A project report on

WEED PLANT DETECTION USING FASTER RCNN METHOD

Submitted in partial fulfillment for award of
Bachelor of Technology

Degree

In

INFORMATION TECHNOLOGY
By

CH. Lakshmipriya	Y20AIT418
D. Sasikala	Y20AIT425
M. Himavanth	L21AIT411
D. Naga Lakshmi Srujana	Y20AIT428

Under the guidance of

MR. Y. MASTANAIH NAIDU **Asst.Prof.,**

Bapatla Engineering College::Bapatla
(Autonomous)

(Affiliated to Acharya Nagarjuna University)

BAPATLA-522101, Andhra Pradesh, INDIA

2023-2024

Bapatla Engineering College

(Autonomous)

Department of Information Technology

Mahatmaji Puram, Bapatla – 522102

CERTIFICATE

This is to certify that the Project entitled

"**Weed Plant Detection Using Faster RCNN Method**"

CH. Lakshmipriya	Y20AIT418
D. Sasikala	Y20AIT425
M. Himavanth	L21AIT411
D. Naga Lakshmi Srujana	Y20AIT428

is a record of bona fide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Technology (Information Technology) at BAPATLA ENGINEERING COLLEGE, Bapatla under the Acharya NagarjunaUniversity. This work is done during the year 2023-2024, under our guidence and supervision.

Date:

(Mr.Y. Mastanaih Naidu)
Project Guide

(Dr. B. Krishnaiah)
Project Coordinator

(Dr. N. Sivaram Prasad)
H.O.D, I.T Department

Sig. of External Examiner

DECLARATION

We declare that this project report titled **"Weed Plant Detection Using Faster RCNN Method"** submitted in partial fulfillment of the degree of **B. Tech in** (**Information Technology**) is a record of original work carried out by us under the supervision of **Mr.Y. Mastanaih Naidu**, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

CH. Lakshmipriya

D. Sasikala

M. Himavanth

D. Srujana

ACKNOWLEDGMENT

We are profoundly grateful to **Mr. Y. Mastanaih Naidu** for his expert guidance and continuous encouragement throughout to see that this Project from commencement to its completion.

We would like to express deepest appreciation towards **Dr. Nazeer Shaik**, Principal, Bapatla Engineering College, **Dr. N. Sivaram Prasad**, Head of Department of Information Technology and **Dr. B. Krishnaiah**, Project Coordinator who's in valuableguidance supported us in completingthis Project.

Finally, we must express our sincere heartfelt gratitude to all the staff members of the Information Technology Department who helped us directly or indirectly during this course of work.

CH. Lakshmipriya
D. Sasikala
M. Himavanth
D. Srujana

ABSTRACT

In agricultural regions, the procedure of weed removal is crucial. Weed removal in the classic way, takes longer and requires greater physical effort. The idea is to eliminate weeds from agricultural fields automatically. Our project uses a deep learning algorithm to detect weeds growing between crops. Deep learning method also known as deep learning is usedto analyze the main properties of agricultural photographs. Weeds and crops have been identified using the dataset. Region Convolution neural network (RCNN) uses a completely attached surface with rectified linear units (RELU) to differentiate weed and crop. It extracts features of crops using deep learning. RCNN uses features of the preceding imageto extract region of interest (ROI). Deep learning network features are used to identify crop. Crop images were frequently captured using robots, drones, and cell phones. It also discusses algorithm accuracy, such as how outperformed all machine learning algorithms in many cases, with the highest accuracy of 99 percent, and how RCNN with its variants also performed well with the highest accuracy of 99 percent, with only VGGNet providing the lowest accuracy of 84 percent. Finally, the project will serve as a starting point for researchers who wish to undertake further research in this area.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

RCNN : Region-based Convolutional Neural Network

CNN : Convolutional Neural Network

VGG16 : Visual Geometry Group

DL : Deep Learning

ANN : Artificial Neural Network

CHAPTER 1
INTRODUCTION

1.1. Weed Detection

In olden days weed detection was done by employing some men, especially for that purpose. They will detect the weed by checking each and every place of the field. Then they will pluck them out manually using their hands. Later with the advancement in technology they started using the herbicides to remove the weeds. But to detect the weeds they are still using manual power in many parts of the world. Later there came a few methods to detect the weeds automatically but due to lack of accuracy, they are unable to reach the people. Then they started using image processing for this purpose. In this proposed project our main aim is to detect the weed in the crop by using image processing. Then we will give the inputs of the weed areas to an automatic spray pesticide only in those areas. For this, we need to take a photograph of the field with good clarity to detect the weeds with more accuracy. Taking a photograph can be done by attaching a camera to a tractor or taking them manually. Then we will apply image processing to that image using Faster RCNN to detect the weed.

1.2. Methodology

1.2.1. Image acquisition

The proposed methodology is developed using a dataset of weed images. The methods extract weed using the tasks followed by images capture, edge identification, and image type identification. Weed images are captured using high quality cameras. Theimages are compared with images stored in the dataset. New images are contentiously added in the dataset. The accuracy of the proposed system increases as the number ofrecords in the dataset is more.

1.2.2. Image pre-processing

Image processing module performs some basic processes to get the required picture for processing. To obtain an accurate and clear image, the algorithm performs various operations like gray scale, conversion, and sharpening, filtering, edging, smoothing and image segmentation. The quality of image is improved in the preprocessing phase by improving image features and reducing noise. The black and white photos are of different shades of gray. The value of each pixel is measured on the gray scale image. The quality of image in the form of sharpness, smoothness is improved using various tasks in the image pre-processing. The images are made sharper with the help of filters. Noise is minimized with the help of smoothing. Multiple algorithms are used to enhance quality and sharpnessof the image in the task of image pre-processing. The image pre-processing is the very important, required and fundamental step in the image identification.

1.2.3. Image segmentation

There are a set of operations included in the image recognition. Image recognition has a diversified set of applications including number plate recognition, and CCTV surveillance. In the process of image segmentation, the actual image is translated into a binary valued image using the maximum valued method. The digital image is characterized into several parts depending on the values of each image component. The pixels with similar values are clubbed. The values are used to identify different portions of the image.In short, the image characterization process is used to extract key features of the object for future analysis.

1.2.4. Feature extraction

The system does further operations on the separated picture in this module. The module deals with feature extraction to extract overall information of the weed image. Image processing and machine learning help to classify weed images. The important features of the weed images are extracted and used for classification based on the range of values. The main point in image classification is to identify the most dynamic features for classification of images. Feature extraction is a key in the image processing to handle.
The image for various operations like dimension reduction.

1.2.5. CNN algorithm

The weed detection is done using CNN algorithm. The layers of CNN like input, processing and output do the work of weed image identification. The image denitrification further leads to image classification. The massive developments in ultimating the distance among human and machine capabilities is achieved using the techniques of machine learning and artificial intelligence. The CNN results are amazing in the area of image processing and classification. The area of machine imaginative and prescient is certainly considered one among numerous such disciplines. The aim of this area is to allow machines to look and understand the sector within the identical manner that people do, and to apply that understanding for a number of responsibilities inclusive of image recognition. The image feature extraction, creation of value vector, clarification training testing etc are

Major components come with CNN algorithm. The image captured using a high quality camera is an input for CNN, the weights are assigned to the various factors of the imagewith the help of extracted features and images are distinguished depending on the feature vector values.

