Plant Disease Classification Using UAV Image

THESIS SUBMITTED TO

Symbiosis Institute of Geoinformatics

FOR PARTIAL FULLFILMENT OF THE M. Sc. DEGREE

By

Ayush Mitra

(Batch 2022-2024 / PRN 22070243007)

Symbiosis Institute of Geoinformatics

Symbiosis International (Deemed University)

5th Floor, Atur Centre, Gokhale Cross Road
Model Colony, Pune-411016

INDEX

ACKNOWLEDGEMENT

In the current era of intense competition, success is reserved for those who possess the determination to step forward and embrace challenges. Projects serve as a vital link between theoretical concepts and practical application. With this notion in mind, I initiated this project and would like to express my gratitude to the divine force, whose grace made this endeavour a reality.

I am indebted to my parents for their unwavering support and encouragement throughout the completion of this project. Their constant belief in me has been invaluable.

I extend my heartfelt appreciation to my mentor, Sahil Shah, for his invaluable guidance and unwavering support during the course of this project. I would also like to acknowledge the assistance provided by Dr. T.P. Singh and Dr. Navendu Chaudhary, who offered their expertise and aided in the allocation of an internal guide.

Furthermore, I am grateful to all my teachers and fellow data science peers at Symbiosis Institute of Geoinformatics, whose continuous assistance and encouragement have been instrumental throughout the year.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

UAV: Unmanned Aerial Vehicle

YOLO: You Only Look Once

SSD: Single Shot Detector

CNN: Convolution Neural Network

R-CNN: Region-Based Convolutional Neural Network

SVM: Support Vector Machine

XML: Extensible Markup Language

NMS: Non-Maximum Suppression

VGG: Visual Geometry Group

GPU: Graphical Processing Unit

CPU: Central Processing Unit

RPN: Region Proposal Network

RoI: Region of Interest

mAP: Mean Average Precision

NVDI: Normalized Difference Vegetative Index

VDI: Difference Vegetative Index

GNDVI: Green Normalized Difference Vegetative Index

PREFACE

Data science is a fairly broad discipline. The more immersed you become in it, the more you realize the breadth of the field. Therefore, it is crucial to possess accurate knowledge of the subject, whether it be statistics. deep learning, machine learning, and mathematics Practical assistance is one technique to gain a deeper grasp of the subject. All Data Science students want to gain as much practical experience as they can since doing so will aid in grasping the material and improving understanding. I learned a lot from this assignment that I was unaware of and had never read about. It was a really wonderful experience for me where I learned a ton of new concepts that needed to be thoroughly studied before being applied here. I got an idea as to how to frame a data science project. This project is a stepping stone of my career.

This project is about classifying if potato plants is a healthy or not using images taken from UAV (unmanned Aerial Vehicle). Agriculture is a very important part for any country's economy, especially for an agriculture depended country like India. So, identifying plants infested by disease can help treat the plant before it can infect other plants. Using images taken from UAV can help identify multiple infected plants at the same time helping to make the identification process faster and easier for very larger farming lands were identifying every plant one by one would be impossible in respect of time and money.

I am pleased to present my project to prospective readers and I am sure this will help to explore recent advancements for UAV image processing.

ABSTRACT

Agriculture plays a crucial role in human civilization, not only by providing food but also by contributing to the economy. However, crops are susceptible to diseases during cultivation, presenting a significant challenge for farmers. Early detection and effective management of these diseases are essential to minimize agricultural plant damage and associated expenses. Unfortunately, traditional methods do not allow for real-time disease detection. To address this issue, this research introduces deep learning-based techniques for disease identification and detection in potato crops. The study investigates the performance of various deep learning models to assess the feasibility and reliability of the proposed approach. The objective is to achieve accurate disease identification and detection. The results show that the suggested solution achieves an accuracy of 96.30% using YOLOV8, 94.80% using SSD, 95.68% using Faster-RCNN and 95.77% using Mask-RCNN. Notably, YOLOV8 outperforms other models, making it well-suited for real-time disease detection. By adopting these deep learning methods, farmers can detect diseases early, enabling them to take timely and appropriate actions to minimize damage to their agricultural plants. This approach offers a promising solution to the problem of disease identification and management in agriculture, contributing to the sustainable growth of the farming industry.

INTRODUCTION

Problem Statement

The prosperity of a nation depends greatly on its agriculture, which encompasses economic, social, and environmental benefits. However, the growth of agriculture is constantly threatened by plant diseases and pests. Traditionally, identifying and assessing the severity of these issues has been a costly and inefficient process, relying on labour-intensive field studies. To address this challenge, there is a need for faster and more accurate detection of plant diseases and pests, enabling early treatment measures and minimizing economic losses. This is where recent advancements in computer vision, powered by deep learning, have opened up new possibilities for utilizing unmanned aerial vehicles (UAVs) [1] in plant disease and pest detection. By leveraging UAV imagery, unhealthy plants can now be swiftly and precisely located, facilitating sustainable agricultural management practices.

Motivation

Plant diseases cause yield reductions that have a direct influence on the domestic and international food production systems and lead to financial losses. The Food and Agriculture Organisation (FAO) of the United Nations International Plant Protection Convention (2017) [2] estimates that plant diseases and pests cause a 20%–40% loss in worldwide food production. 13% of the worldwide crop production loss is attributed to plant diseases. These figures demonstrate how crucial it is to detect plant diseases in order to minimise production losses. That's why there is a need for a model to predict an infected plant before it's too late and necessary steps could be taken.

Project Scope

One of the most significant variables impacting agricultural growth and product quality is plant disease, which lowers the economic value of the commodities and services provided by the Agricultural ecosystem [3]. The timely and accurate diagnosis of diseases plays a vital role in their prevention and management. While unmanned aerial vehicles (UAVs) have been extensively employed for capturing multispectral imagery, their application in identifying forest diseases has been limited. Previous research primarily focuses on classifying diseases based on individual leaves of infected plants. However, from a practical standpoint, manually inspecting each plant in large agricultural fields is not feasible. Therefore, utilizing UAV imagery allows for the simultaneous identification of multiple infected plants, making it a more economically viable and feasible solution for implementation at an industrial scale.

Methodology

During the methodology stage, the data will be collected and undergo pre-processing before being fed into various neural architectures such as YOLO, SSD, Faster-RCNN, and Mask-RCNN. Once trained, validated, and tested, the outputs of these models will be compared using evaluation metrics such as accuracy, mean Average Precision (mAP), and F1 score. This comparison will help determine the best model for the project.

The main contributions of the research work undertaken can be summarized as follows:

1. Utilizing pre-processing techniques on the dataset, including scaling and augmentation, to

enhance the quality and suitability of the data for training the models.

2. Developing deep learning models specifically designed for the detection and diagnosis of diseases in potato crops. These models will play a crucial role in accurately identifying and classifying diseases, aiding in effective disease management.

3. Incorporating Vegetative Indices as part of the validation process to further validate the results obtained from the models. Vegetative Indices provide additional metrics for evaluating the health and condition of plants, offering a comprehensive assessment of the model's performance.

These contributions collectively contribute to improving disease detection and diagnosis in potato crops, enabling more efficient agricultural practices.

The main objectives of the research work undertaken can be summarized as follows:

1. To implement and evaluate multiple deep learning models for object detection, including YOLOV8, RCNN and SDD.

2. To compare the performance of the individual deep learning models in terms of detection accuracy, speed and robustness.

3. To identify the strength and weaknesses of different deep learning models.

4. To use vegetative index to find the validate the individual model’s performance.

5. Create a GUI interface to using the models.

LITERATURE REVIEW

There are several techniques used for plant disease classification using images.

Disease Classification

Plant disease classification using UAV images involves utilizing unmanned aerial vehicles (UAVs) or drones to capture high-resolution images of crops. These images are then processed using machine learning algorithms to classify plants based on disease type, severity, and overall health. This approach offers several advantages over traditional methods, including faster coverage of large areas, high accuracy, and the potential for early disease detection.

The process begins with flying the drone over the target area and capturing images using various sensors, including visible light and near-infrared cameras. Computer vision and machine learning algorithms are then applied to analyse the images and identify specific patterns or anomalies associated with disease presence.

A research paper by Aanis Ahmad and colleagues [3] explores the recent advancements in using computer vision techniques based on deep learning algorithms and UAV technologies for crop disease identification and treatment. It highlights the limitations of traditional methods that rely on human observation or ground-based devices, which often lack accuracy and efficiency, especially when covering vast agricultural fields. In contrast, precision agriculture leveraging deep learning algorithms and UAVs presents an effective solution for various agricultural applications, including plant disease identification and treatment.

This approach to plant disease classification using UAV images holds promise for improving agricultural practices, enabling early intervention, and ultimately leading to higher crop yields and reduced reliance on pesticides.

Dataset Used

Many existing research papers on disease detection primarily focus on datasets comprising single leaf images, such as the widely used PlantVillage Dataset or PlantDoc dataset. These datasets are often open-source or collected by researchers themselves. However, there are relatively fewer papers that concentrate on UAV image datasets, particularly in India, due to the higher investment cost associated with capturing images using UAVs.

In a paper by Mosleh Hmoud Al-Adhaileh and colleagues [4], the researchers utilized the PlantVillage dataset, which is publicly accessible. The dataset consisted of images depicting three distinct potato leaf conditions: late blight, early blight, and healthy. Each image in the dataset had a resolution of 256 × 256 pixels. The images representing early and late blight showcased different stages of the detrimental potato leaf disease, while the images of healthy potato leaves represented leaves in a normal, disease-free state.

On the other hand, research papers such as the one by Kuo Liao and colleagues [5] exemplify efforts to incorporate UAV multispectral images and implement vegetative indices in the model training process. These studies explore the potential benefits of using UAV imagery and additional vegetation indices to enhance disease detection models.

While the focus on UAV image datasets may be relatively limited, the available research highlights the significance of expanding the scope of data sources and considering the integration of UAV imagery and vegetative indices to improve disease detection and diagnosis models.

Different Algorithm used

In a paper by Abdelmalek Bouguettaya and colleagues [6], the recent advancements in computer vision techniques based on deep learning algorithms and UAV technologies for crop disease identification and treatment are analysed. Traditional methods relying on human observation or ground-based devices face limitations in terms of accuracy and time required to cover large fields. The paper compares various algorithms, including R-CNN, Fast R-CNN, Faster R-CNN, YOLO, and Single Shot MultiBox Detector (SSD), to assess their performance in disease detection.

In another paper by Pallepati Vasavi and colleagues [7], the focus is on identifying the most suitable algorithms for deployment in standard systems, mobile/embedded systems, drones, robots, and UAVs. Crop leaf diseases exhibit variations in shape, size, and color, and the model development process involves capturing images of diseased leaves and recognizing disease patterns to prevent crop loss. The study incorporates sample designs from mobile phone cameras, digital cameras, drones, and UAVs, both in real-time (on-site) and controlled conditions.

The results indicate that modified CNNs, optimized deep learning models, and transfer learning models outperform basic CNNs. Modified deep learning techniques demonstrate better performance than traditional machine learning techniques. Specifically, the multi-channel model based on modified CNN achieves the highest accuracy of 99.5% in deep learning, while SVM with a linear kernel achieves 99% accuracy in machine learning techniques.

These research papers highlight the advancements in computer vision, deep learning algorithms, and the integration of UAV technologies in crop disease identification and treatment. The findings demonstrate the potential for improved accuracy and performance in disease detection models, paving the way for more effective agricultural practices.

Performance Parameter used

In the paper by S. Yegneshwar Yadhav and colleagues [8], they utilized performance metrics such as validation and training loss along with accuracy, recall, and precision metrics. These evaluation metrics are based on the concepts of true positive, false positive, true negative, and false negative, which are used to assess the model's performance in disease classification.

Similarly, in the paper by Hasin Rehanaa and colleagues [9], they employed evaluation metrics such as F2 score, accuracy, recall, precision, and mean Average Precision (mAP) to assess the performance of their model in localizing and recognizing tomato leaf diseases. They achieved a high mAP of 96.31%, indicating the model's efficiency in disease detection. The recall and F2 scores were also reported as 97.71% and 98.07% respectively.

In your project, the focus will be on using YOLO, SSD, Faster-RCNN, and Mask-RCNN algorithms for disease classification based on UAV-captured images. To evaluate the performance of these models, you will employ evaluation metrics such as F1 score, accuracy, and mAP. These metrics will help you determine the most suitable model for your project.

By adopting these evaluation metrics, you can effectively assess the performance and capabilities of different models, ultimately leading to the selection of the best-performing model for disease classification using UAV images.

Table 1: Comparison of different works.

Author	objective	Model Used	Year published	Application	Accuracy
Dashuang Liang **[10]**	Detection	CNN	2021	Improved Convolutional Neural Network for Plant Disease Detection	96.7%
Abdelmalek Bouguettaya **[6]**	Detection	R-CNN, Fast R-CNN, Faster R-CNN, YOLO, and Single Shot MultiBox Detector (SSD).	2022	A survey on deep learning-based identification of plant and crop diseases	94.74%
Pallepati Vasavi **[11]**	Classification	NN, NB, CNN	2021	Crop leaf disease detection and classification using machine learning and deep learning algorithms	99.5%
Anna Anbumozhi **[12]**	Classification	Progressive Groundnut Convolutional Neural	2023	Leaf Diseases Identification and Classification of Groundnut Crop using PGCNN	97.58%

		Network (PGCNN)			
Mosleh Hmoud Al-Adhaileh **[13]**	Detection	CNN + ReLU	2023	Potato Blight Detection Using Fine-Tuned CNN Architecture	99%
Gugan Kathiresan **[14]**	Detection	RiceDenseNet + GAN	2021	Disease detection in rice leaves using transfer learning techniques	99.97%
Ümit Atila **[15]**	Classification	EfficientNet	2020	Plant leaf disease classification using EfficientNet deep learning model	99.39%
Guneet Sachdeva **[16]**	Classification	Bayesian Learning, CNN	2021	Plant leaf disease classification using deep Convolutional neural network with Bayesian learning	98.9%
Everton Castelao Tetila **[17]**	Detection	Inception-v3, VGG-19, ResNet-50	2020	Automatic recognition of soybean leaf diseases using UAV images and deep convolutional neural networks	99.04%
Kuo Liao **[5]**	Detection	RF and SAM	2022	Detection of Eucalyptus Leaf Disease with UAV Multispectral Imagery	90.1%.

Sumit Kumar **[18]**	Detection	Faster R-CNN, R-FCN, SSD	2021	Plant Disease Detection Using CNN	91.7%
S.Yegneshwar Yadhav **[19]**	Classification	CNN Algorithm implemented in Raspberry Pi kit	2020	Plant Disease Detection and Classification using CNN Model with Optimized Activation Function	95%

METHODOLOGY

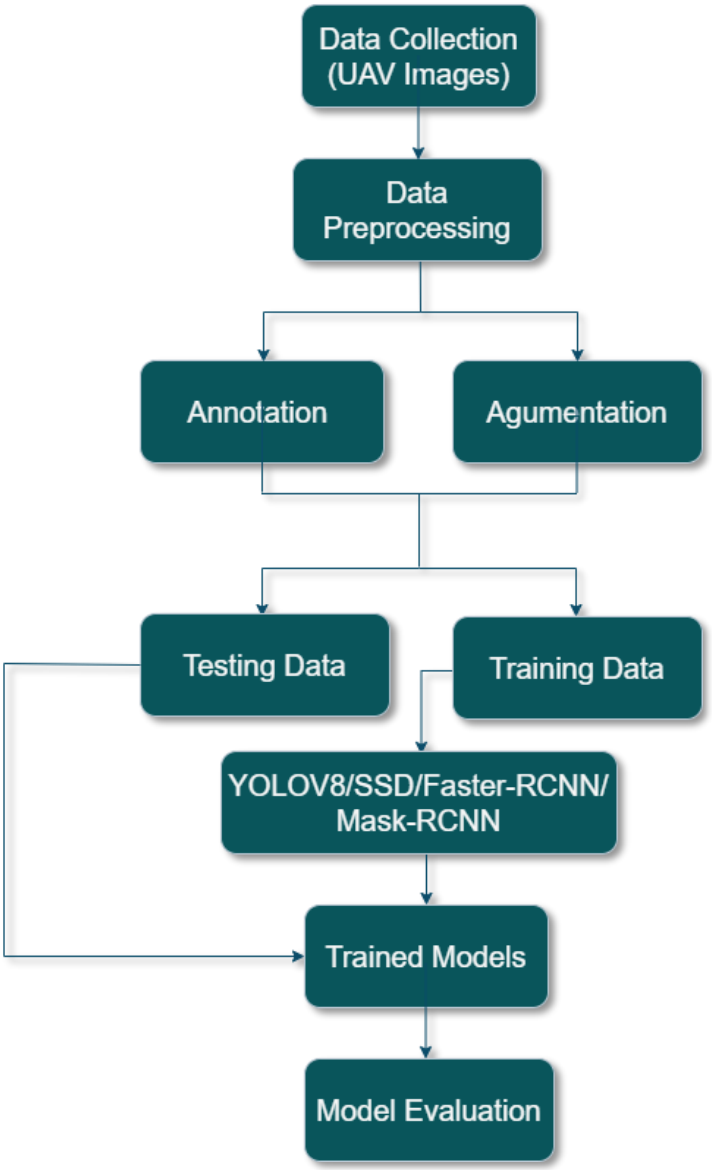


Figure 1: Methodology Flowchart

1. Data Collection:

The dataset used in this project consists of aerial images of a potato crop, specifically focusing on assessing crop health in precision agriculture applications. The images were obtained from a field at the Aberdeen Research and Extension Centre, located at the University of Idaho. To capture the images, a Parrot Sequoia multispectral camera was mounted on a 3DR Solo drone. The Parrot Sequoia camera comprises various sensors, including an RGB sensor with a resolution of 4,608×3,456 pixels (16 Megapixels), and four monochrome sensors with a resolution of 1,280×960 pixels (1.2 Megapixels) that capture narrow bands of light wavelengths: green (550 nm), red (660 nm), red-edge (735 nm), and near-infrared (790 nm). The drone was flown over the potato field at a low altitude of 3 meters (approximately 10 feet) to acquire the images. The specific objective was to capture drought stress in Russet Burbank potato plants caused by premature plant senescence. The dataset consists of 360 RGB image

patches, each measuring 750×750 pixels, in JPG format. These patches were extracted from the high-resolution aerial images using operations such as cropping, rotating, and resizing. The dataset is divided into two subsets: a training subset consisting of 300 images and a testing subset containing 60 images. The availability of this dataset enables the training and evaluation of machine learning models for crop health assessment, particularly focusing on the detection of drought stress in potato plants using aerial imagery in precision agriculture applications. [20]

Table 2: Instances of Each Class in the Data.

Classes	Instances
Healthy	7600
Stressed	11000

Figure 2: Examples of the Dataset

2. Data Pre-processing:

Data pre-processing is a vital component in the field of digital image processing as it encompasses the conversion of raw image data into a format that is well-suited for subsequent analysis and interpretation. It involves employing a range of techniques that aim to enhance image quality, accentuate relevant features, and mitigate the presence of noise or artifacts within the images.

a. Image Annotation:

To make the image suitable for the model, images needed to be annotated for training, so IabelImg [21] software was used to annotate the images based on the two classes (healthy, stressed). The annotation was done using the XML file provided with the dataset, where each

image had its respective XML file.

