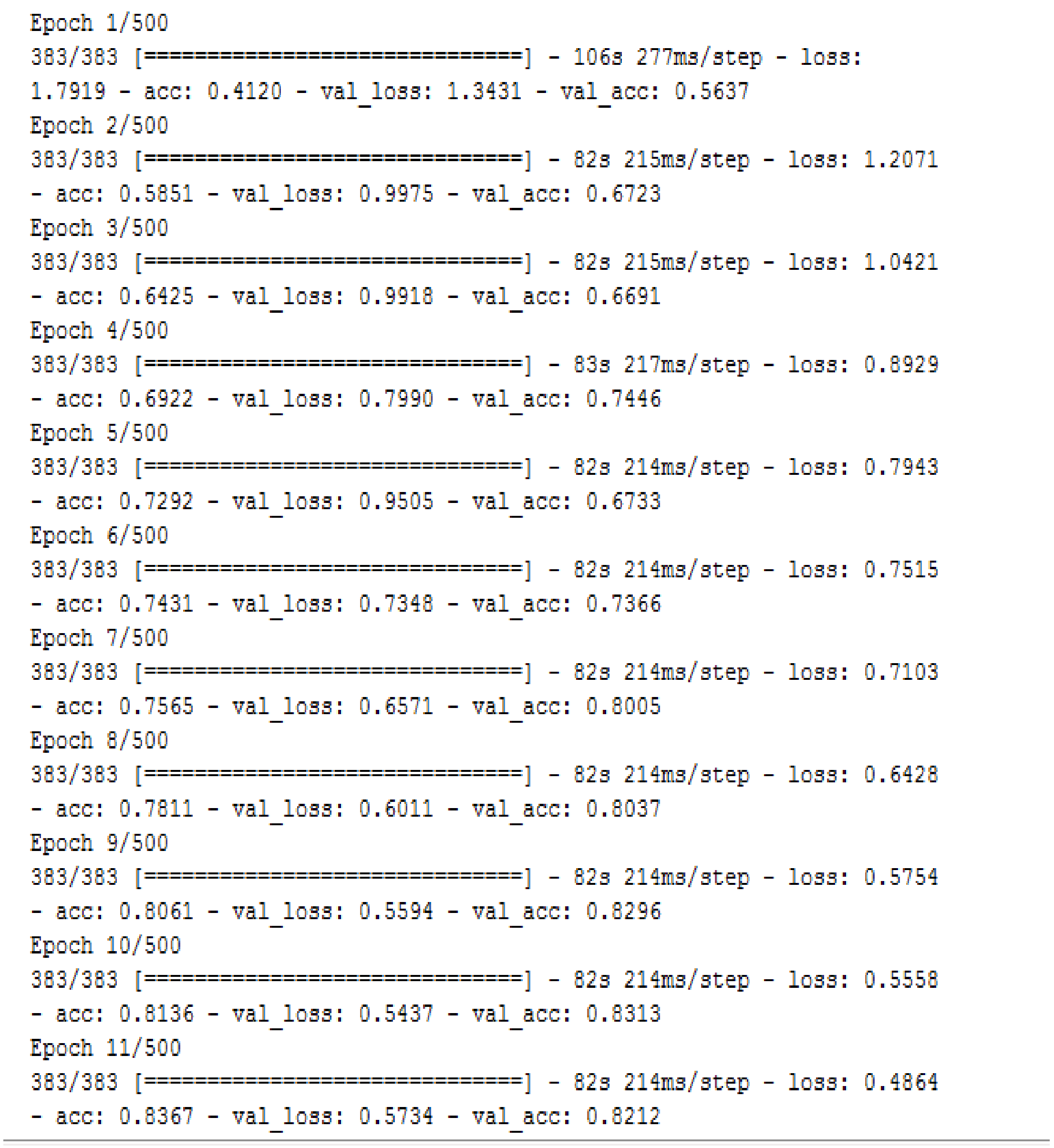
```
Epoch 55/100
383/383 [==============================] - 82s 213ms/step - loss: 0.1895 - acc: 0.9355 - val_loss: 0.5008 - val_acc: 0.8769

Epoch 00055: val_acc did not improve from 0.87904
Epoch 56/100
383/383 [==============================] - 82s 213ms/step - loss: 0.1884 - acc: 0.9373 - val_loss: 0.4887 - val_acc: 0.8712

Epoch 00056: val_acc did not improve from 0.87904
Epoch 57/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1869 - acc: 0.9372 - val_loss: 0.4530 - val_acc: 0.8766

Epoch 00057: val_acc did not improve from 0.87904
Epoch 58/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1872 - acc: 0.9369 - val_loss: 0.5738 - val_acc: 0.8587

Epoch 00058: val_acc did not improve from 0.87904
Epoch 59/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1749 - acc: 0.9426 - val_loss: 0.5267 - val_acc: 0.8602

Epoch 00059: val_acc did not improve from 0.87904
Epoch 60/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1819 - acc: 0.9395 - val_loss: 0.5535 - val_acc: 0.8606

Epoch 00060: val_acc did not improve from 0.87904
Epoch 61/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1800 - acc: 0.9391 - val_loss: 0.4560 - val_acc: 0.8806

Epoch 00061: val_acc improved from 0.87904 to 0.88056, saving model to ./models/weights-61.h5
Epoch 62/100
383/383 [==============================] - 83s 217ms/step - loss: 0.1859 - acc: 0.9379 - val_loss: 0.4713 - val_acc: 0.8777

Epoch 00062: val_acc did not improve from 0.88056
Epoch 63/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1707 - acc: 0.9435 - val_loss: 0.4488 - val_acc: 0.8777

Epoch 00063: val_acc did not improve from 0.88056
Epoch 64/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1847 - acc: 0.9390 - val_loss: 0.4721 - val_acc: 0.8762

Epoch 00064: val_acc did not improve from 0.88056
Epoch 65/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1669 - acc: 0.9441 - val_loss: 0.5183 - val_acc: 0.8671
```

Figure 5. 8: CNN training data screen shots for 100 epoch Cont...



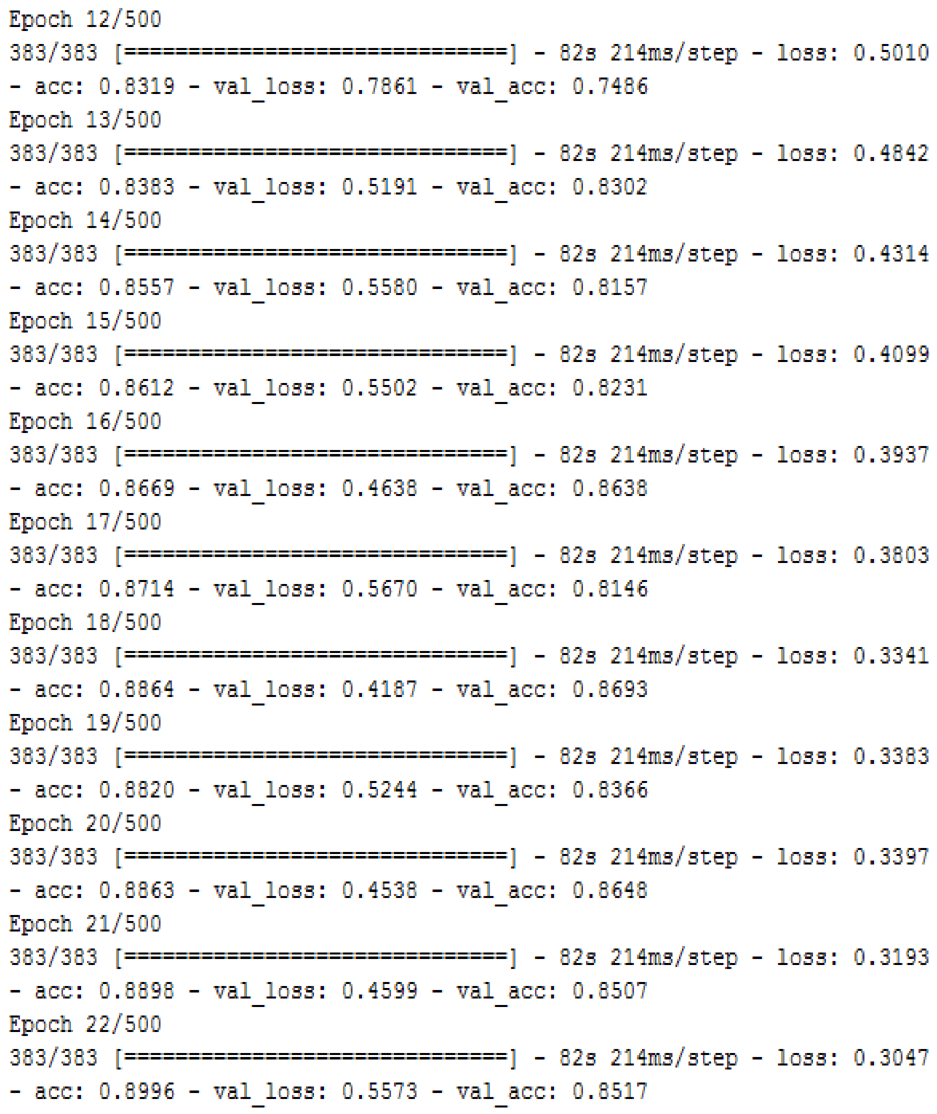
```
Epoch 00067: val_acc did not improve from 0.88056
Epoch 68/100
383/383 [==============================] - 82s 213ms/step - loss: 0.1775 - acc: 0.9404 - val_loss: 0.4426 - val_acc: 0.8834

Epoch 00068: val_acc improved from 0.88056 to 0.88342, saving model to ./models/weights-68.h5
Epoch 69/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1635 - acc: 0.9435 - val_loss: 0.5064 - val_acc: 0.8730

Epoch 00069: val_acc did not improve from 0.88342
Epoch 70/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1649 - acc: 0.9438 - val_loss: 0.5004 - val_acc: 0.8766

Epoch 00070: val_acc did not improve from 0.88342
Epoch 71/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1673 - acc: 0.9441 - val_loss: 0.4557 - val_acc: 0.8832

Epoch 00071: val_acc did not improve from 0.88342
Epoch 72/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1743 - acc: 0.9417 - val_loss: 0.4923 - val_acc: 0.8731

Epoch 00072: val_acc did not improve from 0.88342
Epoch 73/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1600 - acc: 0.9442 - val_loss: 0.5143 - val_acc: 0.8674

Epoch 00073: val_acc did not improve from 0.88342
Epoch 74/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1626 - acc: 0.9460 - val_loss: 0.5396 - val_acc: 0.8640

Epoch 00074: val_acc did not improve from 0.88342
Epoch 75/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1728 - acc: 0.9422 - val_loss: 0.4902 - val_acc: 0.8756

Epoch 00075: val_acc did not improve from 0.88342
Epoch 76/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1642 - acc: 0.9450 - val_loss: 0.5024 - val_acc: 0.8731

Epoch 00076: val_acc did not improve from 0.88342
Epoch 77/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1617 - acc: 0.9471 - val_loss: 0.4911 - val_acc: 0.8800

Epoch 00077: val_acc did not improve from 0.88342
Epoch 78/100
383/383 [==============================] - 82s 213ms/step - loss: 0.1587 - acc: 0.9477 - val_loss: 0.4955 - val_acc: 0.8726

Epoch 00091: val_acc did not improve from 0.88342
Epoch 92/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1535 - acc: 0.9506 - val_loss: 0.5800 - val_acc: 0.8722

Epoch 00092: val_acc did not improve from 0.88342
Epoch 93/100
383/383 [==============================] - 83s 216ms/step - loss: 0.1598 - acc: 0.9469 - val_loss: 0.4698 - val_acc: 0.8808

Epoch 00093: val_acc did not improve from 0.88342
Epoch 94/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1560 - acc: 0.9497 - val_loss: 0.4708 - val_acc: 0.8779

Epoch 00094: val_acc did not improve from 0.88342
Epoch 95/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1496 - acc: 0.9493 - val_loss: 0.4746 - val_acc: 0.8800

Epoch 00095: val_acc did not improve from 0.88342
Epoch 96/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1623 - acc: 0.9468 - val_loss: 0.4984 - val_acc: 0.8676

Epoch 00096: val_acc did not improve from 0.88342
Epoch 97/100
383/383 [==============================] - 83s 217ms/step - loss: 0.1523 - acc: 0.9477 - val_loss: 0.4753 - val_acc: 0.8785

Epoch 00097: val_acc did not improve from 0.88342
Epoch 98/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1586 - acc: 0.9457 - val_loss: 0.4819 - val_acc: 0.8790

Epoch 00098: val_acc did not improve from 0.88342
Epoch 99/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1542 - acc: 0.9483 - val_loss: 0.5129 - val_acc: 0.8690

Epoch 00099: val_acc did not improve from 0.88342
Epoch 100/100
383/383 [==============================] - 82s 214ms/step - loss: 0.1515 - acc: 0.9499 - val_loss: 0.5850 - val_acc: 0.8711

Epoch 00100: val_acc did not improve from 0.88342
['acc', 'loss', 'val_acc', 'val_loss']
{0: 'Albisiya', 1: 'Attoniya', 2: 'Burutha', 3: 'Gammalu', 4: 'Garnish', 5: 'GiniSapu', 6: 'Hora', 7: 'Kaluwara', 8: 'Kos', 9: 'Kumbuk', 10: 'Mahogani
', 11: 'Paramara', 12: 'RathuKumbuk', 13: 'Sapu', 14: 'SuriyaMara', 15: 'Tekka', 16: 'Walkohomba'}
```

