

**“E-KETHA” : ENRICHING RICE FARMER’S QUALITY
OF LIFE THROUGH A MOBILE APPLICATION.**

2022-81

Final Report

P.Y.D Jayasinghe

B.Sc. (Hons) Degree in Information Technology

Department of Computer Science and
Software Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

October 2022

**“E-KETHA” : ENRICHING RICE FARMER'S QUALITY
OF LIFE THROUGH A MOBILE APPLICATION.**

2022-81

Final Report

P.Y.D Jayasinghe – IT19117256

Supervisor: Mr. Adeepa Gunarathna

Co Supervisor: Ms. Amali Upeka Gunasinghe

B.Sc. (Hons) Degree in Information Technology

Department of Computer Science and
Software Engineering


Sri Lanka Institute of Information Technology

Sri Lanka

October 2022

DECLARATION, COPYRIGHT STATEMENT, AND THE STATEMENT OF THE SUPERVISOR

We declare that this is our own work and this proposal does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgment is made in the text.

Name	Student ID	Signature
P.Y.D Jayasinghe	IT19117256	

The supervisor/s should certify the proposal report with the following declaration.
The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

Signature of the supervisor:

Date:

Signature of the supervisor:

Date:

ABSTRACT

In our country of Sri Lanka, rice is the most common type of food that is consumed on a daily basis. Due to that rice farmers face a huge amount of stress to supply according to the massive demand. One of the main problems rice farmers are currently facing is the prevalence of weeds that plague rice fields. These weeds because they occupy rice fields, absorb nutrients from the soil all the while taking valuable space. These topics were chosen due to it being found that 35% of all harvest potentially being lost in Sri Lanka alone because of weeds. The aim is to develop a mobile application that will help farmers solve this particular problem. The application will use areal images to conduct image processing to analyze crops and highlight the weed-infested areas. This will enable the farmer to find the weed and take a picture of it to identify the type of weed also using image processing to finally receive solutions using machine learning and deep learning.

Keywords :- machine learning, image processing, deep learning, rice, weeds, mapping

TABLE OF CONTENTS

DECLARATION, COPYRIGHT STATEMENT, AND THE STATEMENT OF THE SUPERVISOR.....	iii
ABSTRACT.....	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	vii
LIST OF TABLES	vii
1. INTRODUCTION.....	1
1.1 Background	1
1.2 Literature Survey	3
1.2.1 Deep Neural Networks to Detect Weeds from Crops in Agricultural Environments in Real-Time	3
1.2.2 Detection of Weed and Wheat Using Image Processing	3
1.2.3 Weed Detection in Rice Fields Using Remote Sensing Technique.....	3
1.3 Research Gap	5
2. RESEARCH PROBLEM	5
3. RESEARCH OBJECTIVES	6
3.1 Main Objectives	6
3.2 Specific Objectives	6
4. METHODOLOGY	7
4.1 Methodology	7
4.1.1 Research Area	8
4.1.2 Requirement Gathering and Analyzing	8
4.1.3 Design	9
4.1.4 Tools and Technologies.....	9
4.1.5 Data acquisition.....	10
4.2 Commercialization aspects of the product.....	10
4.2.1 Target audience	10
4.2.2 Design of the app	10
4.2.3 Gap in the market	11
4.2.4 Marketing plan.....	11
4.2.5 Pricing	11
4.2.6 Budget	11

4.2.7 WBS.....	12
4.2.8 Gantt Chart	13
5. Testing & Implementation	14
5.1 Implementation	14
5.1.1 Pre-processing	14
5.1.2 Model.....	14
5.2 Testing and Maintenance	20
6. RESULTS AND DISCUSSION	20
6.1 Results	20
6.1.1 Identification of weeds and providing solutions:.....	20
6.1.2 Weed Mapping	22
6.2 Research Findings.....	24
6.2.1 Identification of weeds and providing solutions:.....	24
6.2.2 Weed Mapping	24
6.3 Discussion.....	25
6.3.1 Identification of weeds and providing solutions:.....	25
6.3.2 Weed Mapping:.....	25
6.4 Summary of Each Student's contribution	25
Conclusions.....	26
REFERENCE LIST	27
Glossary	28
Appendices.....	29

LIST OF FIGURES

Figure 1: Percentage of rice yield lost due to weeds in multiple countries.....	1
Figure 2: Percentage of economic loss due to weeds in multiple crops.....	2
Figure 3: Weed detection overview	7
Figure 4:WBS Structure.....	12
Figure 5:Gantt Chart	13
Figure 6:Jaccard accuracy chart.....	22
Figure 7:Jaccard loss chart.....	23
Figure 8:Weed mapping output	24

LIST OF TABLES

Table 1: Comparing existing applications and our application features	5
Table 2: Modal Accuracy.....	20
Table 3: Modal Accuracy.....	22

1. INTRODUCTION

1.1 Background

As the most consumed food in Sri Lanka, the demand for rice is quite high. One of the major reasons why we cannot supply according to the demand is the produce lost due to the weeds that can be widespread through the fields such as Bermuda grass and Filicopsida.

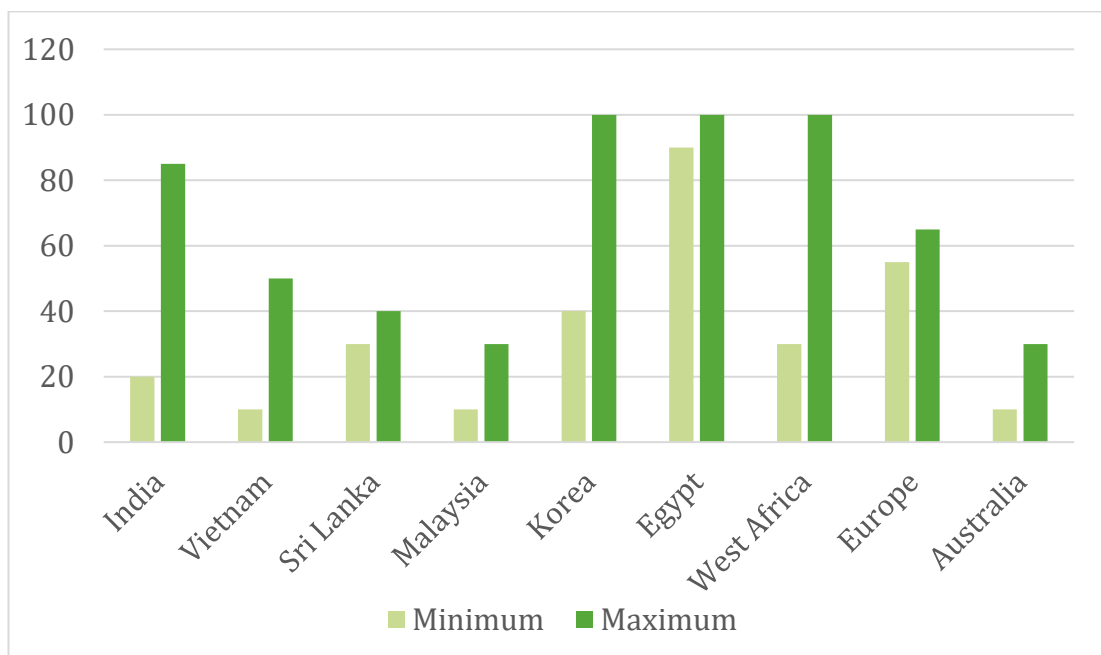


Figure 1: Percentage of rice yield lost due to weeds in multiple countries

As it is shown in figure 1 [1], an average of 35% of rice is lost annually in the country of Sri Lanka alone. When it comes to majorly developed countries and continents such as Australia and Europe, have fluctuating averages with the likes of 52% and 20% each. This is enough evidence to understand the threat that weeds possess to rice yields. Due to these losses, Sri Lankan farmers lose much of their profit as well as the Sri Lankan general populous who has to depend on foreign imports since the supply does not match the demand.

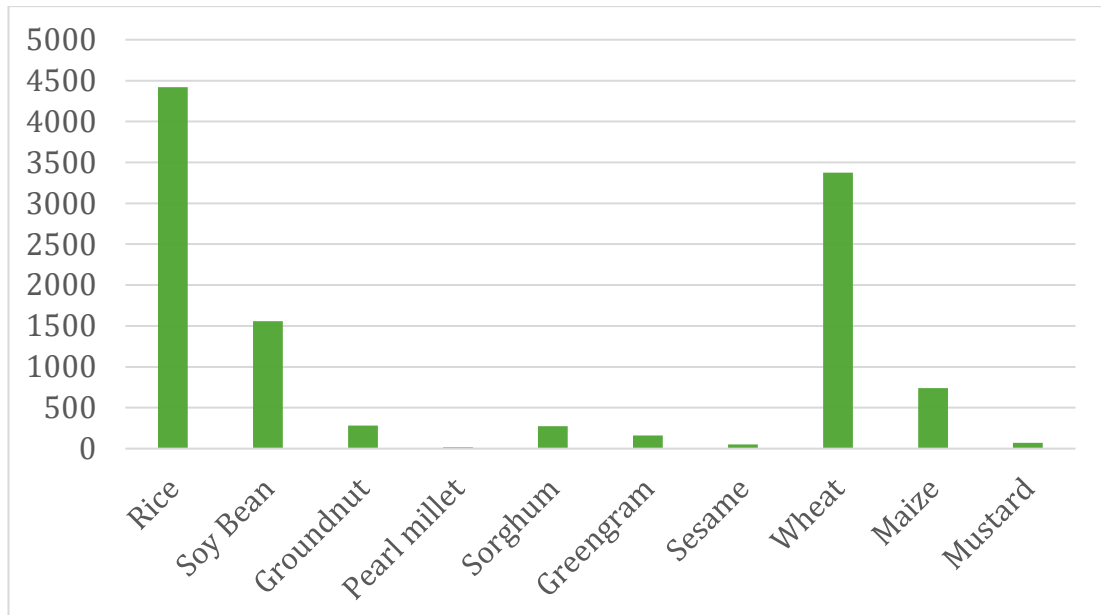


Figure 2: Percentage of economic loss due to weeds in multiple crops

Even when looking at economic losses due to weeds in a multitude of crops, rice takes the most losses at a whopping 4420 million USD in India with only wheat coming a close second at 3376 million. This is proof enough that rice demands the most attention rather than any other crop [2].

