

**A MACHINE LEARNING BASED DESIGN AND
DEVELOPMENT OF CROP LEAF DISEASE IDENTIFYING
SYSTEM**

A PROJECT REPORT

(Project Work II)

Submitted By

HARISH M

(19ECR056)

KASIRAM P

(19ECR076)

MOUNESH K

(19ECR109)

*in partial fulfilment of the requirements
for the award of the degree
of*

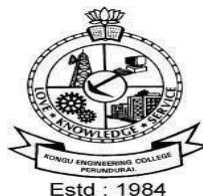
BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

ENGINEERING



KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE-638 060

MAY 2023

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE – 638060

MAY 2023

BONAFIDE CERTIFICATE

This is to certify that the project work II report entitled **A MACHINE LEARNING BASED DESIGN AND DEVELOPMENT OF CROP LEAF DISEASE IDENTIFYING SYSTEM** is the bonafide record of project work done by **HARISH M (19ECR056), KASIRAM P (19ECR076), MOUNESH K (19ECR109)** in partial fulfilment of the requirements for the award of Degree of Bachelor of Engineering in Electronics and Communication Engineering of Anna University, Chennai during the year 2022-2023.

SUPERVISOR

Dr.S.MAHESWARAN M.E., Ph.D.,

Associate Professor

Department of ECE

Kongu Engineering College

Perundurai-638060

HEAD OF THE DEPARTMENT

Dr.T.MEERADEVI M.E., Ph.D.,

Professor & Head

Department of ECE

Kongu Engineering College

Perundurai-638060

Date:

Submitted for the VIII end semester viva voce examination held on_____.

INTERNAL EXAMINER

EXTERNAL EXAMINER

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE – 638060

MAY 2023

DECLARATION

We affirm that the project work II report titled **A MACHINE LEARNING BASED DESIGN AND DEVELOPMENT OF CROP LEAF DISEASE IDENTIFYING SYSTEM** being submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Date:

(Signature of the Candidates)

**HARISH M
(19ECR056)**

**KASIRAM P
(19ECR076)**

**MOUNESH K
(19ECR109)**

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date:

Name & Signature of the supervisor with seal

ABSTRACT

In recent years, there is an important development in the research of computer vision using the deep learning technique. The large amount of the training samples will determine the success of these tasks. The implementation of intelligent software to identify and classify objects is a technology of growing importance in many fields, including agriculture. Deep convolution networks are achieving a tremendous performance in object detection and classification. The current study proposed a deep architecture of convolutional neural network for the purpose of improving the accuracy of detection and classification of crop leaf diseases in apple, corn, tomato, grape and potato. The process of identifying the location, existence, and the class of the object which they belong to by using a bounding box is known as object detection. The crop leaf disease detection technique creates a bounding box and allocating a label as per the leaf disease for the object using YOLO v4 technique. In YOLO v4, the picture cells are split and then sent into CNN networks where the characteristics are acquired, and each cell states a bounding box before directly classifying the objects at the end. If more than one bounding box is given to a leaf, the system must use the non-maximum suppression technique to decrease the count of the bounding boxes marked for the leaves in the image. Finally, the leaves which are present in the image, or the real time video captured by the camera can be calculated using the number of the bounding boxes marked for the leaves and calculate the accuracy of the detected object compared with the original object. For the object detection, the dataset is created by using the collection of leaves images. Creation of the dataset requires the labelling of the images which contains the leaves, and it can be done with the help of LabelImg software. Nearly 17000 images are taken for the creation of the dataset. After labelling the images, the dataset is trained using the Google Colab GPU. Created dataset is used to mark the bounding boxes to the leaves present in the image or real time video. The framework for implementing YOLOv4 technique is the darknet. Nearly 80 images which is present in the collection are used for testing purposes. The crop leaf disease detection method had an accuracy of 93% when tested on the dataset.

ACKNOWLEDGEMENT

First and foremost, we acknowledge the abundant grace and presence of almighty throughout different phases of the project and its successful completion.

We wish to express our hearty gratitude to our honourable Correspondent **Thiru.A.K.ILANGO B.Com., M.B.A., LLB.,** and other trust members for having provided us with all necessary infrastructures to undertake this project.

We would like to express our hearty gratitude to our honourable Principal **Dr.V.BALUSAMY M.Tech., Ph.D.,** for his consistent encouragement throughout our college days.

We would like to express our profound interest and sincere gratitude to our respected Head of the Department **Dr.T.MEERADEVI M.E., Ph.D.,** for her valuable guidance.

We would like to express our sincere gratitude to our respected project coordinators **Dr.S.BALAMBIGAI M.E., Ph.D.,** Associate Professor and **Ms.K.AARTHI M.E.,** Assistant Professor, Department of Electronics and Communication Engineering, for their encouragement and valuable advice that made us carry out the project work successfully.

We would like to express our sincere gratitude to our beloved guide **Dr.S.MAHESWARAN M.E., Ph.D.,** Associate Professor, Department of Electronics and Communication Engineering for his ideas and suggestions, which have been very helpful to complete the project.

We are grateful to all the faculty and staff of the Department of the Electronics and Communication Engineering and persons who directly and indirectly supported this project.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	iii
	LIST OF FIGURES	vii
	LIST OF TABLES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
	1.1 OBJECT DETECTION	1
	1.2 YOLO	2
	1.3 CONVOLUTIONAL NEURAL NETWORKS	2
	1.4 OPENCV	3
	1.5 CROPS	3
	1.6 STREAMLIT	3
2	LITERATURE REVIEW	4
	2.1 GAP OF ANALYSIS	10
3	EXISTING METHODOLOGY	11
	3.1 METHODOLOGY	11
	3.2 DISADVANTAGES OF EXISTING METHOD	14
4	PROPOSED METHODOLOGY	15
	4.1 MATERIALS AND METHODS	15
	4.1.1 Need for Yolo	15
	4.1.2 Yolo	15
	4.1.3 Convolutional neural networks	17
	4.1.4 Convolutional layer	17
	4.1.5 Pooling layer	18
	4.1.6 Fully connected layer	19

		vi
	4.2 DERIVATION FOR FULLY CONNECTED CNN	19
	4.3 BLOCK DIAGRAM OF THE PROPOSED METHOD	29
	4.4 PRE TRAINED MODEL	30
	4.5 CUSTOM TRAINED MODEL	30
	4.6 DARKNET IN YOLO	31
	4.7 DATA COLLECTION	34
	4.8 IMAGE ANNOTATION	35
	4.9 TRAINING	39
	4.10 DETECTION	43
5	RESULTS AND DISCUSSION	45
6	CONCLUSION AND FUTURE SCOPE	48
	6.1 CONCLUSION	48
	6.2 FUTURE SCOPE	48
	REFERENCES	49

LIST OF FIGURES

FIGURE. NO	FIGURE	PAGE. NO
1.1	A simple neural network	2
3.1	Sample of training images	12
3.2	Confusion matrix for DenseNet121	13
3.3	Training history of the DenseNet121	13
4.1	Comparison of YOLOv4 vs YOLOv3 with the pretrained COCO dataset	16
4.2	CNN Architecture	17
4.3	Representation of the convolutional layer	18
4.4	Sigmoid activation	28
4.5	Block diagram of the proposed method	29
4.6	Making changes in the make file	31
4.7	Configuration file	33
4.8	Configuration file	34
4.9	Labeling the images using LabelImg software	36
4.10	Text file generated for an image	36
4.11	Text file containing the file path of the images want to train	37
4.12	Text file containing the file path of the images want to test	38
4.13	Contents of the crop data file	39
4.14	Command to get the runtime details	40
4.15	Mean average precision for 36000 th iteration	42
4.16	Mean average precision for 60000 th iteration	42
4.17	Newly created folder containing weights and image files	43
4.18	Detection of leaf disease for the input images	44
5.1	Homepage of the website	45
5.2	The predicted leaf disease in the uploaded image with the accuracy	46
5.3	Confusion matrix	47

LIST OF TABLES

TABLE.NO	TABLE	PAGE.NO
5.1	Performance metrics	46

LIST OF ABBREVIATIONS

ANN	Artificial Neural Networks
CNN	Convolutional Neural Networks
COCO	Common Objects In Context
ConvNet	Convolutional Neural Networks
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DCNN	Deep convolutional neural networks
DL	Deep Learning
DOTA	Dataset of Object deTection in Aerial images
GPU	Graphics Processing Unit
HTML	Hypertext Markup Language
IoT	Internet of Things
KNN	K-Nearest Neighbor
mAP	mean Average Precision
MCNN	Multilayer Convolutional Neural Network
ML	Machine Learning
NoCs	Networks on Convolutional feature maps
OpenCV	Opensource Computer Vision
R-CNN	Region-based Convolutional Neural Network
ResNet	Residual Network
SaCNN	Standard Template Library
SMOTE	Synthetic Minority Over-sampling TEchnique
SSD	Single Shot Multi-Box Detector
STL	Dataset of Object deTection in Aerial images
VGG	Visual Geometry Group
YOLO	You Only Look Once

CHAPTER 1

INTRODUCTION

A system that enables the computers to observe and comprehend the content of complex images, such as photos and recordings, is the field of study known as Computer vision aims to create. Industrial inspection, mobile robot navigation, human-computer interface in the armed forces, and image processing in medicine are just a few of the multidisciplinary applications of computer vision. Object classification, object verification, object identification, object recognition, and object detection are some of the common computer vision applications used to identify objects in photos or videos.

1.1 OBJECT DETECTION

Object localization and object tracking are the two degrees of object detection. The technique of locating and following both single and grouped items is known as object tracking. The technique of object detection has been made much easier by the creation of artificial neural networks. Neural networks are used to replicate the human cerebral structure in order to solve the fundamental learning issues. Numerous techniques are used by the neural networks to find patterns. The class of machine learning techniques are known as deep learning. The topic of computer vision known as deep learning has better functioning models that may need more training data but less knowledge of digital signal processing technique to operate by training it [1].

Deep learning needs a lot of annotated images used for training, hence it is preferable to make use of a GPU to speed up the process of training or else it takes one or more month to complete the dataset creation with the help of the processor in the laptop or computers. The approaches use deep learning for object detection uses the techniques like convolutional neural network (CNN), including R-CNN method and YOLO method. For the object detection make use an already trained object detector or create a new custom dataset to meet the requirements to satisfy the conditions provided by the industries. With this technique,

one can start with a network that has already been trained and afterwards modify it, that contributes for the purpose based on the requirement.

1.2 YOLO

YOLO refers to You Only Look Once. A deep CNN is used in the approach known as YOLO to find the object identification. In 2016, Redmon et al proposed the YOLO architecture model. It simply takes a single look at the entire image which was given as an input to the convolutional neural networks before moving through the system and identifying things [2].

A source image is split into $M \times M$ matrices using YOLO. If a certain region in an image corresponds for object distinguish, the centre of an object will mark into a matrix region. The confidence value will show how the leaf has been considered as an object inside it and how exactly the bounding box will be. YOLO has structure of organisation made up of twenty-four convolutional layers, two completely connected layers present in the end and a max pooling layer.

1.3 CONVOLUTIONAL NEURAL NETWORKS

A CNN is also known as ConvNet is attaining so much popular in the area of deep learning. A convolutional neural network consists of an input, many hidden and finally an output layer. The most popular pooling layer, which takes the structure of non-linear down sampling, is a function of max pooling. Finally, high complex tasks were completed by the fully connected layers. Convolutional networks are said to be the feedforward networks because there is only one direction in which information can flow: through their sources to their outputs. The figure 1.1 shows a basic simple convolutional neural network, the structure of the convolutional neural network may vary for the different purposes.

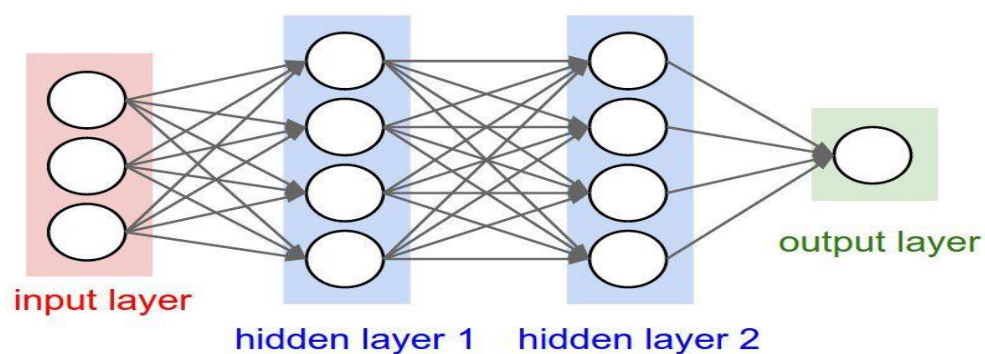


Figure 1.1 A simple neural network

1.4 OPENCV

A standard architecture for applications of computer vision was created with OpenCV, which is now utilized to speed up the incorporation of artificial intelligence into products for sale. Businesses may quickly use and alter the code to serve their needs thanks to OpenCV. It is a large, well-rounded library with 2500 optimized algorithms, as well as a full range of traditional and modern computer vision techniques and machine learning approaches. These algorithms are utilized for a variety of tasks, including finding and recognizing faces. Categorize human acts by identifying items. OpenCV provides a functionalized user interface that is compatible with STL containers and is written locally in the simple to understand programming language C++.

1.5 CROPS

The current study proposed a deep architecture of convolutional neural network for the purpose of improving the accuracy of detection and classification of crops like apple, corn, tomato, grape and potato illness. These diseases can be caused mainly by the fungi, virus, and bacterial organisms. Apple leaf disease can be classified into apple rust leaf disease and apple scab leaf disease. Corn leaf disease can be classified into corn grey leaf spot, corn leaf blight and corn rust leaf disease. Grape leaf disease named grape leaf black rot disease. Potato leaf disease can be classified into potato leaf early and late blight diseases. Tomato leaf disease can be classified into tomato bacterial spot, late blight, mosaic virus, yellow virus, mold leaf and septoria leaf spot disease. The identification of the crop leaf disease is still a challenging task for the farmers.

1.6 STREAMLIT

Streamlit, an open-source Python framework, makes it simple to create and distribute beautiful, customized web apps for data science and machine learning. One of the most important steps in the data science stream is model deployment. Python offers a variety of options for sharing the model. The disadvantage of using the popular frameworks like Django and Flask is that the user needs a knowledge in HTML, CSS, and JavaScript. With the aid of streamlit, the user can deploy any Python project or machine learning model with ease and without regard for the front end. In this project, the user can upload their image on the website, and it will process the image and give the output in the same website.

