IMAGE PROCESSING

Project

โดย นายชนกันต์ ชุมทัพ 6410301022 และ นายฐิติภัทร์ ปรีดีดิลก 6410301024

Plant seedling recognition

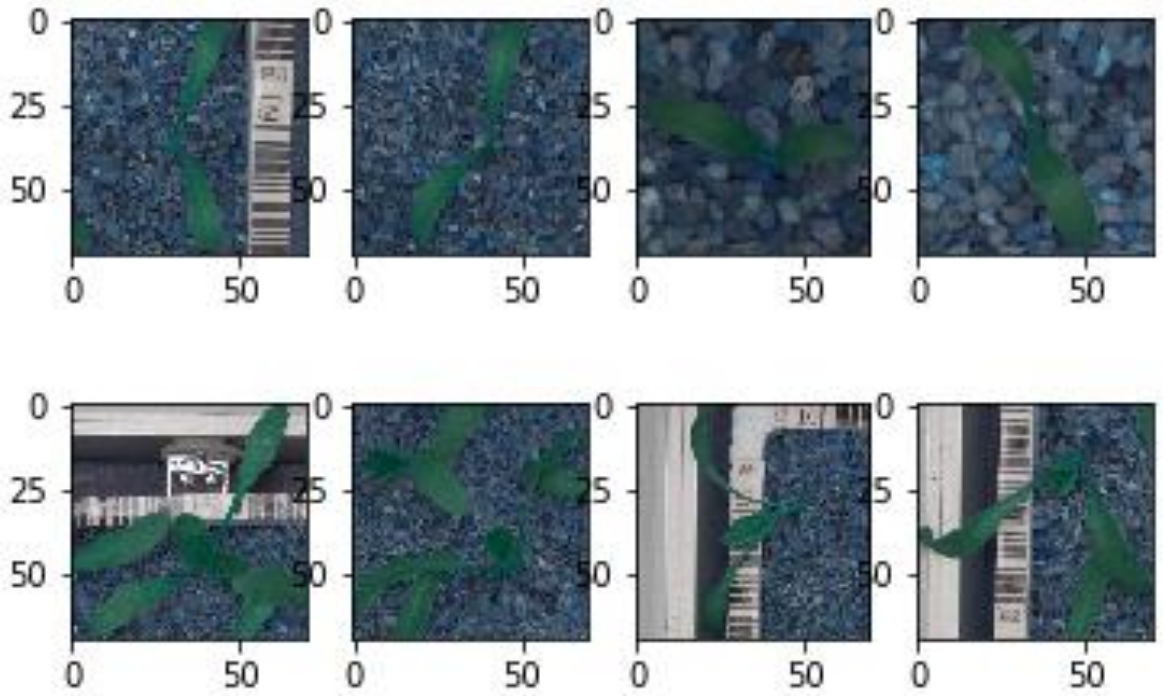
with Image Processing and CNN

What is this project for?

“ ที่มาของโครงการนี้เริ่มมาจากเกษตรกรประสบปัญหาในด้านการคัดแยกต้นอ่อนของพืช เพราะ พืชหลาย ๆ ชนิดที่อยู่ในวงตระกูลเดียวกัน จะมีต้นอ่อนที่เพิ่งขึ้นคล้ายคลึงกันมาก ทำให้การคัดแยกสามารถทำได้ยาก เราจึงไปค้นหาโครงงานภาพที่เกี่ยวข้องหรือคล้ายคลึงกับโครงงานที่เราจะทำ มาศึกษาขั้นตอนและวิธีการทำงานของโครงงาน เพื่อใช้ในการคัดแยกต้นอ่อนของพืชในงานต่าง ๆ ”

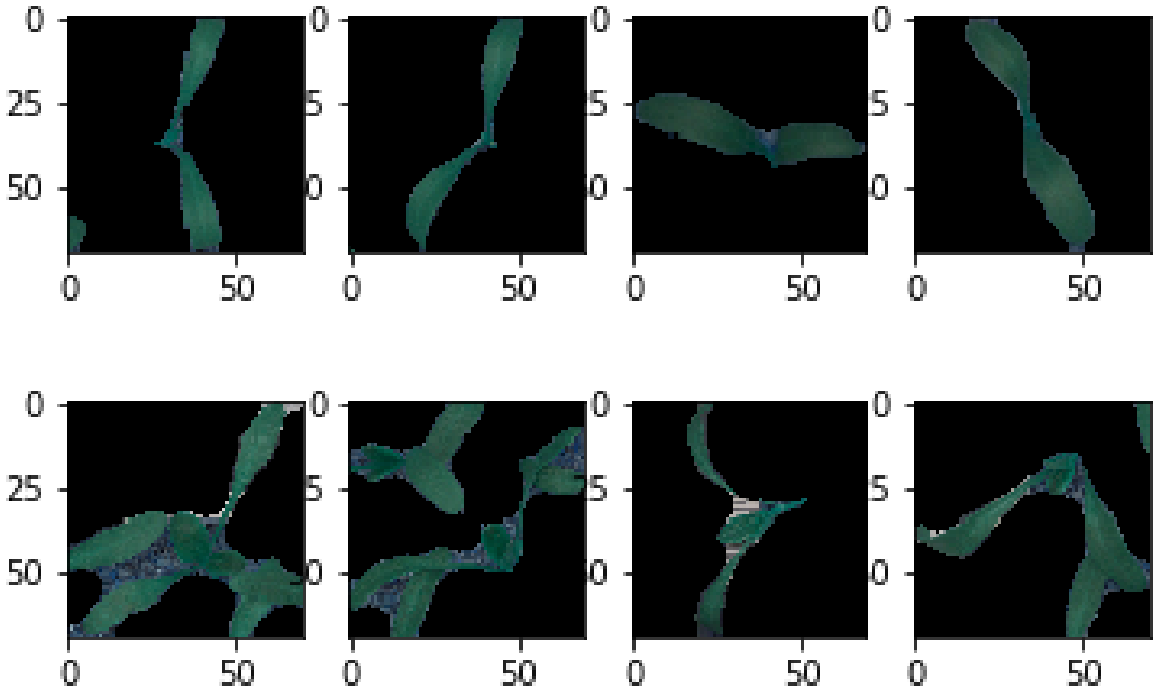
ข้อจำกัดในการทำ Image processing ของ Project นี้