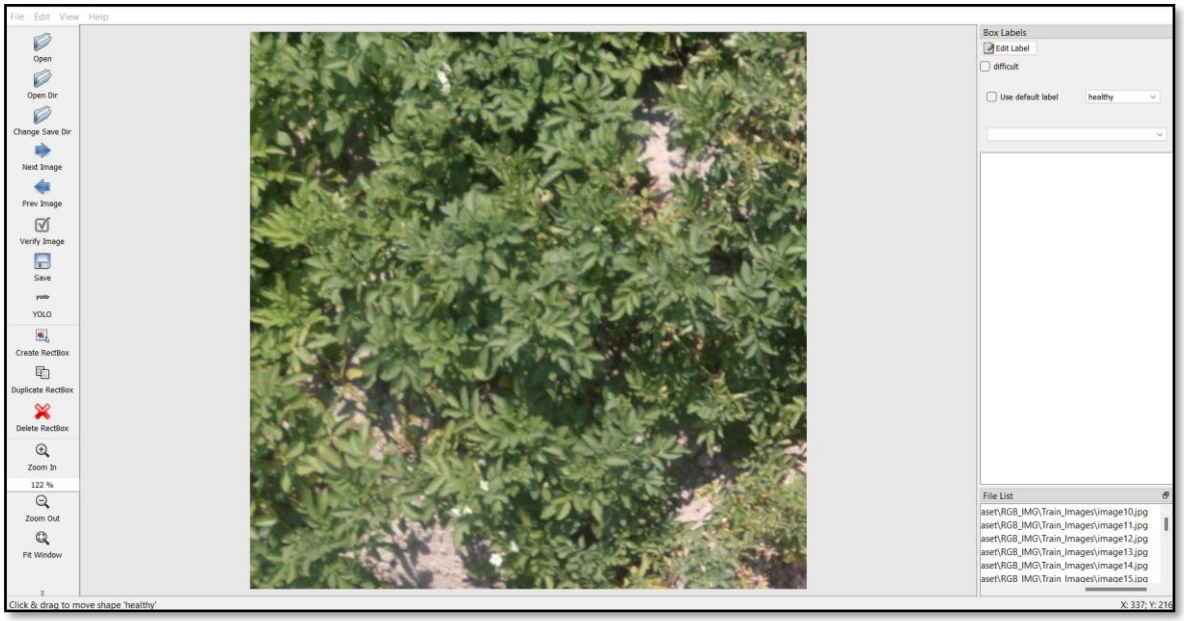


Figure 3: picture of the LabelImage tool

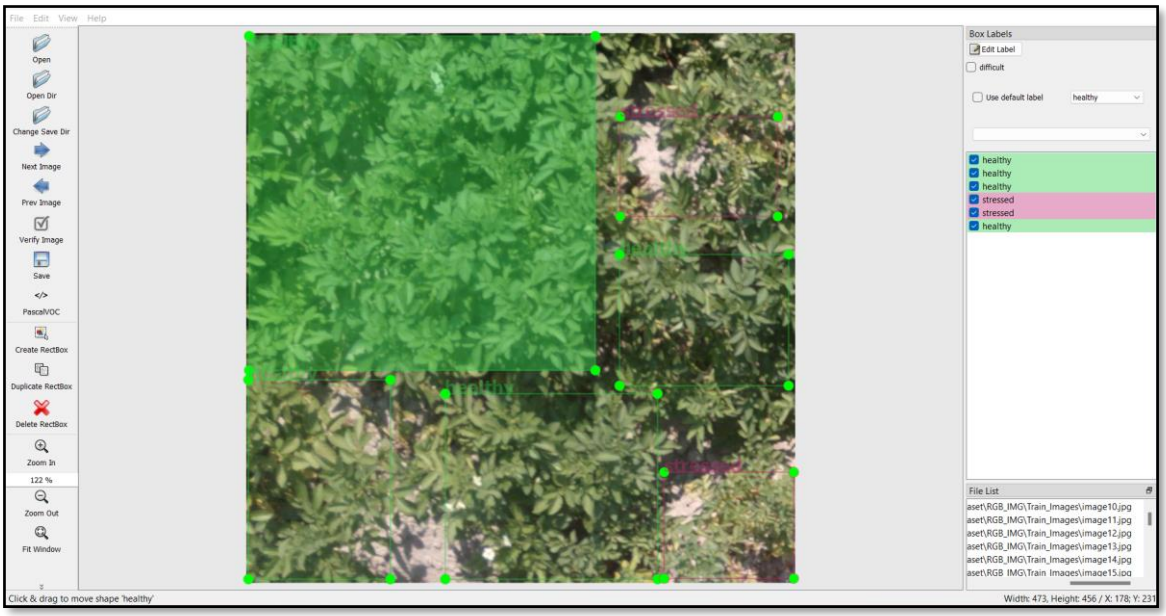


Figure 4: Image being annotated in LabelImage

b. Image Augmentation:

Image data augmentation [22] is a technique used to increase the diversity and richness of a given image dataset by generating modified versions of the existing pictures. When processed by a computer, images are essentially represented as a grid of numerical values that correspond to pixel intensities. By manipulating these pixel values in various ways, new and improved images can be created. These augmented images maintain a resemblance to the original dataset while incorporating additional information that aids in the generalization abilities of machine learning systems. Through transformations like rotations, translations, scaling, and adding

noise, data augmentation expands the range of image variations, enhancing the model's robustness and performance without fundamentally changing the underlying content of the images

3. Data Preparation:

In order to input the images into the model, it is necessary to prepare them in a specific format. This involves creating a text file for each image, containing information about the class and the coordinates of the bounding box for each class. This format is crucial for training the YOLO model, as it relies on the bounding box information associated with different classes. In the provided figure, the TXT file represents the class labels, where "0" corresponds to the healthy class and "1" corresponds to the stress class.

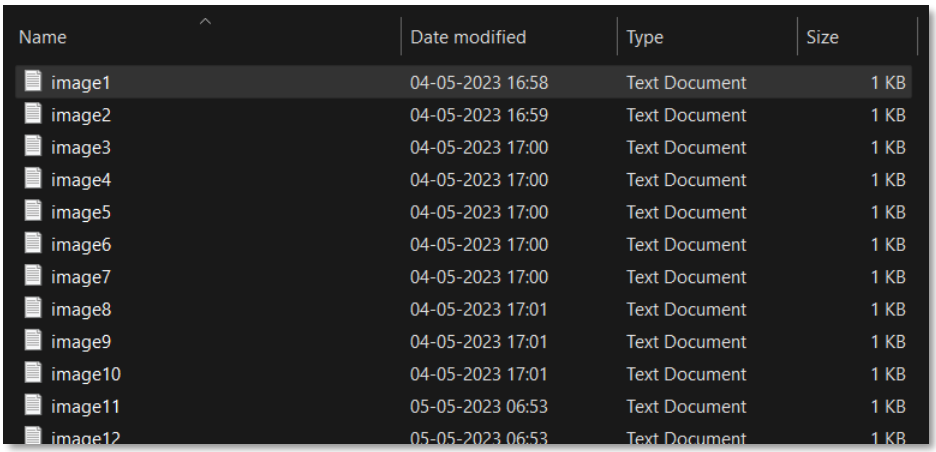


Figure 5: Annotation file of each image in .TXT format

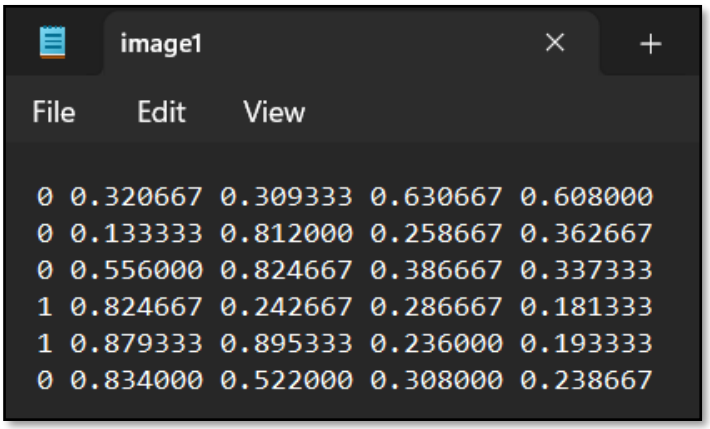


Figure 6: Bounding Box values for each class

a. Split Data:

The data need to be in a particular order for the model. The model has two modes Training and Testing. So, the images need to be divided into two folders for each train and test, and inside each folder there should be two other folders where one folder holds all the images and other folder holds all the TXT file related to the images. In the figure below we can see the folder hierarchy and each subsequent folder (train, validation, test) have two folders within it. The Image folder holds all the images and the labels folder holds all the annotation files in the form of .txt file.

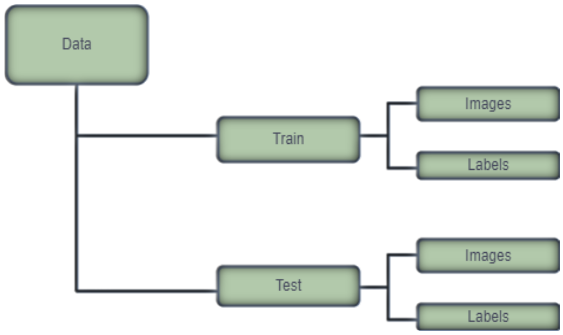


Figure 7: File Hierarchy of the data

4. Model Design:

The design of a model in image object detection involves developing a structure or framework capable of accurately detecting and localizing objects in an image. These models are essential in numerous applications, including autonomous driving, surveillance systems, and medical imaging. In this project, the following architectures are used:

a. YOLO:

YOLO [23], short for "You Only Look Once," is a renowned approach for object detection that utilizes a single neural network to simultaneously predict bounding boxes and class probabilities for objects in an image. Unlike traditional object detection algorithms that involve multiple steps to locate objects, YOLO analyzes the entire image in a single pass. This leads to a faster and more efficient process. Over time, YOLO has evolved through several iterations, each improving upon the performance, speed, and accuracy of its predecessor.

The latest iteration of YOLO, known as YOLOv8 [24], achieves state-of-the-art results on various benchmark datasets, solidifying its position as the fastest and most accurate YOLO model to date. YOLO finds applications in diverse fields such as robotics, surveillance, and autonomous vehicles.

The You Only Look Once (YOLO) object detection technique follows a multi-stage process. The steps below outline the functioning of YOLO:

- **Input:** The algorithm requires a picture as input.

- **Grid Division:** The picture has been split into cells using a grid. It is the responsibility of each cell to identify the items that are present there. The input picture size and the size of the network's most recent convolutional feature map determine the grid's size.

- **Feature Extraction:** In order to extract features, each cell is run through a convolutional neural network (CNN) [25]. A huge collection of photos was utilised to pre-train this CNN on attributes that might be used to object detection.

- **Objectness Score:** An object's presence or absence is indicated by the predicted objectness score for each cell. Using a logistic regression function [26], which forecasts the likelihood of an object existing in the cell, this is accomplished.

- **Class Probability:** YOLO forecasts the class and probability of the item for each cell that

anticipates an object. This is accomplished by computing the conditional probability of an object belonging to each class using the softmax [27] function.

- **Bounding Box:** YOLO predicts both the bounding box and the object for each cell that predicts an item. The bounding box is anticipated in relation to the cell size and is represented by its centre coordinates, width, and height.

- **Non-Maximum Suppression:** The projected bounding boxes are subjected to non-maximum suppression (NMS) [28] by YOLO in order to eliminate unnecessary bounding boxes and increase the detection accuracy. A lower confidence level for NMS causes it to exclude all overlapping bounding boxes.

- **Output:** YOLO's final output is a collection of bounding boxes that reflect the recognised objects in the input image, together with the associated class and confidence score.

The architecture of YOLO is based on GoogleNet [29] architecture. The figure below shows the architecture of YOLO.

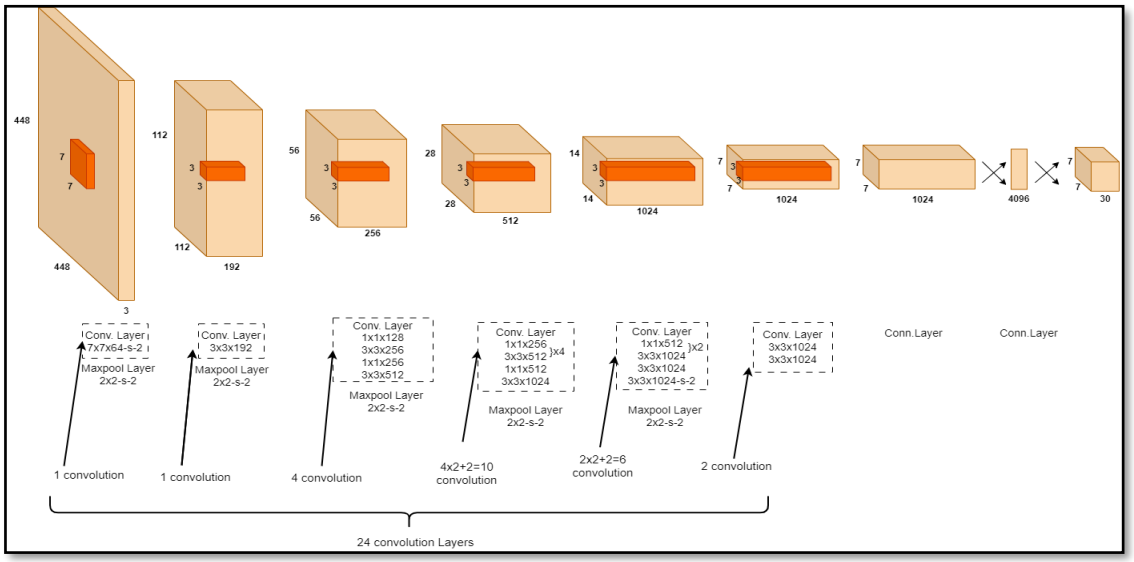


Figure 8: Architecture of YOLO

The YOLO architecture got updated many times by many different researchers and organisations. The most famous YOLO versions are YOLOV5 [30] and YOLOV7 [31]. The most recent YOLO version is YOLOV8 which released in January 2023. The YOLOV5 architecture is shown in figure 8. Yolov5, a state-of-the-art object detection model, is composed of three main components: (1) Backbone: CSPDarknet, (2) Neck: PANet, and (3) Head: YOLO Layer. The architecture begins with inputting data into CSPDarknet for feature extraction, followed by feature fusion in PANet. Finally, the YOLO Layer produces detection results including class, score, location, and size.

YOLOv8, an advancement of earlier YOLO models, adopts a fully convolutional neural network approach known as "backbone and head" for visual information processing. YOLOv8 builds upon the CSPDarknet53 architecture [32], which consists of 53 convolutional layers and employs cross-stage partial connections to enhance information flow across different levels of

the network. The head of YOLOv8 is formed by multiple convolutional layers, followed by fully connected layers responsible for estimating object detection bounding boxes, objectness scores, and class probabilities.

One distinctive feature of YOLOv8 is the incorporation of a self-attention mechanism within the network. This mechanism allows the model to dynamically allocate attention to different aspects of the image based on their relevance to the task at hand. Another notable capability of YOLOv8 is its ability to perform multi-scale object detection. By utilizing a feature pyramid network, the model can detect objects of varying sizes and scales. The feature pyramid network consists of multiple layers that recognize objects at different scales, enabling the model to identify both large and small objects in an image.

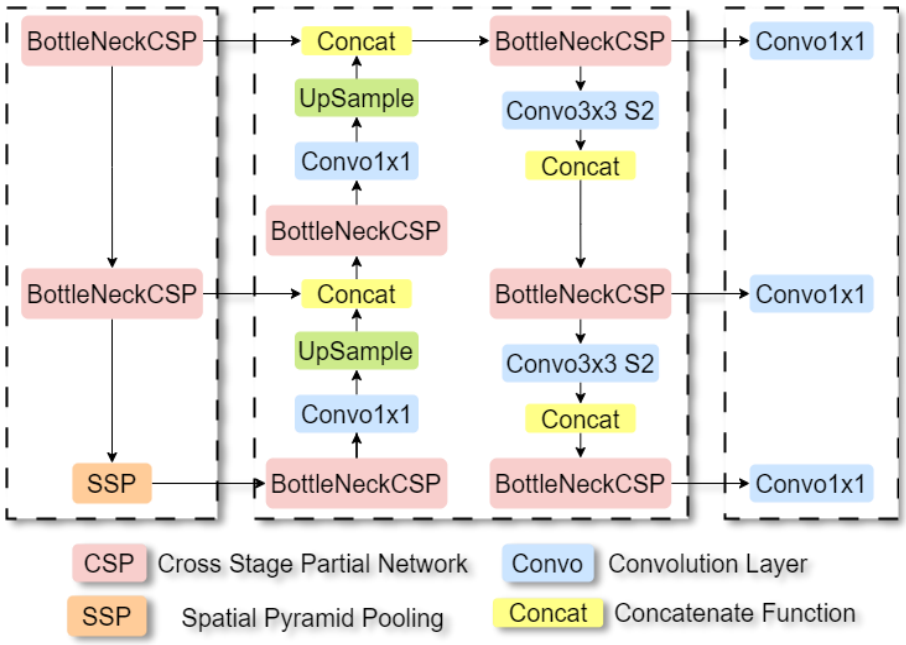


Figure 9: Architecture of YOLOV5

b. SSD:

The Single Shot Detector (SSD) is an object detection model that, similar to YOLO, is capable of detecting multiple objects in an image with a single pass using multibox techniques. SSD utilizes a VGG-16 base network followed by additional convolutional layers known as multibox convolution layers.

The key components of the SSD model include:

1. VGG-16 Base Network: The VGG-16 network is a well-established convolutional neural network architecture primarily used for high-quality image classification. In SSD, the VGG-16 base network is utilized for feature extraction, although the final classification layers are not included.

2. Additional Convolutional Layers: SSD incorporates extra convolutional layers on top of the VGG-16 base network specifically designed for object detection. These additional layers aim to detect objects at different scales by gradually reducing in size towards the end of the base network. Each feature layer has its own set of convolutional models dedicated to object

detection.

Prediction of bounding boxes and confidence scores for different objects in an image is accomplished using multiple feature maps of varying sizes that represent different scales. As the convolutional layers progress, the size of the feature maps decreases while their depth increases. This hierarchical representation allows the model to capture more complex and larger objects by covering wider receptive fields. The initial convolutional layers, on the other hand, focus on smaller receptive fields, aiding in the detection of smaller objects within the image.

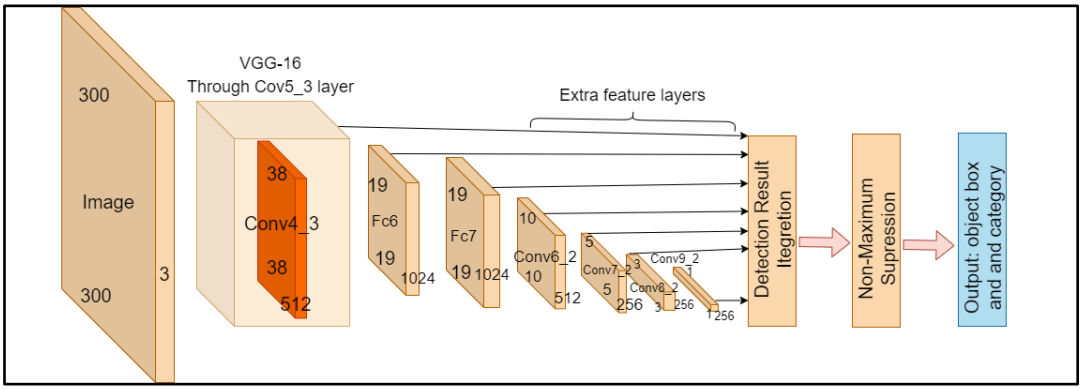


Figure 10: Architecture of SSD

c. Faster-RCNN:

Introduced in 2015 by Ross Girshick, Shaoqing Ren, Kaiming He, and Jian Sun [33], Faster R-CNN (Region-based Convolutional Neural Networks) is a widely used object detection architecture based on convolutional neural networks. Alongside models like YOLO and SSD, Faster R-CNN has become a popular choice for object detection tasks.

Faster R-CNN comprises three main components:

1. Convolutional Layers: These layers are responsible for learning filters that can extract meaningful features from input images. For example, filters trained to recognize human faces would capture specific shapes and colours associated with faces. Convolutional networks typically include convolution layers, pooling layers, and a final component such as fully connected layers or specialized modules for tasks like classification or detection. Convolution involves sliding a filter over the input image, producing a feature map. Pooling reduces the number of features by discarding pixels with low values. In Faster R-CNN, fully connected layers are utilized to classify attributes that may not be explicitly captured by the network.

2. Region Proposal Network (RPN): The RPN is a compact neural network that slides across the final feature map generated by the convolutional layers. It predicts the presence or absence of objects and estimates bounding boxes for those objects.

3. Classes and Bounding Boxes Prediction: Another fully connected neural network is employed to predict both the class labels and the bounding box coordinates for the proposed regions generated by the RPN.

By combining these components, Faster R-CNN achieves accurate object detection by effectively identifying regions of interest and predicting their class labels and bounding box coordinates.