CHAPTER 2

LITERATURE REVIEW

John C Valdoria et al., in [3] have developed an android based plant leaf disease detection mobile application through image processing using neural networks. This study focuses on using deep learning neural networks for image processing to identify prevalent illnesses on agricultural plants are found in the Philippines. In order to identify the illness of the plant, photos of terrestrial plants were taken using Android-based smartphones, and the sickness was then distinguished using a Deep Learning Neural Network Algorithm. Given the quantity of photos utilized, classification techniques that could detect the illnesses with a specific rate and accuracy were employed to train the findings. As a result, the classification using Deep Learning Neural Networks shows how to identify the terrestrial plants in the Philippines that are commonly affected by disease.

Uday Pratap Singh et al., in [4] have proposed a study about a fungal disease which highly affects the mango tree named Anthracnose disease. The major goal of this article is to create a suitable system for an early and affordable solution to this problem by developing a method that is acceptable and effective for diagnosing the disease and its symptoms. The categorization of various fungal infections has been more and more popular over the past few years as a result of computer vision and deep learning approaches' improved performance in terms of efficiency and accuracy. As a result, a Multilayer Convolutional Neural Network (MCNN) is suggested for this study in order to classify mango leaves that have been infected with the fungus Anthracnose. The suggested MCNN model is predicted to have a greater classification accuracy than existing state-of-the-art methods, according to the results.

Shruthi U et al., in [5] have proposed a review on plant disease detection using machine learning classification techniques. With the nation's rapidly expanding population and rising food consumption, agriculture is crucial in India. As a result, agricultural output needs to grow. The presence of bacterial, viral, and fungal diseases has a significant impact on low crop output. Techniques for detecting plant illnesses can be used to stop it. Methods

of machine learning can be used to identify illnesses since they focus primarily on the data itself and give importance to the results of certain tasks. This study compares several machine learning classification algorithms for detection of plant diseases and outlines the steps of a comprehensive plant disease detection system. In this study, it was shown that convolutional neural networks provide good accuracy and can identify a greater variety of agricultural illnesses.

Bin Wang et al., in [6] have proposed a paper that focuses on plant leaves classification using few-shot learning method based on Siamese network. In order to address a leaf classification issue with a short sample size, a few-shot learning strategy dependent on the Siamese network architecture is suggested in this research. Initially, a simultaneous two-way CNN with shared weights extracts the characteristics of two separate pictures. The network then uses a loss function to train the metric space, whereby comparable leaf sample distances are near together, and differing leaf sample distances are far apart. In addition, the construction of the metric space using the spatial structure optimizer (SSO) approach is suggested, which will assist to increase the accuracy of leaf categorization. Lastly, leaves are categorized in the learnt metric space using a k-nearest neighbor (kNN) classifier. As a performance indicator, consider using average classification accuracy. The performance of the technique is assessed using the open Flavia dataset, Leafsnap dataset, and Swedish dataset.

Achyut Morbekar et al., in [7] have presented a paper on crop disease detection using the YOLO. In this paper, they said that farmers can choose from a broad variety of crops that are ideal for their needs. Yet, due to a lack of information, farmers are confused about the various illnesses that plague farms. Most of the farmers struggle and lose a lot of time harvesting ill crops. In order to prevent severe damage and increase output, the issue must be identified and evaluated quickly. The suggested system uses the object detection technique's innovative method, known as YOLO, to find plant diseases (You Only Look Once). YOLO analyses leaf photos at a real-time rate of 45 frames per second, which is quicker than existing object recognition methods. Before processing the picture, it splits the image into several grid cells. One neural network makes a single assessment to anticipate the boundary boxes and class probabilities. This significantly improves the efficiency and precision of disease diagnosis on leaves.

Vijayakumar Ponnusamy et al., in [8] have presented a paper about smart glass: real-time leaf disease detection using YOLO transfer learning. ANN architectures have high degrees of accuracy. However, these architectures necessitate GPUs with strong computing power, which in turn expands the system's total size. Machine learning algorithms can now be processed by existing systems, but they are either expensive or not portable. They offer a Smart Glass that is capable of extremely accurate binary categorization of data in real-time, combining the benefits of mobility with a cutting-edge You Only Look Once (YOLOv3) technology. With agricultural data used to train the architecture, this wearable gadget would be able to distinguish between healthy and diseased leaves in real-time. The architecture might be trained using various datasets to help researchers find answers to a wide range of issues in the agriculture, healthcare, etc.

Qiufeng Wu et al., in [9] have recently proposed and successfully implemented tomato leaf disease detection technique-based data augmentation called DCGAN. In this study, a novel technique of data augmentation via generative adversarial networks (GANs) is presented for leaf disease detection, with the goal of increasing the detection performance of tomato leaf diseases. This model can obtain a top-1 average identification accuracy of 94.33% using generated pictures supplemented by deep convolutional generative adversarial networks (DCGAN) and actual photos as the input of GoogLeNet. A better model for training and evaluating 5 classes of tomato leaf photos was created by tweaking the convolutional neural system architecture, altering the hyper-parameters, and using alternative generative adversarial networks. As a result of the diversity of the pictures produced by DCGAN, which also increase the data set's size, the model has a high generalization impact.

Vaishnave M et al., in [10] has been shown that deep convolutional neural networks (DCNN) are an effective and efficient way to detect and classify diseases in groundnut plants. To reveal the sick image of the groundnut leaf, they employed a variety of DCNN techniques. They developed a deep learning-based convolutional neural network (DCNN) that recognizes visual features automatically without the need for human vision. During each learning phase, the stochastic gradient descent algorithm (SGD) with six convolutional layers is more advantageous for CNN training for a comparable and equivalent collection of random input. Deep learning is highly used to detect leaf diseases, and the DCNN classification is heavily used to classify illnesses. The simulation was run

using the MATLAB R2017b platform. The effectiveness of the suggested DCNN algorithm is assessed using a number of innovative methods. In comparison to the other layers, the sixth DCNN layer has an accuracy rate of 95.28 percent.

Zhang S et al., in [11] have proposed a study that discuss emphasizes the advantages of feature extraction and autonomous learning because they are crucial to both industrial and educational applications. It is widely used in the processing of natural languages, films, voices, and photos. Additionally, it has developed into a focus for study on crop security in agriculture, including the assessment of pest ranges and the identification of plant diseases. Deep learning can increase the objectivity of the features retrieved from plant illnesses, eliminate the negative consequences of arbitrarily picking disease spot characteristics, and speed up the creation of new technologies. This paper explains how deep learning algorithms have developed recently to detect leaf diseases in agricultural plants. In this study, they use deep learning and state-of-the-art imaging techniques to illustrate the existing situation and difficulties in the detection of plant leaf disease. They provided a synopsis of current research on crop leaf disease detection using deep learning as well as an introduction to deep learning in this article.

Haixia Qi et al., in [12] proposed stack ensemble technique to identify the peanut leaf disease automatically. To identify diseases linked to peanut leaf diseases, deep learning models as well as conventional machine-learning techniques were employed. The stacking ensemble method was then used to merge deep learning models with conventional machine learning techniques. The models' performance in fitting the dataset of illnesses brought on by peanut leaves was further assessed using the training methodologies of data augmentation and transfer learning. The findings demonstrated that deep learning models outperformed conventional machine learning techniques in identifying peanut-leaf illnesses. Networks with more layers, such ResNet50 and DenseNet121, performed well in terms of dataset prediction. Using the conventional machine-learning method for the ensemble, the F1 score for data augmentation was 90.50, and the greatest accuracy was 97.59%. The outcomes from the other models demonstrated that, even when a peanut leaf exhibits two diseases simultaneously, the machine-learning approach is still a valuable tool for diagnosing diseases in peanut leaves.

Upesh Nepal et al., in [13] have presented a paper in which they specifically look at finding potential safe landing spots in the event that a critical failure is detected while monitoring the health of the UAV, as well as investigating the feasibility of using object detection methods to spot safe landing spots in the event that the UAV fails during flight. They directly compare the many iterations of the YOLO objection detection approach for the specific application of finding a safe landing place for a UAV that has encountered a problem during flight. They compare the performance of YOLOv3, YOLOv4, and YOLOv5 when trained on the huge DOTA aerial photo dataset.

Melike Sardogan et al., in [14] have presented a paper about the method uses for detection and classification, learning vector quantization (LVQ) and convolutional neural networks (CNN) are used. Early illness diagnosis is essential for agriculture to produce high-quality crops. Numerous diseases, such as late blight, septoria leaf spot, yellow bent leaf, and bacterial spot, have an impact on the quality of the tomato crop. Automatic techniques for identifying plant diseases make it easier to respond when symptoms of leaf disease are observed. The collection includes more than 500 images of tomato leaves displaying four distinct disease symptoms. CNN was created to automate feature extraction and categorization. Color data is commonly used in studies on plant leaf diseases. Three streams based on RGB components are received by their system's filters. The experimental findings demonstrate how effectively the suggested method detects four distinct tomato leaf diseases.

Maheshwaran S et al., in [15] used the convolutional neural network to identify and classify groundnut leaf diseases. The first thing done is to purchase the digital or smartphone photos that were shot in the field. Following this step, image pre-processing methods were employed to retrieve the acquired images in order to get them ready for further research. The CNN algorithm was then supplied with the neural network and all of the previously processed images. Image analysis is used to extract an image that best represents the best extraction. Based on the most appropriate removed features, the testing and training data used for identification are extracted. A new image is finally classified by trained knowledge into its class of syndrome. The accuracy rate for the grayscale dataset and CNN model is 97.5%. The model's overall performance is 95.8% accurate at identifying leaf diseases in plants that produce groundnuts.

Rajnish Rakholia et al., in [16] proposed the recognition of groundnut Leaf disease is identified using a neural network with a progressive scaling technique for better model generalization and a higher identification rate. To create the dataset, the five main categories of groundnut leaves—leaf spot, armyworm effects, wilts, yellow leaf, and healthy leaf—were manually combined. Symptomatic leaves of all the major types were physically removed from the plants and placed on the background of the photographs. An 80:20 split of the generated dataset was made into training and testing groups. The recommended model was first built in 32x32 dimensions before being enlarged up to 4X. The results from several scenarios show that the progressive resizing CNN-based model performed better than the basic CNN-based architecture, and the focal loss function helped to fix the unbalanced dataset problem.

Sivasankaran S et al., in [17] proposed a Comparative CNN Based Deep Learning Model an investigation to identify and categorize the leaf maladies affecting the *Arachis hypogea* (groundnut crop) grown in Tamil Nadu's Villupuram District in semi-arid areas. Performance metrics, correlation metrics, and error metrics were used to compare the proposed experimental CNN-based VGG16 + Adam optimization to a select subset of the top CNN models, including the AlexNet and Inception_V3 models. In terms of performance comparisons, the comparative analysis demonstrated that the CNN-based VGG16 in combination with the Adam optimization has overall accuracy rates that have decisively surpassed all baseline models and ensures very low error rates. This analysis of CNN models has led to a more accurate way of identifying the top five leaf diseases that affect planted groundnut crops.

2.1 GAP OF ANALYSIS

- From the literature review, it is observed that different types of methods like Convolutional Neural Networks (CNN), Regions with Convolutional Neural Networks (R-CNN), Faster Regions with Convolutional Neural Networks (Faster R-CNN) and You only look once (YOLO) are used.
- Comparing the above methods, YOLO has the fast process of CNN, and it is known for its upgradation in terms of AP and FPS.
- The pre trained dataset like DOTA dataset which contains large aerial images, COCO dataset which contains large image collection of 80 classes, ShanghaiTech dataset which contains of 1198 annotated crowd images are used for specific detection.
- When it comes for detection of leaf diseases, there is no pretrained dataset available for the crops that are chose for this study. So, it is necessary to create a custom dataset for the identification of leaf diseases in the image.

CHAPTER 3

EXISTING METHODOLOGY

Misdiagnosis among the various illnesses that affect agricultural crops can result in the abuse of pesticides, the creation of pathogen strains that are resistant to those chemicals, increased input costs, and further outbreaks that have serious negative effects on the economy and the environment. Although computer vision-based models have the potential to increase efficiency, the current disease diagnosis method of human scouting is time-taking and expensive. Additionally, the accuracy of detection is decreased by the wide variation in symptoms caused by the age of damaged tissue, genes involved, and light situations within trees.

3.1 METHODOLOGY

The common existing method for the crop disease detection is the method which involves the diagnosis and detection of crop leaf diseases can be done by visual estimation of plant disease by expertise. The expert must examine the affected leaves in person with the naked eyes to identify the disease affected the leaves and the root cause of the disease in the leaves. The other existing methods to identify the leaf disease are molecular diagnostic technique, serological diagnostic technique, and microbiological diagnostic techniques. In molecular diagnostic technique, the molecules of the affected leaves are sent to the diagnosis laboratory, and they will diagnosis the diseased leaves to identify the disease caused. Serological diagnosis technique involves the diagnosis and testing of the serum fluid taken from the diseased leaves. Microbiological diagnostic technique to identify the pathogens like bacteria, virus or other microorganisms that can cause disease. The deep learning method can also be used to detect the disease caused in the leaves. Orsenigo et al have conducted research on the identification of the disease in the leaves.

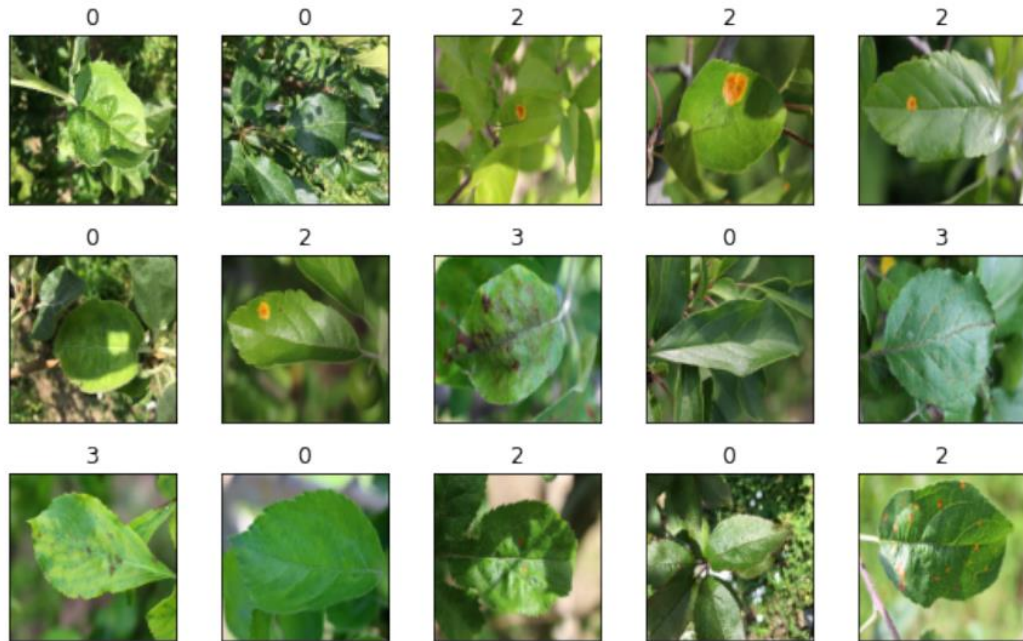


Figure 3.1 Sample of training images

Although rust, scab, or both could affect a leaf that was labelled as having multiple diseases, they treated the classes as mutually exclusive because there is no taxonomy. In theory, the model should be able to distinguish between all four classes since none of them are abstractions of any of the others. The figure 3.1 shows the collection of the images are taken for the creation of dataset.