ข้อจำกัดทางสภาพแวดล้อม



ภาพที่ได้มาจะมีสภาพแวดล้อมโดยธรรมชาติของพืช เช่น มี Background ที่เป็นดินหรือกรวด หรือ Noise ที่มาจากการถ่ายรูป ซึ่งทำให้วัตถุที่เราสนใจ ยากต่อการตรวจจับ เราจึงต้องหาวิธีมาประยุกต์ใช้งานให้เหมาะสม

ข้อจำกัดของระยะการถ่ายภาพ



จากการหาข้อมูลพบว่าการระบุชนิดของพืชที่แม่นยำที่สุดต้องดูจากใบ ทำให้ภาพที่จะนำมาประมวลผล ต้องเป็นภาพที่เห็นใบอย่างชัดเจน (Top View) เพื่อให้สามารถนำมาประมวลผลและระบุชนิดของพืชได้

ขั้นตอนการทำ

Steps to do

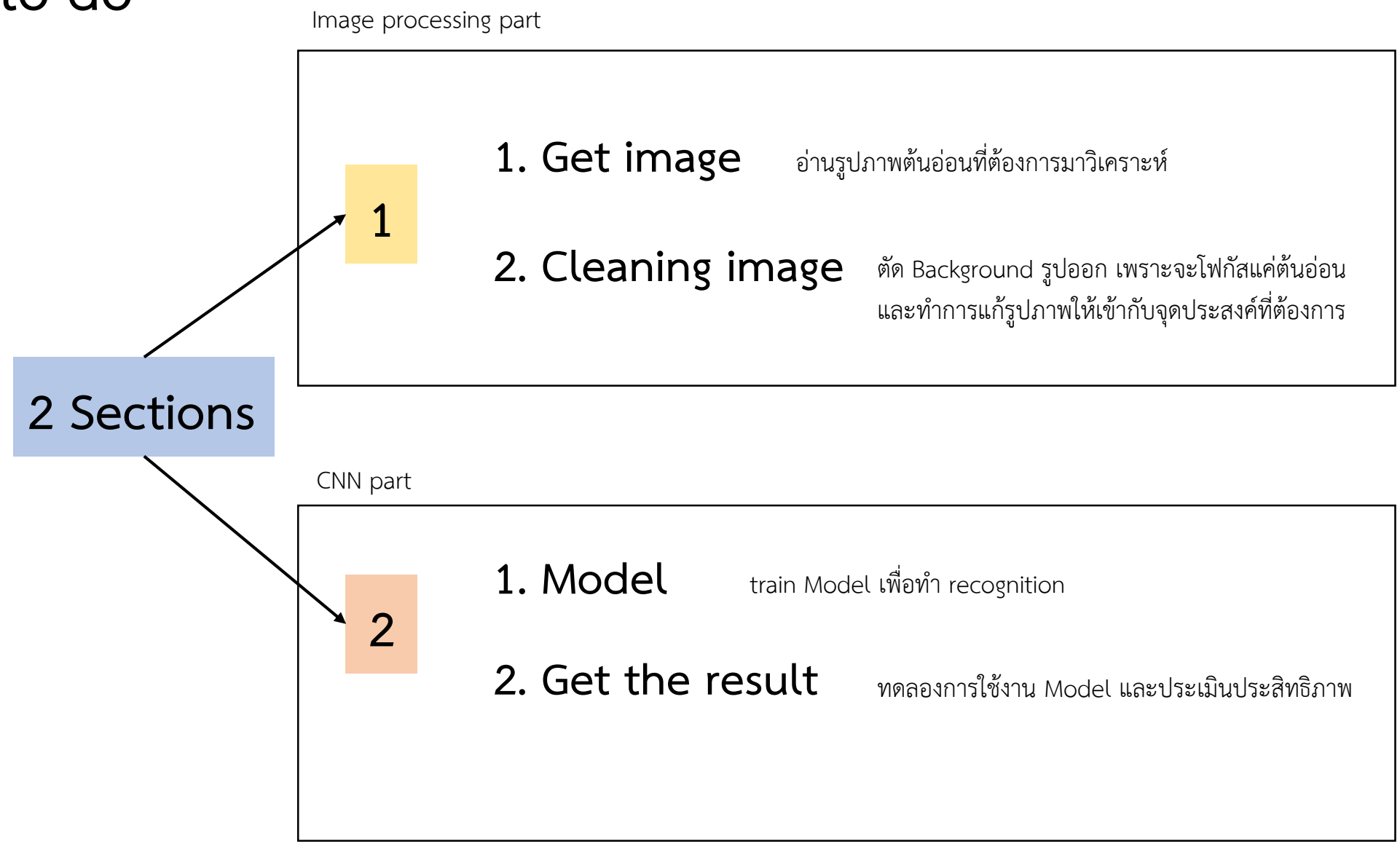


Image processing part
1
1. Get image
อ่านรูปภาพต้นอ่อนที่ต้องการมาวิเคราะห์
2. Cleaning image
ตัด Background รูปออก เพราะจะโฟกัสแค่ต้นอ่อน
และทำการแก้รูปภาพให้เข้ากับจุดประสงค์ที่ต้องการ
2 Sections
CNN part
2
1. Model
train Model เพื่อทำ recognition
2. Get the result
ทดลองการใช้งาน Model และประเมินประสิทธิภาพ

Image processing part

Step 1 – Get image

Image processing part

Step 1 – Get image

ในการอ่านค่ารูปภาพในงานนี้ เราจะทำการอ่านภาพและชื่อของพืชจากชุดข้อมูลที่มี
จากนั้นให้ Resize ภาพให้อยู่ที่ขนาด 70x70 px. เพื่อที่จะให้ในกระบวนการของ Training model นั้นรวดเร็วยิ่งขึ้น จากการใช้ภาพขนาดเล็ก