1.2.6. Classification

The classification leads to the identification of images. The training of the image is again performed using three phases. Input image, image pre-processing, feature extraction are three stages used in the training phase. The pre-processing phase makes the image clearer. All the borders are pixel values extracted in this phase. The values identified in the pre-processing phase are used for training of the module.

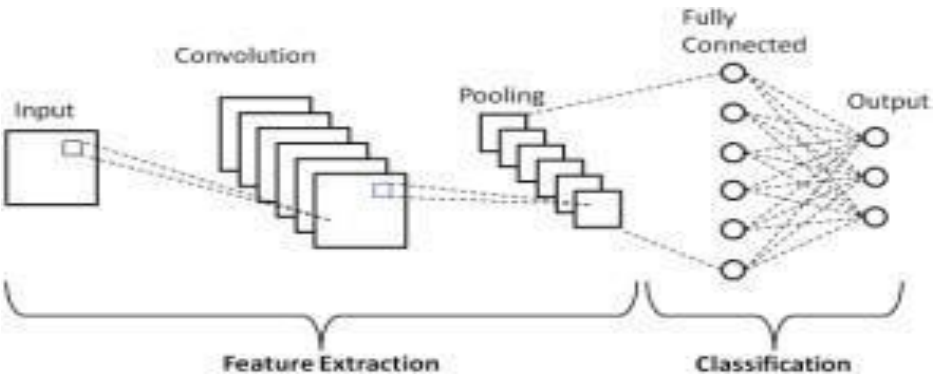

Figure 1.1: Convolutional Neural Network

Different Layers of CNN are:

1) Convolutional Layer

This is the primary layer used to extract diverse capabilities from the enter images. This layer plays the mathematical operation of convolution among the enter picture and a clear out out of length MxM. By swiping the clear out out over the enter picture, the dot productis calculated among the clear out out and the elements of the enter picture which can be proportional to the clear out out's length (MxM).

2) Pooling Layer

This layer's number one aim is to lessen the dimensions of the convolved function map as a way to lessen computational costs. This is finished via way of means of decreasing the connections among layers and running independently on every function map.

3) Fully Connected Layer

The Fully Connected layer, which incorporates weights and biases in addition to neurons, is used to attach neurons from special layers. These layers are commonly located before the output layer and represent the very last few layers of CNN Architecture.

4) Activation Functions

It determines which version of statistics must be performed inside the ahead path and which must now no longer be on the community's end. It consists of variability in the community.

CHAPTER 2
LITERATURE SURVEY

Several authors have contributed to the emergence of weed detection systems. The most current innovations proposed through papers are outlined below:

In paper [1] the author describes “A Survey on weed Detection using Image Processing”. Author discussed various methods like pre-processing, feature extraction, classification on the weed images. The disadvantage is that the classification accuracy depends on the proper images and image resolution.

In paper [2] the author describes the algorithm of weed detection in crops by computational vision. The paper talks about developing their own algorithm for the detection and processing of images in MATLAB. The advantage is that the use of low-level characteristics like plant color and the area make the algorithm useful for a vast range of vegetable crops. The major limitation of the system is it will lose its significance if the weedand good crop are of the same size.

In paper [3] the author puts forth a review for selective spraying on weed detection. It reviews the YOLOV2 and CNN techniques. As feature extractors, deep CNN architectures provide higher performance and allow for faster application development. If weed is not managed properly in the early stage this will make it hard to discriminate.

In the paper [4] the author describes the results of the ANN and AlexNet algorithms for weed detection utilizing UAV- based photos. The downside is that the complexity of these algorithms may lead to slower image processing.

In paper [5] the author describes Weed Identification Using Deep Learning and Image Processing in Vegetable Plantation. A CenterNet model was trained to detect vegetables. The disadvantage is that due to similarities between weed and background, some weeds willbe falsely identified as background noise.

2.1 EXISTING SYSTEM

2.1.1. Introduction

YOLOv3 is a real-time object detection algorithm that identifies specific objects in videos, live feeds or images. The YOLO machine learning algorithm uses features learned by a deep convolutional neural network to detect objects located in an image.

2.1.2. Working

YOLO is a Convolutional Neural Network (CNN), a type of deep neural network, for performing object detection in real time. CNNs are classifier-based systems that process input images as structured arrays of data and recognize patterns between them. It allows the object detection model to look at the whole image at test time. This means that the global context in the image informs the predictions. YOLO and other Convolutional neural network algorithms "score" regions based on their similarities to
predefined classes.

2.1.3. YOLOV3 Algorithm

Firstly, we randomly pick out multiple width and peak values (wi , hi) as one preliminary cluster middle c1 from the set φ. To achieve the opposite k − 1 preliminary cluster facilities, we repeat the subsequent manner for k − 1 instances with a view to construct k − 1 Markov chains with duration m. The manner starts through computing all inspiration distributions, or q (φj). It then paperwork the clusters primarily based totally on the randomly decided factors from the cluster. Then we compute the brand-new cluster facilities and the usage of those cluster facilities we decide the bounding boxes' new width and peak.

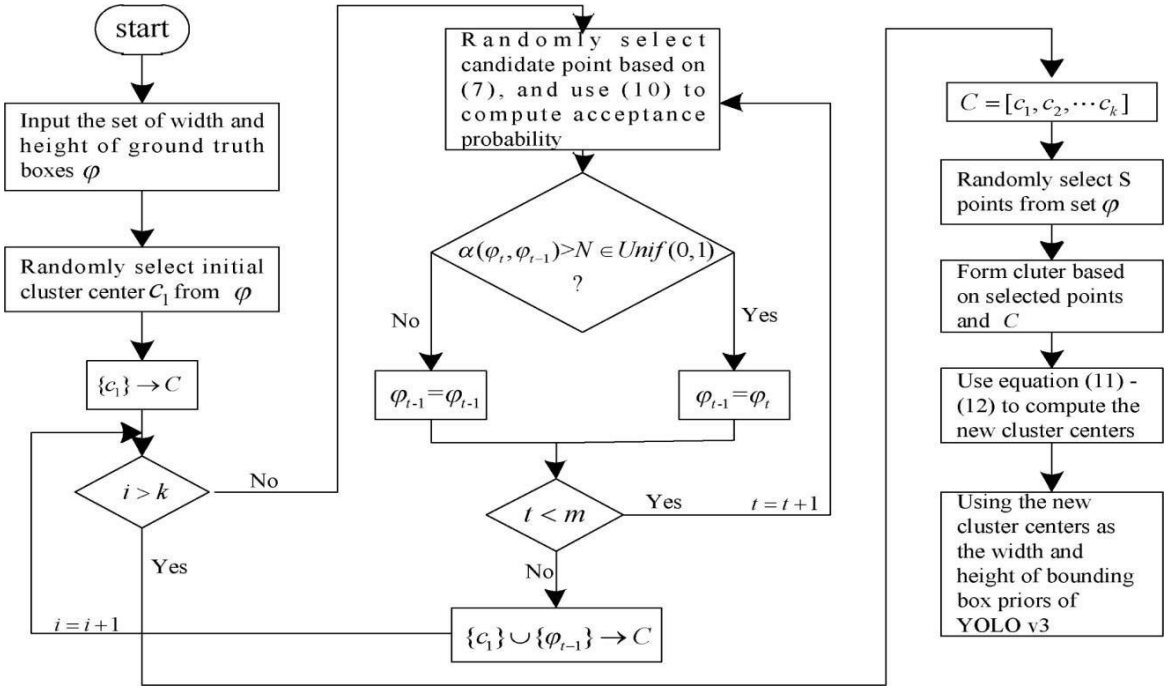

Figure 2.1: Yolov3 Algorithm

2.1.4. Drawbacks

- YOLOv3 may struggle with accurately detecting small objects in images. Since it divides the image into a grid and predicts bounding boxes within each grid cell.
- Its single-pass approach can face challenges when multiple objects overlap in the image.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1. Introduction