How these weeds affect rice crops is by absorbing the many nutrients that should have gone towards rice crops as well as occupying the already limited space of paddy fields [7]. Since weeds tend to be widespread in a swift manner, traditional farmers have difficulties removing them one by one [10]. Due to there being many types of weeds, choosing the proper weedicide can be an arduous task as well [8][9]. This is all the while there are talks about improper use of weedicide causing cancer and other negative side effects [11].

As a solution for all these issues, a mobile application will be proposed.

1.2 Literature Survey

1.2.1 Deep Neural Networks to Detect Weeds from Crops in Agricultural Environments in Real-Time

This study mainly done by Ildar Rakhmatuulin and Andreas Kamilaris aims to use machine vision with artificial intelligence and deep learning methods. This is due to pre-trained models, large datasets and deep learning tend to be more accurate than other previous techniques. This is all with the additional use of even detecting pests. This research proposes an AI-based system that has the capability of detecting weeds by emphasizing new deep-learning trends. This is so that this can be used by robots to effectively control the spread of weeds. While it has been concluded that there is no current neural network that is best suited for real-time weed detection, nonetheless this research encourages scientists to continue their efforts with the expectation that more solutions will be discovered in the near future [3].

1.2.2 Detection of Weed and Wheat Using Image Processing

This study mainly done by Sarmad Hameed and Imran Amin aims to use image processing as the main technology to detect weeds in the field of wheat production. What is being proposed here is the use of UAV (Unmanned Aerial Vehicles) to acquire data on wheat crops in the model of RGB images. Then background subtraction is used to identify weeds, wheat, and the barren land in the field. In the end, even though this was found to be good enough, the use of Neural Networks, if applied in the future would grant better results. Another main challenge was the sunlight intensity issue that also plagued the images [4].

1.2.3 Weed Detection in Rice Fields Using Remote Sensing Technique

This study done by mainly Rhushalshafira Rosle, Nik Norasma Che'Ya, and a few others, aims to detect the weeds in rice fields using remote sensing techniques. The process is to first use a device such as a camera or UAV (unmanned aerial vehicle) to capture images of paddy fields from above. Then the different types of sensors attached to those devices will collect the data with the image-capturing process. The sensors are the RGB Sensor, Multispectral Sensor, and Hyperspectral Sensor with each of them having its own sets of advantages and disadvantages, acting together to capture the best images possible. Finally, machine learning and deep learning

algorithms will be used to classify the images with them being found to generate a map that ranges from 85% to 99% accuracy [5].

1.2.4 Weed Classification for Site-Specific Weed Management Using Automated Stereo Computer-Vision

This study is done by Mojtaba Dadashzadeh, Yousef Abbaspour-Gilandeh, Tarahom Mesri-Gundoshmian, and Sajad Sabzi with the goal of classifying weeds in a specific site using a stereo vision system to distinguish rice plants and weeds. This system is further augmented using an artificial neural network and two other metaheuristic algorithms, them being y particle swarm optimization (PSO) and the bee algorithm (BA). With stereo videos being recorded of the site beforehand and decomposed into singular frames, rice plants were extracted using color, shape, and even texture. Then the previously mentioned metaheuristic algorithms were used to optimize the neural network and classify the weed detected as well. According to the K-nearest neighbors (KNN) classifier, this reached f 88.74% and 87.96% for right and left channels without accounting arithmetic or the geometric means as the basis and with it o 92.02% and 90.7% respectively [6].

1.2.5 ID Weeds

Originally created by the University of Missouri's College of Agriculture, Food and Natural Resources Division of Plant Science. This mobile application aims to allow the user to search for weed names through their common or Latin name, view a weed list, and also search using different characteristics inherent to weeds. Weed details and their respective photographs are presented here [12].

1.3 Research Gap

	1.2.1	1.2.2	1.2.3	1.2.4	1.2.5	E-Ketha
Weed Type Detection	✓	✓	✓	✓	✓	✓
Rice Centric	✗	✗	✓	✓	✗	✓
Ground Imagery Used	✓	✗	✗	✓	✗	✓
Aerial Imagery Used	✗	✓	✓	✗	✗	✓
Classify Weed Details	✗	✗	✓	✓	✓	✓
Propose Solution	✓	✗	✗	✗	✗	✓
Preparation Needed	✗	✗	✓	✓	✗	✗

Table 1: Comparing existing applications and our application features

2. RESEARCH PROBLEM

One of the major issues found is the growth of unwanted weeds that are prevalent in paddy fields. While weeds do not directly harm rice crops, weeds absorb nutrients from the paddy fields that should have gone for the development of healthy rice plants [7].

The identification of weeds however is not difficult for the common farmer on small fields, but it becomes arduous for larger ones. So there is a need for the farmer to see where the weed growth is concentrated in the paddy field [10].

Another challenge lies in the recognition of proper weedicide to combat the identified weed by identifying the type of the weed. This is due to the vast amount of weed types and the equally wide variety of weedicides being difficult to recollect for the common farmer. [8].

There are issues also with the Sri Lankan government banning weedicides containing glyphosate due to it being classified as a "probable carcinogen" by the World Health Organization. Even though this ban was later lifted it still is fresh in the mind of farmers and they are wary of it. [11]

3. RESEARCH OBJECTIVES

3.1 Main Objectives

The main objective of this component is to identify weeds that are commonly spread in rice fields. This is done using a mobile-based application and images that are inputted into it. The first part is to highlight the areas of the paddy field that are rich in weed through the use of aerial imagery fed to the application. Then image processing technology is used to compare and contrast the weed-rich areas and show them to the app user. Then the user has the ability to take an image of the weeds up close to identify their types also with the help of image processing. Finally, the application will use machine learning and deep learning to show the best solution to combat the particular weed type.

3.2 Specific Objectives

01. The application identifies the weed-infested areas using the aerial imagery provided.

The application user inputs aerial imagery captured using an application device or any other. The application compares the rice crops to any other weeds and vice versa using the data in the database to identify weed-ridden areas.

02. Application identifies the type of weed using the images captured.

Once the images are taken, the application identifies and classifies what type of weed is prominently growing in the rice field. This is then shown to the user.

03. Application will provide solutions on how to best deal with the weeds.

The application will suggest treatments and solutions on the best on how to remove the weeds from the paddy field.

4. METHODOLOGY

This section will entail the details on the techniques and mechanisms that are employed to create the Weed Detection component, belonging to the “e-ketha” application from the data gathering stage all the way to implementation.

4.1 Methodology

When it comes to rice fields or any other related crop fields, weeds are one of the primary concerns that a farmer has to deal with.

In order to prepare a solution, large databases and datasets were analyzed to create a comprehensive database for the application to function. This database contains various kinds of weed information, let it be their type, color, shape, etc.

Once the user uploads an image of the weed as an aerial image, FCN (Fully Convolutional Network) algorithm that was implemented will highlight the areas with weeds using image segmentation.

Users will also have the capability to upload images containing weed up close, which will in turn prompt the CNN (Convolutional Neural Network) technology to identify the weed information such as their type. Using this information, the machine learning and deep learning algorithms in the application will give the best solution possible so as to remove them without harming the rice crops.

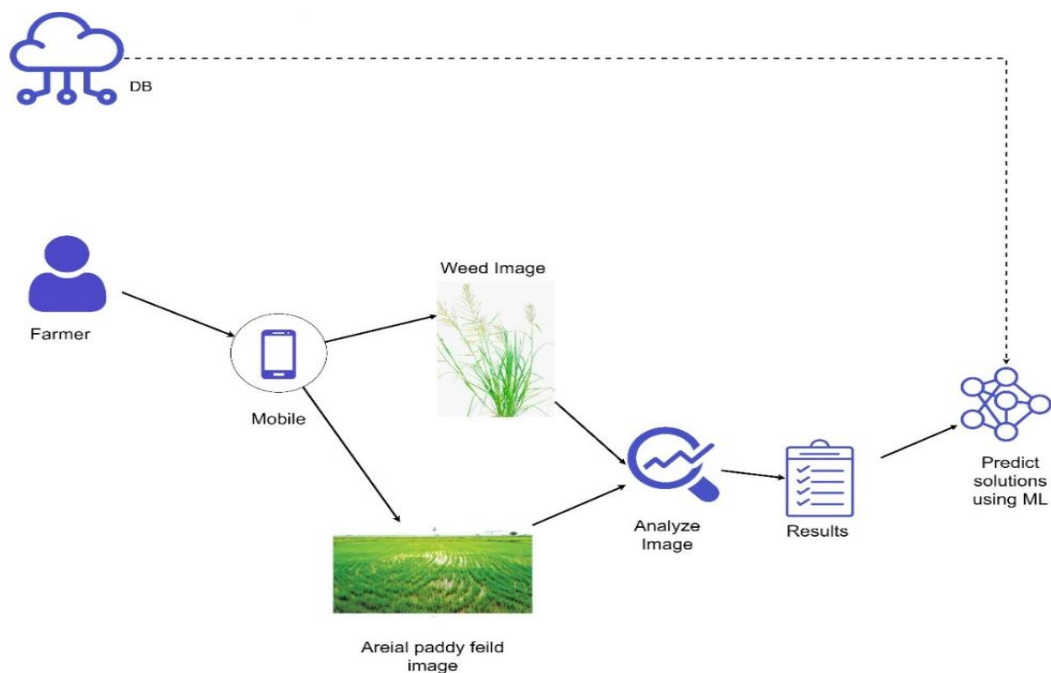


Figure 3: Weed detection overview

4.1.1 Research Area

When it comes to the research area, four features were identified. Such as Image processing activities, Classification activities, Detection activities, and finally solution prediction. In order to conduct the research, deep learning technology has been taken as the core foundation.