The class balancing algorithm SMOTE(sampling technique, k neighbours) works as one of the minority classes is considered, a random point is selected, and its first n closest neighbours are identified, the vector between this point and the first chosen point is then drawn after selecting one of those closest neighbours at random, the synthetic point produced by multiplying this vector by an integer between 0 and 1 is then added to the dataset.

After manually reviewing the photos using the Keras Image Data Generator, they discovered that a random planar rotation along with a random horizontal flip was the optimal data augmentation strategy.

The figures 3.2 explains the confusion matrix of the DenseNet121 for the research on the identification of the disease in the leaves conducted by the Orsenigo.

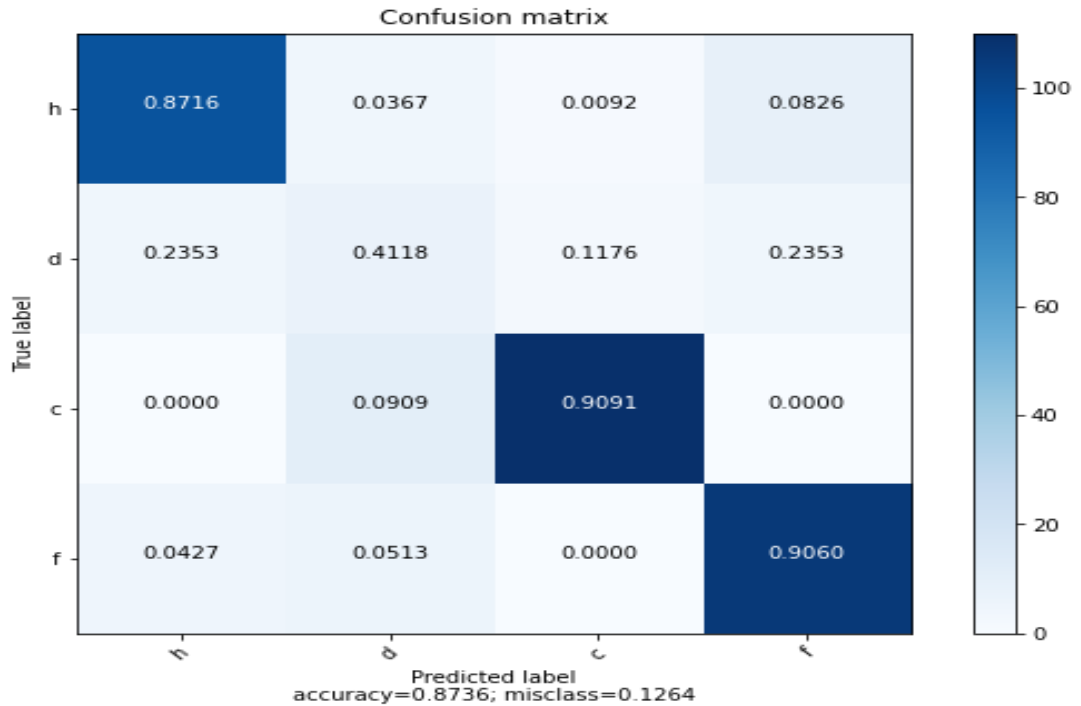


Figure 3.2 Confusion matrix for DenseNet121

The figures 3.3 explains the training history of the DenseNet121 for the research on the identification of the disease in the leaves conducted by the Orsenigo.

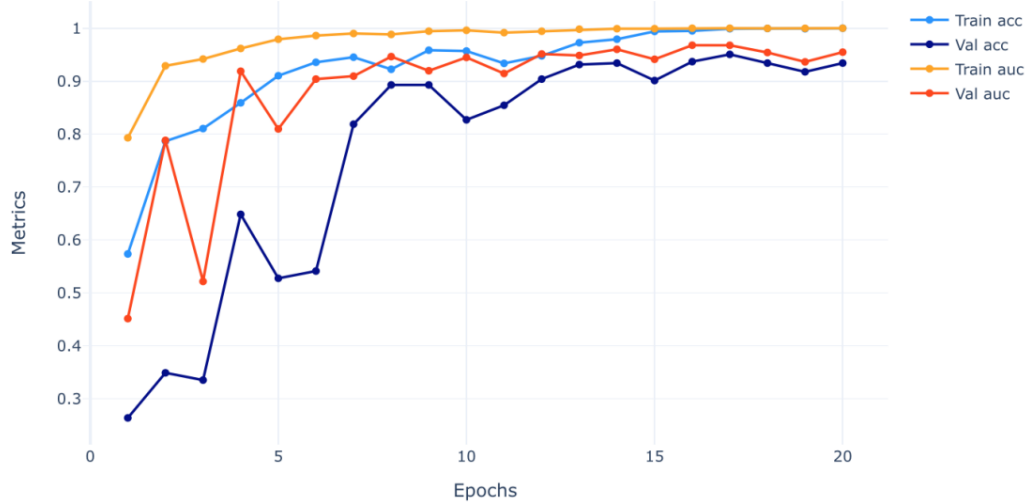


Figure 3.3 Training history of the DenseNet121

3.2 DISADVANTAGES OF EXISTING METHOD

- CNN is much slower because the max pooling process take more time than other achitecture.
- It fails to find the exact position and orientation of the object that needs to be found and it takes much time to classify the position.
- Due to class imbalance and exploding gradient, it cause problem in prediction.
- The accuracy of the above model is 87%
- CNN is two staged networks, so it needs to be look into the image twice. But YOLO is single stage network and it will into the image only once.

CHAPTER 4

PROPOSED METHODOLOGY

Object detection is used to locate and find the object present in an image. It also determines the class of the object present in that image. The object detection process marks a bounding box for all the objects present in the image and gives the objects with the label representing the name of the disease. In this proposed method, YOLO is used for the object detection process. YOLO is an object detection method which uses the deep CNN to find the object present in an image. In 2016, YOLO was developed by Redmon et al. The name represents that this method only looks the image only once. So, it is named as YOLO.

4.1 MATERIALS AND METHODS

4.1.1 NEED FOR YOLO

Object detection can be done using many methods. The single stage networks like YOLO uses a single stage for object localization. The CNN in the YOLO produces the ultimate bounding boxes for the items are created by decoding the predictions of the network or regions throughout the full picture using anchor boxes. It mainly uses only one stage for object identification. This is the main advantage of using the YOLO.

But the double staged networks like R-CNN, Faster R-CNN uses two stages for object localization. In the initial stage, it locates regions or portions of the picture that may contain an item. In the next stage, it determines the object's class inside the zone of proposals. Typically, the double stage network is slower than the single stage networks.

4.1.2 YOLO

YOLO refers to You Only Look Once. A deep CNN is used in the approach known as YOLO to find the object identification. A source image is splitted into $M \times M$ matrices using YOLO. If a certain region in an image is corresponds for object distinguish, the centre of an object will mark into a matrix region. YOLO has structure of organisation made up of twenty-

four convolutional layers, two completely connected layers present in the end and a max pooling layer.

YOLOv2 is a next level of the YOLO, and it was introduced in the year of 2017. It is created to improve the accuracy. But when it comes to the object detections of smaller size YOLOv2 has its complications.

Then they introduced the YOLOv3 in April 2018 and it has few considerable improvements compared to YOLOv2. Later they introduced the YOLOv4 with the improvements compared to the YOLOv4. When it comes to the result speed, YOLOv4 performs well than the other models in a defined manner.

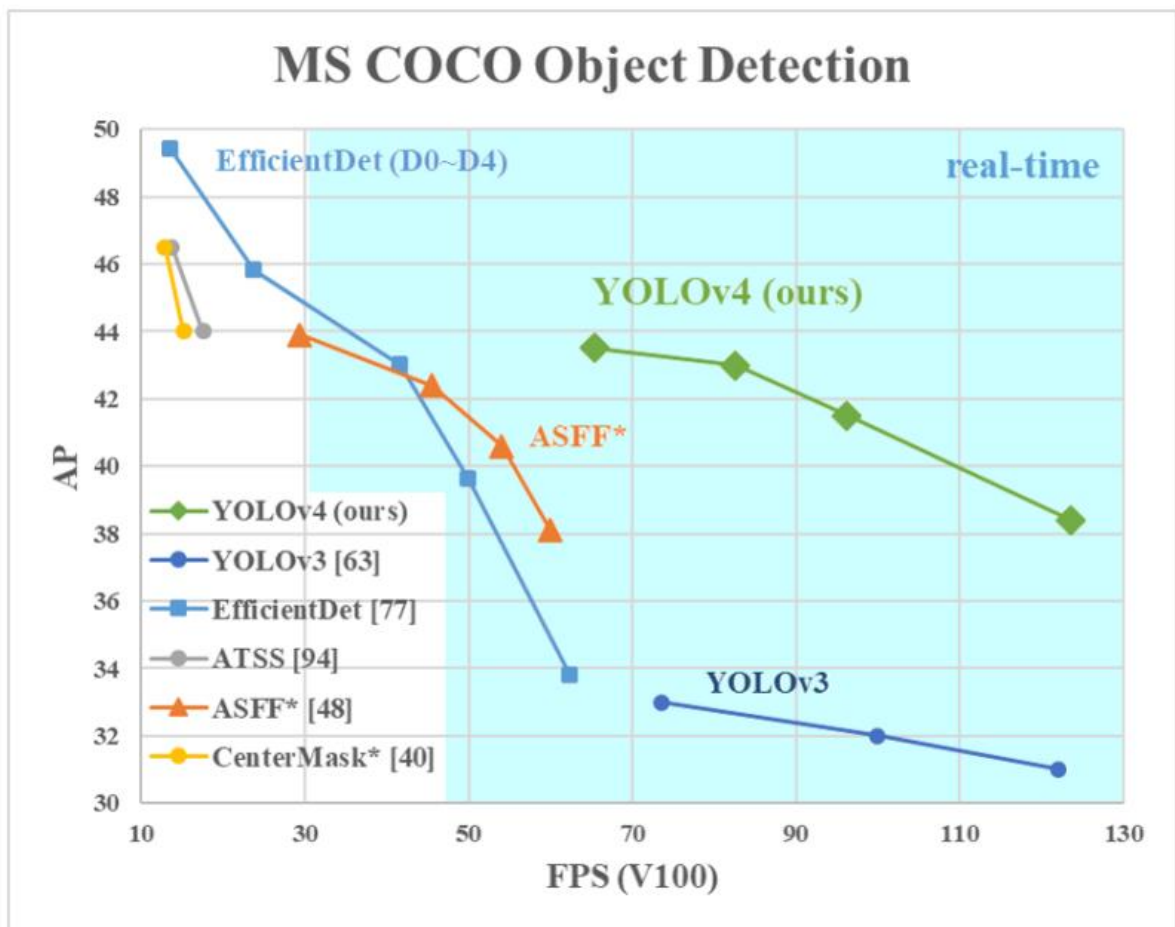


Figure 4.1 Comparison of YOLOv4 vs YOLOv3 with the pretrained COCO dataset

The figure 4.1 will gives the statistical comparison of the performance of YOLOv4 vs YOLOv3 when they tested with the pretrained COCO dataset. The graph is drawn for the FPS

and AP of the YOLOv4 and YOLOv3. YOLOv4 outperforms the YOLOv3 by gives greater values in AP. So YOLOv4 is far better than the YOLOv3.

4.1.3 CONVOLUTIONAL NEURAL NETWORKS

The information flows from CNN inputs to outputs in a single direction, making them straight forward networks. The term "Convolutional networks" refers to artificial neural networks that were inspired by biological processes. They consist of pooling or sub - sampling layers and convolutional layers, which are organized into modules. As in a typical feed-forward neural network, there may be one or more fully linked layers that come after these modules. The process is shown in figure 4.2, where convolutional neural networks are layered and repeatedly followed by pooling layers before feeding into the fully connected layers.

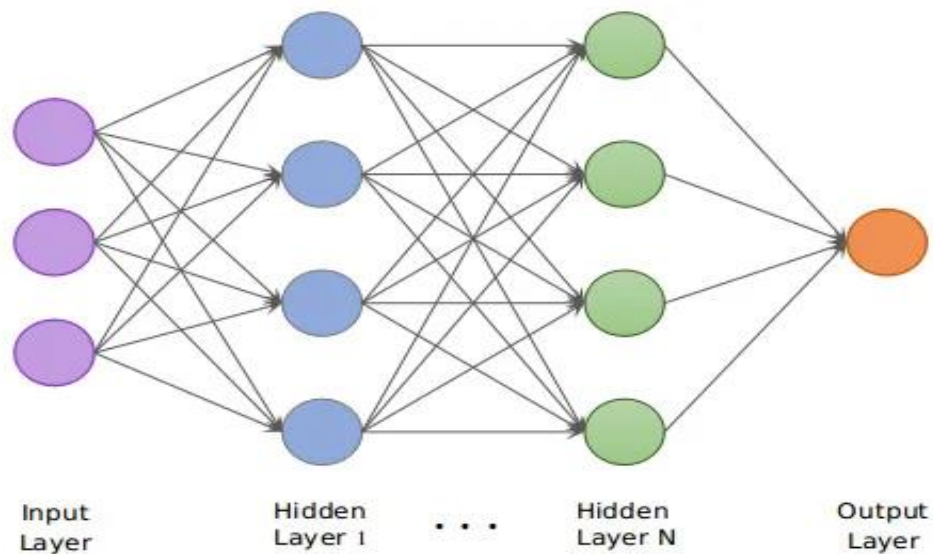


Figure 4.2 CNN Architecture

4.1.4 CONVOLUTIONAL LAYER

The convolutional layer will calculate the multiplication between the input volume in the input region and the weights of the neurons for which they are connected to the local input regions in order to identify the output of those neurons. Convolutional layers learn feature representations from the input images and act as feature extractors. To create a new feature map, the given signals are convolved with the aid of learnt weights, and the resulting outputs are then provided to the non - linear activation function.

The convolutional layer, as its name suggests, is crucial to how the CNNs function. The layers' settings will be centered on the utilization of teachable kernels. Through the output optimization, convolution layer will have the ability to considerably lower the model's complexity. The three hyperparameters of depth, stride, and setting zero-padding can be used to improve these.

