BOLL ROT DISEASE DETECTION IN COTTON CROP USING IMAGE BASED DEEP LEARNING

Anjum Ali
2013-ag-357

A thesis submitted in partial fulfillment of
the requirements for the degree of

MS (SOFTWARE ENGINEERING)

DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF SCIENCES
UNIVERSITY OF AGRICULTURE, FAISALABAD,
PAKISTAN

2022

DECLARATION

I, hereby declare that contents of the Thesis "**Boll Rot Disease Detection in Cotton Crop using Image based Deep Learning**" are product of my own research and no part has been copied from any published sources (except the references, some standard mathematical or genetic models/equations/protocols etc.).I further declare that this work has not been submitted for award of any other diploma/degree. This university may take action if the information provided is found inaccurate.

_____

Name: Anjum Ali

Reg. No: 2013-ag-357

The Controller of Examination,
University of Agriculture,
Faisalabad, Pakistan

We, the supervisory committee, certify that the contents and form of the thesis submitted by **Anjum Ali** Reg. No **2013-ag-357** have been found satisfactory and recommend that it may be processed for evaluation by the External Examiner(s) for the award of the degree.

SUPERVISORY COMMITTEE:

SUPERVISOR _____
Dr. M. Ahsan Latif

MEMBER _____
Mrs. Sidra Shahid

MEMBER _____
Dr. Yasir Jamil

DEDICATED

To

HOLY PROPHET MUHAMMAD (PBUH)

My Lovely Parents

And

Family Members

Who helped and encouraged me throughout my academic career. They have always been a great source of encouragement and inspiration for me. They are very nice and careful parent. They have great knowledge about life and always guide me in every part of my life.

ACKNOWLEDGEMENTS

First of all due praises to Almighty "**ALLAH**", the most merciful and compassionate, the most gracious and beneficent, whose bounteous blessings enable me to perceive and peruse higher ideals of life. All praises and respects are to His Holly Prophet "**MUHAMMAD**" (Peace Be Upon Him) who enables us to recognize our creator.

I extend the words of thanks to my Supervisor **Dr. M. Ahsan Latif**, Assistant Professor, Department of Computer Science, University of Agriculture, Faisalabad who provided me guidance and showed kindness in the form of keen interest during my course of study. He always guided me through difficult times I faced during completion of my Research Project. I am not wrong if I say that he remained source of inspiration for me during my study at University. I feel utmost pride in his knowledge the inspiring guidance and encouraging attitude during my whole educational carrier.

I am extremely obliged and grateful to **Mrs. Sidra Shahid,** Lecturer, Department of Computer Science, University of Agriculture, Faisalabad and **Dr. Yasir Jamil**, Chairman, Department of Physics, University of Agriculture, Faisalabad, the member of my supervisory committee for their valuable suggestions and encouragement during my studies.

I also acknowledge praise worthy contribution, moral support and best wishes of **Dr. Azeem Iqbal khan** Associate Professor, Department of Plant Breeding and Genetics, University of Agriculture, Faisalabad. I pray to almighty ALLAH to give them successes and love throughout their lives (Amen)!

Finally, I would like to extend heart full thanks to my family for their moral support and countless prayers throughout the study.

Anjum Ali

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

Plant diseases are one of the major factors affecting crop yield. Early identification of these diseases can improve productivity, save money and time for the farmer. Cotton also known as White Gold or The King of Fibers, is a valuable cash crop in several industrialized and developing countries. Cotton crops in Pakistan are plagued by a variety of diseases that damage the fruits, stem, and branches of the plants. Cotton boll rot disease caused by a variety of Fungi, Bacteria, and Viruses. The development of an automated method that can correctly identify and diagnose these diseases in their early stages is thus critical. The goal of this research is to combine deep learning and agriculture by developing a system for identifying cotton boll rot disease. We proposed a model that can identify boll rot attack in the cotton crops. We will determine out how much of our crops have been damaged. To train our models, an open dataset consisting of 487 images of cotton boll rot which were shot using DSLR camera and then used SSD with MobileNet-V2 and Faster RCNN with Inception -V2 architectures to train our models on the dataset. The implemented model achieved an accuracy of 89% and 67% respectively.

CHAPTER 1

INTRODUCTION

1.1 Overview

Pakistan is an Agricultural country, according to World Bank report of 2021 that approximately 62.82% of the population lives in rural areas, whose livelihoods are directly or indirectly associated with agriculture. Experts believe that Pakistan is an ideal place for all types of crops including cotton. Among the countries that produced cotton, Pakistan ranks 5th in cotton production after Brazil India, China, and the United States (FAO, 2021). In 2020 Cotton and cotton products account for approximately 22.69% of GDP. In the textile sector, cotton used for raw materials, so it is necessary to appropriately deal with the cotton crisis at the right time to save the textile industry. The problem related to cotton might be a direct and important cause of the crisis in the textile industry. Cotton is a profitable crop in developed and developing countries. It is predicted that 33.1 million hectares of cotton are cultivated overseas, producing 116.7 million bales, whereas 2.6 million hectares of cotton are cultivated in Pakistan, yielding 113.9 million bales (Kumar *et al*., 2021).

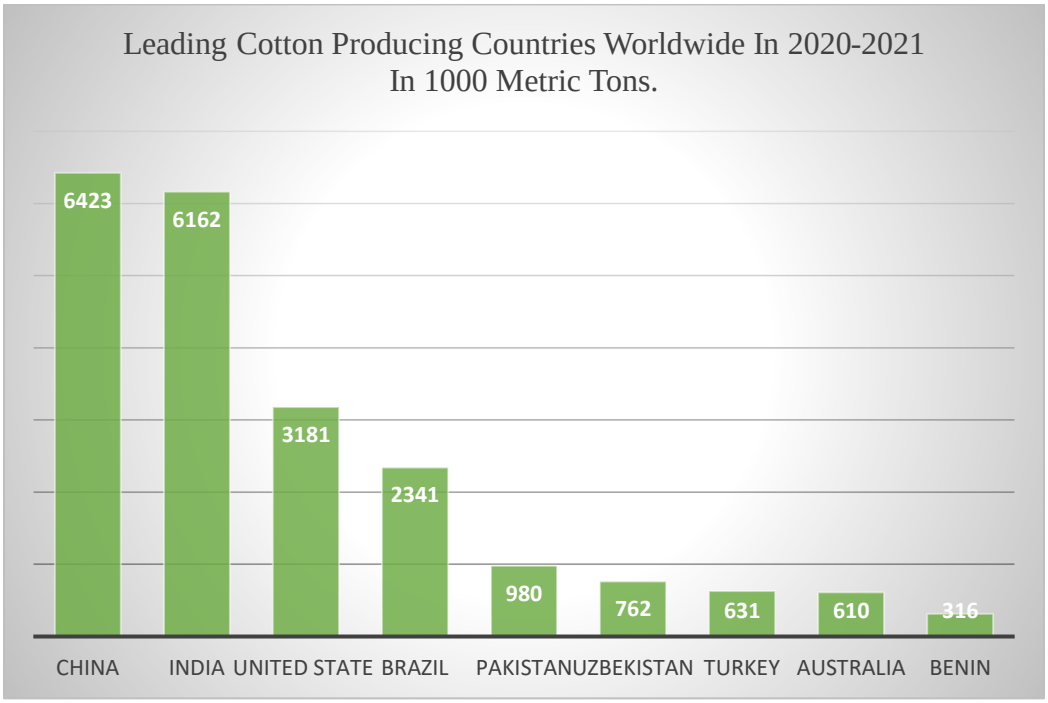


Figure 1.1: Leading Cotton Production Countries in the World

1.2 Importance of the Cotton

Pakistan is an ideal location for growing cotton and other crops such as rice, sugar cane and wheat. Pakistan is the world's 5th cotton producer and 6th cotton fiber exporter, following only the USA, China, Brazil, and India. Cotton is cash crop that produces fiber, edible oil, seeds, and livestock feed. It is grown in over 80 different nations. Pakistan Cotton exports are estimated 22.69 percent of the gross domestic product in 2020. (GDP). According to a World Bank research published in 2021, agricultural production dominates Pakistan's economy (Rafiq *et al*., 2021).

Cotton has been a prominent raw material crop in the textile industry for many years. As a result, it's vital to address the cotton problem effectively and at the right time if the textile industry is to survive. Cotton's problem has the potential to be a significant factor to the textile industry's current predicament. Cotton is Pakistan's most important cash crop, with a substantial economic and income impact on the country's rural people. Cotton is used for domestic consumption in the form of finished items in about 30%–40% of the time, with the rest being exported as raw material, clothing, and textiles (Saleem *et al.*, 2021).

Figure 1.2 Cotton Export of Pakistan

Crop production goes through various stages of change, influenced by a number of factors. Healthy crop yields and early detection of plant diseases are essential to good agriculture production. Around 18% of crop yields are lost each year due to diseases and pests, which results in a loss of millions of dollars worldwide. In Pakistan farmer used the conventional method to detect diseased plants and pests by their own knowledge and experience. The bared eyes result in a loss of low-level accuracy in order to detect any diseases and it can be expensive and time taken. On high demand, new advanced technologies were incorporated to utilize systems. Plant diseases were identified by using advanced technology, increasing productivity, which led to raising the economy through boosting the production. For this reason, IT-based solutions are used in agriculture. Yet climate change and plant diseases continue to pose a huge threat to food security (Chen *et al*., 2021).

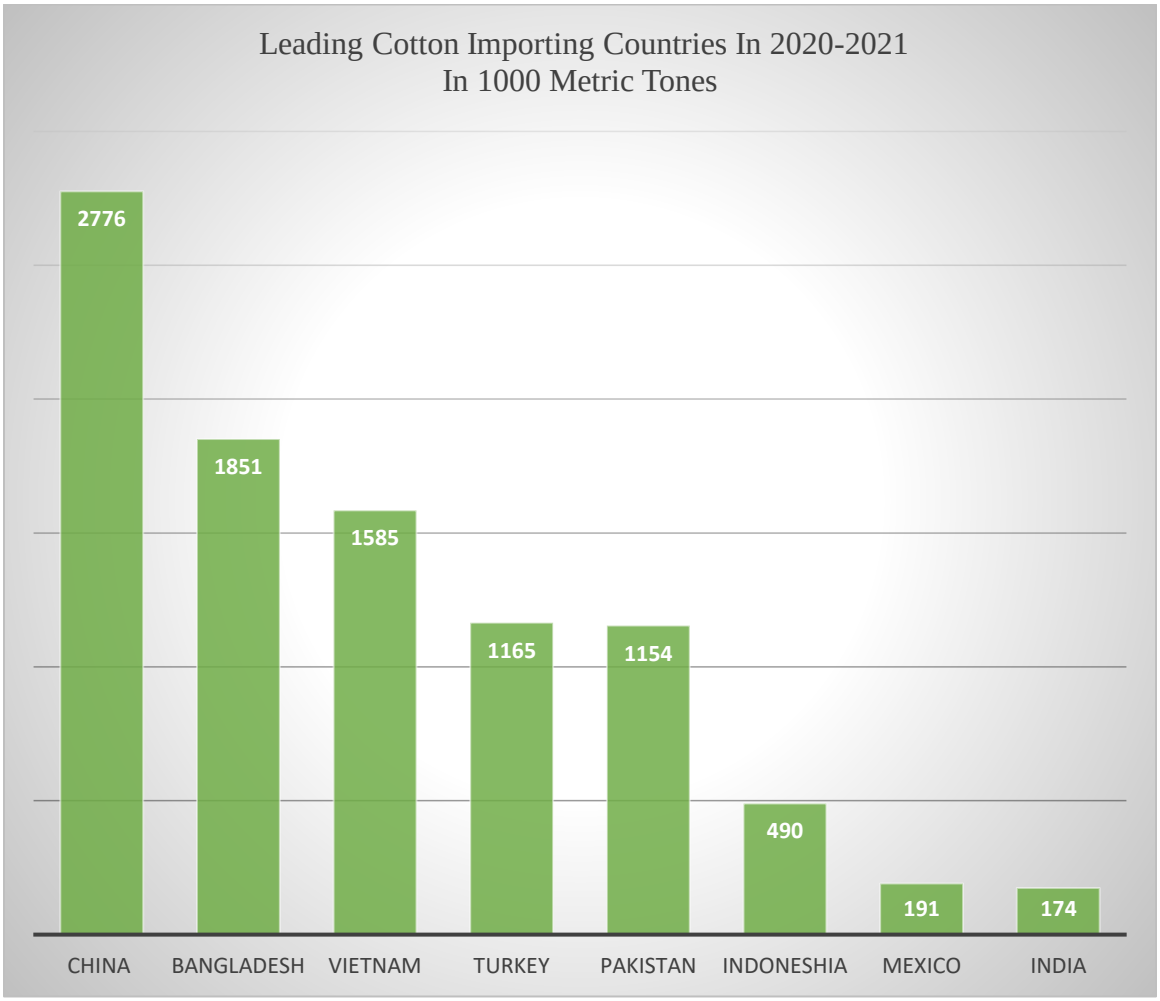


Figure 1.3 Leading Cotton importing Countries in the World

Nearly three million hectares of land are cultivated along the Indus, and the economy is heavily reliant on it. In Pakistan, the cotton value chain employs around 1.5 million people. Cotton seed oil provides almost 17.7% of Pakistan's oil consumption.

According to statistical bureau of Pakistan cotton crops area and production from 2011 to 2021.

Table 1.1 Area and Production of Cotton in Pakistan

Area and Production of Cotton Crops		
Crops/ Year	Cotton	
	Area	Production
2011-12	2834.5	13595.0
2012-13	2878.8	13030.7
2013-14	2805.7	12768.9
2014-15	2961.3	13959.6
2015-16	2901.9	9917.4
2016-17	2488.9	10670.6
2017-18	2700.3	11945.6
2018-19	2373.0	9860.8
2019-20	2517.3	9148.0
2020-21(P)	2078.9	7063.9
*Cotton production is in thousand bales of 375lbs. each P= Provisional (area‘000’ hectors, production ‘000’tones)		

Cotton has traditionally been a popular crop for raw material in the textile industry, thus correctly dealing with the cotton problem at the right moment is critical to saving the textile sector. Cotton's issue could be a direct and important contributor to the textile industry's predicament. The cotton bolls are the most important portion of the cotton plant, and around 20%–30% of illnesses may be identified by looking at the cotton bolls. Plant diseases are a serious threat to crops and plants. Cotton crops can be severely harmed, if not completely destroyed, by these diseases. Diseases wreak havoc on about half of the world's agricultural crops each year (Latif *et al*., 2021).

Cotton, is constantly harmed by various pests and illnesses during its growth, putting its output and quality at risk. According to estimates from the United Nations' Food and Agriculture Organization, the loss rate of cotton owing to illnesses might be as high as 24%. To ensure high-quality cotton output, it is critical to accurately detect and anticipate cotton illnesses and to apply suitable control measures in a timely way. Cotton pest management has proven to be a difficult undertaking for agricultural specialists and cotton growers. In Pakistan, insect infestation is expected to inflict yearly yield and quality losses of 20–40 percent (Karar *et al*., 2020)**.**

Insects destroy cotton yield and production in hot, humid conditions, necessitating the use of a huge quantity of pesticides. Cotton production requires a variety of environmental conditions, including fertilizer, pesticides, well-drained soil, and water, all of which can be damaging to the crop if used in excess. These factors have the disadvantage of causing emissions of greenhouse gases and water contamination. Cotton is the most widely farmed crop on the planet, owing to its many uses in the creation of agricultural-based products, textiles, and branded items. Cotton bolls will promote seed dispersal in natural conditions, which will help to boost the food supply for creatures in the food chain. It also leads to an overall increase in agricultural crop production that may be used as a staple food (Sajid *et al.*, 2018).

1.3 Cotton Boll Rot Diseases

Pests, fungi, viral, and bacterial pathogens can cause diseases in cotton plants. In Pakistan, boll rotting is a novel cotton disease. Boll rotting is caused by a variety of fungal infections. In the beginning, the disease’s small spots, which are brown or black in color, appear that expand to

enclose the complete bolls. The contamination goes to tissues present inside, resulting in the infected stem or lint. The bolls never crack open, and they do not immediately fall off in the early stages. The rotting of the stem may be external in some cases, which causes infection to the pericarp that leaves the internal tissues freely. A large number of fungi may be observed on the infected bolls (Ehetisham-ul-Haq *et al*., 2017).

A country with a large proportion of its population reliant on agriculture must produce varied crops to attain the maximum yields and quality products feasible. It can be improved with the help of technology. The textile industry requires cotton all around the world. We interested in examining the cotton sapling's greenery than the whole cotton bush. We know that the human eye's ability to identify an infected region of an image is limited since a minute alteration may signify the presence of a distinct disease on cotton plants. Cotton output is severely hampered by these species.

Figure 1.4 Cotton Boll Rot

Cotton yields were reduced by 20–30% due to boll rot. Cotton boll rot losses vary from year to year, owing to meteorological circumstances, insect vector population, disease presence, and topography. Until infected bolls show outward symptoms, it is extremely difficult to detect them at the early stages of their development (Nagrale *et al.*, 2020).

At first, the illness appears as little brown or black dots that grow to encompass the entire boll. Contamination spreads to inner tissue, infected stem or lint. Cotton bolls never crack open, and they don't fall off right away in the beginning. When the stem is rotting externally, the pericarp can become infected and the interior tissues can move freely. Bolls may display a great number of fungi on the diseased surfaces (Salman Qadri, 2021).

1.4 Statement of the Problem

Cotton is a cash crop which is also known as White Gold and King of Fiber suffering from several diseases. Most of cotton diseases are caused by fungi, bacteria, and viral, so (20 to 30) % of cotton yield loss due to boll rotting. Detection of cotton bolls rot disease is a very imperative factor to prevent severe outbreak. A new method is proposed for careful detection of diseases and timely handling to prevent the crops from heavy losses.

1.5 Research Motivation

In Pakistan, the majority of farmers are still relying on the traditional methods of farming. Cotton is also being grown utilizing traditional agricultural methods. This results in greater production costs for conventional cotton farmers, as well as a worse utilization of resources (Imran *et al.*, 2018).

Cotton is a highly profitable cash crop that is produced all over the world in tropical and subtropical areas. Farmers in Pakistan relied on their expertise and experience to identify diseased plants and pests in their fields. It can be costly and time-consuming to fix the problem. Increasing demand necessitated the incorporation of increasingly advanced technologies into systems. . Cotton crops in Pakistan are plagued by a variety of illnesses that damage the fruits, stem, and branches of the plants. Improved technology led to the discovery of plant diseases, which in turn led to more output and, ultimately, a stronger economy. Boll rot caused by a variety of Fungi, Bacteria, and Viruses As a result, IT-based agricultural solutions are becoming more common (Iqbal *et al*., 2021).

In this study, first, we'll go over the CNN architecture, which was created expressly for object recognition and categorization in image data. We also go through some of the CNN architectures that have been created over the last decade. By using a Convolutional Neural Networks (CNN) to produce region recommendations in order to localise and segment objects, we overcome the problem of detection time and training data availability.

Second, we go over the Faster R-CNN with Inception-V2 and SSD with MobileNet-V2 in detail, including architectural details, fine-tuning each one for object identification and localization, and finally comparing and discussing their advantages and shortcomings in terms of detection time, network complexity, and training time. Finally, we implement our R-CNN for depth estimation task after determining which R-CNN framework is suited for real-time object recognition and localization in an image.

1.6 Statement of Objectives

This research has the following objectives:

- Propose a machine-driven approach for cotton bolls disease identification using deep learning.
- Distinguish between affected and non-affected cotton bolls.
- Implement this approach in real-time counting the cotton bolls rotting disease with the whole population.

CHAPTER 2

REVIEW OF LITERATURE

The concept of cotton disease detection has been around long before the arrival of advanced computing techniques. All sorts of studies have been performed in the last decade to achieve the task at hand with fair amount of accuracy. The ones we came across to develop our proposed method are discussed below:

Yadav and Binay (2017) suggested that it was possible to recognize the SAR and ISAR images to the use of the targeted deep learning detection method. Because of the limitations of the hardware, it was difficult to render radar images with low resolution for plane position indicator (PPI) images. This study proposed a marine target detection method that is both fast and accurate, based on navigation radar PPI images as well as the Marine-Faster R-CNN method (e.g., sea clutter). Responder-back data from radars is used to create PPI images, and convolutional neural networks are used to extract targeted recognition from these images (CNN). Researchers were able to make improvements to an innovative network structure, connector size, tightly packed target detection, and magnitude normalization by utilizing the Marine Faster-RCNN, according to the findings. Following that, a dataset of navigation targets for radars, developed by the Joint Research Centre (JRC), was presented for public consumption. Faster R-CNN outperforms the Faster R-CNN approach in terms of effectiveness, consistency, and expandability, according to the technique that has been proposed. Their hypotheses were tested using data from multiple instances of different sea conditions, radar metrics, and different objectives to ensure that they were correct.

Khan *et al.* (2018) developed a model for detecting and classifying various fruit diseases using correlation coefficients and deep characteristics (CCDF). Derived on feature extraction and classification, the suggested technique for identifying infected areas categorizes infected areas. To identify infected patches, the recommended correlation coefficient-based segmentation method is applied first. Using two deep pre-trained models (VGG16 and caffe AlexNet), disease-specific properties are collected from a sample of selected illnesses (apple scab, apple rot, banana sigotka, banana cordial leaf spot, banana diamond leaf spot and deightoniella leaf

and fruit spot). A parallel feature fusion step precedes the max-pooling stage. After selecting the most discriminating qualities using an evolutionary strategy, multi-class support vector machines classify the data. Testing is presently underway on publicly available datasets like the plant village and CASC-IFW to attain 98 percent accuracy. The proposed solution beats other existing approaches in terms of enhanced precision and classification accuracy, according to qualitative results.