Figure 5. 9: CNN training data accuracy



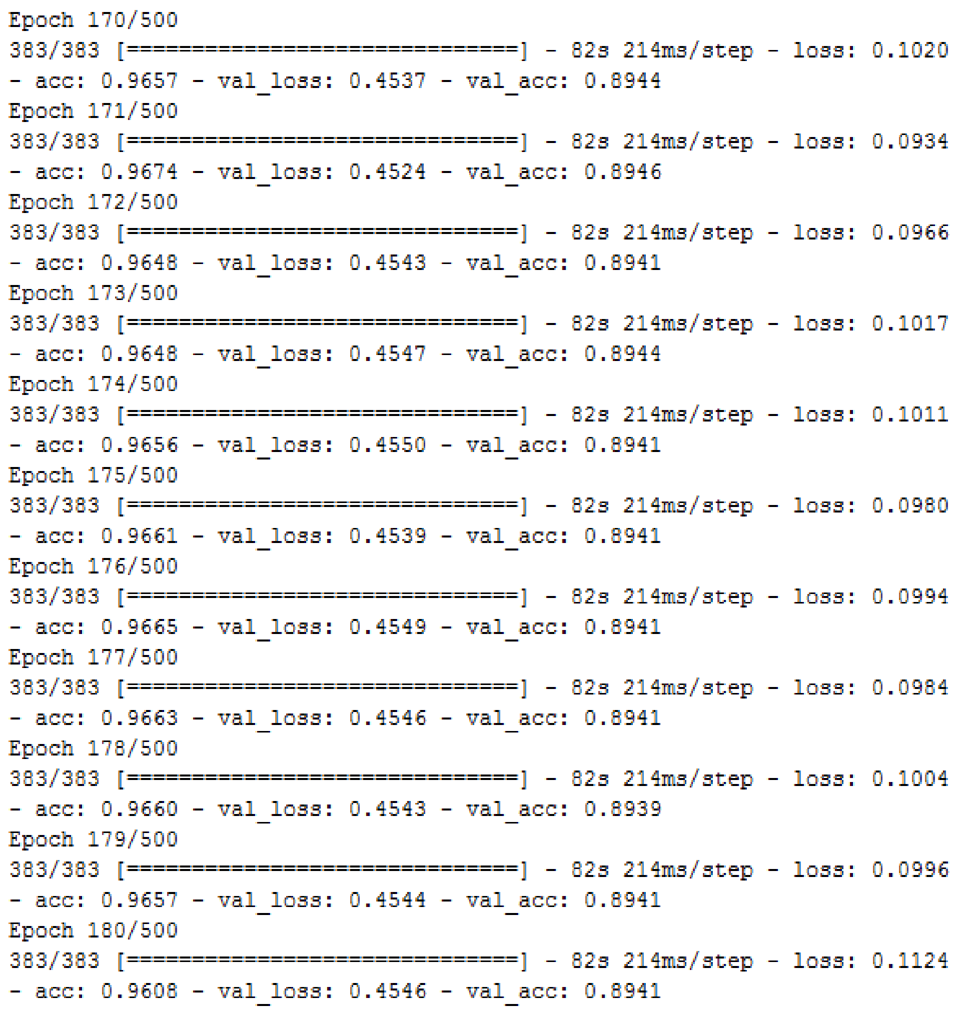
```
Epoch 1/500
383/383 [==============================] - 106s 277ms/step - loss:
1.7919 - acc: 0.4120 - val_loss: 1.3431 - val_acc: 0.5637
Epoch 2/500
383/383 [==============================] - 82s 215ms/step - loss: 1.2071
- acc: 0.5851 - val_loss: 0.9975 - val_acc: 0.6723
Epoch 3/500
383/383 [==============================] - 82s 215ms/step - loss: 1.0421
- acc: 0.6425 - val_loss: 0.9918 - val_acc: 0.6691
Epoch 4/500
383/383 [==============================] - 83s 217ms/step - loss: 0.8929
- acc: 0.6922 - val_loss: 0.7990 - val_acc: 0.7446
Epoch 5/500
383/383 [==============================] - 82s 214ms/step - loss: 0.7943
- acc: 0.7292 - val_loss: 0.9505 - val_acc: 0.6733
Epoch 6/500
383/383 [==============================] - 82s 214ms/step - loss: 0.7515
- acc: 0.7431 - val_loss: 0.7348 - val_acc: 0.7366
Epoch 7/500
383/383 [==============================] - 82s 214ms/step - loss: 0.7103
- acc: 0.7565 - val_loss: 0.6571 - val_acc: 0.8005
Epoch 8/500
383/383 [==============================] - 82s 214ms/step - loss: 0.6428
- acc: 0.7811 - val_loss: 0.6011 - val_acc: 0.8037
Epoch 9/500
383/383 [==============================] - 82s 214ms/step - loss: 0.5754
- acc: 0.8061 - val_loss: 0.5594 - val_acc: 0.8296
Epoch 10/500
383/383 [==============================] - 82s 214ms/step - loss: 0.5558
- acc: 0.8136 - val_loss: 0.5437 - val_acc: 0.8313
Epoch 11/500
383/383 [==============================] - 82s 214ms/step - loss: 0.4864
- acc: 0.8367 - val_loss: 0.5734 - val_acc: 0.8212
```

Figure 5. 10: CNN training data screen shots for 500 epoch continue ….



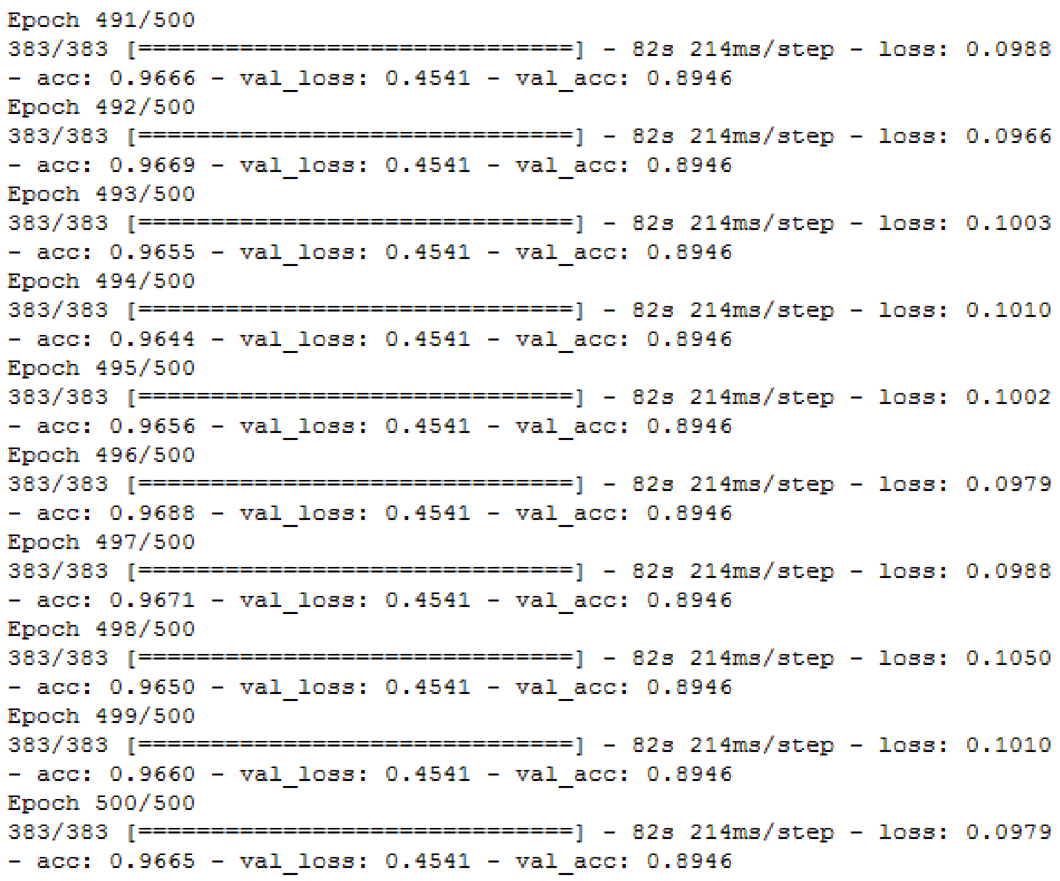
```
Epoch 12/500
383/383 [==============================] - 82s 214ms/step - loss: 0.5010
- acc: 0.8319 - val_loss: 0.7861 - val_acc: 0.7486
Epoch 13/500
383/383 [==============================] - 82s 214ms/step - loss: 0.4842
- acc: 0.8383 - val_loss: 0.5191 - val_acc: 0.8302
Epoch 14/500
383/383 [==============================] - 82s 214ms/step - loss: 0.4314
- acc: 0.8557 - val_loss: 0.5580 - val_acc: 0.8157
Epoch 15/500
383/383 [==============================] - 82s 214ms/step - loss: 0.4099
- acc: 0.8612 - val_loss: 0.5502 - val_acc: 0.8231
Epoch 16/500
383/383 [==============================] - 82s 214ms/step - loss: 0.3937
- acc: 0.8669 - val_loss: 0.4638 - val_acc: 0.8638
Epoch 17/500
383/383 [==============================] - 82s 214ms/step - loss: 0.3803
- acc: 0.8714 - val_loss: 0.5670 - val_acc: 0.8146
Epoch 18/500
383/383 [==============================] - 82s 214ms/step - loss: 0.3341
- acc: 0.8864 - val_loss: 0.4187 - val_acc: 0.8693
Epoch 19/500
383/383 [==============================] - 82s 214ms/step - loss: 0.3383
- acc: 0.8820 - val_loss: 0.5244 - val_acc: 0.8366
Epoch 20/500
383/383 [==============================] - 82s 214ms/step - loss: 0.3397
- acc: 0.8863 - val_loss: 0.4538 - val_acc: 0.8648
Epoch 21/500
383/383 [==============================] - 82s 214ms/step - loss: 0.3193
- acc: 0.8898 - val_loss: 0.4599 - val_acc: 0.8507
Epoch 22/500
383/383 [==============================] - 82s 214ms/step - loss: 0.3047
- acc: 0.8996 - val_loss: 0.5573 - val_acc: 0.8517
```

Figure 5. 11: CNN training data screen shots for 500 epoch continue ….



```
Epoch 170/500
383/383 [==============================] - 82s 214ms/step - loss: 0.1020
- acc: 0.9657 - val_loss: 0.4537 - val_acc: 0.8944
Epoch 171/500
383/383 [==============================] - 82s 214ms/step - loss: 0.0934
- acc: 0.9674 - val_loss: 0.4524 - val_acc: 0.8946
Epoch 172/500
383/383 [==============================] - 82s 214ms/step - loss: 0.0966
- acc: 0.9648 - val_loss: 0.4543 - val_acc: 0.8941
Epoch 173/500
383/383 [==============================] - 82s 214ms/step - loss: 0.1017
- acc: 0.9648 - val_loss: 0.4547 - val_acc: 0.8944
Epoch 174/500
383/383 [==============================] - 82s 214ms/step - loss: 0.1011
- acc: 0.9656 - val_loss: 0.4550 - val_acc: 0.8941
Epoch 175/500
383/383 [==============================] - 82s 214ms/step - loss: 0.0980
- acc: 0.9661 - val_loss: 0.4539 - val_acc: 0.8941
Epoch 176/500
383/383 [==============================] - 82s 214ms/step - loss: 0.0994
- acc: 0.9665 - val_loss: 0.4549 - val_acc: 0.8941
Epoch 177/500
383/383 [==============================] - 82s 214ms/step - loss: 0.0984
- acc: 0.9663 - val_loss: 0.4546 - val_acc: 0.8941
Epoch 178/500
383/383 [==============================] - 82s 214ms/step - loss: 0.1004
- acc: 0.9660 - val_loss: 0.4543 - val_acc: 0.8939
Epoch 179/500
383/383 [==============================] - 82s 214ms/step - loss: 0.0996
- acc: 0.9657 - val_loss: 0.4544 - val_acc: 0.8941
Epoch 180/500
383/383 [==============================] - 82s 214ms/step - loss: 0.1124
- acc: 0.9608 - val_loss: 0.4546 - val_acc: 0.8941
```

Figure 5. 12: CNN training data screen shots for 500 epoch continue ….



```
Epoch 491/500
383/383 [==============================] - 82s 214ms/step - loss: 0.0988
- acc: 0.9666 - val_loss: 0.4541 - val_acc: 0.8946
Epoch 492/500
383/383 [==============================] - 82s 214ms/step - loss: 0.0966
- acc: 0.9669 - val_loss: 0.4541 - val_acc: 0.8946
Epoch 493/500
383/383 [==============================] - 82s 214ms/step - loss: 0.1003
- acc: 0.9655 - val_loss: 0.4541 - val_acc: 0.8946
Epoch 494/500
383/383 [==============================] - 82s 214ms/step - loss: 0.1010
- acc: 0.9644 - val_loss: 0.4541 - val_acc: 0.8946
Epoch 495/500
383/383 [==============================] - 82s 214ms/step - loss: 0.1002
- acc: 0.9656 - val_loss: 0.4541 - val_acc: 0.8946
Epoch 496/500
383/383 [==============================] - 82s 214ms/step - loss: 0.0979
- acc: 0.9688 - val_loss: 0.4541 - val_acc: 0.8946
Epoch 497/500
383/383 [==============================] - 82s 214ms/step - loss: 0.0988
- acc: 0.9671 - val_loss: 0.4541 - val_acc: 0.8946
Epoch 498/500
383/383 [==============================] - 82s 214ms/step - loss: 0.1050
- acc: 0.9650 - val_loss: 0.4541 - val_acc: 0.8946
Epoch 499/500
383/383 [==============================] - 82s 214ms/step - loss: 0.1010
- acc: 0.9660 - val_loss: 0.4541 - val_acc: 0.8946
Epoch 500/500
383/383 [==============================] - 82s 214ms/step - loss: 0.0979
- acc: 0.9665 - val_loss: 0.4541 - val_acc: 0.8946
```

Figure 5. 13: CNN training data screen shots for 500 epoch continue ….

5.3 Summary

This chapter provides an overall implementation of the android phone application user interfaces with the data collection and the back end of the module we described. Next chapter is evaluating and disuse about the system.

Chapter 6

Results

6.1 Results

The convolutional neural network is tested with a dataset of 17 timber species where there are many images taken per species. We have gathered digital images from 17 different indigenous trees of Sri Lanka as described in table 1.1. The number of training and testing images are split into 12260 and 5254 respectively.

The dataset is available in figure 3.2 and table 3.2.

The framework we used to implement the neural network API is Keras; written in Python. As Keras does not work by itself, we used backend for low-level operations namely TensorFlow. The development environment is machine learning. As we are following a supervised machine learning approach, we trained the neural network with trained data and created a model. In order to create such a model, we followed the steps of building a model: model construction, model training, model testing, and model evaluation.

6.2 Testing

6.2.1 Model construction

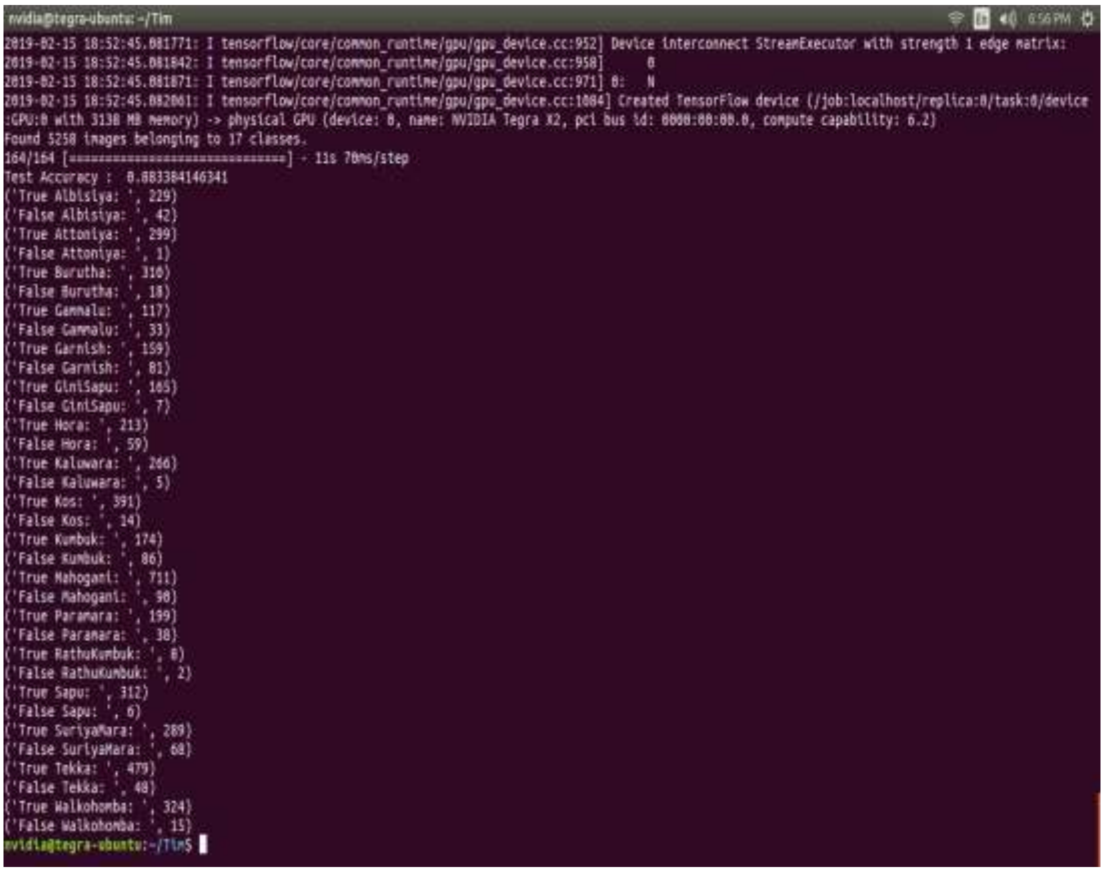
We have used CNN developed with Keras using TensorFlow as backend. The CNN consist of 2 convolutional layers and one pool layer. The convolutional layers are in the size of 3 and 2 respectively. The pooling setting is max-pooling with 2x2 receptive fields.