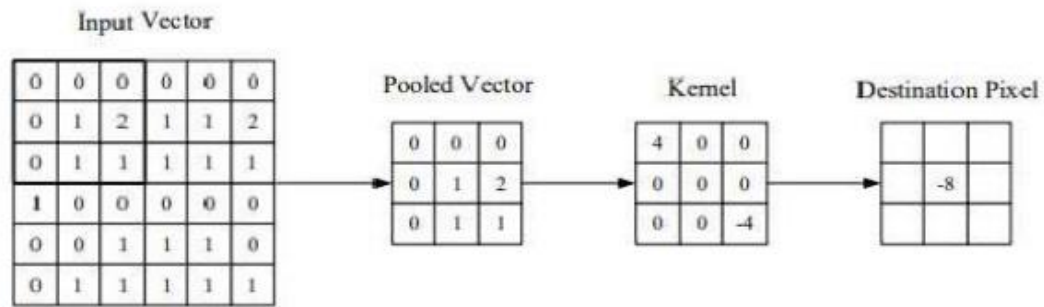


Figure 4.3 Representation of the convolutional layer

The figure 4.3 shows the representation of the convolutional layer in which the input is given to the pooled layer and kernel layer consecutively.

4.1.5 POOLING LAYER

These layers' goal is to lower the feature maps spatial resolution in order to achieve their geometric normalization against input distortions and translations. In order to achieve geometric normalization to incoming distortions and translations, the pooling layers are used to decrease the temporal resolution of the feature maps. In order to transfer the average of all the input parameters from an image's neighborhood to the following layer, max - pooling aggregating layers are used.

The maximum value inside a receptive field is propagated to the following layer using the max pooling aggregator layers. It's crucial to realize that, in addition to max-pooling, CNN architecture may include general pooling layers. These layers are made up of pooling neurons, which are equipped to carry out common tasks like average pooling.

4.1.6 FULLY CONNECTED LAYER

Fully connected layers are present at the network's termination point. These layers carry out high-level reasoning and interpret the feature representations. In order to solve classification issues, it employs the SoftMax operator. The fully linked layer is made up of neurons that are only connected to the neurotransmitters in the two neighboring layers, not to any neurons within those layers. This is comparable to how the neurons are set up in more conventional ANN models.

4.2 DERIVATION FOR FULLY CONNECTED CNN

The convolutional neural network can be achieved with the help of

- Convolutional operation
- Convolutional Layer

In order to achieve the goal, it needs to implement few more things namely,

- Reshape Layer
- Binary cross entropy loss
- Sigmoid activation
- Solve MNIST

The kernel will produce another matrix. The value inside the output matrix is obtained by standing the kernel on Top of the input and by performing a multiplication of adjacent values then summing them up.

For example,

$$\begin{array}{|c|c|c|} \hline 1 & 6 & 2 \\ \hline 5 & 3 & 1 \\ \hline 7 & 0 & 4 \\ \hline \end{array} \times \begin{array}{|c|c|} \hline 1 & 2 \\ \hline -1 & 0 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 8 & 7 \\ \hline 4 & 3 \\ \hline \end{array}$$

$$1. \ 1.1 + 2.6 + (-1.5) + 0.3 = 8$$

$$2. \ 1.6 + 2.2 + (-1.3) + 0.1 = 7$$

Size of the output can be easily computed as the size of the input as equation 4.1.

$$N = I - K + 1 \quad (4.1)$$

This is actually a cross correlation, and the real convolution is that same operation as equation 4.2.

$$\text{Conv}(I, K) = I * \text{rot}180(k) \quad (4.2)$$

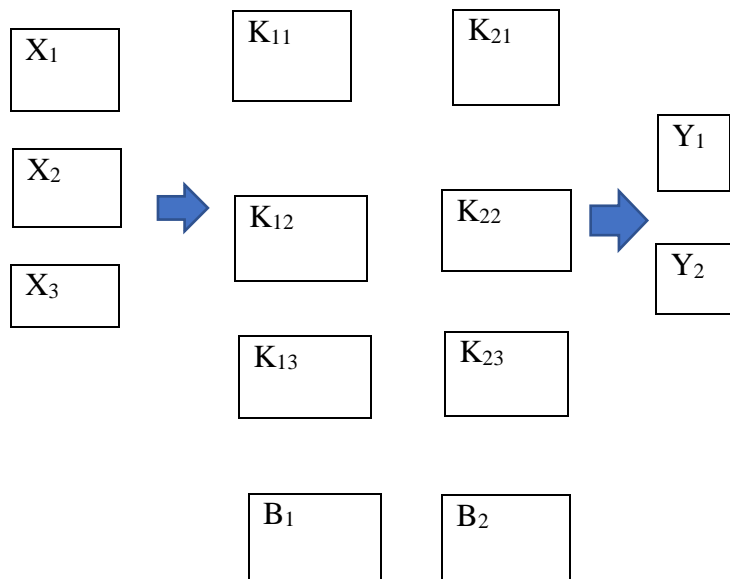
i.e., Cross convolution of I and rotated version of k.

Convolutional layers take three-dimensional block of data as input, and it is given to the kernel.

INPUT			KERNEL	
X ₁₁	X ₁₂	X ₁₃	K ¹ ₁₁	K ¹ ₁₂
X ₂₁	X ₂₂	X ₂₃	K ¹ ₂₁	K ¹ ₂₂
X ₃₁	X ₃₂	X ₃₃		

X ² ₁₁	X ² ₁₂	X ² ₁₃	K ² ₁₁	K ² ₁₂
X ² ₂₁	X ² ₂₂	X ² ₂₃	K ² ₂₁	K ² ₂₂
X ² ₃₁	X ² ₃₂	X ² ₃₃		

X ³ ₁₁	X ³ ₁₂	X ³ ₁₃	K ³ ₁₁	K ³ ₁₂
X ³ ₂₁	X ³ ₂₂	X ³ ₂₃	K ³ ₂₁	K ³ ₂₂
X ³ ₃₁	X ³ ₃₂	X ³ ₃₃		



$$Y_1 = B_1 + X_1 * K_{11} + X_2 * K_{12} + X_3 * K_{13}$$

$$Y_2 = B_2 + X_1 * K_{21} + X_2 * K_{22} + X_3 * K_{23}$$

For depth d is given by the equation 4.3,

$$Y_d = B_d + X_1 * K_{d1} + X_2 * K_{d2} + X_3 * K_{d3} \quad (4.3)$$

Forward propagations is given by the equation 4.4,

$$Y_i = B_i + \sum_{j=1}^n X_j * K_{ij} \quad \text{Since } i = 1, \dots, d \quad (4.4)$$

Expand the above one,

$$\begin{aligned} Y_1 &= B_1 + X_1 * K_{11} + \dots + X_n * K_{1n} \\ Y_2 &= B_2 + X_1 * K_{21} + \dots + X_n * K_{2n} \\ &\vdots \\ Y_d &= B_d + X_1 * K_{d1} + \dots + X_n * K_{dn} \end{aligned} \quad (4.5)$$

Where * - represents cross correlation.

$$\begin{bmatrix} Y1 \\ Y2 \\ \vdots \\ Yd \end{bmatrix} = \begin{bmatrix} B1 \\ B2 \\ \vdots \\ Bd \end{bmatrix} + \begin{bmatrix} K11 & K12 & \dots & K1n \\ K21 & K22 & \dots & K2n \\ \vdots & \vdots & \ddots & \vdots \\ Kd1 & Kd2 & \dots & Kdn \end{bmatrix}$$

Where $Y = B + K$.

In Forward propagation equation, $Y = B + K \cdot X$ is the exact equation as 4.5, it had for the dense layer that instead of having matrices of scalars. The matrices of matrices and instead of a regular dot product have that cross correlated dot product.

For example,

If each of these matrices were one dimensional i.e each matrix contains a single element then the cross correlation between two matrices containing one element is equal to one dimensional matrix contain product. It shows that the new operator is being a simple and regular dot product. So, from that analyse is the convolutional layer is nothing but a generalisation of a dense layer.

In order to update the kernel and biases are needed to compute their gradients, The derivatives of e the error of the neural network and it needs to compute two things. On one hand the derivative of e with respect to triangle parameters of the layer.

$$\partial E / \partial Y_i = \partial E / \partial K_{ij}, \partial E / \partial B_i \quad \text{where } \partial E / \partial K_{ij} \text{ is Kernel, } \partial E / \partial B \text{ is bias} \quad (4.6)$$

On other hand it needs to compute the derivative of e with respect to the input of the laser is given by the equation 4.6 so that the previous laser can take it as the derivative of e with respect to its own output and perform the same operations.

Now derivate the e with respect to K is given by the equation 4.7,

$$Y_i = B_i + \sum_{j=1}^n X_j * K_{ij} \quad (4.7)$$

This is the equation for the forward propagation if it exempts the sum as equation 4.8,

$$Y_i = B_i + X_i * K_{i1} + \dots + X_n * K_{in} \quad (4.8)$$

Now break this equation in half for looking simplified version is given by the equation 4.9,

$$Y_i = B_i + X_1 * K_{i1} \quad (4.9)$$

Y ₁₁	Y ₁₂
Y ₂₁	Y ₂₂

=

B ₁₁	B ₁₂
B ₂₁	B ₂₂

+

X ₁₁	X ₁₂	X ₁₃
X ₂₁	X ₂₂	X ₂₃
X ₃₁	X ₃₂	X ₃₃

×

K ₁₁	K ₁₂
K ₂₁	K ₂₂

Here Y equals to B plus X correlated with K.

The output of each variable is

$$Y_{11} = b_{11} + K_{11}X_{11} + K_{12}X_{12} + K_{21}X_{21} + K_{22}X_{22}$$

$$Y_{12} = b_{12} + K_{11}X_{12} + K_{12}X_{13} + K_{21}X_{22} + K_{22}X_{23}$$

$$Y_{21} = b_{22} + K_{11}X_{22} + K_{12}X_{23} + K_{21}X_{31} + K_{22}X_{32}$$

$$Y_{22} = b_{22} + K_{11}X_{22} + K_{12}X_{23} + K_{21}X_{32} + K_{22}X_{23} \quad (4.10)$$

$$\frac{\partial E}{\partial Y} = \begin{array}{|c|c|} \hline \frac{\partial E}{\partial Y_{11}} & \frac{\partial E}{\partial Y_{12}} \\ \hline \frac{\partial E}{\partial Y_{21}} & \frac{\partial E}{\partial Y_{22}} \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|} \hline \frac{\partial E}{\partial K_{11}} & \frac{\partial E}{\partial K_{12}} \\ \hline \frac{\partial E}{\partial K_{21}} & \frac{\partial E}{\partial K_{22}} \\ \hline \end{array}$$

$$\frac{\partial E}{\partial K_{11}} = \frac{\partial E}{\partial Y_{11}} \frac{\partial Y_{11}}{\partial K_{11}} + \frac{\partial E}{\partial Y_{21}} \frac{\partial Y_{21}}{\partial K_{11}} + \frac{\partial E}{\partial Y_{22}} \frac{\partial Y_{22}}{\partial K_{11}} \quad (4.11)$$

The terms that are looking for are the derivatives of X with respect to K₁₁. Looking for the forward propagation equations, see that K₁₁ appears inside equation 4.11 and the derivatives are looking for are simply the factor multiplying K₁₁ in each equation 4.12.

$$\frac{\partial E}{\partial K_{11}} = \frac{\partial E}{\partial Y_{11}} X_{11} + \frac{\partial E}{\partial Y_{12}} X_{12} + \frac{\partial E}{\partial Y_{21}} X_{21} \quad (4.12)$$

The same will do for all other elements of the kernel set from 4.13 to 4.15,

$$\frac{\partial E}{\partial K_{11}} = \frac{\partial E}{\partial Y_{11}} X_{12} + \frac{\partial E}{\partial Y_{12}} X_{13} + \frac{\partial E}{\partial Y_{21}} X_{22} + \frac{\partial E}{\partial Y_{22}} X_{23} \quad (4.13)$$

$$\frac{\partial E}{\partial K_{21}} = \frac{\partial E}{\partial Y_{11}} X_{21} + \frac{\partial E}{\partial Y_{12}} X_{22} + \frac{\partial E}{\partial Y_{21}} X_{31} + \frac{\partial E}{\partial Y_{22}} X_{32} \quad (4.14)$$

$$\frac{\partial E}{\partial K_{22}} = \frac{\partial E}{\partial Y_{11}} X_{22} + \frac{\partial E}{\partial Y_{12}} X_{23} + \frac{\partial E}{\partial Y_{21}} X_{32} + \frac{\partial E}{\partial Y_{22}} X_{33} \quad (4.15)$$

The way to complete the four derivatives in a single formula. The result is obviously an operation between the input matrix and the matrix contains the derivatives of e with respect to the output e.

$$\begin{array}{|c|c|} \hline \frac{\partial E}{\partial K_{11}} & \frac{\partial E}{\partial K_{12}} \\ \hline \frac{\partial E}{\partial K_{21}} & \frac{\partial E}{\partial K_{22}} \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline X_{11} & X_{12} & X_{13} \\ \hline X_{21} & X_{22} & X_{23} \\ \hline X_{31} & X_{32} & X_{33} \\ \hline \end{array} \times \begin{array}{|c|c|} \hline \frac{\partial E}{\partial Y_{11}} & \frac{\partial E}{\partial Y_{12}} \\ \hline \frac{\partial E}{\partial Y_{21}} & \frac{\partial E}{\partial Y_{22}} \\ \hline \end{array}$$

The cross correction between input matrix and output gradient in the equation 4.16,

$$\begin{aligned} \frac{\partial E}{\partial K} &= X * \frac{\partial E}{\partial Y} \\ Y = B + X * K &\Rightarrow \frac{\partial E}{\partial K} = X * \frac{\partial E}{\partial Y} \end{aligned} \quad (4.16)$$

With the help of this equation, compute the kernel gradient and it is given in the equation 4.17,

$$Y_i = B_i + \sum_{j=1}^n X_j * K_{ij}, \quad i = 1, \dots, d \quad (4.17)$$

Expanded form of kernel gradient is shown in the equation 4.18,

$$\begin{aligned} Y_1 &= B_1 + X_1 * K_{11} + \dots + X_n * K_{1n} \\ Y_2 &= B_2 + X_1 * K_{21} + \dots + X_n * K_{2n} \\ Y_d &= B_d + X_1 * K_{d1} + \dots + X_n * K_{dn} \\ \partial E / \partial K_{ij} &= X_j * \partial E / \partial Y_i \end{aligned} \quad (4.18)$$

The next step is bias gradient and the same as what it takes in the kernel representation of the matrix form. Give the derivative of e with respect to the output and needs to calculate the derivative of e with respect to the bias is shown in 4.19 and 4.20.

$$\partial E / \partial Y = \begin{array}{|c|c|} \hline \partial E / \partial Y_{11} & \partial E / \partial Y_{12} \\ \hline \partial E / \partial Y_{21} & \partial E / \partial Y_{22} \\ \hline \end{array} \quad \longrightarrow \quad \begin{array}{|c|c|} \hline \partial E / \partial b_{11} & \partial E / \partial b_{12} \\ \hline \partial E / \partial b_{21} & \partial E / \partial b_{22} \\ \hline \end{array}$$

$$\begin{aligned} \partial E / \partial b_{11} &= \partial E / \partial Y_{11} \partial Y_{11} / \partial b_{11} + \partial E / \partial Y_{12} \partial Y_{12} / \partial b_{11} + \\ &\quad \partial E / \partial Y_{21} \partial Y_{21} / \partial b_{11} + \partial E / \partial Y_{22} \partial Y_{22} / \partial b_{11} \end{aligned} \quad (4.19)$$

$$\begin{aligned} \partial E / \partial b_{11} &= \partial E / \partial Y_{11} (1) + 0 + 0 + 0 + 0 \\ \partial E / \partial b_{11} &= \partial E / \partial Y_{11} \\ \partial E / \partial b_{12} &= \partial E / \partial Y_{12} \\ \partial E / \partial Y_{22} &= \partial E / \partial Y_{22} \end{aligned} \quad (4.20)$$

The bias equation 4.21 is equal to the output gradient,

$$\begin{aligned} \partial E / \partial B &= \partial E / \partial Y \\ Y &= B + X * K \Rightarrow \partial E / \partial B = \partial E / \partial Y \end{aligned} \quad (4.21)$$

For the simplified equation of the above one is given in 4.22,

$$Y_1 = B_1 + X_1 * K_{11} + \dots + X_n * K_{n1}$$

$$Y_2 = B_2 + X_1 * K_{21} + \dots + X_n * K_{2n}$$

.