4.1.2 Requirement Gathering and Analyzing

Due to the importance of requirement gathering and analysis, major emphasis was put on this section. Since there is a need for this process to be strictly on the "weed identification and solution finding" part below mentioned approaches were used.

- Reading research papers relevant to the research problem.
- Studying existing systems related to our research area.
- Contacted experts at Rice Research and Development Institute(RRDI), Bathalagoda.
- Met with Sri Lankan paddy farmers.

To get an idea about the research problem, studying related research papers is a must. The next step was to understand what types of systems that already exist, so as to see what is lacking and needs improvements. Finally, to see if the proposed solution is viable in the current environment, specialists on the field and traditional farmers were contacted.

4.1.2.1 Functional requirements

- Ability to upload aerial imagery.
- Mapping the weeds.
- Ability to upload weed imagery.
- Identify weed type.
- Propose solutions.
- Show proposed solutions.

4.1.2.2 Non-functional requirements

- Reliability
- Accuracy
- Availability
- Performance
- User-friendliness

4.1.3 Design

The design phase encompasses what is needed for the estimation of hardware and system requirements by the creation of a system architecture, due to the needs and specifications being included. The architecture will entail the components separated into manageable levels according to the respective research project member. In this case, it will be the "identification of weeds and proposing solutions" component.

4.1.4 Tools and Technologies

4.1.4.1 Tools

- Android Studio
 - This is chosen due to it being the primary IDE recommended for Java mobile application development. The user-friendliness coupled with the performance, security, and feature richness also makes this the most suitable option.
- Google collab
 - Since some of the deep learning models require high amounts of computational resources a virtual environment like google collab is most appropriate.
- Google Drive
 - Since google collab is used for the model training, the dataset cannot be stored on personal computers thus google drive is needed for storing the dataset.

4.1.4.2 Technologies

- Deep learning
 - Deep learning is the only solution for image classification and identification tasks such as this. Due to there being no similar prior work, a model has to be created and trained from scratch.
- Models
 - U-Net
 - U-Net was identified to be the best for the mapping of weeds.
 - ResNet
 - Resnet was identified to be the best for the identification of weeds.

The evidence for this is provided below within the methodology.

- Android Java

- Since the application is initially targeted toward android devices, in order to provide the smoothest experience possible native android java is used.
- Firebase
 - Due to the application requiring a real-time online connection to the database firebase is chosen as the primary database. Since the data set mostly consists of images the need for a document-based database is further insinuated.
- Python
 - For the machine learning and deep learning parts of the application, python is used due to the wide range of libraries and frameworks available for such tasks compared to other languages. The simplicity and consistency with the large community are also a benefit.
 -

4.1.5 Data acquisition

Deep weeds TensorFlow dataset containing 17,509 images split into 8 different weed species has been used for the weed identification model training and they are "Black River", "Charters Towers", "Cluden", "Douglas", "Hervey Range", "Kelso", "McKinlay" and "Paluma" used for detection of weeds and providing solutions.

The dataset used for the weed mapping component contains UAV (Unmanned Aerial Vehicle) imagery including ground truth images corresponding to the UAV imagery. The labels used for the training are “Weed”, “Rice”, “Sand” and “Other” used for weed mapping from UAV images.

4.2 Commercialization aspects of the product

4.2.1 Target audience

The primary target audience for this application will be rice farmers with rice suppliers, researchers, buyers, sellers, and any person who is connected to the rice farming process being the secondary audience.

4.2.2 Design of the app

A comprehensive and easily understandable UI and UX are created so that even non tech-savvy users will not be confused while using the application. This will make sure that the application will reach a wide audience.

4.2.3 Gap in the market

Currently, in the play store, there is no other similar application to be found. This already makes the application unique and stand out.

4.2.4 Marketing Plan

The initial incentive will be to introduce this application to the farmers themselves. This will enable us to get feedback directly from the primary target audience which will then make it easier to enhance and optimize the application further, thus making a better product.

This application will be promoted by famous influencers and through social media platforms.

4.2.5 Pricing

Most of the functionality will be provided for free with them including,

- Pest identification
- Disease identification
- Weed identification
- Fertilizer type identification
- Weed mapping
- Growth deficiency identification

This will be with a free application.

In order for the solution providing functionality associated with the above-mentioned features to be enabled, a small price will have to be paid on a monthly basis.

4.2.6 Budget

A price will have to be allocated for the influencer and social media promotions. In order to publish the application another, sum also has to be assigned. Finally, the database will also be required of a monthly payment.