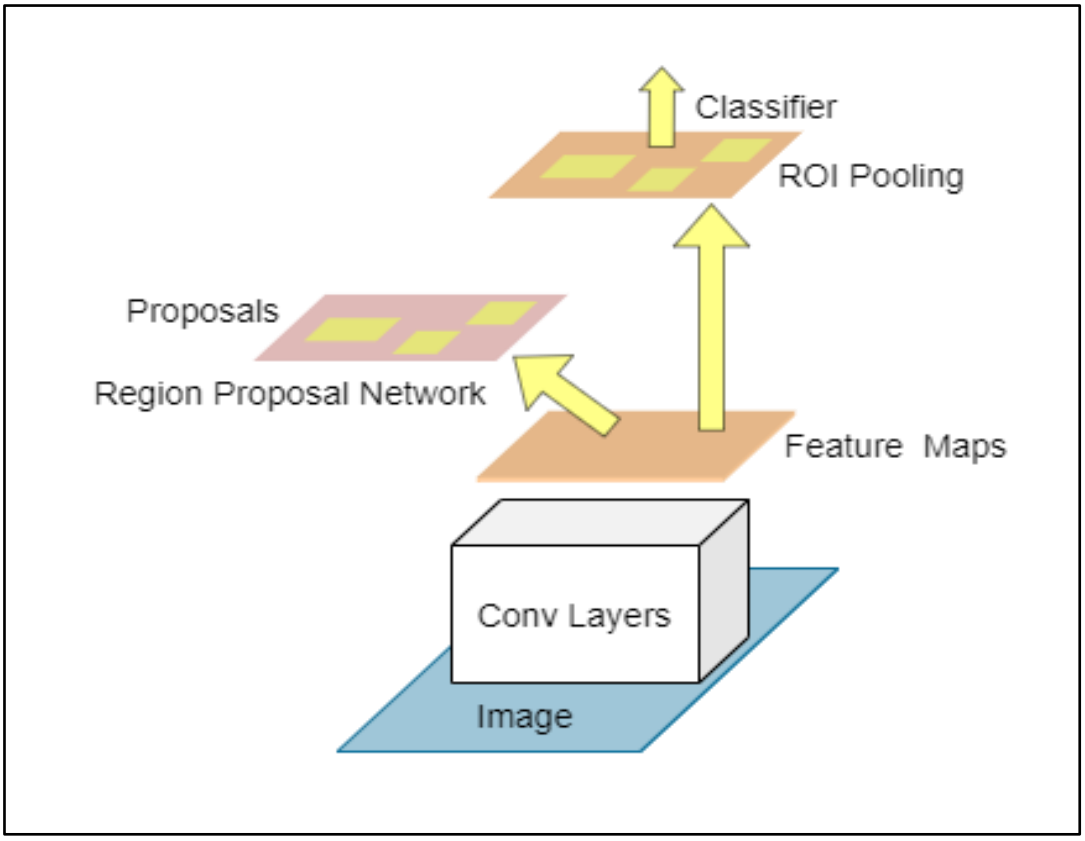


Figure 11: Architecture of Faster-RCNN

d. Mask-RCNN:

In 2017, Kaiming He et al [34]. introduced Mask R-CNN, a variation of Faster R-CNN that includes an additional layer for predicting segmented data. While both architectures share the same stage for region proposal generation, the second stage differs. In Mask R-CNN, this stage operates in parallel to predict the item class, construct the bounding box, and generate a binary mask for each Region of Interest (RoI). This addition allows Mask R-CNN to provide more detailed segmentation information alongside the object detection and classification capabilities of Faster R-CNN.

Mask RCNN is made up of:

i. Backbone Network: The creators of Mask R-CNN conducted experiments with two types of backbone networks: regular ResNet (ResNet-C4) and ResNet with a feature pyramid network (ResNet-FPN). The traditional ResNet design showed similarities to Faster R-CNN, while ResNet-FPN introduced some modifications. ResNet-FPN utilized a multi-layer approach to generate Regions of Interest (RoI) at multiple scales, which improved the accuracy compared to the previous ResNet design.

ii. Region Proposal Network: The convolutional feature map generated by the preceding layer undergoes a 3x3 convolutional layer. Subsequently, it is fed into two parallel branches that perform computations for the objectness score and regression of bounding box coordinates. In this case, a single anchor stride and three anchor ratios are employed for the feature pyramid,

as we already possess feature maps of different sizes to detect objects of various scales.

iii. Mask Representation: Mask R-CNN incorporates a mask branch that differs from the classification and bounding box regression layers by providing spatial information about objects. Unlike simply reducing the output to a fully connected layer, which proved ineffective for mask prediction, Mask R-CNN utilizes a fully connected network dedicated to mask prediction. This network, implemented as a convolutional network, takes a Region of Interest (RoI) as input and generates a mask representation of size m * m. During inference, the mask is upscaled to match the input image, and a 1 * 1 convolutional layer is applied to reduce the number of channels to 256.To enable the fully connected network to predict the mask, RoIAlign is employed. RoIAlign transforms the variable-sized feature map produced by the region proposal network into a fixed-sized feature map. The Mask R-CNN paper describes two architectural options: one where RoIAlign is applied after passing the mask-generating convolutional neural network (CNN) through a specific form (ResNet C4), and another where RoIAlign is applied just before the fully connected layer in a different network (FPN Network). The mask generation branch is implemented as a fully convolutional network, generating a K * (m * m) mask, where K represents the number of classes (one mask per class), and m is set to 14 for ResNet C4 and 28 for ResNet-FPN.

iv. RoI Align: RoIAlign and RoIPool have a similar purpose, which is to generate fixed-size regions of interest from proposed regions.

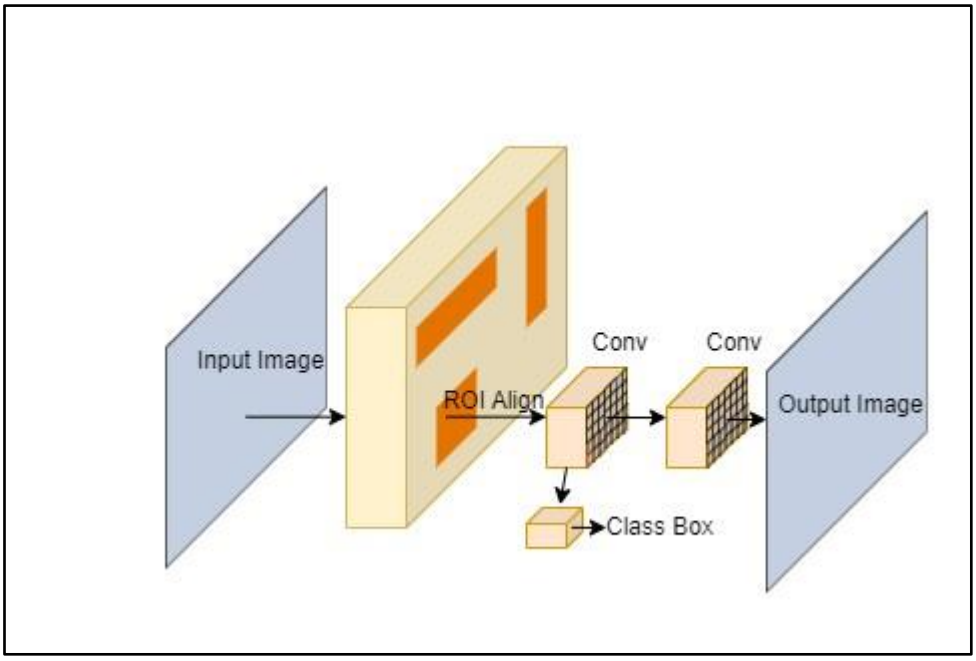


Figure 12: Architecture of Mask-RCNN

5. Model Development:

The model can be developed using YOLOV8 which can be used using a python library known as ultalytics [35], and for SSD TensorFlow object detection API [36] was used. The code for class detection could be done using a python script or a command line interface. But the problem with deep learning models is that they require a very high computational power to train, so in a local machine it's very hard to train a model. Google Colab gives a very good

alternative. It is a cloud-based platform with allows to run code on a virtual machine. It has a free version which could be used to train a model. Even if the free GPU system is used to make the use of both GPU and CPU the model will train faster. The only problem with Google Colab is that it doesn't hold data permanently after the runtime is terminated all data, installed packages and variables are lost. To store data, we may need to use google drive and mount it in the google Colab notebook to use the data directly from the drive.

The models can be built on different pretrained models or a model can be developed from scratch using the available model's architecture. YOLOV8 have many different pretrained models and model architecture, the model's name depends on the number of convolution neural network it has. The base model architecture is known as yolov8.pt [18] or yolov8.yaml depending upon if it is a pretrained model or the model architecture. Whereas the highest number of convolution neural network model is known as yolov8x.pt or yolov8x.yaml. Generally, the yolov8x variant of the model have a better performance but takes the most amount of time to train. For the project yolov8.pt was used for training. For SSD, the TensorFlow model zoo was used to select the pretrained SSD model. In both Faster-RCNN and Mask RCNN Detectron2 model zoo was used and both the models were trained for 3000 steps. The models were trained for 1500 images and after that was tested on 60 images to find the model's accuracy and other evaluating measures.

Table 3: Different Pretrained models for YOLOV8

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Testing:

After the models have been trained and tested on 60 test images. To check how well the model is doing the prediction. The output of the model is in the form of an images with bounding box drawn on them based on the class the model predicted with their confidence level on the top.

a. YOLO:

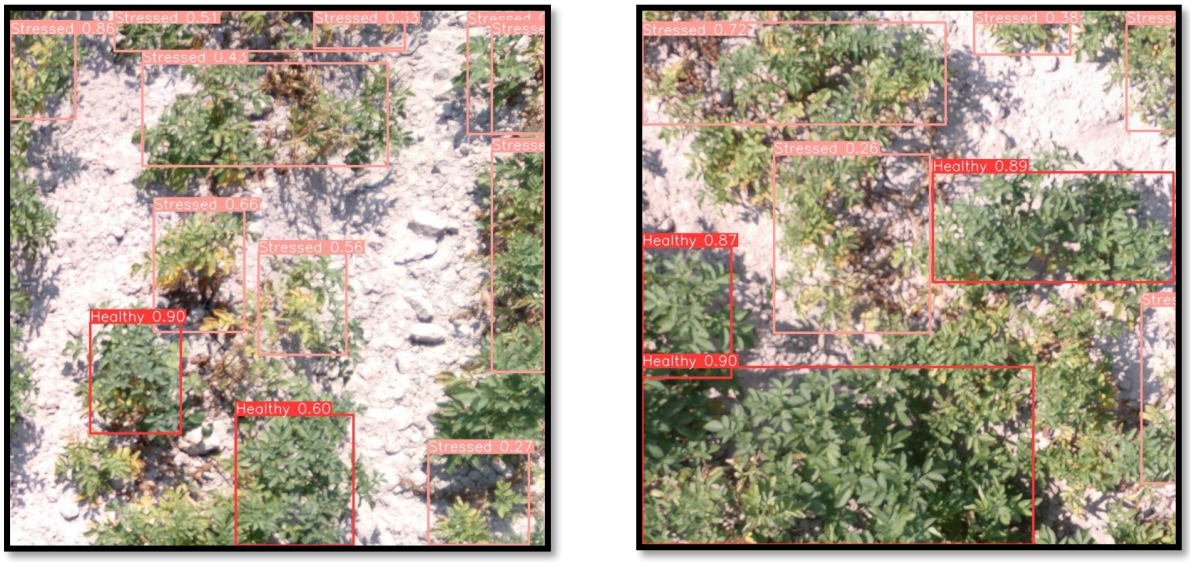


Figure 13: Output of the YOLOV8 model

b. SSD:

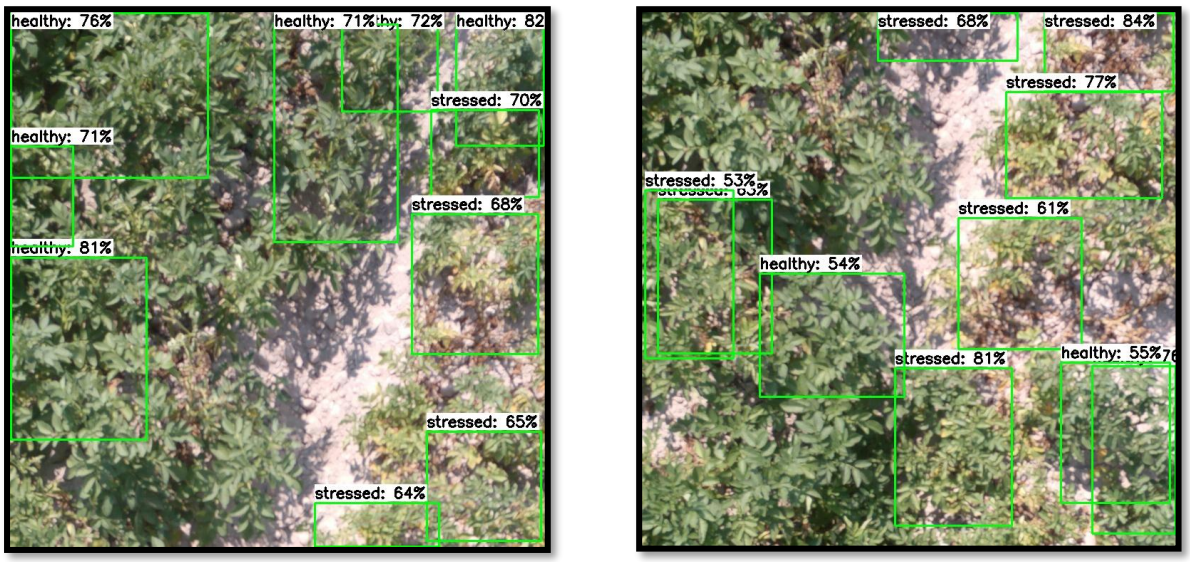


Figure 14: Output of the SSD model

c. Faster-RCNN:

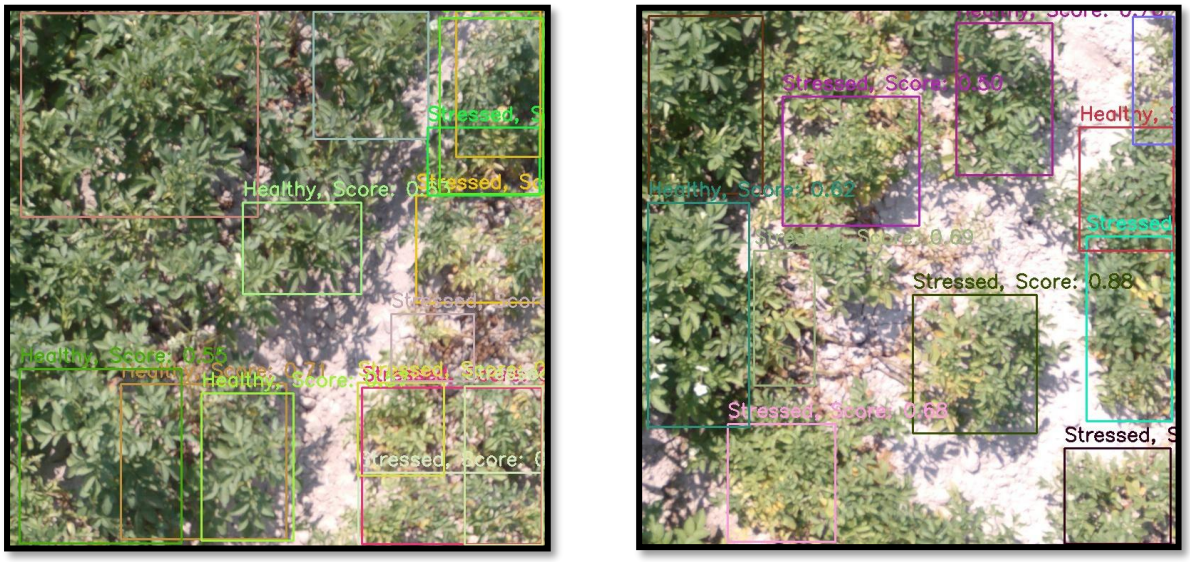


Figure 15: Output of the Faster-RCNN model

d. Mask-RCNN:

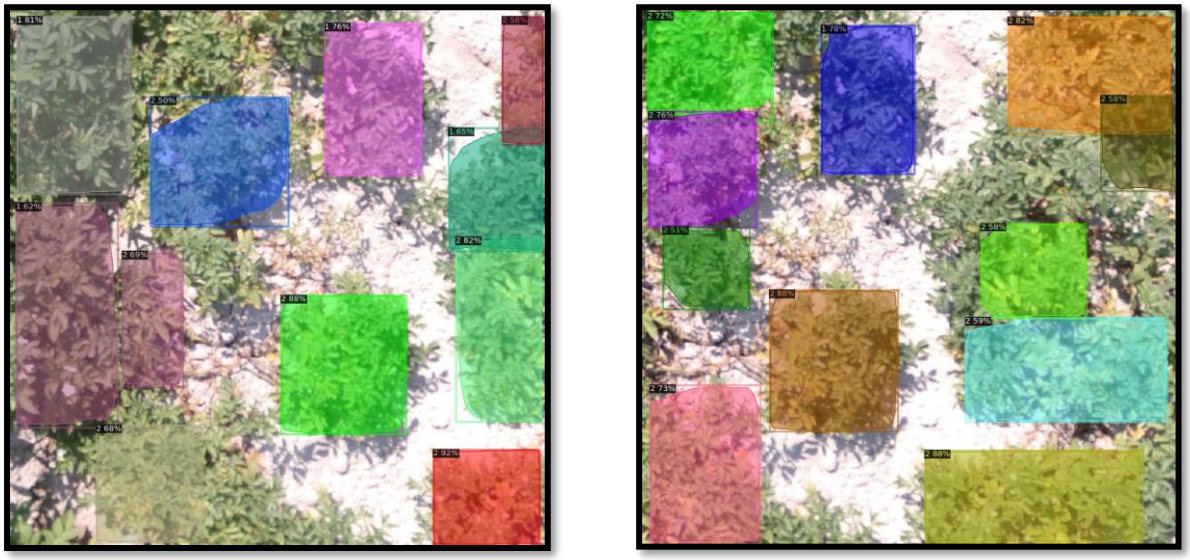


Figure 16: Output of the Mask-RCNN model

RESULTS

The first observation during the model training and testing is the number of instances in each class. The healthy class has around 8000 instances where the stressed class has more than 10,000 instances.

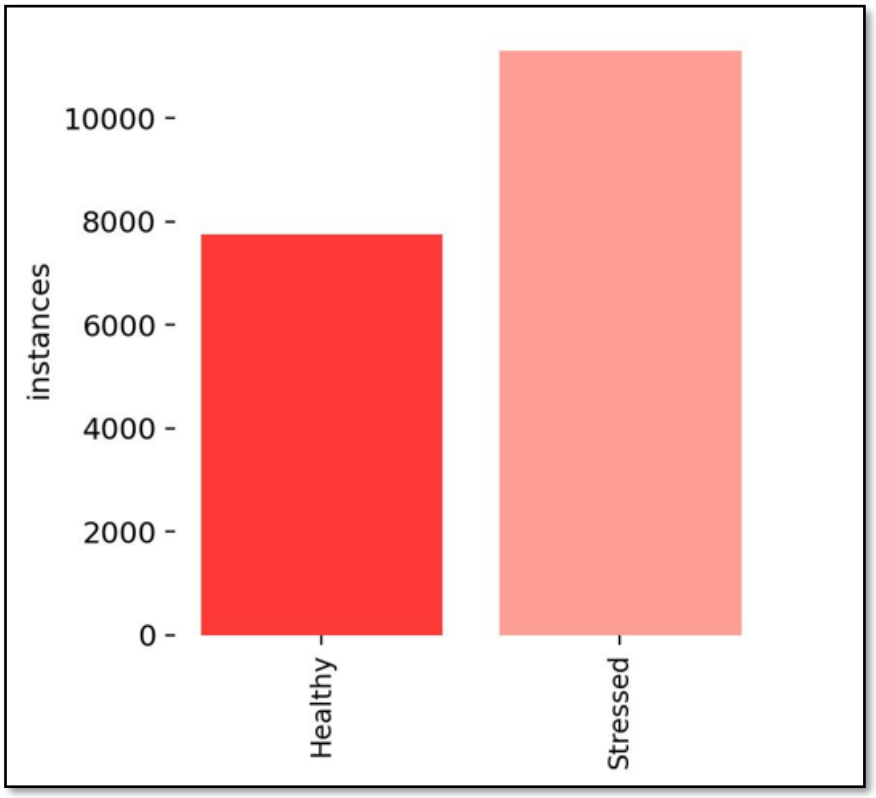


Figure 17: Class Instance Distribution

At first, the results have been discussed for each individual neural network approach, later a comparison of all the approaches are done to find the best approach.

- **Performance Metrices:**

I. **Accuracy:** In machine learning, accuracy is a metric that measures the model's ability to make correct predictions by determining the proportion of accurately classified instances.

Accuracy = Number of correct predictions / Total number of predictions

II. **Recall:** A model's recall, also known as sensitivity or true positive rate, quantifies its capability to correctly identify positive instances. It is calculated as the ratio of correctly identified positive outputs to the total number of actual positive instances.

Recall = True Positive / (True Positive + False Negative)

III. **Precision:** Precision is a metric that measures the ability of a model to accurately identify relevant items while minimizing false positives. It is calculated by dividing the

number of correctly identified positive outputs by the total number of predicted positive instances.

Precision = True Positive / (True Positive + False Negative)

IV. **F1 Score:** The F1 score is a commonly used metric in binary classification evaluation, particularly when there is an imbalance between the classes. It provides a balanced assessment of the model's performance by considering both precision and recall. By combining these two measures into a single score, the F1 score offers a comprehensive evaluation of the model's effectiveness.

*F1 score = 2 * (Precision * Recall) / (Precision + Recall)*

V. **mAP:** mAP (mean Average Precision) is a performance metric used to evaluate object detection models. It calculates the accuracy of the model by considering the Intersection over Union (IOU) between the predicted bounding boxes and the ground truth bounding boxes. The IOU represents the overlap between the two regions of interest. mAP is computed by comparing the intersection area of the true and predicted regions of interest to their union. It provides an overall measure of the model's detection performance across different IOU thresholds. For example, mAP50 refers to the mAP value when the IOU threshold is set to 0.5, while mAP50:90 represents the average mAP score across IOU thresholds ranging from 0.5 to 0.95.

