A SMART PLANT DISORDER IDENTIFICATION SYSTEM

CropMedic 2.0

Research Problem

1. **39.4%** — Agriculture’s contribution to global GDP [1]
2. **43%** — Agriculture’s contribution to global export market [1]
3. **42%** — Agriculture production loss due to plant diseases [2]
4. ☑ — Leading cause of plant diseases is nutrition deficiency [3]
5. 🏆 — Feasible smart solution for early detection of nutrition deficiencies recommending accurate chemical remedies

COMPONENT ONE

Identification of nutrient deficiency in plants

IT17110808 – M.Sukanya

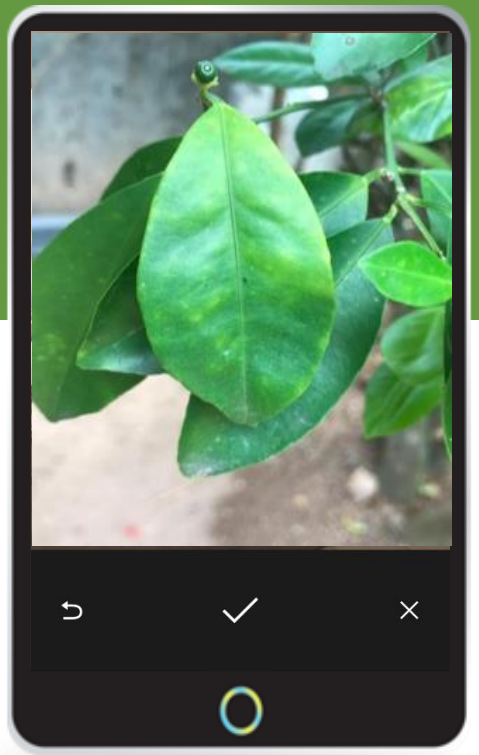
Research Problem

How to distinctly identify the presence of nutrient deficiency in a particular plant?

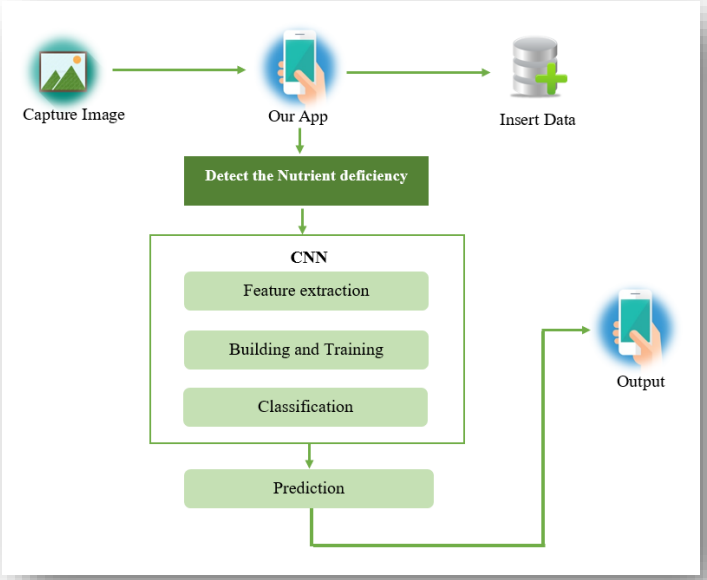
- Many Nutrient deficiencies have similar symptoms

FINAL SOLUTION

A mobile application that captures the image and identify the nutrient deficiency in plants with more accuracy



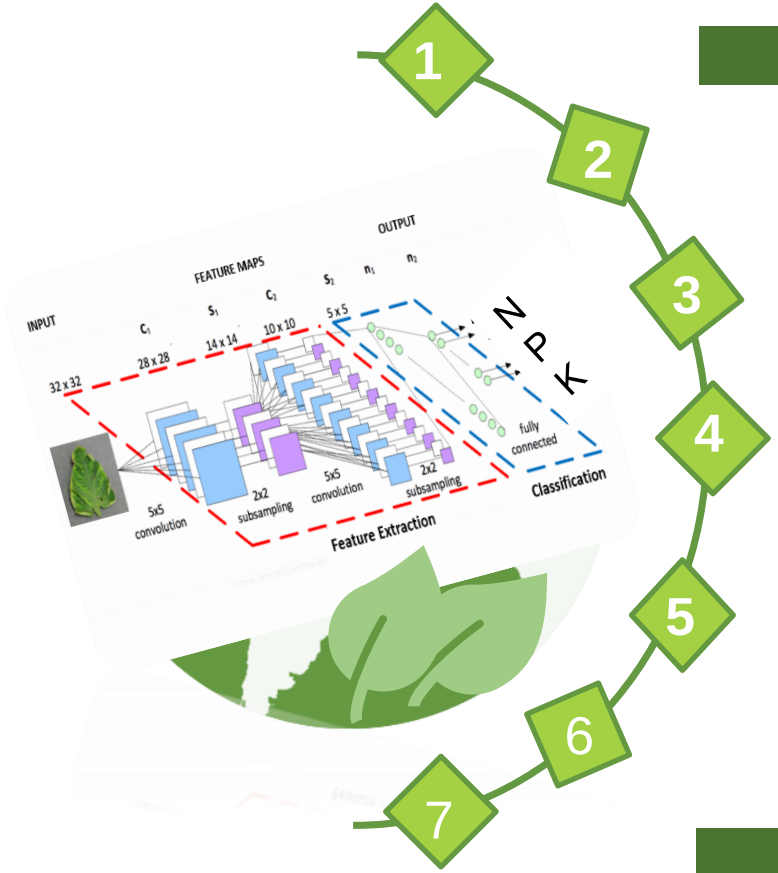
Solution implementation



01 Nutrient Deficiency Prediction Using the CNN Model.

02 Develop interface for the mobile application using Flutter

Nutrient deficiency prediction using CNN



Data Collection

Image Preprocessing

Build CNN Model

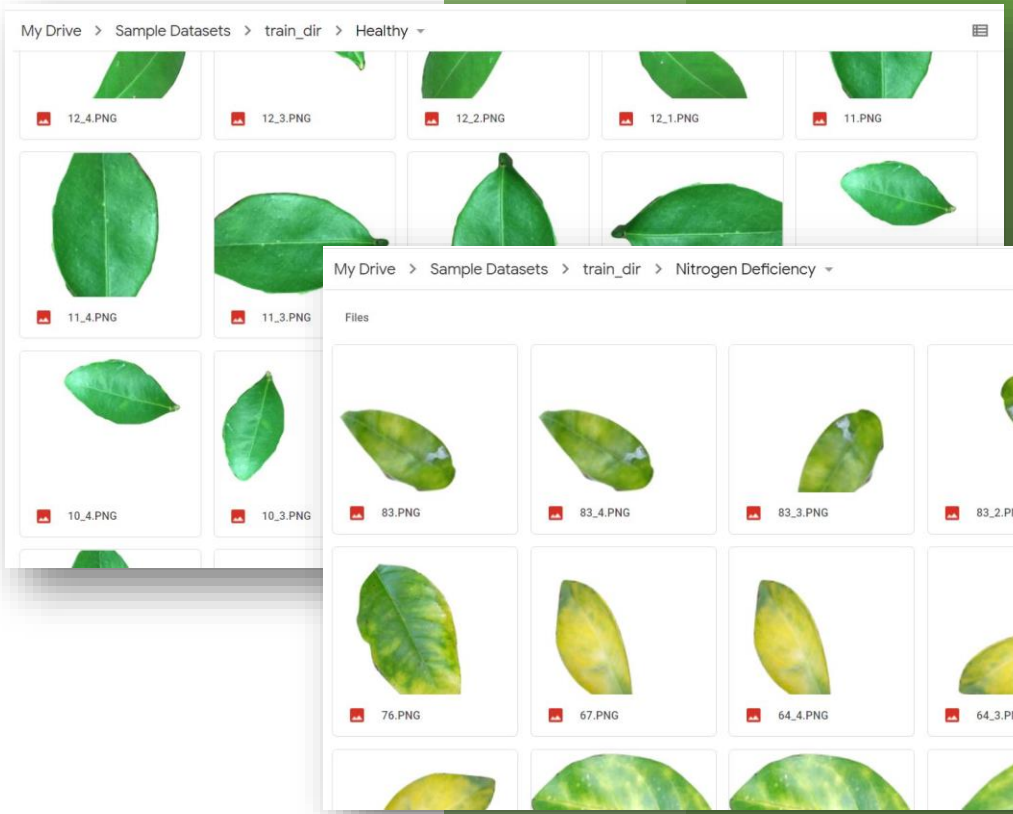
Visualization of images after every layer