4.2.7 WBS

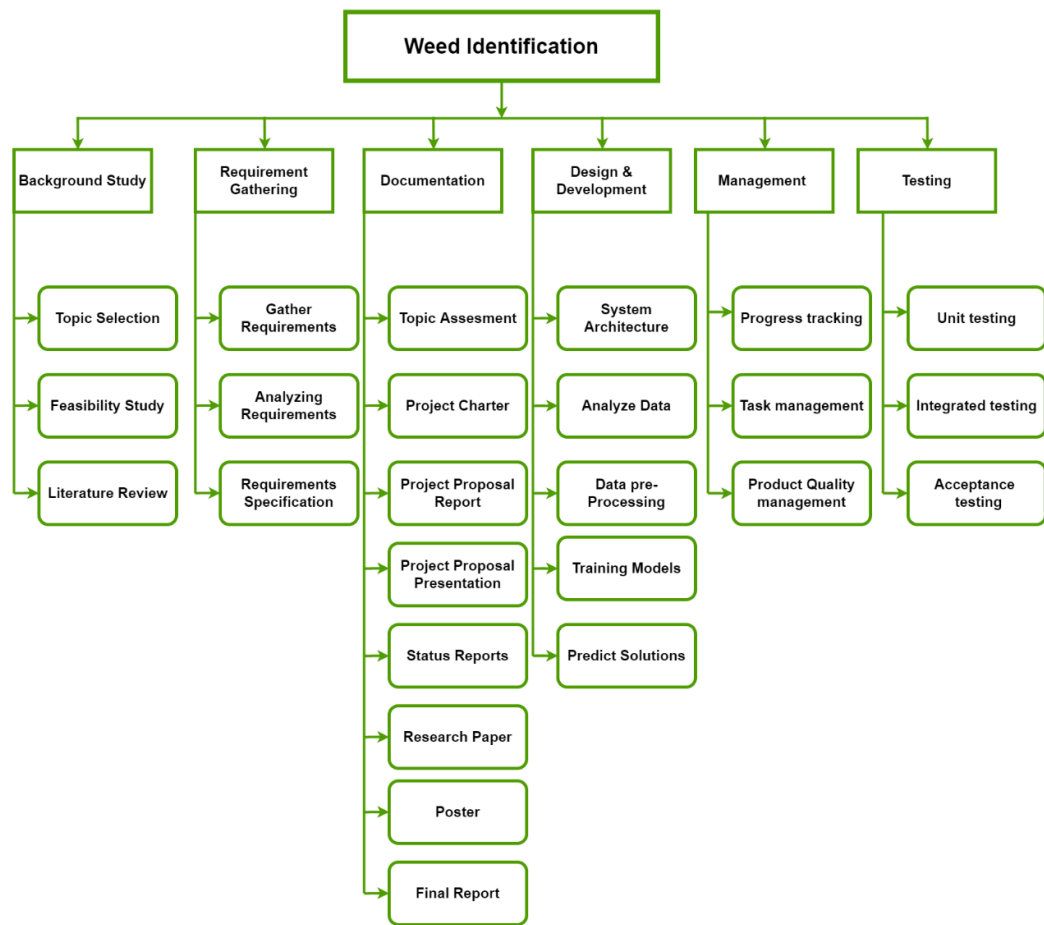


Figure 4:WBS Structure

4.2.8 Gantt Chart

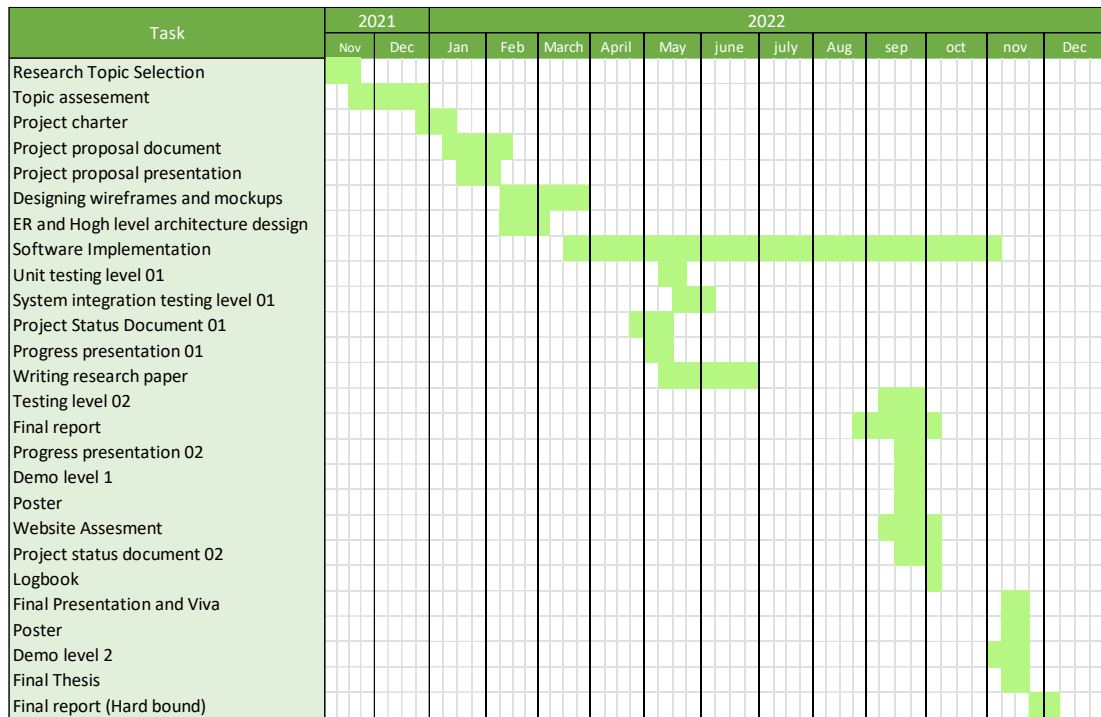


Figure 5:Gantt Chart

5. Testing & Implementation

5.1 Implementation

In this stage of the project, the implantation of the system will be started. This will be in accordance with the system architecture proposed in the previous design phase. The "Identification weeds and proposing solutions" component will be further split into three subcomponents, with them being

- Mapping of weeds using aerial imagery
- Identification of weeds using imagery
- Proposing solutions.

5.1.1 Pre-processing

When it comes to pre-processing all the models that are described below went through the same process. Which includes shuffling, resizing, rescaling, and flipping horizontally and vertically. Finally, normalization was performed according to the mean and standard deviation calculated for the datasets.

5.1.2 Model

5.1.2.1 DETECTION OF WEEDS AND PROVIDING SOLUTIONS

For Weed identification, the ResNet (Residual Network) [8] model was used with modifications made to improve the model. This model was chosen to answer the issue of vanishing or exploding gradient which is a nuisance in deep neural networks that have many layers. Vanishing or exploding gradient is the gradient becoming zero or becoming a large number with the increase of layers thus providing a high error rate on both training and test datasets. How ResNet archives this is by using the concept of residual blocks which utilize the technique of skip connections which connect activation layers to oncoming layers by skipping layers in the middle. It decides to skip by seeing if the next layer is damaging the performance. ResNet50 model is used here due to the reasons of giving the best results as well as 224x224 pixel size images being used as the dataset. The 50 after the model's name is the number of layers in the model as such ResNet50 contains 50 layers.

Torch is used as the main framework, providing the ResNet library as well as its lr_scheduler which gives the optimal learning rate. Torch is used as it provides flexibility and speed for deep neural network implementation such as in this case. The dataset was separated into 80% and 20% respectively for the train and test sets. As for the valid set, a further 10% was taken from the train set. Top-k accuracy is

used to predict the label for the prediction as the classes can look very similar to each other and in this dataset, some classes look very similar to each other. In the model, there is an initial 7x7 convolutional layer including batch normalization. This batch normalization allows for re-scaling and re-centering, thus making the training process significantly faster. ReLU has been chosen as the activation function due to it not activating all the neurons at the same time, thus keeping the operational costs at a manageable level. Downsampling max pooling is used in a similar vein. For the basic block that comes after the initial layers, 3x3 convolutional layers have been given, and lastly for the bottleneck block 1x1,3x3 along with 1x1 convolutional layer. Hyperparameter tuning [6] was performed for the parameters of batch size, learning rate, and epochs. Due to there being research showing that higher values for learning rate and batch size do not always provide Higher results, a lower number was chosen initially with it gradually going higher. As for the epochs, a brute force method was used to see which would be best.

ResNet model summary

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (layer1): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (layer2): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
    )
  )
)
```