Requirements define the expected services of the system (service statements) that the system must obey (constraint statements). The service statements continue the system's functional requirements. Functional requirements can be grouped into those that describe the scope of the system, the necessary business functions, and the required data structures. The constraint statement can be classified according to different categories of restriction imposed on the system, such as the required correctness, performance, availability, orsecurity. The constraint statement that constitutes the system's non-functional requirements. Non-functional requirements are also known as software qualities.

3.2. Requirements Overview

3.2.1. Data Requirements

Data Collection: A large dataset of images containing various weeds and crops in different environmental conditions (lighting, weather, background, etc.). Data should be annotated with bounding boxes around each weed and crop instance, ideally in a format compatible with the chosen deep learning framework (e.g., JSON, XML). Data Preprocessing: Image resizing and normalization to match the input dimensions expected by the Faster R-CNN model. Data augmentation techniques (e.g., rotation, flipping, scaling) to enhance model robustness.

3.2.2. Performance Metrics

- Accuracy: Overall accuracy in correctly detecting and classifying weeds among crops.
- Precision and Recall: Especially in scenarios where missing a weed ormisclassifying crops as weeds can have significant implications.

- Inference Time: Speed of the model in processing new images, important for real-time or near-real-time applications.
- Model Size and Efficiency: Optimized for deployment in constrained environments,if necessary.

3.2.3. Technical Requirements

Hardware:

- GPU: High-performance GPU(s) for training and inference (e.g., NVIDIA series), considering Faster R-CNN's computational intensity.
- CPU: Multi-core processor for efficient data preprocessing and parallel tasks.
- RAM: Sufficient memory (e.g., 16GB or more) to handle the data and model training operations.
- Storage: High-capacity storage (both SSD and HDD) to store large datasets, model checkpoints, and logs.

Software:

- Operating System: Windows with support for all required libraries.
- Programming Language: Python (version 3.6 or newer).
- Deep Learning Framework: PyTorch or Tensor Flow for implementing the Faster R-CNN model.
- Libraries and Tools: CUDA and cuDNN for GPU acceleration, OpenCV for image processing, NumPy for numerical operations, Matplotlib for visualization, and Google collab or similar for prototyping.

3.2.4 Libraries Requirements:

OpenCV

Open-source Computer Vision is a popular computer vision library started by Intel in 1999. OpenCV is a cross-platform library using which we can develop real-time computer vision applications. OpenCV mainly focuses on image processing, video capture and analysis including features like face

detection and object detection. OpenCV uses machine learning algorithms to search for faces within a picture.

NumPy

NumPy (Numerical Python) is an open-source Python library that's used in almost every field of science and engineering. It's the universal standard for working with numerical data in Python, and it's at the core of the scientific Python and PyData ecosystems. The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages.

MATPLOTLIB

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

OS

The OS module in python provides functions for interacting with the operating system. OS, comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The os and os.path modules include many functions to interact with the filesystem.

TENSORFLOW

TensorFlow is a Python-friendly open-source library for numerical computation that makes machine learning faster and easier. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition. word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE .

CHAPTER 4
SYSTEM ANALYSIS

4.1 Introduction

System analysis is a general term that refers to an orderly structured process for identifying and solving problems. It is carried out to make the system more effective. Analysis is a detailed study of the various operation performed by a system and their relationships within and outside the system. A key question is what must be done to solve the problem.

4.2 Proposed System

Faster Region-based Convolutional Neural Network is a type of deep learning architecture used for object detection in computer vision tasks. RCNN was one of the pioneering models that helped advance the object detection field by combining the power of convolutional neural networks and region-based approaches. R-CNN is a two-stage detection algorithm. The first stage identifies a subset of regions in an image that might contain an object. The second stage classifies the object in each region.

4.2.1. Working

1. Input Image: The process begins with an input image containing objects that need to be detected and classified.
2. Feature Extraction: The input image is passed through a convolutional neural network (CNN), such as ResNet or VGG, to extract features. This process generates a feature map that represents the spatial distribution of features in the image.
3. Region Proposal Network (RPN): The feature map generated by the CNN is fed into a Region Proposal Network (RPN), which efficiently generates region proposals (bounding boxes) that are likely to contain objects. The

4. The RPN slides a small window (called an anchor) over the feature map and predicts whether each anchor contains an object and the offsets needed to adjust the anchor to better fit the object. These predictions are used to generate a set of candidate region proposals.

5. Region of Interest (RoI) Pooling: Once the region proposals are generated, RoI pooling is applied to extract fixed-size features from each proposal. RoI pooling ensures that features from differently sized regions are mapped to the same size, allowing for consistent processing in subsequent layers.
6. Object Classification and Bounding Box Regression: The extracted features from the RoI pooling layer are passed through a detection network, which consists of fully connected layers. This network performs two tasks: Object Classification: It classifies each proposal into object categories (e.g., person, car, dog) Bounding Box Regression: It refines the coordinates of the bounding boxes to better localize the objects within the proposals.

7. Output: The final output consists of the detected objects along with their corresponding class labels and bounding box coordinates. Objects that meet a certain confidence threshold are considered detected, while others are discarded as background or false positives.

8. Training: During training, Faster R-CNN is trained end-to-end using back propagation. The model is optimized using a multi-task loss function, which includes classification loss (e.g., cross-entropy) for object classification and regression loss (e.g., smooth L1 loss) for bounding box regression. The parameters of both the RPN and the detection network are updated jointly to minimize the total loss.

9. Evaluation: After training, the performance of the Faster R-CNN model is evaluated on a separate validation set to assess its accuracy and generalization capability. The model's performance is typically measured in terms of metrics such as precision, recall, and mean average precision (mAP) across different object categories.