Sandler *et al.* (2018) described MobileNetV2, a new mobile framework that enhances the execution of mobile models activities and benchmark, as well as across a range of model size. They also demonstrated how to effectively employ these mobile models to do object recognition in a new framework called SSDLite, which they developed. They also showed how to utilize a reduced version of DeepLabv3 dubbed Mobile DeepLabv3 to produce mobile segmentation algorithms, which was developed in collaboration with the researchers. The shortest connections were between thin bottleneck layers, and it was built on an inverted residue architecture, which allowed for the shortest connections. Because of the nonlinearity, the secondary expanding layer used a lightweight depth wise convolution layer to filter characteristics as a component of nonlinearity. They demonstrated how this improves performance and served as the idea for this particular design. Lastly, their strategy enabled the input/output domains to be separated from the expressiveness of the transformation, which provided a solid platform on which to build subsequent research. The techniques of ImageNet segmentation, COCO object identification, and VOC image segmentation were all applied in this investigation. There was a balance between the advantages and disadvantages of accuracy, the number of operations measured by multiply-ads, actual latencies, and the number of components.

Ali *et al*. (2019) addressed resolving the problem of real-time object recognition of dental equipment using deep learning algorithms SSD MobileNet. For dental equipment such as a spatula, elevator, and mouth mirror, among other things. To evaluate the success of this unique method, two different neural network models were deployed, each with its own set of parameters (Faster RCNN Inception v2 with SSD, MobileNet V2 with SSD). In compared to earlier trivial system methods and normal machine learning models, their method was more precise and faster in recognizing instruments than any of the other two. In this project, their

team was able to achieve 87.3 percent precision and 98.8 percent accuracy with excellent results.

Wang *et al*. (2019) developed tomato disease detection methods based on deep convolutional neural networks. Faster R-CNN and Mask R-CNN are two different models used in these methods; Faster R-CNN is used to identify the types of tomato diseases, and Mask R-CNN is used to detect and segment the locations and shapes of infected areas. A total of four alternative deep convolutional neural networks are merged with two object identification models in order to determine which one is most appropriate for the tomato disease detection challenge. Using the Internet as a source of data, the dataset is separated into three parts: a training set, a validation set, and a test set that will be utilized in the experiments Plus, the experimental findings demonstrate that the suggested models are capable of properly and swiftly distinguishing between the eleven tomato disease kinds, as well as segmenting the locations and shapes of diseased regions.

Židek *et al.* (2019) described a unique object identification approach that takes use of deep learning networks that are taught remotely using 3D virtual models and is based on deep learning networks. A virtual 3D model generator is used to produce training input datasets for a remote web application that is run from a distant location. To evaluate the success of this unique method, two different neural network models were deployed, each with its own set of parameters (Faster RCNN Inception v2 with SSD, MobileNet V2 with SSD). With this technique, the sample training dataset for 2D models is created incredibly quickly from virtual 3D models, which is a significant benefit. The cost-effectiveness of this strategy cannot be overstated. The full procedure may be carried out in the cloud environment. Because the experiments using typical components (nuts, screws, and washers), they were able to achieve detection accuracy levels that were comparable to those acquired via training with genuine samples. To evaluate the success of this unique method, two different neural network models were deployed, each with its own set of parameters (Faster RCNN Inception v2 with SSD, MobileNet V2 with SSD). Additionally, the recognition processing delays of the learned models operating in portable systems based on an ARM processor and in portable systems based on a normal x86 processing unit were investigated in order to conduct an efficiency comparison between the two processor architecture types.

Adiwinata *et al.* (2020) experimented object detection approach Faster R-CNN to determine the species of fish. Their investigation sought to determine the performance of the Faster R-CNN in comparison to other object identification models such as SSD. The accuracy of the Faster R-CNN reached 80.4 percent, which is a huge improvement over the accuracy of the SSD Model, which achieved 49.2 %. The proposed that machine and deep learning models were the most widely used for object detection. The (Faster R-CNN) is the most critical in deep learning because it is faster than other convolutional neural networks. It has good detection capabilities in everyday situations. It is possible that detection performance will be unsatisfactory when the object is occluded, deformed, or of a tiny size. The research work proposed a novel and enhanced model that made the use of the Faster R-CNN as a framework, contextual information fusion and skip pooling to achieve better performance and efficiency. Under certain conditions, this algorithm makes use of Faster-RCNN to enhance the improvement in detection and performance of designed system. The improvement was further divided into three sections: An additional context information feature extraction model was added after the convolutional layer in order to obtain the object's contextual information, which is especially important when the object is occluded or deformed; and the RPN is replaced with a more efficient guided anchor RPN, which can maintain recall while improving detection performance. The deep neural network technique's various feature layers may be used to extract more specific information from the data, which is very useful in cases where there are a lot of little items. Object recognition approaches such as faster R-CNN, you only look once (YOLOv3), the single shot detector (SSD512), and others are all exceeded by the proposed strategy in terms of mean average accuracy, according to the results (mAP). As a result, the new strategy is more effective than previous methods and has a greater detection rate.

Giron *et al.* (2020) analyzed that currently available R-CNN based generic object detection systems have detection performance that is significantly lower than the overall average for items with low ORPs (object-region-percentages) of the bounding boxes, compared to previous generations of generic object detection systems. Objects that are elongated are a good example of this. To address the issues of reduced ORPs for extended object recognition, they proposed a new model that made use of Faster R-CNN to achieve robust detection systems of object categories, as well as an innovative prototype classification algorithm to group similar partial detection methods and suppress false detections in order to achieve robust detection systems

of object categories. First, a Faster R-CNN was trained with different classifiers of suitable and reliable ORPs that were provided as part of the training process. In order to categorize the alignment on the partial detection systems, a deep CNN was implemented. Faster R-CNN and DCNN outcomes are utilized by the responsive prototype cluster - based algorithm to first activate a framework of an extended object that was created using an information process on local partial detection methods, and then refines the prototype iteratively through the use of prototype grouping and statistical model updating. A strict focused bounding box for extended object detection is produced by combining Faster R-CNN to generate robust partial detection systems with prototype clustering to establish a global representation. This process is described in detail in the paper. Aside from two typical elongated objects from the COCO dataset, we also tested our method on a variety of other commonly encountered elongated objects such as rigid (pencils, pliers, and spanners), as well as other objects. Extensive research has demonstrated that, when compared to current state of the art strategies, our motivation suggests both the detection and characterization of extended objects in images by an extremely large margin.

Lin *et al.* (2020) implemented the CNN models to detect five common apple leaf diseases using images from the field. By applying data augmentation, such as rotational transformations, horizontal and vertical flips, intensity disturbances, and Gaussian noise, 26,377 disease images were generated. The problem with SDD is that it cannot detect a small object, and also, an object can be detected multiple times. To overcome these drawbacks, they developed an SSD model with a VGG-INCEP as the backbone, where two GoogLeNet inception layers replace two convolutional layers of the VGG model. Moreover, the structure of feature extraction and fusion is designed by applying the Rainbow concatenation method instead of the pyramid (which is used in the SSD model) to improve the performance of feature fusion. The results showed that the data augmentation improved the accuracy of the model by 6.91% compared to the original dataset. Moreover, the proposed model achieved the best performance compared to faster R -CNN and SSD with VGG and Rainbow SSD (RSSD) model. To investigate the effect of using a deep model as a backbone, ResNet-101 was used as a feature extractor for SSD. The results show that ResNet-101 does not lead to any improvement. In terms of speed, Faster RNN was the slower model with more accuracy. The proposed model with 78.80% mAP and 23.13 Frames per Second (FPS) was more accurate than SDD and RSSD, but SSD

outperformed the other models in terms of time inference. By examining the misclassified images, it was indicated that similarity between diseases, misclassified background, and light condition were the challenges in classification.

Lorgus Decker *et al.* (2020) implemented an SSD model with two lightweight backbones MobileNetV2 and InceptionV3, to develop an Android APP called Kiwi Detector that detects kiwis in the field. Quantization is a technique for performing computations and storing tensors with bit widths smaller than floating-point numbers. When training a neural network, 32-bit floating-point weights and activation values are typically used. A quantized model performs some or all operations with tensors using integers instead of floating point values. This allows the computational complexity to be reduced and the trained model can be used on devices with lower resource requirements. The quantization method was used to compress the model size and improve the detection speed by quantizing the weight tensor and activation function data of convolutional neural networks from 32 to bit floating-point numbers to an 8-bit integer. The results showed that MobileNetV2, quantized MobileNetV2, InceptionV3, and quantized InceptionV3 achieve a true detection rate of 90.8%, 89.7%, 87.6%, and 72.8%, respectively. The quantized MobileNetV2 on the Huawei P20 smartphone outperformed the other models in terms of detection speed (103 ms/frame) and size. Although the SSD with MobileNetV2 was more accurate than the SSD with quantized MobileNetV2, the SSD with quantized MobileNetV2 was 37% faster. The problem with kiwi detection was that overlapping kiwis were reported, which the model counts as one kiwi.

Patel and Patel (2020) tested and assessed pre-trained frameworks on two datasets, the flower 30 dataset and the flower 102 dataset, both of which include 19679 flower photos each. In this research, the transfer learning approach was used. The suggested model can identify, find, and categorize flowers as well as offer other crucial data such as the flower's name, division, class, subclass, order, family, and herb flower, among other things, through the use of multiclass classification and multi-labeling techniques. This approach cut off preprocessing phase by using Faster R-CNN method and inception V2 architecture directly to find the flowers in image. With a homogenous background as a simple input image, this approach showed a good accuracy in its performance. It has 90% and 91% accuracy in way to detect flowers datasets. Moreover, model gave 94% and 92% accuracy that showed the different pattern between both.

Sivakumar *et al*. (2020) developed and tested object-based CNN models for weed detection in soybean fields using low-altitude UAV video. The mean Intersection over Union (IoU) and inference speed of two object identification models, the Faster RCNN and the Single Shot Detector (SSD), were tested. Except for accuracy and recall, f1 score (IoU), and inference time, the Faster-RCNN model outperformed the SSD framework. But inference time was identical to SSD. It had 0.85 IoU and 0.65 accuracy with 200 ideas (intercept of uncertainty). The SSD with 200 proposals scored 66 (accuracy), 68 (recall), and 67 (f1) (intercept of uncertainty). There was also a lower SSD model optimism threshold than Faster RCNN, suggesting that SSD may have less generalization success than Faster RCNN for mid- to late-season weed identification in soybean fields using UAV data. The object detection model was compared to a patch-based CNN model to assess how well it worked. In weed detection, the Faster RCNN model beat the patch-based CNN with and without overlap. This was faster than a patch-based CNN without overlap, but slower than a patch-based CNN with overlap. The Faster RCNN model outperformed the other models in terms of weed detection and inference time. In order to properly grasp the technology's potential and limits, one must first understand the weed identification and control algorithms.

SARDOĞAN *et al.* (2020) trained the SSD object detection model to identify three diseases, two types of pest damage, and nutrient deficiency in cassava at the mild and pronounced stages. A dataset was collected from the field, which was divided into 80–20 as training and testing sets. The model achieved 94 ± 5.7% mAP on the test dataset. The trained model was used on a mobile phone to investigate the performance of the model in the real world. One hundred twenty images of leaves were captured using a mobile device of the study experiment, and the model inference was run on a desktop and mobile phone to calculate the performance metrics of the trained model. The results show that the average precision of the model on the real dataset decreases by almost 5%, but the average recall decreases by almost half, and the F1-score decreases by 32%. Furthermore, the results show that the model performs better on the leaves with pronounced symptoms than on the leaves where the symptoms are only mildly pronounced.

Sethy *et al.* (2020) applied the SSD network to detect mango and almond on tree fruits. The dataset was used to train the model. The SSD model used the original and sampled a patch

such that the minimum Jaccard overlap with the objects is 0.1, 0.3, 0.5, 0.7, or 0.9, and finally randomly sampled the input image. The size of each sampled area and the minimum Jaccard overlap are critical for object detection. These two parameters were changed to adopt the SSD model for small mango detection. Using VGG16 as the basic framework for the SDD outperformed the SSD with ZFNet. Furthermore, using the input image size of 400 × 400 and 500 × 500 as the input of the model, the model achieved better performance than SSD with an input size of 300 × 300. The model outperformed the faster RCNN in terms of speed and accuracy. The challenges related to mango detection on the tree were considered in the paper: the size of mango on the whole image, blocking the mango with leaves, branches, resizing the original image, making mango smaller, and similarity between mango and background.

Srivastava *et al.* (2020) used Faster R-CNN and SSD MobileNet-V2 to highlighted speed, accuracy, and training methods for explicit content identification. Using depth wise break the chain of causation, the integration of the MobileNet-V2 and SSD architecture resulted in an effective object recognition model with precision of 83.33 percent on the multiple classification task and 91.66 percent on the binary classification task when applied to both tasks. They suggested that in order to track an object across multiple video frames, it is necessary to consider a number of factors, including occlusions, clutter in the background, lighting, and variations in object and camera view-points, all of which have an impact on the object detection process. In the case of images captured by unmanned aerial vehicles (UAVs), these considerations are emphasized even further because vehicle movement can have an impact on the quality of the image captured. The most common strategy for dealing with these issues involves analyzing the image data from the input source at various scales in order to obtain as much information as possible, which is then used to correctly detect and track the objects across video sequences. They presented a novel multi-stream (MS) architecture that is simple but effective, it conducted experiments with a variety of kernel sizes to simulate multi-scale image processing and published the findings. In order to build on the architecture that they supplied as the foundation for the widely used Faster-R-CNN pipeline, the researchers were able to create an MS-Faster R-CNN object detector that successfully recognizes objects in video sequences. In order to acquire real-time tracking capabilities on UAV photographs using the Simple Online and Real-time Tracking with a Deep Association Metric approach, this detector is used. Deep SORT is a technology that allows for real-time tracking capabilities

on UAV photos. Testing of the proposed architecture and its components was performed on the different datasets, as well as on other datasets. Based on the performance of the supplied pipeline, which was considered to be state-of-the-art at the time of publication, it was demonstrated that the proposed multi-stream approach can properly imitate the robust multi-scale image analysis paradigm.

Vasconez *et al.* (2020) tested Faster R-CNN with Inception V2 and the Single Shot Multibox Detector (SSD) with MobileNet, which are both used in conjunction with Inception V2. In a variety of field situations, these detection models were trained on avocados, lemons, and apples, and their performance was assessed. They used multi-object tracking based on Gaussian estimate to solve the challenge of video-based fruit counting, which they found to be effective. While combining Faster-RCNN with Inception V2, their model achieves fruit counting efficiency of up to 93 percent, and when employing SSD with MobileNet, their method obtains fruit recounting efficiency of 90 %. These findings have the potential to improve the efficiency with which agricultural decisions are made.

Amisse *et al*. (2021) used faster R-CNN with Inception-V2, SSD with Inception-V2, and SSD MobileNet-V2. Following the examination of the detectors, it was discovered that the amount of training data variation has a bigger influence on accuracy and recall than the quality of the training data variance. Over five variations of the amount of training data, R-CNN with Inception-V2 backbone, SSD with Inception-V2 backbone, and SSD with MobileNet-V2 backbone were shown to be faster than R-CNN with Inception-V2 backbone. During the training stage, on the other hand, the SSD reacts very quickly. Even with just 700 datasets, splitting 80% trained models resulted in an effective detector when 80 percent of the overall data was split for fine-tuning models.

Chen *et al.* (2021) stated that Detecting moving vehicles and pedestrians was a significant problem that needed to be addressed. Discovering a detection mechanism that strikes an optimal balance between detecting precision, detection speed, and memory utilization was becoming increasingly difficult to come by. This was accomplished by taking into consideration the most popular feature extractors, as well as prominent object detection techniques such as Faster R-CNN and SSD MobileNet V2. Faster-RCNN achieved the highest

overall accuracy of 58 percent for vehicle and pedestrian detection when it was trained on the most widely used and well-known KITTI street dataset, which was used in the study. The detection of automobiles and pedestrians was accomplished successfully using R-CNN and Inception V2 as feature extractors (74.5 percent and 47.3 percent). When it came to memory and parameters, ResNet101 had used the most (9.907 Giga-Byte) and the most (64.42 million), whereas Inception with ResNet V2 was the slowest (3.05 frames per second). SD MobileNet V2 was the fastest in terms of frames per second (70 FPS) and the lightest (less than a Giga-Byte) of the three detectors tested (875 Mega-Byte). In the case of architectural analysis, genetic machine learning and reinforcement learning were employed. The networks that resulted were also found to be extremely difficult to understand. They had hoped to learn a great deal about neural networks and then apply what they had learned to build the most basic network feasibility possible. The approach was also comparable and more efficient than previous approaches, while also revealing the fundamental mechanics of the system as a whole. The MobileNetV2 framework was used in the development of the network. However, while mobile apps kept things simple and didn't necessitate the addition of additional operators, they improved accuracy, resulting in the best possible results on a variety of categorization and detection tasks.

Darapaneni *et al*. (2021) suggested the creation of a system for traffic monitoring and reporting at toll plazas in India, which was rejected. Vehicle detection and identification, vehicle registration detection, character recognition are only a few of the areas where research is being carried out at the present time. When it comes to computer vision, object detection and localization precision has always been a difficult problem to solve. To address this issue, the Tensor Flow Object detection API includes implementations provided by the RCNN family as well as Single-shot detection models in the form of Tensor Flow. They discovered by experimenting with a variety of models, including Faster RCNN with inception v2, SSD MobileNet, and SSD inception v2, for the vehicle and vehicle registration number detection problem. To identify the automobile registration number that was discovered, we used an SVM-based multiclass classifier that was trained on the data. The model is trained with the help of a special picture dataset that was built expressly for this application. For the car make classification, the VGG16 classifier is used, and the dataset was selected such that it comprised only autos from India.

Hassan *et al.* (2021) used pre-trained MobileNet V2 networks to identify twelve rice plant diseases. The dataset was collected from online sources and real agricultural fields. An attention mechanism was added to the model, and transfer learning was performed twice during model training to achieve better performance. The MobileNet-V2 achieved 94.12% accuracy on the Plant Village dataset, while the MobileNet-V2 with attention and transfer learning achieved 98.26%. The five CNNs such as MobileNetv1, MobileNet-V2, NASNetMobile, EfficientNet-B0, and DenseNet121 were selected for comparison with the proposed models. In addition, two other networks named MobileNetv2-2stage and MobileAtten-alpha were trained. In the MobileNetv2-2stage model, transfer learning was performed twice to identify the images of plant diseases. Similarly, in MobileAtten-alpha, the attention method was used instead of transfer learning twice. The proposed model (using attention and transfer learning twice) achieved the the second best accuracy of 98.84%. The DenseNet121 with an accuracy of 98.93% outperformed other models. MobileNetv2-2stage and MobileAtten-alpha achieved an accuracy of 98.68% and 96.80%, respectively. The proposed model was trained with images from the field and achieved an accuracy of 89.78%.

Iyer *et al.* (2021) developed a DL model, called Mango-YOLO, based on YOLO-v3 and YOLO-v2 (tiny) for counting mangoes on trees. YOLOv2 (tiny) has a small architecture with only nine convolutional layers, six pooling layers, and one detection layer, sacrificing accuracy for speed. YOLOv3 is based on Darknet-53 and improves upon YOLOv2. The Mango YOLO had 33 layers compared to 106 layers in YOLO-v3 and 16 layers in YOLO v2 (tiny). The reduction in the number of layers is expected to reduce computation time and detect mangoes more accurately. The model was trained on the dataset collected from five orchards. The Mango-YOLO achieved better performance with an accuracy of 0.967% compared to YOLO-v2 (tiny) (0.9%) and YOLO-v3 (0.951%). In terms of time inference, the Mango YOLO with 15 ms was faster than YOLO-v3 (25 ms) and slower than YOLO-v2 (tiny) (10 ms). Moreover, Mango-YOLO was trained once from scratch on the augmented dataset, and the second time transfer learning was used using the COCO dataset. The models had the same performance on the test set, and the reason was explained by the fact that the COCO dataset does not contain Mango images. The false detection over images taken with Canon camera shows resizing of the images and results in image distortion with leaves taking a curved shape resembling the fruit and overexposed areas on branches, trunks, and leaves.

Karaman *et al.* (2021) stated a smart camera system prototype built utilizing an artificial intelligence network that uses a bird's-eye perspective to analyze social distance. Individuals have been identified using "Faster-RCNN ResNet-50," "Faster-RCNN Inception-2," and "Faster-RCNN Inception-3" models. While the first prototype used a camera to measure social distance, the second is an integrated embedded system. Users are notified by audible and visual notifications when social distance was violated, and the programed subsequently uploads the data it has acquired to a server. The program was developed using the "Nvidia Jetson Nano" programming environment and the Raspberry Pi camera module. "Sustainable smart cities" have been proposed as a framework through which to incorporate the smart camera prototype produced by researchers into public settings. They suggested that in the near future, it was possible that mobile cancer detection cameras will be developed for smartphones. Skin cancer detection was pre-installed on the phone's memory. Convolutional neural networks are capable of detecting and categorizing diseases in real time. In order to use this method, a computer with more processing power and memory than a smartphone is required. Images of actinic keratosis and melanomas were grouped together regardless of the gender or age of the subjects. In this study, skin cancer was detected with the help of a smartphone camera. To create an intuitive screening system, faster R-CNN and MobileNet v2 are used in conjunction with MobileNet v2. This experiment was carried out with the help of an Android camera. R-CNN performed better with Jupyter on a smartphone than slower R-CNN, and MobileNet v2 performed similarly on both.

Liu *et al.* (2021) conducted an exhaustive assessment of newly developed deep learning algorithms for the detection of microscopic objects. They discussed the problems and solutions associated with the recognition of small items, as well as the key deep learning strategies, such as the fusion of feature maps, the integration of context information, the balance between foreground and background instances and the provision of adequate positive examples. Researchers have developed comparable algorithms to face detection and object detection in aerial photos in four different domains. Segmentation is only one of a number of similar approaches that they discussed. Additionally, this study compares the performance of many common deep learning algorithms for the detection of small objects, including YOLOv3, Faster R-CNN, and SSD. Faster R-CNN and YOLOv3 were the two deep learning algorithms

that fared the best in our tests, with a detection accuracy of less than 0.4. Due to the algorithms being trained on little things, we think this is the cause.