.

.

.

.

$$Y_d = B_d + X_1 * K_{d1} + \dots + X_n * K_{dn} \quad (4.22)$$

$$\partial E / \partial B_1 = \partial E / \partial Y_1 \quad \rightarrow \quad \partial E / \partial B_i = \partial E / \partial Y_i$$

$$\partial E / \partial B_i = \partial E / \partial Y_1 \quad (4.23)$$

Now update the parameters using gradient descent. This layer gives the input gradient in order to give it to the previous layer. From the equation 4.23, calculate the derivative of e with respect to the input.

$$\partial E / \partial Y =$$

$\partial E / \partial Y_{11}$	$\partial E / \partial Y_{12}$
$\partial E / \partial Y_{21}$	$\partial E / \partial Y_{22}$

 \rightarrow

$\partial E / \partial X_{11}$	$\partial E / \partial X_{12}$	$\partial E / \partial X_{13}$
$\partial E / \partial X_{21}$	$\partial E / \partial X_{22}$	$\partial E / \partial X_{23}$
$\partial E / \partial X_{31}$	$\partial E / \partial X_{32}$	$\partial E / \partial X_{33}$

$$\begin{aligned} \partial E / \partial X_{11} = & \partial E / \partial Y_{11} \partial Y_{11} / \partial X_{11} + \partial E / \partial Y_{12} \partial Y_{12} / \partial X_{11} + \\ & \partial E / \partial Y_{21} \partial Y_{21} / \partial X_{11} + \partial E / \partial Y_{22} \partial Y_{22} / \partial X_{11} \end{aligned} \quad (4.24)$$

Expand the derivatives of Y with respect to X_{11} it turns appears in the first equation in equation 4.24,

$$\partial E / \partial X_{11} = \partial E / \partial Y_{11} (1) + 0 + 0 + 0 + 0$$

$$\partial E / \partial X_{11} = \partial E / \partial Y_{11} K_{11}$$

$$\partial E / \partial X_{12} = \partial E / \partial Y_{11} K_{12} + \partial E / \partial Y_{12} K_{11}$$

$$\partial E / \partial X_{13} = \partial E / \partial Y_{11} K_{12}$$

$$\partial E / \partial X_{21} = \partial E / \partial Y_{11} K_{22} + \partial E / \partial Y_{12} K_{11}$$

$$\partial E / \partial X_{22} = \partial E / \partial Y_{11} K_{22} + \partial E / \partial Y_{12} K_{21} + \partial E / \partial Y_{21} K_{12} + \partial E / \partial Y_{22} K_{11}$$

$$\partial E / \partial X_{23} = \partial E / \partial Y_{12} K_{22} + \partial E / \partial Y_{22} K_{12}$$

$$\partial E / \partial X_{31} = \partial E / \partial Y_{21} K_{21}$$

$$\partial E / \partial X_{32} = \partial E / \partial Y_{21} K_{22} + \partial E / \partial Y_{22} K_{21} \quad (4.25)$$

It is a full correlation except that the kernel has been rotated by 180 degree and the input gradient is equal to the output gradient fully cross-correlated with the 180-degree rotated kernel and its relation is given in 4.26.

$$\partial E / \partial X = \partial E / \partial Y * \text{rot180}(k)$$

$$\partial E / \partial X = \partial E / \partial Y * K$$

$$Y = B + X * K \Rightarrow \partial E / \partial X \Rightarrow \partial E / \partial Y * K$$

$$\partial E / \partial X_j = \sum_{i=1}^n \partial E / \partial Y_i K_{ij} \quad (4.26)$$

Finally, the three equations for backward propagation is shown in 4.27,

$$\partial E / \partial K_{ij} = X_j * \partial E / \partial Y_i$$

$$\partial E / \partial B_i = \partial E / \partial Y_i$$

$$\partial E / \partial X_j = \sum_{i=1}^n \partial E / \partial Y_i K_{ij} \quad (4.27)$$

The main goal is to detect the image and what is present in the image, but the code is not super optimized and currently it is bounded to the computational power of the CPU are bounded to the computational power of the CPU and for these reasons it might take a while to train a neural network on to whole dataset which contains 60000 images.

Reshape layer:

Reshape layer is needed because the output of the convolutional layer is a 3d block and typically at the end of a network uses dense layers which take in a column vector as input therefore in need a mechanism that reshape the data.

Binary cross entropy loss:

They decide to make a classification on two digits and this loss is pretty good for that task.

Sigmoid activation:

The binary cross entropy uses the logarithm function, and the logarithm function cannot take negative inputs, therefore implement the sigmoid activation.

Binary cross entropy:

The given vector containing the desired output of the neural network and each of the value inside the vector must be either zero or 1 then give the actual output of the neural network.

$$Y^* = \begin{bmatrix} Y1^* \\ Y2^* \\ Y3^* \end{bmatrix} \quad Y = \begin{bmatrix} Y1 \\ Y2 \\ Yi \end{bmatrix}$$

The binary cross entropy loss is shown in 4.28,

$$E = -1/n \sum_{i=1}^n Y_i^* \log(y_i) + (1 - Y_i^*) \log(1 - Y_i) \quad (4.28)$$

The goal is to compute the derivate of e with respect to the output which is what give to backward and the last layer of the neural network.

$$\partial E / \partial Y = \begin{bmatrix} \partial E / \partial Y_1 \\ \partial E / \partial Y_2 \\ \partial E / \partial Y_i \end{bmatrix}$$

The $\partial E / \partial Y_1$ calculated from the matrix and it is shown in 4.29,

$$\partial E / \partial Y_1 = \partial / \partial Y_1 (-1/n \sum_{i=1}^n Y_i^* \log(Y_i) + (1 - Y_i^*) \log(1 - Y_i))$$

$$= \partial / \partial Y_1 (-1/n (Y_1 \log(y_1) + (1 - Y_1^*) \log(1 - Y_1)))$$

$$= -1/n (Y_1^* / Y_1 - 1 - Y_1^* / 1 - Y_1)$$

$$= 1/n (1 - Y_1^* / 1 - Y_1 - Y_1^* / Y_1)$$

$$\partial E / \partial Y_1 = 1/n (1 - Y_1^* / 1 - Y_1 - Y_1^* / Y_1)$$

$$\partial E / \partial Y_i = 1/n (1 - Y_i^* / 1 - Y_i - Y_i^* / Y_i) \quad (4.29)$$

Sigmoid activation:

It is defined as one over one plus the exponential of minus x. It is bounded between leaves. The figure 4.4 will show the sigmoid activation bounded between the leaves represented in the form of graph.

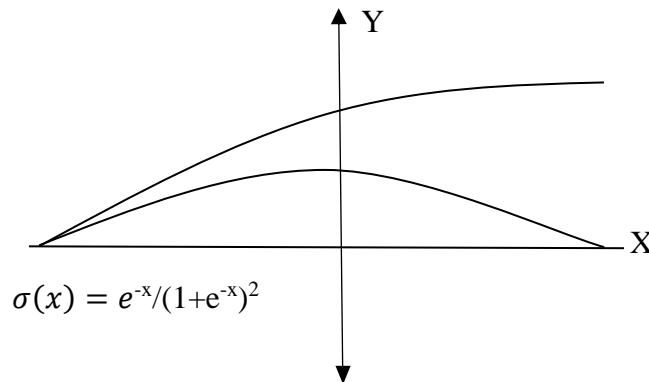


Figure 4.4 Sigmoid Activation

$$\begin{aligned}\sigma(x) &= e^{-x}/(1+e^{-x})^2 \\ &= \sigma(x)(1-\sigma(x))\end{aligned}$$

The derivative can be explained in terms of the function sigmoid itself in 4.30,

$$\sigma(x) = 1/1+e^{-x}$$

$$\sigma(x) = \sigma(x) * (1 - \sigma(x)). \quad (4.30)$$

4.3 BLOCK DIAGRAM OF THE PROPOSED METHOD

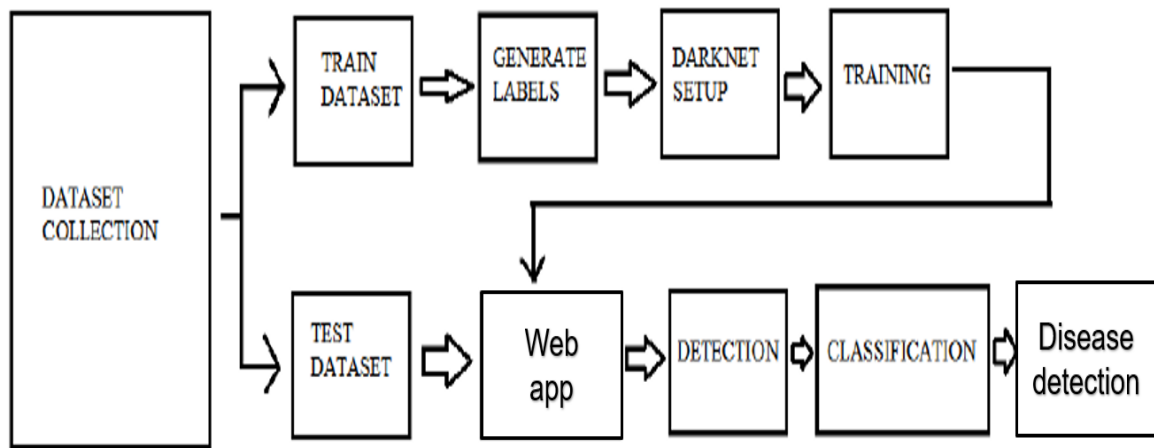


Figure 4.5 Block diagram of the proposed method

The figure 4.5 shows the working flow of the proposed method and it starts with the creation of the dataset. For the creation of the dataset, it is necessary to have the collection of the images with the class of the object has to be labelled.

After that the darknet is needs to be setup which helps for the training of the dataset. Upon completion of the training, the testing will happen. In that testing of the dataset, it will detect, classify, and counts how many objects present in the input image.

4.4 PRE TRAINED MODEL

Pre trained model is easy to use and it has its own advantages and disadvantages. It is easily download from the internet and it can be used normally. It does not require the training period because it is already trained by someone. So that it does not need to spend hours and days for training the model. The dataset can be created for single class or multiple classes. The dataset created with single class have to trained with large number of collection of images. Normally a dataset was created with hundreds of images.

The pre trained dataset have many number of classes in it. But the only disadvantage of the pre trained model is it may or may not have the desired class in it. The custom trained model is introduced has been to resolve the problem in proposed method.

4.5 CUSTOM TRAINED MODEL

To solve the problem in pre trained model, custom trained model is introduced. The custom trained model is used based on the requirements. If the dataset does not contain the desired class of the object, have to create the custom trained model. It requires the large collection of images of the object which is present in it. Using the images able to create the desired dataset. Divide the collection of images in to two groups. The first group is used for the training of the dataset. For the training of the dataset or model, use the 80% of the images from the collection of images. The another group is used for the testing of the dataset. For the testing of the dataset, use the remaining 20% of the collection of images.

For the creation of the custom trained dataset, it requires very high computational resources. Because downloading the weights from the internet is very easy. When it comes for the training in the normal CPU, it requires high computational power. But training with the GPU can improve the training time dramatically approximately more than 500 times than the training with the noraml CPU. GPU actually takes the load of processing the frames from CPU to dedicated processing unit. The main difference between the GPU with CPU is GPU has multiple cores but CPU has limited number of cores.

But GPU is very expensive compared to the CPU. For the decent configuration of the GPU, the price will go higher. By considering this disadvantage of the GPU, the use of the google

colab is recommended. Google is renting tesla K80 GPU which is one among the fastest GPU available right now. Also they can renting it for free for the users. So the users can access it in online and just upload the thing that they require.

4.6 DARKNET IN YOLO

Darknet is acting as the framework of the YOLO. It is maintained by Alexy's github repository. He is the one of the authors of YOLO. He provided the darknet of the YOLO for the users at a free of cost. For using the darknet, it has to get the clone of the darknet framework.

- Get the clone of the darknet from the github repository.
- Download and install the darknet from the repository.
- Get the source code.

Once the above process is completed, darknet framework is cloned. Next move to the process of modifying the make file to include the GPU CUDNN and OPENCV.

Inside the darknet folder, choose the file named make file. Make the changes inside the file like the format and the figure 4.6 shows the changes in the make file.

GPU=1

CUDNN=1

OPENCV=1

 Makefile - Notepad

```
GPU=1
CUDNN=1
CUDNN_HALF=0
OPENCV=1
AVX=0
OPENMP=0
LIBSO=0
ZED_CAMERA=0
ZED_CAMERA_v2_8=0
```

Figure 4.6 Make file

For testing, download the weights for the testing the darknet build and download the pre trained YOLO weights and carry out the detection with the test image that the file name is yolov4.weights.

The next step is copy the yolov4 configuration file to darknet and edit the yolov4 configuration file. During editing of the configuration file, make the changes for the desired configuration. The changes in the configuration file can be mentioned below.

#line no 2,3

Batch= 64

Subdivisions= 64

Where batch is the number of images per iteration and the subdivisions are the number of pieces of the batch is broken for GPU memory. If the value for the subdivisions is smaller, the processing will be faster. But in order to do the faster processing, the GPU will also be a faster GPU. Larger subdivisions require less memory.

#line no 8,9

Width= 416,

Height= 416

#line 20

Max_batches= 36000

Where Max_batches is used to represent the maximum number of iterations for which the network should be trained (number of classes * 2000).