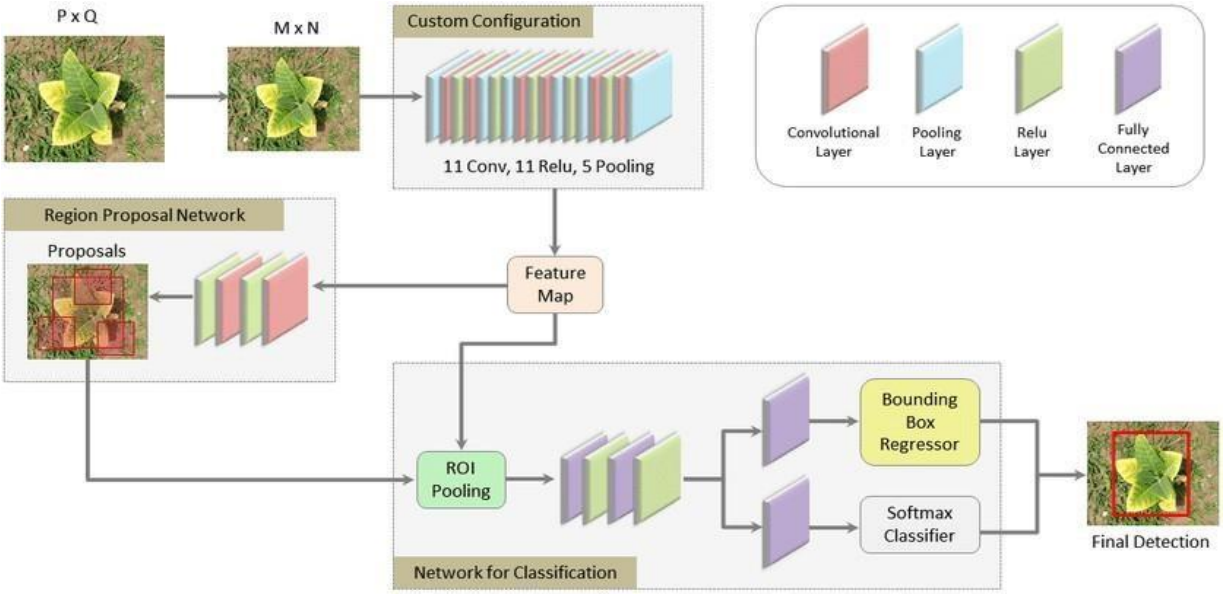

Figure 4.1: Proposed System Architecture

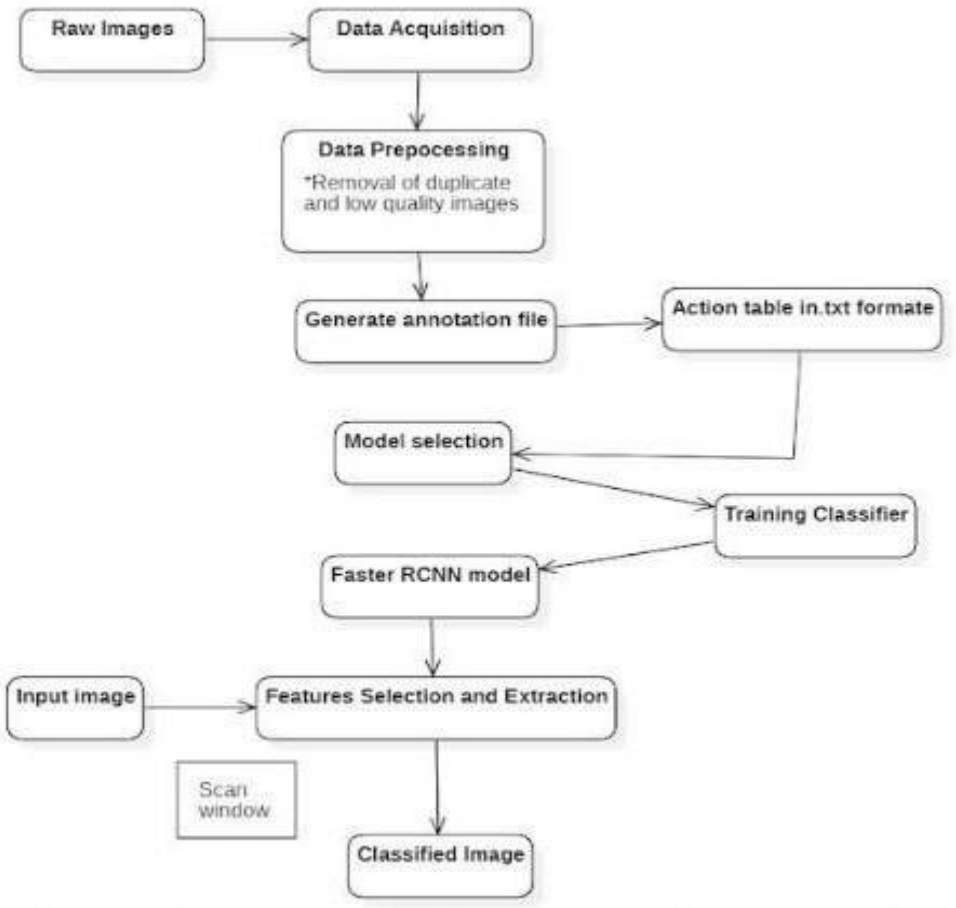

Figure 4.2: Weed Detection Design

4.2.2 Advantages of Proposed System

- It provides accurate object detection by leveraging region-based convolutional features.
- It can efficiently scale to handle large datasets and complex scenes.
- Faster R-CNN achieves state-of-the-art accuracy in object detection tasks.
- It can handle objects with different sizes, orientations, and scales, making them suitable for real-world scenarios with diverse objects and complex backgrounds.

CHAPTER 5
SYSTEM DESIGN

5.1 Introduction

Designing a system for weed detection using Faster R-CNN involves several key components and steps that work together to detect and classify weeds from crops in images. Faster R-CNN is a popular deep learning model for object detection, which can be fine- tuned for the specific task of distinguishing weeds from crops.

5.2 Levels of system design

5.2.1 Data Collection and Preprocessing

- Data Collection: Gather a diverse dataset of images from agricultural fields that includes various types of crops and weeds under different conditions (lighting, growth stages, background, etc.).
- Annotation: Annotate the images by marking bounding boxes around weeds and labeling them. Tools like LabelImg or CVAT can be used for annotation.
- Preprocessing: Preprocess the images to a uniform size as required by the Faster R-CNN model. Apply data augmentation techniques (e.g., rotation, flipping, scaling) to increase the diversity of your training data.

Figure 5.2 Dataset Images

Label	Width	Height	xmin	ymin
crop	512	512	20	38
weed	512	512	80	97
weed	512	512	25	14
crop	512	512	14	56

Table 5.2 Presents an example of the annotations.

5.3 UML Diagrams

UML (Unified Modeling Language) diagrams are a standardized notation used to visually represent the structure, behavior, and interactions of systems. In the context of weed detection, UML diagrams can help to conceptualize and communicate the various components, processes, and relationships involved in the system.

5.3.1 Use case Diagram

Use case diagrams depict the interactions between users (actors) and the system, illustrating the functionalities or services provided by the system. In Weed detection, use case diagrams could outline the different user roles and their interactions with the system, such as uploading imaging data, initiating weed detection, reviewing results, and generating reports.