Rahmaniar and Hernawan (2021) highlighted the performance of edge devices for person identification techniques like PedNet, SSD MobileNet V1, SSD MobileNet V2, and SSD inception V2 performed well. They provided an overview of alternative methodologies and compared their performance for real-time applications in terms of accuracy and computation time. While compared to other approaches, the SSD MobileNet V2 model gave the accuracy while also requiring the shortest calculation time when evaluating video datasets under a range of conditions.

Siddiqua *et al.* (2021) focused on developing a model to increase the identification of cotton leaf pest and diseases using the CNN deep learning method Cotton leaf diseases and pests such as bacterial blight, spider mites, and leaf miners have been used by the researchers in this study. The K-fold cross validation approach was used to partition the dataset and increase the CNN model's generalizability. Over 2400 samples (600 photos each class) were available for this study's training implemented in Python 3.7.3 and is provided with Keras, TensorFlow and Jupyter as the development environment for a deep learning model developed in Python. In cotton plants, this model was able to correctly identify 96.4 percent of the different types of leaf disease and pests shown the potential for real-time applications and the necessity for IT-based solutions in order to assist manual payroll disease and pest diagnosis.' stated that a combination of image processing, faster RCNN with Inception version 2.0, and other machine learning techniques were used to detect dengue mosquitoes during the epidemic of the disease. For that purpose of evaluating the overall usefulness and accuracy of the system, the researchers employed a mosquito dataset that had been specifically constructed and assembled from publically available internet habitats. Models such as SSD MobilenetV2 ResNet 101 were compared to the proposed Faster RCNN with Inception version 2.0 model, which uses a faster RCNN with Inception version 2.0. To evaluate the success of this unique method, two different neural network models were deployed, each with its own set of parameters (Faster RCNN Inception v2 with SSD, MobileNet V2 with SSD). When evaluating the model's detection accuracy, a range of performance measures were utilized, including the number of false positives and false negatives.

Salman Qadri (2021) compared the performance of two item identification models, the Faster RCNN and the Single Shot Detector (SSD), on weed recognition using mean Intersection over Union (IoU) and inference speed. Surprisingly, the Faster RCNN model with 200 box proposals outperformed the SSD model. For soybean field weed detection utilizing unmanned aerial vehicle imagery, the SSD model's optimal optimism threshold was shown to be substantially lower than Faster RCNN's. The object detection model was tested using a patch-based CNN model. Faster RCNN enhanced weed detection in CNN models with and without overlap. Both models had this flaw. Faster RCNN was faster than patch-based CNN without overlap but slower than patch-based CNN with overlap. Finally, the Faster RCNN model outperformed the other models in terms of weed recognition and inference time. To fully grasp the technology's promise and limitations, near real-time weed identification and management algorithms must be recognized.

Wang and Xiao (2021) investigated DCNN models under diverse settings for a full comparative analysis SSD with MobileNet-V2, Faster RCNN with ResNet-50, and Faster RCNN with Inception-ResNet-V2. When GAN-based augmentation is used, all three models show a consistent and considerable increase in mean average accuracy. The appropriately balanced dataset also eliminates category asymmetry, allowing deep convolutional neural models to be trained equally across all categories as a result of the right balance. The descriptive data also reveal that models trained in the richer environment were able to distinguish important regions and object boundaries, which resulted in an improvement in mean absolute precision (mAP). Last but not least, they revealed that the least costly model, the SSD-MobileNet V2, had a same mAP (91.81 percent) and a faster inference speed (102 frames per second), demonstrating that it was well suited for real-time.

Zaki *et al.* (2021) proposed that the visual symptoms of citrus diseases, as well as testing in agriculture laboratories, can be used to diagnose the diseases. These methodologies may not be available to all farmers due to the high cost of experts and the scarcity of testing laboratories in rural areas. The proposed study compares different Convolutional Neural Network (CNN) structures for the classification of citrus leaf diseases, with the goal of improving classification accuracy. To recognize and identify citrus leaf diseases in the vegetative stage, this paper used two types of CNN architectures: the SSD MobileNet and the Self Structured Convolutional

Neural Network (SSCNN) classifiers, both of which were developed by the authors. The research work resulted in the creation of a citrus disease dataset that was based on smartphone photos. Both models were trained and tested using data from the citrus dataset. For the purpose of evaluating the model's performance, the precision and loss of training and testing sets were calculated, respectively. The SSD MobileNet CNN's (Convolutional Neural Network) best training accuracy was 98 percent at epoch number 10, and its best validation accuracy was 92 percent at epoch number 10. In spite of this, the SSCNN had the highest training precision of 98 percent and the highest accuracy rate of 99 percent at epoch 12.

Anuar *et al.* (2022) trained SSD MobilenetV1 and the Inception model to detect Grape Bunch in Mid Stage and early stages and then transferred the trained model to the TPU Edge device to investigate the temporal accuracy of the model. The same strategy in was used to collect the dataset and it is publicly available with 1929 vineyard images and their annotation. Overall, SSD MobileNet-V1 performed better than the Inception model, as it had a higher F_1 score, AP, and mAP. The early stage was more difficult to detect than the middle growth stage. The first class represented smaller clusters that were more similar in color and texture to the surrounding foliage, making them more difficult to detect. SSD MobileNet-V1 showed an AP of 40.38% in detecting clusters at an early growth stage and 49.48% at a medium growth stage. In terms of time, SSD MobileNet-V1 was more than four times faster than the Inception model on TPU-Edge Device.

Mostafa *et al.* (2022) distinguished between distinct guava plant species, they employed a deep convolutional neural network (DCNN)-based data improvement approach that combined color-histogram equalization with the unsharp masking technique. Nine 360-degree viewpoints were used to increase the number of converted plant images. They were then fed into cutting-edge classification networks to improve their classifications even more. It was necessary to normalize and preprocess the proposed strategy before putting it into action. Pakistani guava disease cases were used for the experimental evaluation, which was based on a dataset of local cases. It is hoped that the proposed study would use five neural network topologies to distinguish diverse guava plant species. With a classification accuracy of 97.74 percent, ResNet-101 was found to become the most effective classification system in the test results.

CHAPTER 3

MATERIALS AND METHODS

This chapter will discuss the proposed methodology for the detection of disease in cotton crops. The data collection, images preparation, data annotation and the implementation of deep learning model.

3.1 Introduction to Deep learning

The term "artificial intelligence" refers to a computer that can mimic human intelligence. Machine learning allows computers to run programs based on a variety of algorithms. DL is a subclass of ML that enables computers to learn in real time from both the training dataset and human experience. Deep learning and computer vision have become a reality in recent decades. Deep neural networks, as well as GPUs or NPUs, have a lot of power when it comes to intense computing. This creates a powerful hardware infrastructure for deep learning development.

Deep learning (DL) algorithms have numbers of advantages over typical machine learning methods. To begin, noisy data can be filtered out, deep learning algorithms detect patterns using a training dataset, then output class labels for use in real-world classification issues. Furthermore, deep learning algorithms, particularly time series analysis, detect and recognize computable events based on valuable patterns. Deep belief methods (DBM) and CNN are used to easily classify unstructured and structured data.

Although artificial neural networks mimic the mechanisms of our human brains, deep learning models require time to learn how to solve issues from beginning to end (LeCun, Bengio, & Hinton) which means that the imported data and its attributes will be unaffected by neural network architecture. Deep learning has its own set of obstacles, such as the requirement for a significant amount of data to be input into deep learning techniques for model training, and the collection of successful pattern classification using deep neural networks.

The model's usefulness will be diluted due to over fitting in pattern classifications. A well-trained deep neural network should be able to do not just pattern classification but also transfer learning, which is a new task. Deep neural networks are very ideal for empirical applications,

eventually serving the community in practical ways. More deep learning approaches will be used in sectors including remote diagnosis, intelligent surveillance, autonomous cars, robotics, and smart agriculture in the near future. Deep learning approaches will have a protracted impact on the field of computer vision because processing power is now skyrocketing. Deep learning will also significantly improve technologies in computer vision and digital image processing. Deep learning will undoubtedly help intelligent learning.

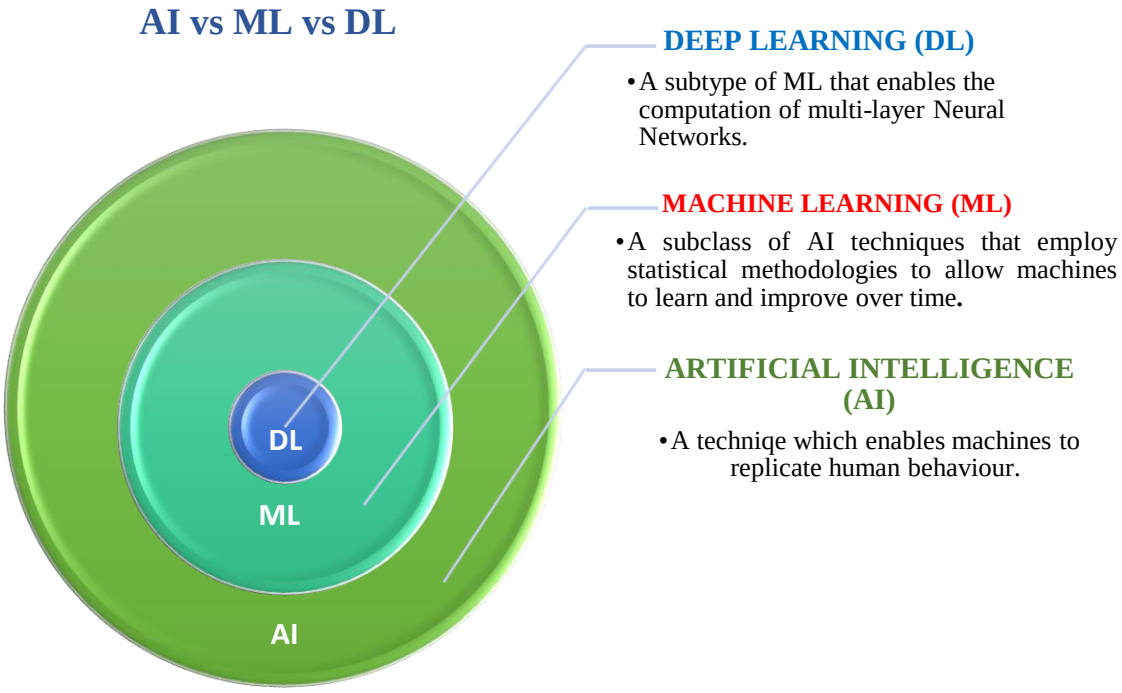


Figure 3.1: Hierarchy of Deep Learning

3.2 Examples of Deep Learning (DL)

3.2.1 Automated driving: Researchers are currently employing deep learning to develop self-driving automobiles in order to minimize the number of accidents.

3.2.2 Aerospace & Defense: The army and other military departments utilize deep learning to locate safe and risky zones.

3.2.3 Medical area: Deep Learning (DL) is now being utilized in the medical field to detect a wide range of disorders. Researchers at the University of California developed a microscope that can generate a data collection with extremely high dimensions, which may then be used

to train Deep Learning (DL) applications for the diagnosis of extremely severe diseases such as cancer.

3.2.4 Industry automation: Deep Learning (DL) can automatically recognize whether a worker is in a dangerous zone while working on heavy machines. As a result, deep learning is being used to save human lives.

3.2.5 Electronics: is an important applications of Deep Learning (DL). It can be used for automatic hearing as well as voice translation.

3.3 Deep Learning Based Object Localization and Classification Methods

To evaluate data and learn features, deep learning relies on the usage of neural networks, which are trained to do so. Data characteristics are extracted from the data by the use of many hidden layers, each of which may be thought of as a perceptron. To extract restricted characteristics, the perceptron is employed. These limited characteristics are then combined to form high-level features, which may meaningfully alleviate the local minimum issue.

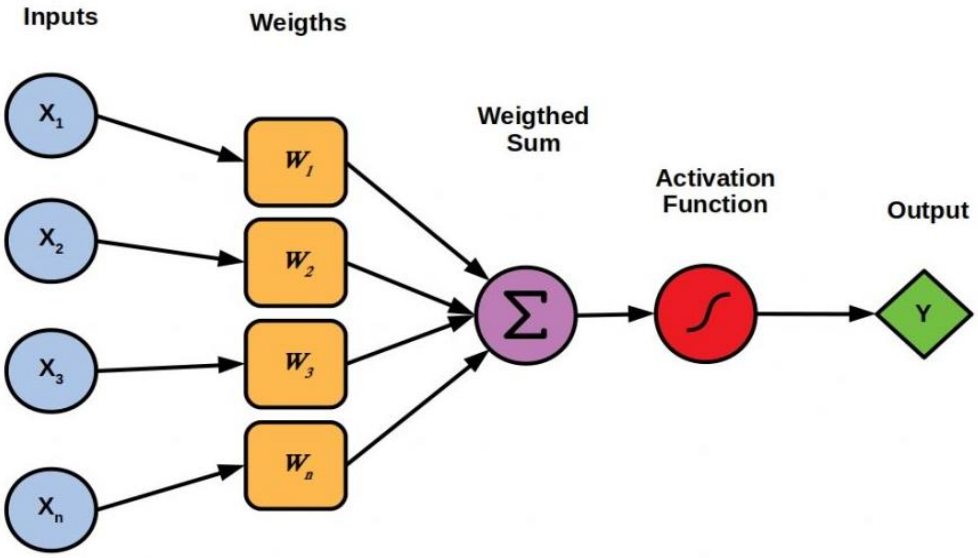


Figure 3.2: Architecture of Perceptron

DL overcomes the shortcomings of previous techniques that rely on artificially produced characteristics, garnering the interest of a growing number of academics. It is currently utilized with considerable success in computer vision. Customary manual design feature classification and identification approaches can only extract the underlying characteristics, making deep and

sophisticated picture feature information impossible to retrieve. It may be possible to overcome this obstacle with the use of deep learning methods. With the original picture as a starting point, it may do unsupervised learning to extract multi-level image feature information, such as semantic features at several levels of abstraction (low, middle, and high). In order to identify plant diseases and pests, traditional algorithms depend on the challenging picture identification approach of manually constructed features, which is incapable of learning and extracting characteristics from the source image on its own initiative. Deep learning, on the other hand, is capable of extracting features from large data without the need for human intervention or supervision (Chen *et al.*, 2021).

The model is made of multiple layers and is capable of extracting visual information independently for the purposes of picture classification and identification, as well as object recognition. It also has a great degree of autonomy in terms of learning and expressing its features. Because of this, deep learning has a lot of promise in the area of plant disease and pest detection, among other fields. Deep neural network models, such as the DBN, SDAE and deep convolutional neural network, have been developed by implementing deep learning techniques (DCNN).

With regard to picture identification, the use of deep neural network models for automated feature extraction from high-dimensional feature space provides substantial benefits over the usage of standard manual design feature extraction approaches. Deep neural networks' characterization power rises as the quantity of training samples and processing capacity increase, and this is particularly true for convolutional neural networks. Deep learning is now spreading across both business and academia, with deep neural network models outperforming classical models by a wide margin in many cases.

DCNNs have been the most extensively used deep learning framework during the last several years. In contrast to object classification, which is concerned with global description, object detection is concerned with local description, that is, with answering the question of whether an object exists in what position in an image, whereas object classification is concerned with global description through feature expression and then determines whether there is a specific kind of object in the image through classification operation; while object classification is concerned with worldwide characterization, object recognition is concerned with local

characterization, that is, with answering what object exists in what position in an image; while object classification is concerned with worldwide.

Learning-based methods, which can adapt to changing environmental conditions, are one way to overcome difficulties with fundamental methods. Basically, there are total two types of machine learning algorithms or methods: 1) Supervised 2) Unsupervised. The Unsupervised techniques like clustering with K-means and fuzzy clustering, were popular at first. Later, artificial neural network-based learning methods were used, beginning with random forest methods and progressing to convolutional neural networks (CNN).

First, category-independent regions (ROIs) based on the objectness measure are generated from an image in these frameworks. In the following step, CNN-based features are extracted from the ROIs, and in the final step, the extracted features are fed into classifiers in order to determine the ROIs belongs to which category. CNN architectures are used for object detection in methods such as RCN, OverFeat, DetectorNet, and MultiBox, which were developed almost simultaneously and use CNN architectures. Some of the methods that fall into this category are briefly described in the following sections.

The convolutional, ReLU, and pooling layers are found in the early layers of a CNN, while the latter layers are generally made up of fully connected layers. The input image is repeatedly convolved in layers of a CNN, and with each layer, the receptive field of the CNN grows larger. CNNs are divided into layers, with the first layer responsible for extracting low-level data (e.g., color, edge) and the second layer responsible for extracting more complicated features. DCNNs may learn very complex functions and represent data in numerous levels of abstraction using hierarchical structure. In other words, it is no longer necessary to look for the best set of attributes that correctly characterize the objects of interest in varied environmental situations and object variants. The existence of huge training sets is crucial to the success of DCNNs.

Deep learning can be applied to a variety of difficult tasks, including:

1. Artificial Neural Networks (ANN) are utilized for regression and classification.

2. Convolutional Neural Networks (CNN) are employed for computer vision tasks.

3. To conduct a time series analysis The RNN (Recurrent Neural Network) is used.

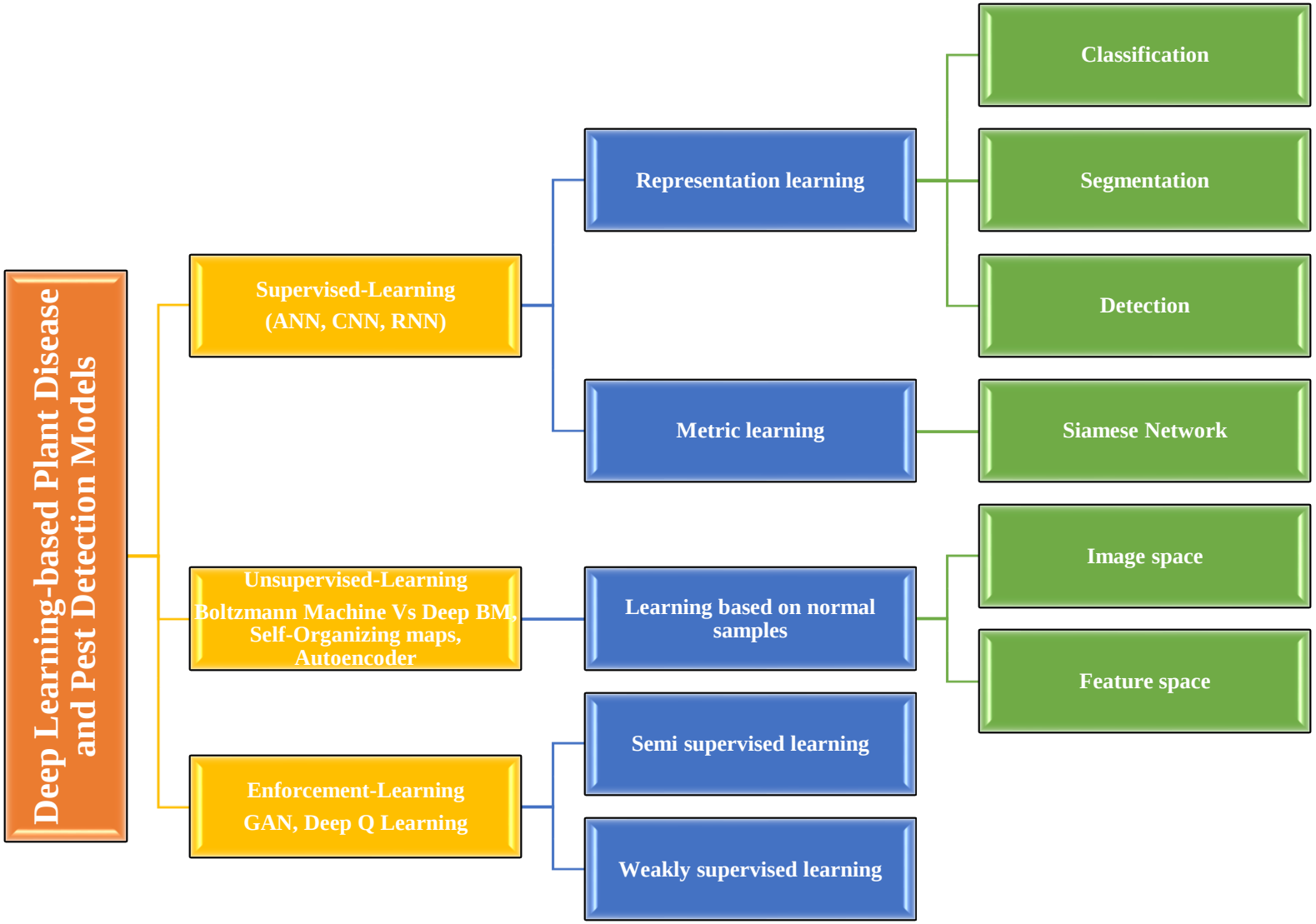


Figure 3.3: Hierarchy of Deep Learning Detection Models

3.4 Architecture of Convolutional Neural Networks (CNN)

The popularity of machine learning was increasing following the popularity of Artificial Neural Networks (ANN). ANN was a non-linear complex learning system that occurs in a network of neurons. Convolutional Neural Network (CNN) was one of the most developed ANN derivatives currently. CNN was a deep learning algorithm that uses a convolutional layer for feature extraction and a fully connected layer for classification. CNN could be applied in image and text classification .This technique is based on multilayer Artificial Neural Networks (ANNs) with an integrated component extraction process, which are used in conjunction with an integrated component extraction process. It is as a result of this that CNNs are capable of reliably recognizing objects in pictures, even when there is background noise present. CNN may be able to add object-detection technology that has been specifically designed for the network (Rivas *et al.* 2018).