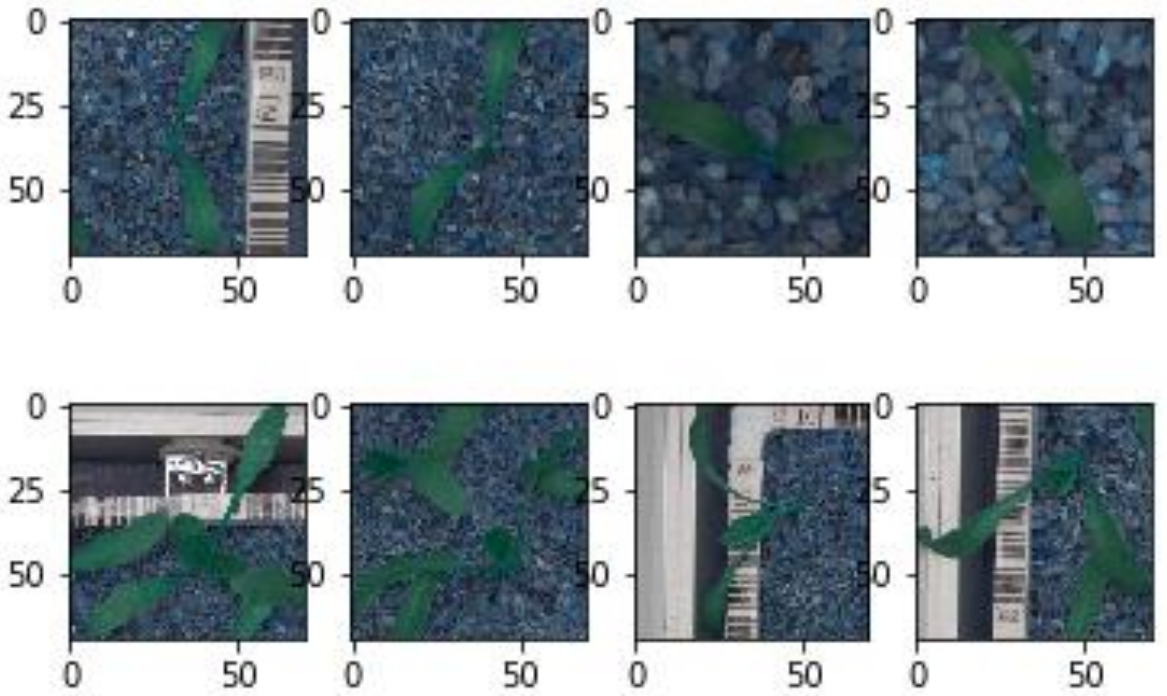
```
import cv2
from glob import glob
import numpy as np
from matplotlib import pyplot as plt
import math
import pandas as pd

ScaleTo = 70  # px to scale
seed = 7  # fixing random

path = '../input/plant-seedlings-classification/train/*/*.png'
files = glob(path)

trainImg = []
trainLabel = []
j = 1
num = len(files)

# Obtain images and resizing, obtain labels
for img in files:
    print(str(j) + "/" + str(num), end="\r")
    trainImg.append(cv2.resize(cv2.imread(img), (ScaleTo, ScaleTo)))  # Get image (with resizing)
    trainLabel.append(img.split('/')[-2])  # Get image label (folder name)
    j += 1

trainImg = np.asarray(trainImg)  # Train images set
trainLabel = pd.DataFrame(trainLabel)  # Train labels set
```

Image processing part

Step 1 – Get image

ทำการแสดงผลภาพที่อ่านมา

```
# Show some example images
for i in range(8):
    plt.subplot(2, 4, i + 1)
    plt.imshow(trainImg[i])
```



จะเห็นได้ว่ายังมี Background อยู่ทุก ๆ รูป ซึ่งเราต้องทำการลบออก
เพื่อให้ Model ที่จะถูก Trained มีความแม่นยำมากขึ้น

Image processing part

Step 2 – Cleaning image

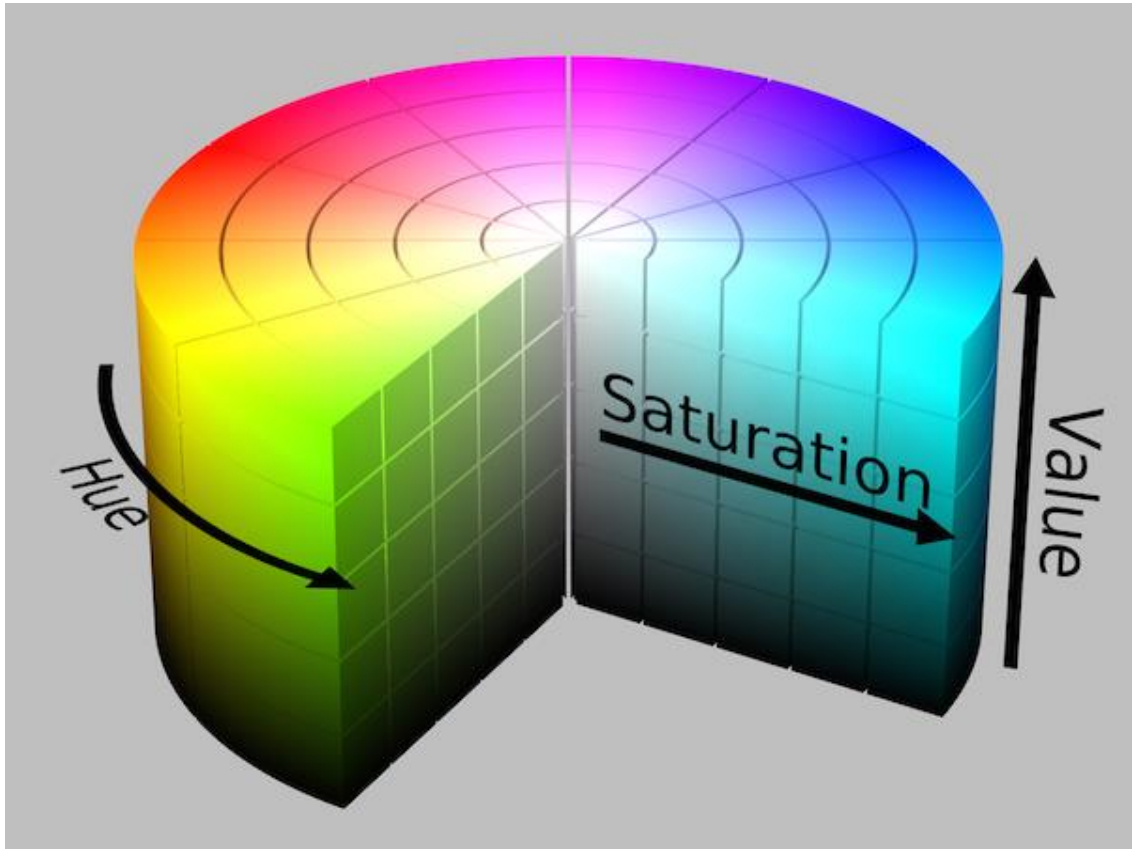
Step 2 – Cleaning image

ในการลบ Background ออกนั้น เราจะ assume ว่า

“สีของพืชทุก ๆ ต้นจากรูปภาพของเราเป็นสีเขียว”