6.2.2 Model training

The number of training data 12260. This number of data set taken 70% in all the data set. In one class represent more than 500 images for each.

6.2.3 Model testing

The number of testing data 5254. It’s taken from 30% for the data sets. The testing data should be varying from the training data set. Then it is worth for take the good accuracy.



```
nvidia@tegra-ubuntu: ~/Tim
2019-02-15 18:52:45.081771: I tensorflow/core/common_runtime/gpu/gpu_device.cc:952] Device interconnect StreamExecutor with strength 1 edge matrix:
2019-02-15 18:52:45.081842: I tensorflow/core/common_runtime/gpu/gpu_device.cc:958]      0
2019-02-15 18:52:45.081871: I tensorflow/core/common_runtime/gpu/gpu_device.cc:971] 0:   N
2019-02-15 18:52:45.082061: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1084] Created TensorFlow device (/job:localhost/replica:0/task:0/device
:GPU:0 with 3138 MB memory) -> physical GPU (device: 0, name: NVIDIA Tegra X2, pci bus id: 0000:00:00.0, compute capability: 6.2)
Found 5258 images belonging to 17 classes.
164/164 [==============================] - 11s 70ms/step
Test Accuracy :  0.883384146341
('True Albisiya: ', 229)
('False Albisiya: ', 42)
('True Attoniya: ', 299)
('False Attoniya: ', 1)
('True Burutha: ', 310)
('False Burutha: ', 18)
('True Gammalu: ', 117)
('False Gammalu: ', 33)
('True Garnish: ', 159)
('False Garnish: ', 81)
('True GiniSapu: ', 165)
('False GiniSapu: ', 7)
('True Hora: ', 213)
('False Hora: ', 59)
('True Kaluwara: ', 266)
('False Kaluwara: ', 5)
('True Kos: ', 391)
('False Kos: ', 14)
('True Kumbuk: ', 174)
('False Kumbuk: ', 86)
('True Mahogani: ', 711)
('False Mahogani: ', 98)
('True Paramara: ', 199)
('False Paramara: ', 38)
('True RathuKumbuk: ', 0)
('False RathuKumbuk: ', 2)
('True Sapu: ', 312)
('False Sapu: ', 6)
('True SuriyaMara: ', 289)
('False SuriyaMara: ', 68)
('True Tekka: ', 479)
('False Tekka: ', 48)
('True Walkohomba: ', 324)
('False Walkohomba: ', 15)
nvidia@tegra-ubuntu:~/Tim$
```

Figure 6. 1: CNN Validation results

Table 6. 1: Results of the timber classes

Timber Name	# of images	Training 70%	Testing 30%	TRUE	FALSE
Albisiya	896	625	271	229	42
Attoniya	1000	700	300	299	1
Burutha	1098	770	328	310	18
Gammalu	500	350	150	122	28
Garnishh	780	540	240	158	82
Ginisapu	578	406	172	165	7
Hora	904	632	272	213	59
Kumbuk	878	618	260	179	81
Koss	1361	956	405	391	14
Kaluwara	906	635	271	267	4
Paramara	787	550	237	206	31
Mahogani	2668	1867	801	720	81
Rathu kumbuk	33	23	10	7	3
Sapu	1058	740	318	314	4
Suriyamara	1188	831	357	299	58
Tekka	1754	1227	527	481	46
Walkohomba	1127	788	339	326	13

6.2.4 Model evaluation

The Image Data Generator class in Keras increases the number of images as needed because of the arguments it supports such as rotation range which gives random rations from 0 to 180, width and height shifts, random zooming, horizontal flip and many more. The convolutional layers are alternate with nonlinear layers (Relu) and pooling layers. The activation function we used is Relu and Max Pooling 2D layer is the pooling operation which we have used in the

The two graphs shown in figure 6.2 and figure 6.3 are on the accuracy of the neural network classification.

The first figure 6.2 shows the dependence of the evaluation accuracy and the validation accuracy on the number of epochs. The second plot figure 6.3 shows the dependence of loss function on the number of epochs.

When observing the first plot, we can see that the high accuracy over 90% is achieved after 40th epoch or iteration for training data. In subsequent epochs on the plot, the accuracy does not improve and maintain a consistent value.

The validation accuracy which is above 88% is important as the validation is tested for a new sample of data which re never been seen in the training dataset. The validation accuracy shows the ability of the model to generalize to new data. If the training data accuracy keeps improving while validation data accuracy gets worse, the model is likely in an overfitting situation, i.e. the model starts to basically just memorize the data. But in the model testing for training and validation data, it shows that there is no such overfitting happens and validation accuracy get increased gradually.

The model loss is illustrated in figure 6.2. The loss function is important in neural networks in order to calculate the inconsistency between the predicted value and the actual value of a dataset. The loss function value is not a percentage like an accuracy but a non-negative value, where the robustness of the model gets increased with the decrement of the loss function value. As figure 6.2 shows the loss value is decreased after each iteration of epoch/ iteration in the model which denotes a successful model building. The loss function which we have used in the study is the categorical cross entropy formula as below:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}\mathbf{1}_{y_i \in C_c}\log p_{model}[y_i \in C_c]$$

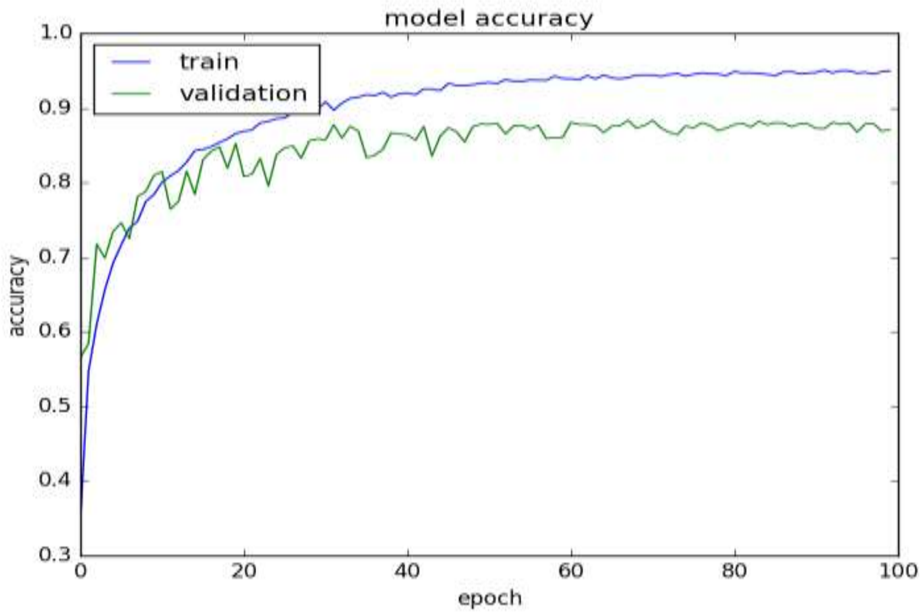


Figure 6. 2: Model accuracy in 100 epoch

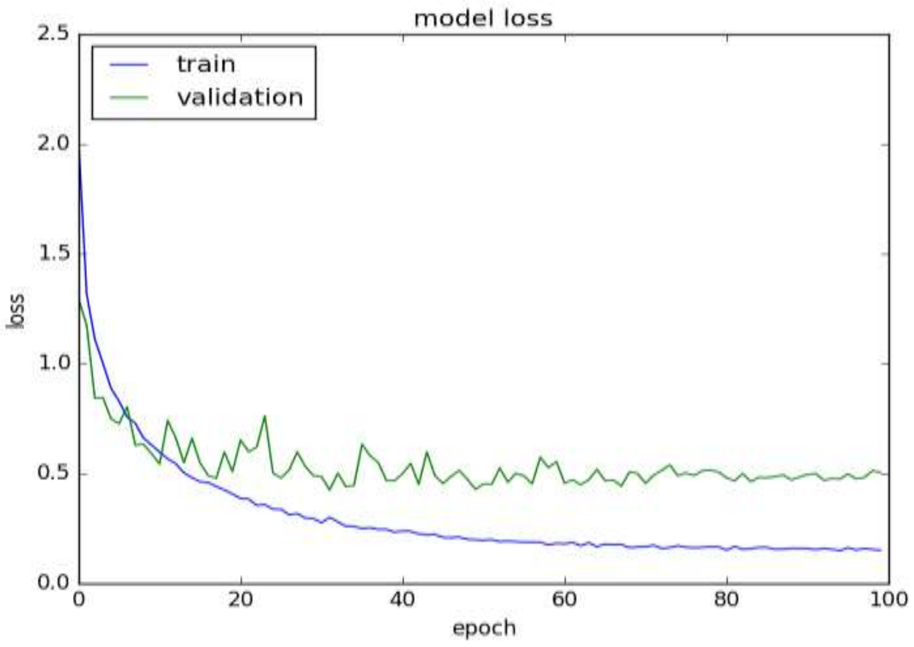


Figure 6. 3: Model loss accuracy 100 epoch

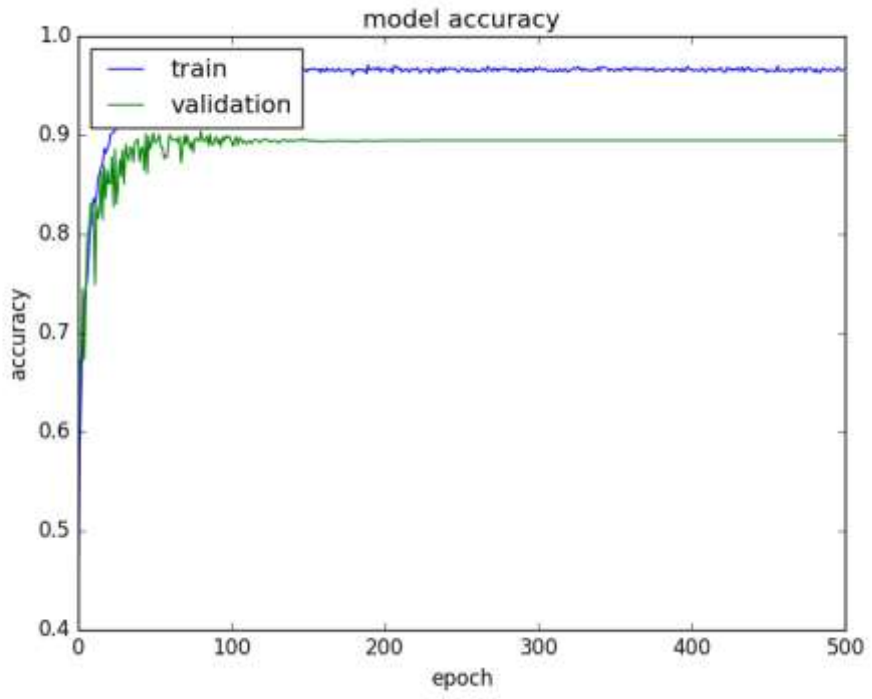


Figure 6. 4: Model accuracy for 500 epoch

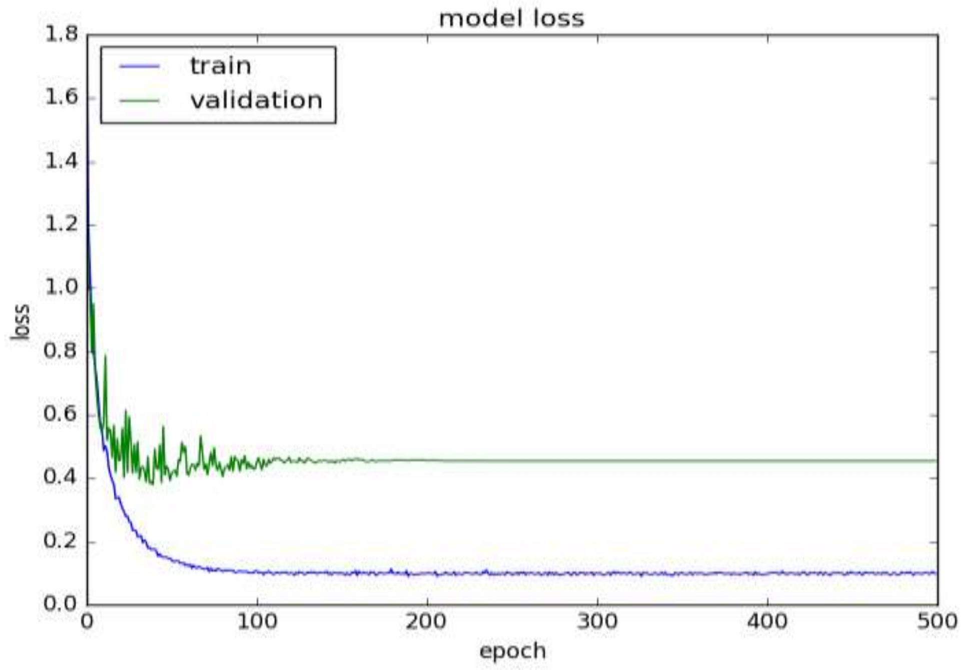


Figure 6. 5: Model loss accuracy in 500 epochs

Chapter 7
Discussion

7.1 Discussion

In this chapter, we discuss the details of the overall system details where we have gathered data, pre-processed, the neural network, and the mobile application developed. The automated timber recognition system which can identify digital images of wood. Usually, it is difficult to identify the timber species from a digital image of wood. But the most practical way of identifying a timber species is by using digital images rather than acquiring microscopic or macroscopic images. The features we mainly used are the texture and color to identify the images. In this research, we referred to several kinds of literature with texture recognition methods.