```

)
(4): Bottleneck(
  (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(5): Bottleneck(
  (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
)
)
(layer4): Sequential(
  (0): Bottleneck(
    (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (downsample): Sequential(
      (0): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (1): Bottleneck(
    (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
  (2): Bottleneck(
    (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
)
)
(avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
(fc): Linear(in_features=2048, out_features=1000, bias=True)

```

5.1.2.2 WEED MAPPING FROM UAV IMAGES

The custom modification of U-net [9], which use a fully convolutional network is used for the semantic segmentation task of mapping the weeds from the paddy field. U-net is specifically chosen due to its high performance when it comes to segmentations that have few labels such as the case in this dataset.

Tensorflow Keras is used as the main framework due to it providing all the necessary abstractions as well as building blocks needed for the implementation of the u-net model. The model itself is made up of 23 convolutional layers split into contraction, bottleneck, and expansion sections. Contraction includes 3x3 convolutional layers with 2x2 max pooling. The expansion layer has 3x3 CNN layers with a 2x2 convolution layer. Finally, the expansion section has multiple blocks that contain 3x3 CNN layers with 2x2 upsampling layers. After some modifications and changes, the default model was proven to give the best output. The conv2D layers in the model use the “ReLU” activation function and “he_normal” kernel initializer. The former is due to the reason mentioned in the previous part and the former is due to it being better coupled together with ReLU. Hyperparameter tuning [6] was performed for the parameters of batch size, learning rate, and epochs. Due to there being research showing that higher values for learning rate and batch size do not always provide Higher results, a lower number was chosen initially with it gradually going higher. As for the epochs, a brute force method was used to see which would be best.

U-net model summary

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 256, 256, 3)]	0	[]
conv2d (Conv2D)	(None, 256, 256, 16)	448	['input_1[0][0]']
dropout (Dropout)	(None, 256, 256, 16)	0	['conv2d[0][0]']
conv2d_1 (Conv2D)	(None, 256, 256, 16)	2320	['dropout[0][0]']
max_pooling2d (MaxPooling2D)	(None, 128, 128, 16)	0	['conv2d_1[0][0]']
conv2d_2 (Conv2D)	(None, 128, 128, 32)	4640	['max_pooling2d[0][0]']
dropout_1 (Dropout)	(None, 128, 128, 32)	0	['conv2d_2[0][0]']
conv2d_3 (Conv2D)	(None, 128, 128, 32)	9248	['dropout_1[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 32)	0	['conv2d_3[0][0]']
conv2d_4 (Conv2D)	(None, 64, 64, 64)	18496	['max_pooling2d_1[0][0]']
dropout_2 (Dropout)	(None, 64, 64, 64)	0	['conv2d_4[0][0]']
conv2d_5 (Conv2D)	(None, 64, 64, 64)	36928	['dropout_2[0][0]']
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 64)	0	['conv2d_5[0][0]']
conv2d_6 (Conv2D)	(None, 32, 32, 128)	73856	['max_pooling2d_2[0][0]']
dropout_3 (Dropout)	(None, 32, 32, 128)	0	['conv2d_6[0][0]']
conv2d_7 (Conv2D)	(None, 32, 32, 128)	147584	['dropout_3[0][0]']
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 128)	0	['conv2d_7[0][0]']
conv2d_8 (Conv2D)	(None, 16, 16, 256)	295168	['max_pooling2d_3[0][0]']
dropout_4 (Dropout)	(None, 16, 16, 256)	0	['conv2d_8[0][0]']
conv2d_9 (Conv2D)	(None, 16, 16, 256)	590080	['dropout_4[0][0]']
conv2d_transpose (Conv2DTranspose)	(None, 32, 32, 128)	131200	['conv2d_9[0][0]']
concatenate (Concatenate)	(None, 32, 32, 256)	0	['conv2d_transpose[0][0]', 'conv2d_7[0][0]']

conv2d_10 (Conv2D)	(None, 32, 32, 128)	295040	['concatenate[0][0]']
dropout_5 (Dropout)	(None, 32, 32, 128)	0	['conv2d_10[0][0]']
conv2d_11 (Conv2D)	(None, 32, 32, 128)	147584	['dropout_5[0][0]']
conv2d_transpose_1 (Conv2DTranspose)	(None, 64, 64, 64)	32832	['conv2d_11[0][0]']
concatenate_1 (Concatenate)	(None, 64, 64, 128)	0	['conv2d_transpose_1[0][0]', 'conv2d_5[0][0]']
conv2d_12 (Conv2D)	(None, 64, 64, 64)	73792	['concatenate_1[0][0]']
dropout_6 (Dropout)	(None, 64, 64, 64)	0	['conv2d_12[0][0]']
conv2d_13 (Conv2D)	(None, 64, 64, 64)	36928	['dropout_6[0][0]']
conv2d_transpose_2 (Conv2DTranspose)	(None, 128, 128, 32)	8224	['conv2d_13[0][0]']
concatenate_2 (Concatenate)	(None, 128, 128, 64)	0	['conv2d_transpose_2[0][0]', 'conv2d_3[0][0]']
conv2d_14 (Conv2D)	(None, 128, 128, 32)	18464	['concatenate_2[0][0]']
dropout_7 (Dropout)	(None, 128, 128, 32)	0	['conv2d_14[0][0]']
conv2d_15 (Conv2D)	(None, 128, 128, 32)	9248	['dropout_7[0][0]']
conv2d_transpose_3 (Conv2DTranspose)	(None, 256, 256, 16)	2064	['conv2d_15[0][0]']
concatenate_3 (Concatenate)	(None, 256, 256, 32)	0	['conv2d_transpose_3[0][0]', 'conv2d_1[0][0]']
conv2d_16 (Conv2D)	(None, 256, 256, 16)	4624	['concatenate_3[0][0]']
dropout_8 (Dropout)	(None, 256, 256, 16)	0	['conv2d_16[0][0]']
conv2d_17 (Conv2D)	(None, 256, 256, 16)	2320	['dropout_8[0][0]']
conv2d_18 (Conv2D)	(None, 256, 256, 4)	68	['conv2d_17[0][0]']

=====

Total params: 1,941,156
Trainable params: 1,941,156
Non-trainable params: 0

5.2 Testing and Maintenance

As the final phase of the Software development life cycle(SDLC) is the testing and maintenance phase which will be done under the disciplines of functional and non-functional testing. The functional testing will mainly consider the functional requirements of the system and unit testing will be taken as the basis. Then in order to check the nonfunctional requirements such as performance and availability various nonfunctional testing will be conducted. As for the maintenance of the application after the publication, various support features will be added.

6. RESULTS AND DISCUSSION

6.1 Results

6.1.1 Identification of weeds and providing solutions:

Epoch	Train loss	Train accuracy	Val_loss	Val_accuracy
06	0.122	96.04%	0.166	95.03%
07	0.064	97.86%	0.199	96.30%
08	0.042	98.53%	0.095	96.73%
09	0.021	99.35%	0.082	97.16%
10	0.019	99.49%	0.088	97.09%

Table 2: Modal Accuracy

- Test Loss: 0.698
- Test Accuracy: 85.62%

This was done for both test and train portions of the dataset in the 20% and 80% split respectively. As for the valid set, a further 0.1 was taken from the train set. 32 batch size was chosen and according to research after providing the best level of predictions with shuffling enabled.

	<u>Chinee Apple</u>	<u>Lantana</u>	<u>Negative</u>	<u>Parkinsonia</u>	<u>Parthenium</u>	<u>Prickly Acacia</u>	<u>Rubber Vine</u>	<u>Siam Weed</u>	<u>Snake Weed</u>
<u>Chinee Apple</u>	209	0	12	0	1	0	0	0	3
Lantana	2	185	21	0	0	0	1	4	0
Negative	14	14	1635	19	21	47	55	1	16
Parkinsonia	1	0	9	197	0	0	0	0	0
Parthenium	1	0	15	0	189	0	0	0	0
<u>Prickly Acacia</u>	9	1	74	13	13	102	0	0	1
<u>Rubber Vine</u>	3	0	13	0	0	0	185	0	1
<u>Siam Weed</u>	0	1	36	0	0	0	0	176	2
<u>Snake Weed</u>	35	10	25	0	1	0	0	4	129

These are the results that were fetched once the validated data has been given a prediction.

6.1.2 Weed Mapping

Epoch	Loss	accuracy	Val_loss	Val_accuracy
16/20	0.1409	0.9458	2.7952	0.6428
17/20	0.1406	0.9461	2.7834	0.6446
18/20	0.1373	0.9465	2.6865	0.6497