YOLOV8:

Total 4 models were trained using YOLOv8 approach, where each model was trained for different number of epochs [37], which are 35,50,75,100 epochs. The time taken for each set of epochs to run is shown in the figure below:

Table 4: Comparison of the models for each set of epochs

No of Epochs	Time Taken(hours)	Validation Loss	Accuracy	mAP@50:95
35	1.15	0.198	94.72	0.946
50	1.65	0.158	95.39	0.975
75	2.48	0.129	95.72	0.990
100	3.30	0.108	96.30	0.992

- **For 35 epochs:**

The model trained for 35 epochs have the accuracy of 94.72%, with the average F1 score of 0.96 and mAP@50 of 0.992. The following figure shows the confusion matrix for the model trained for 35 epochs.

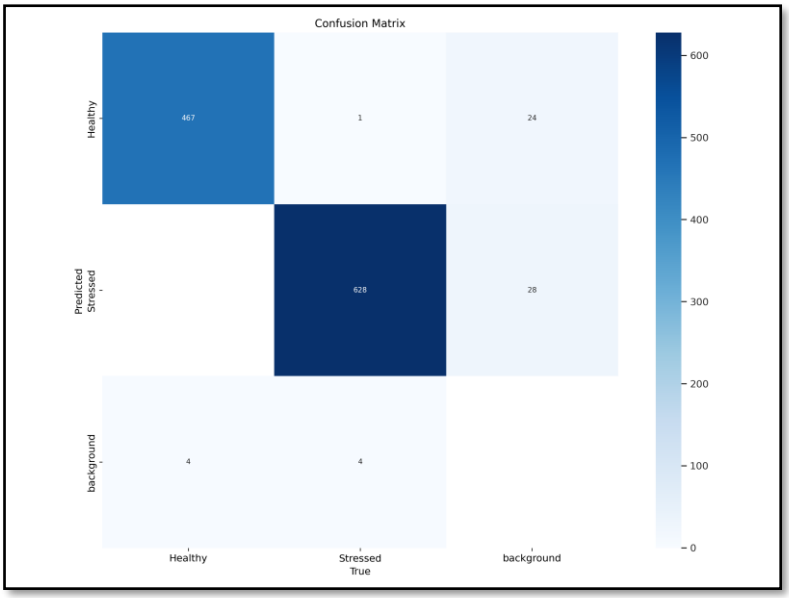


Figure 18: Confusion Matrix of the Model trained for 35 Epochs

The following figure shows the F1 curve, precision curve, Recall curve and Precision-Recall curve.

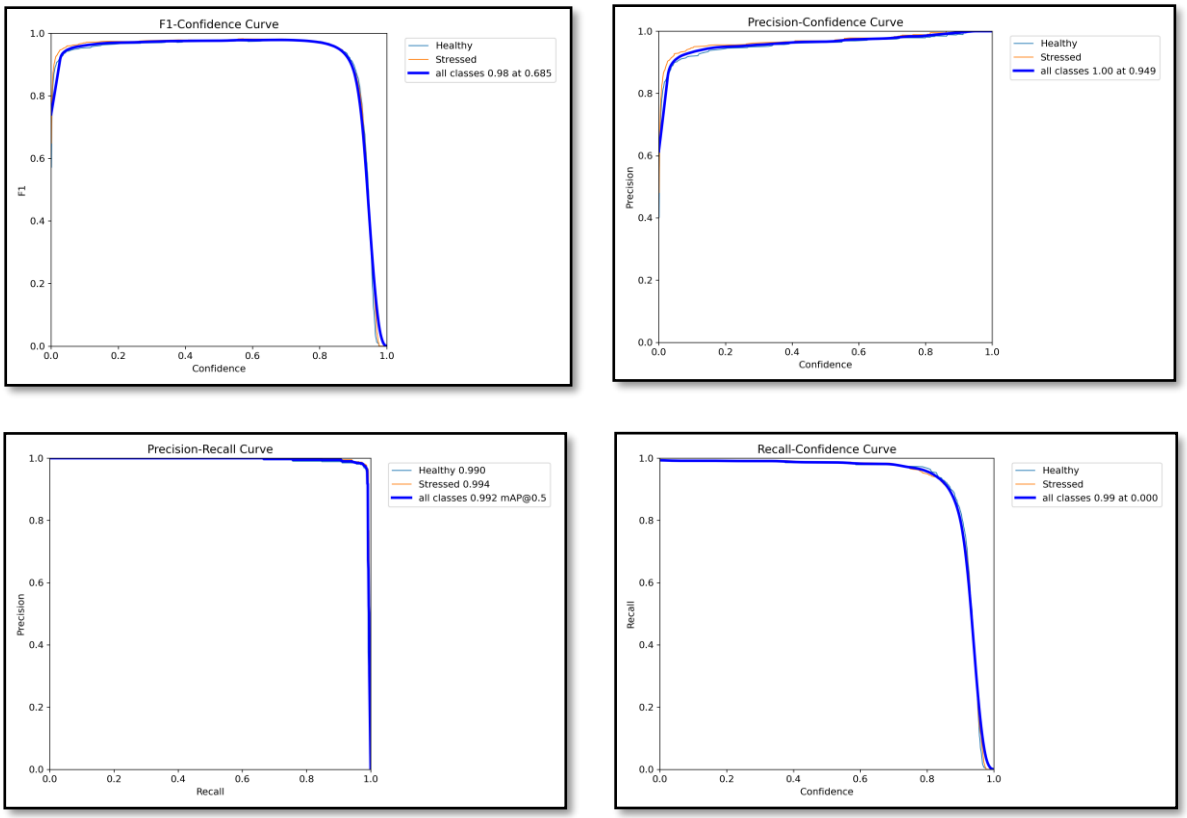


Figure 19: Different Evaluation metrices vs Confidence graphs for 35 epochs

The following figure shows the all the losses, and metrices for both train and validation.

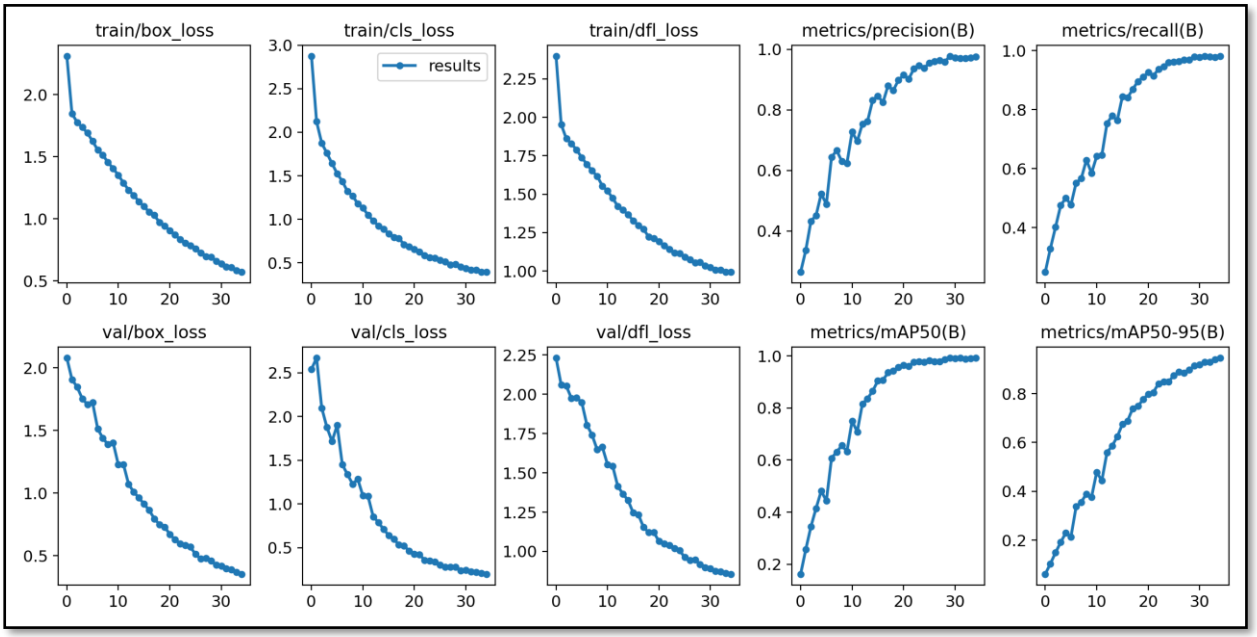


Figure 20: Graphs showing the different parameter values till 35 epochs

- **For 50 epochs:**

The model trained for 50 epochs have the accuracy of 95.39%, with the average F1 score of 0.97 and mAP@50 of 0.993. The following figure shows the confusion matrix for the model trained for 50 epochs.

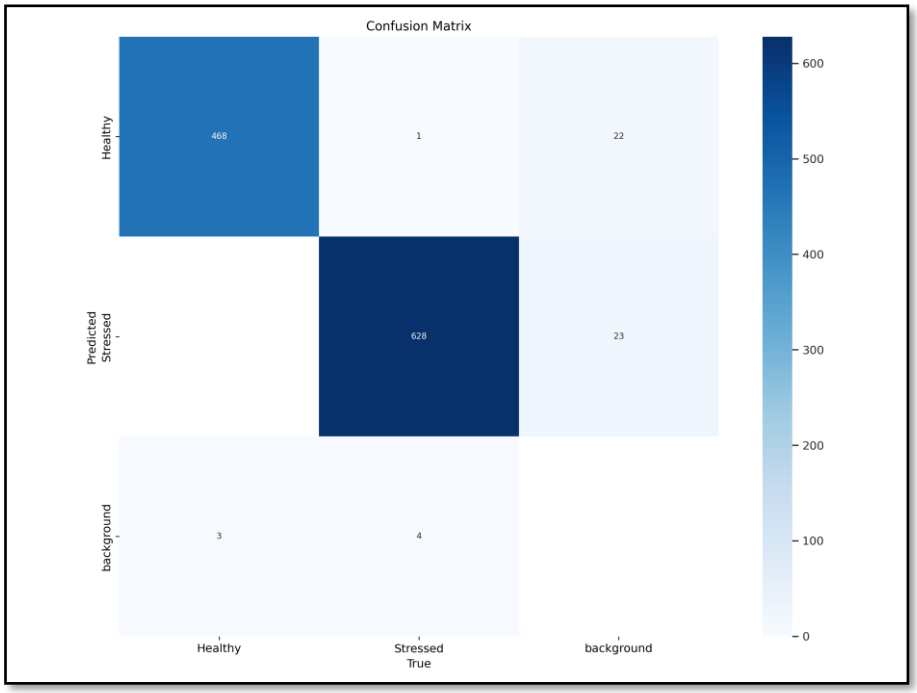


Figure 21: Confusion Matrix of the Model trained for 50 Epochs

The following figure shows the F1 curve, precision curve, Recall curve and Precision-Recall curve.

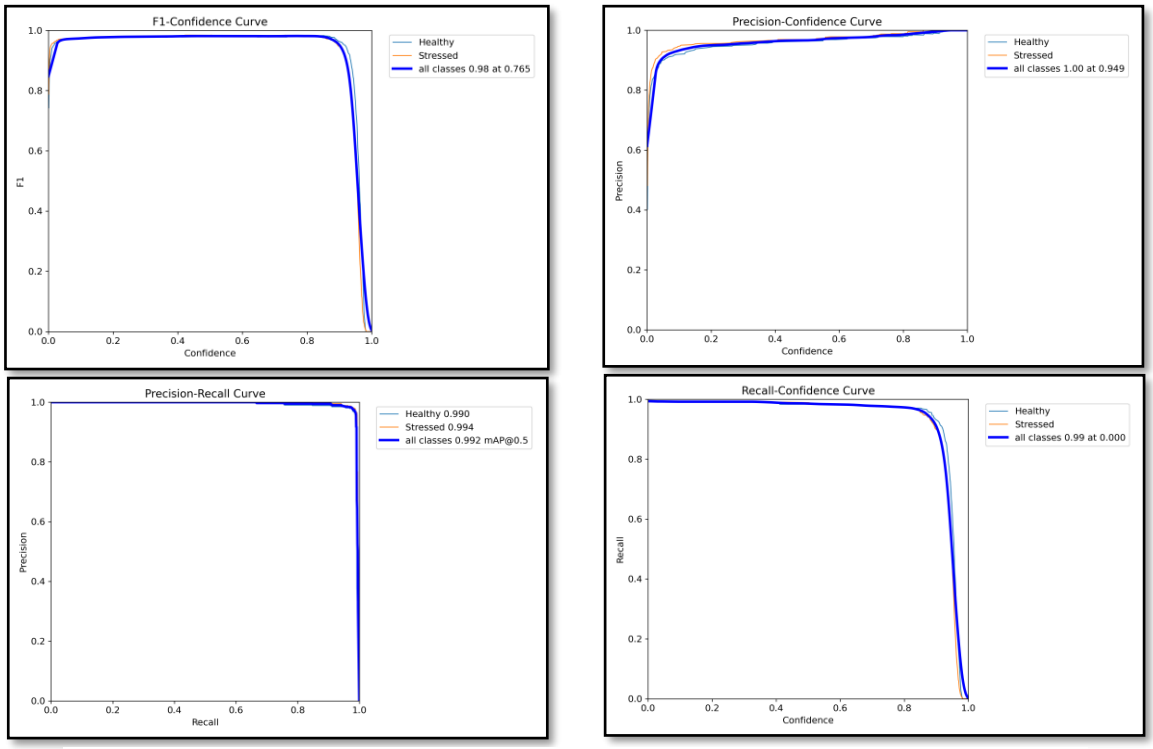


Figure 22: Different Evaluation metrices vs Confidence graphs for 50 epochs

The following figure shows the all the losses, and metrices for both train and validation.

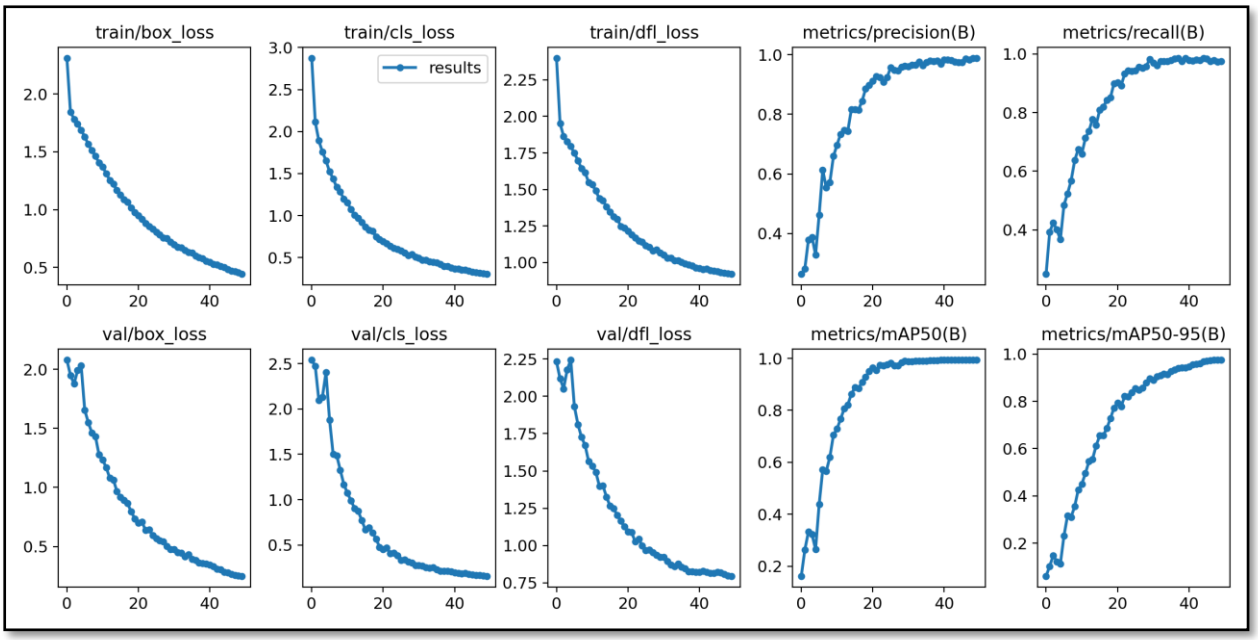


Figure 23: Graphs showing the different parameter values till 50 epochs

- **For 75 epochs:**

The model trained for 75 epochs have the accuracy of 95.72%, with the average F1 score of 0.97 and mAP@50 of 0.994. The following figure shows the confusion matrix for the model trained for 75 epochs.

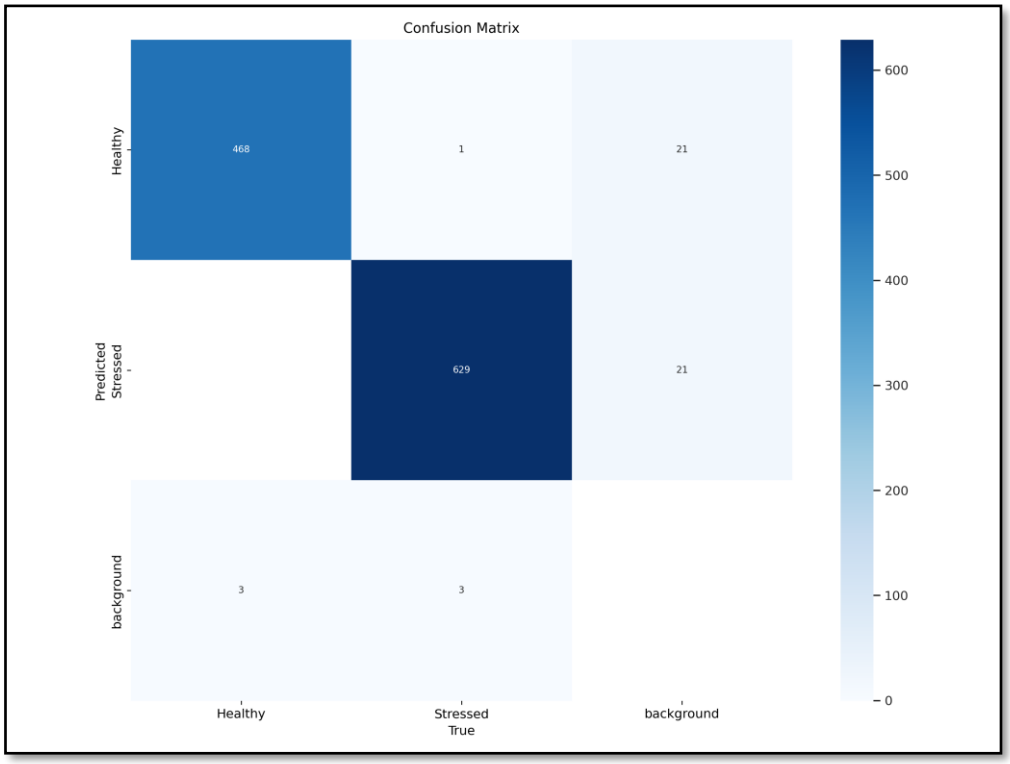


Figure 24: Confusion Matrix of the Model trained for 75 Epochs

The following figure shows the F1 curve, precision curve, Recall curve and Precision-Recall curve.

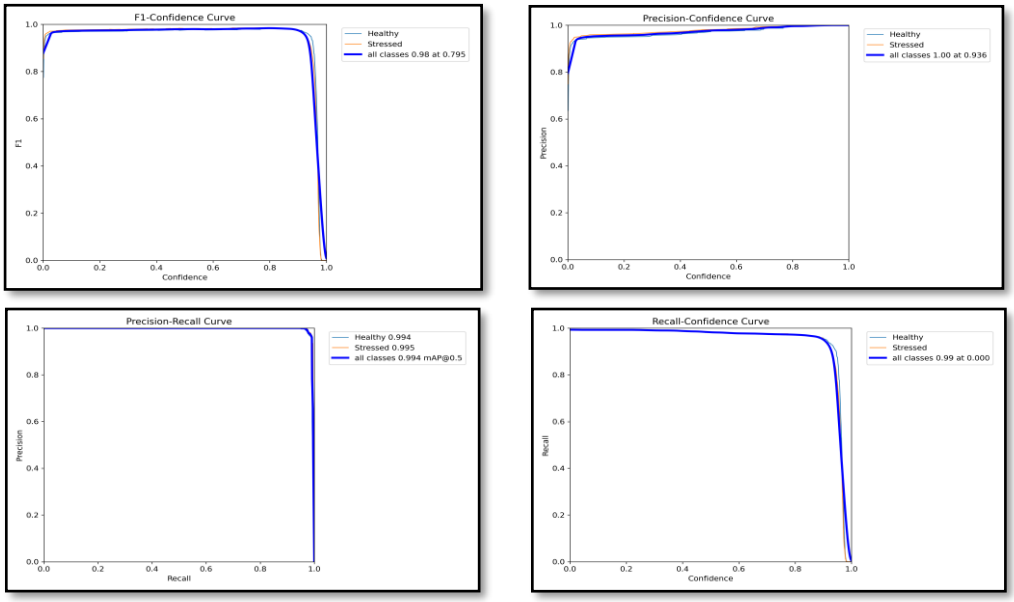


Figure 25: Different Evaluation metrices vs Confidence graphs for 75 epochs

The following figure shows the all the losses, and metrices for both train and validation.