In chapter 3 we have discussed some past research methodologies in identifying texture. One such technique on the identification of the wood species is GLCM. GLCM extract features from the training data to identify the examples of wood. This method was not much efficient in our research because of the digital images we used. The features extracted from GLCM were then put into the k- Nearest Neighbor Algorithm to classify the species but we were only able to get 50% of accuracy for a dataset containing 17000 images. Timber species which have similar color and texture (get vary depending on the way of sawing) are not recognized to a satisfactory level with the GLCM technique.

As the second step, we used SIFT and SURF techniques. These edge detection techniques are developed to match key points between two images and then identify the similarities. SIFT and SURF both did not give accurate results even for similar types of timber by matching key points.

This led us to apply a deep learning technique which is popular and proven to be efficient in image classification at present. The convolutional neural network which is specially designed for image classification is used to identify timber species using the digital images of wood. We used a CNN developed with Keras using TensorFlow as backend. The CNN consist of 2 convolutional layers and one pool layer. The convolutional layers are in the size of 3 and 2 respectively. The pooling setting is max-pooling with 2x2 receptive fields. This setting of CNN was able to give 89% of accuracy. The validation accuracy we got was 80%. Even the loss function reduced after each epoch/ iteration denoting the model created by the training data to be a successful and efficient model without overfitting.

Chapter 8

Conclusion

8.1 Conclusion

Wood identification is an important process in the timber-related industries. They are many methods are being used by timber industries for wood identification purpose. This paper proposes an automatic visual recognition system for the identification of tropical timber species based on image processing and data mining techniques. The system was factually planned to be cost-effective and to replace wood inspector’s duty.

Based on the past research approaches, we can identify that has not been many implementations in automatic wood recognition, due to the following reasons: Difficulty in finding an extensive range of wood database, absence of availability of proven techniques for timber identification, current research uses expensive devices and availability of human inspectors especially in developing countries.

A clustering algorithm can be performed for the training samples within each species to cluster the different texture within the same species into multiple clusters. Observation of the wood samples shows that there are possibilities of having more than one single texture from the same species. This will ensure that the means and standard deviations calculated will be better to describe the cluster of similar texture.

After evaluating different approaches for timber classification, such as GLCM feature extraction along with artificial neural networks, Speeded-Up Robust Feature for the comparison of images, it was found that convolutional neural network approach has given a high performance and accuracy of 88% in classification. Therefore, wood recognition was mainly based on a learning model for classifying timber images as the main outcome of the proposed system. Finally, a mobile application was developed on timber recognition, based on convolutional neural network approach.

The main functionalities of the mobile application are to recognize images of different wood species and provide informative details on various timber types along with necessary sample images. So, the users can easily capture the image of the wood and recognize the type of uploaded images using the application. The system uses the learned model on identifying the input image and it will display the name of the wood species that has a high possibility to match with the features of the given image.

So, this application will be useful in various users of the timber industry, government agencies, and museums as well as in other educational and scientific fields.

8.2 Limitation

As most of the wood species have the same texture pattern, it was a challenge to differentiate the same types of wood types in an accurate manner. As an example, Burutha and Attoniya has same texture pattern and the same color variation across the timber. Therefore, the accuracy of classifying Burutha and Attoniya is less compared to other wood species. The effect of lightning, while capturing the image and cutting or blade marks on the timber along with the direction of the hewing may also affect the accuracy of the classification process. It is important to consider the effect of different forms of saws as the blade marks along with the texture pattern will affect the result in a negative manner. Other possible defects types of timber may also be a problematic situation for an accurate classification, as various timber defects such as wind cracks, ring galls (curved swelling of trees), water or chemical stain, knots in timber, coarse grain defect in timber and defect of the timber due to fungi will affect to clearly extract the features of the images.

8.3 Future Developments

In the future, we wish to further develop the wood recognition system to a mobile application. The integration of the application into an embedded platform provides mobility and compactness of the system. This is helpful for the wood recognition problem as many wood samples are available and collected in the timber industry. However, a large wood data set is comprised of all the species that are encountered is needed before a robust embedded wood recognition system can be achieved. More experiments are needed to be run on large wood datasets to find the best solutions for recognizing a larger number of species.

The proposed solution can be further extended to identify the woods based on macroscopic features. Such as color luster, anatomic features, vessel arrangement and grouping, ray size relative to vessel diameter, ray height presence or absence of storied structure and colors of vessels.

8.4 Summary

This chapter concludes the thesis by presenting proposed approaches for the automated timber recognition system to identify various types of wood species, which helps to classify unknown timber in different primary and secondary industries. The outcome of the application would be benefitted in different areas for the identification of wood images based on its texture patterns.

References

[1] M. Khalid, R. Yusof, and A. S. M. Khairuddin, “Tropical wood species recognition system based on multi-feature extractors and classifiers,” in *2011 2nd International Conference on Instrumentation Control and Automation*, 2011, pp. 6–11.

[2] C. K. Muthumala and H. S. Amarasekara, “Construction of a Dicotomous Key for Common Local and Imported Timber Species in Sri Lanka,” *Proc. Int. For. Environ. Symp.*, vol. 18, no. 0, 2013.

[3] R. M. Haralick, K. Shanmugam, and I. Dinstein, “Textural Features for Image Classification,” *IEEE Trans. Syst. Man Cybern.*, vol. SMC-3, no. 6, pp. 610–621, Nov. 1973.

[4] J. Y. Tou, Y. H. Tay, and P. Y. Lau, “Gabor Filters and Grey-level Co-occurrence Matrices in Texture Classification,” 2007.

[5] I. A. Majid, Z. A. Latif, and N. A. Adnan, “Tree species classification using worldview-3 data,” in *2016 7th IEEE Control and System Graduate Research Colloquium (ICSGRC)*, 2016, pp. 73–76.

[6] P. Barmpoutis, K. Dimitropoulos, I. Barboutis, N. Grammalidis, and P. Lefakis, “Wood species recognition through multidimensional texture analysis,” *Comput. Electron. Agric.*, vol. 144, pp. 241–248, Jan. 2018.

[7] B. R, N. B, and S. R, “Wood Species Recognition Using GLCM and Correlation,” in *2009 International Conference on Advances in Recent Technologies in Communication and Computing*, 2009, pp. 615–619.

[8] T. M. Khan, D. G. Bailey, M. A. U. Khan, and Y. Kong, “Efficient Hardware Implementation For Fingerprint Image Enhancement Using Anisotropic Gaussian Filter,” *IEEE Trans. Image Process.*, vol. 26, no. 5, pp. 2116–2126, May 2017.

[9] “Wood species recognition based on SIFT keypoint histogram - Semantic Scholar.” [Online]. Available: https://www.semanticscholar.org/paper/Wood-species-recognition-based-on-SIFT-keypoint-Hu-Li/0dfb6ace8c57bf0b7eab96d4b04404a36ba95929. [Accessed: 20-Feb-2019].

[10] M. M. Hittawe, S. M. Muddamsetty, D. Sidibé, and F. Mériaudeau, “Multiple features extraction for timber defects detection and classification using SVM,” in

2015 IEEE International Conference on Image Processing (ICIP), 2015, pp. 427–431.

[11] R. Yusof, N. R. Rosli, and M. Khalid, “Using Gabor Filters as Image Multiplier for Tropical Wood Species Recognition System,” in *2010 12th International Conference on Computer Modelling and Simulation*, 2010, pp. 289–294.

[12] J. Y. Tou, Y. H. Tay, and P. Y. Lau, “Rotational Invariant Wood Species Recognition through Wood Species Verification,” in *2009 First Asian Conference on Intelligent Information and Database Systems*, 2009, pp. 115–120.

[13] R. B. Ameur, L. Valet, and D. Coquin, “A fusion system for tree species recognition through leaves and barks,” in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016, pp. 1–8.

[14] D. Zhao, “Automated Recognition of Wood Damages Using Artificial Neural Network,” in *2009 International Conference on Measuring Technology and Mechatronics Automation*, 2009, vol. 3, pp. 195–197.

[15] M. Valdenegro-Toro, “Improving Sonar Image Patch Matching via Deep Learning,” *ArXiv170902150 Cs*, Sep. 2017.

[16] “Forest Species Recognition Using Deep Convolutional Neural Networks - IEEE Conference Publication.” [Online]. Available: https://ieeexplore.ieee.org/document/6976909. [Accessed: 20-Feb-2019].

[17] “(PDF) Deep Learning Applied to Identification of Commercial Timber Species from Peru,” *ResearchGate*. [Online]. Available: https://www.researchgate.net/publication/329189279_Deep_Learning_Applied_to_Identification_of_Commercial_Timber_Species_from_Peru. [Accessed: 19-Feb-2019].

[18] W. H. Rankothge, D. M. S. B. Dissanayake, U. V. K. T. Gunathilaka, S. A. C. M. Gunarathna, C. M. Mudalige, and R. P. Thilakumara, “Plant recognition system based on Neural Networks,” in *2013 International Conference on Advances in Technology and Engineering (ICATE)*, 2013, pp. 1–4.

[19] M. Nasirzadeh, A. A. Khazael, and M. b Khalid, “Woods Recognition System Based on Local Binary Pattern,” in *2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks*, 2010, pp. 308–313.

[20] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[21] A. Y. Ivanov, G. I. Borzunov, and K. Kogos, “Recognition and identification of the clothes in the photo or video using neural networks,” in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2018, pp. 1513–1516.

[22] Z. Zhang and N. Ye, “A Novel Nonlinear Dimensionality Reduction Method for Robust Wood Image Recognition,” in *2009 International Joint Conference on Bioinformatics, Systems Biology and Intelligent Computing*, 2009, pp. 533–536.

[23] Y. Xin and W. Xue, “Application of some valid methods in logs counting system based on digital image processing,” in *2010 IEEE 17Th International Conference on Industrial Engineering and Engineering Management*, 2010, pp. 400–403.

Appendix A: GLCM-ANN

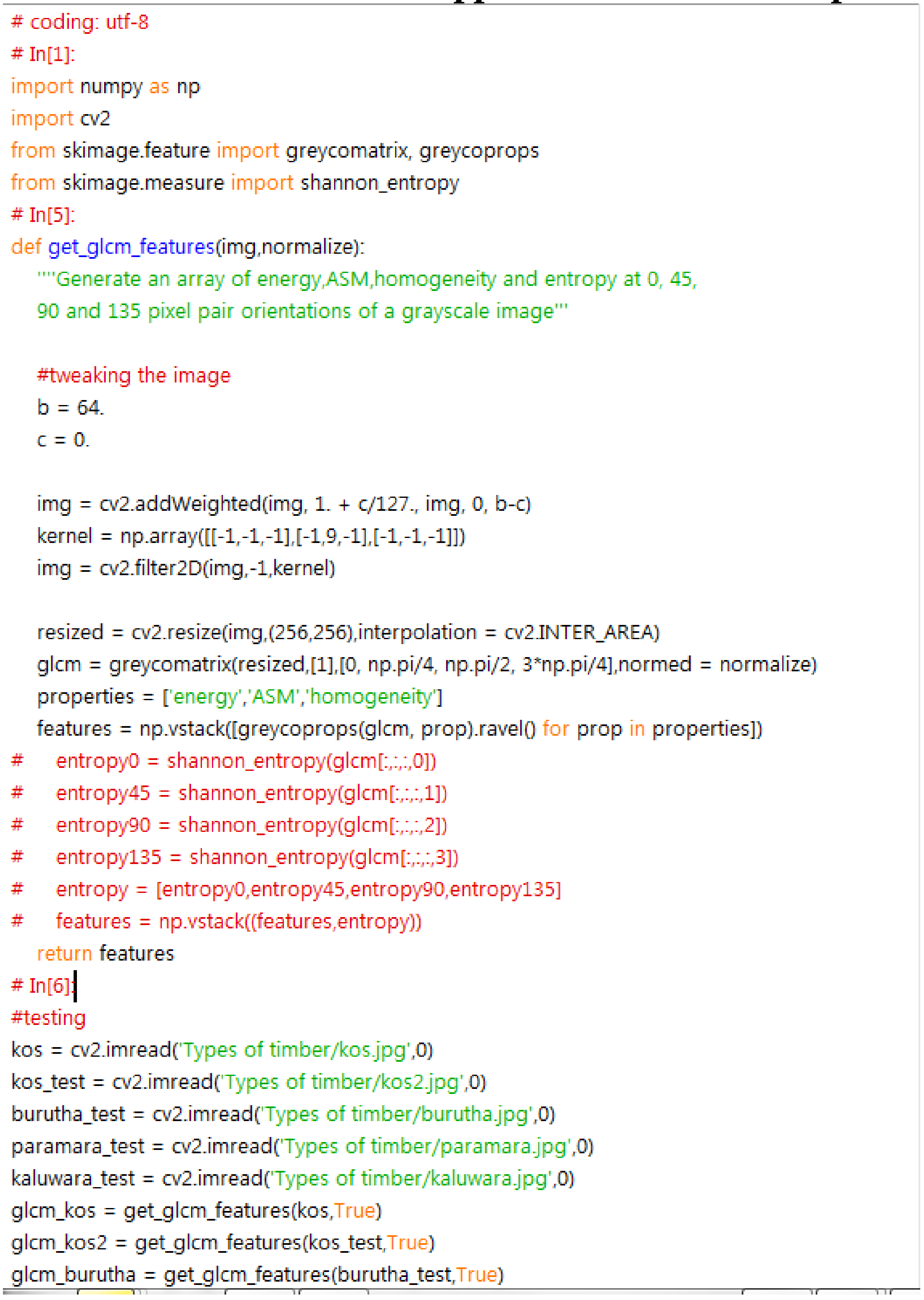
```
import cv2
import sys
import numpy as np
np.random.seed(1)
import tensorflow as tf
#from cv2 import *
from tensorflow import keras
from skimage import feature
from skimage.feature import greycomatrix, greycoprops
from skimage.measure import shannon_entropy
import glob

#Load images
burutha = glob.glob('Types of timber/to_ann/burutha/*.jpg')
burutha = [cv2.imread(img) for img in burutha]

gammalu = glob.glob('Types of timber/to_ann/gammalu/*.jpg')
gammalu = [cv2.imread(img) for img in gammalu]

kaluwara = glob.glob('Types of timber/to_ann/kaluwara/*.jpg')
kaluwara = [cv2.imread(img) for img in kaluwara]

kos = glob.glob('Types of timber/to_ann/kos/*.jpg')
kos = [cv2.imread(img) for img in kos]

mahogany = glob.glob('Types of timber/to_ann/mahogany/*.jpg')
mahogany = [cv2.imread(img) for img in mahogany]

paramara = glob.glob('Types of timber/to_ann/paramara/*.jpg')
paramara = [cv2.imread(img) for img in paramara]

pinus = glob.glob('Types of timber/to_ann/pinus/*.jpg')
pinus = [cv2.imread(img) for img in pinus]

teak = glob.glob('Types of timber/to_ann/teak/*.jpg')
teak = [cv2.imread(img) for img in teak]

#INCREASE TRAINING SET DATA

#check if the images have loaded
```

```
def get_glcm_features(*img_list,normalize = True):
    '''Generate an array of energy,ASM,homogeneity and entropy at 0, 45,
    90 and 135 pixel pair orientations of a list of grayscale images
    and hue of HSV images'''

    #tweaking the image
    b = 64.
    c = 0.

    features_ = np.array([])
    for wood in img_list:
        for img in wood:
            gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            hsv = cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
            gray = cv2.addWeighted(gray, 1. + c/127., gray, 0, b-c)
            kernel = np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])
            gray = cv2.filter2D(gray,-1,kernel)

            resized = cv2.resize(gray,(256,256),interpolation = cv2.INTER_AREA)
            glcm = greycomatrix(resized,[1],[0, np.pi/4, np.pi/2, 3*np.pi/4],normed = normalize)
            properties = ['energy','ASM','homogeneity']
            features = np.vstack([greycoprops(glcm, prop).ravel() for prop in properties])
            avg_hue = np.average(hsv[0])
            avg_hue= np.ones(4)*(avg_hue/255.0)
            features = np.vstack((features,avg_hue))
            if features_.size == 0:
                features_ = np.copy(features)
                features_ = features_[np.newaxis]
            else:
                features_ = np.vstack((features_,features[np.newaxis]))
    return features_
#THE AVERAGE HUE WAS ADDED AS THE LAST ROW OF THE features_ ARRAY.
#NOT SURE IF THIS IS CORRECT.
#Get glcm features
features = get_glcm_features(burutha, gammalu, kaluwara, kos, mahogany, paramara, pinus, teak)
print(features.shape)

#Labels: 0 - burutha, 1 - gammalu, ... , 7 - teak
labels = np.arange(8)
labels = np.repeat(labels, 8)
```

```
labels.shape

#Shuffle the data
order = np.argsort(np.random.random(labels.shape))
features = features[order]
labels = labels[order]

#Get 50 examples for training and 14 for testing
train_data = features[:80]
test_data = features[20:]
train_labels = labels[:80]
test_labels = labels[20:]

#Create the ANN model
model = keras.Sequential()
model.add(keras.layers.Flatten(input_shape = (4,4)))
model.add(keras.layers.Dense(10, activation = tf.nn.relu))
model.add(keras.layers.Dense(20, activation = tf.nn.relu))
model.add(keras.layers.Dense(8, activation = tf.nn.softmax))


#Compile the model
model.compile(optimizer = tf.train.AdamOptimizer(),
           loss = 'sparse_categorical_crossentropy',
            metrics = ['accuracy'])

model.summary()

trained = model.fit(train_data, train_labels, epochs = 500)

#- TRY TO BUILD THE MODEL WITHOUT THE "FLATTEN" LAYER
#- TRY A DIFFERENT NEURAL NETWORK LAYOUT

#Test model
test_loss, test_acc = model.evaluate(test_data, test_labels)

#Analysis
print('Test Accuracy: {:.2f}%'.format(test_acc*100))
```

```
#Analysis
print('Test Accuracy: {:.2f}%'.format(test_acc*100))

#Predictions
predictions = model.predict(test_data)

#prediction for the first image in the test set,
print(predictions[4])
print('Most probable label:', np.argmax(predictions[4]))
print('Actual label:',train_labels[4])
#The actual label and the most probable label, estimated by the ANN
#match.


# POINTERS FROM DHARSHANA AND BUDDHI:
# - USE RESNET (OR ALEXNET). GET THE FIRST 8 OR SO LAYERS AND INPUT THE ENTIRE RGB OR HSV IMAGES AND SEE WHAT HAPPENS.
```