โดยในกระบวนการทำนั้น เราจะ**ทำหน้ากากที่มากรองสีในรูปภาพที่ไม่ได้อยู่ใน Range สีเขียวออก และเหลือ Range สีเขียวไว้**

Step 2 – Cleaning image



Concept การทำ “หน้ากากกรองแต่สีเขียว”

กระบวนการทำในการตัด Background คือ เราจำเป็นต้องเปลี่ยนการใช้ Color model ของรูปภาพจาก RGB เป็น HSV เพราะ HSV เป็นระบบที่ใกล้เคียงกับการรับรู้สีของมนุษย์มากกว่า และเป็นระบบสีที่จะให้แง่ของเฉดสีและความอิ่มตัวมากกว่าแบบ RGB ทำให้ระบบนี้เหมาะกับการทำกับรูปภาพธรรมชาติที่มีสีจากความเป็นจริง อีกทั้งยังกำหนด Range ของสีได้ง่ายกว่าแบบ RGB

Step 2 – Cleaning image

การลบ Background แล้วโฟกัสแต่พืช

มี **Steps การทำ** ดังนี้

1. ใช้ Gaussian blur เพื่อลบ Noise ของภาพ
   (สามารถใช้ Blur filter แบบอื่นได้)
2. แปลง Color model จาก RGB เป็น HSV
3. สร้างหน้ากาก เพื่อกรองแต่สีเขียว
4. สร้างหน้ากาก Boolean
5. นำหน้ากากไปใช้กับรูปภาพเพื่อลบ Background

Image processing part

Step 2 – Cleaning image

ใช้ Gaussian blur เพื่อลบ Noise ของภาพ

แปลง Color model จาก RGB เป็น HSV

สร้างหน้ากาก เพื่อกรองแต่สีเขียว โดยกำหนด Range ของสีเขียว ทั้งด้านของ Lower และ Upper

สร้างหน้ากาก Boolean เพื่อลบสีในรูปภาพที่ไม่ต้องการออก

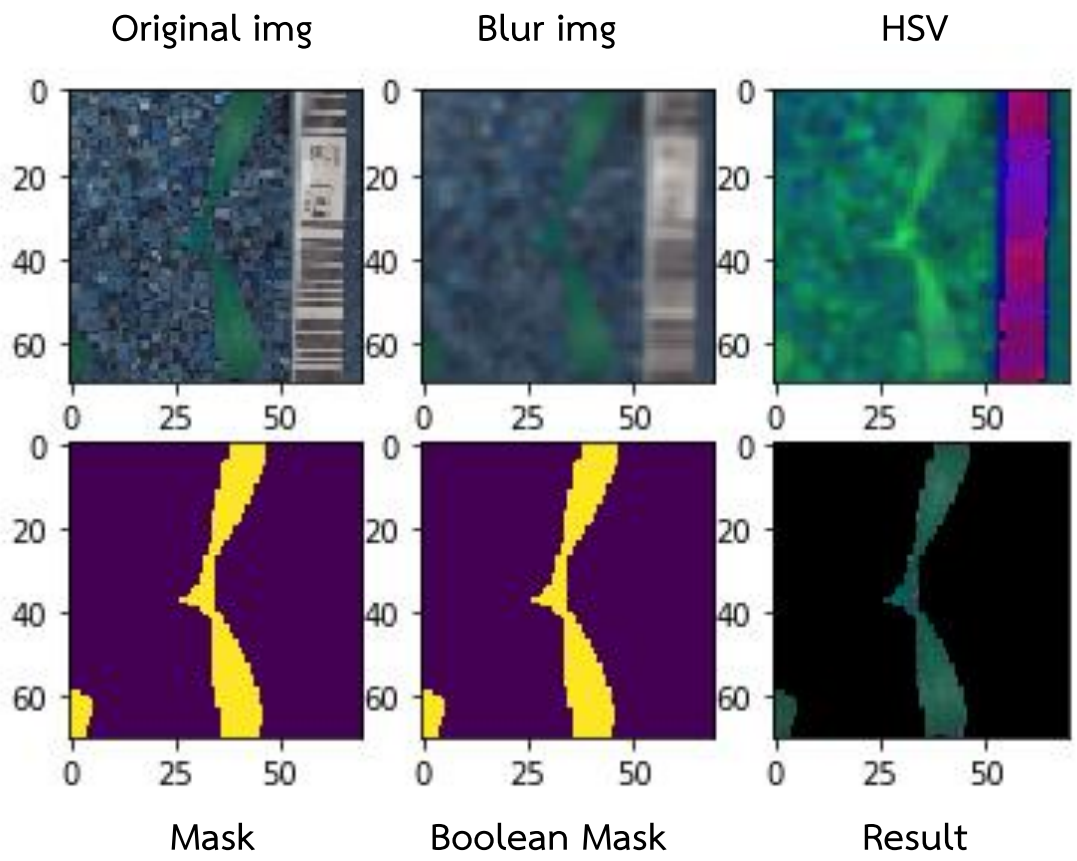
นำหน้ากากไปใช้กับรูปภาพเพื่อลบ Background แล้วเก็บผลลัพธ์