Training CNN with Parameters

Save the Model and Model Weights

Prediction

Data Collection



268 Images of Citrus Plant

02 Classes

- Healthy
- Nitrogen Deficiency

Image preprocessing

```
# Preprocessing of Image.
print('Image Preprocessing')
training_datagenerator=ImageDataGenerator(
                rescale=1./255,
                zoom_range=0.2,
                shear_range=0.2,
                validation_split=0.2,
                horizontal_flip=True
                )
testing_datagenerator=ImageDataGenerator(rescale=1./255)
```

```
Image Preprocessing
```

Data Standardization

Rescale the Image

Horizontal Flip Image

Zooming and Shear Transformation of Image

Data Augmentation

Resize Image

```
# setting width, height and color for the input image.
image_width,image_height =256,256
inputImageShape=(image_width,image_height,3)
batch_size =32

trainGenerator =training_datagenerator.flow_from_directory(training_directory,
                                                           target_size=(image_width,image_height),
                                                           batch_size=batch_size)
testGenerator=testing_datagenerator.flow_from_directory(testing_directory,shuffle=True,
                                                        target_size=(image_width,image_height),
                                                        batch_size=batch_size)
```

```
Found 268 images belonging to 2 classes.
Found 28 images belonging to 2 classes.
```

Building CNN Model

32 Features Extract from the Input Image

ReLU Rectified Linear activation to fire up neurons

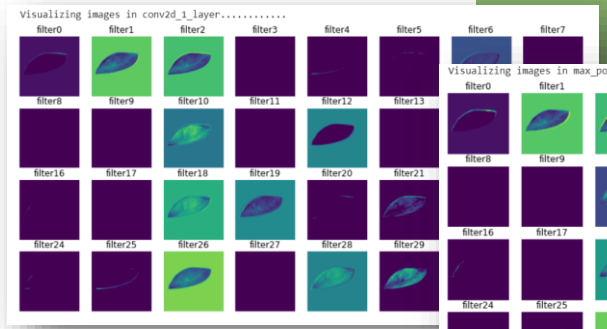
```
# Building CNN model.
model = Sequential()
model.add(Conv2D(32, (5, 5),input_shape=inputImageShape,activation='relu'))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Conv2D(32, (3, 3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(128,activation='relu'))
model.add(Dense(no_of_classes,activation='softmax'))
model.summary()
```

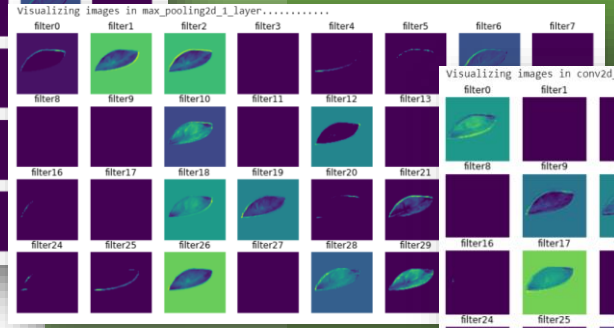
Flatten layer 4D array to 1D array

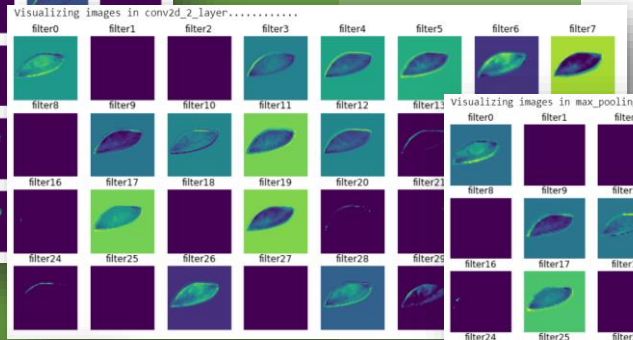
Softmax activation for probability results

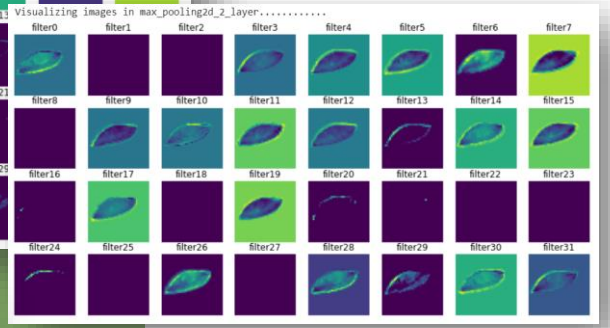
Dropout makes some neurons to relax

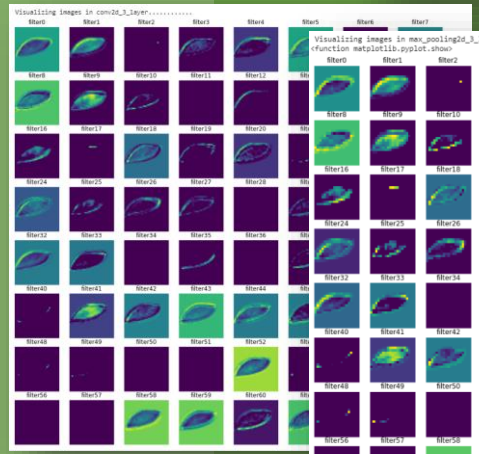
Visualization of Images in every layer

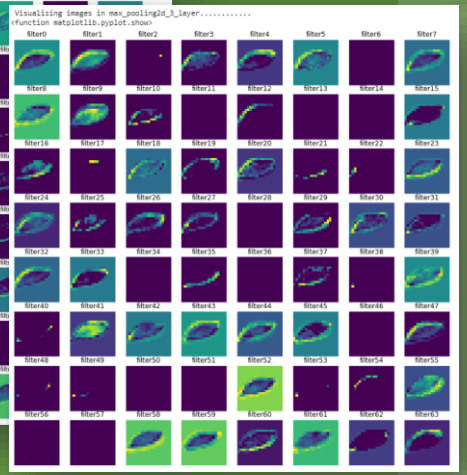






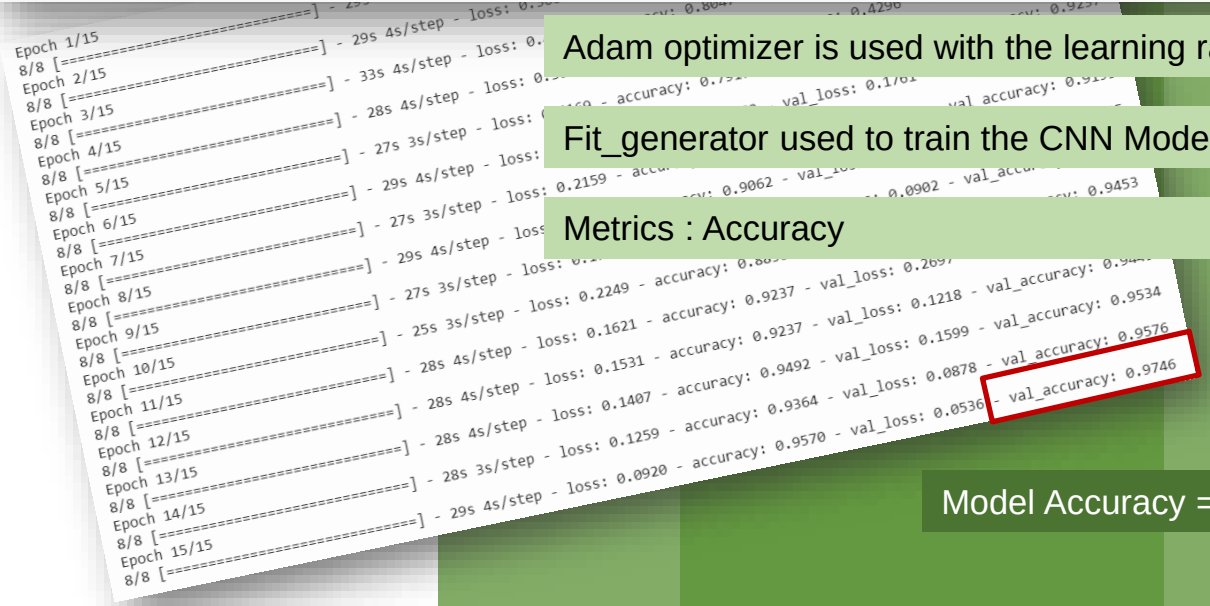






Training CNN with parameter

```
# CNN Building and Training.
optimizer=keras.optimizers.Adam(lr=0.001)
model.compile(optimizer=optimizer ,loss= 'categorical_crossentropy' ,metrics=['accuracy'])
training=model.fit_generator(trainGenerator,nb_epoch=15,
                             steps_per_epoch=trainGenerator.samples // batch_size,
                             validation_data=validationGenerator,
                             nb_val_samples=validationGenerator.samples// batch_size,verbose=1)
```



Adam optimizer is used with the learning rate = 0.001

Fit_generator used to train the CNN Model.