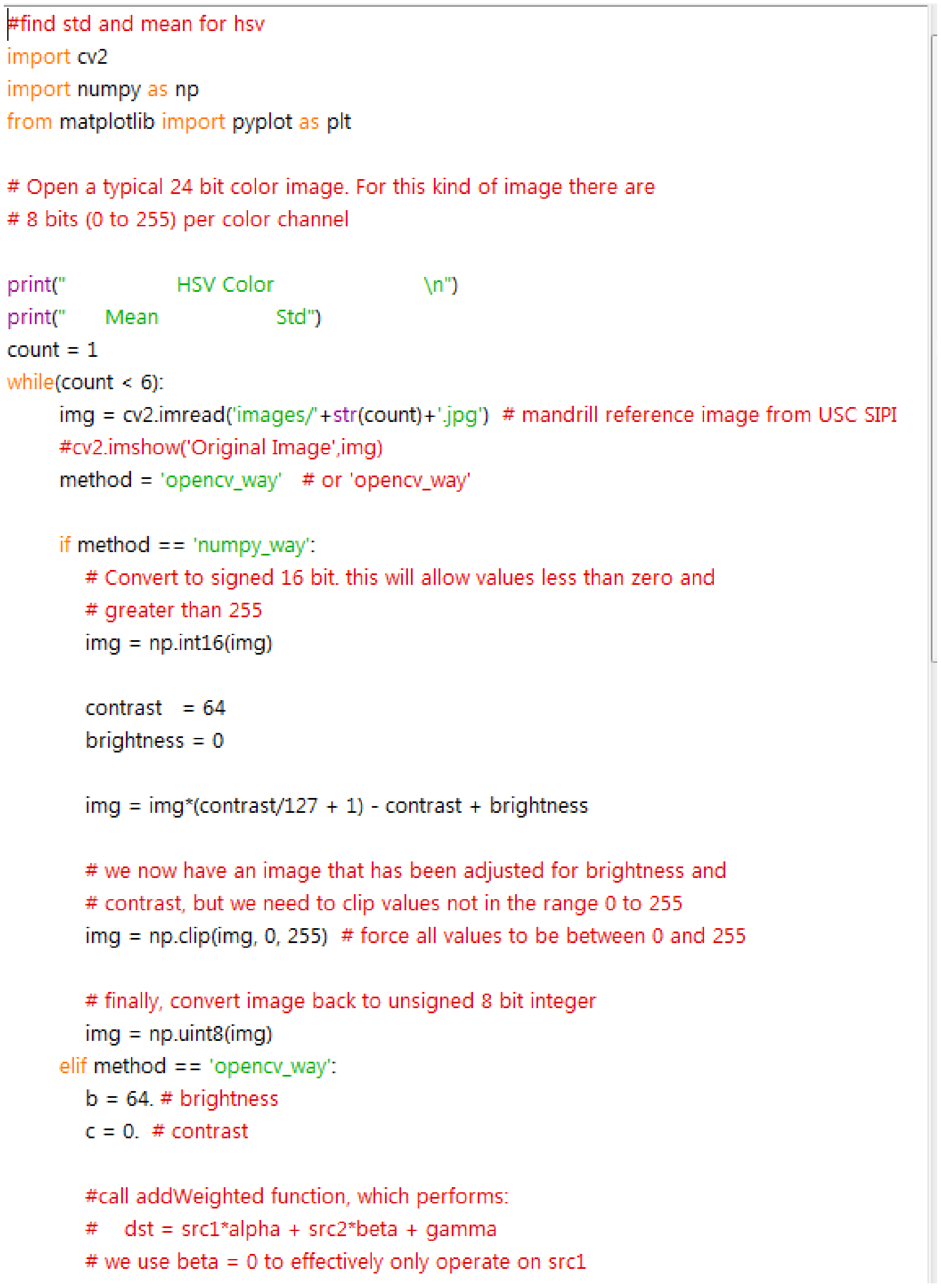
Appendix B: GLCM – Opencv



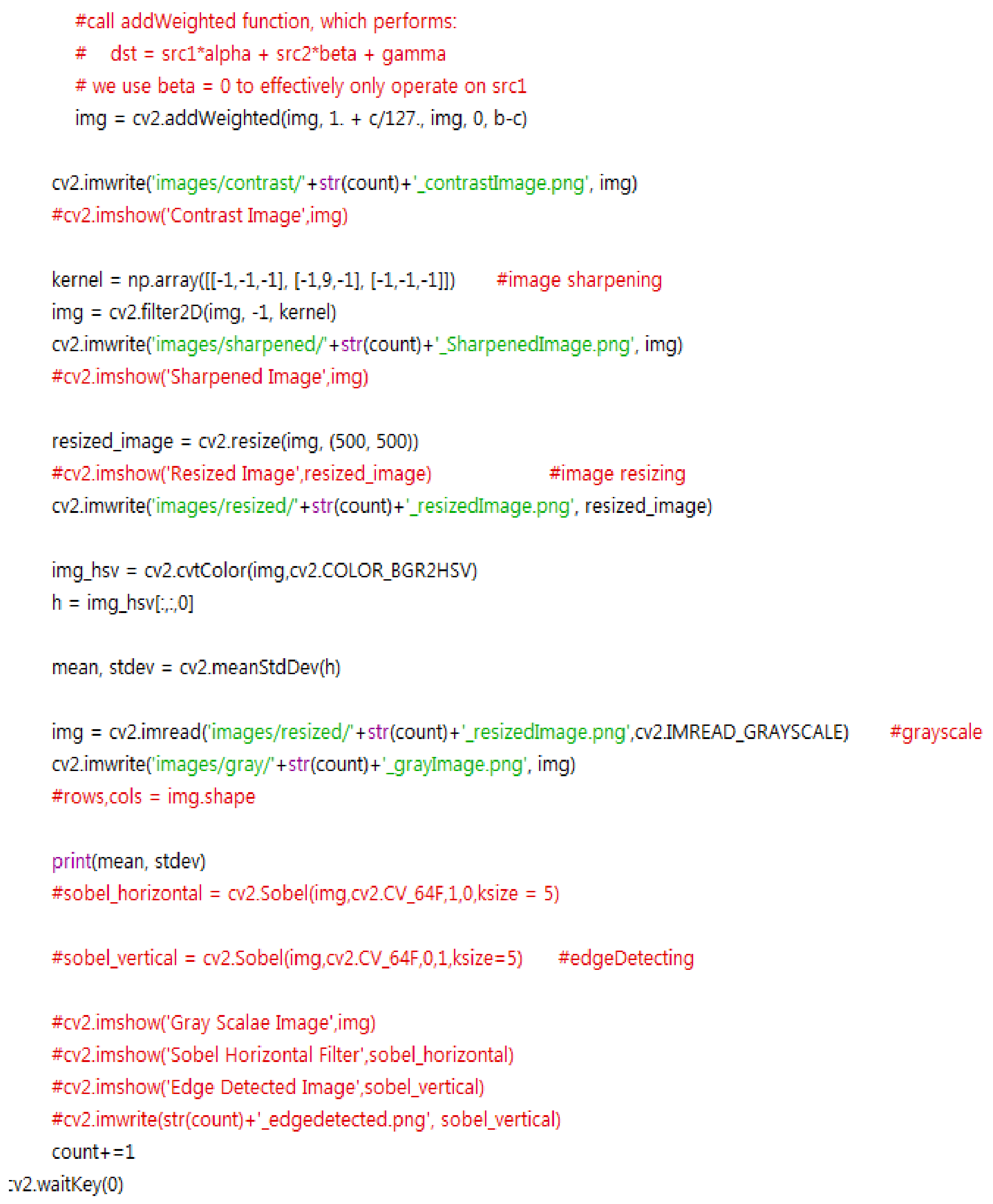
```python
# coding: utf-8
# In[1]:
import numpy as np
import cv2
from skimage.feature import greycomatrix, greycoprops
from skimage.measure import shannon_entropy
# In[5]:
def get_glcm_features(img,normalize):
    '''Generate an array of energy,ASM,homogeneity and entropy at 0, 45,
    90 and 135 pixel pair orientations of a grayscale image'''

    #tweaking the image
    b = 64.
    c = 0.

    img = cv2.addWeighted(img, 1. + c/127., img, 0, b-c)
    kernel = np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])
    img = cv2.filter2D(img,-1,kernel)

    resized = cv2.resize(img,(256,256),interpolation = cv2.INTER_AREA)
    glcm = greycomatrix(resized,[1],[0, np.pi/4, np.pi/2, 3*np.pi/4],normed = normalize)
    properties = ['energy','ASM','homogeneity']
    features = np.vstack([greycoprops(glcm, prop).ravel() for prop in properties])
#    entropy0 = shannon_entropy(glcm[:,:,:,0])
#    entropy45 = shannon_entropy(glcm[:,:,:,1])
#    entropy90 = shannon_entropy(glcm[:,:,:,2])
#    entropy135 = shannon_entropy(glcm[:,:,:,3])
#    entropy = [entropy0,entropy45,entropy90,entropy135]
#    features = np.vstack((features,entropy))
    return features
# In[6]:
#testing
kos = cv2.imread('Types of timber/kos.jpg',0)
kos_test = cv2.imread('Types of timber/kos2.jpg',0)
burutha_test = cv2.imread('Types of timber/burutha.jpg',0)
paramara_test = cv2.imread('Types of timber/paramara.jpg',0)
kaluwara_test = cv2.imread('Types of timber/kaluwara.jpg',0)
glcm_kos = get_glcm_features(kos,True)
glcm_kos2 = get_glcm_features(kos_test,True)
glcm_burutha = get_glcm_features(burutha_test,True)
```

```
kos = cv2.imread('Types of timber/kos.jpg',0)
kos_test = cv2.imread('Types of timber/kos2.jpg',0)
burutha_test = cv2.imread('Types of timber/burutha.jpg',0)
paramara_test = cv2.imread('Types of timber/paramara.jpg',0)
kaluwara_test = cv2.imread('Types of timber/kaluwara.jpg',0)
glcm_kos = get_glcm_features(kos,True)
glcm_kos2 = get_glcm_features(kos_test,True)
glcm_burutha = get_glcm_features(burutha_test,True)
glcm_paramara = get_glcm_features(paramara_test,True)
glcm_kaluwara = get_glcm_features(kaluwara_test,True)

corr1 = np.corrcoef([glcm_kos.ravel(),glcm_kos2.ravel()])
print('PCC for kos and kos2:',corr1[0,1])

corr2 = np.corrcoef([glcm_kos.ravel(),glcm_burutha.ravel()])
print('PCC for kos and burutha:',corr2[0,1])

corr3 = np.corrcoef([glcm_kos.ravel(),glcm_paramara.ravel()])
print('PCC for kos and paramara:',corr3[0,1])

corr4 = np.corrcoef([glcm_kos.ravel(),glcm_kos.ravel()])
print('PCC for kos with itself:',corr4[0,1])

corr5 = np.corrcoef([glcm_kaluwara.ravel(),glcm_kos.ravel()])
print('PCC for kos and kaluwara:',corr5[0,1])



# cv2.imshow('img',burutha_test)
# cv2.waitKey(0)
# cv2.destroyAllWindows()
```