19/20	0.1338	0.9479	2.8509	0.6445
20/20	0.1298	0.9488	3.0347	0.6439

Table 3: Modal Accuracy

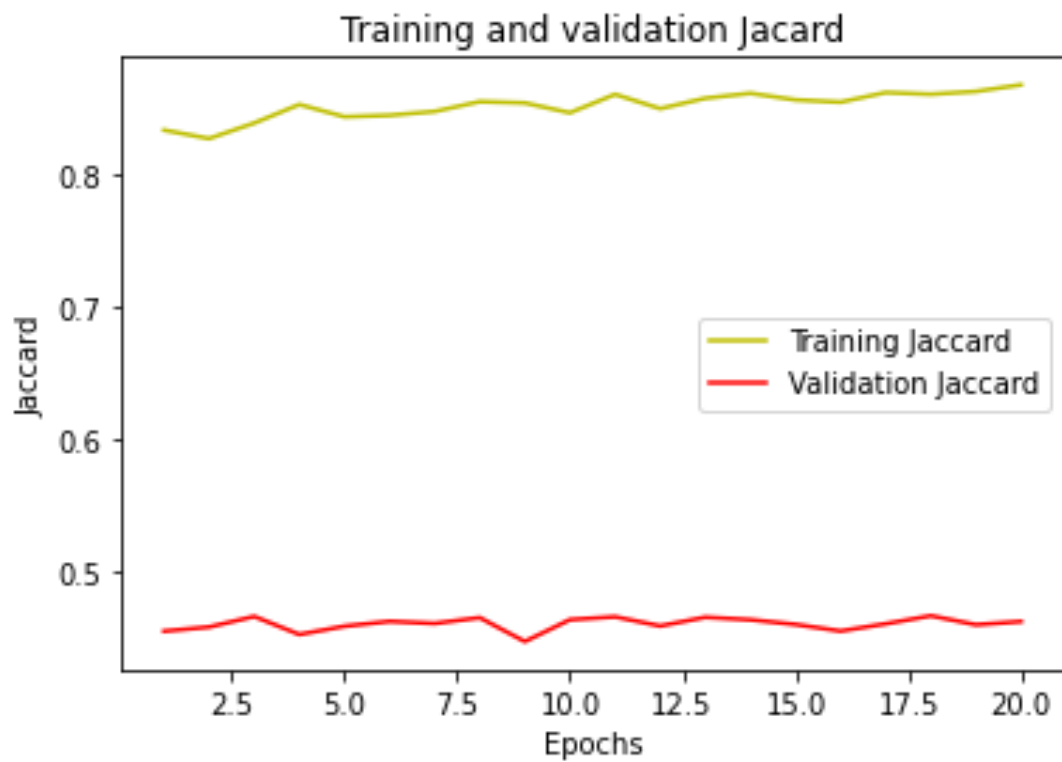


Figure 6: Jaccard accuracy chart

The above figure represents training and validation of Jaccard accuracy. The y-axis depicts the accuracy and the x-axis depicts the number of epochs. Although the validation Jaccard remains relatively the same value, the training Jaccard accuracy only gets higher with the later epochs.

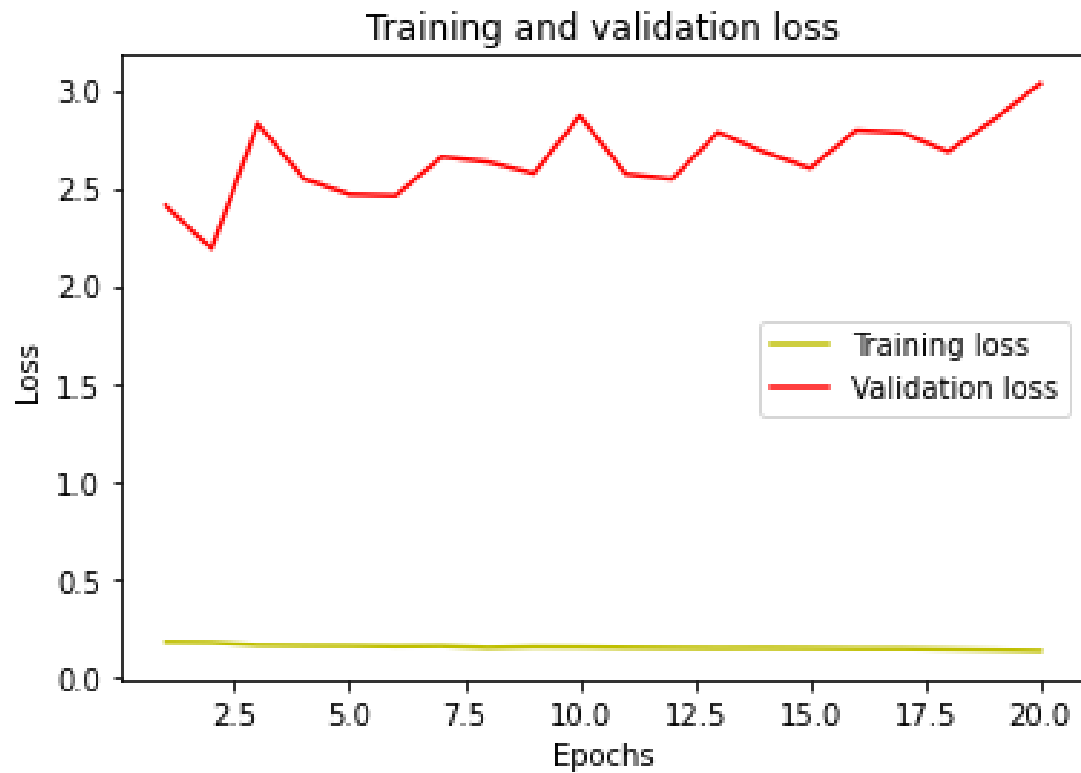


Figure 7:Jaccard loss chart

The above figure represents training and validation loss. The y-axis depicts the loss and the x-axis depicts the number of epochs. The validation loss changes on a small scale but it is apparent that the training loss is only steadily reduced. Similarly to the Jaccard accuracy chart near the 20th epoch is the optimal zone.

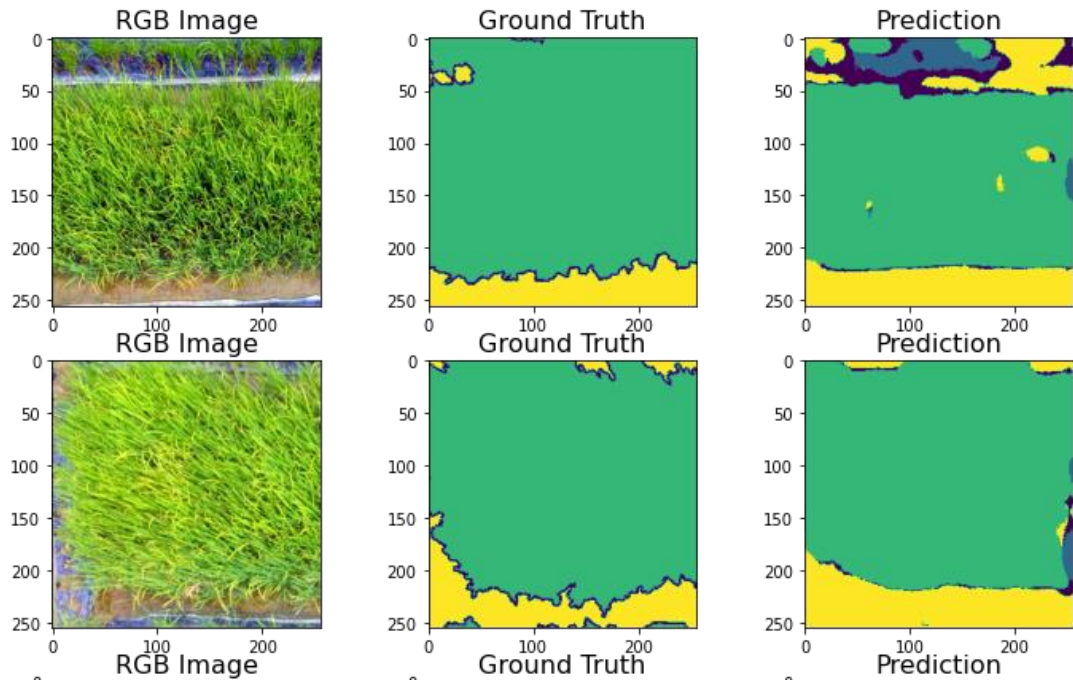


Figure 8: Weed mapping output

6.2 Research Findings

6.2.1 Identification of weeds and providing solutions:

99.49% training accuracy and 85.62% test accuracy were able to be reached using this particular model as shown in the examples above with the predictions for the test data shown below. As for the hyperparameters for this model, 10 epochs, and 32 batch size were chosen as this gives the best accuracies, while the number of classes was 9 due to the 8 weed species and another for negative samples. The learning rate was then chosen to be 0.001 as the learning rate finder function gave that amount as the number with the lowest error rate.

Adam optimizer was also used here when it comes to model compilation due to the previously mentioned reasons with categorical cross entropy as this has multiple classes.

6.2.2 Weed Mapping

Seeing as the training accuracy only gets higher with a large number of epochs it is apparent that nearly 20 epochs would be ideal for this situation.

A training accuracy of 94.88% and a validation accuracy of 64.39% was able to be reached after 20 epochs and 32 batch size. These numbers were chosen after performing hyperparameter tuning in the methodology.

6.3 Discussion

6.3.1 Identification of weeds and providing solutions:

After comparing with the other models such as customized AlexNet and custom CNN models, this ResNet model with the above-mentioned configurations was found to give the best accuracies for the weed identification task.

6.3.2 Weed Mapping:

After comparing with the other models such as customized FCN, this U-Net model with the above-mentioned configurations was found to give the best accuracies for the weed mapping from aerial images task.

6.4 Summary of Each Student's contribution

- Discovering the best model for weed identification from a pool of different CNN models
- Discovering the best configurations for that said model to acquire the best results
- Discovering the best configurations in order to perform weed mapping for the U-Net model
- Creating a mobile application for the created components
- Making the application as user-friendly as possible
- Find an appropriate dataset to train the models

Conclusions

This research paper was performed in order to provide rice farmers with solutions to the four major issues that they are currently facing which include pests, disease, weeds, fertilizers, and growth defects. In this research four CNN models are compared and contrasted in order to identify which one of them is best suited when it comes to rice and paddy farm datasets. Considering the outputs provided by four models which are used for image classification, the resnet50(modal 02) model performed best with it providing 99.43% for training accuracy and 97.04% for validation accuracy. Some additional research has also been done for the purpose of creating approaches for weed mapping, rice plant height measurement, and area calculation of the paddy fields. In order to map the weed u-net, FCN architecture was used and calibrated, with it providing 94.88% training accuracy and 64.39% validation accuracy. Hight measuring and area calculation was implemented

REFERENCE LIST

- [1] S. and K., "Weed Management in Dry Direct-Seeded Rice: A Review on Challenges and Opportunities for Sustainable Rice Production," *Agronomy*, vol. 10, p. 1264, 2020.
- [2] G. and Y., "Assessment of Yield and Economic Losses in Agriculture due to Weeds in India," *Crop Protection*, vol. 107, p. 12–18, 2018.
- [3] R. and I., "Deep Neural Networks to Detect Weeds from Crops in Agricultural Environments in Real-Time: A Review," *Remote Sensing*, vol. 13, p. 4486, 2021.
- [4] S. Hameed and I. Amin, "Detection of Weed and Wheat Using Image Processing," researchgate, 11 2018. [Online]. Available: https://www.researchgate.net/publication/330796498_Detection_of_Weed_and_Wheat_Using_Image_Processing.
- [5] R. Rhushalshafira, "Weed Detection in Rice Fields Using Remote Sensing Technique: A Review," *Applied Sciences*, vol. 11, p. 10701, 2012.
- [6] D. and M., "Weed Classification for Site-Specific Weed Management Using an Automated Stereo Computer-Vision Machine-Learning System in Rice Fields," *Plants*, vol. 9, p. 559, 2020.
- [7] Dalhousie University, "Weeds when are they a problem?," Dalhousie University, [Online]. Available: <https://www.dal.ca/faculty/agriculture/oacc/en-home/resources/pest-management/weed-management/organic-weed-mgmt-resources/weeds-problem.html#:~:text=Weeds%20use%20the%20same%20nutrients,might%20have%20gone%20to%20crops.&text=Weeds%20that%20compete%20aggre>.
- [8] Integrated Pest Management, "Identifying weeds in field crops," Integrated Pest Management, 2022. [Online]. Available: https://www.canr.msu.edu/ipm/agriculture/field_crops/identifying_weeds_in_field_crops.
- [9] Npic, "Weed Control and Herbicides," Npic, [Online]. Available: <http://npic.orst.edu/pest/weeds.html#:~:text=Make%20sure%20the%20label%20clearly,almost%20any%20type%20of%20plant>.
- [10] Ipm, "Weed Management in Landscapes Management Guidelines--UC IPM," Ipm, [Online]. Available: <http://ipm.ucanr.edu/PMG/PESTNOTES/pn7441.html>.

- [11] France, "Sri Lanka lifts ban on weedkiller linked to cancer cases," France, 2022. [Online]. Available: <https://www.france24.com/en/live-news/20211124-sri-lanka-lifts-ban-on-weedkiller-linked-to-cancer-cases>.
- [12] Play.google.com. 2022. [online] Available at:
<<https://play.google.com/store/apps/details?id=com.extension.idweeds&hl=en&gl=US>> [Accessed 11 February 2022].

Glossary

CNN – Convolutional Neural Network

FCN — Fully Convolutional Network

SDLC - Software development life cycle

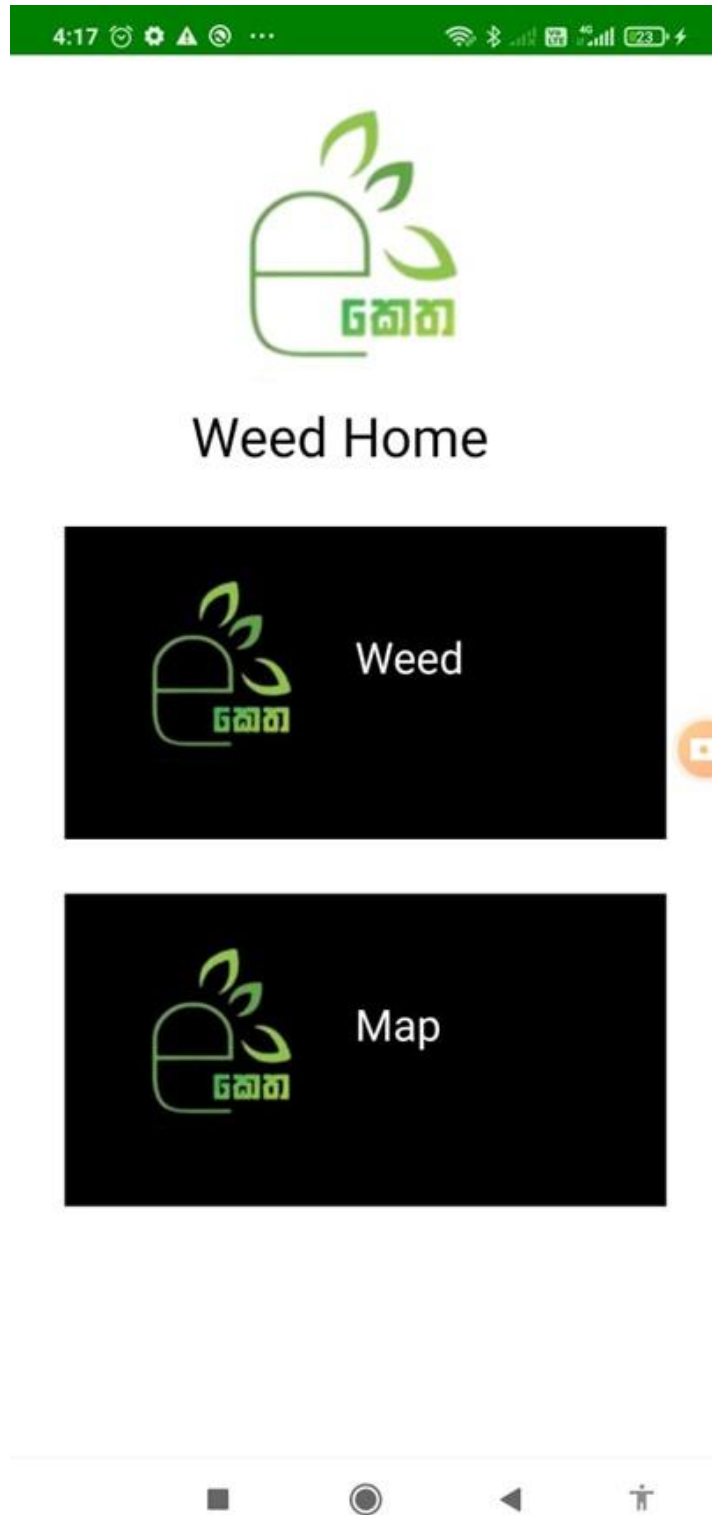
GPS - Global Positioning System

UI – User interface

UX – User experience

Appendices

Screenshots of the Application





Snake weed

destock paddocks where snakeweed is a problem promote pasture growth; native pasture is usually not competitive enough once snakeweed has established itself; improved pasture grasses may have to be sown

Take Picture

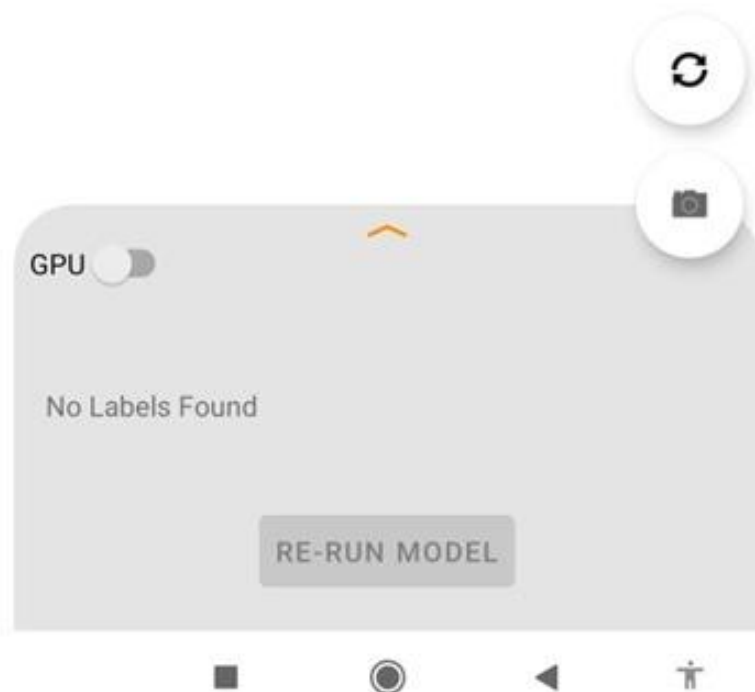
Launch Gallery

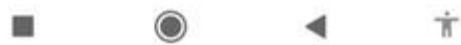


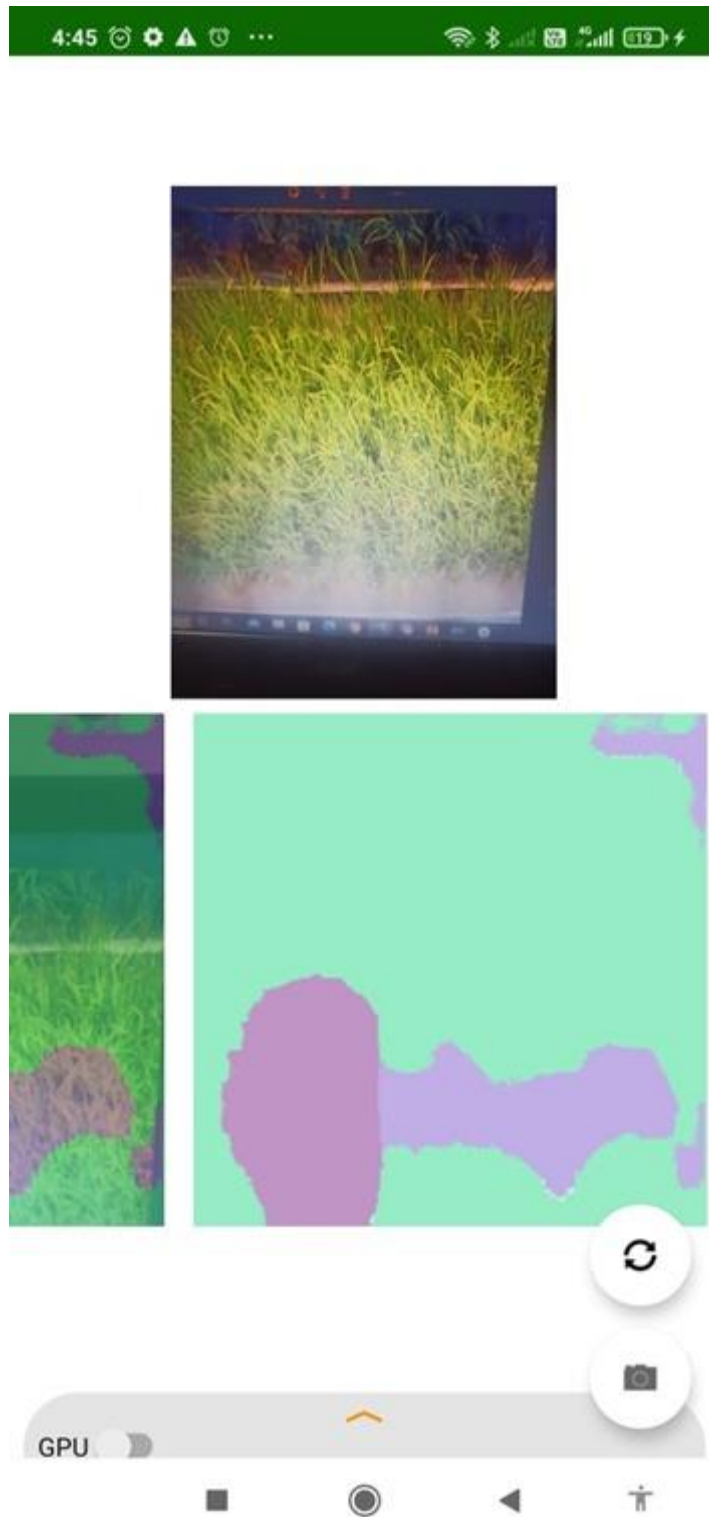
Negative
Not Recognized

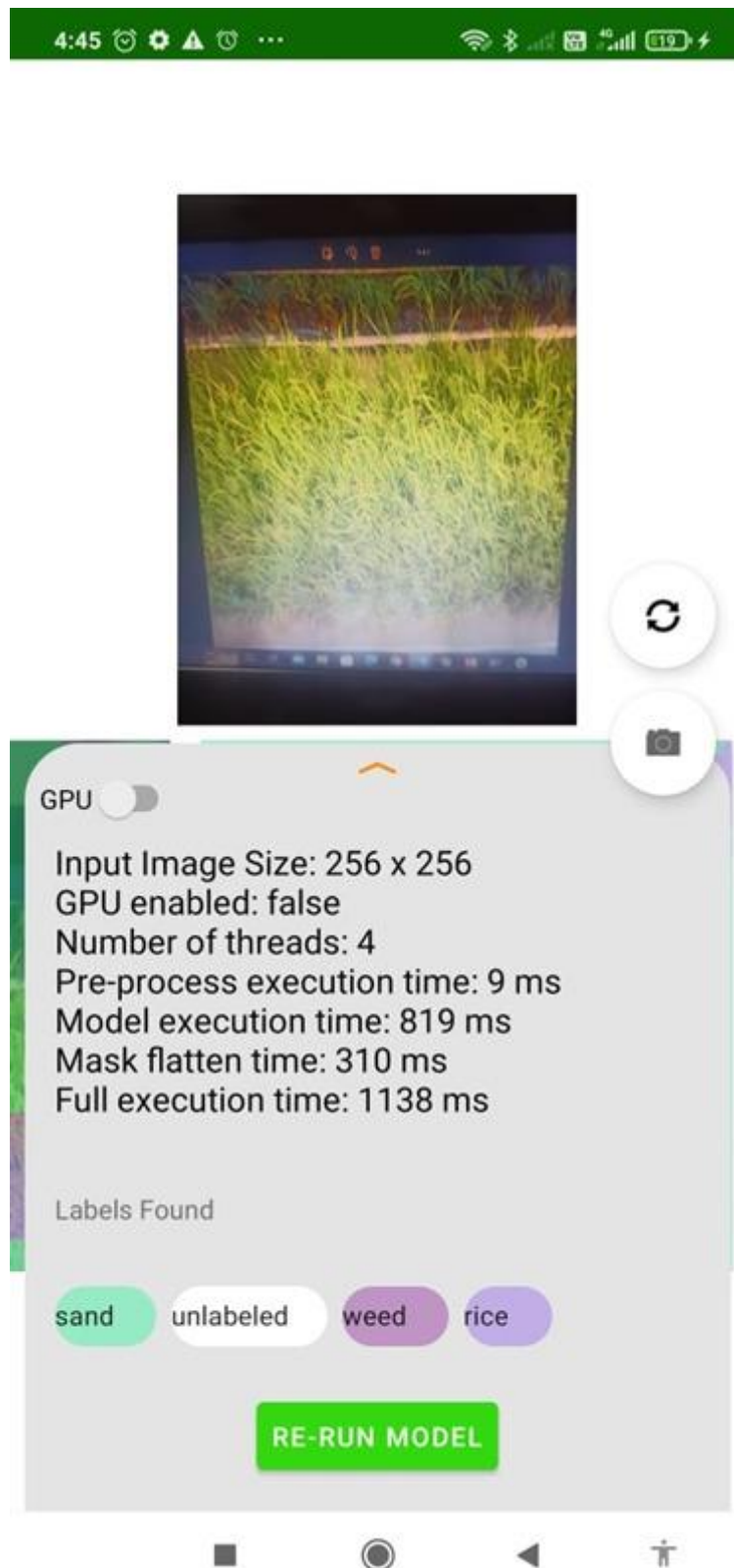
Take Picture

Launch Gallery









Training weed classification model