Recent years have seen significant improvements in the accuracy of object identification and classification using CNNs (Convolutional Neural Networks). LeCun and colleagues were the first to propose Convolutional Neural Networks (CNN). The CNN's initial two layers are Convolution and Maximum Pooling. The convolutional layer of a convolutional neural network employs a variety of convolution kernels to produce feature maps, which are then used to train the network. The output layer of a CNN makes classifications based on characteristics gathered from the input layer of the network.

These sophisticated and extensive characteristics are recovered in down-pooling by shrinking the size of feature maps, which is a technique used in machine learning. Agriculture has been practiced by humans for thousands of years, making it an old and important activity in the history of mankind. Recent population expansion has led in a rise in demand for agricultural goods, which has resulted in an increase in the price of agricultural products. It is being fulfilled by integrating automation into agriculture, which does not affect the environment as a result of its implementation. The goal of this project is to combine deep learning with agriculture in order to identify the cotton boll rotting disease and prevent it from spreading. On the basis of the illness's characteristics, plant disease detection algorithms often improve on the backbone structure or its feature map, anchor ratio, ROI pooling, and loss function (Magsi *et al.*, 2021).

At first image will be taken as an input and then performed the sequence of convolution and pooling layer which is a fully connected layer. The output dimensions of a layer can be computed using this equation in any case:

$$InputSize + 2 * Padding - FilterSize\ Stride + 1$$

The following equation for calculating the number of hyper parameters in each layer:

$$(n * m * l + 1) * k$$

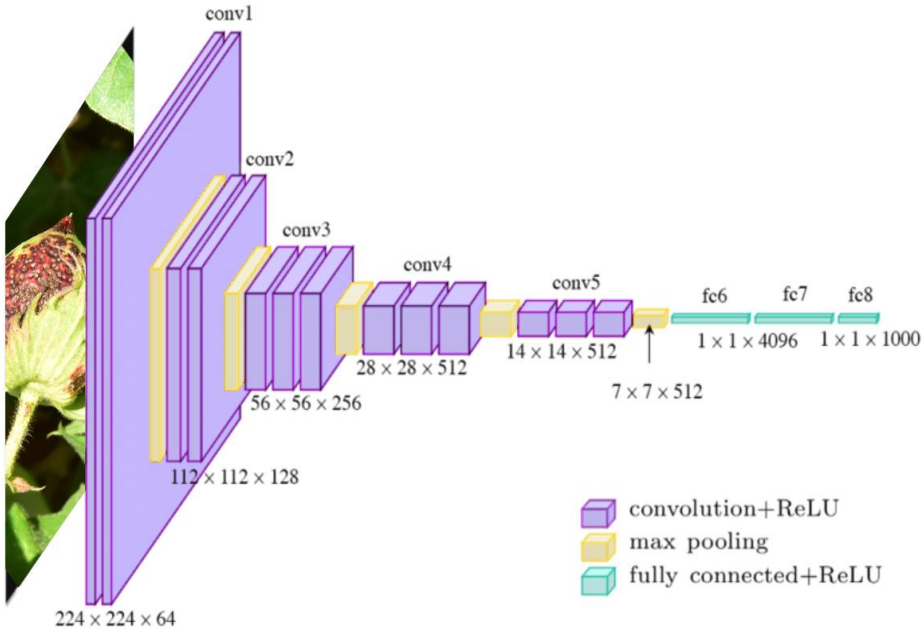


Figure 3.4: Architecture of CNN

Softmax is an output if multiclass classification is performed. It is divided into different components are as follows:

3.4.1 Convolutional layer

CNN relies heavily on the convolutional layer, which is a crucial component. The information is combined in a convolutional manner. A convolution filter is applied to a convolution layer in order to perform convolution. A feature map is generated. Confusion is a two-way street. It is possible to swap out the convolution filter on the right for the input images on the left. The filter's convolution form is 3 x 3. On the input, this operation employs the filter. Do-matrix multiplication and sum the results are done at every spot. Receptive field dimensions are 3x3 When this is done, slide the filter again and repeat the same process. Show the convolution

operations as you go along in this process. Matrixes of height, width, and depth are used to represent an image. Depth utilizes a variety of distinct color channels (RGB). Various filters are used in the convolutions we carry out. The depth of a filter is completely covered by its height and width (3*3 and 5*5). CNN relies heavily on the convolutional layer, which is a crucial component. The information is combined in a convolutional manner. A convolution filter is applied to a convolution layer in order to perform convolution.

A feature map is generated. Confusion is a two-way street. It is possible to swap out the convolution filter on the right for the input images on the left. The filter's convolution form is 3 x 3. On the input, this operation employs the filter. Do-matrix multiplication and sum the results are done at every spot. Receptive field dimensions are 3x3 When this is done, slide the filter again and repeat the same process. Show the convolution operations as you go along in this process. Matrixes of height, width, and depth are used to represent an image. Depth utilizes a variety of distinct color channels (RGB). Various filters are used in the convolutions we carry out. There is a 3 x 3 x 5 filter that spans the entire depth.

3.4.2 Non- linearity

Non-linearity enhances the power of neural networks. The activation function of ANNs and auto encoders, as well as CNNs, is the same for all three. The activation function is passed through the convolution result. The ReLU function is applied to the values of the features map, not the sum of the individual values. ReLU is a convolution function that incorporates any form of function.

3.4.3 Padding and stride

In each step, stride indicates how much movement there is in the convolution filter. Because a feature map's size is smaller than the input, the convolution filter can fit the data in. The padding may be seen in the image as the grey region. The values remain on the cutting edge. CNN makes use of padding to adjust the size of its graphics.

3.4.4 Pooling

Dimensionality is reduced to execute the pooling after the convolution procedures. As a result, there are fewer variables to manage and training time is cut in half. In pooling layers, the height and weight of each feature map are lowered independently. Two-by-two windows are used for pooling, and three-by-three windows are used for convolution.

3.4.5 Hyper parameters

In total, there are four hyper-parameters to choose from. The 3 x 3 filters are the most commonly used, however 5 x 5 and 7 x 7 are also utilized depending on the application. The depth of the filter at a specific layer is equal to the depth of the filter in the input. The power of this parameter ranges from 32 to 1024, making it primarily a variable. The powerful model's output is filtered using these filters. Start with a minimal number of filters in the initial layers and gradually increase the number in the network. The default value for stride is one. Padding: In the past, padding was commonly applied.

3.4.6 Fully connected

After the pooling layers, add the entirely connected layer. There are no gaps in the network. It is clear that the more vectors that a layer can handle, the more 3D volume it will produce.

3.4.7 Batch Normalization

The batch normalization layer allows each layer to learn more individually. Learning becomes more rapid when batch normalization is utilized it can also be employed as a regularization to prevent model over-fitting. To standardize the outputs/ inputs, the layer is added to the model. Batch normalization is a normalization operation, such as the min–max data normalization performed in data cleaning. Batch normalization is a normalization in which a batch of input data is normalized. It normalizes data, where the data transformation has taken place, such as a mean output close to 0, and the standard deviation output remains close to 1.

3.5 Architecture of Region-based Convolutional Neural Networks (R-CNN)

The R-CNN algorithm was introduced in 2016, with the basic premise of starting with a non-depth approach. The detection object is first and foremost a region chosen from a given image. The R-CNN network then extracts the attributes and position of the visual object from the input image, filtering and merging these regions. The training is a multistage process that includes altering the object region's CNN, adapting SVM to the convolutional network's function, and lastly learning the boundary box regression. First, the image is segmented into regions using the non-depth approach. If the sliding window approach is used for object detection, scanning windows to be the same size and aspect ratio. As a result, a method selective search has been made possible. If the sliding window method is utilized for target detection and recognition,

the scanning window will be the same size and aspect ratio. A selective search approach is adapted in this way. This approach starts by eliminating areas that are unlikely to be part of the target object. This is a lengthy procedure. R-CNN, for example, is extremely sluggish, taking anywhere from 10 to 45 seconds to process each image.

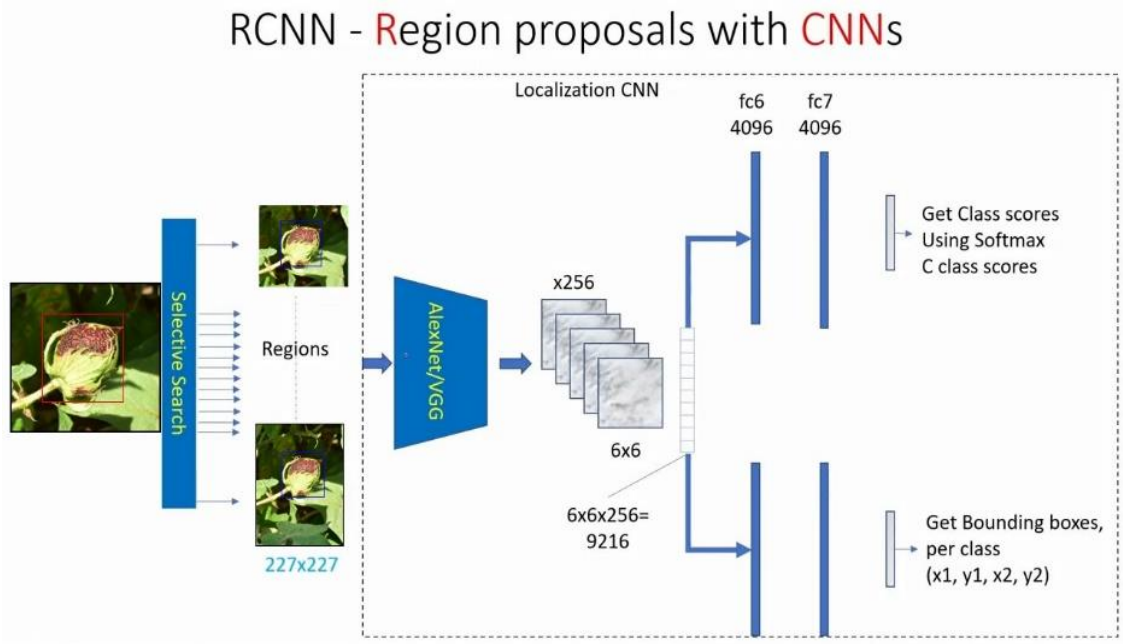


Figure 3.5: Architecture of RCNN

R-CNN has significant progress in the area of computer vision. However, have some drawbacks, such as low efficiency, a long processing time, and a slew of issues. As a result, R-CNN applications do not have a broad variety of applications. Furthermore, R-CNN must extract visual information for several candidate regions ahead of time. This procedure can consume a significant amount of disc space.

Traditional CNN requires a fixed-size input map; however, distortion of images during the normalization process causes the image's size to fluctuate, which is deadly to CNN feature extraction. For calculation, each area proposal must be entered into the CNN network. As a result, the same feature extraction is repeated many times. RNN is typically applied to time series data. It is utilized in agriculture for land classification, phenotypic recognition, and weather prediction, as well as for determining soil moisture levels and predicting total crop production. Agriculture's phenotyping is a relatively new issue. It is the recognition of a plant's

kind based on its look and characteristics. In conclusion, deep learning algorithms have aided in many aspects of life, particularly agriculture.

3.6 General Architecture

The general architecture of Deep Learning model training and testing comprises of dataset images with xml files. The xml data was changed into a csv file. TensorFlow takes the TF records to input into the network during training Deep Learning models, hence TF-records were produced from the csv files. The Deep Learning detectors were built by combining training photos with bounding boxes and then evaluating their performance on a testing dataset.

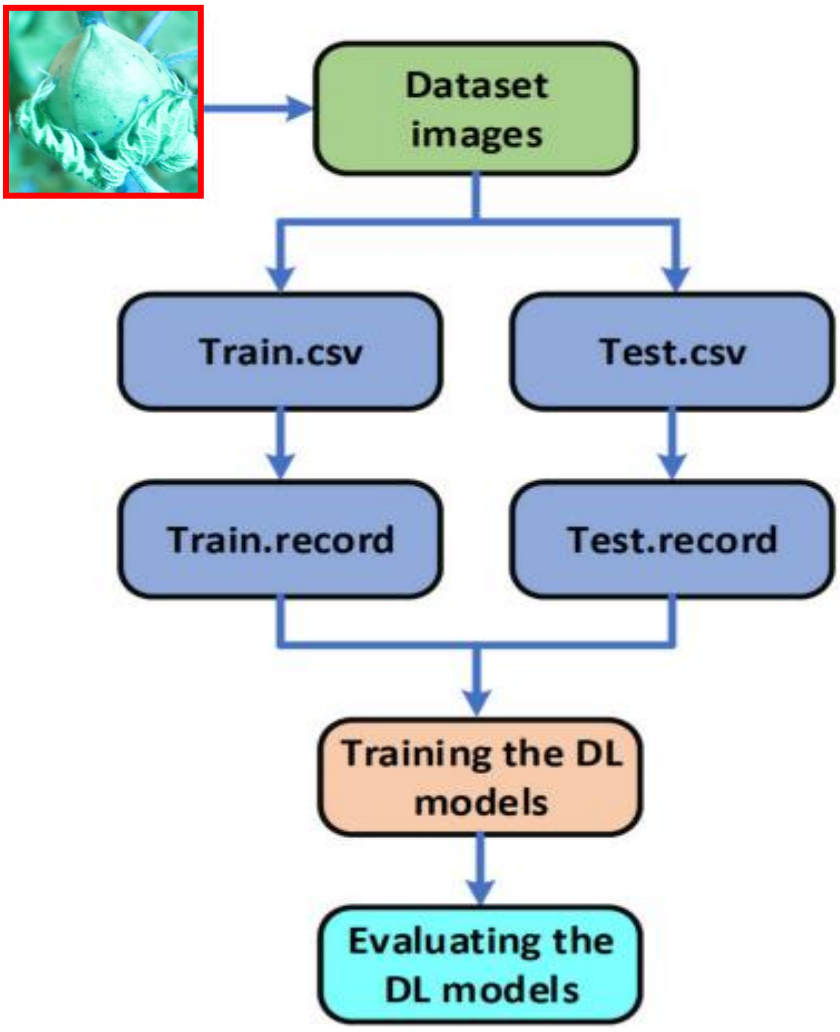


Figure 3.6: General Architecture

3.7 Methodology

Deep learning is latest and most thoroughly researched technology these days. Computer vision is used for numerous applications in image processing, including object detection. The humans have been involved in agriculture for thousands of years, though it is an ancient and significant activity. In recent years, the increase in population has increased the demand for farming products. Automation is being introduced into agriculture to meet this demand without destroying the environment. The purpose of this is to combine deep learning and agriculture.

This research work aims to develop a detection model that can detect boll rot disease in cotton. The proposed method is intended to deal with the field circumstances where boll rot disease is identified. With this automated system, we can find out how much our crop has been destroyed. It will help us make decisions about precise sprays to mitigate their harmful effects on the crop and the environment.

As shown in Figure 3.3**,** the approach is divided into six steps; first step is to acquire a dataset from the field where both classes affected and non-affected cotton bolls. Second, the step is to pre-process these images. The third step is to split dataset into training, validation, and testing with 70%, 20%, and 10% respectively. As a fourth step, the dataset is labeled with its class and validated by agrarians. The fifth step involves converting annotations files into comma-separated values (CSV) files. In the sixth step, the model is trained on the SSD MobileNet-V2 and Faster R-CNN Inception-V2 architectures.

3.7.1 Image Acquisition

The first step is to gather the images needed for further processing. .bmp, .jpg, .png, and.gif are acceptable formats for image data input. On September 17 and 20, 2021, cotton boll images were taken at three sites on the main campus of the University of Agriculture Faisalabad from 9:00 a.m. to 1:30 p.m. in order to increase data diversity. It used a DSLR camera with a JPEG image format and a resolution of 6061*4016 pixels to capture the images were taken from various angles and at a distance of 10–30 cm from each cotton bolls. Over 487 images of cotton bolls were gathered for this project.

3.7.2 Image Preprocessing

In order to improve the image data, several pre-processing techniques such as image cropping and smoothing are used, as well as image enhancement and contrast enhancement to increase

the images' contrast and color conversion. We reduced the resolution of the image to 640*640 pixels in order to lessen their computational burden during training.

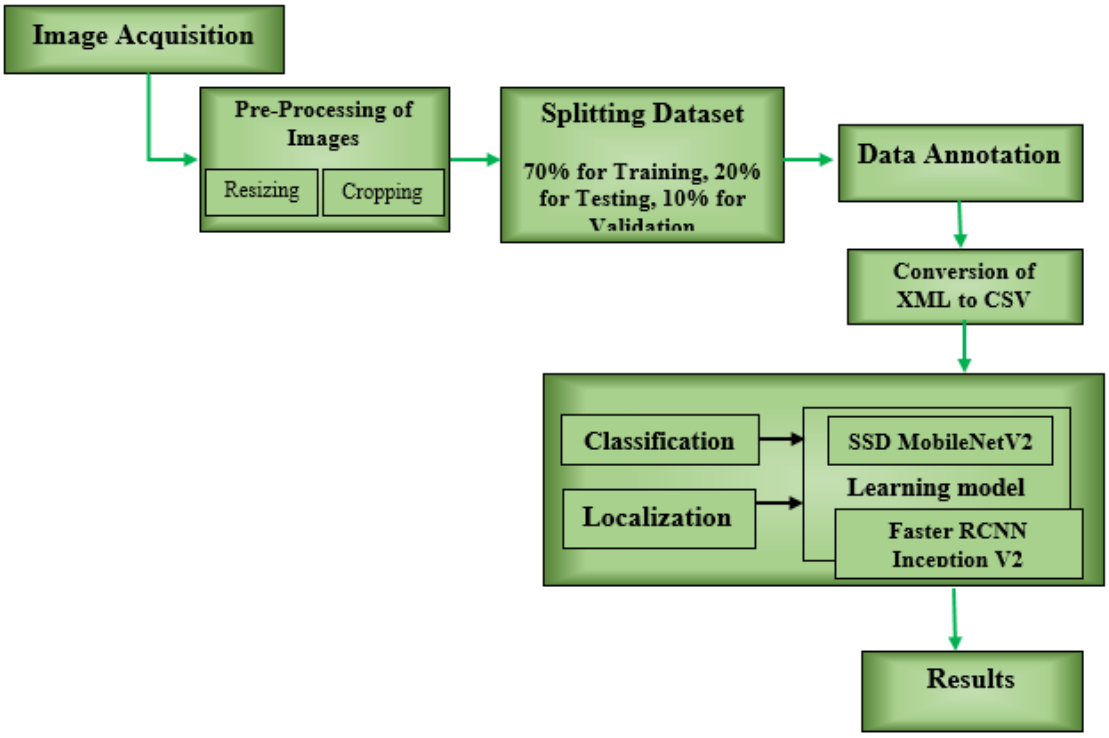


Figure 3.7: Flow diagram of Methodology

3.7.3 Image Cropping and Resizing

The rotted cotton bolls recognized by the detection algorithm are cropped in this phase to get a wider and sharper view of the bolls. It is the removal of unnecessary parts of an image or illustration. In most cases, the process entails removing a piece of the image's periphery in order to clear the image of unwanted elements, improve the surrounding environment, alter the perspective proportion, or draw attention to or distance the subject from its surroundings. The size of the photos changes after a cropping operation is performed on the frames. As a result, cropped images are resized to a standard resolution, in this case 640x640 pixels, in order to ensure consistency. Information numbers displaying different shades of red, green, and blue in a specified location on a pixel framework are all a digital image is. Pixels on a computer screen are most commonly seen as squares that are smaller than normal and wedged together. The data can be shown in a variety of ways with only a little ingenuity and some low-level management of pixels with code. The processing time of a frame is determined on its size. As

a result, resizing is critical to speeding up the process. Image properties should be preserved through the use of more effective scaling procedures. As a result, improving the classification accuracy is as simple as fine-tuning the selected features.

3.7.4 Dataset Augmentation

In order to properly train a neural network, a significant amount of labelled training data is required. We used CNN-based data augmentation to increase the model's generalization ability and avoid over fitting. The training set has a class imbalance problem prior to augmentation, where the three categories account for 70% of training, 20% of testing, and 10% of validation respectively. Poor performance is frequently the result of an unbalanced class. Synthetic images, particularly for underrepresented categories, can be generated using the RCNN trained on the original training data set.

3.7.5 Image Labeling

Data labelling is the initial step in creating a supervised DCNN model, as each image in the dataset must be manually labelled to specify its category. Additionally, we used LabelImg, a program that allows you to put a box around an image and label it as a cotton bolls. A typical cotton boll object is represented as a blue rectangle enclosed within a bounding box. The coordinates of the bounding box and the cotton boll's category are stored in a file accompanying each image. Both the train and the test folder photos had the desired objects identified. Label Image is a tool for identifying and categorizing images. Infected and uninfected images were separated out. LabelImg generated the xml files that include the picture label information.

3.7.6 Transfer Learning

It is very rare to have sufficient amount of dataset to train an entire CNN network from scratch. Deep Neural Network has a large number of parameters, training on a small datasets affects the ability of the network to generalize which results in the over fitting of the network. Transfer learning is an approach where in a pre-trained ConvNet model is used either as a fixed feature extractor or is fine tuned for similar problem domain. If ConvNet is used as a fixed feature extractor fully connected last layer is removed and rest of the network is treated as an extractor

for the new dataset. In another strategy replace, retrain the classifier on the top of the ConvNet and also fine-tune the weights of the pre trained network by back propagating. When training a new model on a new dataset, a previously trained model is used as a starting point for the new model in a process known as "transfer learning". COCO object detection dataset is used for our task and contains 330K images, 1.5 million object instances, and 80 item categories. Our task is based on the trained DCNN backbone network.

After extensive training on a huge dataset, the backbone network has honed its ability to extract general object properties including color, shape, texture, and edge.