APPENDIX C: HSV values



```
#find std and mean for hsv
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Open a typical 24 bit color image. For this kind of image there are
# 8 bits (0 to 255) per color channel

print("               HSV Color               \n")
print("      Mean                Std")
count = 1
while(count < 6):
        img = cv2.imread('images/'+str(count)+'.jpg')  # mandrill reference image from USC SIPI
        #cv2.imshow('Original Image',img)
        method = 'opencv_way'   # or 'opencv_way'

        if method == 'numpy_way':
            # Convert to signed 16 bit. this will allow values less than zero and
            # greater than 255
            img = np.int16(img)

            contrast   = 64
            brightness = 0

            img = img*(contrast/127 + 1) - contrast + brightness

            # we now have an image that has been adjusted for brightness and
            # contrast, but we need to clip values not in the range 0 to 255
            img = np.clip(img, 0, 255)  # force all values to be between 0 and 255

            # finally, convert image back to unsigned 8 bit integer
            img = np.uint8(img)
        elif method == 'opencv_way':
            b = 64. # brightness
            c = 0.  # contrast

            #call addWeighted function, which performs:
            #    dst = src1*alpha + src2*beta + gamma
            # we use beta = 0 to effectively only operate on src1
```



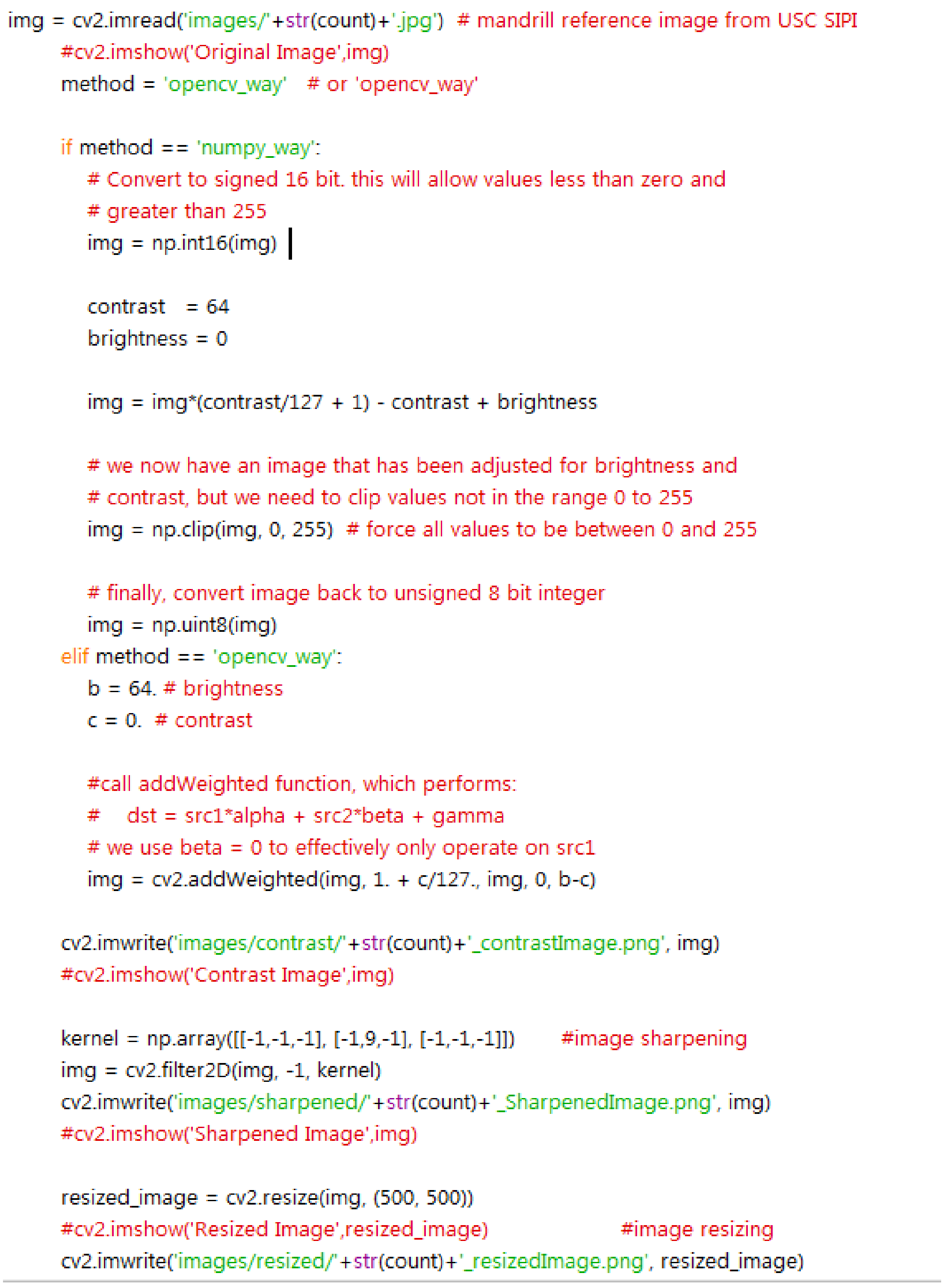
```
      #call addWeighted function, which performs:
      #    dst = src1*alpha + src2*beta + gamma
      # we use beta = 0 to effectively only operate on src1
      img = cv2.addWeighted(img, 1. + c/127., img, 0, b-c)

    cv2.imwrite('images/contrast/'+str(count)+'_contrastImage.png', img)
    #cv2.imshow('Contrast Image',img)

    kernel = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])      #image sharpening
    img = cv2.filter2D(img, -1, kernel)
    cv2.imwrite('images/sharpened/'+str(count)+'_SharpenedImage.png', img)
    #cv2.imshow('Sharpened Image',img)

    resized_image = cv2.resize(img, (500, 500))
    #cv2.imshow('Resized Image',resized_image)                #image resizing
    cv2.imwrite('images/resized/'+str(count)+'_resizedImage.png', resized_image)

    img_hsv = cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
    h = img_hsv[:,:,0]

    mean, stdev = cv2.meanStdDev(h)

    img = cv2.imread('images/resized/'+str(count)+'_resizedImage.png',cv2.IMREAD_GRAYSCALE)      #grayscale
    cv2.imwrite('images/gray/'+str(count)+'_grayImage.png', img)
    #rows,cols = img.shape

    print(mean, stdev)
    #sobel_horizontal = cv2.Sobel(img,cv2.CV_64F,1,0,ksize = 5)

    #sobel_vertical = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=5)     #edgeDetecting

    #cv2.imshow('Gray Scalae Image',img)
    #cv2.imshow('Sobel Horizontal Filter',sobel_horizontal)
    #cv2.imshow('Edge Detected Image',sobel_vertical)
    #cv2.imwrite(str(count)+'_edgedetected.png', sobel_vertical)
    count+=1
:v2.waitKey(0)
```

Appendix D: Standard deviation

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Open a typical 24 bit color image. For this kind of image there are
# 8 bits (0 to 255) per color channel
count = 1
while(count < 6):
        img = cv2.imread('images/'+str(count)+'.jpg')  # mandrill reference image from USC SIPI
        #cv2.imshow('Original Image',img)
        method = 'opencv_way'   # or 'opencv_way'

        if method == 'numpy_way':
            # Convert to signed 16 bit. this will allow values less than zero and
            # greater than 255
            img = np.int16(img)

            contrast   = 64
            brightness = 0

            img = img*(contrast/127 + 1) - contrast + brightness

            # we now have an image that has been adjusted for brightness and
            # contrast, but we need to clip values not in the range 0 to 255
            img = np.clip(img, 0, 255)  # force all values to be between 0 and 255

            # finally, convert image back to unsigned 8 bit integer
            img = np.uint8(img)
        elif method == 'opencv_way':
            b = 64. # brightness
            c = 0.  # contrast

            #call addWeighted function, which performs:
            #    dst = src1*alpha + src2*beta + gamma
            # we use beta = 0 to effectively only operate on src1
            img = cv2.addWeighted(img, 1. + c/127., img, 0, b-c)

        cv2.imwrite('images/contrast/'+str(count)+'_contrastImage.png', img)
        #cv2.imshow('Contrast Image',img)
```

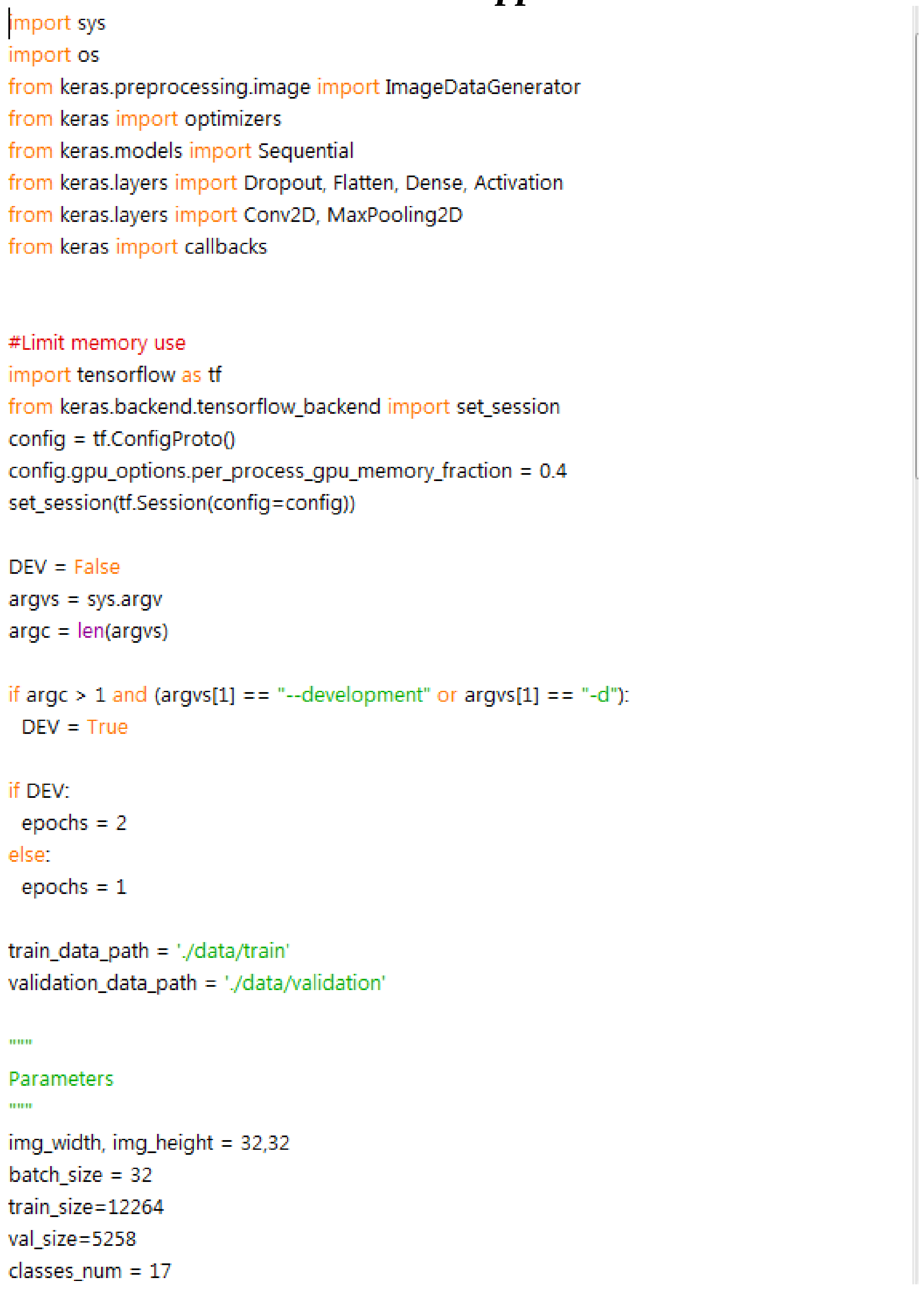
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Open a typical 24 bit color image. For this kind of image there are
# 8 bits (0 to 255) per color channel
count = 1
while(count < 6):
        img = cv2.imread('images/'+str(count)+'.jpg')  # mandrill reference image from USC SIPI
        #cv2.imshow('Original Image',img)
        method = 'opencv_way'   # or 'opencv_way'

        if method == 'numpy_way':
            # Convert to signed 16 bit. this will allow values less than zero and
            # greater than 255
            img = np.int16(img)

            contrast   = 64
            brightness = 0

            img = img*(contrast/127 + 1) - contrast + brightness

            # we now have an image that has been adjusted for brightness and
            # contrast, but we need to clip values not in the range 0 to 255
            img = np.clip(img, 0, 255)  # force all values to be between 0 and 255

            # finally, convert image back to unsigned 8 bit integer
            img = np.uint8(img)
        elif method == 'opencv_way':
            b = 64. # brightness
            c = 0.  # contrast

            #call addWeighted function, which performs:
            #    dst = src1*alpha + src2*beta + gamma
            # we use beta = 0 to effectively only operate on src1
            img = cv2.addWeighted(img, 1. + c/127., img, 0, b-c)

        cv2.imwrite('images/contrast/'+str(count)+'_contrastImage.png', img)
        #cv2.imshow('Contrast Image',img)
```

Appendix E: image preprocess



```
img = cv2.imread('images/'+str(count)+'.jpg')  # mandrill reference image from USC SIPI
    #cv2.imshow('Original Image',img)
    method = 'opencv_way'   # or 'opencv_way'

    if method == 'numpy_way':
        # Convert to signed 16 bit. this will allow values less than zero and
        # greater than 255
        img = np.int16(img)

        contrast   = 64
        brightness = 0

        img = img*(contrast/127 + 1) - contrast + brightness

        # we now have an image that has been adjusted for brightness and
        # contrast, but we need to clip values not in the range 0 to 255
        img = np.clip(img, 0, 255)  # force all values to be between 0 and 255

        # finally, convert image back to unsigned 8 bit integer
        img = np.uint8(img)
    elif method == 'opencv_way':
        b = 64. # brightness
        c = 0.  # contrast

        #call addWeighted function, which performs:
        #    dst = src1*alpha + src2*beta + gamma
        # we use beta = 0 to effectively only operate on src1
        img = cv2.addWeighted(img, 1. + c/127., img, 0, b-c)

    cv2.imwrite('images/contrast/'+str(count)+'_contrastImage.png', img)
    #cv2.imshow('Contrast Image',img)

    kernel = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])       #image sharpening
    img = cv2.filter2D(img, -1, kernel)
    cv2.imwrite('images/sharpened/'+str(count)+'_SharpenedImage.png', img)
    #cv2.imshow('Sharpened Image',img)

    resized_image = cv2.resize(img, (500, 500))
    #cv2.imshow('Resized Image',resized_image)                 #image resizing
    cv2.imwrite('images/resized/'+str(count)+'_resizedImage.png', resized_image)
```