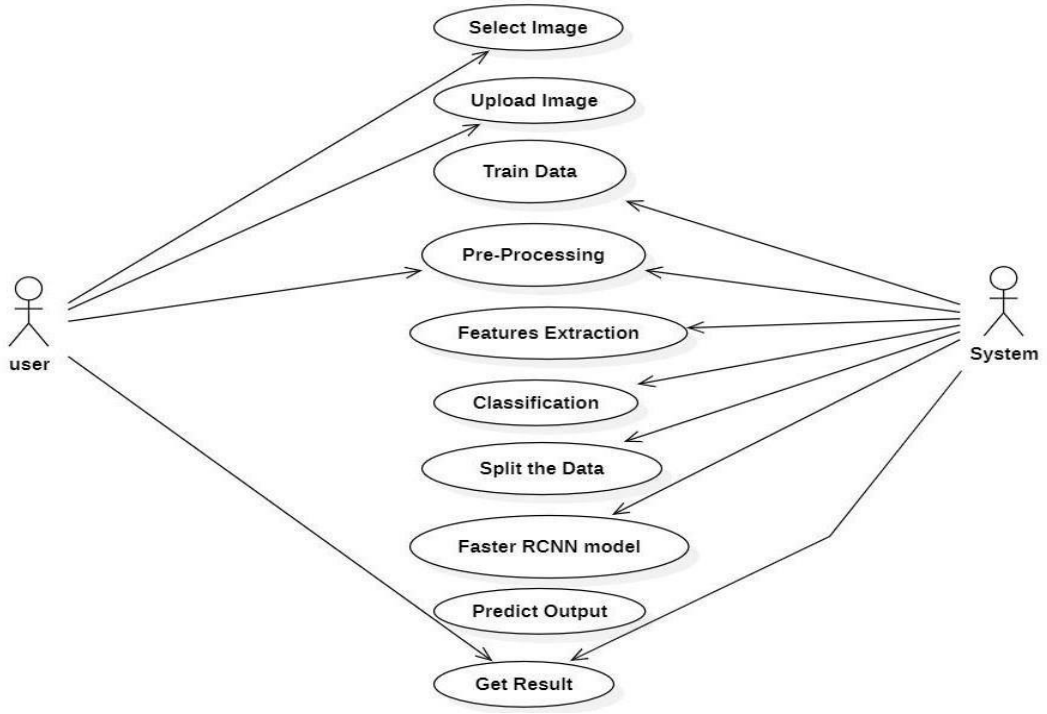

Figure 5.3: Use case Diagram

5.3.2 Activity Diagram

Activity diagrams model the flow of activities or processes within the system, depicting the sequence of actions performed to achieve a specific goal. In Weeddetection, activity diagrams could illustrate the workflow of Weed plants detection, including steps such as image preprocessing, feature extraction, classification, and result visualization. These diagrams help to visualize the sequence of tasks and decision points in the detection process.

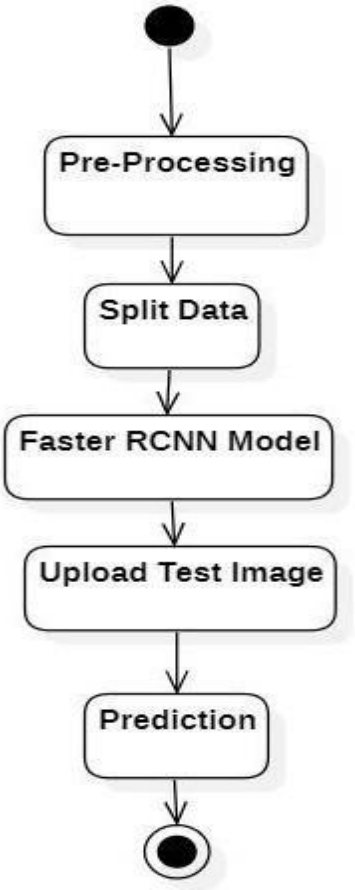

Figure 5.4: Activity Diagram

CHAPTER 6

CODING

6.1 Data Loading into Model

Given path to a CSV file containing the annotations for the images. This fileincludes information such as image filenames, label names, and bounding box coordinates for objects within each image.Image Loading, For a given index the method locates the image file path from the root directory and the filename specified in the CSV. It then loads the image, ensuring it is in RGB format, which is a common requirement for many processing techniques in image analysis. Retrieves the label name for the object in the image and its bounding box coordinates. Converts the bounding box data into a PyTorch tensor, which is required for most machine learning frameworks that handle operations on GPUs.

```
class CustomDataset(Dataset):
    def __init__(self, csv_file, root_dir, transform=None):
        self.annotations = pd.read_csv(csv_file)
        self.root_dir = root_dir
        self.transform = transform or transforms.ToTensor()

    def __len__(self):
        return len(self.annotations)

    def __getitem__(self, idx):
        img_path = os.path.join(self.root_dir, self.annotations.iloc[idx, 0])
        image = Image.open(img_path).convert("RGB")
        label_name = self.annotations.iloc[idx, 3]

        # Ensure the bbox is a float array to prevent TypeError
        bbox = np.array(self.annotations.iloc[idx, 4:8], dtype=np.float32)

        # Convert the bbox to a PyTorch tensor
        bbox = torch.from_numpy(bbox).unsqueeze(0)  # Add an extra dimension for batch

        # Convert labels to numeric values and handle the 'none' class
        label_dict = {'weed': 1, 'crop': 2, 'none': 3}
        label = label_dict.get(label_name, 0)  # Using 0 as a default for unrecognizable classes

        labels = torch.tensor([label], dtype=torch.int64)

        # If label is 'none', use a placeholder box (e.g., [0, 0, 1, 1])
        if label_name == 'none':
            bbox = torch.tensor([[0, 0, 1, 1]], dtype=torch.float32)

        target = {}
        target['boxes'] = bbox
        target['labels'] = labels

        if self.transform:
            image = self.transform(image)

        return image, target
```

Figure 6.1: Data Loading

6.2 Setting up the Model

Faster R-CNN model with a ResNet-50 Feature Pyramid Network (FPN) as the backbone. This is a powerful object detection model that is efficient in detecting objects at various scales.The number of classes the model should recognize is set to 4 (three specific classes plus one class for the background). The code customizes the model by adjusting the final layer of the region of interest (ROI) heads to predict these four classes, changing the number of output features to match the number of classes.The model is moved to a computing device, which is typically a GPU for faster computation.

The optimizer is set up to update only the parameters of the model that require gradients. This is a common practice to ensure only the trainable parameters are updated during backpropagation.The Stochastic Gradient Descent (SGD) optimizer is chosen for training, which is a common choice for training deep neural networks. It includes momentum for smoothing out updates during training and a weight decay for regularization to prevent overfitting. The learning rate, momentum, and weight decay are specifically set based on either prior experimentation or literature recommendations.

```
from torch.optim.lr_scheduler import StepLR

# Create the model
model = fasterrcnn_resnet50_fpn(pretrained=True)
num_classes = 4  # 3 classes (your classes) + background
in_features = model.roi_heads.box_predictor.cls_score.in_features
model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)
model.to(device)

# Define the optimizer
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.0025, momentum=0.9, weight_decay=0.0005)
# Define the learning rate scheduler
lr_scheduler = StepLR(optimizer, step_size=3, gamma=0.1)
```

Figure 6.2: Model set-up

6.3 Training the Model

Inside each epoch, the data is processed in batches. This is done by iterating over train_data_loader, which provides batches of images and their corresponding targets.