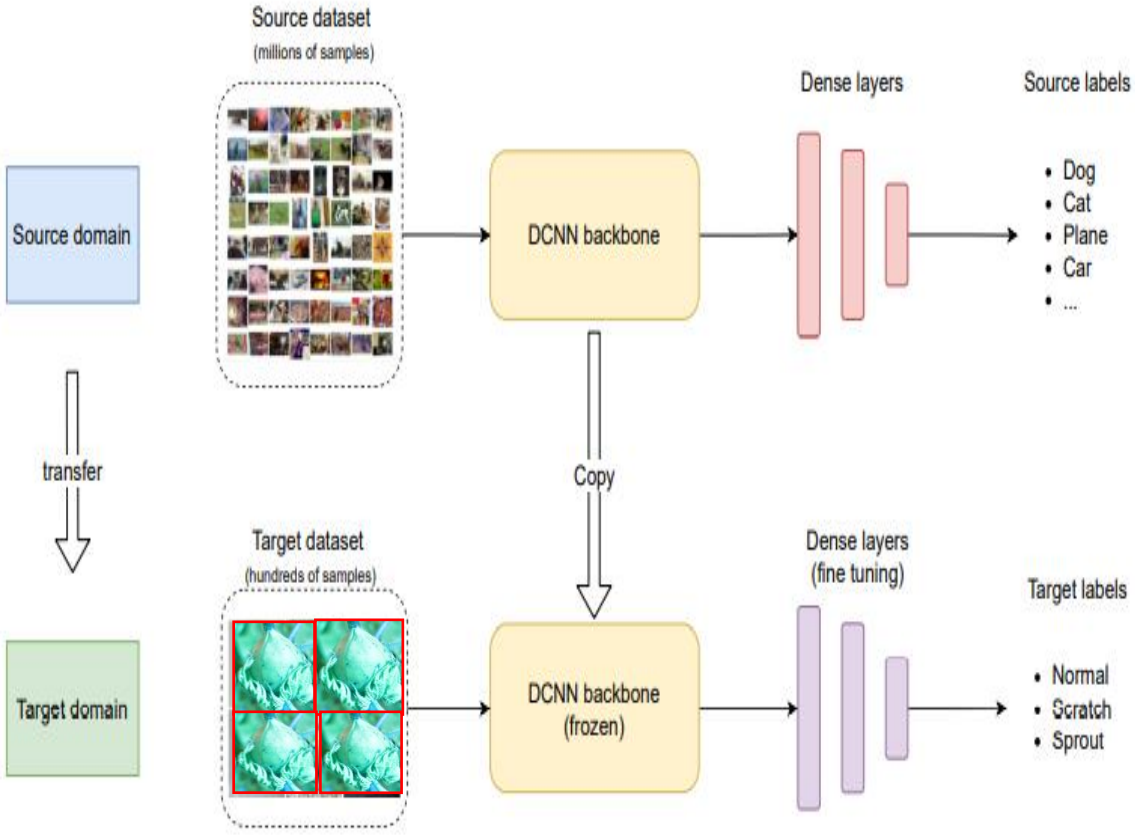


Figure 3.8: Transfer Learning

3.7.7 Generate Training Data

Dataset were generated with the help of .xml files through commands. Dataset were in the form of .csv files carrying information of both folder (test and train) images. Then

generate_tfrecord.py file was opened and replace the already existing label map with a new label map.

3.8 Create Label Map and Configure Training

3.8.1 Label map

Label map instructs the trainer to define every object by defining a map of class id numbers and names. Text editor was used to generate a labelmap.pbtxt file in a training folder and write down a label map in .pbtxt file.

3.8.2 Configure training

To configure a training some changes were made in the Faster R-CNN- Inception-V2_Pets.config and SSD-MobileNet-V2. After the changes, file was saved and configuration was completed.

3.8.3 Feature Extraction

Feature extraction is the process of extracting some of the important features of the defected cotton bolls which are used in training of neural network. This method is done by transforming the subset of observed vector to the observation vector. The observation is processed in such a way so that it could preserve the important information for further processing. The processed information is the mixture of red, green, blue and hue with components of defected part of the cotton bolls image. This comes as a result of the artificial neural network back propagation.

3.8.4 Tuning and Hyper parameter

We used stochastic gradient descent with momentum for Faster R-CNN and SSD since the image sizes employed were different and therefore they were trained separately. We selected 4 as the batch size for Faster R-CNN and SSD because the images were shrunk to a set form. The learning rates for each feature extractor were individually adjusted to get the best results. The networks were trained using the Google Collaboratory Pro dataset, which had 487 images for validation, which we used to train the networks. The model was assessed using the official TENSORFLOW API, which measures the mean area under the curve (mAP).

3.8.5 System Configuration

Deep learning algorithms need the use of powerful GPUs and Compute Unified Device Architecture. A GPU NVIDIA GeForce 740M (version 417.35) with 32GB Random Access Memory was used to complete this work, which was programmed in the Python programming language and the Tensorflow library using the Google Collaboratory.

3.9 Model 1: Faster R-CNN with Inception-V2

3.9.1 Fast R-CNN

It is the improved version of RCNN. The Fast R-CNN detector also uses an algorithm like Edge Boxes to generate region proposals. Unlike the R-CNN detector, which crops and resizes region proposals, the Fast R-CNN detector processes the entire image. Whereas an R-CNN detector must classify each region, Fast R-CNN pools CNN features corresponding to each region proposal. Fast R-CNN is more efficient than RCNN, because in the Fast R-CNN detector, the computations for overlapping regions are shared. R-CNN works really well, but is really quite slow for a few simple reason. It requires a forward pass of the CNN for every single region proposal for every single image. It has to train three different models separately the CNN to generate image features, the classifier that predicts the class, and the regression model to tighten the bounding boxes. This makes the pipeline extremely hard to train. Both these problems were solved in Fast R-CNN by the creator of R-CNN himself.

For the forward pass of the CNN, Girshick realized that for each image, a lot of proposed regions for the image invariably overlapped causing us to run the same CNN computation again and again. His insight was simple. Why not run the CNN just once per image and then find a way to share that computation across the proposals? This is exactly what Fast R-CNN does using a technique known as RoI Pool (Region of Interest Pooling). At its core, RoI Pool shares the forward pass of a CNN for an image across its sub regions. In the above, notice how the CNN features for each region are obtained by selecting a corresponding region from the CNN feature map. Then, the features in each region are pooled (usually using max pooling). So all it takes us is one pass of the original image.

The second insight of Fast RCNN is to jointly train the CNN, classifier, and bounding box regressor in a single model. Where earlier we had different models to extract image features (CNN), classify (SVM), and tighten bounding boxes (regressor), Fast RCNN instead used a

single network to compute all three. Fast R-CNN replaced the SVM classifier with a softmax layer on top of the CNN to output a classification. It also added a linear regression layer parallel to the softmax layer to output bounding box coordinates. In this way, all the outputs needed came from one single network. A total of two output layers were provided: one for the classification scores and another for the bounding box location offset, which was represented as four values: the centroid of the bounding box's x and y parameters, as well as the bounding box's width and height (Arsenovic *et al.*, 2019).

3.9.2 Faster R-CNN

Faster R-CNN was a deep learning algorithm developed from CNN that could be used in object detection systems. The object detection system was a system that has a function to localize objects in the image, so the classification process would get better results. Faster R-CNN was the development of Fast R-CNN. Fast R-CNN was an object detection method that used the selective search method in the region proposal search process. The region Proposal module task was to find regions or areas that may contain objects in it.

Shaoqing Ren, in his research on the implementation of Faster R-CNN as Real-Time Object Detection, revealed that this method generally consists of two modules, namely Region Proposal Network (RPN) and Fast R-CNN. The input that was entered into the system will be processed in the Convolutional Network first to get the feature of the object in the image, named Feature Maps. Then Feature Maps from the Convolutional Network will be forwarded to the Region Proposal Network (RPN) module and the Fast R-CNN module. The Fast R-CNN module function was refining the region proposals of the RPN and classifying the objects in it. Largely Regions with a high likelihood of occurrence of an item are created by RPN in the first stage of the quicker R-CNN, which is then used to build regional anchors. In the next phase, we will utilize the identified areas to categorize the object (a pedestrian) and extract the bounding box information from the object. Convolutional feature maps (CFMs) are produced by feeding a batch of pictures (of any size) through a convolutional neural network that has been previously trained (CNN).

The Faster R-CNN model is built on the Inception v2 model, which was developed by Google. Inception v2 is the sequel to the original. When it comes to Inception v2, the 5x5 convolution

layer has been replaced by two 3x3 layers, making it a more refined version of its predecessor, Inception v1. Finally, the 3x3 convolutions are subdivided into 1x3 and 3x1 convolutions. A candidate window, or area of interest, is created by the RPN utilizing convolutional feature maps (extracted from the Inception v2 backbone) and is shown on the screen (ROI).

It is necessary to perform ROI pooling on convolutional features maps for each of the suggested areas in order to provide a fixed-length feature vector for the classification network in order to build a classification network. The fixed-length feature vectors from each ROI are put into two fully connected layers which are then fed into the pooling ROI layer. As indicated in the illustration, one of the output layers generates estimate softmax probability on object classes and background color using one of the output layers (classification). The revised bounding box locations for each item class are encoded in the second layer, which uses four real-valued number outputs (x, y, and w, h) to do so (regression). When estimating regression and classification losses, the loss function L is used (Alamsyah and Fachrurrozi, 2019).

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{\text{reg}}}\sum_i p_i^* L_{\text{reg}}(t_i, t_i^*)$$

- In a small batch, I is the index of an anchor box inside the batch.
- Pi is the chance that an item will be present in the ith index anchor box predicted by the model for the given ith index.
- In the Label, P*i denotes an item that is present in the ith index anchor box stated in the Label (1 denotes a positive anchor, 0 denotes a negative anchor).
- Ti is the anticipated bounding box for the item that is now present (only for positive anchor and do not care for negative anchor).
- T*i denotes the ground truth of the bounding box for the ith positive anchor in the set.
- Lclassification is the loss in classification for an item that is present or not.
- Lregression is a regression loss for the purpose of bounds.
- Pi is the result of the categorization layers.
- Ti represents the output of the regression layers.

Object detection, as mentioned in the previous section, has two objectives: classification and localization of the detected objects. 1/N courses are available. * It is the loss function for

classification that deals with whether or not an item is present. It is defined as L calcification (pi, p*i). 1/N regression is a kind of regression in which one number is substituted for another number. * The regression loss that is the topic of localization is denoted by the symbol L regression (t, t*i). In order to normalize and balance the trade-off between the two losses, the weights of 1/Nclases and 1/N regression are multiplied by the balancing parameter in the balancing equation. When normalized by the number of anchor sites (i.e., N regression 2, 400), the number of classes equals the size of the mini batch in this example (N regression 2, 400). Only the positive anchor boxes in the dataset are used to compute the regression losses in the model.

$$t_x = \frac{(x - x_a)}{w_a} \qquad t_y = \frac{(y - y_a)}{h_a} \qquad t_w = \log\left(\frac{w}{w_a}\right) \qquad t_h = \log\left(\frac{h}{h_a}\right)$$

$$t_x^* = \frac{(x^* - x_a)}{w_a} \qquad t_y^* = \frac{(y^* - y_a)}{t_w} \qquad t_w^* = \log\left(\frac{w^*}{w_a}\right) \qquad t_h^* = \log\left(\frac{h^*}{h_a}\right)$$

Here, the forecast box, proposal box, and ground-truth box are denoted by the letters (x, y, w, h), (xa, ya, WA, ha), and (x*, y*, w*, h*) respectively.

The R-CNN model, which is faster than SSD, calculates regression and classification loss in a manner similar to SSD. For the purpose of calculating the loss function, two losses are used: confidence loss and expectation loss. Lconf, which is based on an individual's degree of trust, and Lloc, which is based on the network's capacity to predict bounding boxes, are two of the most important metrics. Both losses are taken into account and added together for the total loss (Liu and Wang, 2020).

$$L(x, c, l, g) = \frac{1}{N}\left(L_{\text{conf}}(x, c) + \alpha L_{\text{loc}}(x, l, g)\right)$$

In this equation, N is the number of default boxes, L is the localization loss, is a weight to balance two losses that is set to 1 in all trials, c is the offset for the center, l is the predicted box parameters, and g is the ground truth box parameters.

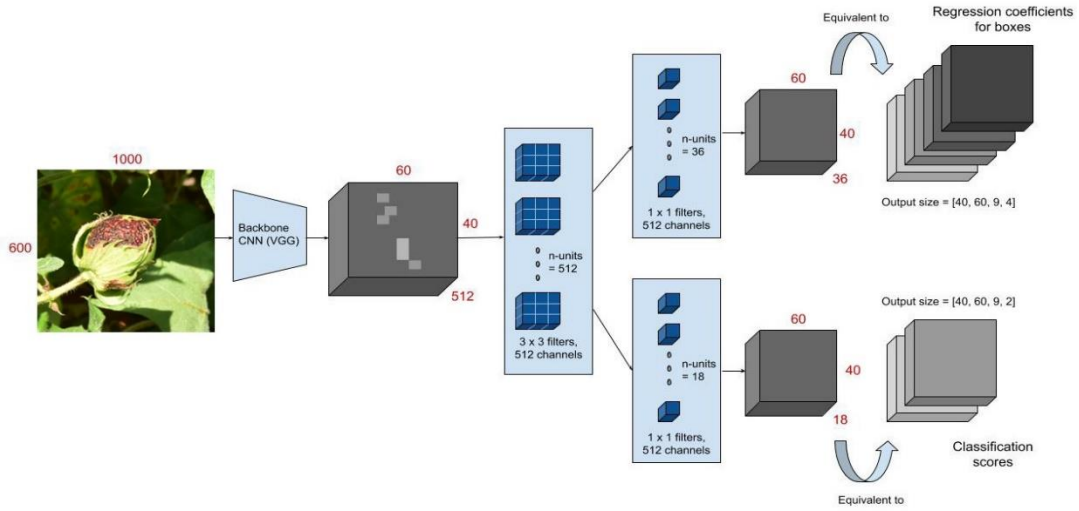


Figure 3.9: Architecture of Faster RCNN

When training the Inception model, there are millions of parameters that must be calibrated against the ImageNet dataset using incredibly powerful and expensive computers; this process may take weeks to complete. Previously, there was a search engine network known as GoogLeNet that existed before Inception-v1. After the architecture was updated with the help of the batch normalization approach and other ways, Inception-v2 was released to the public. On the basis of the factorization technique, the convolution of 7*7 has been divided into three convolutions of 3*3 each. Through the use of the grid reduction approach, the 35x35 grid with 288 filters on each side is reduced to a 17x17 grid with 768 filters on each side, which represents the network's inception component. The factorized inception modules are then employed a total of five times throughout the programme. It is reduced to an 8 × 8 x 1280 grid using the grid reduction technique. There are two Inception modules in the coarsest 8 × 8 level, and each tile has a 2048-tile output filter bank that has been concatenated. Instead of removing the bottleneck filters in the deepening module, it is preferable to remove them in the broadening module.

3.9.3 Inception-V2

In 2015, the GoogLeNet architecture was introduced as the initial version of the Inception architecture. Inception-V2 was launched after the architecture was modified with the addition of batch normalization Inception-V2 is the detection and classification standard for the ILSVRC2014 competition, and it is the first of its kind. With the help of Inception modules, it is feasible to expand the breadth and depth of a network without raising the overall

computational cost of the project. Rather of segmenting Inception V2 into regions, SDD when implemented as a conceptual model extracts features from the full network model as the major feature extractor. Supplemental convolutional feature maps of varying sizes are needed to finish the investigation. Both a high-resolution lower-level feature map and a convolutional feature map at the top of the hierarchy are selected. There is a two-fold reduction in spatial resolution with each successive layer.

The layers Mixed 4c and mixed 5c in Inception V2 generate multi-resolution feature maps at various pixel densities**.** Inception V2 is a module designed to reduce the complexity of convolutional networks. This module makes the convolution network wider and deeper, replacing the 5x5 convolution with the 3x3 convolution. It also follows the principle that says spatial aggregation can be done through the insertion of lower dimensions without much or loss of representational power. By carrying out 3x3 convolution, the convolution performance is improved (Sujatha *et al*., 2021)**.**

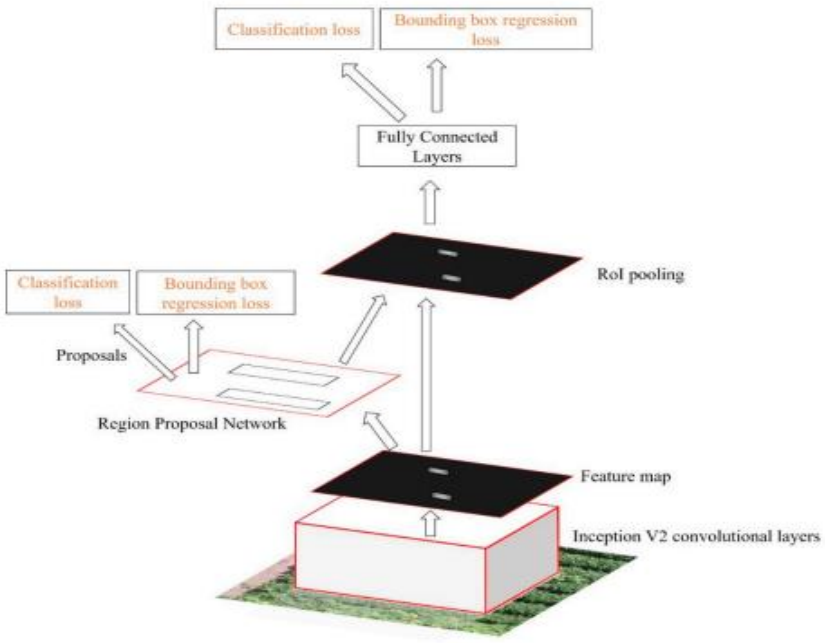


Figure 3.10: Architecture of Inception V2

3.10 Model 2: SSD-MobileNet V-2

3.10.1 Single Shot Multibox Detector (SSD)

SSD was a single-shot detector that did not have a stage for region proposal, making it a single-shot detector. Instead of using just the final layer for detection, SSD employed numerous layers to better capture multi-scale objects, as opposed to Faster R-CNN, which only used the last layer for detection. In order to accommodate the anchors being deployed in numerous feature maps, SSD created several anchor spectrum variations among layers. Scales were discovered in the lower layers to be quite little, whilst bigger scales were found in the top layers to be rather huge. As a result, this architecture is capable of processing a wide variety of items and has a greater recall rate. SSD employed completely convolutional layers to predict scoring and segmentation to aid mitigate, in contrast to YOLO, which used fully connected layers for object recognition and identification of objects. SSD achieved the status of accomplishment on a number of benchmark datasets with the use of certain data augmentation and hard negative mining algorithms, among other things.

SSD on the other hand, struggled with little objects since it was built with shallow layers that lacked deep semantic information. SSD was a single-shot detector for multiple class objects that was faster than YOLO. The SSD method was based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression step to produce the final detections. SSD only needs an input image and ground truth boxes for each object during training. SSD object detection method was designed to create a deep learning object detection method with a lighter process than other object detection methods based on deep learning processes like YOLO and Faster R-CNN.

The proposed SSD MobileNet-V2 methodology has been clearly explained using the two algorithms. First the images were pre-processed and were trained on the whole dataset. Second the model trained in the first part was used to detect the cotton boll rot with the appropriate accuracy. In Algorithm images along with their pixel values were taken as an input, resized and normalized. Data augmentation technique was then applied on the images in order to get more accuracy. Data was then splitted into training and testing batches and MobilenetV2 model was applied on it. SGD with momentum optimizer was used to compile the whole model. The model trained in previous part was then deployed on both classification on static images and on real time. If the cotton bolls are detected using SSD, a bounding box showing the cotton bolls are affected or non-affected is shown in the output.

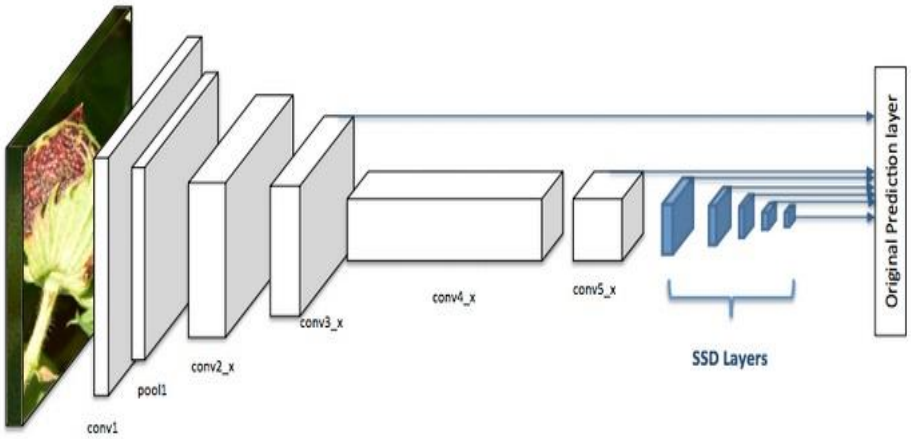


Figure 3.11: Architecture of SSD

With SSD, a single forward network pass may do both object localization and categorization. A collection of auxiliary convolutional layers is added to SSD's VGG-16 backbone network, allowing feature extraction at many scales while decreasing the input size with each successive layer. Using multi-scale feature mapping, SSD mimics the human eye's ability to see. As a starting point for classification and regression, the technique employs a collection of multi-level feature maps. The goal of this technique is to reliably detect different scales of object properties. To put it another way: The high-level and low-level receptive fields are vastly different in terms of their size. Features maps of different scales in SSD are represented by default boxes of varying sizes and aspect ratios. The basic box technique is used by SSD, although the feature units on the same feature layer might have various alternative aspect ratios. MobileNet is a type of neural network that is both tiny and quick.