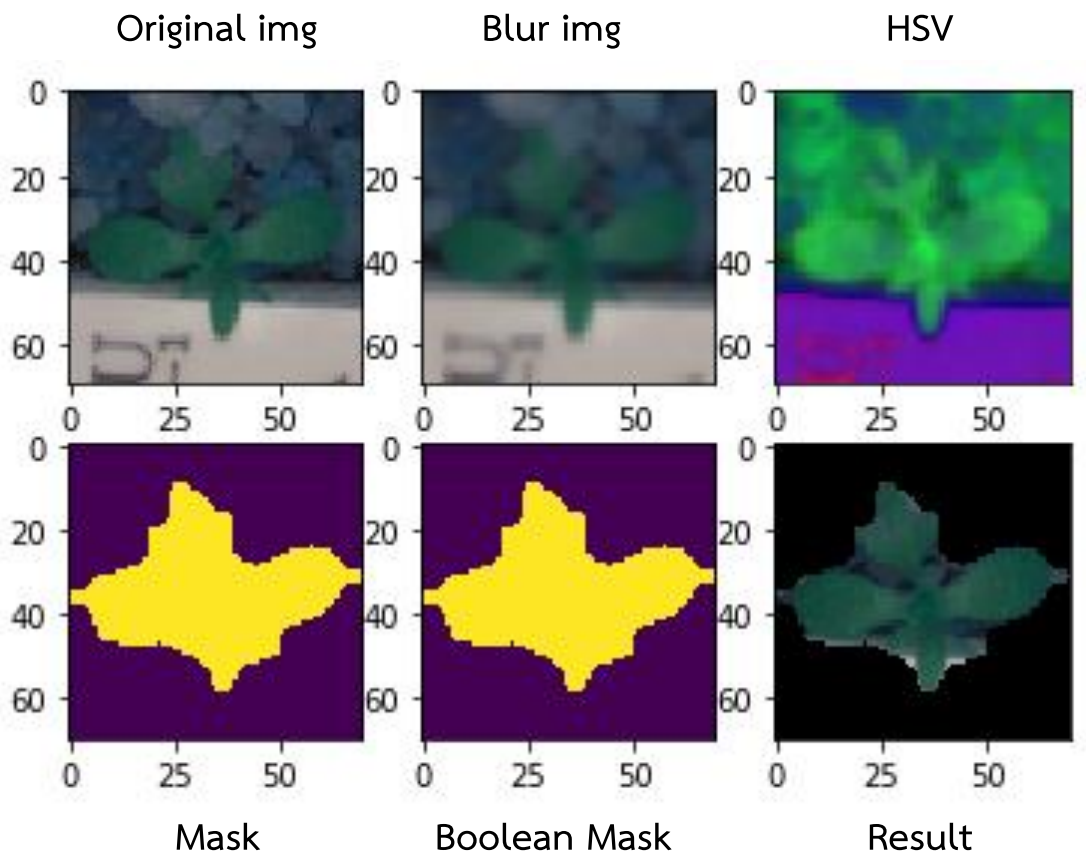
```
clearTrainImg = []
examples = []; getEx = True
for img in trainImg:
    # Use gaussian blur
    blurImg = cv2.GaussianBlur(img, (5, 5), 0)

    # Convert to HSV image
    hsvImg = cv2.cvtColor(blurImg, cv2.COLOR_BGR2HSV)

    # Create mask (parameters - green color range)
    lower_green = (25, 40, 50)
    upper_green = (75, 255, 255)
    mask = cv2.inRange(hsvImg, lower_green, upper_green)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11, 11))
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

    # Create bool mask
    bMask = mask > 0

    # Apply the mask
    clear = np.zeros_like(img, np.uint8)  # Create empty image
    clear[bMask] = img[bMask]  # Apply boolean mask to the origin image

    clearTrainImg.append(clear)  # Append image without backgroung

    # Show examples
    if getEx:
        plt.subplot(2, 3, 1); plt.imshow(img)  # Show the original image
        plt.subplot(2, 3, 2); plt.imshow(blurImg)  # Blur image
        plt.subplot(2, 3, 3); plt.imshow(hsvImg)  # HSV image
        plt.subplot(2, 3, 4); plt.imshow(mask)  # Mask
        plt.subplot(2, 3, 5); plt.imshow(bMask)  # Boolean mask
        plt.subplot(2, 3, 6); plt.imshow(clear)  # Image without background
        getEx = False

clearTrainImg = np.asarray(clearTrainImg)
```