Metrics : Accuracy

Model Accuracy = 0.9746

My Drive - Google Drive | Nitrogen_Deficiency_Prediction_v | Success code=4/1wGe8wyKvN1I

colab.research.google.com/drive/1hVzdT7JjgsY3li_B2GRsPOhtRkc641Vt#scrollTo=Twq1a9Wmr1oY

Apps | Research papers | SLIIT | Virtusa | CNN | CTSE | Work | ML | Research | AR | Intern | Exam | ML Exam | Bill Payments | Methodology

Nitrogen_Deficiency_Prediction_v16.ipynb

File Edit View Insert Runtime Tools Help

Comment | Share

+ Code + Text

RAM Disk | Editing

[44]
```
def prepare(image_path):
    img = image.load_img(image_path, target_size=(256, 256))
    x = image.img_to_array(img)
    x = x/255
    return np.expand_dims(x, axis=0)

output = model.predict_classes([prepare('/content/drive/My Drive/Sample Datasets/test_dir/Healthy/01.jpeg')])
diseases=image.load_img('/content/drive/My Drive/Sample Datasets/test_dir/Healthy/01.jpeg')
plt.imshow(diseases)
print(Classes[int(output)])
```

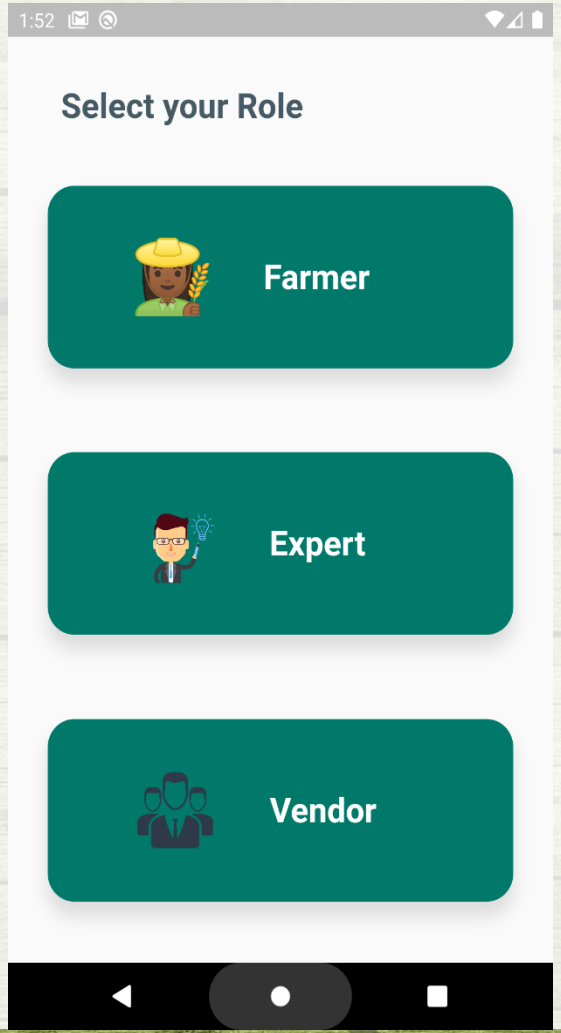
```
Healthy
```


```
output = model.predict_classes([prepare('/content/drive/My Drive/Sample Datasets/test_dir/NitrogenDeficiency/1.jpeg')])
diseases=image.load_img('/content/drive/My Drive/Sample Datasets/test_dir/NitrogenDeficiency/1.jpeg')
plt.imshow(diseases)
print(Classes[int(output)])
```