The initial version of 13 convolution layers, an average pooling layer, and a fully connected layer make up Mobile Net's architecture. Each convolutional layer is followed by a BN layer, which is then activated with a ReLU. Models can trade off latency or accuracy for speed and compact size using two hyper-parameters, the width multiplier and the resolution multiplier. The number of feature maps in the model is reduced, while the size of feature maps is reduced, which reduces the computational burden. In order to reduce computation time, a deep separable convolution (Depth wise Separable Convolution) can be used to the traditional kernel. Conventional convolutions can be converted into depth-wise separable ones by first creating a depth-wise separable (Point wise convolution). Using the depth-wise convolution approach, the resultant outputs are blended using point-wise convolutions to get the desired effect while spatially convolving each input channel.

There are two new features in MobileNet V2: a linear bottleneck connecting layers and a shortcut link across obstacles. Using a linear bottleneck as a learning tool, a deeper layer can learn how to turn pixel level concepts into higher-level descriptors. Shortcuts to a deep CNN can be used to speed up the learning process of a neural network. The SSD MobileNet V2 reduces the computing burden by replacing SSD's VGG-16 backbone with MobileNet V2 and hence is more energy-efficient in this instance. MobileNet V2 is used as a feature extractor in the SSD-MobileNet V2 neural architecture, followed by additional layers for multi-scale learning and a feature extraction layer.

Even though it is just a single-stage object detector, it is used to minimize the size and complexity of the model. The model makes use of a single network with a large number of feature maps. Because of this information, the network may utilize it to increase speed while simultaneously decreasing suggested zones by anticipating both very big things via deeper layers and extremely tiny targets through shallow layers. MobileNet's backbone is made up of two convolutions: a 33-depth wise convolution and an 11-point wise convolution, both of which are rather straightforward. SSD-MobileNet is a handy tool when it comes to working with mobile and embedded devices (Liu *et al*., 2021).

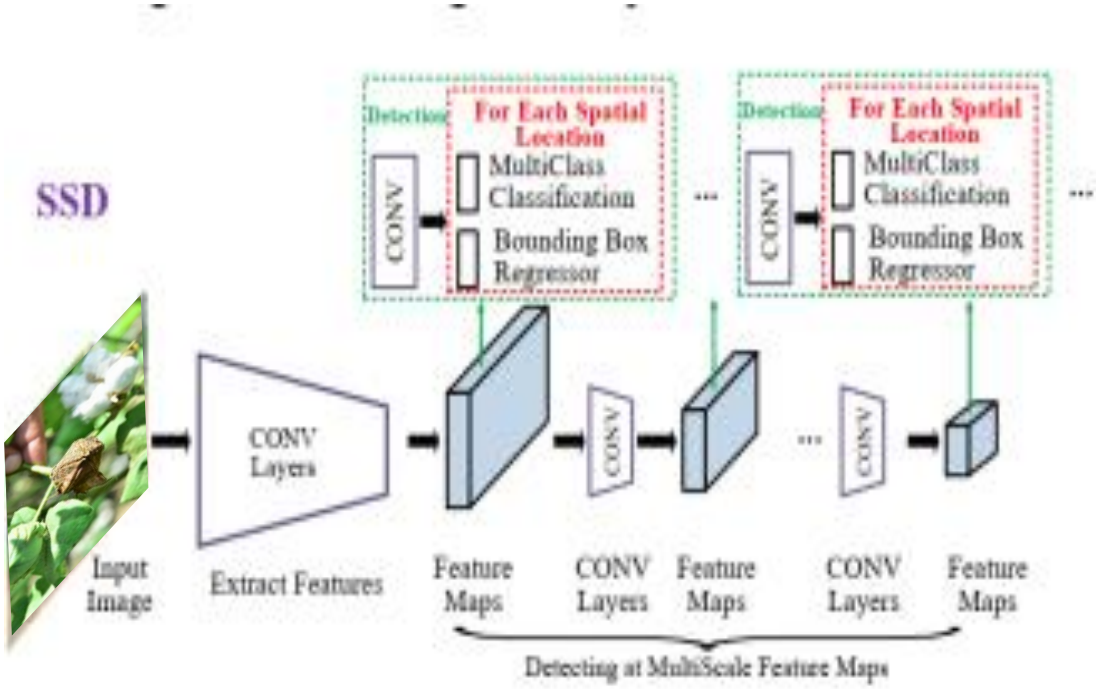


Figure 3.12: Architecture of SSD with MobileNet V2

The characteristics of numerous convolutional layers in the SSD system model may be used to produce classification regression and bounding box regression, which are both useful in machine learning applications. They are referred to as "standard boxes" in the industry. Afterwards, these default boxes are scaled for each feature map and may be determined using the formula:

$$S_k = S_{min} + \frac{S_{max} - S_{min}}{m - 1}(k - 1), (k\epsilon[1, m])$$

Where m is the number of feature maps and Smin, Smax are parameters that can be set.

Single Shot Multibox Detector (SSD) network requires a full image as an input, combined with a label of the bounding box as a ground truth for each visual object. Some of the convolutions in the feature maps with multiple sizes are confirmed by a small number of default bounding boxes for each object. The visual object's shape offset and confidence 28 probability are projected for each default box. CNN is a deep neural network that creates a sequence of fixed-size border boxes and scores for each box. As a result, the final prediction is obtained using a non-maximum suppression procedure. A major responsibility of the MobileNet layers is converting pixels in an input picture into features that characterize the image's contents and transmitting these information to the other layers. As a consequence, in this example, MobileNet is utilized as a feature extractor for a second neural network that is trained on the same data. In order to overcome the difficulty of object detection compared to classification, SSD adds many extra convolution layer to the underlying network (Iyer *et al.*, 2021).

3.10.2 MobileNet V-2

MobileNet-V2 is an enhanced version of MobileNet-V1 it is consists of 54 layers and has an image input size of 224×224×3. Its key feature is that it uses two 1D convolutions with 2 kernels rather than having a single kernel 2D convolution. This implies that less storage, parameters, small and efficient model are needed for training. Two types of blocks can be distinguished: the first is a residual block with a stride of 1 and the other is blocked with a stride of 2 for downscaling. There are layers for each block: the first layer is ReLU6 1×1 convolution, the second layer is the deep convolution and another 1×1 convolution is the third layer but deprived of any non-linearity.

MobileNetV2 is a Deep Neural Network that has been deployed for the classification problem. Pre trained weights of ImageNet were loaded from TensorFlow. Then the base layers are frozen to avoid impairment of already learned features. Then new trainable layers are added, and these layers are trained on the collected dataset so that it can determine the features to classify a face wearing a mask from a face not wearing a mask. Then the model is fine-tuned, and then the weights are saved. Using pre-trained models helps avoid unnecessary computational costs and helps in taking advantage of already biased weights without losing already learned features.
MobileNetV2 is a deep learning model based on Convolutional Neural Network, which uses the following layers and functions.

3.10.2.1 Convolutional Layer

This layer is the fundamental block of the Convolutional Neural Network. The term convolution implies a mathematical combination of two functions to get the third function. It works on a sliding window mechanism, which helps in extracting features from an image. This helps in generation feature maps.

3.10.2.2 Pooling Layer

Applying the pooling operations can make calculations go faster by allowing a reduction in the size of the input matrix without losing m any features. Different kind of pooling operations can be applied out of which some are explained below:

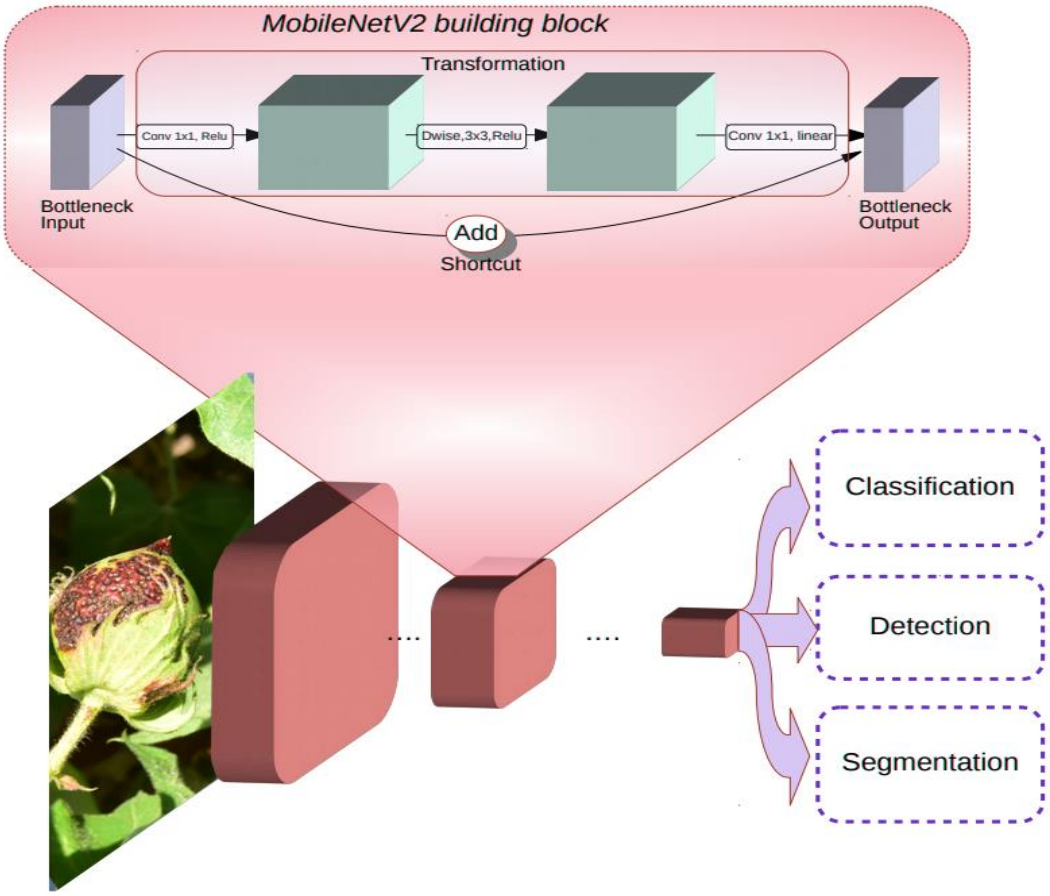


Figure 3.13: Architecture of MobileNet V2 building block

3.10.2.3 Max Pooling

It takes the maximum value present in the selected region where the kernel is currently at as the value for the output of matrix value for that cell.

3.10.2.4 Average Pooling

It takes the average of all the values that are currently in the region where the kernel is at and takes this value as the output for the matrix value of that cell.

3.10.2.5 Dropout Layer

This helps reduce the over fitting, which may occur while training by dropping random biased neurons from the model.

These neurons can be a part of hidden layers as well as visible layers. The likelihood for a neuron to be dropped can be changed by changing the dropout ratio.

3.10.2.6 Non-Linear Layer

These layers usually follow the convolutional layers. Most commonly used non-linear functions include different kinds of Rectified Linear Unit (ReLU), Leaky ReLU, Noisy ReLU, Exponential ReLU etc., sigmoid function as well as Tanh functions.

Towards the conclusion of the model, fully-connected layers are introduced, and these levels have complete access to the activation layers. These layers are used to aid in the categorization of pictures in multi-class or binary classification systems. Softmax is an example of an activation function that is employed in these layers, and it is responsible for calculating the chance of the expected output classes being present or absent. Bottlenecks caused by linear activation functions because many matrix multiplications cannot be reduced to a single numerical operation, non-linear activation functions such as ReLU-6 are employed in neural networks to quickly resolve multiple disparities. This technique may be used to construct a multilayer neural network. Because the ReLU activation function ignores values less than zero, in order to counteract information loss, the network's dimensions must be increased in order to accommodate the increased number of channels. Layers of the blocks are compressed to create a reversed residual block, and the technique described above is continued in the other direction. This is done at the point where the skip connections are joined; the performance of the network may be affected as a result of this operation. Consequently, the linear bottleneck was created, in which the last convolution of a leftover block is given a linear output before being added to the original activation.

3.11 Activation Function

Activation functions are functions used in neural networks to computes the weighted sum of input and biases, of which is used to decide if a neuron can be fired or not. It manipulates the presented data through some gradient processing usually gradient descent and afterwards produce an output for the neural network that contains the parameters in the data. Activation function can be either linear or non-linear depending on the function it represents, and are used to control the outputs of out neural networks. Activation functions such as the Sigmoid, Tanh, Softplus, and ReLu. In order to avoid the problems associated with slow convergence speed and gradient dispersion, the trend of the activation function in the neural network model is the unsaturated nonlinear type, such as ReLu, Softplus, and Softsign, and the saturated nonlinear type is the Sigmoid and Tanh. ReLu is the most widely used and has a number of upgrades,

including Relu6, Elu, Leaky Relu, PRect Relu, RRect Relu, and so on, all of which contribute significantly to the improvement in the performance of neural networks.

3.11.1 ReLU-6

In deep learning models, the Rectified Linear Unit is the most widely employed activation function. If the input is negative, the function returns 0, but if the input is positive, it returns the value xx. So it can be written as f(x) =max (0, x).

Graphically it looks like this

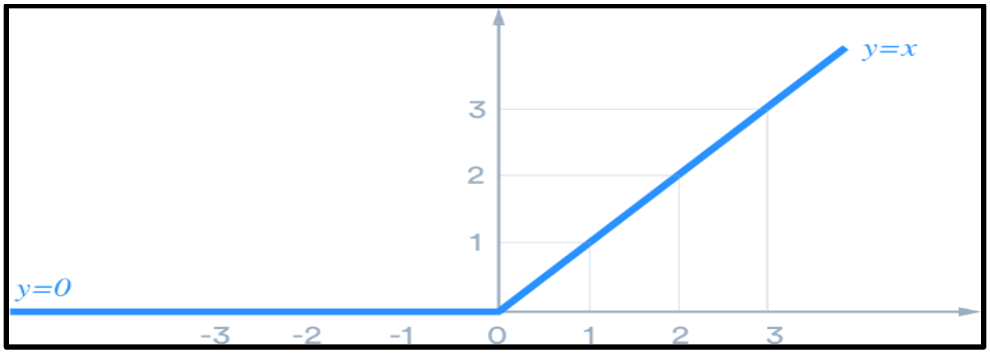


Figure 3.14: Graphical representation of ReLU

ReLU units differ from in two respects. First, we cap the units at 6, so our ReLU activation function is y = min (max(x, 0), 6).

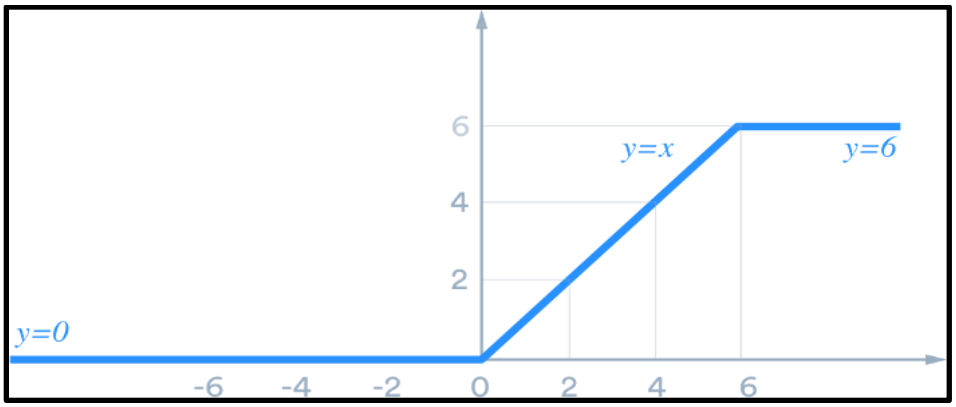
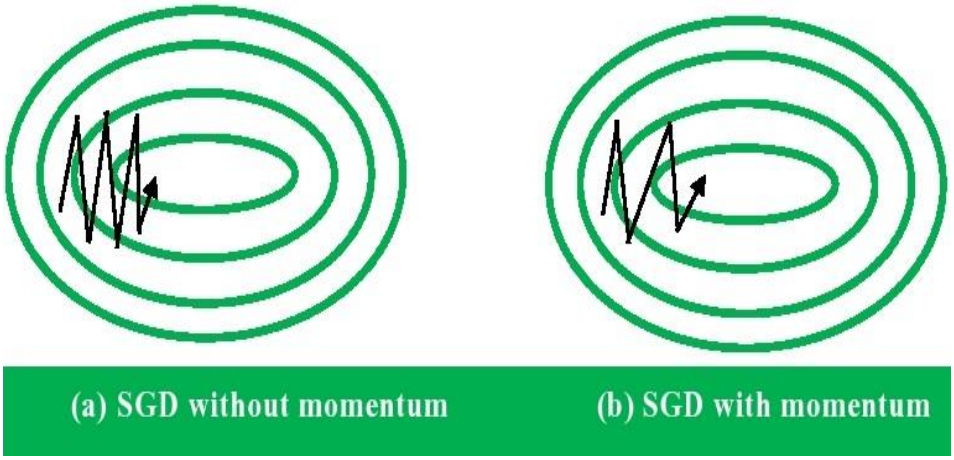


Figure 3.15: Graphical representation of ReLU-6

This makes it easier for the model to pick up on sparse characteristics. Instead of assuming an unlimited number of replicated bias-shifted Bernoulli units, the formulation assumes that each ReLU consists of just six such units.

3.11.2 Stochastic Gradient Descent with Momentum

The most extensively used neural network optimization approach is stochastic gradient descent. Its momentum variant is more efficient than the usual approach at convergent iterations. Stochastic gradient descent slows down gradient descent and avoids a high learning rate by gently oscillating the loss towards the minimum to optimize the cost function. Momentum it is difficult for SGD to go across ravines, which are widespread near local optima and are regions where the terrain bends strongly in one dimension but not the other. As seen in Figure, in these cases, SGD zigzags around the ravine's slopes while only making slow progress towards the local optimum on the bottom.



3.12 Google Collab Pro

Google Collaboratory, or simply Google Collab, was the IDE of choice for the project. Google Collab is a free Jupyter notebook environment that does not need any installation and operates fully in the cloud, making it ideal for collaborative work. Colab is a browser-based tool that allows us to write code, record our findings, and access powerful computational resources all without having to leave our browsers. Google Drive was used to upload a CSV file of the dataset to Google Collab. Accessing Colab First, sign in the Google accounts. And proceed to the Google collab welcome page. Click on the new python three notebook preference to begin the session fresh. Choose runtime menu choice or notebook choice to select GPU. Configure notebook instance, to download the necessary packages. Click on mount drive choice, an authorization code is produced and it is entered in Google drive, which produces the image folder path (Ramacharan, 2021).

3.13 Python 3.7

Programming at a high level is what Python is best known for. Released in 1991, it's a classic. It's useful for large-scale endeavors. Python is a dynamically typed language. It can be used for functional and object-oriented programming. Many operating systems support Python. It is a free software application. The various extensions were supplied via a variety of paradigms. For memory management, Python uses a hybrid of a cycle-detecting garbage collector and a counting reference. Python's extensibility makes it an excellent choice for web development. Giving meaning and style to a program is what Python does. Python is an easy-to-learn and understandable programming language.

3.14 TensorFlow

Tensorflow is an open-source software library. Graphs of data flow are utilized to do numerical calculations. In a graph, each node represents a certain mathematical procedure. Data arrays (also known as tensors), which are multidimensional, are represented by the graph's edges. You may use many GPUs or CPUs in a server, mobile device, or desktop without having to write a single line of code. Tensorflow includes TensorBoard as well. This data visualization toolset is called TensorBoard. It provides C and Python APIs that are well-documented and well-supported. APIs for Java, C++, Swift, Go and JavaScript are not guaranteed to be backwards compatible. For generating Deep Learning networks, Tensorflow object detection API is an essential supporting framework. In "Model Zoo" trained models are present. New models can also be build.

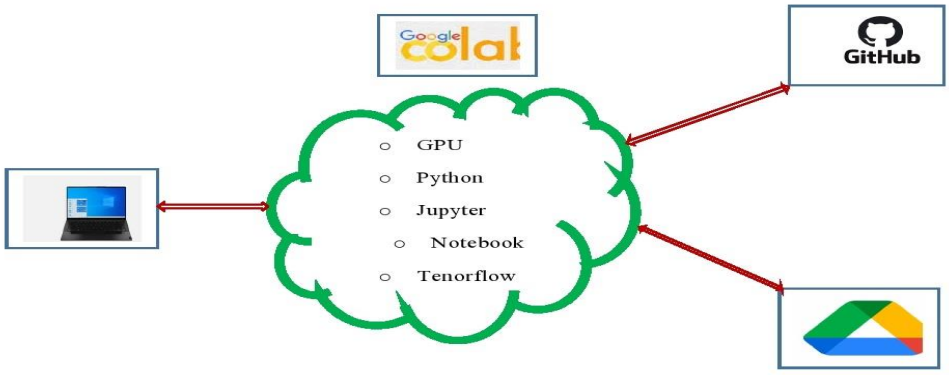


Figure 3.16: Google Collab Environment

CHAPTER 4

RESULTS AND DISCUSSIONS

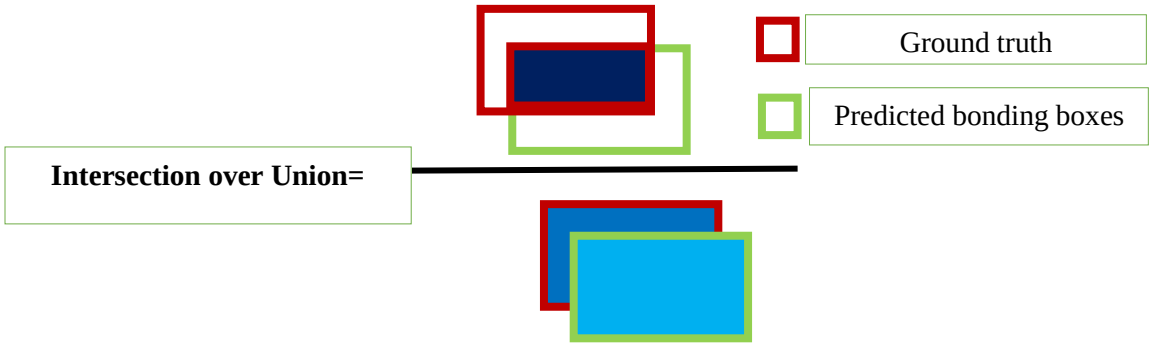
For the identification of cotton boll rot disease, the suggested research made use of augmented data collected data from the field. There were enough images in the data to train the DL models.