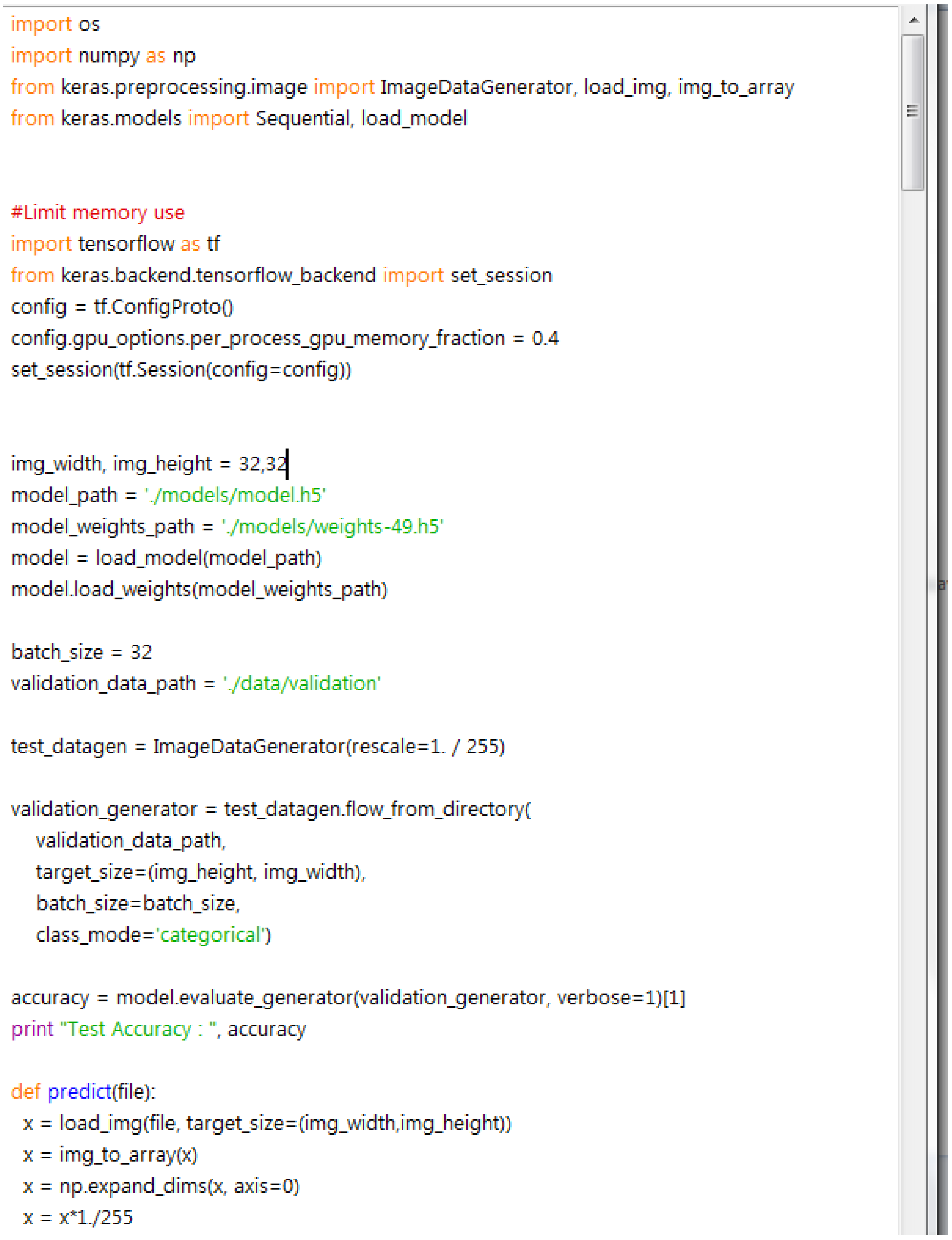
Appendix F: CNN-Trainiee



```python
import sys
import os
from keras.preprocessing.image import ImageDataGenerator
from keras import optimizers
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense, Activation
from keras.layers import Conv2D, MaxPooling2D
from keras import callbacks


#Limit memory use
import tensorflow as tf
from keras.backend.tensorflow_backend import set_session
config = tf.ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.4
set_session(tf.Session(config=config))

DEV = False
argvs = sys.argv
argc = len(argvs)

if argc > 1 and (argvs[1] == "--development" or argvs[1] == "-d"):
  DEV = True

if DEV:
  epochs = 2
else:
  epochs = 1

train_data_path = './data/train'
validation_data_path = './data/validation'

"""
Parameters
"""
img_width, img_height = 32,32
batch_size = 32
train_size=12264
val_size=5258
classes_num = 17
```

```
"""
img_width, img_height = 32,32
batch_size = 32
train_size=12264
val_size=5258
classes_num = 17
lr = 0.001
decay = 0.95
steps_per_epoch = int(train_size/batch_size)
validation_steps = int(val_size/batch_size)


model = Sequential()
model.add(Conv2D(64, (3,3), activation='relu', input_shape=(img_width, img_height, 3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(128, (3,3), activation='relu'))
model.add(Conv2D(256, (3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), dim_ordering='th'))

model.add(Flatten())
model.add(Dense(256))
model.add(Activation("relu"))
model.add(Dropout(0.5))
model.add(Dense(classes_num, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.Adam(lr=lr),
              metrics=['accuracy'])

train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    vertical_flip=True)

test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
train_generator = train_datagen.flow_from_directory(
    train_data_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    validation_data_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

"""
Tensorboard log
"""
log_dir = './tf-log/'
target_dir = './models/'
if not os.path.exists(target_dir):
  os.mkdir(target_dir)

#tb_cb = callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
#cbks = [tb_cb]
checkpoint = callbacks.ModelCheckpoint(target_dir + 'weights-{epoch:02d}.h5', monitor='val_acc
                                       save_best_only=True, save_weights_only=True, verbose=1)
lr_decay = callbacks.LearningRateScheduler(schedule=lambda epoch: lr * (decay ** epoch))

model.fit_generator(
    train_generator,
    steps_per_epoch=steps_per_epoch,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=validation_steps,
    callbacks=[checkpoint, lr_decay])

model.save('./models/model.h5')
model.save_weights('./models/weights.h5')
```

Appendix G: CNN-Training



```
import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from keras.models import Sequential, load_model


#Limit memory use
import tensorflow as tf
from keras.backend.tensorflow_backend import set_session
config = tf.ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.4
set_session(tf.Session(config=config))


img_width, img_height = 32,32
model_path = './models/model.h5'
model_weights_path = './models/weights-49.h5'
model = load_model(model_path)
model.load_weights(model_weights_path)

batch_size = 32
validation_data_path = './data/validation'

test_datagen = ImageDataGenerator(rescale=1. / 255)

validation_generator = test_datagen.flow_from_directory(
    validation_data_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

accuracy = model.evaluate_generator(validation_generator, verbose=1)[1]
print "Test Accuracy : ", accuracy

def predict(file):
  x = load_img(file, target_size=(img_width,img_height))
  x = img_to_array(x)
  x = np.expand_dims(x, axis=0)
  x = x*1./255
```

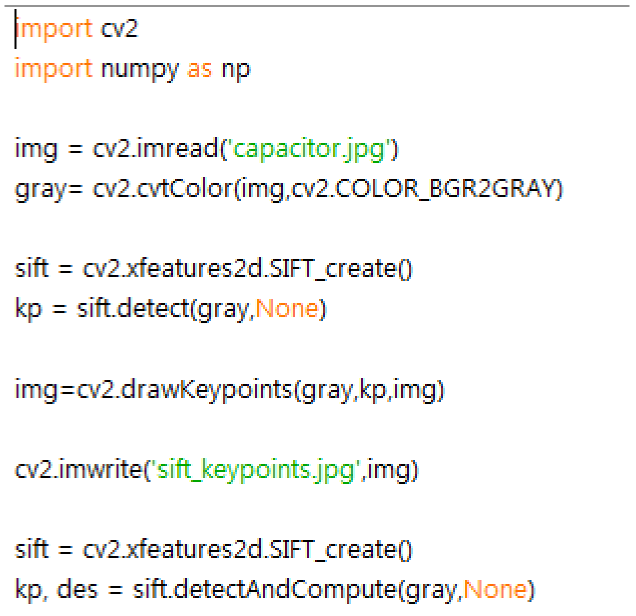
```
validation_generator = test_datagen.flow_from_directory(
    validation_data_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

accuracy = model.evaluate_generator(validation_generator, verbose=1)[1]
print "Test Accuracy : ", accuracy

def predict(file):
  x = load_img(file, target_size=(img_width,img_height))
  x = img_to_array(x)
  x = np.expand_dims(x, axis=0)
  x = x*1./255
  array = model.predict(x)
  result = array[0]
  answer = np.argmax(result)
  return answer

Albisiya_t = 0
Albisiya_f = 0
Attoniya_t = 0
Attoniya_f = 0
Burutha_t = 0
Burutha_f = 0
Gammalu_t = 0
Gammalu_f = 0
Garnish_t = 0
Garnish_f = 0
GiniSapu_t = 0
GiniSapu_f = 0
Hora_t = 0
Hora_f = 0
Kaluwara_t = 0
Kaluwara_f = 0
Kos_t = 0
Kos_f = 0
Kumbuk_t = 0
```

```
Paramara_t = 0
Paramara_f = 0
RathuKumbuk_t = 0
RathuKumbuk_f = 0
Sapu_t = 0
Sapu_f = 0
SuriyaMara_t = 0
SuriyaMara_f = 0
Tekka_t = 0
Tekka_f = 0
Walkohomba_t = 0
Walkohomba_f = 0


for i, ret in enumerate(os.walk('data/validation/Albisiya')):
  for i, filename in enumerate(ret[2]):
    if filename.startswith("."):
      continue
    #print("Label: Albisiya")
    result = predict(ret[0] + '/' + filename)
    if result == 0:
      Albisiya_t += 1
    else:
      Albisiya_f += 1

for i, ret in enumerate(os.walk('data/validation/Attoniya')):
  for i, filename in enumerate(ret[2]):
    if filename.startswith("."):
      continue
    #print("Label: Attoniya")
    result = predict(ret[0] + '/' + filename)
    if result == 1:
      Attoniya_t += 1
    else:
      Attoniya_f += 1
```

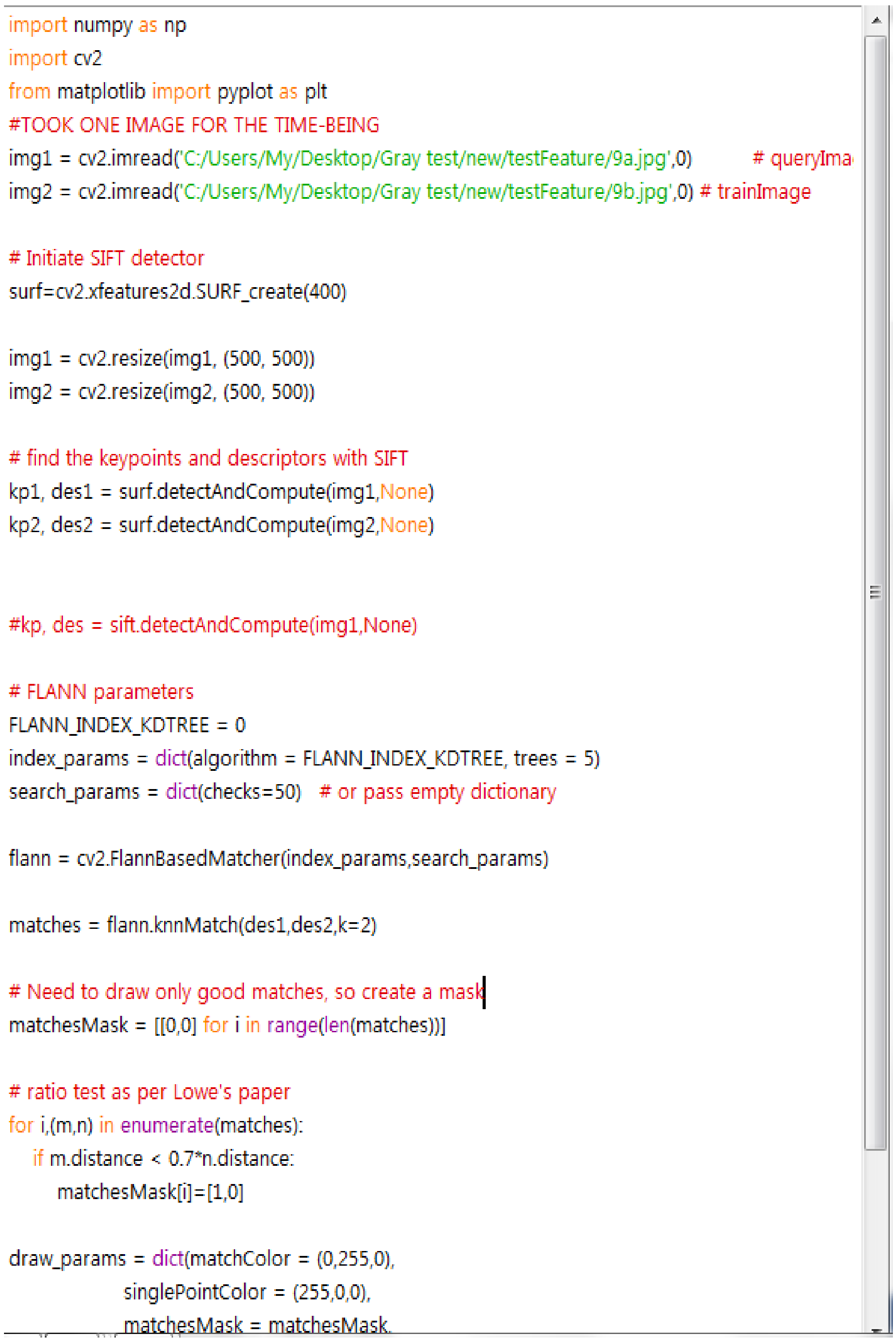
```
for i, ret in enumerate(os.walk('data/validation/Burutha')):
  for i, filename in enumerate(ret[2]):
    if filename.startswith("."):
      continue
    #print("Label: Burutha")
    result = predict(ret[0] + '/' + filename)
    if result == 2:
      Burutha_t += 1
    else:
      Burutha_f += 1

for i, ret in enumerate(os.walk('data/validation/Gammalu')):
  for i, filename in enumerate(ret[2]):
    if filename.startswith("."):
      continue
    #print("Label: Gammalu")
    result = predict(ret[0] + '/' + filename)
    if result == 3:
      Gammalu_t += 1
    else:
      Gammalu_f += 1
```

```
"""
Check metrics
"""
print("True Albisiya: ", Albisiya_t)
print("False Albisiya: ", Albisiya_f)
print("True Attoniya: ", Attoniya_t)
print("False Attoniya: ", Attoniya_f)
print("True Burutha: ", Burutha_t)
print("False Burutha: ", Burutha_f)
print("True Gammalu: ", Gammalu_t)
print("False Gammalu: ", Gammalu_f)
print("True Garnish: ", Garnish_t)
print("False Garnish: ", Garnish_f)
print("True GiniSapu: ", GiniSapu_t)
print("False GiniSapu: ", GiniSapu_f)
print("True Hora: ", Hora_t)
print("False Hora: ", Hora_f)
print("True Kaluwara: ", Kaluwara_t)
print("False Kaluwara: ", Kaluwara_f)
print("True Kos: ", Kos_t)
print("False Kos: ", Kos_f)
print("True Kumbuk: ", Kumbuk_t)
print("False Kumbuk: ", Kumbuk_f)
print("True Mahogani: ", Mahogani_t)
print("False Mahogani: ", Mahogani_f)
print("True Paramara: ", Paramara_t)
print("False Paramara: ", Paramara_f)
```