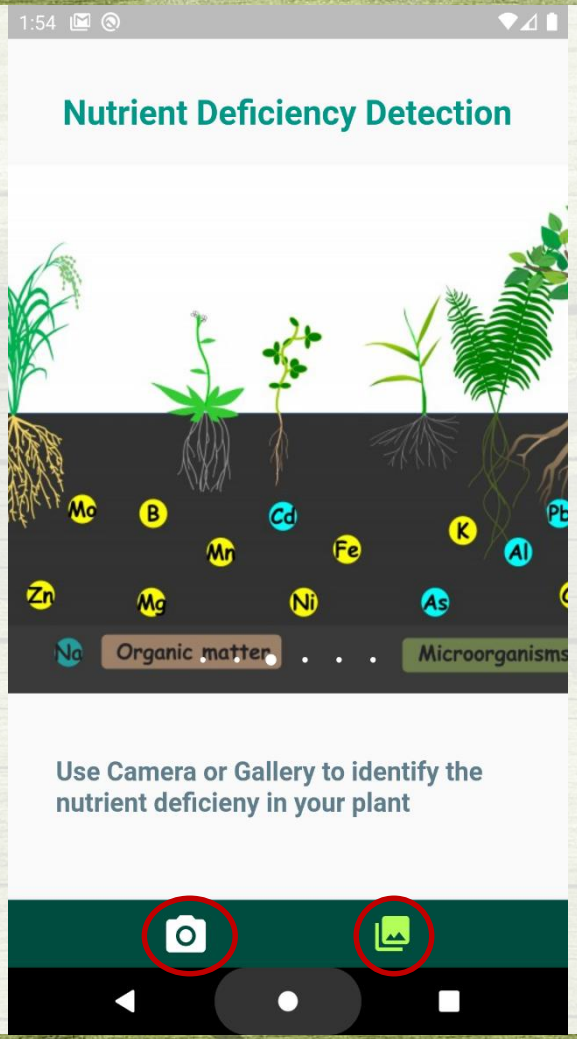
Type here to search

11:14 AM

Mobile App Interfaces

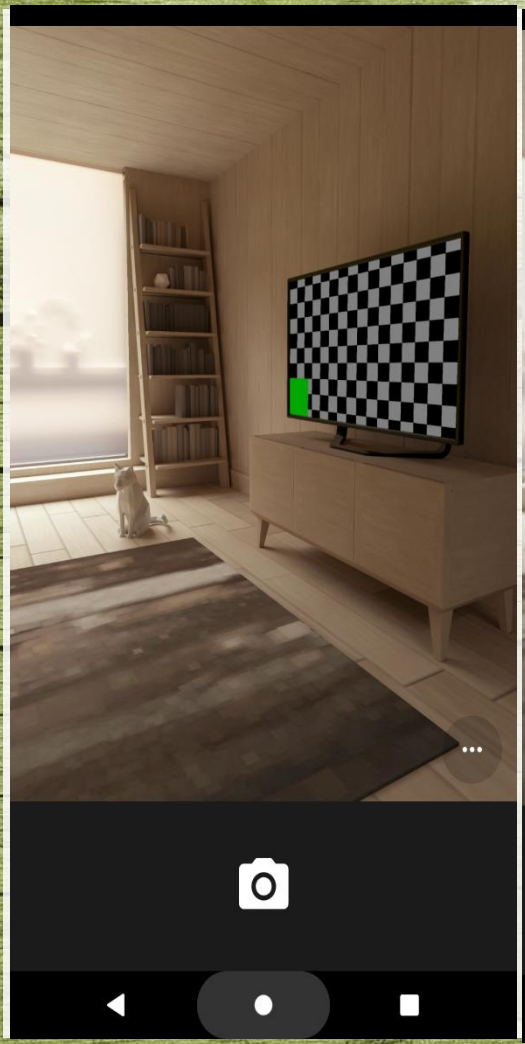


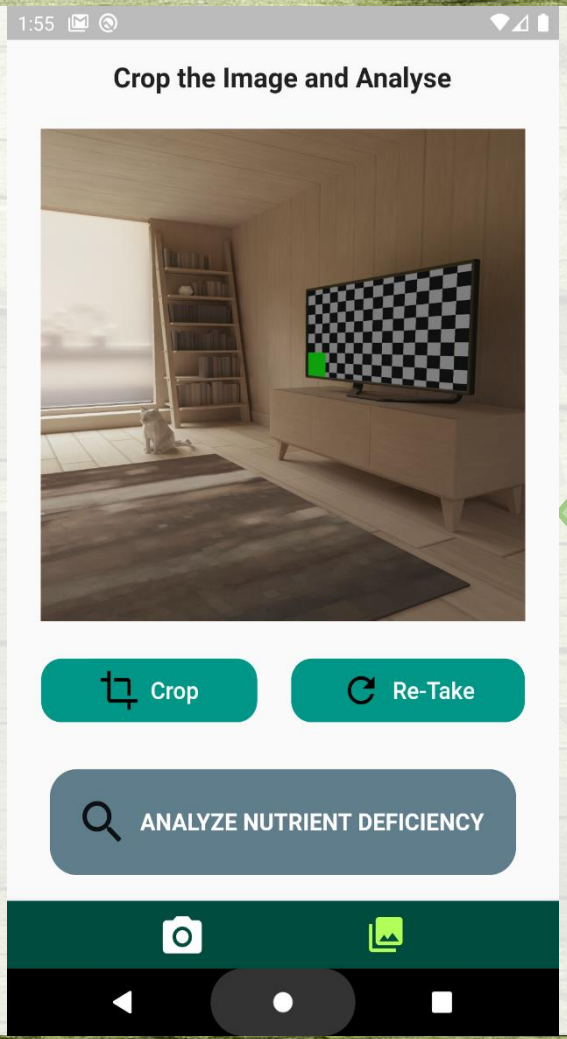
User can select their role



Ability to capture image of crop leaves

Ability to upload images from the gallery

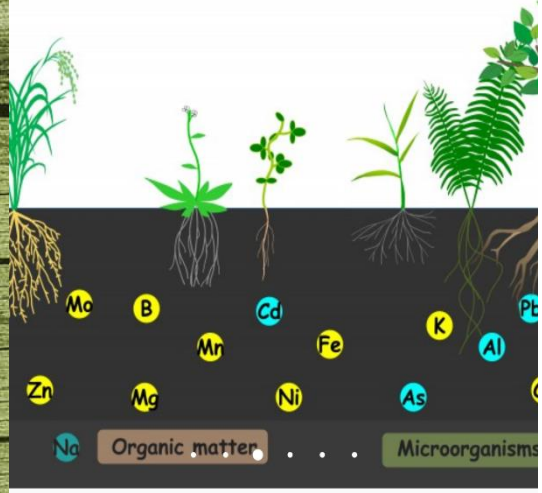




Capture Images

1:54

Nutrient Deficiency Detection

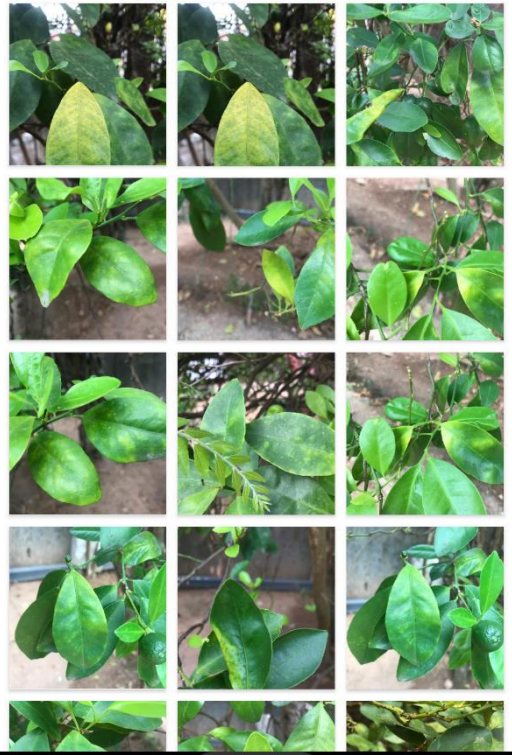


Use Camera or Gallery to identify the nutrient deficieny in your plant

1:56

NitrogenDeficiency

IMAGES FROM DRIVE / SUKANYA02MANOHARAN@GMAIL.C...