Image processing part

Step 2 – Cleaning image

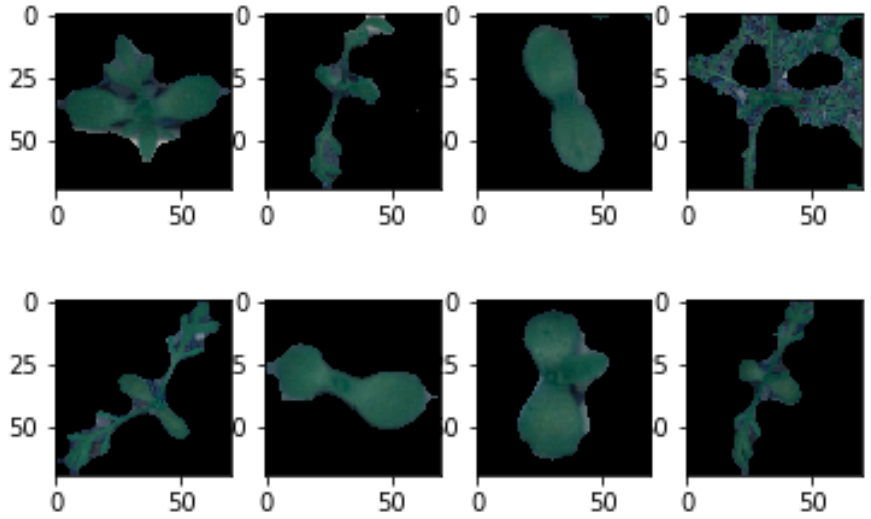
ผลลัพธ์

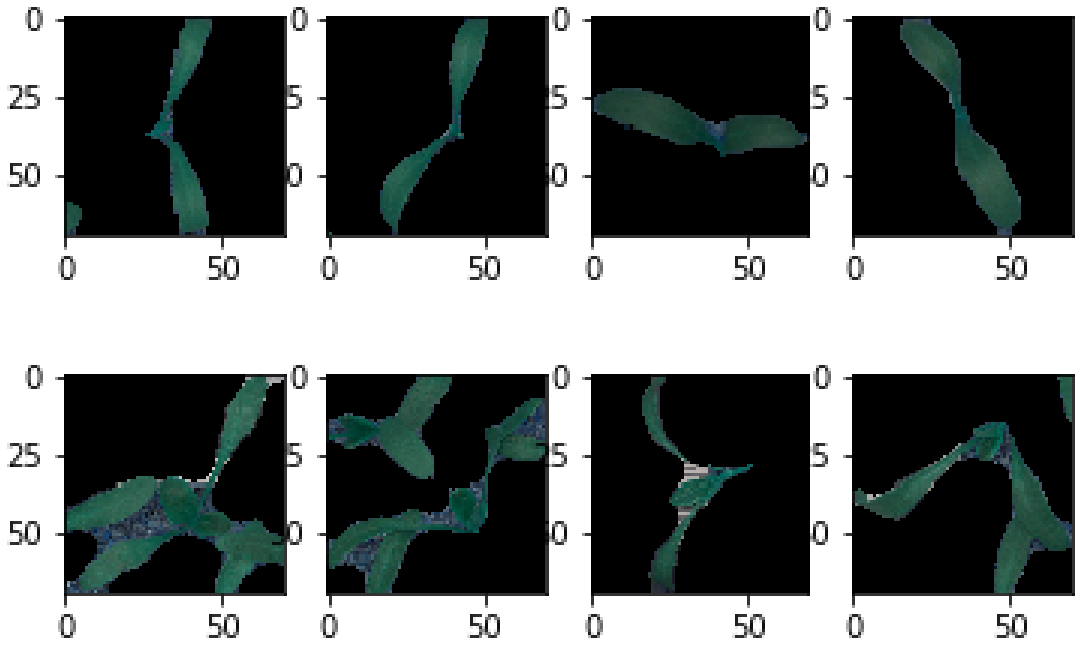




Step 2 – Cleaning image

ตัวอย่างผลลัพธ์ของรูปภาพทั้งหมด

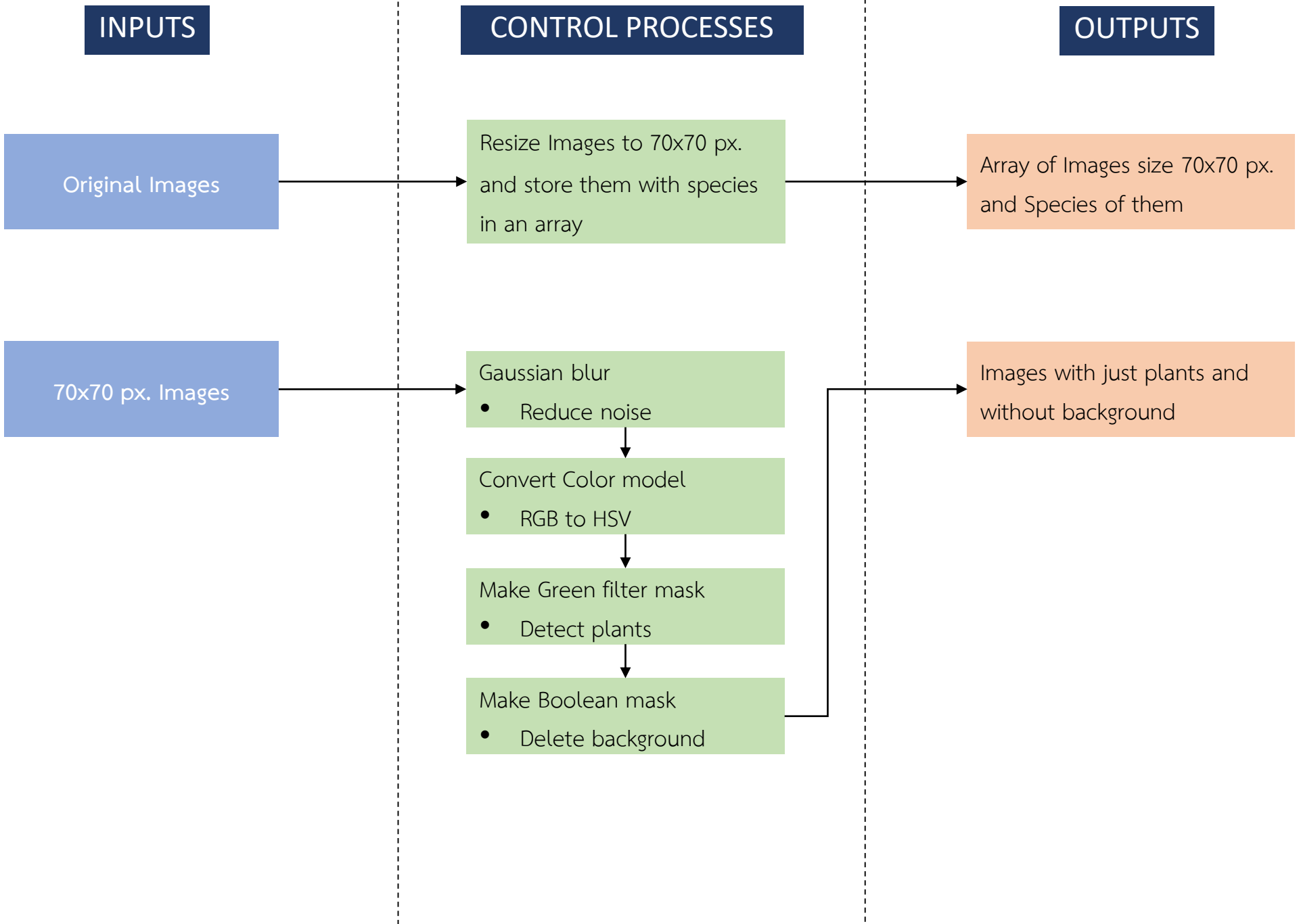




```
# Show sample result
for i in range(8):
    plt.subplot(2, 4, i + 1)
    plt.imshow(clearTrainImg[i])
```