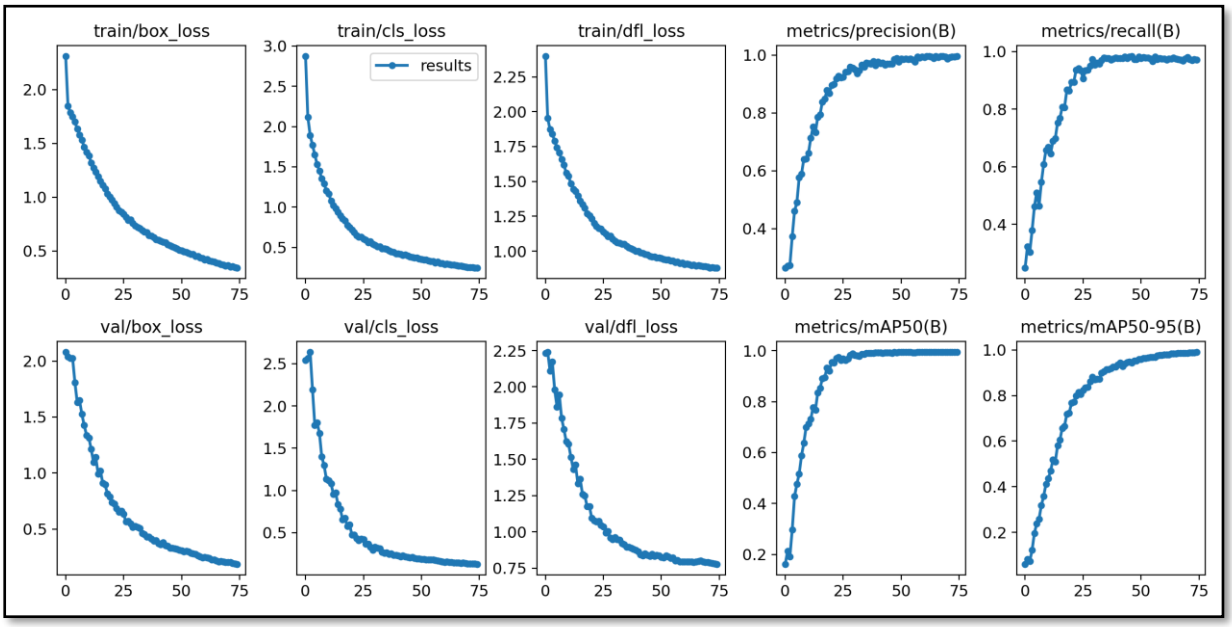


Figure 26: Graphs showing the different parameter values till 75 epochs

- **For 100 epochs:**

The model trained for 100 epochs have the accuracy of 96.30%, with the average F1 score of 0.97 and mAP@50 of 0.995. The following figure shows the confusion matrix for the model trained for 100 epochs.

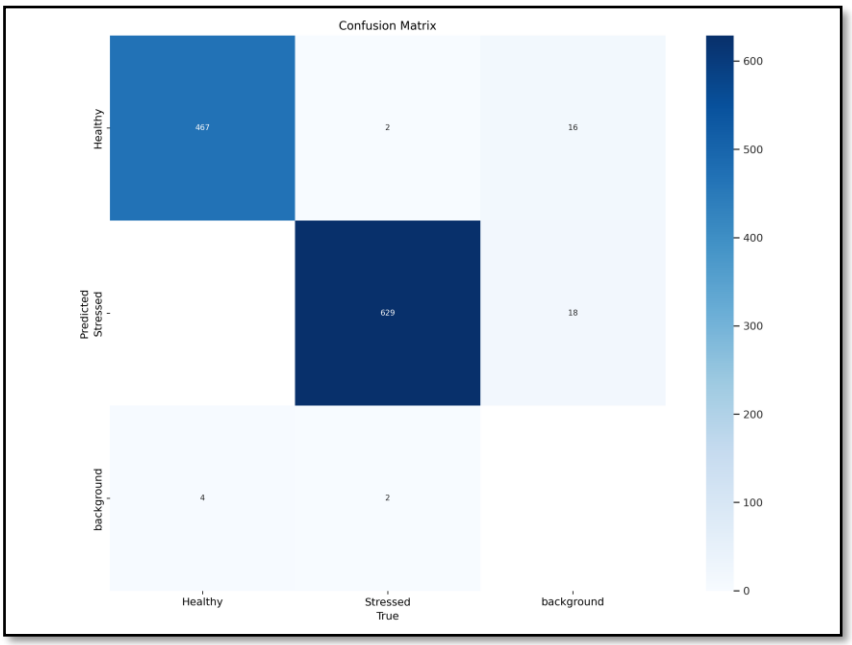


Figure 27: Confusion Matrix of the Model trained for 100 Epochs

The following figure shows the F1 curve, precision curve, Recall curve and Precision-Recall curve.

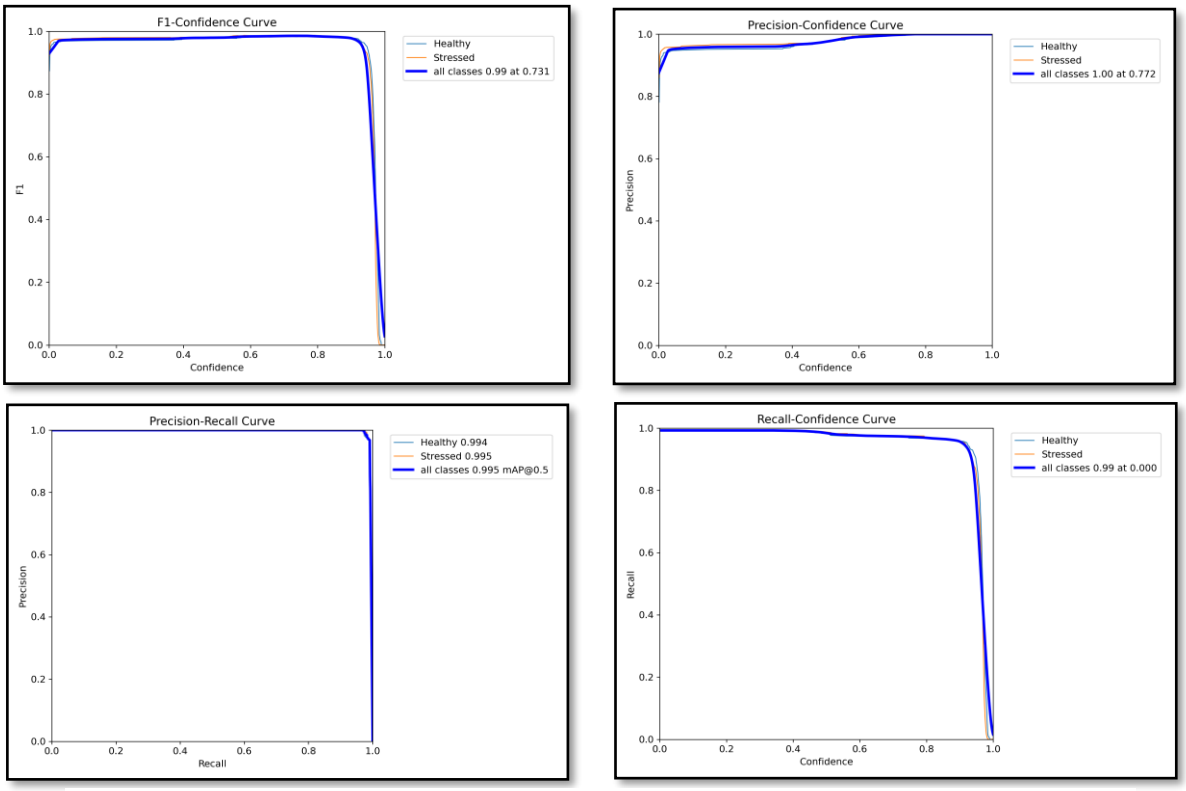


Figure 28: Different Evaluation metrices vs Confidence graphs for 75 epochs

The following figure shows the all the losses, and metrices for both train and validation.

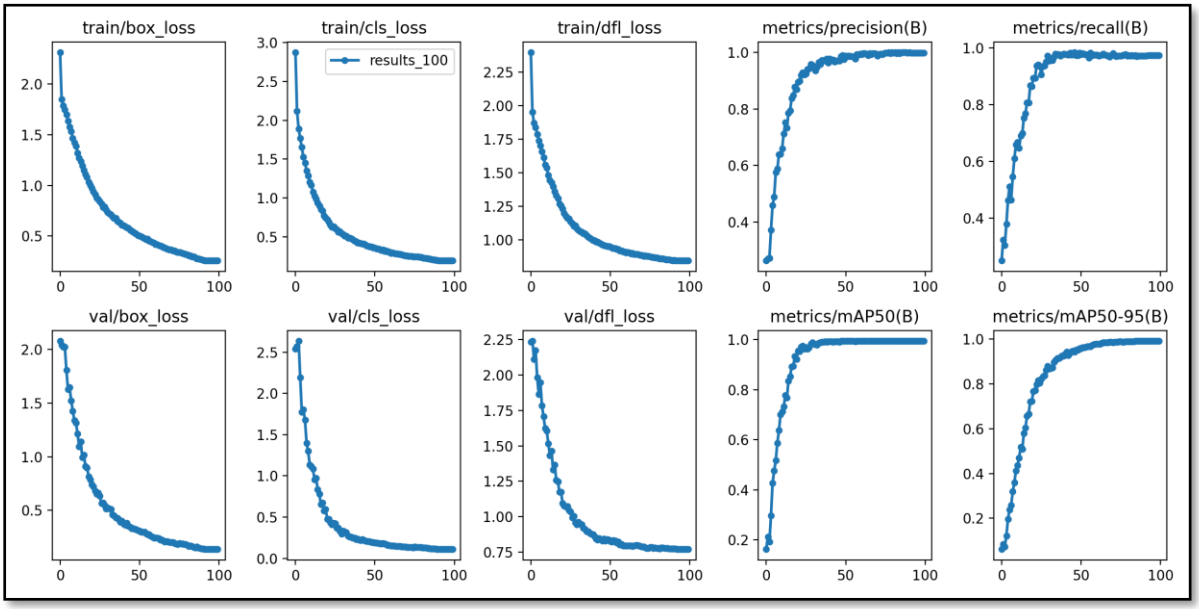


Figure 29: Graphs showing the different parameter values till 100 epochs

The model trained for 100 epochs gives the highest accuracy among all the YOLOV8 models.

SSD Object Detection:

In SSD object, the model was trained for 20000 steps. After training the accuracy of the model is 94.80% with the average F1 score of 0.94 and mAP@50 of 0.944. The following figure shows the confusion matrix for the model trained.

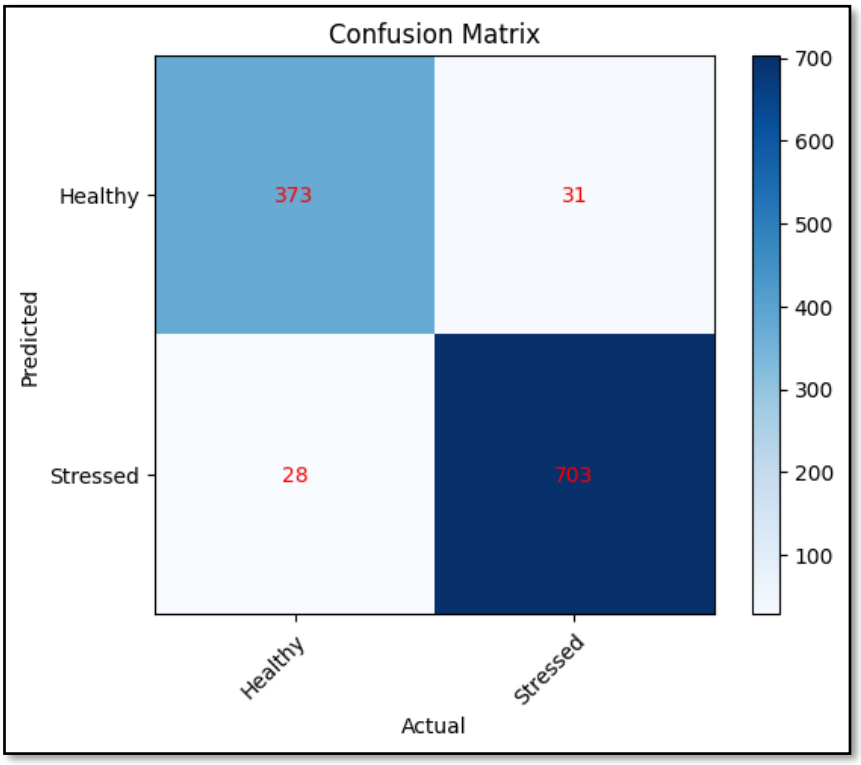


Figure 30: Confusion Matrix for the SSD Model

The following figure shows the all the losses for the trained model.

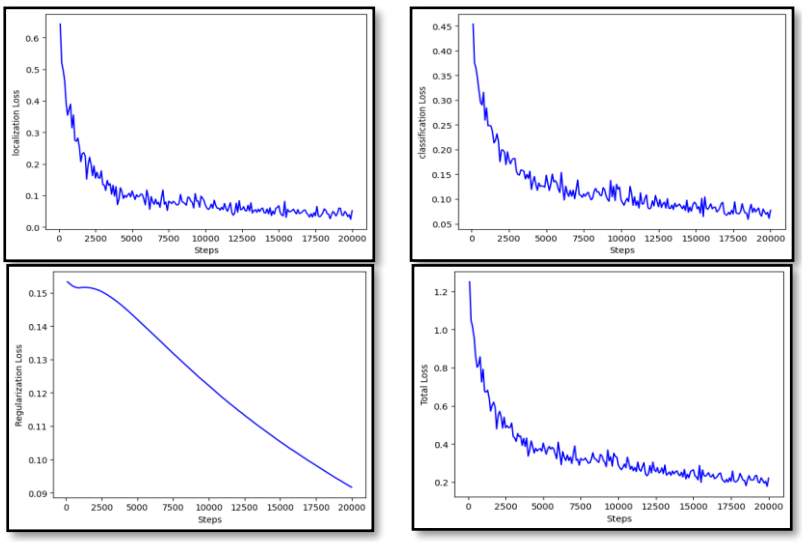


Figure 31: Graphs showing the different parameter values for 20000 steps

Faster-RCNN:

In Faster-RCNN, the model was trained for 3000 steps. After training the accuracy of the model is 95.68% with the average F1 score of 0.95 and mAP@50 of 0.95. The following figure shows the confusion matrix for the model trained.

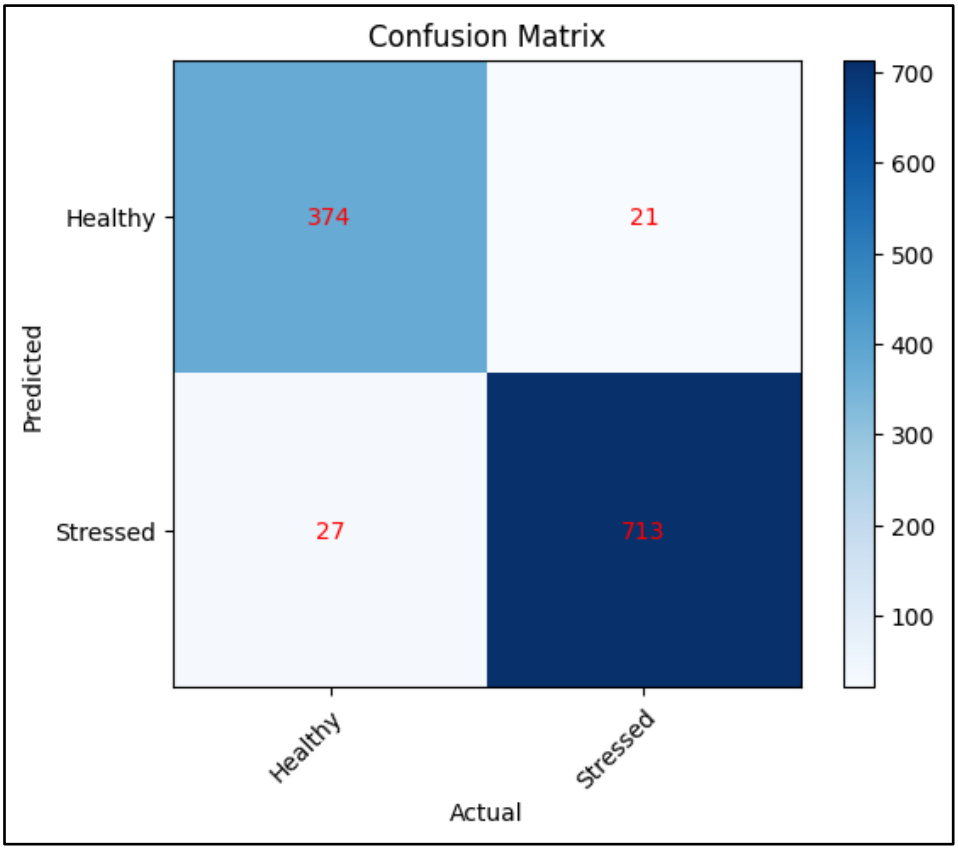


Figure 32: Confusion Matrix for the Faster-RCNN Model

The following figure shows the all the losses for the trained model.

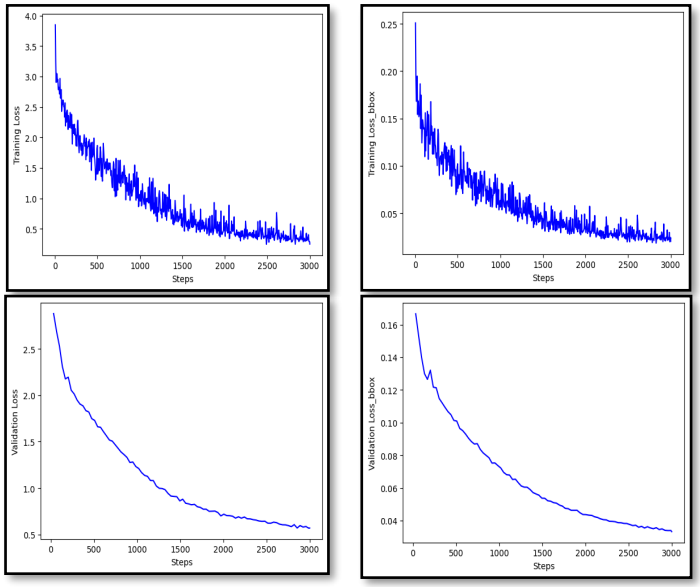


Figure 33: Graphs showing the different parameter values for 20000 steps

Mask-RCNN:

In Faster-RCNN, the model was trained for 3000 steps. After training the accuracy of the model is 95.77% with the average F1 score of 0.94 and mAP@50 of 0.95. The following figure shows the confusion matrix for the model trained.

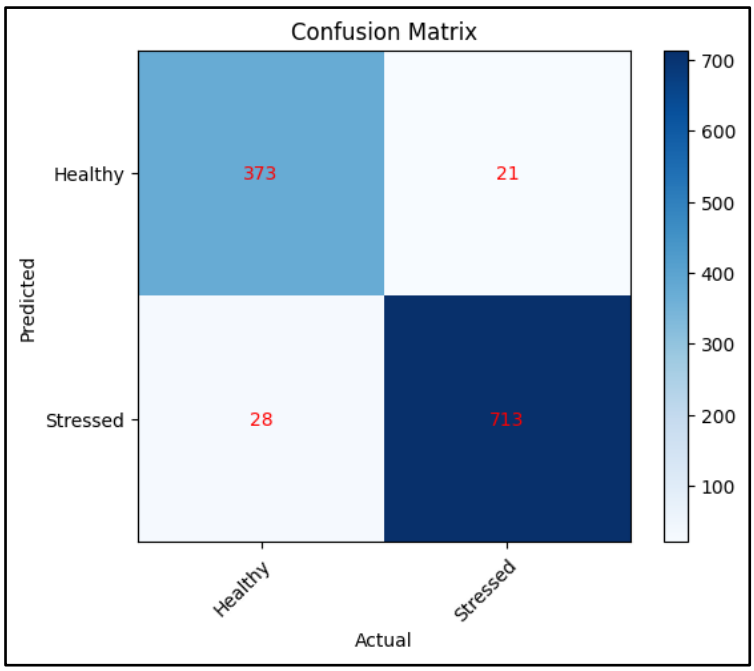


Figure 32: Confusion Matrix for the Mask-RCNN Model

The following figure shows the all the losses for the trained model.