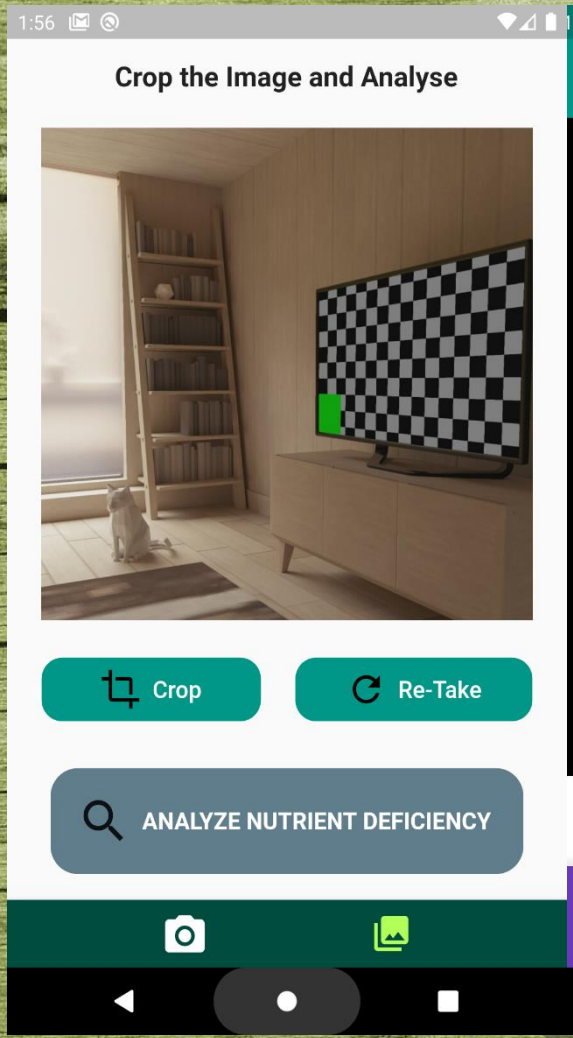
1:57

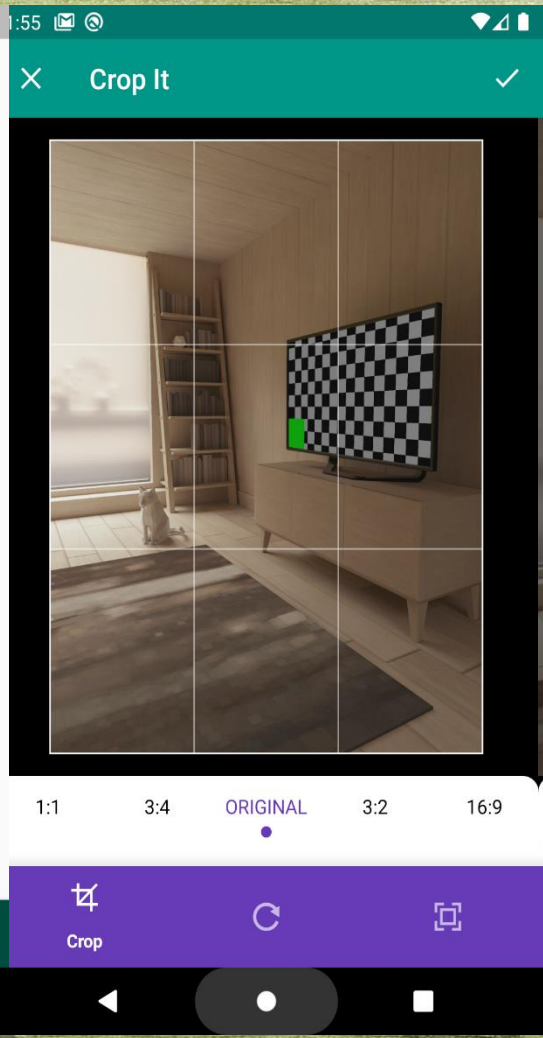
Crop the Image and Analyse

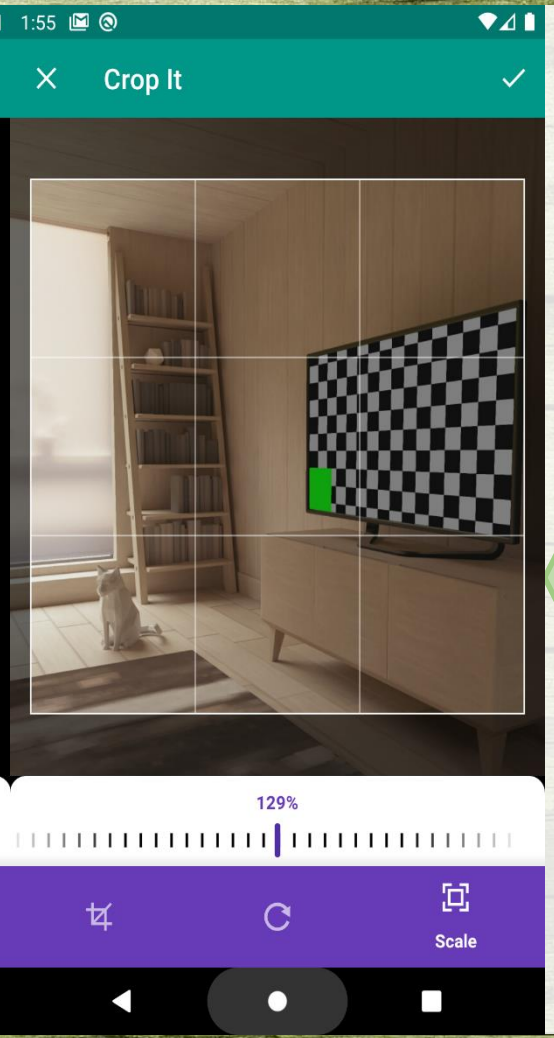
Crop | Re-Take

ANALYZE NUTRIENT DEFICIENCY

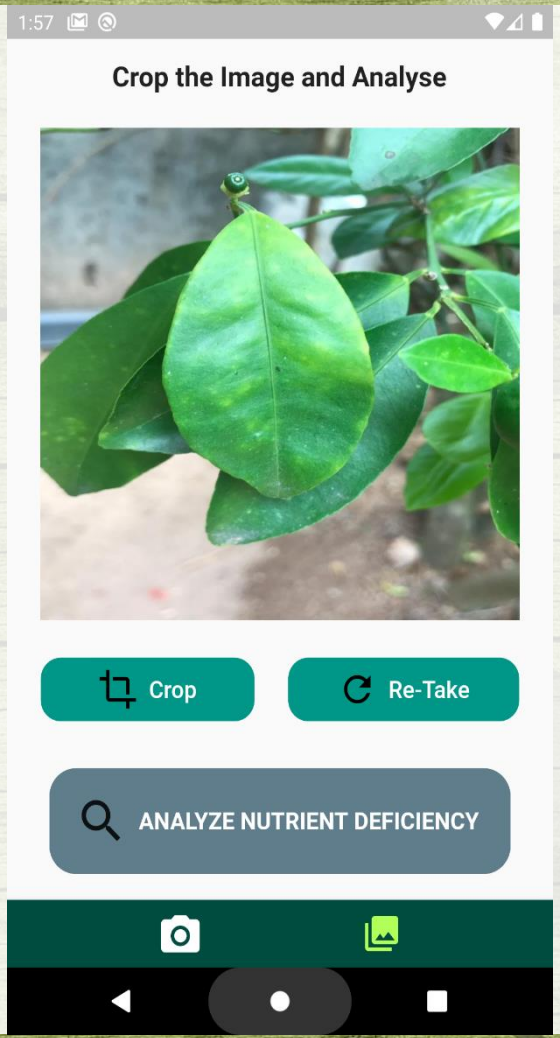
Select From gallery







Able to Crop the selected Image



Analyze Nutrient deficiency

Next steps

01 Collect the datasets for NPK in cash crops and do prediction using this model

02 Connect the trained model with our mobile app – Backend Development

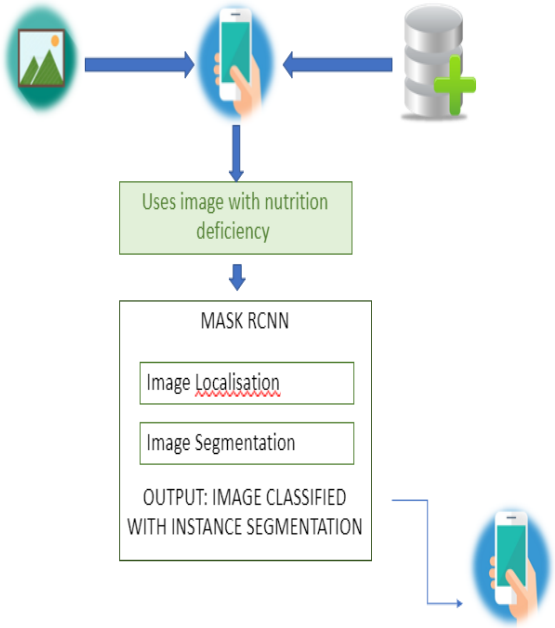
03 Connect the mobile app with the CloudStore and store the captured/ uploaded images into the Cloud.

04 Integrating the System

COMPONENT TWO

Detection of Degree of Nutrient Deficiency and Recommendation of Solution

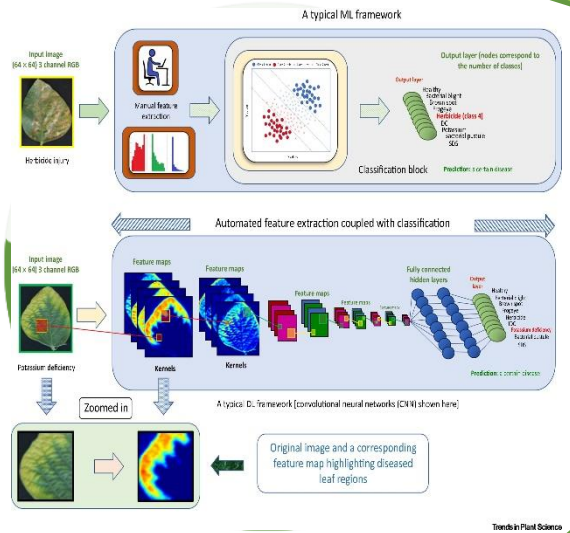
SOLUTION IMPLEMENTATION



01 Identify extent of nutrition deficiency by implementing a Mask RCNN

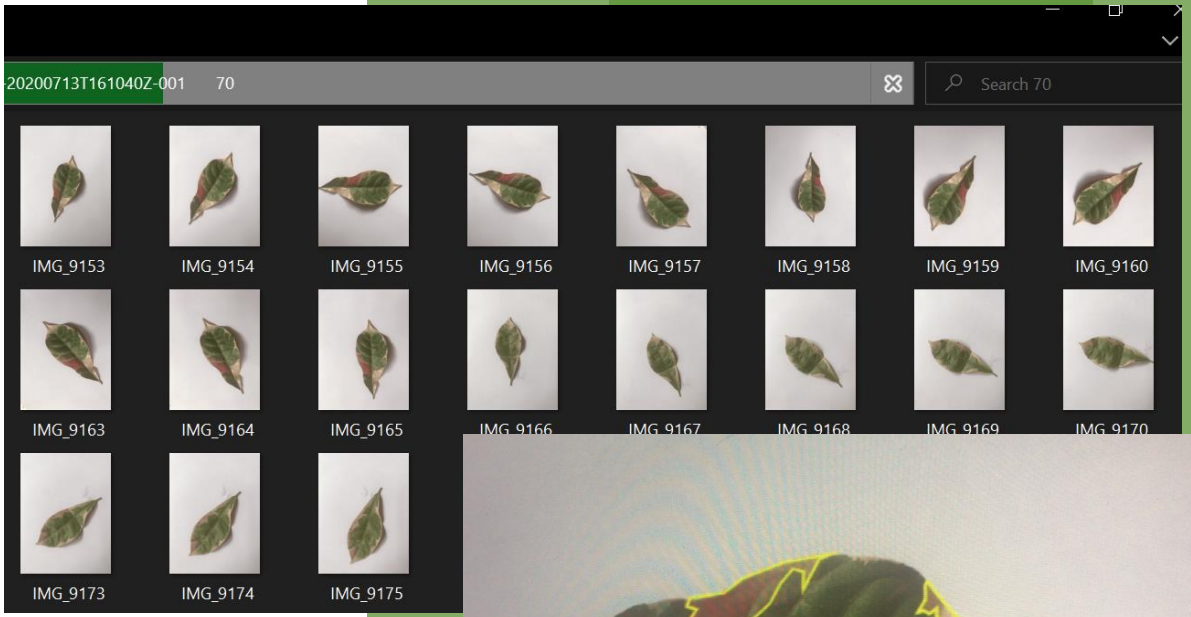
02 tegrate it to the main application and other components

DEGREE OF NUTRIENT DEFICIENCY PREDICTION -WORK FLOW



1. Data Collection √
2. Image annotation √
3. Transfer Learning √
4. Deploying the Mask RCNN Model √
5. Training the Model with annotated dataset √
6. Provide image classified output in a numeric form for better user-ability
7. Integrating model with main application and the rest of the components

DATA COLLECTION AND ANNOTATION



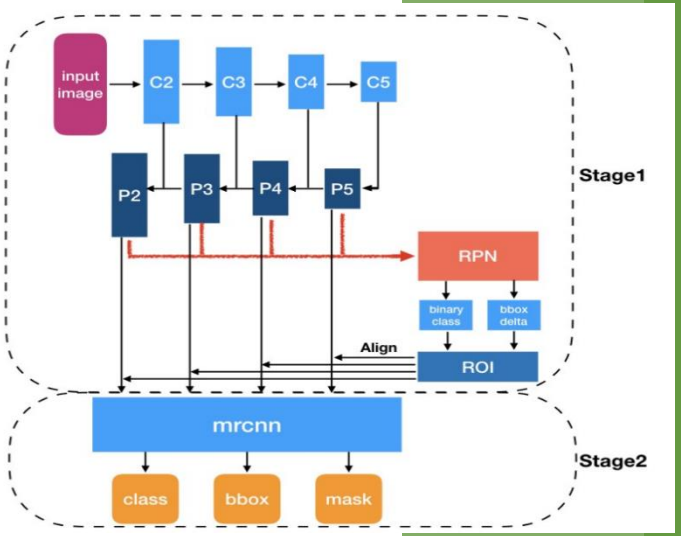
300

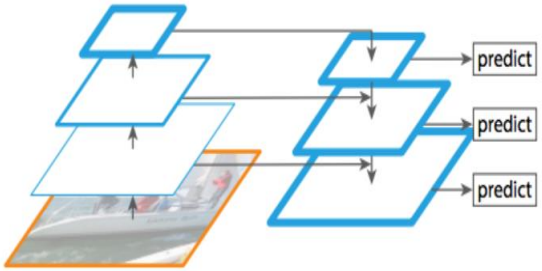
Images of green-white leaf categorisd

03 Classes

<30%, 30%-70%, >70%

TRANSFER LEARNING AND DEPLOYING MODEL





ARCHITECTURE

- ResNet101
- Feature Pyramid Network
- COCO Weights

Components - Functional

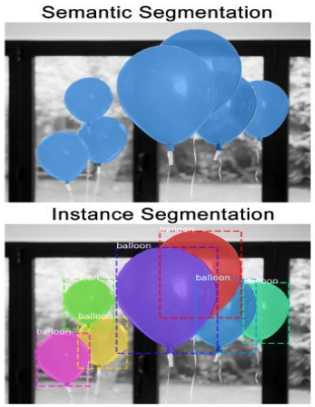
- Region Proposal Network
- Region of Interest Classifier