4.1 Performance Metrics

The accuracy, precision, recall, and intersection over union of the model were all utilized to create a comprehensive picture of how well it performed. Accuracy is defined as the proportion of accurate predictions made relative to the total number of forecasts. Pre represents the percentage of true positives (TP) among the detected positives, Recall represents the percentage of TP among the correct predictions, specificity represents the percentage of negatives that are incorrectly classified as positives represents the mean of precision and recall, respectively.

4.1.1 Intersection over Union (IoU)

A function called intersection over union can be used to calculate an image's IoU. The following two parameters are accepted: a) Ground-truth bounding box: gt box, b) Bounded box predicted by prediction box. When two boxes in the intersection and union variables intersect, it computes the intersection and union of those two points on the graph. We employ Intersection over Union as a notion for object detection (IOU). Intersection of the two bounding boxes is computed using a Jaccard index-based algorithm, which computes the union of these two bounding boxes. As a result, a perfect overlap would be 1.



The prediction may be stated as either positive or negative when using a threshold value. Let's imagine the IOU threshold is set at 0.5. In that scenario, the following would happen:

• If a IOU is less than 0.5, the item detection is classified as True Positive (TP).

• If a IOU is less than 0.5, the detection is incorrect, and the result is classified as a False Positive (FP).

$$\text{class } (IoU) = \begin{cases} \text{Positive } \rightarrow \ I_0U \geq \text{ Threshold} \\ \text{Negative} \rightarrow I_0U < \text{ Threshold} \end{cases}$$

4.1.2 Accuracy

An overall measure of the model's performance across all classes is called accuracy. When all classes are of similar relevance, it is beneficial to employ it. The ratio of correct guesses to total predictions is used to arrive at this number.

$$\text{Accuracy } = \frac{\text{TP } + \text{TN}}{\text{TP} + \text{FP} + \text{FN}}$$

4.1.3 Precision

The precision is a metric for gauging how accurately a sample is classified as positive by a model.

$$\text{Precision } = \frac{\text{True }_{\text{positive}}}{\text{True }_{\text{positive}} + \text{ False }_{\text{positive}}}$$

When the model produces a large number of inaccurate Positive classifications or a small number of correct Positive classifications, the denominator grows and the precision decreases. On the other hand, accuracy is great when the following conditions are met:

1. The model correctly classifies a large number of positive categories (maximize True Positive).
2. The model makes fewer incorrect Positive classifications (minimize False Positive).

4.1.4 Average Precision (AP)

It is possible to summarize the precision-recall curve into a single number that represents the average of all precisions by using the average precision (AP) method. The AP is determined in accordance with the following equation:

$$AP = \sum_{k=0}^{k=n-1} [\text{Recalls } (k) - \text{Recalls } (k+1)] * \text{Precisions } (k)$$

$$\text{Recalls } (n) = 0, \text{Precisions } (n) = 1$$

$$n = \text{ Number of thresholds.}$$

The difference between the current and next recalls is determined using a loop that iterates over all precisions/recalls, and the result is multiplied by the current precision. In other words,

the average precision (AP) is the weighted sum of precisions at each threshold, where the weight represents the increase in recall over the previous threshold.

4.1.5 Mean Average Precision (mAP)

The mean average precision (mAP) is a performance statistic that is extensively used in the field of object identification and recognition. Calculating the mAP is based on the concept of Intersection over Union (IoU), which is defined as a ratio between the area where two bounding boxes intersect and the area where they join in the anticipated and ground truth bounding boxes. This equation is used to compute the mean annual percentage rate (mAP).

Formula:

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$AP_k =$ the AP of class k

$n =$ the number of classes

4.1.6 Recall

The recall rate is computed as the ratio of the number of Positive samples that were properly categorized as Positive to the total number of Positive samples that were correctly classified as Positive. When a model is recalled, it indicates how well it can recognize positive samples. The greater the recall, the greater the number of positive samples found.

Formula:

$$\text{Recall} = \frac{\text{True}_{\text{positive}}}{\text{True}_{\text{positive}} + \text{False}_{\text{negative}}}$$

The recall is simply concerned with how the positive samples are categorized. This is irrespective of how the negative samples are categorized, for example, whether they are categorized for accuracy or not. When the model mistakenly labels all of the positive samples as Positive, the recall will be 100 percent, even if all of the negative samples were incorrectly categorized as Positive by the model.

4.2 Graphs

Visualization approaches have proven to be an invaluable tool for assessing the success of model training after it has been completed. The goal of using visualization to expose semantic elements inside a trained model is to aid engineers in understanding the model's output. TensorBoard offered the visualization and machine learning tools required for this study: Monitoring metrics such as error or success

- Visualizing the model graph (op and layer)
- Viewing histograms of weighted tenor values as they vary over time.
- Displaying pictures and text.
- Prompting TensorFlow applications by running them in the background.

4.3 Graphs of Model: 1 Faster R-CNN with Inception V2

4.3.1 Learning Rate

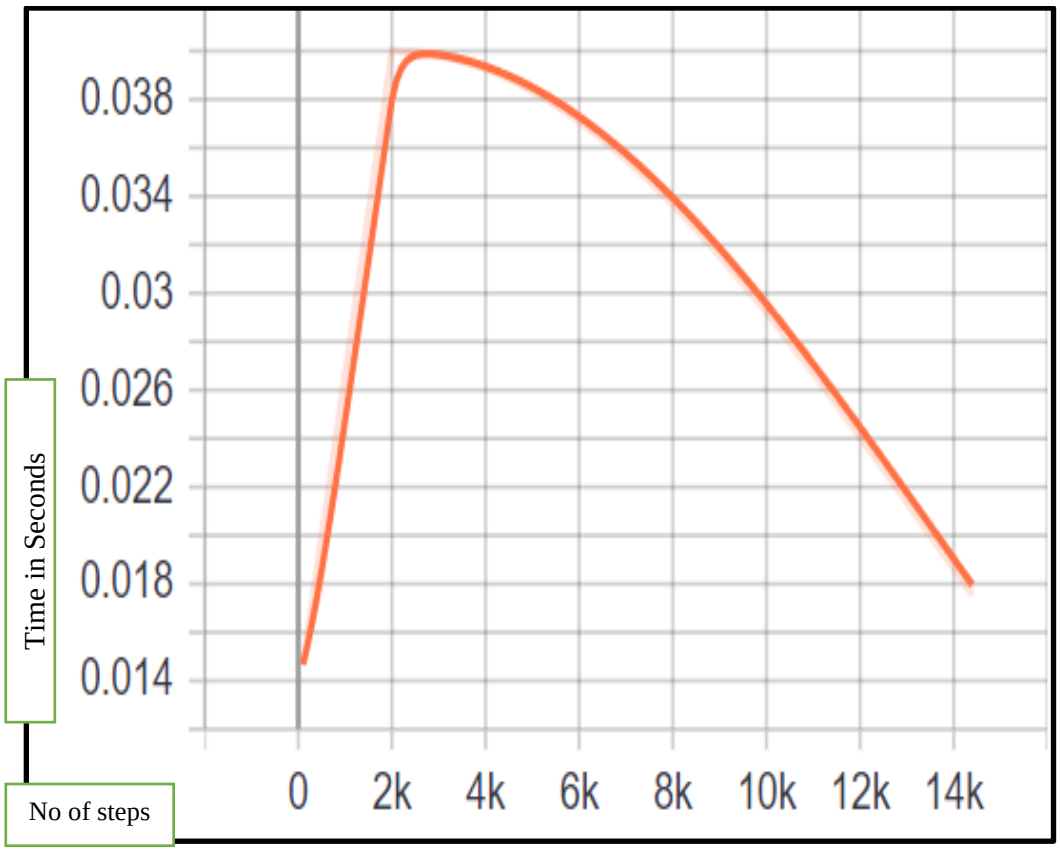


Figure 4.1: Histogram of Learning Rate

The learning rate shows the model is learned to the problem. This graph shows the learning rate of model trained for this research.

4.3.2 Classification Loss

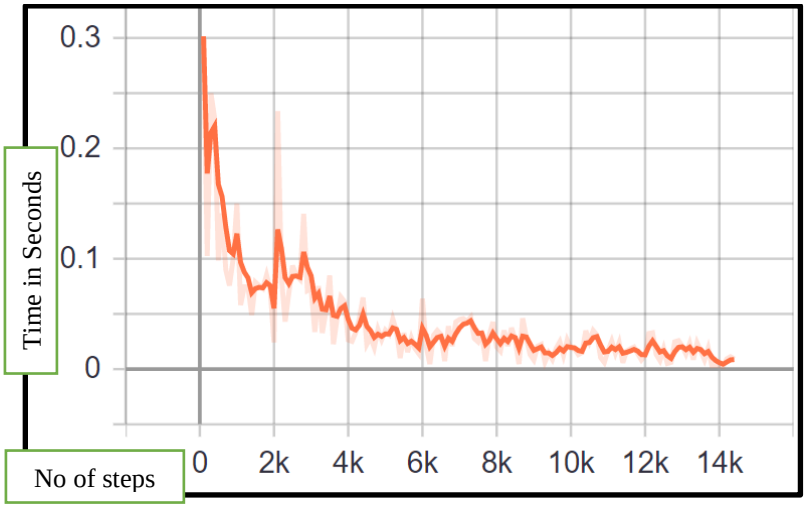


Figure 4.2: Histogram of Classification Loss

Classification loss refers to the loss during training the model fails to classify a vehicle. This graph represents the classification loss with respect to steps.

4.3.3 Localization Loss

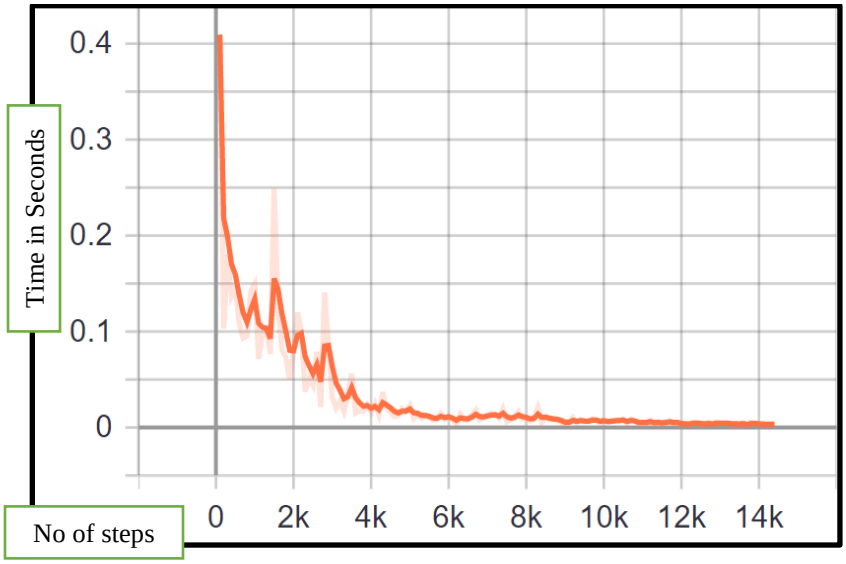


Figure 4.3: Histogram of Localization Loss

Localization refers to detection of object and drawing the bounding box around an object in an image. This graph shows the localization loss, at horizontal axis, steps are presented and at vertical axis, loss is presented in points.

4.3.4 Regulization Loss

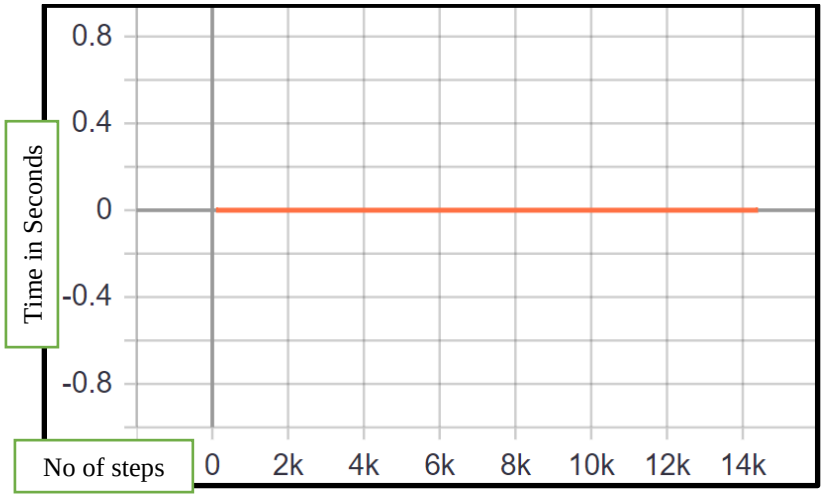


Figure 4.4: Histogram of Regulization Loss

Regulization refers to detection of object and drawing the bounding box around an object in an image. This graph shows the regulization loss, at horizontal axis, steps are presented and at vertical axis, loss is presented in points.

4.3.5 Total Loss

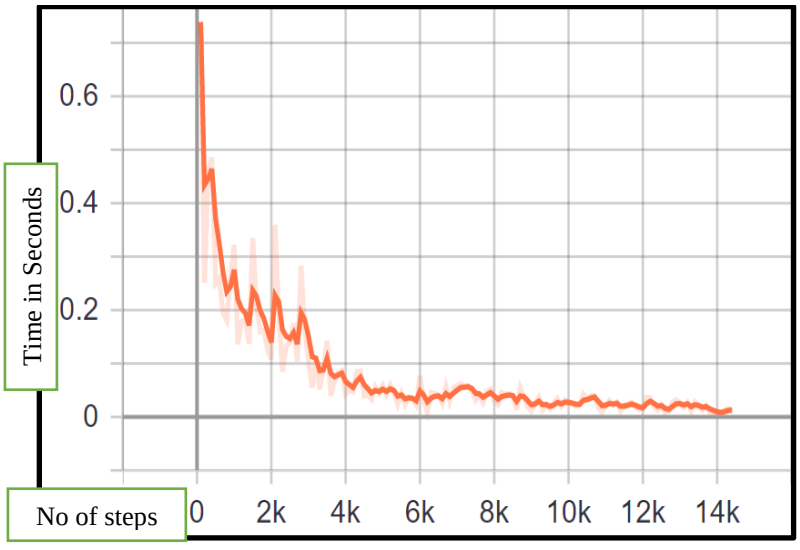


Figure 4.5: Histogram of Total Loss

In this graph, loss is represented on different steps during training of Model. At horizontal axis, steps are presented and at vertical axis, loss is represented in points.

4.3.6 Global Steps per second Graph

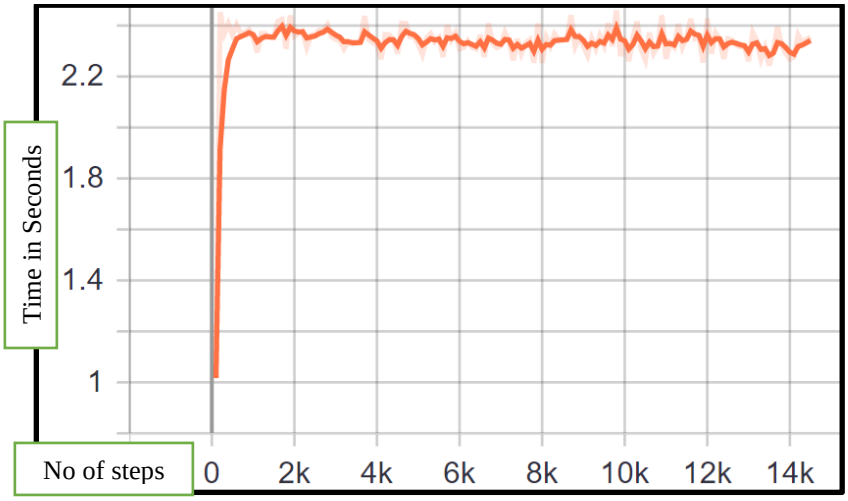


Figure 4.6: Histogram of Global Steps per second

This graph shows the global steps, the global step records the total number of iterations.

4.4 Graphs of Model 2: SSD-MobileNet V-2

4.4.1 Learning Rate

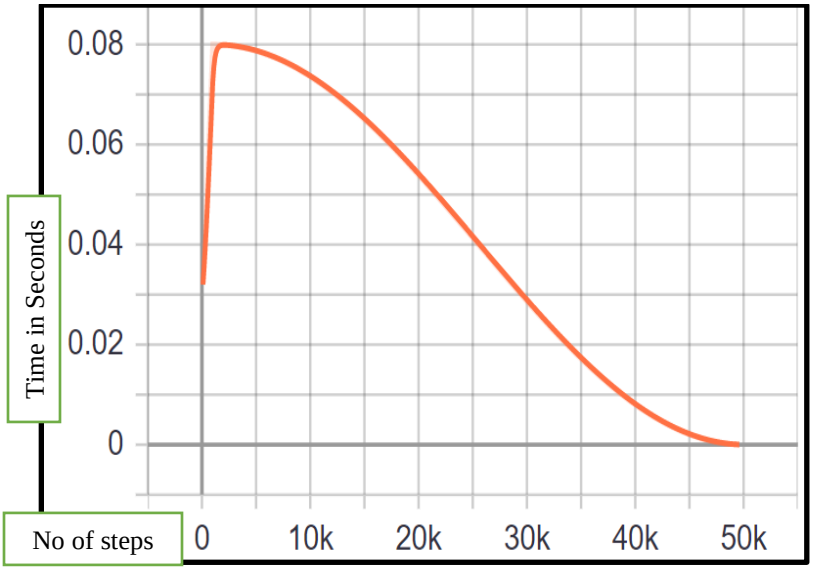


Figure 4.7: Histogram of Learning Rate

The learning rate shows how quickly the model is adapted to the problem. This graph shows the learning rate of model trained for this research.

4.4.2 Classification Loss

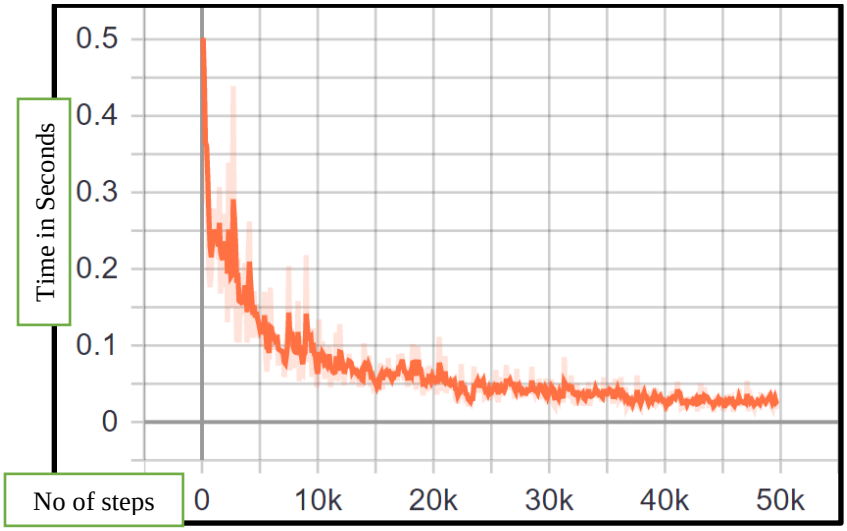


Figure 4.8: Histogram of Classification Loss

Classification loss refers to the loss during training the model fails to classify a vehicle. This graph represents the classification loss with respect to steps.

4.4.3 Localization Loss

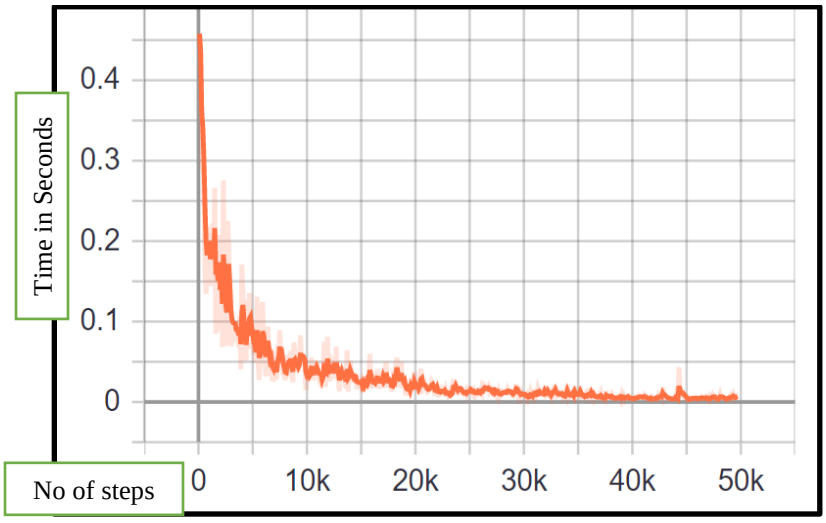


Figure 4.9: Histogram of Localization Loss

Localization refers to detection of object and drawing the bounding box around an object in an image. This graph shows the localization loss, at horizontal axis, steps are presented and at vertical axis, loss is presented in points.