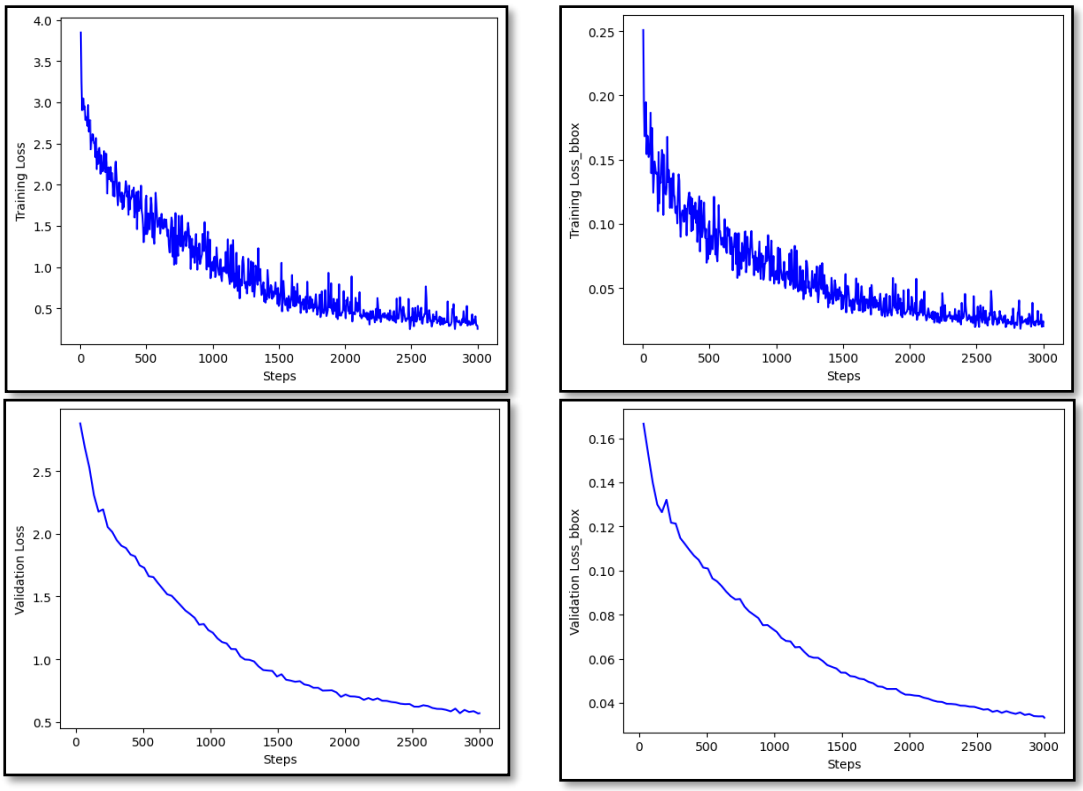


Figure 35: Graphs showing the different parameter values for 20000 steps

From the above comparisons we can see that YOLOV8 have the highest performance whereas SSD have the lowest Performance. The following Table shows the comparison between all the models.

Models	Accuracy	F1 score	mAP@50
YOLOV8	0.9630	0.97	0.99
SSD	0.9480	0.94	0.94
Faster-RCNN	0.9566	0.95	0.95
Mask-RCNN	0.9577	0.94	0.95

Table 5: Comparison of Different Models

Using Vegetative Indices for validating the result on Test data:

Vegetative indices [38] are spectral image transformations that combine two or more image bands to enhance the representation of vegetation-related characteristics. These indices aim to improve the accuracy of assessing changes in terrestrial photosynthetic activity and canopy structure over time and across geographical regions. By analysing the values of vegetative indices, it becomes possible to potentially distinguish healthy and stressed plant areas. This analysis can serve as a means to validate the performance of the model on test data. In this project, three vegetative indices have been considered: the Normalized Difference Vegetative Index (NDVI), the Difference Vegetative Index (VDI), and the Green Normalized Difference Vegetative Index (GNDVI).

NDVI:

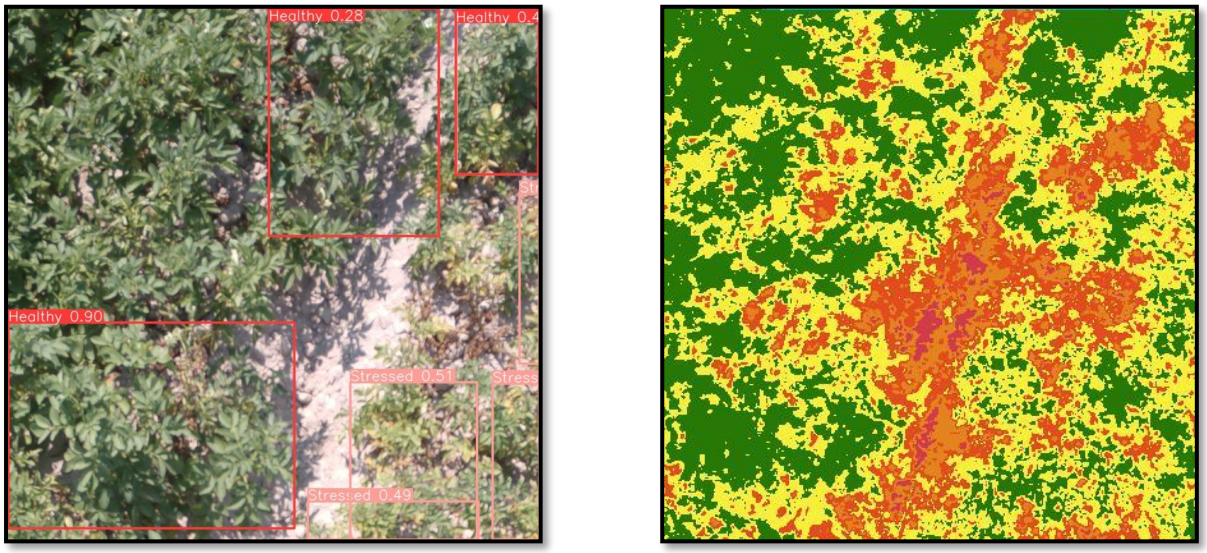
The Normalised Difference Vegetation Index (NDVI) is a numerical indicator frequently used in remote sensing and satellite images analysis to measure and monitor the health and vigour of vegetation. It is based on the idea that healthy vegetation absorbs the majority of visible light (mostly in the red spectrum) for photosynthesis while reflecting a large proportion of near-infrared (NIR) radiation.

$$NDVI = (NIR - Red) / (NIR + Red)$$

The terms "NIR" and "Red" in this formula refer to spectral reflectance in the near-infrared band and spectral reflectance in the red band, respectively. The NDVI value ranges from -1 to +1, with higher values suggesting a denser and healthier plant cover.

a. YOLOV8

In the figure below, in the image (a), we have the output of the YOLO model and in image (b), we can see the NDVI index of the same test image. By comparing the sections where the model has labelled as stressed or healthy, with the NDVI index, we can observe a high correlation between them.

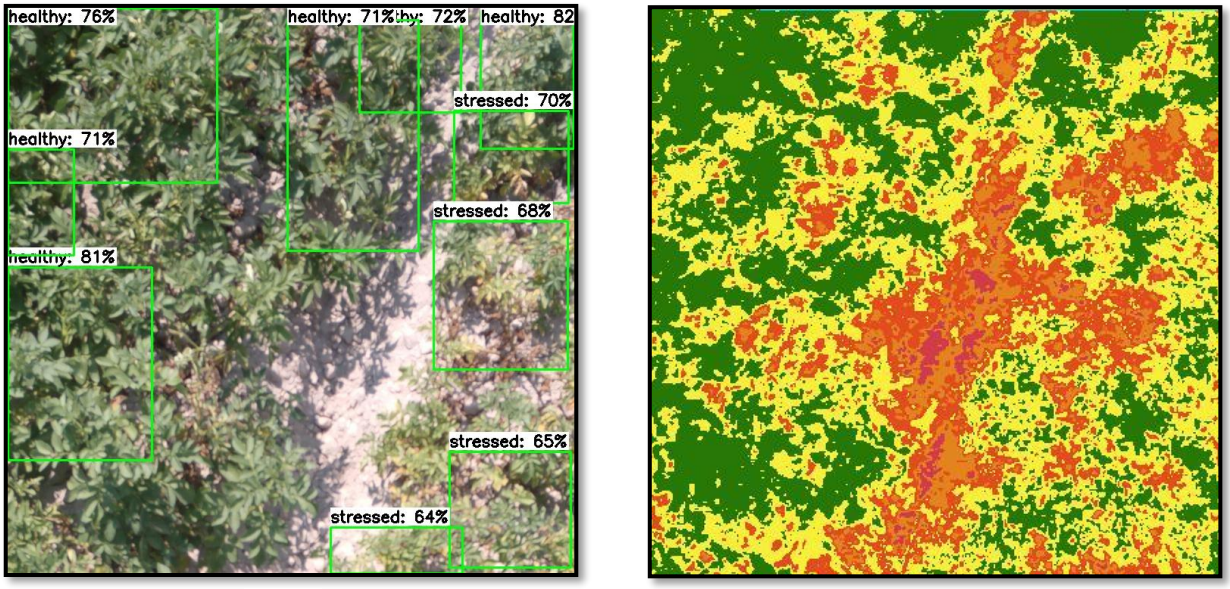


(a) *(b)*

Figure 36: comparison between YOLO Model's output and NDVI Vegetative Index

b. SSD:

In the figure below, in the image (a), we have the output of the SSD model and in image (b), we can see the NDVI index of the same test image. By comparing the sections where the model has labelled as stressed or healthy, with the NDVI index, we can observe a high correlation between them

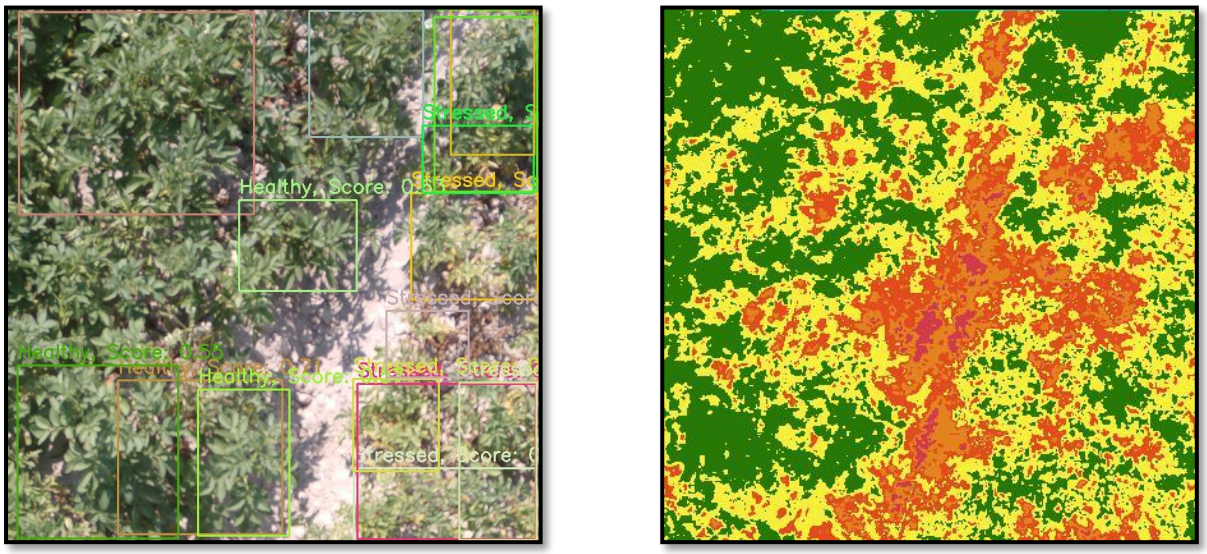


(a) *(b)*

Figure 37: comparison between SSD Model's output and NDVI Vegetative Index

c. Faster-RCNN:

In the figure below, in the image (a), we have the output of the Faster-RCNN model and in image (b), we can see the NDVI index of the same test image. By comparing the sections where the model has labelled as stressed or healthy, with the NDVI index, we can observe a high correlation between them.

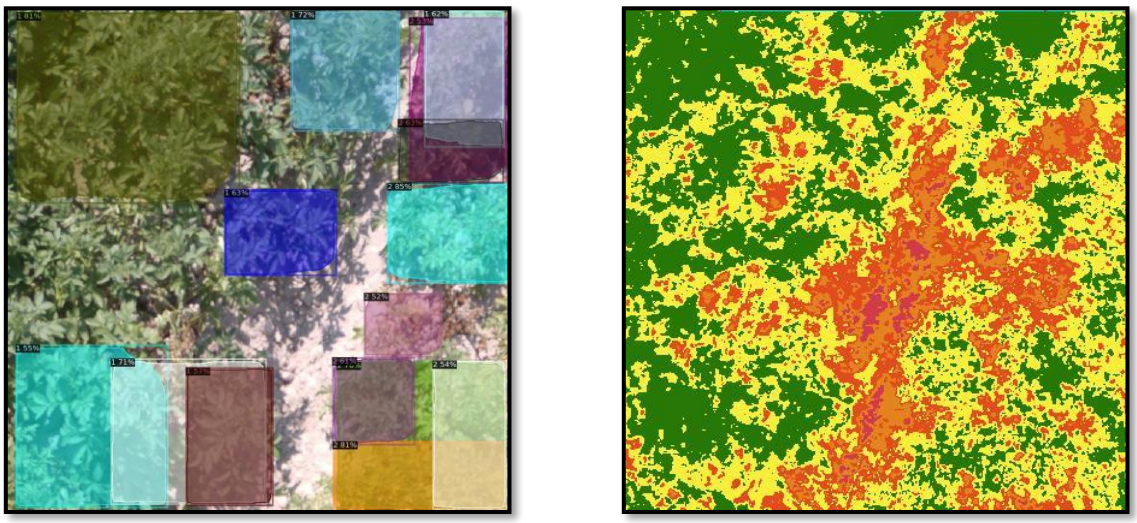


(a) *(b)*

Figure 38: comparison between Faster-RCNN Model's output and NDVI Vegetative

d. Mask-RCNN:

In the figure below, in the image (a), we have the output of the Mask-RCNN model and in image (b), we can see the NDVI index of the same test image. By comparing the sections where the model has labelled as stressed or healthy, with the NDVI index, we can observe a high correlation between them.



(a) *(b)*

Figure 39: comparison between Mask-RCNN Model's output and NDVI

DVI:

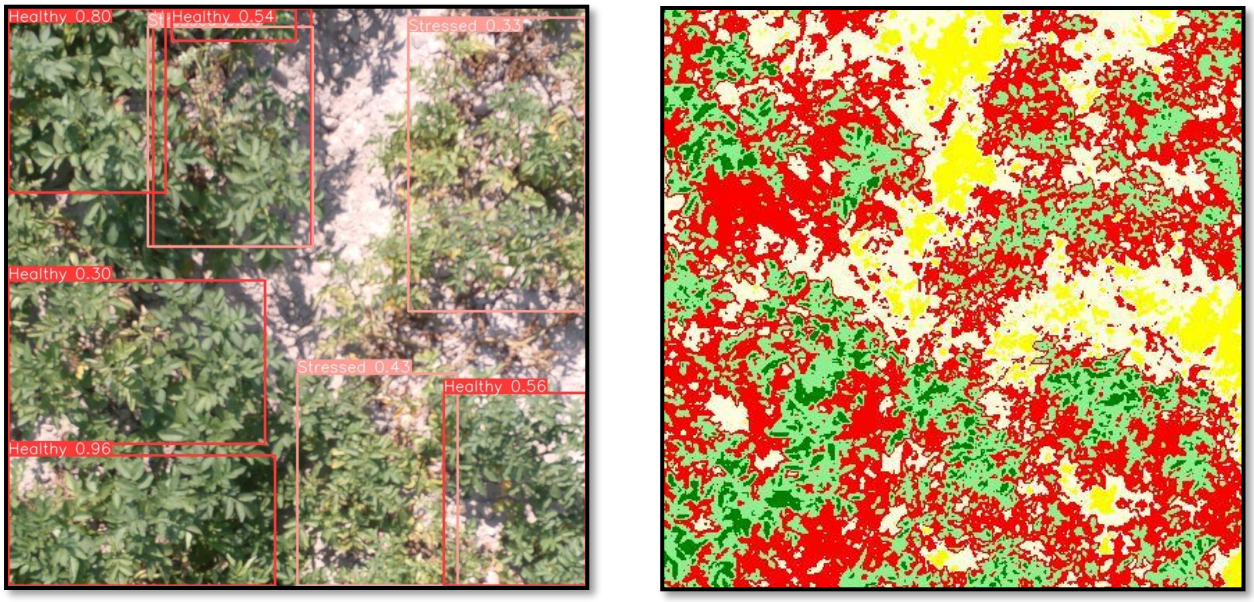
DVI is an abbreviation for Difference Vegetation Index. It is yet another vegetation indicator used in remote sensing and satellite picture analysis to assess the health and vigour of plants. DVI is determined using the difference between the green and red bands, as opposed to NVDI, which uses both the red and near-infrared bands.

$$DVI = (NIR - Red)$$

The terms "NIR" and "Red" in this formula refer to spectral reflectance in the near-infrared band and spectral reflectance in the red band, respectively.

a. YOLOV8:

In the figure below, in the image (a), we have the output of the YOLO model and in image (b), we can see the DVI index of the same test image. By comparing the sections where the model has labelled as stressed or healthy, with the DVI index, we can observe a high correlation between them.

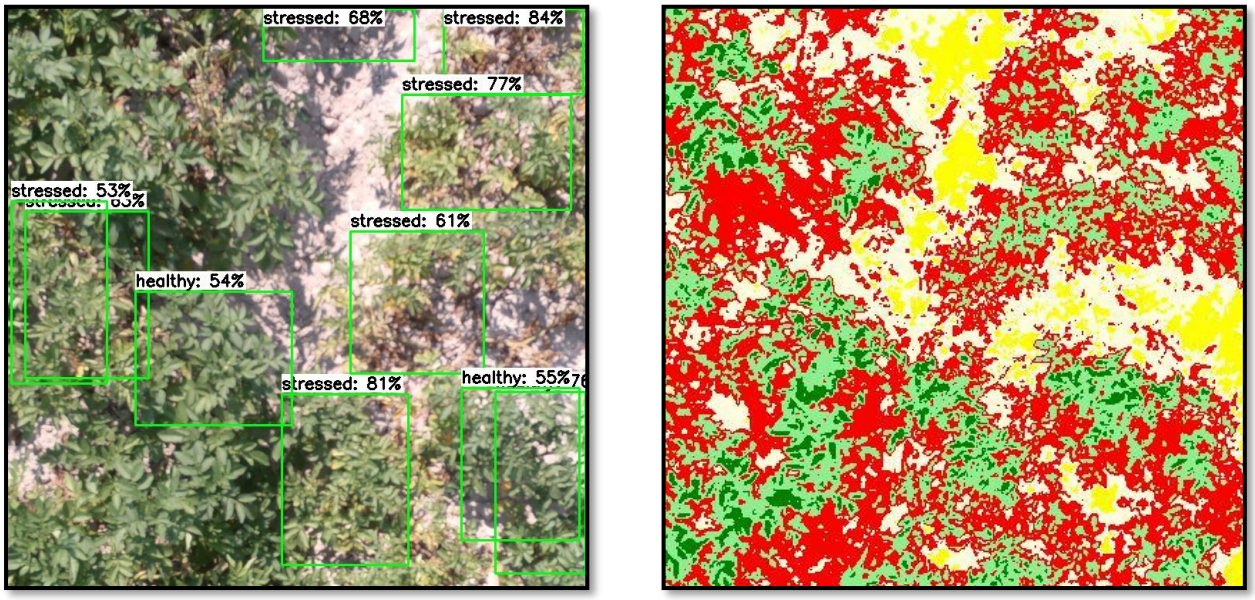


(a) *(b)*

Figure 40: comparison between YOLO Model's output and DVI Vegetative Index

b. SSD:

In the figure below, in the image (a), we have the output of the SSD model and in image (b), we can see the DVI index of the same test image. By comparing the sections where the model has labelled as stressed or healthy, with the DVI index, we can observe a high correlation between them.