TRANSFER LEARNING AND DEPLOYING MODEL

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ResNet structure

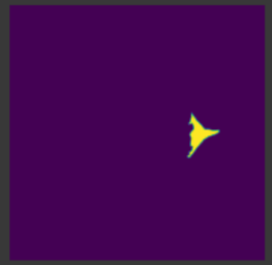
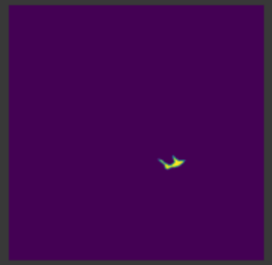
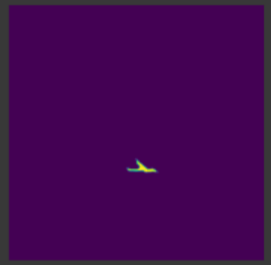
Final Outcome:



Instance segmentation with bounding boxes

RESULTS AND CODE SEGMENTS

```
image                    shape: (1024, 1024, 3)       min:    0.00000  max:  243.00000  uint8
image_meta               shape: (14,)                 min:    0.00000  max: 4032.00000  float64
class_ids                shape: (5,)                  min:    1.00000  max:    1.00000  int32
bbox                     shape: (5, 4)                min:  146.00000  max:  842.00000  int32
mask                     shape: (1024, 1024, 5)       min:    0.00000  max:    1.00000  bool
```

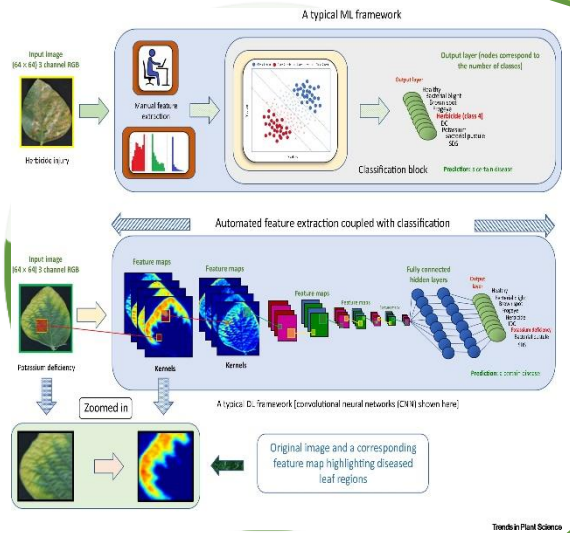
RESULTS AND CODE SEGMENTS

```
[54] #Bounding Boxes
     # Load random image and mask.
     image_id = random.choice(dataset.image_ids)
     image = dataset.load_image(image_id)
     mask, class_ids = dataset.load_mask(image_id)
     # Compute Bounding box
     bbox = utils.extract_bboxes(mask)

     # Display image and additional stats
     print("image_id ", image_id, dataset.image_reference(image_id))
     log("image", image)
     log("mask", mask)
     log("class_ids", class_ids)
     log("bbox", bbox)
     # Display image and instances
     visualize.display_instances(image, bbox, mask, class_ids, dataset.class_names)
```

```
image_id  0 /content/Mask_RCNN/datasets/leaves/train/IMG_9152.jpg
image                    shape: (4032, 3024, 3)       min:   27.00000  max:  255.00000  uint8
mask                     shape: (4032, 3024, 5)       min:    0.00000  max:    1.00000  bool
class_ids                shape: (5,)                  min:    1.00000  max:    1.00000  int32
```

DEGREE OF NUTRIENT DEFICIENCY PREDICTION – NEXT

1. Data Collection √
2. Image annotation √
3. Transfer Learning √
4. Deploying the Mask RCNN Model √
5. Training the Model with annotated dataset √
6. Provide image classified output in a numeric form for better user-ability
7. Integrating model with main application and the rest of the components



COMPONENT THREE

Develop an IOT device to measure key nutrient values of soil

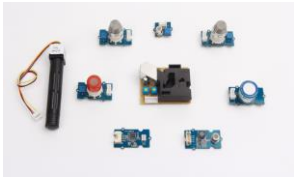
K.T. Ramasinghe
IT15070418

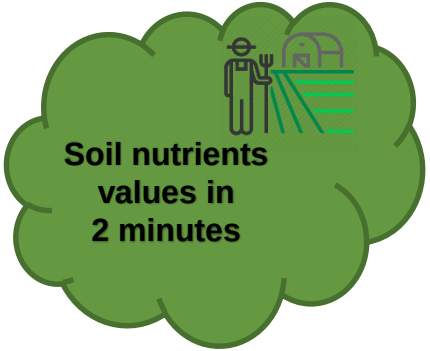
Modifications of The component?

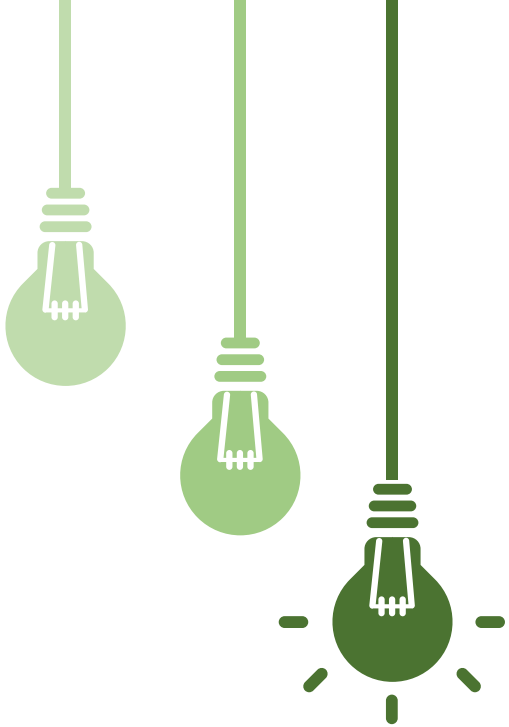
Developing IOT device to calculate soil nutrients level (N,P,K) by using colorimetric principle

Developing IOT device to calculate pH, Humidity, Electrical Conductivity, temperature and predict N,P,K values by a machine learning model.

Solution implementation







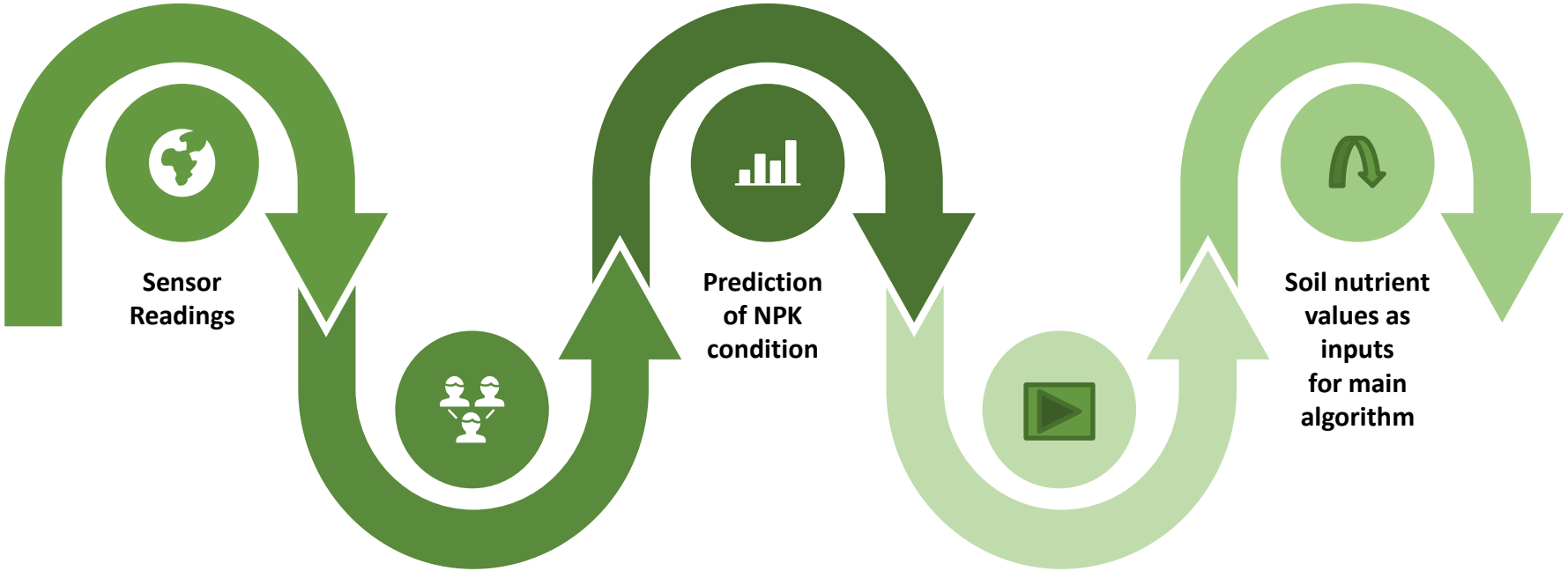
01 Machine learning model to predict NPK level of soil with usage of pH, Humidity, Electrical Conductivity and Temperature.