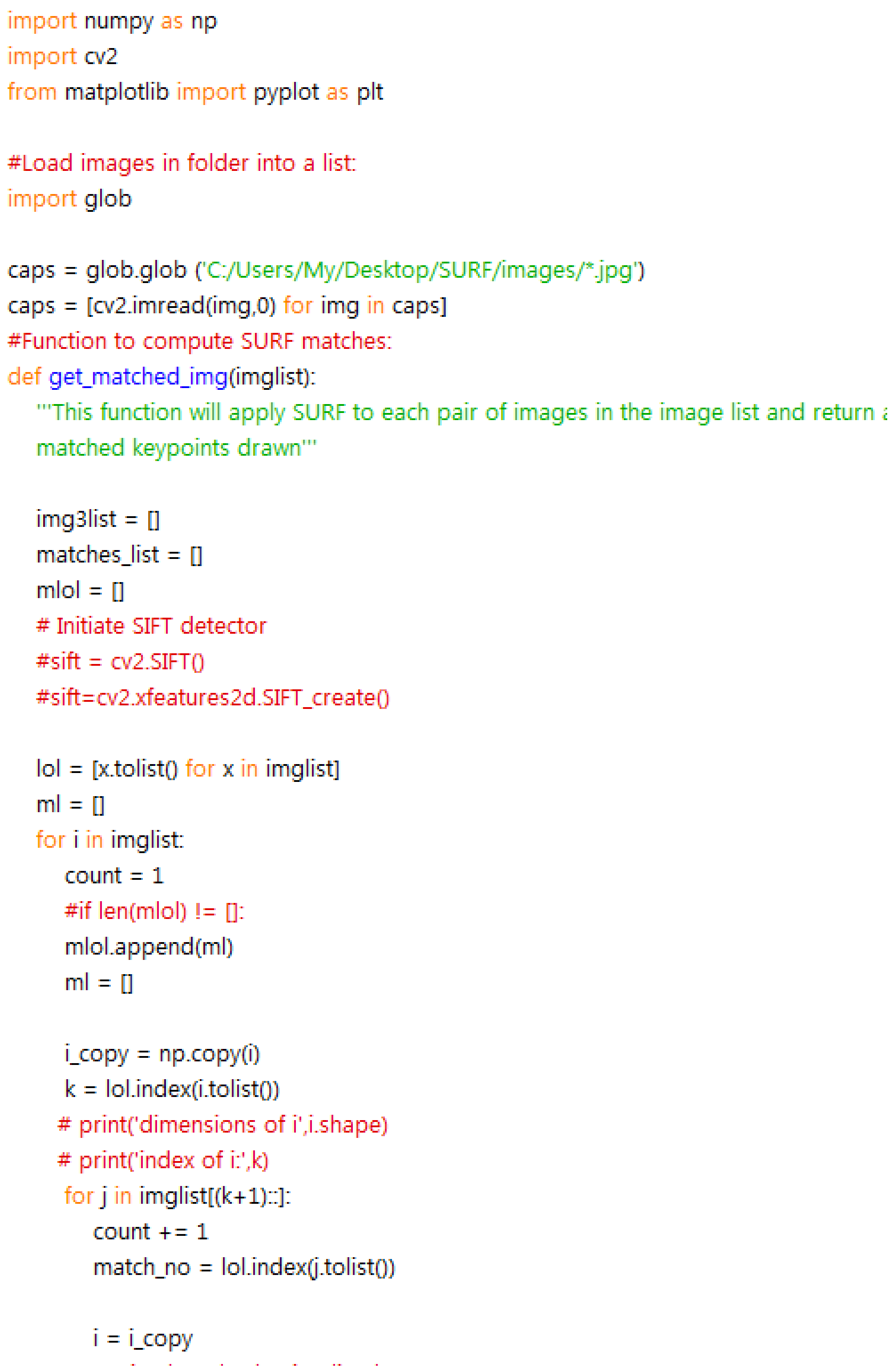
Appendix H: sift or surf



```
import cv2
import numpy as np

img = cv2.imread('capacitor.jpg')
gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

sift = cv2.xfeatures2d.SIFT_create()
kp = sift.detect(gray,None)

img=cv2.drawKeypoints(gray,kp,img)

cv2.imwrite('sift_keypoints.jpg',img)

sift = cv2.xfeatures2d.SIFT_create()
kp, des = sift.detectAndCompute(gray,None)
```

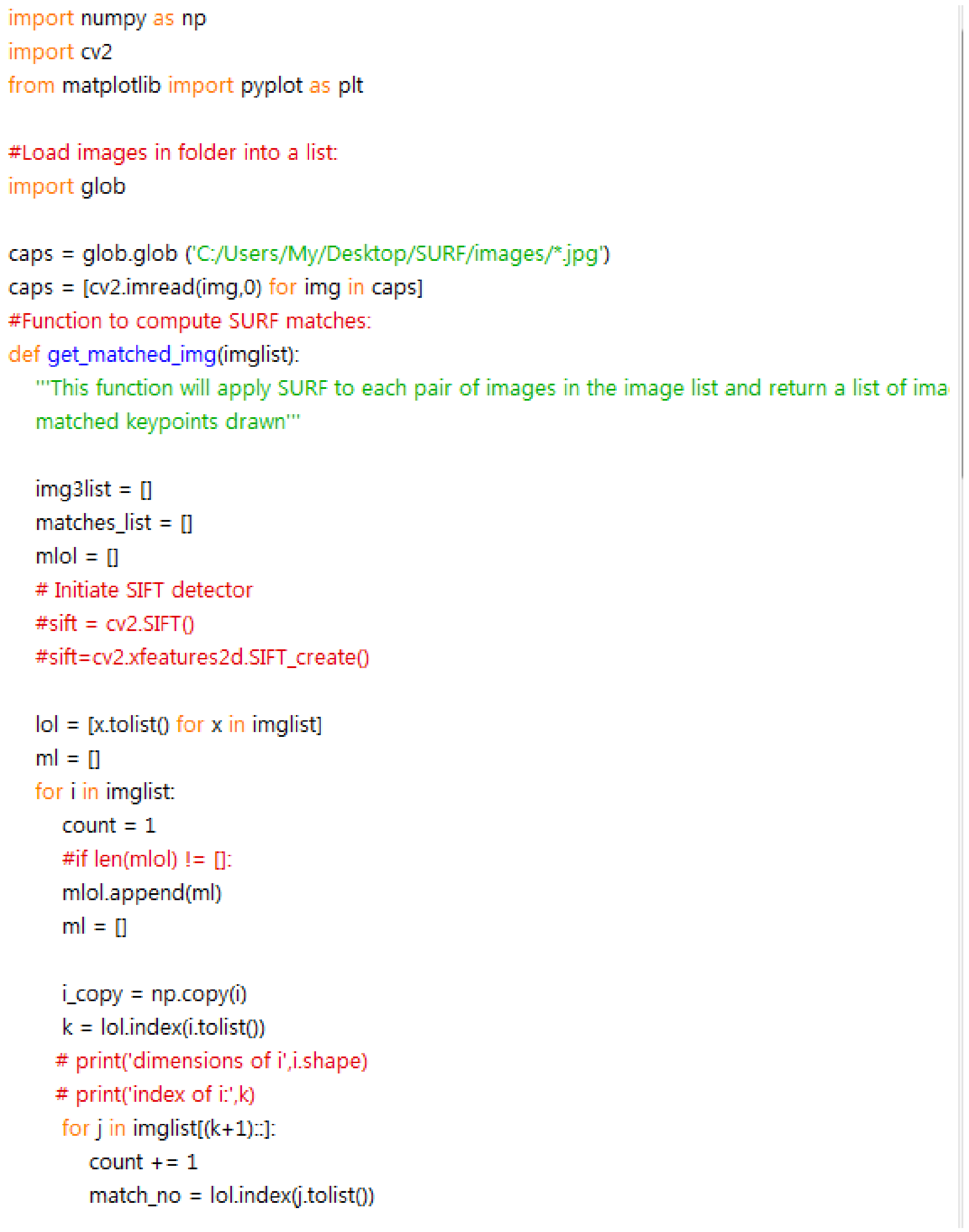


```
import numpy as np
import cv2
from matplotlib import pyplot as plt
#TOOK ONE IMAGE FOR THE TIME-BEING
img1 = cv2.imread('C:/Users/My/Desktop/Gray test/new/testFeature/9a.jpg',0)          # queryIma
img2 = cv2.imread('C:/Users/My/Desktop/Gray test/new/testFeature/9b.jpg',0) # trainImage

# Initiate SIFT detector
surf=cv2.xfeatures2d.SURF_create(400)

img1 = cv2.resize(img1, (500, 500))
img2 = cv2.resize(img2, (500, 500))

# find the keypoints and descriptors with SIFT
kp1, des1 = surf.detectAndCompute(img1,None)
kp2, des2 = surf.detectAndCompute(img2,None)


#kp, des = sift.detectAndCompute(img1,None)

# FLANN parameters
FLANN_INDEX_KDTREE = 0
index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
search_params = dict(checks=50)   # or pass empty dictionary

flann = cv2.FlannBasedMatcher(index_params,search_params)

matches = flann.knnMatch(des1,des2,k=2)

# Need to draw only good matches, so create a mask
matchesMask = [[0,0] for i in range(len(matches))]

# ratio test as per Lowe's paper
for i,(m,n) in enumerate(matches):
    if m.distance < 0.7*n.distance:
        matchesMask[i]=[1,0]

draw_params = dict(matchColor = (0,255,0),
                   singlePointColor = (255,0,0),
                   matchesMask = matchesMask,
#sift = cv2.SIFT()
#sift=cv2.xfeatures2d.SIFT_create()
surf=cv2.xfeatures2d.SURF_create(400)
#img1 = cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)
img1 = cv2.resize(img1, (500, 500))
img2 = cv2.resize(img2, (500, 500))

# find the keypoints and descriptors with SIFT
kp1, des1 = surf.detectAndCompute(img1,None)
kp2, des2 = surf.detectAndCompute(img2,None)
#print(des1)

#kp, des = sift.detectAndCompute(img1,None)

# FLANN parameters
FLANN_INDEX_KDTREE = 0
index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
search_params = dict(checks=50)   # or pass empty dictionary

flann = cv2.FlannBasedMatcher(index_params,search_params)

matches = flann.knnMatch(des1,des2,k=2)

# Need to draw only good matches, so create a mask
matchesMask = [[0,0] for i in range(len(matches))]

# ratio test as per Lowe's paper
for i,(m,n) in enumerate(matches):
    if m.distance < 0.7*n.distance:
        matchesMask[i]=[1,0]

draw_params = dict(matchColor = (0,255,0),
                   singlePointColor = (255,0,0),
                   matchesMask = matchesMask,
                   flags = 0)

img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,matches,None,**draw_params)
```



```
import numpy as np
import cv2
from matplotlib import pyplot as plt

#Load images in folder into a list:
import glob

caps = glob.glob ('C:/Users/My/Desktop/SURF/images/*.jpg')
caps = [cv2.imread(img,0) for img in caps]
#Function to compute SURF matches:
def get_matched_img(imglist):
    '''This function will apply SURF to each pair of images in the image list and return a
    matched keypoints drawn'''

    img3list = []
    matches_list = []
    mlol = []
    # Initiate SIFT detector
    #sift = cv2.SIFT()
    #sift=cv2.xfeatures2d.SIFT_create()

    lol = [x.tolist() for x in imglist]
    ml = []
    for i in imglist:
        count = 1
        #if len(mlol) != []:
        mlol.append(ml)
        ml = []

        i_copy = np.copy(i)
        k = lol.index(i.tolist())
       # print('dimensions of i',i.shape)
       # print('index of i:',k)
        for j in imglist[(k+1)::]:
            count += 1
            match_no = lol.index(j.tolist())

            i = i_copy
```

All the list of item checking the key points.



```python
import numpy as np
import cv2
from matplotlib import pyplot as plt

#Load images in folder into a list:
import glob

caps = glob.glob ('C:/Users/My/Desktop/SURF/images/*.jpg')
caps = [cv2.imread(img,0) for img in caps]
#Function to compute SURF matches:
def get_matched_img(imglist):
    '''This function will apply SURF to each pair of images in the image list and return a list of ima
    matched keypoints drawn'''

    img3list = []
    matches_list = []
    mlol = []
    # Initiate SIFT detector
    #sift = cv2.SIFT()
    #sift=cv2.xfeatures2d.SIFT_create()

    lol = [x.tolist() for x in imglist]
    ml = []
    for i in imglist:
        count = 1
        #if len(mlol) != []:
        mlol.append(ml)
        ml = []

        i_copy = np.copy(i)
        k = lol.index(i.tolist())
       # print('dimensions of i',i.shape)
       # print('index of i:',k)
        for j in imglist[(k+1)::]:
            count += 1
            match_no = lol.index(j.tolist())
```

```
 i_copy = np.copy(i)
 k = lol.index(i.tolist())
# print('dimensions of i',i.shape)
# print('index of i:',k)
 for j in imglist[(k+1)::]:
     count += 1
     match_no = lol.index(j.tolist())

     i = i_copy
  #    print('length: ',len(imglist[(k+1)::]))
    # print('dimensions of j',j.shape)
 img1, img2 = i,j

 surf=cv2.xfeatures2d.SURF_create(400)

 img1 = cv2.resize(img1, (500, 500))
 img2 = cv2.resize(img2, (500, 500))

 # find the keypoints and descriptors with SIFT
 kp1, des1 = surf.detectAndCompute(img1,None)
#  print('1')
 kp2, des2 = surf.detectAndCompute(img2,None)
# print('2')
 #print(des1)

 #kp, des = sift.detectAndCompute(img1,None)

 # FLANN parameters
 FLANN_INDEX_KDTREE = 0
 index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
 search_params = dict(checks=50)   # or pass empty dictionary

 flann = cv2.FlannBasedMatcher(index_params,search_params)

 matches = flann.knnMatch(des1,des2,k=2)
 matches_list.append(len(matches))
 matches_ = list(matches_list)

 thresh = 30
 if len(matches) >= thresh:
     ml.append(k + count)
     #print(ml)

 # Need to draw only good matches, so create a mask
 matchesMask = [[0,0] for i in range(len(matches))]

 # ratio test as per Lowe's paper
 for i,(m,n) in enumerate(matches):
     if m.distance < 0.7*n.distance:
        # ratio test as per Lowe's paper
        for i,(m,n) in enumerate(matches):
            if m.distance < 0.7*n.distance:
                matchesMask[i]=[1,0]
        print('Number of matches:',len(matches))

        draw_params = dict(matchColor = (0,255,0),
                           singlePointColor = (255,0,0),
                           matchesMask = matchesMask,
                           flags = 0)

        img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,matches,None,**draw_params)
        img3list.append(img3)

        print('***')

    return img3list, matches_list, mlol[1::]

#Apply function:
matched_list, matches, mlol = get_matched_img(caps)

print('Keypoints matched: \n',matches)
print(mlol)

#Show the images:
fig = plt.figure()

for i in range(len(matched_list)):
    r = int(len(matched_list)/3) + 1
    c = 3
    p = i + 1
    #print(r,c,p)
    fig.add_subplot(r,c,p)
    plt.imshow(matched_list[i]),plt.xticks([]),plt.yticks([]) #,plt.title(str(i + 1))

#One image per window

plt.show()
```