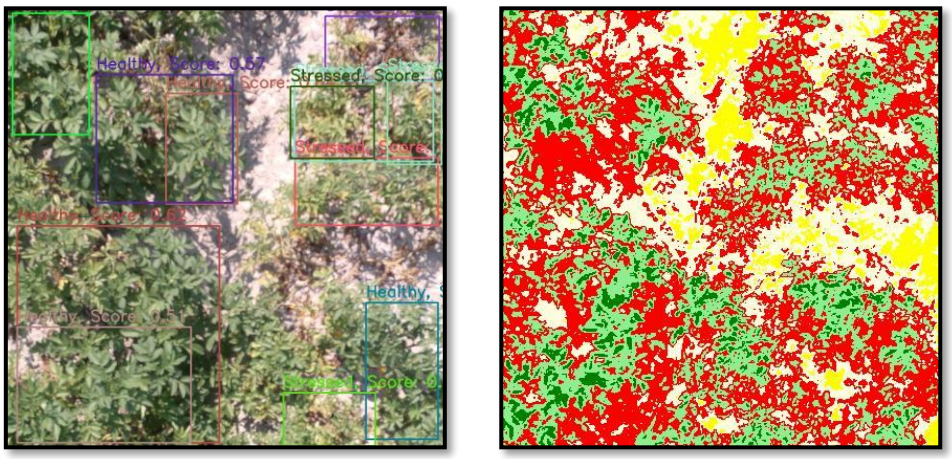


(a) *(b)*

Figure 41: comparison between SSD Model's output and DVI Vegetative Index

c. Faster-RCNN:

In the figure below, in the image (a), we have the output of the Faster-RCNN model and in image (b), we can see the DVI index of the same test image. By comparing the sections where the model has labelled as stressed or healthy, with the DVI index, we can observe a high correlation between them.

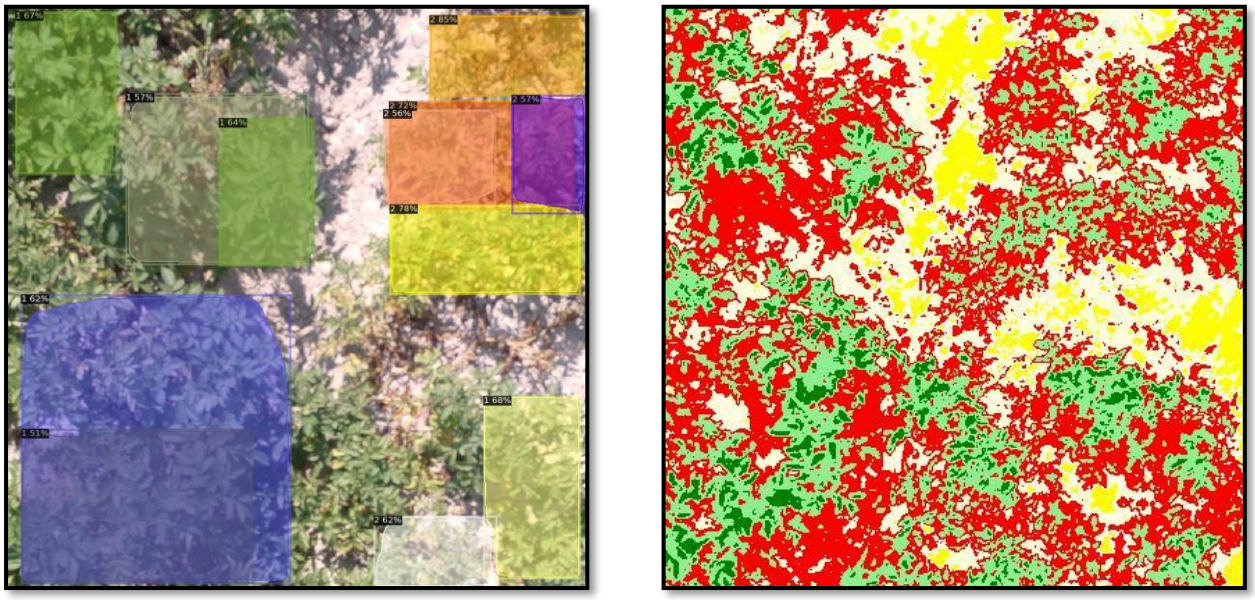


(a) *(b)*

Figure 42: comparison between Faster-RCNN Model's output and DVI Vegetative Index

d. Mask-RCNN:

In the figure below, in the image (a), we have the output of the Mask-RCNN model and in image (b), we can see the DVI index of the same test image. By comparing the sections where the model has labelled as stressed or healthy, with the DVI index, we can observe a high correlation between them.



(a) *(b)*

Figure 43: comparison between Mask-RCNN Model's output and DVI Vegetative

GNDVI:

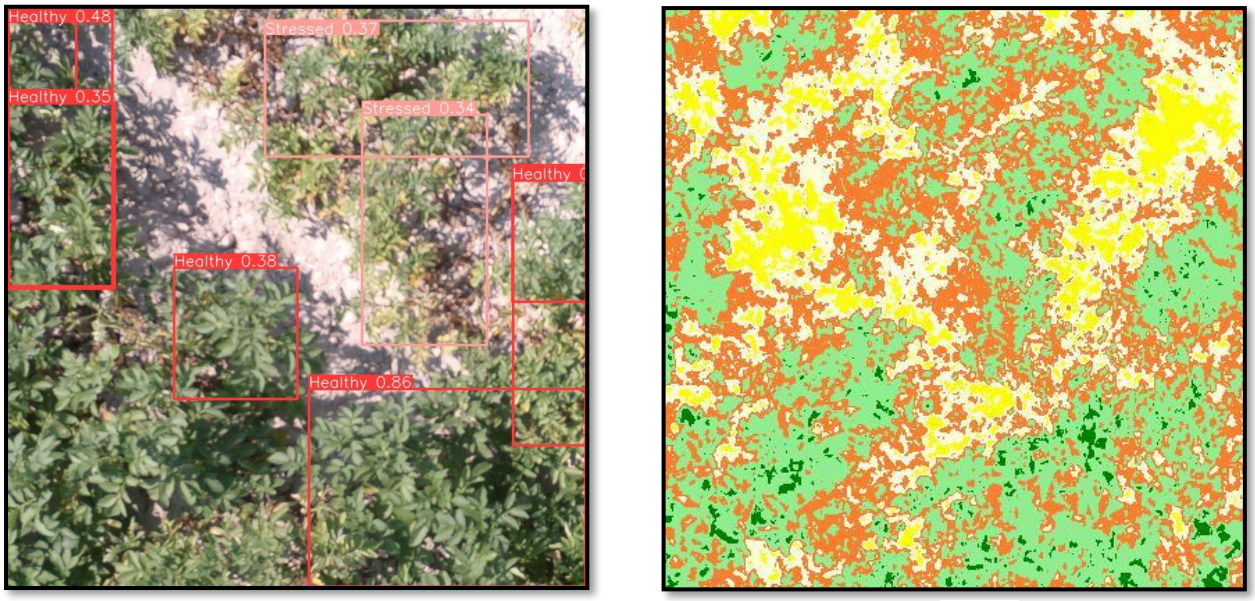
GNDVI is an abbreviation for Green Normalised Difference Vegetation Index. It is a vegetation index that is similar to NVDI but is derived particularly utilising the green and near-infrared wavelengths. GNDVI measures the difference in reflectance between these two bands to estimate vegetation health and density.

$$GNDVI = (NIR - Green) / (NIR + Green)$$

The terms "NIR" and "Green" in this formula refer to spectral reflectance in the near-infrared band and spectral reflectance in the green band, respectively. GNDVI's final value, like other vegetation indices, runs from -1 to +1, with higher positive values indicating healthier and denser vegetation.

a. YOLOV8:

In the figure below, in the image (a), we have the output of the YOLO model and in image (b), we can see the GNDVI index of the same test image. By comparing the sections where the model has labelled as stressed or healthy, with the GNDVI index, we can observe a high correlation between them.

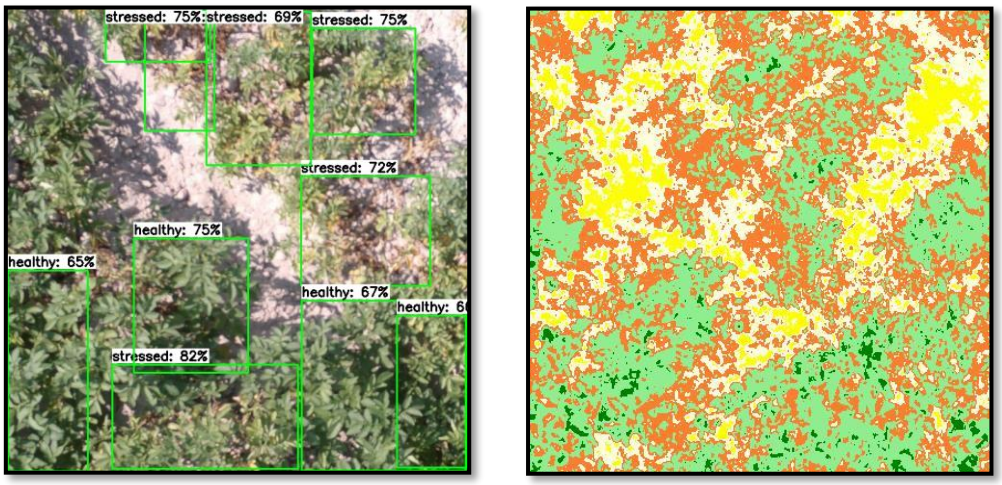


(a) *(b)*

Figure 44: comparison between SSD Model's output and GNDVI Vegetative Index

b. SSD:

In the figure below, in the image (a), we have the output of the SSD model and in image (b), we can see the GNDVI index of the same test image. By comparing the sections where the model has labelled as stressed or healthy, with the GNDVI index, we can observe a high correlation between them.

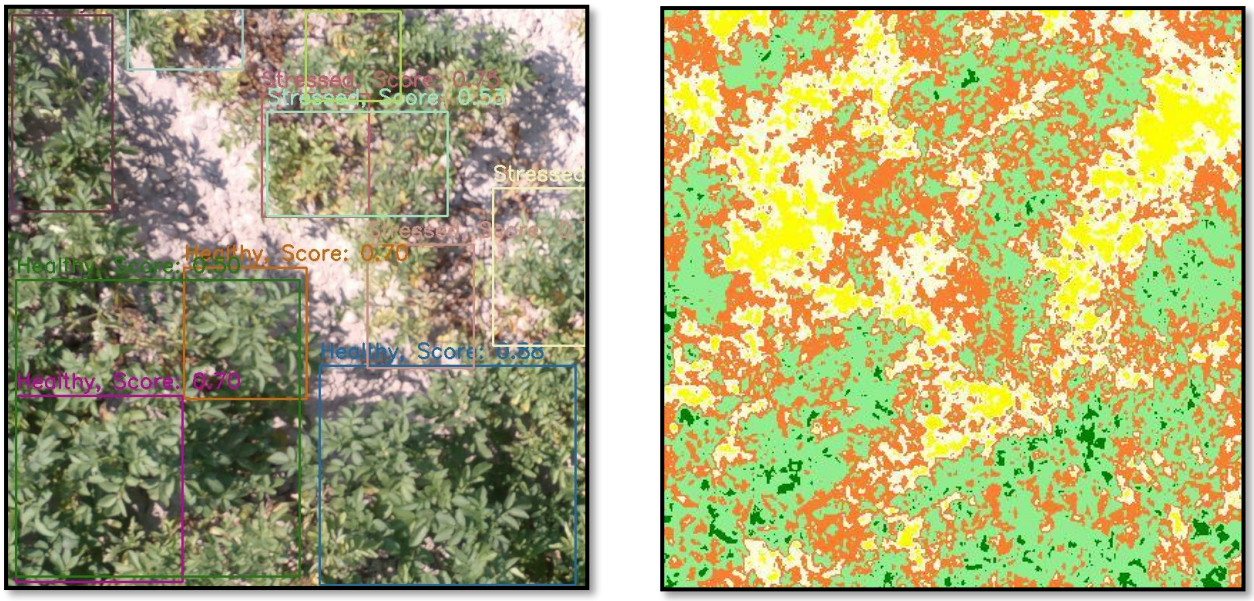


(a) *(b)*

Figure 45: comparison between SSD Model's output and GNDVI

c. Faster-RCNN:

In the figure below, in the image (a), we have the output of the Faster-RCNN model and in image (b), we can see the GNDVI index of the same test image. By comparing the sections where the model has labelled as stressed or healthy, with the GNDVI index, we can observe a high correlation between them.

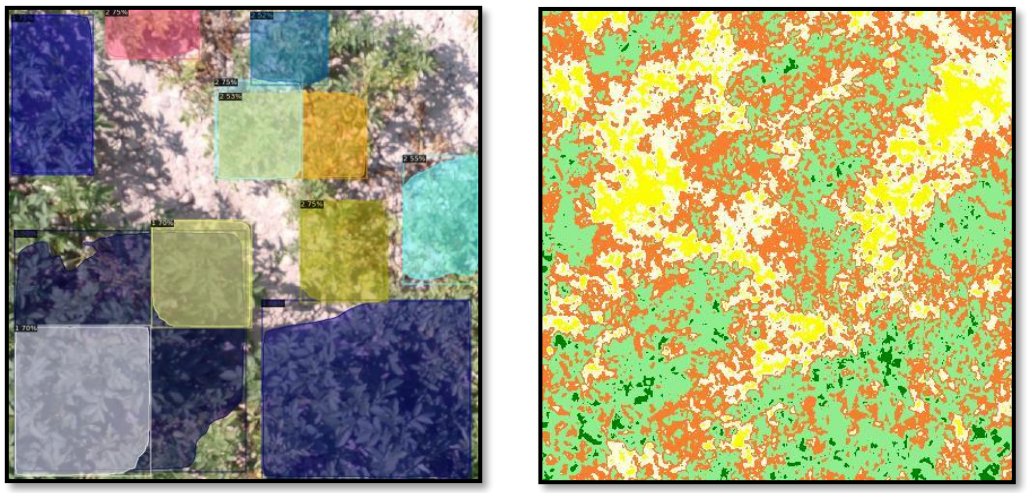


(a) *(b)*

Figure 46: comparison between Faster-RCNN Model's output and GNDVI Vegetative

d. Mask-RCNN:

In the figure below, in the image (a), we have the output of the Mask-RCNN model and in image (b), we can see the GNDVI index of the same test image. By comparing the sections where the model has labelled as stressed or healthy, with the GNDVI index, we can observe a high correlation between them.



(a) *(b)*

Figure 47: comparison between Mask-RCNN Model's output and GNDVI

Implementing The GUI for the Models:

The GUI was implemented using Streamlit python Library. It will take image from the user and will detect the classes on the image and will display the detected image on the screen. The following image shows How the output will be shown in the GUI.

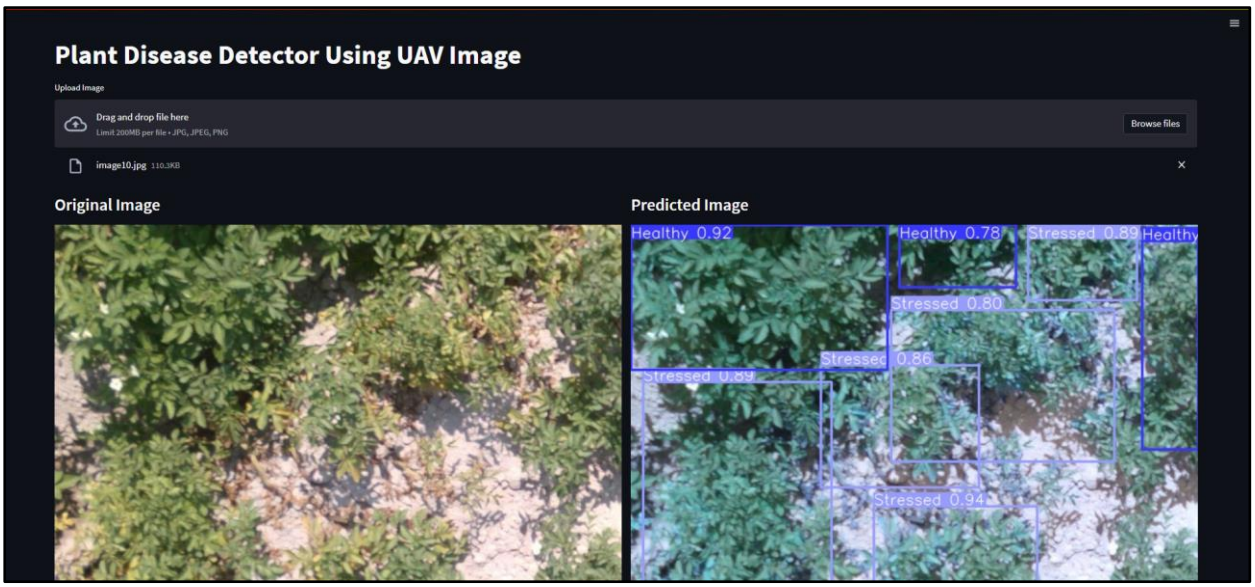


Figure 48: GUI implementation of the Models

DISCUSSION

A comparison was conducted between various methods found in existing literature and the proposed approach. The current solution incorporates models like Random Forest Classification, CNN, and others, which, although effective, fail to provide the precise location of stressed plants in the photos. This limitation poses a challenge as the model may misclassify the image. In contrast, when the image is manually processed by human intelligence, a bounding box can be drawn around the target plant, facilitating the model's determination of its health status. The comparison of multiple papers is presented in Table 1. However, the existing system lacks several capabilities that AI-based solutions offer for crop health monitoring. Deep learning techniques, such as convolutional neural networks (CNN), can significantly expedite image processing by leveraging their initial layers for critical feature extraction. Notably, the reviewed research does not explore state-of-the-art methodologies like YOLOv8. To overcome these limitations, the proposed approach incorporates models such as YOLOv8, SSD, Faster-RCNN, and Mask-RCNN. These models achieve high accuracy levels ranging from 94% to 96%. Specifically, the YOLOv5 model stands out for its rapid detection capabilities compared to other models, making it well-suited for real-time identification of stressed plants. This advancement paves the way for the development of mobile applications dedicated to real-time disease detection. The following graph shows the comparison of all the algorithms in terms of evolution measures.

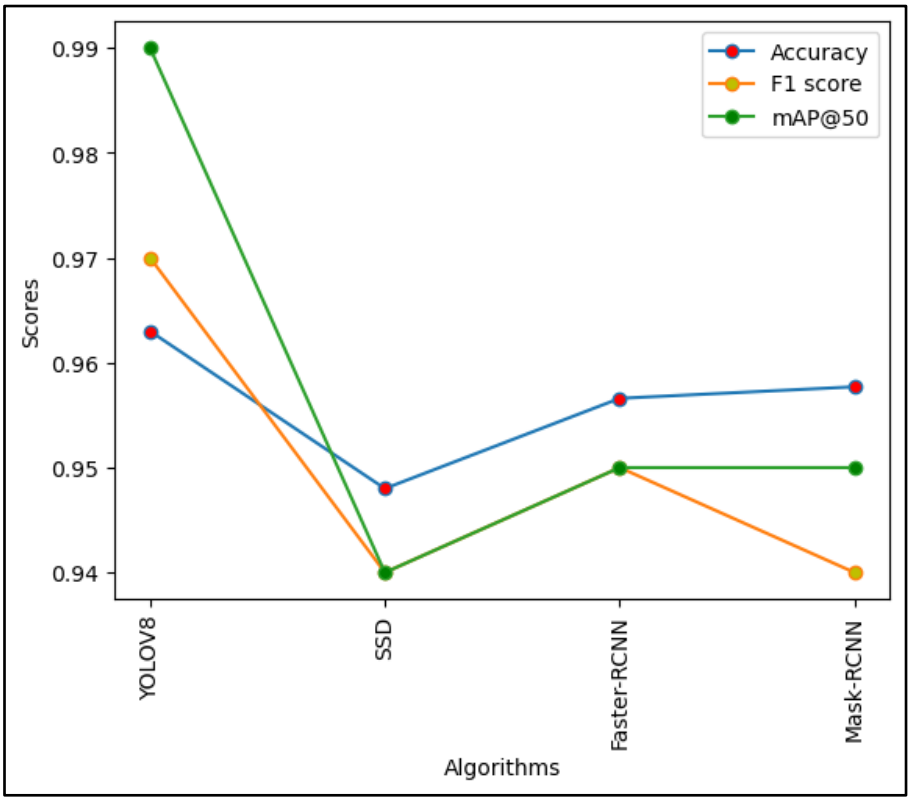


Figure 49: Graphical Comparison of all the Models

CONCLUSION

This research showcases the application of machine/deep learning in agriculture and presents a deep-learning-based strategy for effectively detecting the health status of potato crop plants. By employing multiple layers, the deep learning model automates the image-processing and feature extraction tasks. The study compares four deep learning models, namely SSD, YOLOv8, Faster-RCNN, and Mask-RCNN, alongside existing solutions in the literature. It is evident that all models achieve impressive accuracy levels ranging from 94% to 96%, with YOLOv8 demonstrating superior performance. The CNN-based technique proves reliable in correctly identifying plants, while the YOLOv5 model stands out for its rapid results, making it suitable for real-time plant identification, detection, and classification. Additionally, its lower computational complexity and compact size contribute to faster detection times, rendering it highly suitable for mobile applications. Moreover, the proposed approach can be adapted for large-scale crops such as rice, wheat, and cotton, which are susceptible to various diseases that cannot be individually inspected. Therefore, the proposed technique holds promise for wider application and can be further developed to cater to other crop species.