02 Develop an IOT device to calculate pH, Humidity, Electrical Conductivity and Temperature Selected Soil Sample.

Analysis Soil Condition using an IoT Device

1. Study about plant disorders
2. Soil analysis
3. Data Collection & Preprocessing
4. Building the linear regression model
5. Studying about IOT
6. Developing the IOT device
7. Developing firebase connection and test model

Workflow…


Sensor Readings
Prediction of NPK condition
Soil nutrient values as inputs for main algorithm

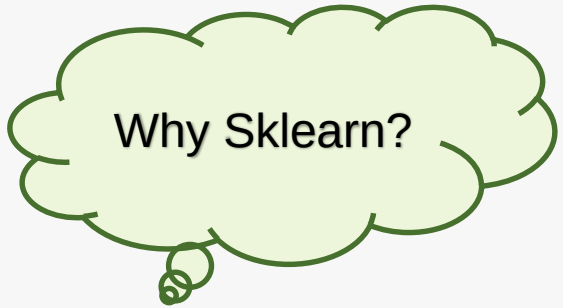
Building The Model

```
cols=['pH ','humidity %','ec ','temperature','n %','p %','k %']

df=pd.read_excel(r'E:\cdap pp\soil\NPK/dataset.xlsx')

pH=df['pH ']
humidity=df['humidity %']
ec=df['ec ']
temperature=df['temperature']
n=df['n %']
p=df['p %']
k=df['k %']
```

```
getData=[pH,humidity,ec,temperature,n,p,k]
list1=[]
for k in range(len(getData[0])):
    i=k
    x=[int(getData[0][i]),int(getData[1][i]),int(getData[2][i]),int(getData[3][i])]
    tot=0
    for j in range(len(x)):
        tot=float(tot)+x[j]
    x=[int(getData[0][i])/tot,int(getData[1][i])/tot,int(getData[2][i])/tot,int(getData[3][i])/tot,getData[4][i],getData[5][i],g
    list1.append(x)
```

Building The Model

```
"""
Nitrogen analysis
"""
list=list1
totPred=0
cp=0
model=None
X=None
Y=None
for i in range(10):
    shuffle(list)
    bigData=pd.DataFrame(list,columns=['','','','','label','t1','t2'])
    target=bigData['label']
    from sklearn.model_selection import  train_test_split
    X_train,X_test,Y_train,Y_test=train_test_split(bigData.drop(['label','t1','t2'],axis='columns'),target,test_size=0.2)
    model = LinearRegression()
    X=X_train
    Y=Y_train
    model.fit(X_train,Y_train)
    pred=model.score(X_test,Y_test)

    if(pred>cp):
        cp=pred
        filename = r'E:\cdap pp\soil\NPK/nModel.sav'
        pickle.dump(model, open(filename, 'wb'))

    totPred=pred+totPred

print("")
print("Algorithm : LinearRegression for n")
print("Train Data Count : ",len(X_train))
```




```
print("")
print("Algorithm : LinearRegression for n")
print("Train Data Count : ",len(X_train))
print("Test Data Count : ",len(X_test))
print("Accuracy :",cp)
```

```
Algorithm : LinearRegression for n
Train Data Count :  39
Test Data Count :  10
Accuracy : 0.9545439627162033
```

Building The Model

```
"""
phosphorus analysis
"""
list=list1
totPred=0
cp=0
model=None
X=None
Y=None
for i in range(10):
    shuffle(list)
    bigData=pd.DataFrame(list,columns=['','','','','t1','label','t2'])
    target=bigData['label']
    from sklearn.model_selection import  train_test_split
    X_train,X_test,Y_train,Y_test=train_test_split(bigData.drop(['t1','label','t2'],axis='columns'),target,test_size=0.02)
    model = LinearRegression()
    X=X_train
    Y=Y_train
    model.fit(X_train,Y_train)
    pred=model.score(X_test,Y_test)

    if(pred>cp):
        cp=pred
        filename = r'/Users/nayanalakshitha/Desktop/NPK/pModel.sav'
        pickle.dump(model, open(filename, 'wb'))

    totPred=pred+totPred

print("")
print("Algorithm : LinearRegression for p")
print("Train Data Count : ",len(X_train))
print("Test Data Count : ",len(X_test))
```

```
print("")
print("Algorithm : LinearRegression for p")
print("Train Data Count : ",len(X_train))
print("Test Data Count : ",len(X_test))
print("Accuracy :",cp)
```

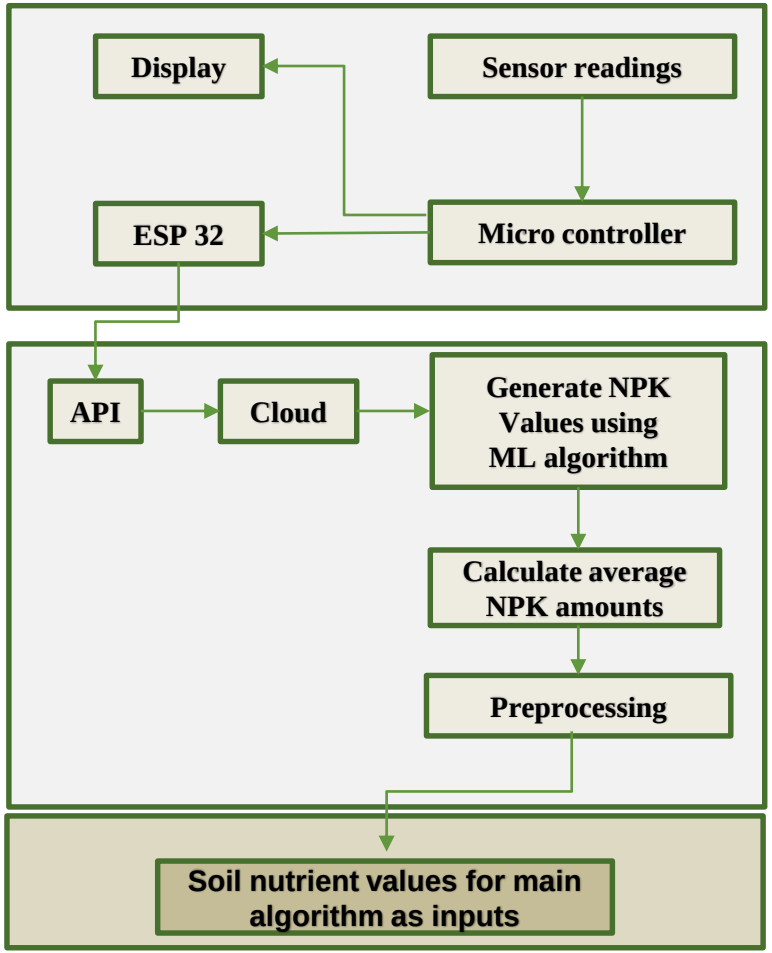
```
Algorithm : LinearRegression for p
Train Data Count :  39
Test Data Count :  10
Accuracy : 0.9627803795908306
```

Building The Model

```
"""
Potassium analysis
"""
list=list1
totPred=0
cp=0
model=None
X=None
Y=None
for i in range(10):
    shuffle(list)
    bigData=pd.DataFrame(list,columns=['','','','','t1','t2','label'])
    target=bigData['label']
    from sklearn.model_selection import  train_test_split
    X_train,X_test,Y_train,Y_test=train_test_split(bigData.drop(['t1','t2','label'],axis='columns'),target,test_size=0.2)
    model = LinearRegression()
    X=X_train
    Y=Y_train
    model.fit(X_train,Y_train)
    pred=model.score(X_test,Y_test)

    if(pred>cp):
        cp=pred
        filename = r'E:\cdap pp\soil\NPK/kModel.sav'
        pickle.dump(model, open(filename, 'wb'))

    totPred=pred+totPred

print("")
print("Algorithm : LinearRegression for k")
print("Train Data Count : ",len(X_train))
print("Test Data Count : ",len(X_test))
print("Accuracy :",cp)
print("")
```

```
Algorithm : LinearRegression for k
Train Data Count :  39
Test Data Count :  10
Accuracy : 0.9645037264356958
```