The model computes the loss, which is a measure of how well the model's predictions match the actual targets. The loss is calculated for each batch. The gradients of the loss are computed with respect to the model parameters using losses.backward(). This is part of the backpropagation algorithm. The optimizer updates the model parameters to minimize the loss using optimizer.step().The gradients are reset to zero using optimizer.zero_grad() to prevent accumulation from previous iterations.

```python
num_epochs = 10  # Adjust as per your requirement

for epoch in range(num_epochs):
    model.train()
    total_loss = 0
    num_batches = 0

    for images, targets in train_data_loader:
        images = list(img.to(device) for img in images)
        targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

        loss_dict = model(images, targets)
        losses = sum(loss for loss in loss_dict.values())

        optimizer.zero_grad()
        losses.backward()
        optimizer.step()

        total_loss += losses.item()
        num_batches += 1

    # Update the learning rate
    lr_scheduler.step()

    # Log the average loss for this epoch
    if num_batches > 0:
        epoch_loss = total_loss / num_batches
        print(f'Epoch {epoch} finished, Loss: {epoch_loss}')
    else:
        print(f'Epoch {epoch} finished, no valid batches.')

    # Save the model after each epoch
    model_save_filename = os.path.join(model_save_path, f'faster_rcnn_epoch_{epoch}.pth')
    torch.save(model.state_dict(), model_save_filename)
```

Figure 6.3: Model Training

CHAPTER 7
TESTING

7.1 Introduction

Software testing is a critical element of software quality assurance and represents theultimate review of specifications, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can be economically and effectively applied to both large and small scale systems.

7.2 Testing Process

It takes three inputs: an image tensor, a prediction dictionary from a model, and an optionalthreshold for displaying predictions.

7.2.1 Image Preparation

Convert Image to Numpy Array: The image tensor, which is likely in a PyTorch format, is first transferred from GPU to CPU (if necessary) and converted into a numpy array. The .transpose((1, 2, 0)) reorders the dimensions from (channels, height, width) to (height, width, channels) which is a standard format for image arrays in many visualization libraries like matplotlib.

7.2.2 Setup Visualization

- figure Setup: The function uses matplotlib to create a figure and an axes object, ax, which will be used to display the image.
- Display Image: The ax.imshow(image) method displays the numpy array as an image on the axes.

7.2.3 Process Predictions

- Define Label Names: A dictionary label names maps integer labels to their corresponding string names (e.g., "weed", "crop", "none").
- Iterate Over Predictions: The function loops through the bounding boxes, scores, and labels provided in the prediction dictionary. Each tuple consists of a bounding box, a score, and a label tensor.

7.2.4 Filtering and Drawing Bounding Boxes

- Score Thresholding: Each prediction score is checked against a threshold (default is 0.75). Predictions with scores below this threshold are ignored.
- Label Check: If the label corresponds to "none" (label 3), it can be optionally skipped, not drawing any bounding box for such predictions.
- Bounding Box and Label Setup: For valid predictions:
    1. Color Coding: The color of the bounding box is set based on the label (e.g., red for "weed", blue for "crop").
    2. Rectangle Creation: A Rectangle object is created using matplotlib patches to represent the bounding box on the image.
    3. Add Rectangle to Axes: The rectangle is added to the axes to outline the detected object.
    4. Add Text: A text label showing the category name (without the score) is displayed at the starting point of the bounding box. This text is displayed with a yellow background forbetter visibility.

7.2.5 Display the Result

Show Plot: Finally, plt.show() is called to display the resulting plot with the image, the bounding boxes, and the labels.

```python
def show_prediction(image, prediction, threshold=0.75):
    # Ensure the image tensor is on CPU and convert it to numpy
    image = image.cpu().data.numpy().transpose((1, 2, 0))
    fig, ax = plt.subplots(1)
    ax.imshow(image)

    # Define label names for convenience, including the 'none' label
    label_names = {1: 'weed', 2: 'crop', 3: 'none'}

    # Draw bounding boxes and labels
    for box, score, label_tensor in zip(prediction['boxes'], prediction['scores'], prediction['labels']):
        #print(score)
        #print(score.cpu().numpy())
        if score.cpu().numpy() > threshold:  # Ensure score is on CPU
            label = label_tensor.cpu().numpy().item()  # Move label tensor to CPU and extract the value

            # Skip drawing boxes for 'none' labels if desired
            if label == 3:
                continue  # Comment this line if you want to include boxes for 'none' labels

            # Choose color based on label
            color = 'r' if label == 1 else 'b' if label == 2 else 'g'
            #label_text = f"{label_names.get(label, 'unknown')}: {score.cpu().numpy():.2f}"
            label_text = f"{label_names.get(label, 'unknown')}"
            # Move box tensor to CPU and convert to numpy
            box = box.cpu().numpy()
            rect = patches.Rectangle((box[0], box[1]), box[2] - box[0], box[3] - box[1],
                                     linewidth=1, edgecolor=color, facecolor='none')
            ax.add_patch(rect)
            ax.text(box[0], box[1], label_text, bbox=dict(facecolor='yellow', alpha=0.5))

    plt.show()
```

Figure 7.1: Visualization Process

CHAPTER 8

RESULTS AND DISCUSSIONS

The results obtained from the proposed system. The Figure 8.1 gives the System user interface of working module of weed detection software system.In this page, the user can choose the input image for detection of a weed in that image. In Figure 8.2, the result of weed detection is shown. By using the Faster RCNN, the system can detect the weed accurately given in Figure 8.2. Shows the selected image, This system predicts the result by using the Faster RCNN classifier.. The system takes the image as an input and by using the algorithm, it detects whether the given image is a weed or not. The system refers to the dataset to identify the weed. In Figure 8.3 , it shows that the result of weed detection is shown. By using the RCNN classifier, The system is able to accurately detect the weed inthe figure.