#line 22

steps=1800

```

[net]
batch=64
subdivisions=64
# Training
#width=512
#height=512
width=416
height=416
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.0013
burn_in=1000
max_batches = 60000
policy=steps
steps=1800
scales=.1,.1

#cutmix=1
mosaic=1

#:104x104 54:52x52 85:26x26 104:13x13 for 416

[convolutional]
batch_normalize=1
filters=32
size=3|
stride=1
pad=1
activation=mish

```

Figure 4.7 Configuration file

The figure 4.7 shows what are the changes have to be made in the configuration file and by changing the configuration the dataset can be created for the desired requirements.

#line 961,968

Filters= 69

Classes= 18

Filters= (classes+5) * 3= 69

Classes= 18

```
#line 1049,1056
```

```
Filters= 69
```

```
Classes= 18
```

```
#line 1137,1144
```

```
Filters= 69
```

```
Classes= 18
```

```
pad=1
filters=256
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=18
activation=linear

[yolo]
mask = 0,1,2
anchors = 12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146, 142, 110, 192, 243, 4
classes=1
num=9
```

Figure 4.8 Configuration file

After making the above changes in the configuration file which is shown in the above figure 4.8 and after that save the file named as crop_yolov4.

4.7 DATA COLLECTION

The next step of the process would be the collection of data will be primarily in the form of images. In that step, collect the images contains the crops in it and annotate the image. After that split the collection of images into two groups, one for training and another group for testing and also do some preparation for the data.

Inside the darknet, go to the folder named code and in that create a new folder in the name of crop_data and inside create another folder named crop_images. Paste the collection of images in the crop_images folder. It has to contain a minimum number of 900 images for a single class in a dataset. It is recommended to have a number of 17000 sample of images.

The collection of the images should be unique as much as possible. If the image uniqueness in the collection increases, it increases the accuracy of the trained dataset. After that name the images as 1 to n for ordering for the convenient of the user where n is the total number of images taken for the dataset creation.

4.8 IMAGE ANNOTATION

After the collection of the images for the custom dataset, do the labelling of the images. It is also known as image annotation. In this step, the system have to find where the object present in the image. LabelImg software is used to mark the position of the object in the image where the model can find the object is looking for. By using the software, the software will identify the position of the object in the image. In starting of the process, the model does not have the knowledge about the presence of the object in the image. So, by using the above steps teach the model to look for objects in the image. Once it learns by practicing by doing more and more iterations, it will become more expert than the humans. To make the model to become expert use the tool named LabelImg software.

LabelImg software is used to tell the system where to look for that particular object in the images. To use the software, adhere to the steps listed below.

- Open the LabelImg software
- Click the open dir
- Choose the image folder and after that the image automatically picked up all those images listed
- Change the format to YOLO
- Click create rectangular box and label the legitimate boxes in the image and the purpose of labelling is to name the class of the object in the image.

After the completion of the labeling of all images, it will automatically creates the separate text file for the each of the image present in the crop_images folder and the values are stored.



Figure 4.9 Labeling the images using LabelImg software

The figure 4.9 shows the user interface of the LabelImg software and how the objects can be labelled and various options available in the software. The text file contains the value representing the desired class of the object and the vertices of the box covering the object. If there are more object present in the same image, it also contains the values for the all the objects.

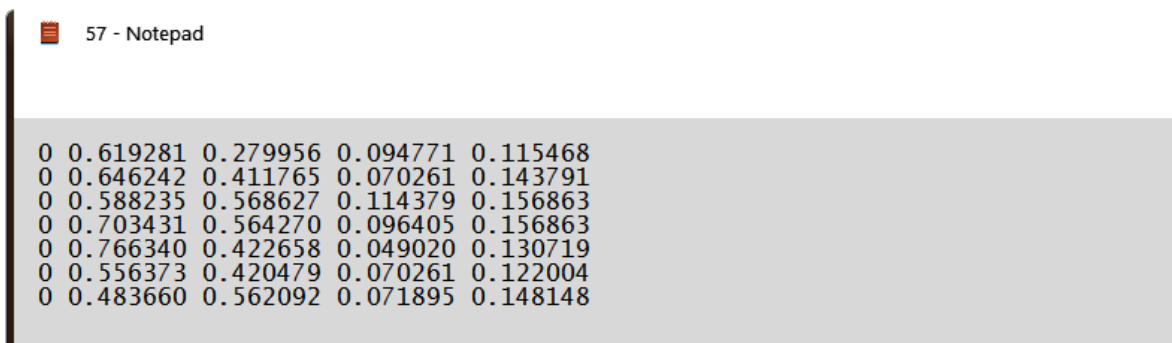


Figure 4.10 Text file generated for an image

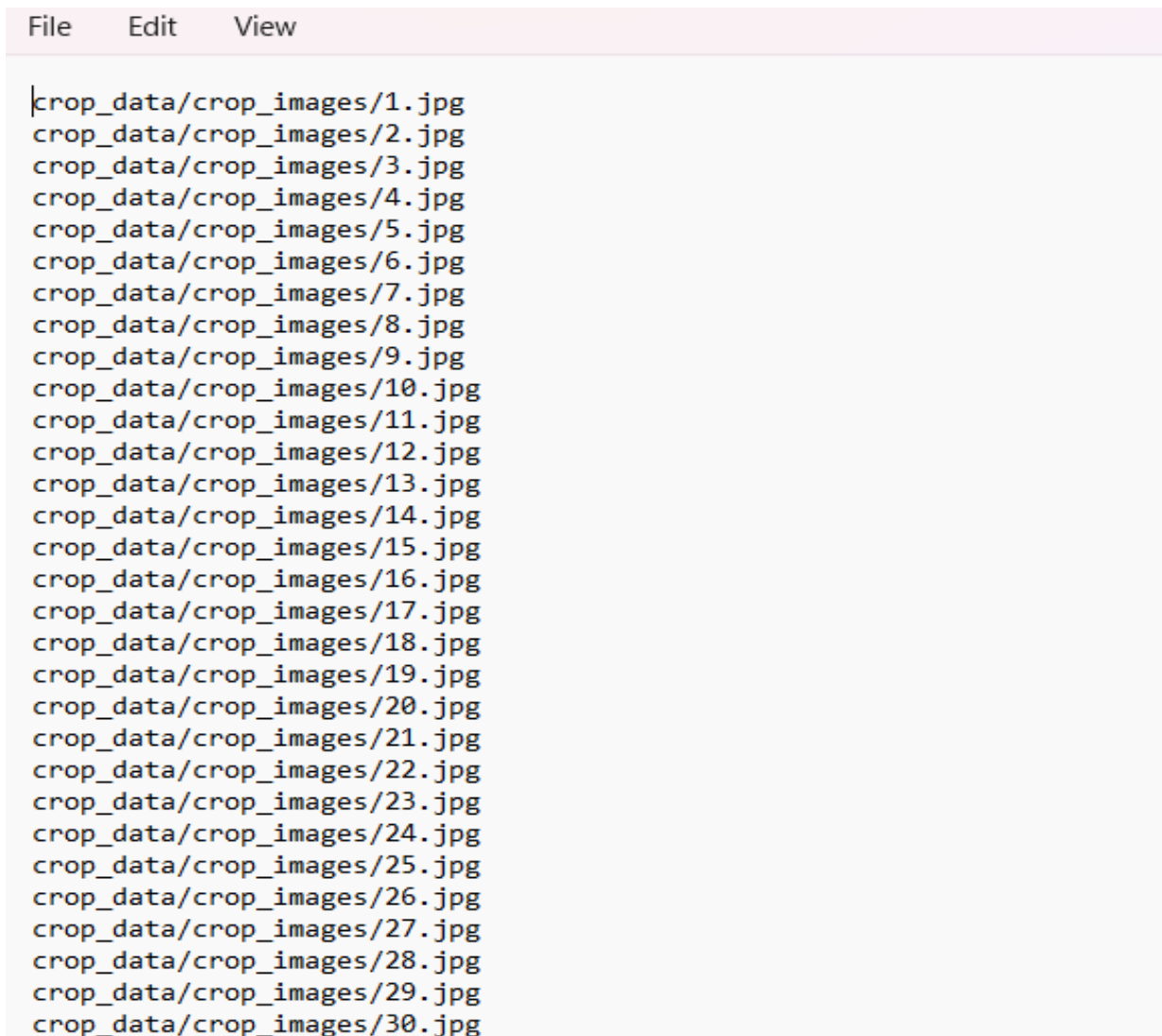
In the figure 4.10, it contains the first value as 0 and it represents the class that respective leaf disease belongs to class. The second value represents the starting x point and the third value represents the starting y point of the box converging the leaf in the image. The fourth and fifth value represents the ending x and y point.

Create two text file named `crop_training.txt` and `crop_testing.txt` in the `crop_data` folder in the `darknet` folder. Arrange the 80% of the image for the training and 20% of the image for the testing. The file path of the two text file is given below.

FILE PATH

1. `crop_data/crop_training.txt`
2. `crop_data/crop_testing.txt`

In the `crop_training.txt` file, place all the images want to train the dataset is saved with the path and it is shown in the figure 4.11.



```
File Edit View
|crop_data/crop_images/1.jpg
crop_data/crop_images/2.jpg
crop_data/crop_images/3.jpg
crop_data/crop_images/4.jpg
crop_data/crop_images/5.jpg
crop_data/crop_images/6.jpg
crop_data/crop_images/7.jpg
crop_data/crop_images/8.jpg
crop_data/crop_images/9.jpg
crop_data/crop_images/10.jpg
crop_data/crop_images/11.jpg
crop_data/crop_images/12.jpg
crop_data/crop_images/13.jpg
crop_data/crop_images/14.jpg
crop_data/crop_images/15.jpg
crop_data/crop_images/16.jpg
crop_data/crop_images/17.jpg
crop_data/crop_images/18.jpg
crop_data/crop_images/19.jpg
crop_data/crop_images/20.jpg
crop_data/crop_images/21.jpg
crop_data/crop_images/22.jpg
crop_data/crop_images/23.jpg
crop_data/crop_images/24.jpg
crop_data/crop_images/25.jpg
crop_data/crop_images/26.jpg
crop_data/crop_images/27.jpg
crop_data/crop_images/28.jpg
crop_data/crop_images/29.jpg
crop_data/crop_images/30.jpg
```

Figure 4.11 Text file containing the file path of the images want to train

Similarly in the crop_testing.txt file, place all the images want to test the dataset is saved with the path and it is shown in the figure 4.12.

```
crop_data/crop_images/83.jpg
crop_data/crop_images/84.jpg
crop_data/crop_images/85.jpg
crop_data/crop_images/86.jpg
crop_data/crop_images/87.jpg
crop_data/crop_images/88.jpg
crop_data/crop_images/89.jpg
crop_data/crop_images/90.jpg
crop_data/crop_images/91.jpg
crop_data/crop_images/171.jpg
crop_data/crop_images/172.jpg
crop_data/crop_images/173.jpg
crop_data/crop_images/174.jpg
crop_data/crop_images/175.jpg
crop_data/crop_images/176.jpg
crop_data/crop_images/177.jpg
crop_data/crop_images/178.jpg
crop_data/crop_images/179.jpg
crop_data/crop_images/180.jpg
crop_data/crop_images/258.jpg
crop_data/crop_images/259.jpg
crop_data/crop_images/260.jpg
crop_data/crop_images/261.jpg
crop_data/crop_images/262.jpg
crop_data/crop_images/263.jpg
crop_data/crop_images/264.jpg
crop_data/crop_images/265.jpg
crop_data/crop_images/266.jpg
crop_data/crop_images/267.jpg
crop_data/crop_images/332.jpg
crop_data/crop_images/333.jpg
```

Figure 4.12 Text file containing the file path of the images want to test

Then create a folder named crop_labels inside crop_data and copy all the label files to crop_labels folder. Open the notepad and create a new file and in that name the labels used for the dataset creation. Give the label as respective disease name in that file. Then save the file with the name as cov.names. Create a new data file like how the other files created.

CONTENTS OF THE CROP DATA FILE

classes=18

train= crop_data/crop _training.txt

valid= crop _data/crop _testing.txt


```
name= crop_data/crop.name
backup= backup
```

```
classes = 18
train = crop_data/crop_training.txt
valid = crop_data/crop_testing.txt
names = crop_data/crop.names
backup = backup
```

Figure 4.13 Contents of the crop data file

This figure 4.13 shows what are the changes have to be made in the crop data file. The changes will be according to the purpose what the requirement of the dataset wants. In this backup folder, the trained weights will keep on accumulating in the folder so that's why specifying the backup in the data file.

5.9 TRAINING

In the process of object detection, Google colab is the platform used for the training of the custom dataset. The training should not happen in the computer for that export all the files to the drive including the darknet folder. By this it can be used with the google colab and proceed with the training.

- Data sync

The first process of the training is the data synchronization. In data sync, it is just the process of copying all the files. All the files inside the darknet directory have to get off the computer to cloud and for this process the best possible way is to zip everything so that it is possible to transfer it as single file to the google drive. First convert the file into compressed zip file, now the folder is getting zipped after that get this off the google drive. Before that create the folder inside the google drive and name the folder as custom_crop_model. Then upload the zip file to the custom_crop_model in the google drive.

- Setting up the google colab

Inside the drive, create the new google collaboratory file in the .ipynb extension file and lets rename the notebook to crop_custom.ipynb. Click on that and that will take to the google colab. After that change the runtime and change the runtime type as GPU based hardware accelerator. Now the runtime connected to the cloud GPU.

Runtime means it is allocating a computer to a virtual computer that is virtual linux based computer is in the google server that will make the virtual computer to run the code.

- Interfacing the colab and google drive to this notebook

Mount the google drive to this colab notebook for that right click on the folder menu and click mount drive, it will mount the google drive.

Extract this zip file and then do some pre-processing work and finally compile it and do the testing. There are some commands to installing darknet and test. Get the system details of the computer which was allocated as a temporary computer by google.



```
%cat /etc/lsb-release

DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.6 LTS"
```

Figure 4.14 Command to get the runtime details

The figure 4.14 shows that how to get the runtime details of the system which was allocated as a temporary computer by google for the dataset training. In the GPU of the computer, the training of the dataset will happen.

Because the GPU of the virtual computer which is provided by the google colab will process the data at a higher speed compared to the GPU of the locally available computer.

The command `!apt-get update` helps to update the Repo URLs are getting updated. Now unzip the file which is in the google drive. Google drive is just like a storage device just having a computer and having a usb zip drive in the google drive. Copy the google drive into the runtime which is the temporary storage. Then only execute the commands in the runtime and set the directory. First know the present working directory using the commands. After that unzip the darknet folder and again zip it to the `/content` folder.

Now the directory shift from content to darknet. The hidden characters are not understood by the unix but it will remove all the widows formatting character from files for that have to install application called dosunix used to remove windows formatting. Convert all the files to the unix format.