Component Overview



Display sub system

Preprocessing sub system

Next progress

01	Developing the IOT device	20%
02	Implementing firebase connection	05%
03	Implementing Test model	05%
04	Integrating the System	10%

Analysis Soil Condition Using an IOT Device

1. Study about plant disorders
2. Soil analysis
3. Data Collection & Preprocessing
4. Building the linear regression model
5. Studying about IOT
6. Developing the IOT device
7. Developing firebase connection and test model

COMPONENT FOUR

Implement a secure and distributed platform for identifying best commercial product for deficiency based on diagnosis

B. I. Sariffodeen
IT17354516

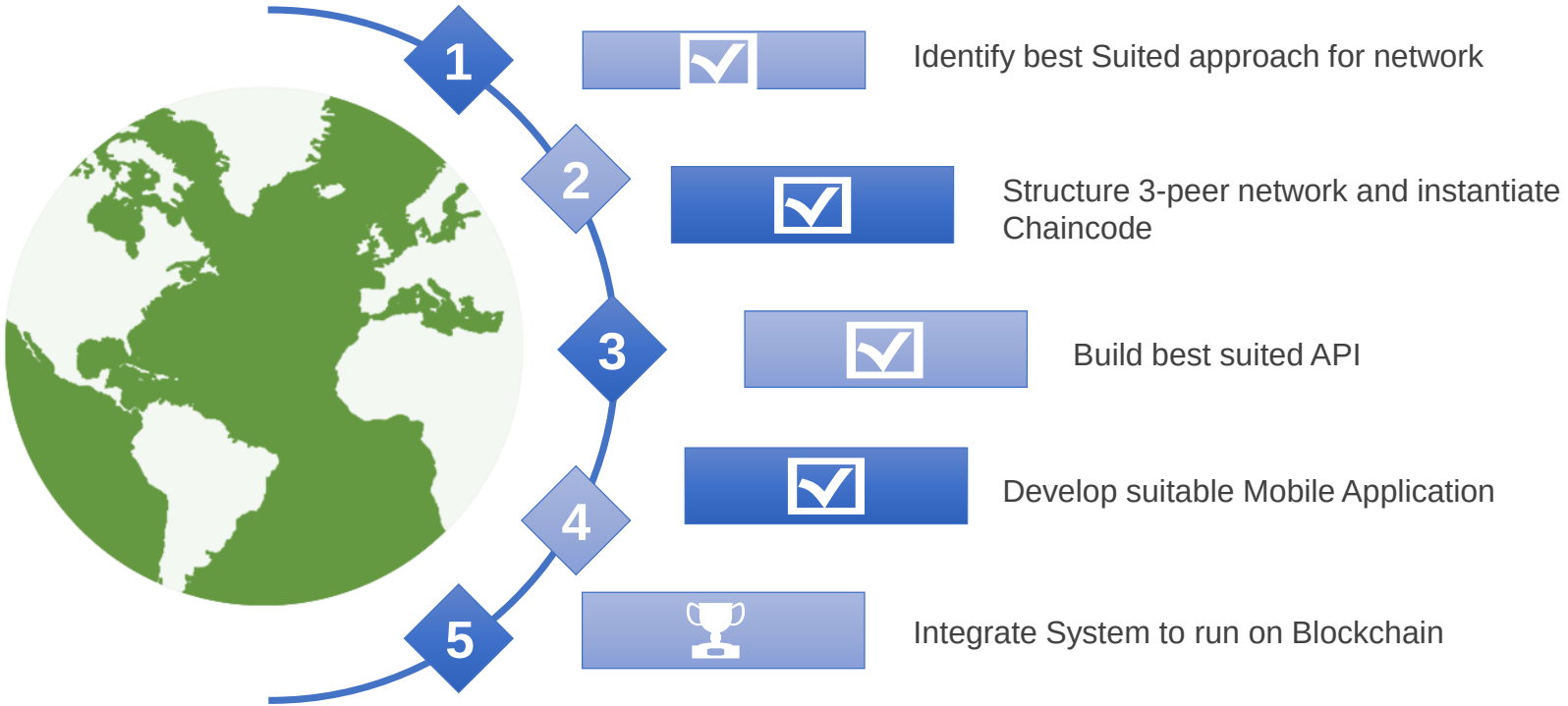
Research question ?

1. What is the context of the current ecosystem for fertilizer purchasing and administering?

2. What are the existing similar systems in the market & how improvements can be made on information symmetry?

3. How can a community driven decentralized solution be implemented to streamline these proceedings?

Workflow


1
2
3
4
5
Identify best Suited approach for network
Structure 3-peer network and instantiate Chaincode
Build best suited API
Develop suitable Mobile Application
Integrate System to run on Blockchain

Current Context

HYPERLEDGER

3 peer network constructed based on byfn.sh shell executable file available in fabric-samples.

Functionality achieved:

- All 3 entities can register and login (first time registration may take time because peers should communicate their copy of ledger on first sign-up)
- Farmer can receive expert advice by **describing** his problem.
- Famer can view his request sent to the expert & see if there's any response
- Farmer can ask vendor for product by describing product and view response.
- Farmer and Expert can also view product posted by vendors and their reviews.
- Vendor can advertise products and handout quotations.
- Expert can give farmer advice privately.
- Expert can rate product without vendor knowing.

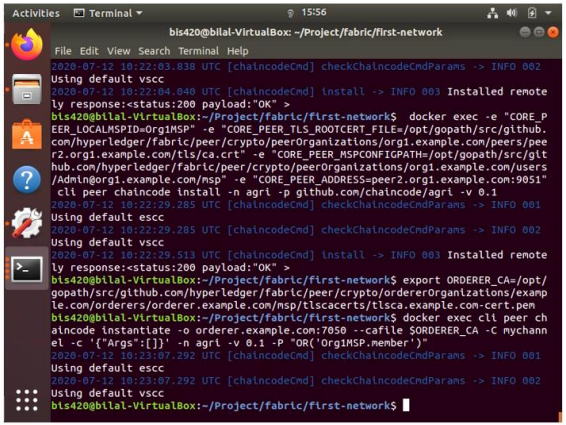
HYPERLEDGER

3-peer network for Farmer, Vendor & Expert parties

Ubuntu HLF 2 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

You have the **Auto capture keyboard** option turned on. This will cause the Virtual Machine to automatically **capture** the keyboard every time the VM window is activated and make it unavailable to other applications running on your host machine: when the keyboard is captured, all keystrokes (including system ones like



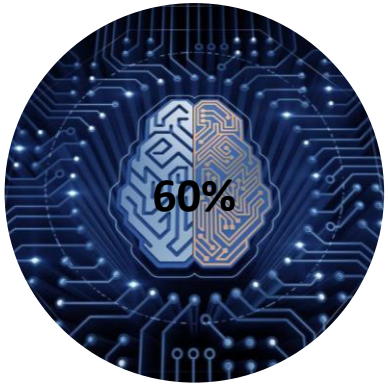
Right Ctrl

ENG INTL 3:56 PM 7/12/2020

Next progress

01	Integrate other Components to System	15%
02	Understand & Configure required changes in Chaincode	05%
03	Implementing Mobile Application	20%

Component 1



Nutrient deficiency Identification

Objectives

- ✓ Study Plant disorders and symptoms
- ✓ Sample Data Collection
- ✓ Preprocessing
- ✓ CNN building, prediction and testing
- • Wireframe design & Mobile UI Dev.
- • Collect dataset for cash crop.
- • Backend Dev. – Integrate model with mobile app and connect to CloudStore.

Component 2

Degree of Disorder Identification & Remedy Suggestion

Objectives

- ✓ Study Plant disorders and symptoms
- ✓ Sample Data Collection
- ✓ Preprocessing
- ✓ RCNN building, prediction and testing
- • Wireframe design & Mobile UI Dev.
- • Collect dataset for cash crop.
- • Backend Dev. – Integrate model with mobile app and connect to CloudStore.

Component 3

Develop an IOT device measure key nutrient values of soil

Objectives

- ✓ Study about Soil Nutrients
- ✓ Find the Colon Metric Mechanism for each element
- ✓ Study about photodiodes and sensors
- ✓ Make a formula for color variation due to nutrient variation.
- • Select suitable API.
- • Develop the Device.

Component 4

Secure and distributed platform to identify best commercial option

Objectives

- • Study about real world applicability to component.
- ✓ Identify best suited approach for blockchain implementation.
- ✓ Structure the required 3-peer network.
- ✓ Construct suitable Chaincode.
- ✓ Construct suitable API.
- • Build suitable Mobile Application.
- • Integrate other components to run on Blockchain network.