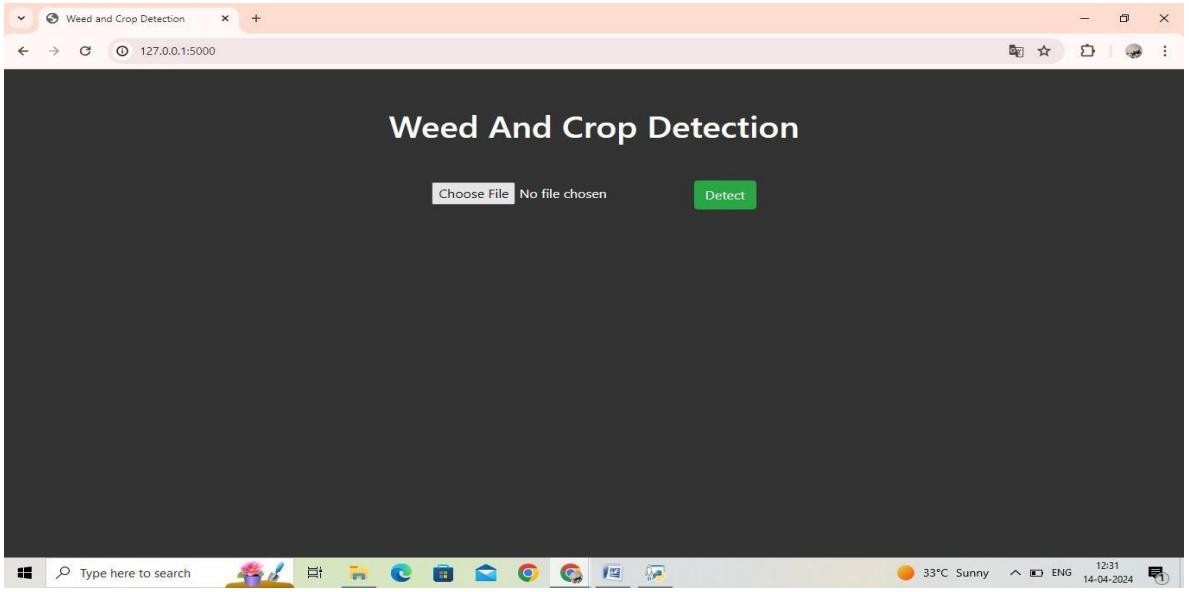

Figure 8.1: User Interface

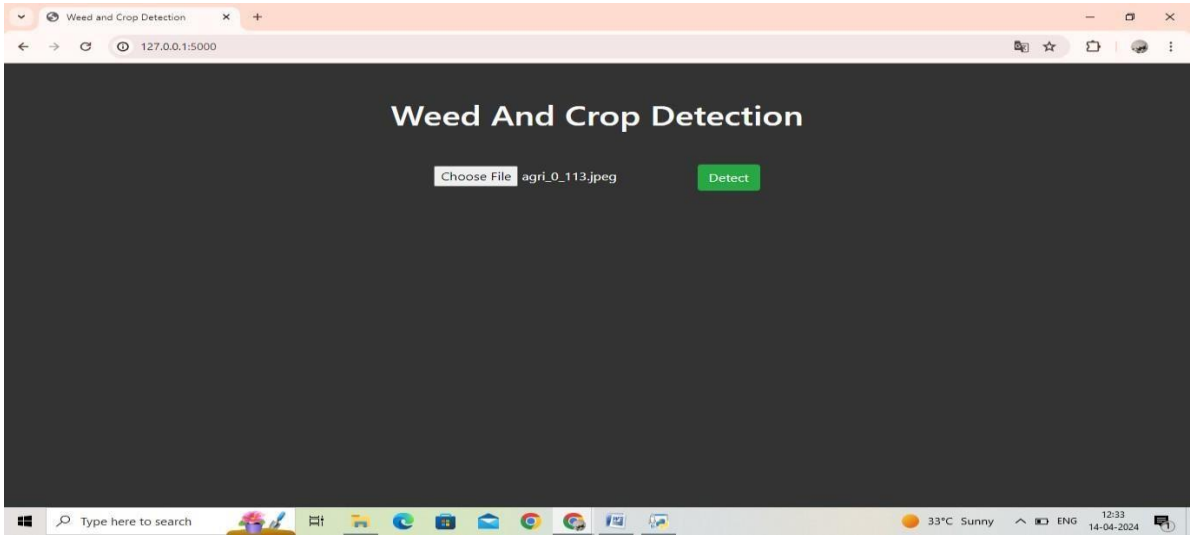

Figure 8.2:Image Uploaded

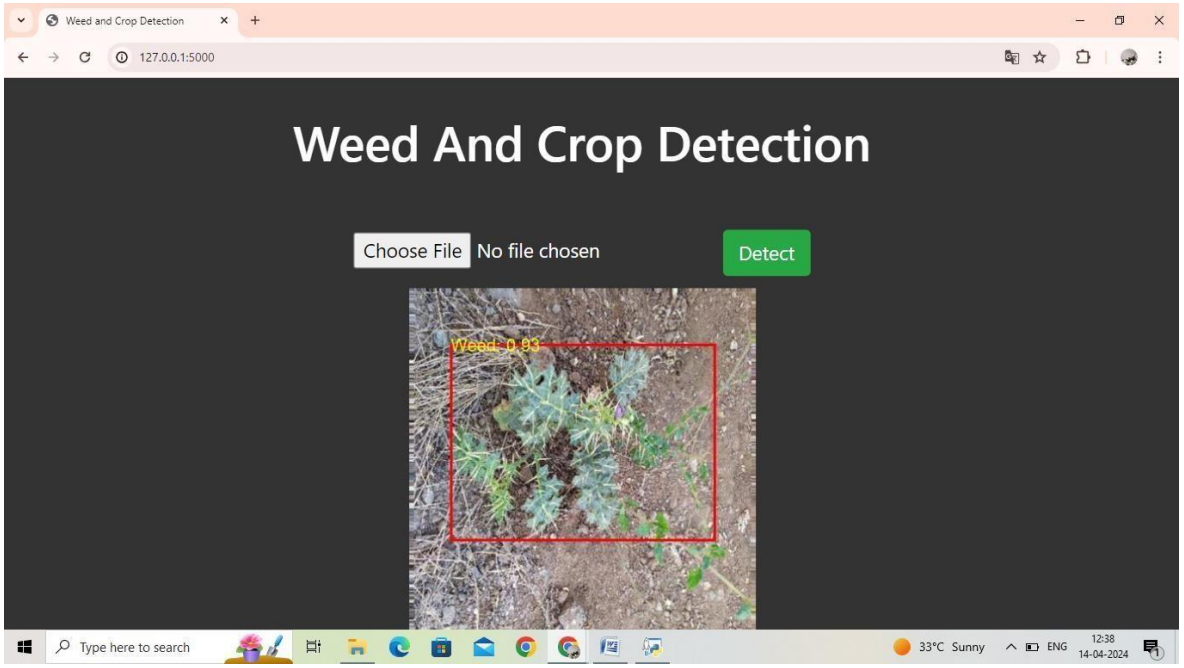

Figure 8.3: Image Detected

We have also developed the live web application, which opens the camera to detect whether the shown plant is weed or crop, These are the output for this live web app.