Future Scope:

The research opens up several promising avenues for future development. Firstly, the deep learning models can be extended to various crops beyond potatoes, such as rice, wheat, and cotton, enabling widespread disease detection. Secondly, integrating the models with IoT and sensor technologies would provide real-time environmental data, enhancing the accuracy of disease detection. Thirdly, data augmentation and transfer learning techniques can be explored to improve model performance. Additionally, collaborative platforms for sharing datasets and knowledge would foster innovation in the field. Lastly, integrating deep learning models into smart farming systems can optimize crop management and resource allocation. These future directions hold great potential for advancing agricultural practices and increasing yields.

REFERENCE

[1] Motlagh, Naser Hossein, Tarik Taleb, and Osama Arouk. "Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives." IEEE Internet of Things Journal 3.6 (2016): 899-922.

[2] Food and Agriculture Organization of the United Nations, “IPPC Annual Report,” 2017.

[3] Ahmad, Aanis, Dharmendra Saraswat, and Aly El Gamal. "A survey on using deep learning techniques for plant disease diagnosis and recommendations for development of appropriate tools." Smart Agricultural Technology 3 (2023): 100083.

[4] Al-Adhaileh, Mosleh Hmoud, et al. "Potato Blight Detection Using Fine-Tuned CNN Architecture." Mathematics 11.6 (2023): 1516.

[5] Liao, Kuo, et al. "Detection of Eucalyptus leaf disease with UAV multispectral imagery." Forests 13.8 (2022): 1322.

[6] Bouguettaya, Abdelmalek, et al. "A survey on deep learning-based identification of plant and crop diseases from UAV-based aerial images." Cluster Computing 26.2 (2023): 1297-1317.

[7] Vasavi, Pallepati, Arumugam Punitha, and T. Venkat Narayana Rao. "Crop leaf disease detection and classification using machine learning and deep learning algorithms by visual symptoms: A review." International Journal of Electrical and Computer Engineering 12.2 (2022): 2079.

[8] Yadhav, S. Yegneshwar, et al. "Plant disease detection and classification using cnn model with optimized activation function." 2020 international conference on electronics and sustainable communication systems (ICESC). IEEE, 2020.

[9] Rehana, Hasin, Muhammad Ibrahim, and Md Haider Ali. "Plant Disease Detection using Region-Based Convolutional Neural Network." arXiv preprint arXiv:2303.09063 (2023).

[10] Liang, Dashuang, et al. "An improved convolutional neural network for plant disease detection using unmanned aerial vehicle images." Nature Environment and Pollution Technology 21.2 (2022): 899-908.

[11] Vasavi, Pallepati, Arumugam Punitha, and T. Venkat Narayana Rao. "Crop leaf disease detection and classification using machine learning and deep learning algorithms by visual symptoms: A review." International Journal of Electrical and Computer Engineering 12.2 (2022): 2079.

[12] Anbumozhi, Anna, and A. Shanthini. "Leaf Diseases Identification and Classification of Self-Collected Dataset on Groundnut Crop using Progressive Convolutional Neural Network (PGCNN)." International Journal of Advanced Computer Science and Applications 14.2 (2023).

[13] Al-Adhaileh, Mosleh Hmoud, et al. "Potato Blight Detection Using Fine-Tuned CNN

Architecture." Mathematics 11.6 (2023): 1516.

[14] Kathiresan, Gugan, et al. "Disease detection in rice leaves using transfer learning techniques." Journal of Physics: Conference Series. Vol. 1911. No. 1. IOP Publishing, 2021.

[15] Atila, Ümit, et al. "Plant leaf disease classification using EfficientNet deep learning model." Ecological Informatics 61 (2021): 101182.

[16] Sachdeva, G., Singh, P., & Kaur, P. (2021). Plant leaf disease classification using deep Convolutional neural network with Bayesian learning. Materials Today: Proceedings, 45, 5584-5590.

[17] Tetila, E. C., Machado, B. B., Menezes, G. K., Oliveira, A. D. S., Alvarez, M., Amorim, W. P., ... & Pistori, H. (2019). Automatic recognition of soybean leaf diseases using UAV images and deep convolutional neural networks. IEEE Geoscience and Remote Sensing Letters, 17(5), 903-907.

[18] Kumar, S. (2021). Plant disease detection using CNN. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(12), 2106-2112.

[19] Yadhav, S. Y., Senthilkumar, T., Jayanthy, S., & Kovilpillai, J. J. A. (2020, July). Plant disease detection and classification using cnn model with optimized activation function. In 2020 international conference on electronics and sustainable communication systems (ICESC) (pp. 564-569). IEEE.

[20] A. Vakanski, "https://www.webpages.uidaho.edu/vakanski/Multispectral_Images_Dataset.html," [Online]. Available: https://www.webpages.uidaho.edu/vakanski/Multispectral_Images_Dataset.html.Accessed on 30th June 2023.

[21] Tzutalin, "GitHub," Label Studio community, [Online]. Available: https://github.com/heartexlabs/labelImg.Accessed on 30th June 2023.

[22] H. Sajid, "What Is Image Data Augmentation?," [Online]. Available: https://www.picsellia.com/post/image-data-augmentation#:~:text=Image%20data%20augmentation%20is%20the,to%20generate%20new%2C%20augmented%20images.Accessed on 30th June 2023.

[23] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[24] A. Mehra, "Labelerr," 2023. [Online]. Available: https://www.labellerr.com/blog/understanding-yolov8-architecture-applications-features/.Accessed on 30th June 2023.

[25] O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

[26] Kumari, K., & Yadav, S. (2018). Linear regression analysis study. Journal of the practice of Cardiovascular Sciences, 4(1), 33.

[27] Wikipedia, “Softmax function,” [Online]. Available: https://en.wikipedia.org/wiki/Softmax_function.Accessed on 30th June 2023.

[28] J. Prakash, “LearnOpenCV,” [Online]. Available: https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/#:~:text=Non%20Maximum%20Suppression%20(NMS)%20is,arrive%20at%20the%20desired%20results.Accessed on 30th June 2023.

[29] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2014). Going deeper with convolutions (2014). arXiv preprint arXiv:1409.4842, 10.

[30] J. Solawetz, “What is YOLOv5? A Guide for Beginners.,” 2020. [Online]. Available: https://blog.roboflow.com/yolov5-improvements-and-evaluation/.Accessed on 30th June 2023.

[31] C. Hughes (2022), “YOLOv7: A Deep Dive into the Current State-of-the-Art for Object Detection,”.

[32] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

[33] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.

[34] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

[35] Ultralytics, “Ultralytics Yolov8 Docs,” [Online]. Available: https://docs.ultralytics.com/.Accessed on 30th June 2023.

[36] L. Vladimirov, “TensorFlow 2 Object Detection API tutorial,” [Online]. Available: https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/.Accessed on 30th June 2023.

[37] Simplilearn, “What is Epoch in Machine Learning?,” [Online]. Available: https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-epoch-in-machine-learning.Accessed on 30th June 2023.

[38] NV5 Geospatial, “Vegetative Indices,” [Online]. Available: https://www.l3harrisgeospatial.com/docs/vegetationindices.html.Accessed on 30th June 2023.

[39] Office of Research and Economic Development (ORED) at the University of Idaho, “Aleksandar (Alex) Vakanski,” [Online]. Available: https://www.webpages.uidaho.edu/vakanski/Multispectral_Images_Dataset.html.Accessed on 30th June 2023.

[40] Liao, K., Yang, F., Dang, H., Wu, Y., Luo, K., & Li, G. (2022). Detection of Eucalyptus leaf disease with UAV multispectral imagery. Forests, 13(8), 1322.

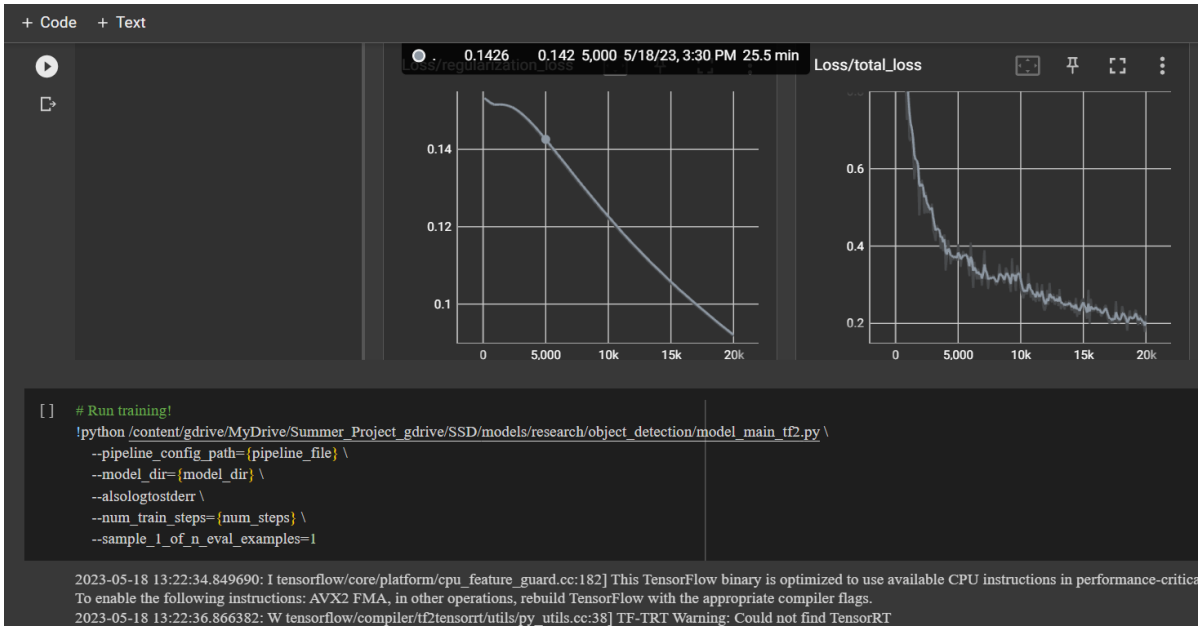
APPENDIX

YOLOV8:

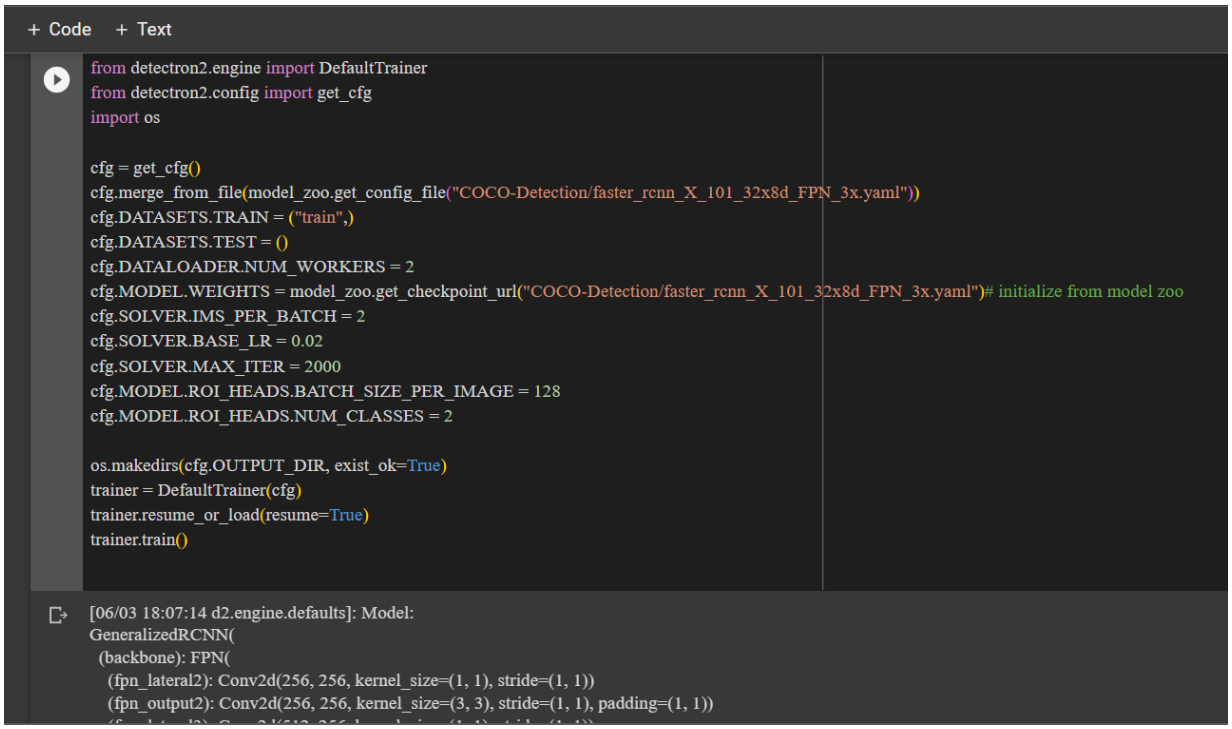
```
%cd /content/drive/MyDrive/Summer_Project
!yolo task=detect mode=train model=yolov8n.pt data= data.yaml epochs=100
```

```
Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
1/100     2.61G      2.19       2.95      2.121        168        640: 100% 88/88 [01:29<00:00,  1.01s/it]
               Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:05<00:00,  1.30s/it]
                 all        100       1104     0.0281      0.749      0.118     0.0462

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
2/100     2.41G      1.919      2.319      1.915        149        640: 100% 88/88 [01:23<00:00,  1.05it/s]
               Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:02<00:00,  1.42it/s]
                 all        100       1104      0.305      0.289      0.209     0.0813

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
3/100      2.4G      1.865      2.189       1.88        163        640: 100% 88/88 [01:25<00:00,  1.03it/s]
               Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:03<00:00,  1.08it/s]
                 all        100       1104      0.308      0.316      0.231      0.089

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
4/100     2.32G       1.85      2.135      1.847        152        640: 100% 88/88 [01:24<00:00,  1.04it/s]
               Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:03<00:00,  1.01it/s]
                 all        100       1104      0.206      0.272      0.125     0.0438

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

SSD:

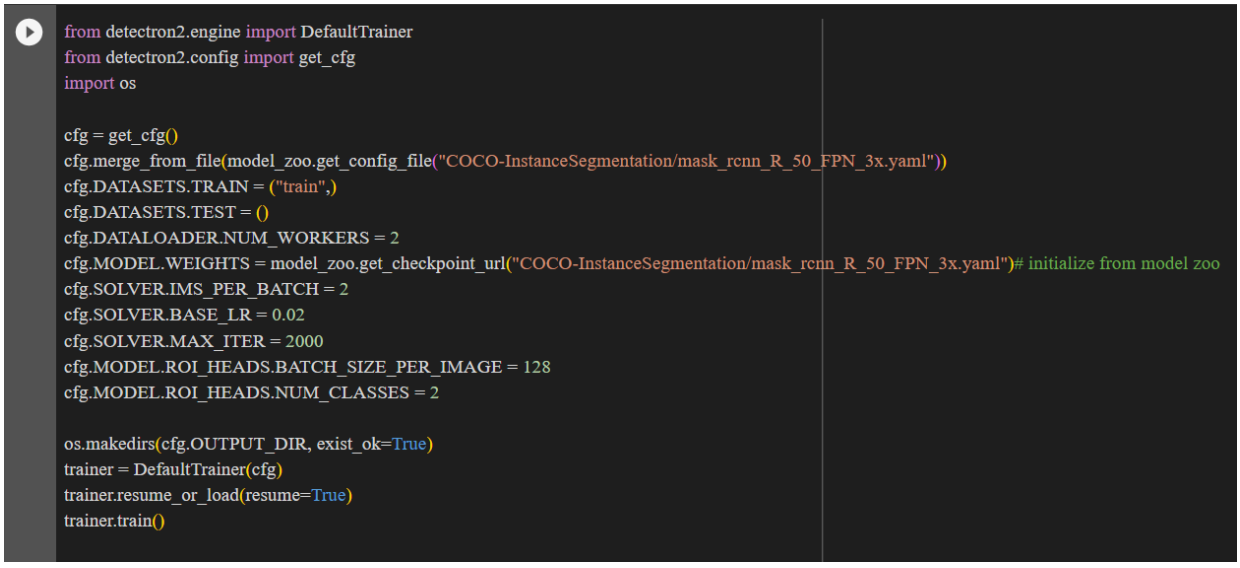


FRCNN:



```
+ Code  + Text

from detectron2.engine import DefaultTrainer
from detectron2.config import get_cfg
import os

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("train",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml")# initialize from model zoo
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.02
cfg.SOLVER.MAX_ITER = 2000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 2

os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = DefaultTrainer(cfg)
trainer.resume_or_load(resume=True)
trainer.train()

[06/03 18:07:14 d2.engine.defaults]: Model:
GeneralizedRCNN(
  (backbone): FPN(
    (fpn_lateral2): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
    (fpn_output2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
```

Mask-RCNN:



```
from detectron2.engine import DefaultTrainer
from detectron2.config import get_cfg
import os

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("train",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")# initialize from model zoo
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.02
cfg.SOLVER.MAX_ITER = 2000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 2

os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = DefaultTrainer(cfg)
trainer.resume_or_load(resume=True)
trainer.train()
```

Proforma 4

Undertaking from the PG student while submitting his/her final dissertation to his respective institute

Ref. No. _____

I , the following student

Sr. No.	Sequence of students names on a dissertation	Students name	Name of the Institute & Place	Email & Mobile
1.	First Author	Ayush Mitra	SIG	Email: 22070243007@sig.ac.in Mobile: 9748062253

Note: Put additional rows in case of more number of students

hereby give an undertaking that the dissertation "Plant Disease Classification Using UAV Image" been checked for its Similarity Index/Plagiarism through Turnitin software tool; and that the document has been prepared by me and it is my original work and free of any plagiarism. It was found that:

1.	The Similarity Index (SI) was: *(Note: SI range: 0 to 10%; if SI is >10%, then authors cannot communicate ms;* ***attachment of SI report is mandatory****)*	9%
2.	The ethical clearance for research work conducted obtained from: *(Note: Name the consent obtaining body; if 'not appliable' then write so)*	NA
3.	The source of funding for research was: *(Note: Name the funding agency; or write 'self' if no funding source is involved)*	Self
4.	Conflict of interest: *(Note: Tick √ whichever is applicable)*	No
5.	The material (adopted text, tables, figures, graphs, etc.) as has been obtained from other sources, has been duly acknowledged in the manuscript: *(Note: Tick √ whichever is applicable)*	Yes

In case if any of the above-furnished information is found false at any point in time, then the University authorities can take action as deemed fit against all of us.

Ayush Mitra
Full Name &
Signature of the student

Sahil Shah
Name &
Signature of SIU Guide/Mentor

Date:

Endorsement by
Academic Integrity Committee (AIC)

Place: Pune

Note: It is mandatory that the Similarity Index report of plagiarism (only first page) should be appended to the UG/PG dissertation

Document Viewer

Turnitin Originality Report

Processed on: 09-Jul-2023 19:07 IST
ID: 2128413295
Word Count: 6448
Submitted: 4

Project_Report By Ayush Mitra

Similarity Index	Similarity by Source	
9%	Internet Sources:	4%
	Publications:	4%
	Student Papers:	5%

exclude quoted | exclude bibliography | exclude small matches | mode: quickview (classic) report | print | download

1% match ("Third International Conference on Image Processing and Capsule Networks", Springer Science and Business Media LLC, 2022)
"Third International Conference on Image Processing and Capsule Networks", Springer Science and Business Media LLC, 2022

1% match (student papers from 14-Nov-2022)
Submitted to University of Carthage on 2022-11-14

<1% match (student papers from 08-Jun-2023)
Submitted to Savitribai Phule Pune University on 2023-06-08

<1% match ("PRICAI 2022: Trends in Artificial Intelligence", Springer Science and Business Media LLC, 2022)
"PRICAI 2022: Trends in Artificial Intelligence", Springer Science and Business Media LLC, 2022

<1% match (student papers from 27-Apr-2023)
Submitted to Queen Mary and Westfield College on 2023-04-27

<1% match (student papers from 13-Dec-2007)
Submitted to Kingston University on 2007-12-13

<1% match (Internet from 30-Mar-2023)
http://eprints.utar.edu.my

<1% match (student papers from 30-Apr-2022)
Submitted to Dhirubhai Ambani Institute of Information and Communication on 2022-04-30

<1% match (W. R. Raun, J. B. Solie, M. L. Stone, K. L. Martin, K. W. Freeman, R. W. Mullen, H. Zhang, J. S. Schepers, G. V. Johnson. "Optical Sensor-Based Algorithm for Crop Nitrogen Fertilization", Communications in Soil Science and Plant Analysis, 2005)
W. R. Raun, J. B. Solie, M. L. Stone, K. L. Martin, K. W. Freeman, R. W. Mullen, H. Zhang, J. S. Schepers, G. V. Johnson. "Optical Sensor-Based Algorithm for Crop Nitrogen