```
Epoch: 01 | Epoch Time: 8m 33s
  Train Loss: 0.828 | Train Acc @1: 73.24% | Train Acc @5: 100.00%
  Valid Loss: 0.405 | Valid Acc @1: 86.45% | Valid Acc @5: 100.00%
Epoch: 02 | Epoch Time: 8m 34s
  Train Loss: 0.354 | Train Acc @1: 87.96% | Train Acc @5: 100.00%
  Valid Loss: 0.343 | Valid Acc @1: 88.50% | Valid Acc @5: 100.00%
Epoch: 03 | Epoch Time: 8m 32s
  Train Loss: 0.308 | Train Acc @1: 89.94% | Train Acc @5: 100.00%
  Valid Loss: 0.417 | Valid Acc @1: 87.98% | Valid Acc @5: 100.00%
Epoch: 04 | Epoch Time: 8m 32s
  Train Loss: 0.231 | Train Acc @1: 92.47% | Train Acc @5: 100.00%
  Valid Loss: 0.225 | Valid Acc @1: 92.50% | Valid Acc @5: 100.00%
Epoch: 05 | Epoch Time: 8m 32s
  Train Loss: 0.182 | Train Acc @1: 93.92% | Train Acc @5: 100.00%
  Valid Loss: 0.350 | Valid Acc @1: 91.31% | Valid Acc @5: 100.00%
Epoch: 06 | Epoch Time: 8m 35s
  Train Loss: 0.122 | Train Acc @1: 96.04% | Train Acc @5: 100.00%
  Valid Loss: 0.166 | Valid Acc @1: 95.03% | Valid Acc @5: 100.00%
Epoch: 07 | Epoch Time: 8m 37s
  Train Loss: 0.064 | Train Acc @1: 97.86% | Train Acc @5: 100.00%
  Valid Loss: 0.119 | Valid Acc @1: 96.30% | Valid Acc @5: 100.00%
Epoch: 08 | Epoch Time: 8m 32s
  Train Loss: 0.042 | Train Acc @1: 98.53% | Train Acc @5: 100.00%
  Valid Loss: 0.095 | Valid Acc @1: 96.73% | Valid Acc @5: 100.00%
Epoch: 09 | Epoch Time: 8m 31s
  Train Loss: 0.021 | Train Acc @1: 99.35% | Train Acc @5: 100.00%
  Valid Loss: 0.082 | Valid Acc @1: 97.16% | Valid Acc @5: 100.00%
Epoch: 10 | Epoch Time: 8m 32s
  Train Loss: 0.019 | Train Acc @1: 99.49% | Train Acc @5: 100.00%
  Valid Loss: 0.088 | Valid Acc @1: 97.09% | Valid Acc @5: 100.00%
```