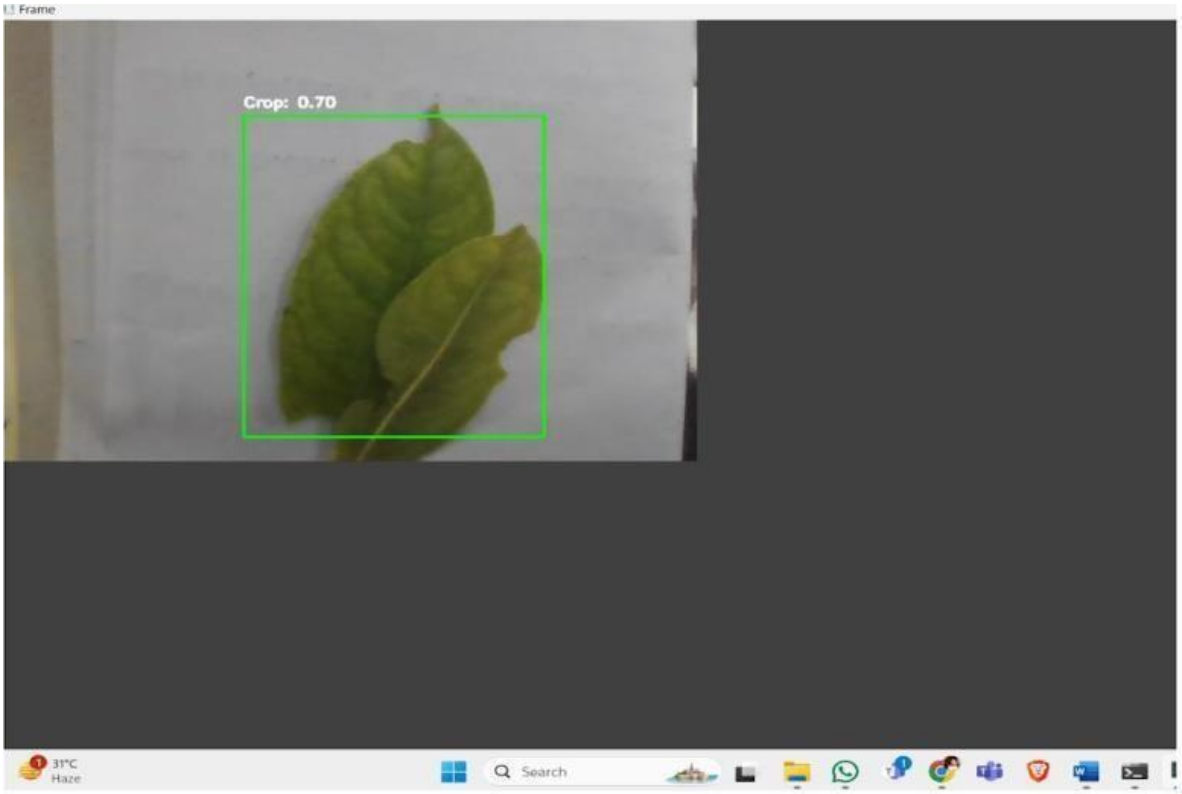

Figure 8.4: Live image detected

A total of 1,280 images were used for testing the system, and the system was able to detect a weed in 1,260 of those images successfully. It means the accuracy of this system is more than 98%. The graph of confidence score against images is plotted for ten images. These four random images were taken for testing, which were not used for training and this systemwas successful in identifying the weed from those images. The results show the ability ofthe system to capture low level features to improve the ability to detect small objects. Hence, the average confidence score of the system is more than 98%

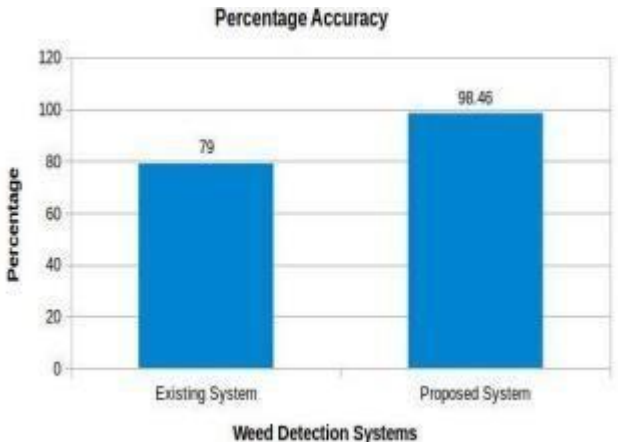

Figure 8.5: Percentage Accuracy

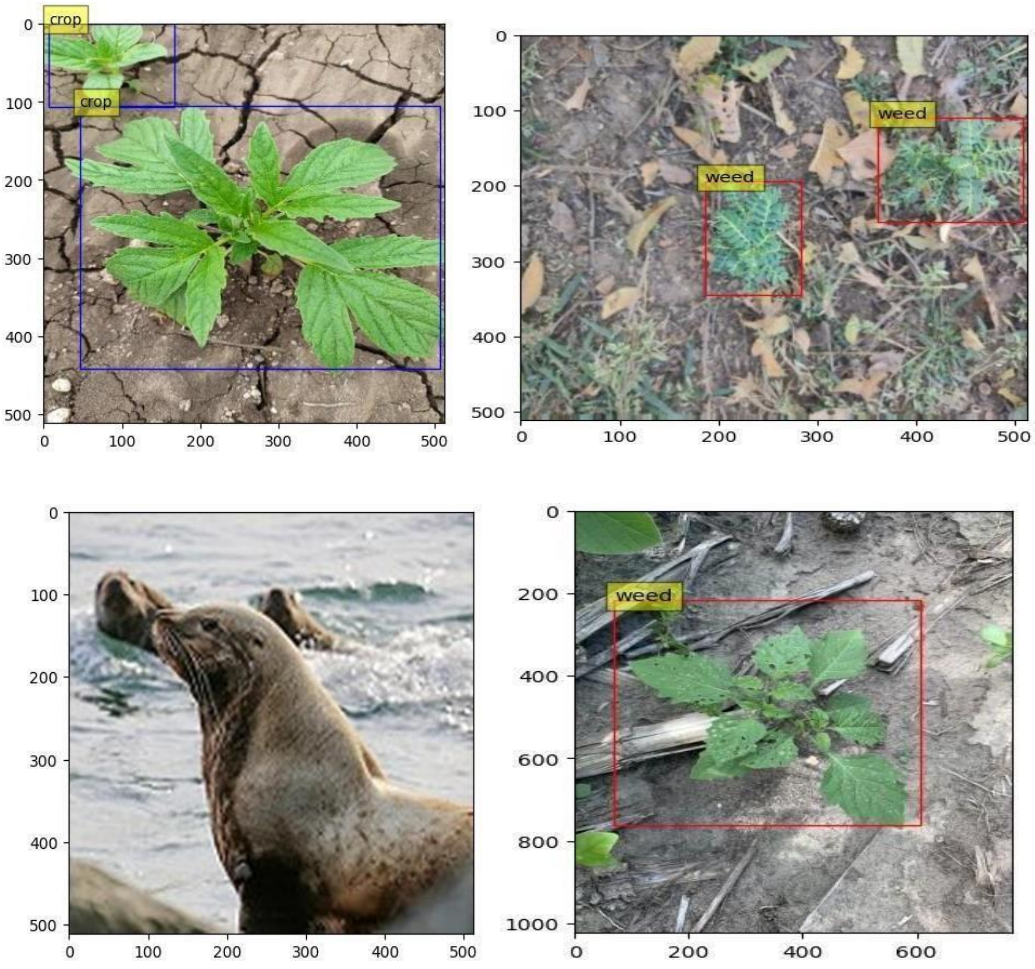

Figure 8.6: Random Images

CHAPTER 9
CONCLUSION

To improve the production of farmers, weed removal plays a vital role. There is a need to distinguish weeds and crops. The proposed work uses Faster RCNN to extract key features of weed images. The image processing and feature extraction using RCNN is a base of the proposed work for identification of weed. Deep learning approach is used to process captured images, assign the values to key attributes as per its features extracted. On the basis of various valued attributes of image the weed images are distinguished and identified.The proposed system uses RCNN based approach for image characterization so the accuracy of the system is more. The future work includes automation in the process of weedremoval from the crop.

REFERENCES

1. Basavarajeshwari, Prof. S. P. Madhavanavar Department of CSE, Basaveshwar Engineering College, Bagalkot Department of CSE, Basaveshwar Engineering College, Bagalkot ``A Survey on weed Detection using Image Processing``.(2018)

2. A. J. Irías Tejeda1 Faculty of engineering and architecture Universidad Tecnológica Centro americana (UNITEC), R. Castro Castro2 Faculty of engineering and architecture Universidad Tecnológica Centro americana (UNITEC) "Algorithm of Weed Detection in Crops by Computational Vision".(2019)

3. Bo Liu, Ryan Bruch "Weed Detection for Selective Spraying: a Review". (2020)

4. Yogesh Beeharry, Vandana Bassoo, Department of Electrical and Electronic Engineering Faculty of Engineering University of Mauritius "Performance of ANN and AlexNet for weed detection using UAV-based images".(2020)

5. XIAOJUN JIN 1, JUN CHE2, AND YONG CHEN1,"Weed Identification Using Deep Learning and Image Processing in Vegetable Plantation."(2021)