4.4.4 Total Loss

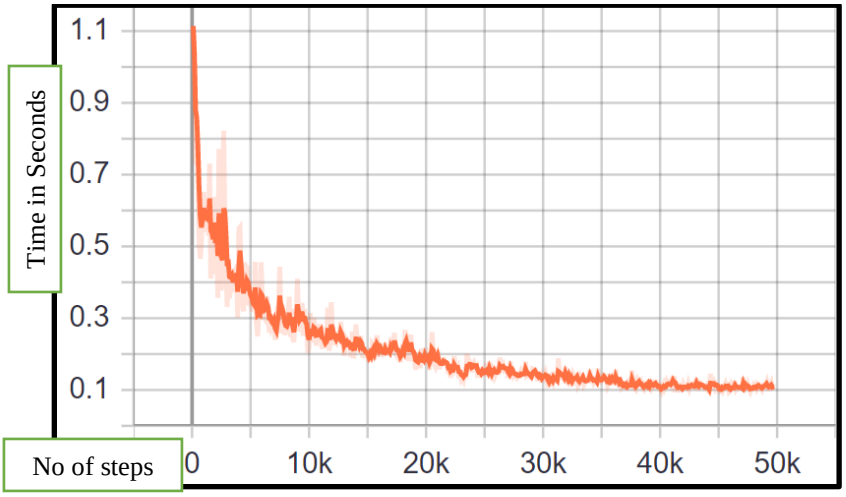


Figure 4.10: Histogram of Total Loss

In this graph, loss is represented on different steps during training of Model. At horizontal axis, steps are presented and at vertical axis, loss is represented in points.

4.4.5 Global Steps per second Graph

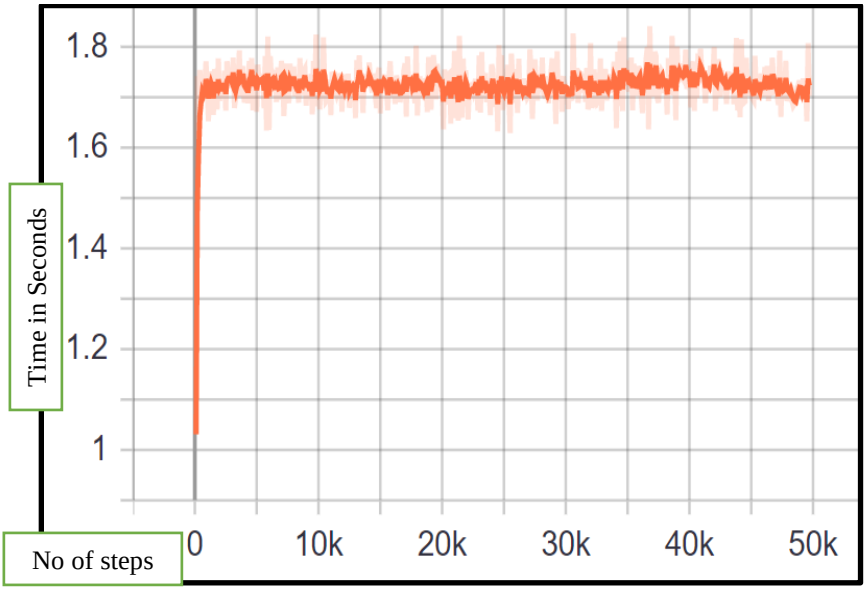


Figure 4.11: Histogram of Global steps per second

This graph shows the global steps, the global step records the total number of iterations.

4.5 Inference

Both the training and testing datasets for this study were split into two separate sets. The dataset included 487 pictures. The model was trained on 350 images from a total of 487. For testing purposes, 137 images were removed from the dataset. After recognizing the cotton bolls in these 150 images, various images were utilized to evaluate the model's ability to recognize the vehicle or vehicles from those images.

The following steps were taken to put the model through its paces:

- For object recognition, open command prompt from directory, then set python path environment variable o Set
- PYTHONPATH=C:/tensorflow1/research;/C:/tensorflow1/research;/C:/tensorflow1/research/slim
- For testing purposes, add a photo of an item or objects into the Object detection image.py file and modify the IMAGE NAME variable to match the file name of the picture in the object detection> folder.

This was followed by the execution of "Python Object detection image.py."

The image's script was opened and run. The initialization process took ten seconds, after which the output was shown.

Figure 4.12: Detected Boll Rot

In the above diagram, the trained model detected the cotton boll rot from the image and draw a bounding box around the object, it identified the cotton boll and classify the cotton boll as Affected, and it also showed the precision i.e. 88%.

Figure 4.13: Detected Boll Rot affected and non-affected

In the above diagram, the trained model detected the cotton bolls and draw a bounding box around the object, it identified the cotton bolls and classify the cotton bolls as affected and non-affected it also showed the precision i.e. 89%.

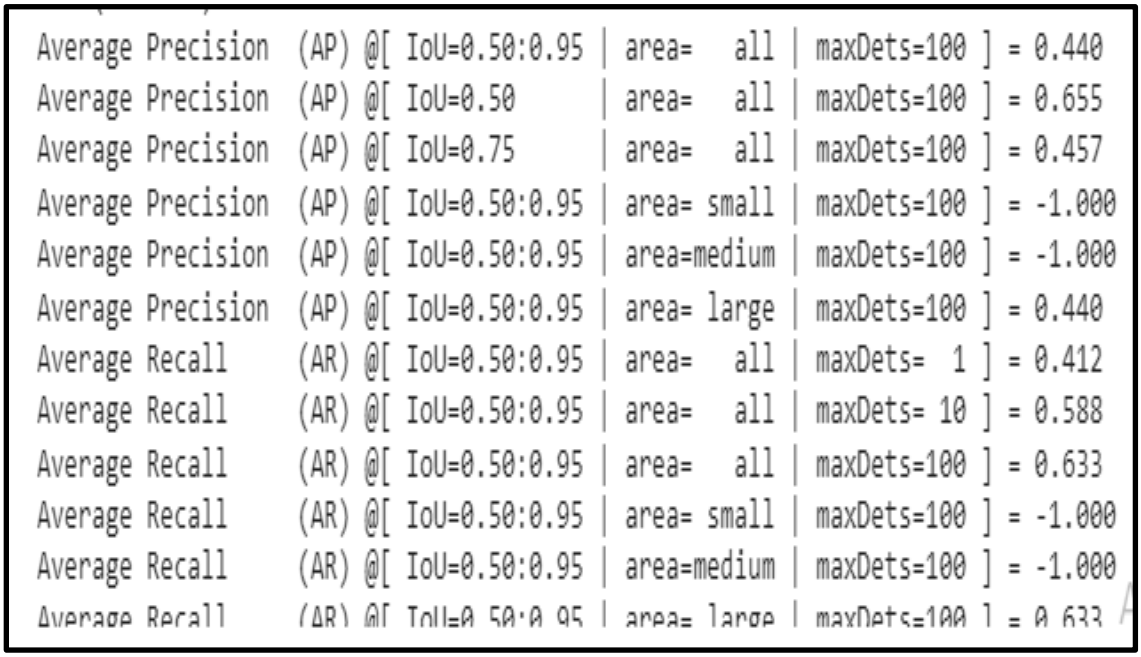
4.6 Results of Model 1: Faster R-CNN with Inception-V2

Table 4.1: Results of Faster RCNN with Inception-V2

```
Accumulating evaluation results...
DONE (t=0.02s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.897
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.897
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.897
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.897
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.699
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.916
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.916
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.916
INFO:tensorflow:Eval metrics at step 14000
I0117 20:21:08.582579 139695634917248 model_lib_v2.py:1015] Eval metrics at step 14000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP: 0.897336
```

4.7 Results of Model 2: SSD-MobileNet V-2

Table 2.2: Results of SSD with MobileNet-V2



4.8 Results Comparison of Models

Table 4.3: Comparison of Models

Model	Training duration	Batch Size	No. of training steps	Average Precision (AP)	Mean Average Precision (AP)	Average Recall (AR)	Intersection over Union (IoU)
Faster RCNN with Inception-V2	3 Hours	4	14000	89%	89%	91%	0.50:0.90
SSD with MobileNet-V2	8 Hours	4	50000	65%	65%	63%	0.50:0.90

4.9 Conclusion

In the proposed cotton boll rot disease detection model named Faster RCNN with Inception V2 and SSD MobileNet-V2, both the training and development of the image dataset, which was divided into categories of affected and non-affected cotton bolls have been done successfully. In this research, we have used deep learning algorithm and image processing techniques to detect cotton bolls rot from the images obtained from various relevant sources. We train these two models and apply them for real-time recognition by using the training data. The experimental results show that the Faster R-CNN with the InceptionV2 model can reliably identify whether the cotton bolls rot disease exists or not in an image compared to the other proposed models. In addition, the method of cotton bolls recognition from the image is very precise and accurate with this deep learning-based approach. The technique of OpenCV deep neural networks used in this model generated fruitful results. Classification of images was done accurately using the Mobilenet-V2 image classifier, which is one of the uniqueness of the proposed approach. Many existing researches faced problematic results, while some were able to generate better accuracy with their dataset. The problem of various wrong predictions have been successfully removed from the model as the dataset used was collected from various other sources and images used in the dataset was cleaned manually to increase the accuracy of the results. Real-world applications are a much more challenging issue for the upcoming future. The SSD MobileNet-V2 model should hopefully help the concerned authorities in this great pandemic situation which had largely gained roots in most of the world. The object detection technique, Faster RCNN Inception V2, uses regions to identify the items. The network does not look at the entire image at once, but instead focuses on different areas of it in a sequential manner. There are two issues that arise as a result of this. To extract all of the items, the program must make multiple runs through a single image. Because there are multiple systems operating simultaneously, the performance of the systems that follow is influenced by the performance of the previous systems. In Faster RCNN, object proposal takes more time because there are multiple systems working in parallel at the same time, the performance of each system is affected by the performance of the previous system. The trained models are often executed on GPUs because they are computationally intensive. When deploying models at scale, however, GPUs become extremely expensive.

CHAPTER 5 SUMMARY

Cotton bolls rot diseases have been a significant concern in agriculture for years. As a consequence of the use of DL techniques, early illness diagnosis has been possible and losses have been minimized. Rapid processing is made possible by readily accessible technology and recent developments in deep learning. However, there is room for improvement in the decision-making process. Currently, the models on the market don't perform well when evaluated in real-world scenarios. An innovative method for detecting cotton boll rot disease was presented to solve the primary practical constraints as a result of this and the authors' earlier work. Images of cotton bolls from a novel dataset were provided in this thesis, which included varied perspectives and meteorological conditions, as well as labelled for classification and detection. As a result, the model's classification accuracy and practical applicability will be improved as a result. A lack of examples is a common stumbling block since it leads to over fitting. Several augmentation methods were used to get around this problem. Faster R-CNN with the inception-V2 and SSD with MobileNet-V2 was suggested for the identification of cotton boll rot disease. For example, Faster RCNN with Inception V2 requires around 2.8% more computation per frame, but SSD with MobileNet V2 only need about 1.7% more computation. An 89% and 67% accuracy rate was attained by the trained model.

1

REFERENCES

Adiwinata, Y., A. Sasaoka, I.P. Agung Bayupati and O. Sudana. 2020. Fish Species Recognition with Faster R-CNN Inception-v2 using QUT FISH Dataset. Lontar Komput. J. Ilm. Teknol. Inf. 11:144.

Alamsyah, D. and M. Fachrurrozi. 2019. Faster R-CNN with inception v2 for fingertip detection in homogenous background image. J. Phys. Conf. Ser. 1196.

Ali, H., M. Khursheed, S.K. Fatima, S.M. Shuja and D.S. Noor. 2019. Object recognition for dental instruments using SSD-mobilenet. 2019 Int. Conf. Inf. Sci. Commun. Technol. ICISCT 2019 1–645444.

Amisse, C., M.E. Jijón-Palma and J.A. Silva Centeno. 2021. Fine-tuning deep learning models for pedestrian detection. Bol. Ciencias Geod. 27:0–1876656.

Anuar, M.M., A.A. Halin, T. Perumal and B. Kalantar. 2022. Aerial Imagery Paddy Seedlings Inspection Using Deep Learning. Remote Sens. 1456498.

Arsenovic, M., M. Karanovic, S. Sladojevic, A. Anderla and D. Stefanovic. 2019. Solving current limitations of deep learning based approaches for plant disease detection. Symmetry (Basel). 1198876.

Chen, L., S. Lin, X. Lu, D. Cao, H. Wu, C. Guo, C. Liu and F.Y. Wang. 2021. Deep Neural Network Based Vehicle and Pedestrian Detection for Autonomous Driving: A Survey. IEEE Trans. Intell. Transp. Syst. 22:3234–3246.

Darapaneni, N., P. Bangade, A. Mane, U. Maheshwari, S.C. Thorawade, R. Borse and A.R. Paduri. 2021. Traffic monitoring and analysis at toll plaza. 2021 IEEE Int. IOT, Electron. MechatronicsConf.IEMTRONICS2021Proc.,doi:10.1109/IEMTRONICS52119.2021.9422543.

Ehetisham-ul-Haq, M., A. Rashid, M. Kamran, M. Idrees, S. Ali, A. Irum, S. Il Yasin and F. Siddique. 2017. Disease forecasting model for newly emerging bacterial seed and boll rot of cotton disease and its vector (Dysdercus cingulatus). Arch. Phytopathol. Plant Prot. 50:885–899.

Giron, N.N.F., R.K.C. Billones, A.M. Fillone, J.R. Del Rosario, A.A. Bandala and E.P. Dadios. 2020. Classification between Pedestrians and Motorcycles using FasterRCNN Inception and SSD MobileNetv2. 2020 IEEE 12th Int. Conf. Humanoid, Nanotechnology, Inf. Technol. Commun. Control. Environ. Manag. HNICEM 2020 1–69876.

Hassan, S.M., A.K. Maji, M. Jasiński, Z. Leonowicz and E. Jasińska. 2021. Identification of plant-leaf diseas[1] S. M. Hassan, A. K. Maji, M. Jasiński, Z. Leonowicz, and E. Jasińska, "Identification of plant-leaf diseases using cnn and transfer-learning approach," Electron., vol. 10, no. 12, 2021, doi: 10.3390/electronics101213. Electron. 10.

Imran, M.A., A. Ali, M. Ashfaq, S. Hassan, R. Culas and C. Ma. 2018. Impact of Climate Smart Agriculture (CSA) practices on cotton production and livelihood of farmers in Punjab, Pakistan. Sustain. 10-12453.

Iqbal, M., M.A. Khan and S. Ul-Allah. 2021. High density cotton population in late sowing improves productivity and tolerance to cotton leaf curl virus under semi-arid subtropical conditions. J. Plant Dis. Prot. 128:685–692.

Iyer, R., P. Shashikant Ringe, R. Varadharajan Iyer and K. Prabhulal Bhensdadiya. 2021. Comparison of YOLOv3, YOLOv5s and MobileNet-SSD V2 for Real-Time Mask Detection Comparison of YOLOv3, YOLOv5s and MobileNet-SSD V2 for Real-Time Mask Detection View project Comparison of YOLOv3, YOLOv5s and MobileNet-SSD V2 for Real-Time Mask Detection. Artic. Int. J. Res. Eng. Technol. 1156–1160.

Karaman, O., A. Alhudhaif and K. Polat. 2021. Development of smart camera systems based on artificial intelligence network for social distance detection to fight against COVID-19. Appl. Soft Comput. 110:107610.

Karar, H., M.A. Bashir, M. Haider, N. Haider, K.A. Khan, H.A. Ghramh, M.J. Ansari, Ç. Mutlu and S.M. Alghanem. 2020. Pest susceptibility, yield and fiber traits of transgenic cotton cultivars in Multan, Pakistan. PLoS One 15:1–15.

Khan, M.A., T. Akram, M. Sharif, M. Awais, K. Javed, H. Ali and T. Saba. 2018. CCDF: Automatic system for segmentation and recognition of fruit crops diseases based on correlation coefficient and deep CNN features. Comput. Electron. Agric. 155:220–236.

Kumar, S., A. Jain, A.P. Shukla, S. Singh, R. Raja, S. Rani, G. Harshitha, M.A. Alzain and M. Masud. 2021. A Comparative Analysis of Machine Learning Algorithms for Detection of Organic and Nonorganic Cotton Diseases. Math. Probl. Eng. 2021.

Latif, M.R., M.A. Khan, M.Y. Javed, H. Masood, U. Tariq, Y. Nam and S. Kadry. 2021. Cotton leaf diseases recognition using deep learning and genetic algorithm. Comput. Mater. Contin. 69:2917–2932.

Lin, T.L., H.Y. Chang and K.H. Chen. 2020. The pest and disease identification in the growth of sweet peppers using faster r-cnN and mask r-CNn. J. Internet Technol. 21:605–614.

Liu, J. and X. Wang. 2020. Early recognition of tomato gray leaf spot disease based on MobileNetv2-YOLOv3 model. Plant Methods 16:1–16.

Liu, Y., P. Sun, N. Wergeles and Y. Shang. 2021. A survey and performance evaluation of deep learning methods for small object detection. Expert Syst. Appl. 172:114602.

Lorgus Decker, L.G., A. da Silva Pinto, J.L. Flores Campana, M.C. Neira, A.A. dos Santos, J.S. Conceição, M.A. Angeloni, L.T. Li and R. da S. Torres. 2020. Mobtext: A compact method for scene text localization. VISIGRAPP 2020 - Proc. 15th Int. Jt. Conf. Comput. Vision, Imaging Comput. Graph. Theory Appl. 5:343–350.

Magsi, A., R. Shaikh, Z. Shar, R. Arain and A. Soomro. 2021. A Novel Framework For Disease Severity Level Identification Of Cotton Plant Using Machine Learning Techniques. Researchgate.Net 367–373.

Mostafa, A.M., S.A. Kumar, T. Meraj, H.T. Rauf, A.A. Alnuaim and M.A. Alkhayyal. 2022. Guava disease detection using deep convolutional neural networks: A case study of guava plants. Appl. Sci. 12.

Nagrale, D.T., S.P. Gawande, N. Gokte-Narkhedkar and V.N. Waghmare. 2020. Association of phytopathogenic Pantoea dispersa inner boll rot of cotton (Gossypium hirsutum L.) in Maharashtra state, India. Eur. J. Plant Pathol. 158:251–260.

Patel, I. and S. Patel. 2020. An optimized deep learning model for flower classification using NAS-FPN and faster RCNN. Int. J. Sci. Technol. Res. 9:5308–5319.

Rafiq, A., W.R. Ali, M. Asif, N. Ahmed, W.S. Khan, S. Mansoor, S.Z. Bajwa and I. Amin. 2021. Development of a LAMP assay using a portable device for the real-time detection of cotton leaf curl disease in field conditions. Biol. Methods Protoc. 6:1–8.

Rahmaniar, W. and A. Hernawan. 2021. Real-time human detection using deep learning on embedded platforms: A review. J. Robot. Control 2:462-468Y.

Ramacharan, D.S. 2021. A 3-Stage Method for Disease Detection of Cotton Plant Leaf using Deep Learning CNN Algorithm. Int. J. Res. Appl. Sci. Eng. Technol. 9:2503–2510.

Sajid, M., A. Rashid, M. Abid, H. Jamil, M.R. Bashir, Z.M. Sarwar, R. Perveen and S. Chohan. 2018. Influence of Bt. germplasm on bacterial blight disease of cotton in Pakistan. Pakistan J. Phytopathol. 30:155–162.

Saleem, R.M., R. Kazmi, I.S. Bajwa, A. Ashraf, S. Ramzan and W. Anwar. 2021. IOT-Based Cotton Whitefly Prediction Using Deep Learning. Sci. Program. 2021.

Salman Qadri. 2021. A Spatial Model of K-Nearest Neighbors for Classification of Cotton (Gossypium) Varieties based on Image Segmentation. Lahore Garrison Univ. Res. J. Comput. Sci. Inf. Technol. 5:25–40.

Sandler,M.,A.Howard,M.ZhuandA.Zhmoginov.2018.Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.pdf. 4510–4520.

SARDOĞAN, M., Y. ÖZEN and A. TUNCER. 2020. Faster R-CNN Kullanarak Elma Yaprağı Hastalıklarının Tespiti. Düzce Üniversitesi Bilim ve Teknol. Derg. 8:1110–1117.

Sethy, P.K., N.K. Barpanda, A.K. Rath and S.K. Behera. 2020. Deep feature based rice leaf disease identification using support vector machine. Comput. Electron. Agric. 175:105527.

Siddiqua, R., S. Rahman and J. Uddin. 2021. A deep learning-based dengue mosquito detection method using faster r-cnn and image processing techniques. Ann. Emerg. Technol. Comput. 5:11–23.

Sivakumar, A.N.V., J. Li, S. Scott, E. Psota, A.J. Jhala, J.D. Luck and Y. Shi. 2020.

Comparison of object detection and patch-based classification deep learning models on mid-to late-season weed detection in UAV imagery. Remote Sens. 12.

Srivastava, A., A. Dalvi, C. Britto, H. Rai and K. Shelke. 2020. Explicit Content Detection using Faster R-CNN and SSD MobileNet v2. 5572–5577.

Sujatha, R., J.M. Chatterjee, N.Z. Jhanjhi and S.N. Brohi. 2021. Performance of deep learning vs machine learning in plant leaf disease detection. Microprocess. Microsyst. 80:103615.

Vasconez, J.P., J. Delpiano, S. Vougioukas and F. Auat Cheein. 2020. Comparison of convolutional neural networks in fruit detection and counting: A comprehensive evaluation. Comput. Electron. Agric. 173:105348.

Wang, C. and Z. Xiao. 2021. Lychee surface defect detection based on deep convolutional neural networks with GAN-based data augmentation. Agronomy 11:1–17.

Wang, Q., F. Qi, M. Sun, J. Qu and J. Xue. 2019. Identification of Tomato Disease Types and Detection of Infected Areas Based on Deep Convolutional Neural Networks and Object Detection Techniques. Comput. Intell. Neurosci. 2019.

Yadav, N. and U. Binay. 2017. Comparative Study of Object Detection Algorithms. Int. Res. J. Eng. Technol. 586–591.

Zaki, M.A., S. Narejo, M. Ahsan, S. Zai, M.R. Anjum and N.U. Din. 2021. Image-based Onion Disease (Purple Blotch) Detection using Deep Convolutional Neural Network. Int. J. Adv. Comput. Sci. Appl. 12:448–458.

Židek, K., P. Lazorík, J. Pitel' and A. Hošovský. 2019. An automated training of deep learning networks by 3D virtual models for object recognition. Symmetry (Basel). 11.