Compile the darknet framework using `!make` command. Go into the google drive and create folder named backup in the `crop_weights`. Remove default backup folder of darknet if exists. Create a symbolic link to save the weight directly into google drive backup folder. Next save the weight directly into the darknet folder of the drive.

The command used for starting training with arguments is in the format of `[data file] [config file] [current weight]`.

Map argument is used to map the graph and saved as jpg format, and it is used to display the mean average precision. Start run the program and it takes long hours to train the training image. In every 200 iterations, it produces the map image. Finally, it generates the three weight files in backup folder.

The three files are

1. `crop_yolov4_1000.weights`
2. `crop_yolov4_best.weights`
3. `crop_yolov4_last.weights`

The first two files are used for the image classification and the last file is used for storing the 1000th iteration weights. The figure 4.15 and 4.16 shows the training history of the dataset.

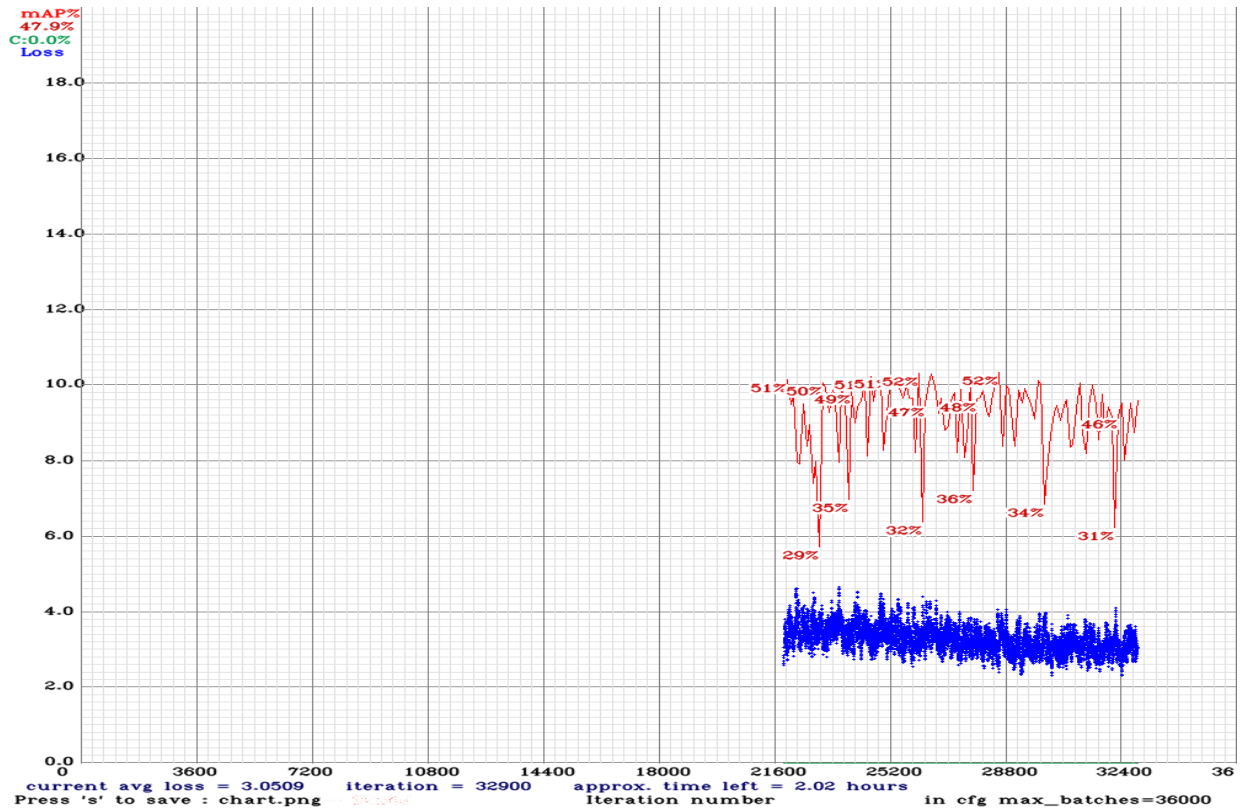


Figure 4.15 Mean average precision for 36000th iteration

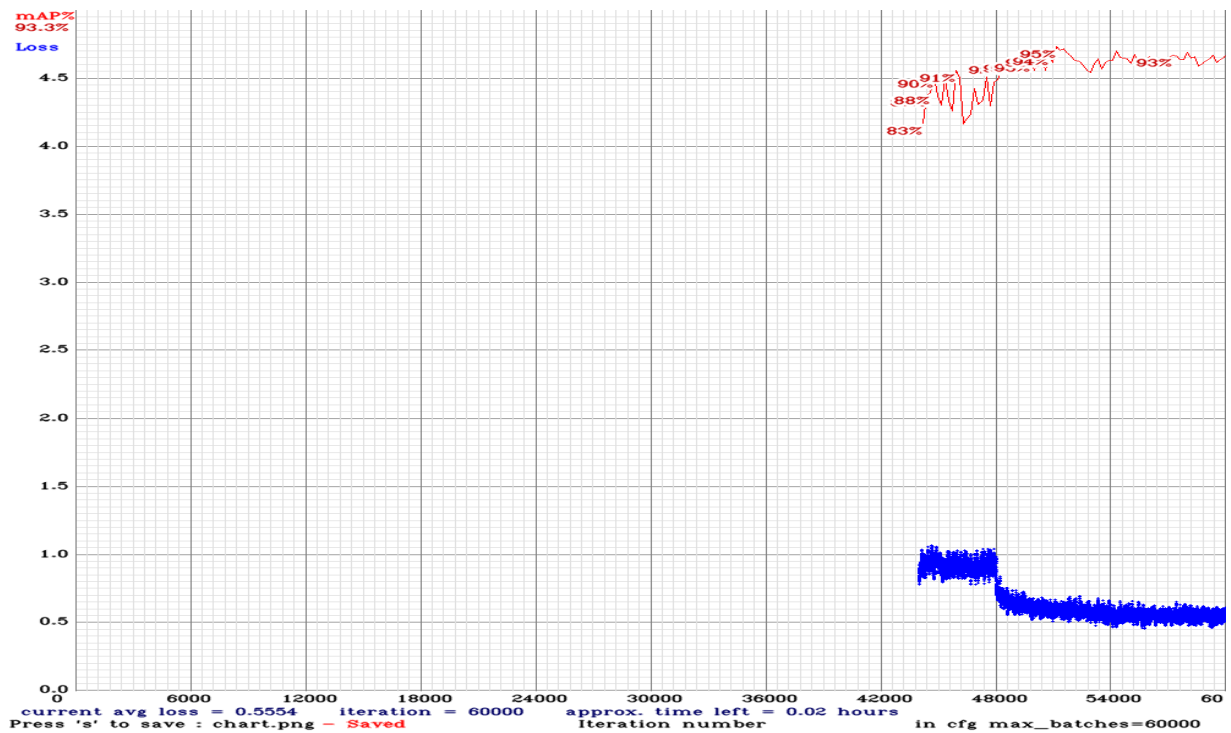


Figure 4.16 Mean average precision for 60000th iteration

The figure 4.15 shows the average precision of the dataset after the 36000th iteration. The figure 4.16 shows the average precision of the dataset after the 60000th iteration.

It gives the mean average precision as 86%. When it reaches the 1400 epoch, the average loss is around 0.5554 and the loss will keep on decreasing at some point it will reach about almost zero. That is not taken as an accuracy point. The accuracy point is where the best mean average precision is present. Now download the weight files in the drive backup folder and the training is come to end at 60000th iteration. Now create a folder and paste the weight files, images want to test and program, and it will show in the figure 4.17.

Name	Date modified	Type	Size
crop_images	2/21/2023 9:42 PM	File folder	
crop_labels	2/21/2023 2:59 PM	File folder	
crop	2/21/2023 3:11 PM	DATA File	1 KB
crop	2/21/2023 3:04 PM	NAMES File	1 KB
crop_testing	2/22/2023 9:30 AM	Text Document	5 KB
crop_training	2/21/2023 9:43 PM	Text Document	48 KB

Figure 4.17 Newly created folder containing weights and image files

4.10 DETECTION

Four input arguments are needed for crop disease detection using YOLOv4: the input image, the YOLOv4 configuration file, the learned weights of YOLOv4 and the file which has all the leaf disease name for the class in the text format. Getting the input image's width and height is the first stage in the procedure. The label and boundary boxes colours are then applied. The deep neural networks process the incoming image. To get the forecast, YOLOv4 employs numerous output layers. The bounding boxes identify the identified leaf region in the image. The bounding boxes are specified over the class label value. The bounding boxes tops display the confidence value, which expresses how confidently a leaf is contained therein. A higher confidence level for an object means that the network trusts the bounding box in which it is contained. Because the model rejects items with low confidence values and only takes into account those with a confidence level more than 0.5. The same object may occasionally appear more than once in a single photograph. With boxes bounding over the objects and a confidence level, the model only requires one detection for a single object. Using the non-maximum suppression technique, this can be accomplished. It involves ignoring the insufficient detection of the same items that are visible in an image. Each leaf in the picture may have many bounding boxes. As an illustration, suppose the YOLOv4 finds a single leaf with triple bounding boxes.

The identification of leaf in the source images with the anticipated accuracy is shown in figure 4.18. The image a shows potato leaf early blight disease, image b shows tomato early blight disease, image c shows the tomato leaf bacterial spot, image d shows apple scab leaf, image e shows tomato leaf mosaic virus and image f shows grape leaf black rot disease.

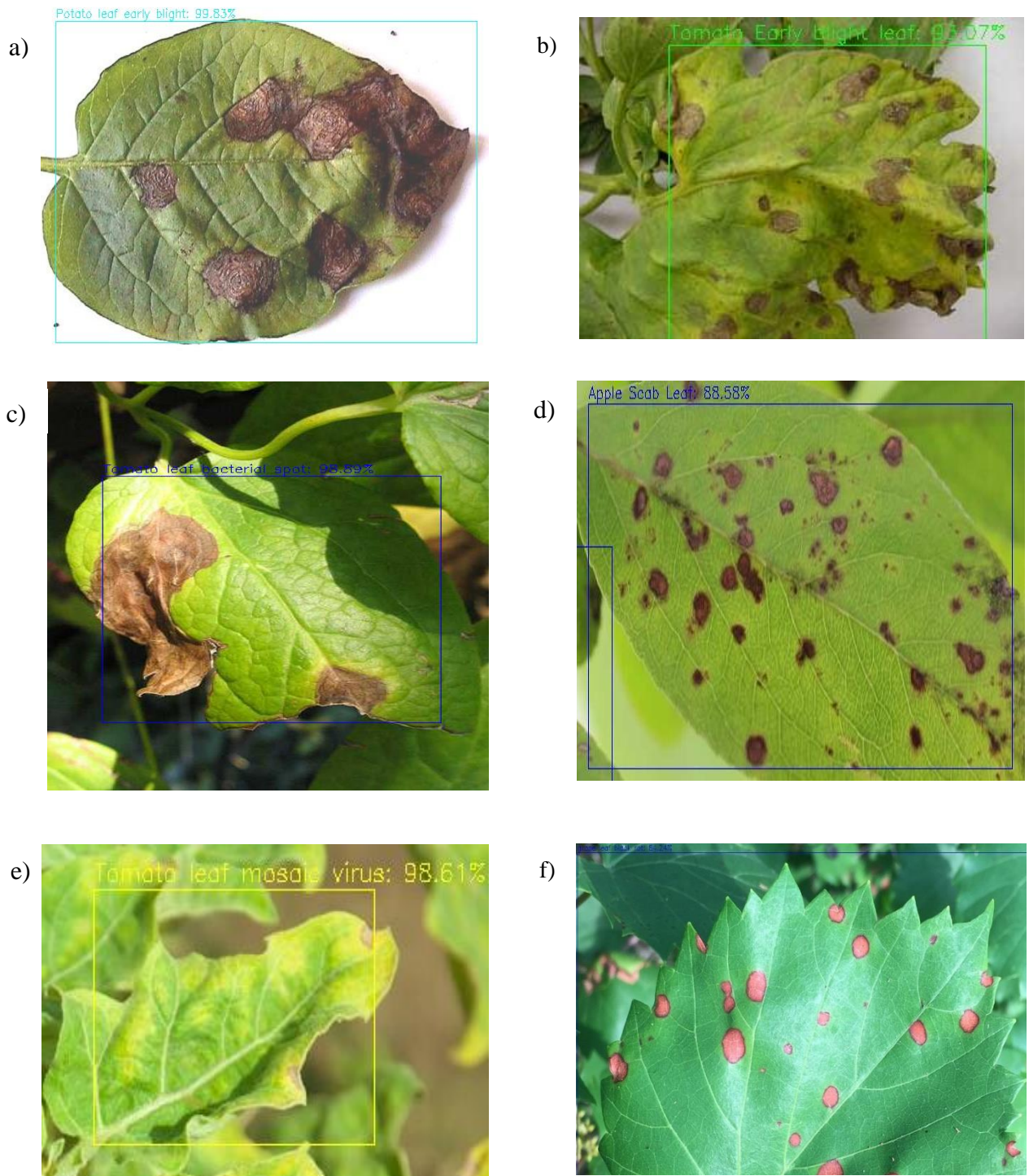


Figure 4.18 Detection of leaf disease in the input images

CHAPTER 5

RESULTS AND DISCUSSION

Deep learning-based object recognition has become a popular topic in recent years because to the neural networks' strong capacity for learning as well as its benefits in handling occlusion, scale transformation, and backdrop shifts. In many other areas, human society suffers from object detection. Convolutional neural network models based on YOLO are utilized for object recognition, and the most current YOLO iteration is known as YOLOv4. When compared to other algorithms that use R-CNN, leaf disease identification utilizing YOLO v4 has been one of the quickest and most accurate. The biggest benefit of the YOLOv4 technology is how efficient it is, it only needs to process the input once. Yolov4's crop disease detection results are currently among the best of all the systems in use. The system's detection procedure takes substantially less time to complete than the ones now in use. In comparison to earlier versions of YOLO, the detection of the leaf at three levels aids in identifying the problem with the identification of extremely small items. The crop disease detection method obtained an overall of 93% accuracy during testing on the provided custom dataset. The figure 5.1 shows the homepage layout of the website.