```
[ ] model.load_state_dict(torch.load('tut5-model.pt'))

test_loss, test_acc_1, test_acc_5 = evaluate(model, test_iterator, criterion, device)

print(f'Test Loss: {test_loss:.3f} | Test Acc @1: {test_acc_1*100:6.2f}% | ' \
      f'Test Acc @5: {test_acc_5*100:6.2f}%')

Test Loss: 0.698 | Test Acc @1: 85.62% | Test Acc @5: 100.00%
```

```
 plot_confusion_matrix(labels, pred_labels, classes)
```

```

@RequiresApi(api = Build.VERSION_CODES.N)
public void classifyImage(Bitmap image){
    try {
        ModelRes model = ModelRes.newInstance(getApplicationContext());

        // Creates inputs for reference.
        TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 224, 224, 3}, DataType.UINT8);
        ByteBuffer byteBuffer = ByteBuffer.allocateDirect(1 * imageSize * imageSize * 3);
        byteBuffer.order(ByteOrder.nativeOrder());

        int[] intValues = new int[imageSize * imageSize];
        image.getPixels(intValues, 0, image.getWidth(), 0, 0, image.getWidth(), image.getHeight());
        int pixel = 0;
        //iterate over each pixel and extract R, G, and B values. Add those values individually to the byte buffer
        for(int i = 0; i < imageSize; i++){
            for(int j = 0; j < imageSize; j++){
                int val = intValues[pixel++]; // RGB
                byteBuffer.put((byte) (((val >> 16) & 0xFF) * (1.f / 1)));
                byteBuffer.put((byte) (((val >> 8) & 0xFF) * (1.f / 1)));
                byteBuffer.put((byte) ((val & 0xFF) * (1.f / 1)));
            }
        }

        inputFeature0.loadBuffer(byteBuffer);
    }
}

```

```

private fun setChipsToLogView(itemsFound: Map<String, Int>) {
    chipsGroup.removeAllViews()

    val paddingDp = TypedValue.applyDimension(
        TypedValue.COMPLEX_UNIT_DIP, value: 10F,
        resources.displayMetrics
    ).toInt()

    for ((label, color) in itemsFound) {
        val chip = Chip(context: this)
        chip.text = label
        chip.chipBackgroundColor = getColorStateListForChip(color)
        chip.isClickable = false
        chip.setPadding(left: 0, paddingDp, right: 0, paddingDp)
        chipsGroup.addView(chip)
    }

    val labelsFoundTextView: TextView = findViewById(R.id.tfe_is_labels_found)
    if (chipsGroup.childCount == 0) {
        labelsFoundTextView.text = "No Labels Found"
    } else {
        labelsFoundTextView.text = "Labels Found"
    }

    chipsGroup.parent.requestLayout()
}

```

```

private fun createModelExecutor(useGPU: Boolean) {
    if (imageSegmentationModel != null) {
        imageSegmentationModel!!.close()
        imageSegmentationModel = null
    }
    try {
        imageSegmentationModel = ImageSegmentationModelExecutor(context = this, useGPU)
    } catch (e: Exception) {
        Log.e(TAG, msg: "Failed: ${e.message}")
        val logText: TextView = findViewById(R.id.log_view)
        logText.text = e.message
    }
}

private fun animateCameraButton() {
    val animation = AnimationUtils.loadAnimation(context = this, R.anim.scale_anim)
    animation.interpolator = BounceInterpolator()
    captureButton.animation = animation
    captureButton.animation.start()
}

```

```

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    if(resultCode == RESULT_OK){
        if(requestCode == 3){
            Bitmap image = (Bitmap) data.getExtras().get("data");
            int dimension = Math.min(image.getWidth(), image.getHeight());
            image = ThumbnailUtils.extractThumbnail(image, dimension, dimension);
            imageView.setImageBitmap(image);

            image = Bitmap.createScaledBitmap(image, imageSize, imageSize, false);
            classifyImage(image);
        }else{
            Uri dat = data.getData();
            Bitmap image = null;
            try {
                image = MediaStore.Images.Media.getBitmap(this.getContentResolver(), dat);
            } catch (IOException e) {
                e.printStackTrace();
            }
            imageView.setImageBitmap(image);

            image = Bitmap.createScaledBitmap(image, imageSize, imageSize, false);
            classifyImage(image);
        }
    }
}

```