Image processing part

Block Diagram


INPUTS
CONTROL PROCESSES
OUTPUTS
Original Images
Resize Images to 70x70 px. and store them with species in an array
Array of Images size 70x70 px. and Species of them
70x70 px. Images
Gaussian blur
• Reduce noise
Convert Color model
• RGB to HSV
Make Green filter mask
• Detect plants
Make Boolean mask
• Delete background
Images with just plants and without background

Step before going to CNN part

Step before going to CNN part

จัดการชุดข้อมูลที่ได้มา

1. **Normalize input** : จัดการช่วงสีของรูปภาพให้มี Range ที่แคบขึ้น เพื่อให้การ Train Model นั้นเร็วขึ้น

2. **Categories labels** : จัดการ encode species ของพืช (name) เป็นลำดับเพื่อใช้ในการ Train Model

Step before going to CNN part

1. **Normalize input** : จัดการช่วงสีของรูปภาพให้มี Range ที่แคบขึ้น เพื่อให้การ Train Model นั้นเร็วขึ้น

โดยการ Normalize จากช่วงของ RGB คือ 0-255 ให้เหลือแค่ช่วง 0-1

```
clearTrainImg = clearTrainImg / 255
```

Step before going to CNN part

2. Categories labels : จัดการ encode species ของพืช เป็นลำดับเพื่อใช้ในการ Train

Model

สร้าง Classes array จาก species ของพืช เช่น

['Black-grass' 'Charlock' 'Cleavers' 'Common Chickweed' 'Common wheat' 'Fat Hen' 'Loose Silky-bent' 'Maize'
'Scentless Mayweed' 'Shepherds Purse' 'Small-flowered Cranesbill' 'Sugar beet’]

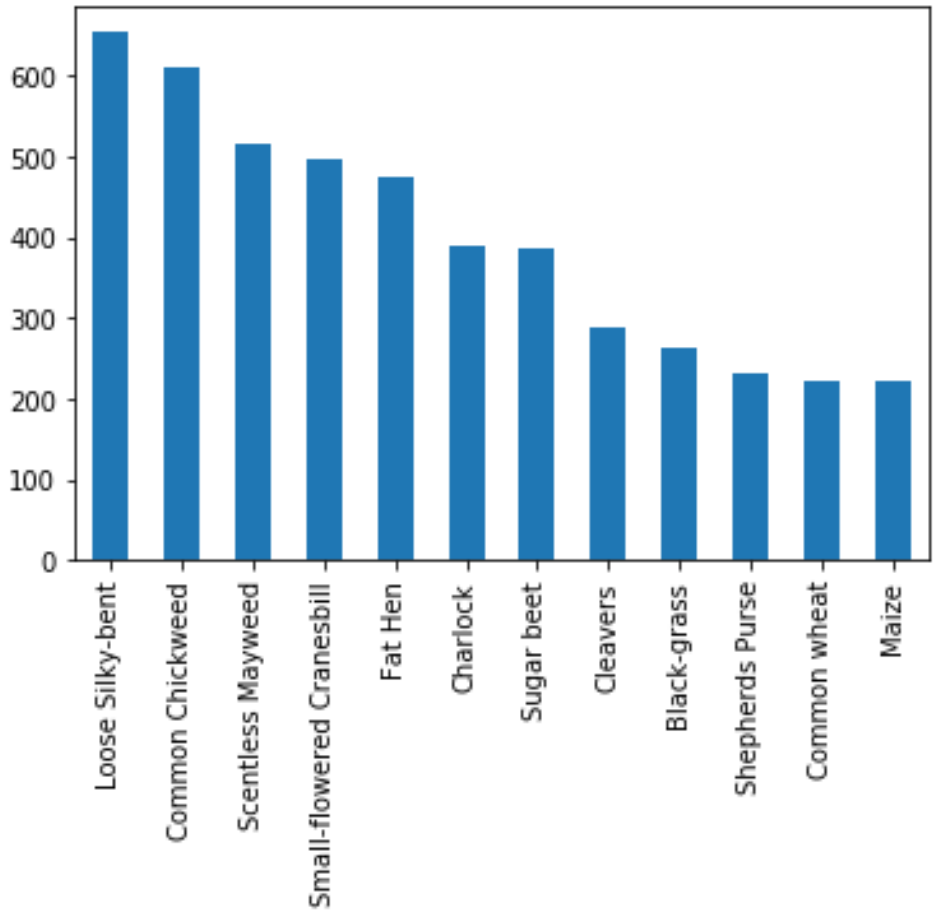
และทำการ Encode ทุก ๆ ชื่อตามลำดับใน Array เช่น

'Charlock' -> [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Step before going to CNN part

```
from keras.utils import np_utils
from sklearn import preprocessing
import matplotlib.pyplot as plt

# Encode labels and create classes
le = preprocessing.LabelEncoder()
le.fit(trainLabel[0])
print("Classes: " + str(le.classes_))
encodeTrainLabels = le.transform(trainLabel[0])

# Make labels categorical
clearTrainLabel = np_utils.to_categorical(encodeTrainLabels)
num_clases = clearTrainLabel.shape[1]
print("Number of classes: " + str(num_clases))

# Plot of label types numbers
trainLabel[0].value_counts().plot(kind='bar')
```

```
Classes: ['Black-grass' 'Charlock' 'Cleaver
s' 'Common Chickweed' 'Common wheat'
 'Fat Hen' 'Loose Silky-bent' 'Maize' 'Scen
tless Mayweed' 'Shepherds Purse'
 'Small-flowered Cranesbill' 'Sugar beet']
Number of classes: 12
```