Figure 5.1 Homepage of the website

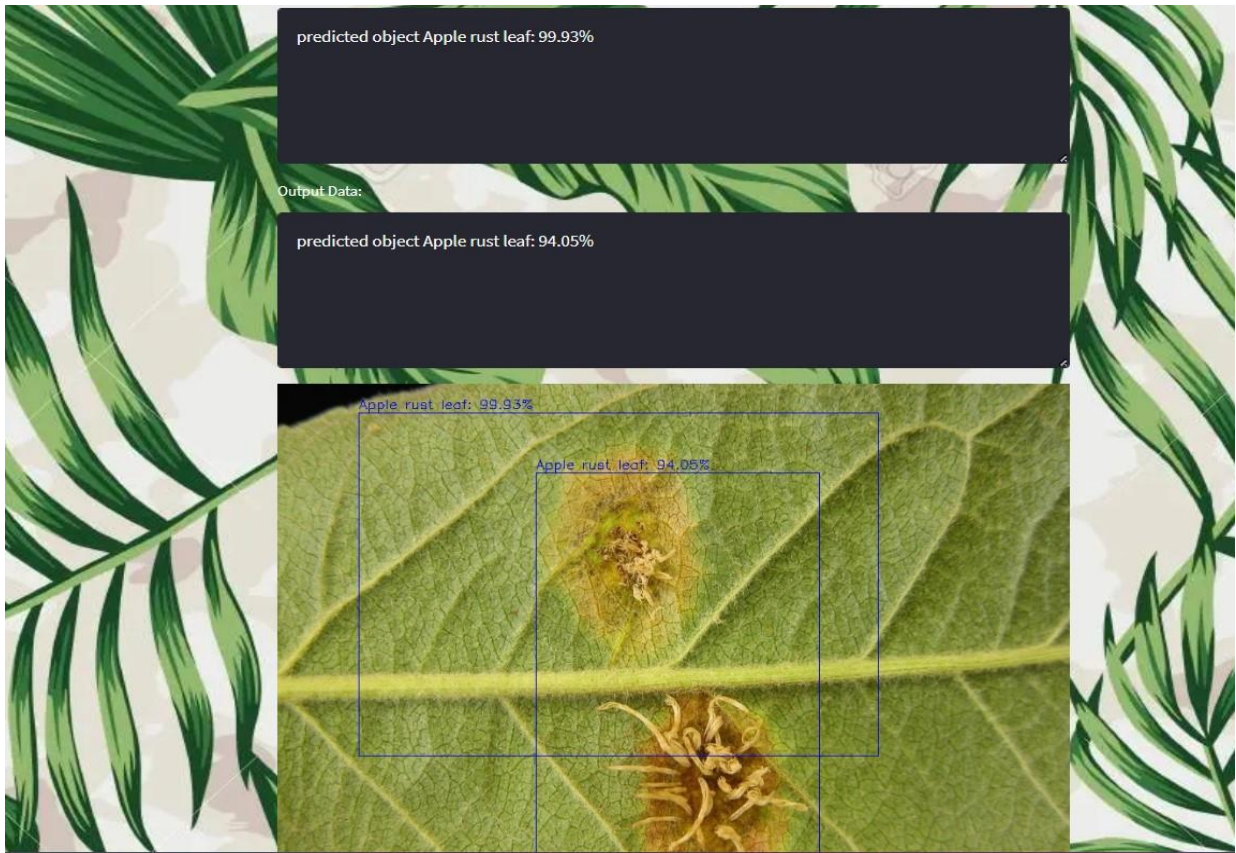


Figure 5.2 The predicted leaf disease in the uploaded image with the accuracy

After receiving an input picture which uploaded on the website, the network analyses it for the existence of leaves and utilizes bounding boxes to identify those leaves. The class name and predicted precision for the identified leaves that the bounding boxes enclose are both shown in 5.2 displays the predicted leaf disease with the accuracy for the uploaded image by the user.

Table 5.1 Performance Metrics

MODEL	YOLO v4
PRECISION	0.92
RECALL	0.89
F1 SCORE	0.91
Insert IoU	71.61%

The table 5.1 shows the performance metrics of the YOLO model that uses the CNN to process the image during the training of the dataset.

	Apple leaf	Apple rust leaf	AppleScab Leaf	Corn Gray leaf spot	Corn leaf blight	Corn rust leaf	grape leaf	grape leaf black rot	Potato leaf early blight	Potato leaf late blight	Tomato Early blight leaf	Tomato leaf	Tomato leaf bacterial spot	Tomato leaf late blight	Tomato leaf mosaic virus	Tomato leaf yellow virus	Tomato mold leaf	Tomato Septoria leaf spot
Apple leaf	0.93	0.1	0.01	0.02	0.01	0.04	0.06	0.01	0.00	0.01	0.00	0.02	0.00	0.02	0.00	0.01	0.06	0.01
Apple rust leaf	0.02	0.93	0.03	0.01	0.01	0.03	0.04	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.00	0.00
Apple Scab Leaf	0.03	0.02	0.93	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.12
Corn Gray leaf spot	0.02	0.02	0.02	0.93	0.00	0.01	0.00	0.01	0.00	0.01	0.03	0.00	0.03	0.00	0.00	0.02	0.03	0.16
Corn leaf blight	0.00	0.01	0.02	0.01	0.93	0.00	0.01	0.12	0.01	0.00	0.00	0.01	0.02	0.00	0.03	0.00	0.00	0.07
Corn rust leaf	0.01	0.00	0.00	0.01	0.02	0.93	0.00	0.02	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.02	0.03	0.16
grape leaf	0.00	0.00	0.00	0.01	0.00	0.01	0.93	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.02	0.03	0.00	0.04
grape leaf black rot	0.00	0.00	0.01	0.00	0.03	0.03	0.02	0.93	0.00	0.00	0.02	0.00	0.00	0.02	0.01	0.03	0.01	0.03
Potato leaf early blight	0.00	0.00	0.00	0.03	0.00	0.00	0.03	0.00	0.93	0.01	0.00	0.03	0.01	0.01	0.02	0.02	0.02	0.01
Potato leaf late blight	0.02	0.00	0.00	0.00	0.02	0.01	0.02	0.01	0.16	0.93	0.07	0.00	0.02	0.03	0.02	0.03	0.03	0.02
Tomato Early blight leaf	0.00	0.00	0.02	0.00	0.03	0.01	0.02	0.01	0.03	0.00	0.92	0.00	0.02	0.01	0.03	0.92	0.02	0.01
Tomato leaf	0.00	0.00	0.00	0.00	0.00	0.01	0.03	0.03	0.00	0.03	0.02	0.92	0.00	0.01	0.00	0.03	0.01	0.16
Tomato leaf bacterial spot	0.02	0.00	0.02	0.00	0.00	0.00	0.00	0.03	0.03	0.02	0.03	0.02	0.91	0.01	0.00	0.01	0.02	0.00
Tomato leaf late blight	0.04	0.07	0.16	0.12	0.03	0.00	0.00	0.00	0.00	0.03	0.01	0.02	0.00	0.90	0.01	0.01	0.03	0.02
Tomato leaf mosaic virus	0.16	0.00	0.03	0.00	0.03	0.00	0.02	0.03	0.02	0.02	0.03	0.01	0.02	0.01	0.91	0.00	0.02	0.00
Tomato leaf yellow virus	0.00	0.00	0.02	0.03	0.02	0.04	0.03	0.02	0.03	0.00	0.00	0.02	0.03	0.02	0.00	0.91	0.00	0.04
Tomato mold leaf	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.02	0.02	0.02	0.03	0.00	0.00	0.00	0.01	0.93	0.00
Tomato Septoria leaf spot	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.92

Figure 5.3 Confusion matrix

Above performance metrics called a classification system's performance and it is evaluated using a confusion matrix. In a confusion matrix, the results of a classification algorithm are displayed and summarised. The figure 5.3 displays a confusion matrix for the crop leaf diseases for the dataset created.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The recommended method for object detection with a singular neural network employs the YOLO algorithm. Since this method surpasses competing methods when applied to areas other than natural photographs, it has been widely employed. It is simple to develop this algorithm, and it may be learned by utilizing the complete image. Use region proposal approaches to confine the classifiers to a certain part of the image. The YOLO algorithm determines object boundaries by looking at the entire scene. Furthermore, it foresees a decrease in the incidence of wrongful convictions in the critical environments. When tried to compare to the other real-time classifier methods previously used, the YOLO algorithm is the fastest.

The outcomes of crop leaf disease identification using YOLOv4 are one of the finest of the systems currently in use. Compared to the present methods, the system creates the outcome much faster. In comparison to past incarnations of YOLO, observations at three levels help address the issue of recognizing small entities that are in the image. When put to the test on the carefully trained dataset, the crop leaf disease detection algorithm achieves an accuracy of 93%.

6.2 FUTURE SCOPE

In future, it may be possible to counter assessing the irregularities experienced while using real-time dataset. More training images can be used to increase accuracy when building the dataset. Using other economically significant plants and accounting for other plant components when estimating the disease's severity. This algorithm may be improved in the future to address this problem. Integrate the crop leaf disease detection technique using Web/Internet of Things (IoT) to implement the real-time disease monitoring system. It may be integrated with robots and drones to perform the detection technique. Even the tiniest adjustments to these algorithms can boost accuracy.

REFERENCES

- [1] Babila, I. F. E., Villasor, S. A. E. and Dela Cruz, J. C., (2022) "Object Detection for Inventory Stock Counting Using YOLOv5," 2022 IEEE 18th International Colloquium on Signal Processing & Applications (CSPA), pp. 304–309.
- [2] Lu, L., Xie, W. and Zhang (2020) "YOLO-compact: an efficient YOLO network for single category real-time object detection," 2020 Chinese control and decision conference (CCDC), pp. 1931–1936.
- [3] Valdoria, J. C., Caballeo, A. R., Fernandez, B. I. D. and Condino, J. M. M., (2019) "iDahon: An Android Based Terrestrial Plant Disease Detection Mobile Application Through Digital Image Processing Using Deep Learning Neural Network Algorithm," 2019 4th International Conference on Information Technology (InCIT), pp. 94-98.
- [4] Singh, U. P., Chouhan, S. S., and Jain, S., (2019) "Multilayer Convolution Neural Network for the Classification of Mango Leaves Infected by Anthracnose Disease," IEEE Access, Vol. 7, pp. 43721-43729.
- [5] Shruthi, U., Nagaveni, V. and Raghavendra, B. K., (2019) "A Review on Machine Learning Classification Techniques for Plant Disease Detection," 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, pp. 281-284.
- [6] Wang, B. and Wang, D., (2019) "Plant Leaves Classification: A Few-Shot Learning Method Based on Siamese Network," IEEE Access, Vol. 7, pp. 151754-151763.
- [7] Morbekar, A., Parihar, A. and Jadhav R., (2020) "Crop disease detection using YOLO," 2020 International Conference for Emerging Technology (INCET), pp. 1–5.
- [8] Ponnusamy, V., Coumaran, A., Shunmugam, A. S., Rajaram, K. and Senthilvelavan, S., (2020) "Smart Glass: Real-Time Leaf Disease Detection using YOLO Transfer Learning," 2020 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, pp. 1150-1154.

- [9] Wu, Q., Chen, Y. and Meng, J., (2020) "DCGAN-Based Data Augmentation for Tomato Leaf Disease Identification," in IEEE Access, Vol. 8, pp. 98716-98728.
- [10] Vaishnnave, M. P., et al. (2019) "Detection and classification of groundnut leaf diseases using KNN classifier," 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN). IEEE, pp. 1-5.
- [11] Li, L., Zhang, S. and Wang, B., (2021) "Plant disease detection and classification by deep learning—a review," IEEE Access, Vol. 9, pp. 56683-56698.
- [12] Qi, Haixia, Yu Liang, Quanchen Ding, and Jun Zou., (2021) "Automatic identification of peanut-leaf diseases based on stack ensemble." Applied Sciences, Vol. 11, No. 4.
- [13] Nepal, U. and Eslamiat, H., (2022) "Comparing YOLOv3, YOLOv4 and YOLOv5 for autonomous landing spot detection in faulty UAVs," Sensors, Vol. 22, No. 2, pp. 464.
- [14] Sardogan, Melike, Adem Tuncer, and Yunus Ozen (2018) "Plant leaf disease detection and classification based on CNN with LVQ algorithm," 2018 3rd international conference on computer science and engineering (UBMK), IEEE, Vol. 3, No. 5, pp. 566-572.
- [15] Maheswaran, S., et al. (2022) "Identification and Classification of Groundnut Leaf Disease Using Convolutional Neural Network," Computational Intelligence in Data Science: 5th IFIP TC 12 International Conference, ICCIDS 2022, Virtual Event, pp. 24–26.
- [16] Rakholia, R., Rajnish, M., et al. (2022) "Groundnuts Leaf Disease Recognition using Neural Network with Progressive Resizing," International Journal of Advanced Computer Science and Applications, Vol. 13, No. 6.
- [17] Sivasankaran, S., Jagan Mohan, K. and Mohammed Nazer, G., (2022) "A Comparative CNN Based Deep Learning Model Investigation for Identifying and Classifying the Leaf Diseases of Arachis Hypogea (Groundnut Crop) Grown in The Semi-Arid Landscapes of Villupuram District of Tamil Nādu," Journal of Algebraic Statistics, Vol. 13, No. 2, pp. 232-249.

- [18] Joshi, P., Das, D., Udutalapally, V., Pradhan M. K. and Misra, S., (2022) "RiceBioS: Identification of Biotic Stress in Rice Crops Using Edge-as-a-Service," *IEEE Sensors Journal*, Vol. 22, No. 5, pp. 4616-4624.
- [19] Redmon, J. Divvala, S., Girshick, R. and Farhadi, A., (2019) "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.
- [20] Kim, J., Sung, J.-Y. and Park, S., (2020) "Comparison of Faster-RCNN, YOLO, and SSD for real-time vehicle type recognition," in *2020 IEEE international conference on consumer electronics-Asia (ICCE-Asia)*, pp. 1–4.
- [21] Sardogan, M., Tuncer, A. and Ozen, Y., (2018) "Plant Leaf Disease Detection and Classification Based on CNN with LVQ Algorithm," *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, Sarajevo, Bosnia and Herzegovina, pp. 382-385.
- [22] Gosai, D., Kaka, B., Garg, D., Patel, R., and Ganatra, A., (2022) "Plant Disease Detection and Classification Using Machine Learning Algorithm," *2022 International Conference for Advancement in Technology (ICONAT)*, Goa, India, pp. 1-6.
- [23] Ma, D., Fang, H., Wang, N., Zhang, C., Dong, J. and Hu, H., (2022) "Automatic Detection and Counting System for Pavement Cracks Based on PCGAN and YOLO-MF," *IEEE Trans. Intell. Transp. Syst*, Vol. 4, No. 2, pp. 56 - 57.
- [24] Hessane, A., El Youssefi, A., Farhaoui, Y., Aghoutane B. and Amounas, F., (2023) "A Machine Learning Based Framework for a Stage-Wise Classification of Date Palm White Scale Disease," in *Big Data Mining and Analytics*, Vol. 6, No. 3, pp. 263-272.
- [25] Zhang, T., Yang, Z., Xu, Z. and Li, J., (2022) "Wheat Yellow Rust Severity Detection by Efficient DF-UNet and UAV Multispectral Imagery," in *IEEE Sensors Journal*, Vol. 22, No. 9, pp. 9057-9068.