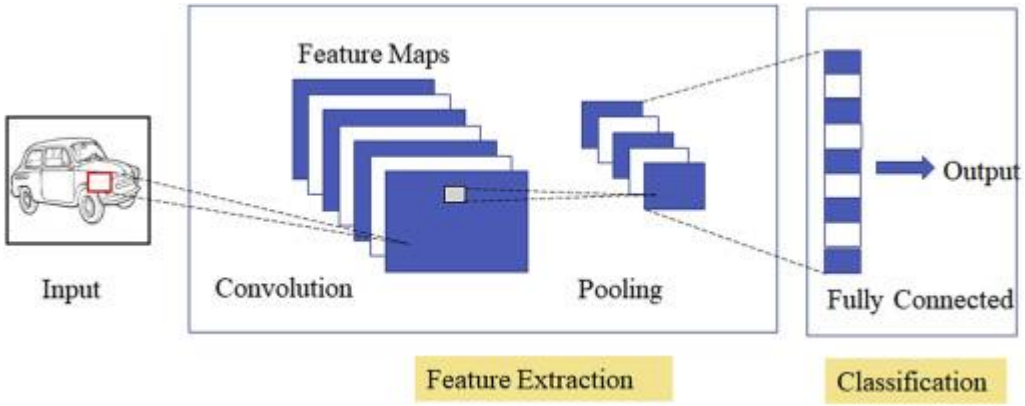
CNN part

Step 1 – Model

Step 1 – Model

Steps การสร้าง Model

1. **Split Dataset:** เพิ่มความแม่นยำและให้ข้อมูล Balanced ให้ Model โดยการมี Validation Set 10%
2. **Data generator:** set ให้ภาพไม่สมบูรณ์แบบเกินไป โดยการ เช่น zoom, rotate, flip เพื่อให้ Model ยืดหยุ่นในการใช้งาน
3. **Create Model:** ใช้หลักการ CNN
4. **Fit Model:** Train model ที่สร้าง



CNN part

Step 1 – Model

```
from sklearn.model_selection import train_test_split

trainX, testX, trainY, testY = train_test_split(clearTrainImg, clearTrainLabel,
                                                test_size=0.1, random_state=seed,
                                                stratify = clearTrainLabel)
```

```
from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(
        rotation_range=180,  # randomly rotate images in the range
        zoom_range = 0.1, # Randomly zoom image
        width_shift_range=0.1,  # randomly shift images horizontally
        height_shift_range=0.1,  # randomly shift images vertically
        horizontal_flip=True,  # randomly flip images horizontally
        vertical_flip=True  # randomly flip images vertically
    )
datagen.fit(trainX)
```

```
import numpy
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers import BatchNormalization

numpy.random.seed(seed)  # Fix seed

model = Sequential()

model.add(Conv2D(filters=64, kernel_size=(5, 5), input_shape=(ScaleTo, ScaleTo, 3), activation='rel
u'))
model.add(BatchNormalization(axis=3))
model.add(Conv2D(filters=64, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(BatchNormalization(axis=3))
model.add(Dropout(0.1))

model.add(Conv2D(filters=128, kernel_size=(5, 5), activation='relu'))
model.add(BatchNormalization(axis=3))
model.add(Conv2D(filters=128, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(BatchNormalization(axis=3))
model.add(Dropout(0.1))

model.add(Conv2D(filters=256, kernel_size=(5, 5), activation='relu'))
model.add(BatchNormalization(axis=3))
model.add(Conv2D(filters=256, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(BatchNormalization(axis=3))
model.add(Dropout(0.1))

model.add(Flatten())

model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(num_clases, activation='softmax'))

model.summary()

# compile model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

CNN part

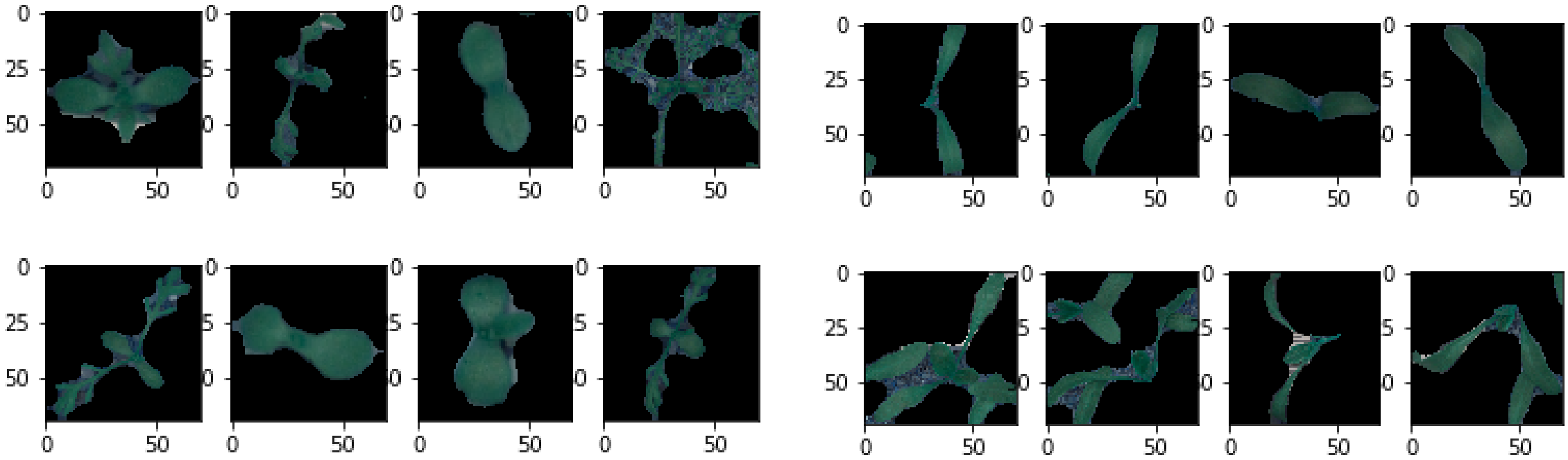
Step 1 – Model

```
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, CSVLogger

# learning rate reduction
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=3,
                                            verbose=1,
                                            factor=0.4,
                                            min_lr=0.00001)

# checkpoints
filepath="drive/DataScience/PlantReco/weights.best_{epoch:02d}-{val_acc:.2f}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_acc',
                             verbose=1, save_best_only=True, mode='max')
filepath="drive/DataScience/PlantReco/weights.last_auto4.hdf5"
checkpoint_all = ModelCheckpoint(filepath, monitor='val_acc',
                                 verbose=1, save_best_only=False, mode='max')

# all callbacks
callbacks_list = [checkpoint, learning_rate_reduction, checkpoint_all]

# fit model
# hist = model.fit_generator(datagen.flow(trainX, trainY, batch_size=75),
#                            epochs=35, validation_data=(testX, testY),
#                            steps_per_epoch=trainX.shape[0], callbacks=callbacks_list)
```

CNN part

Step 2 – Get the Result

Step 2 – Get the result



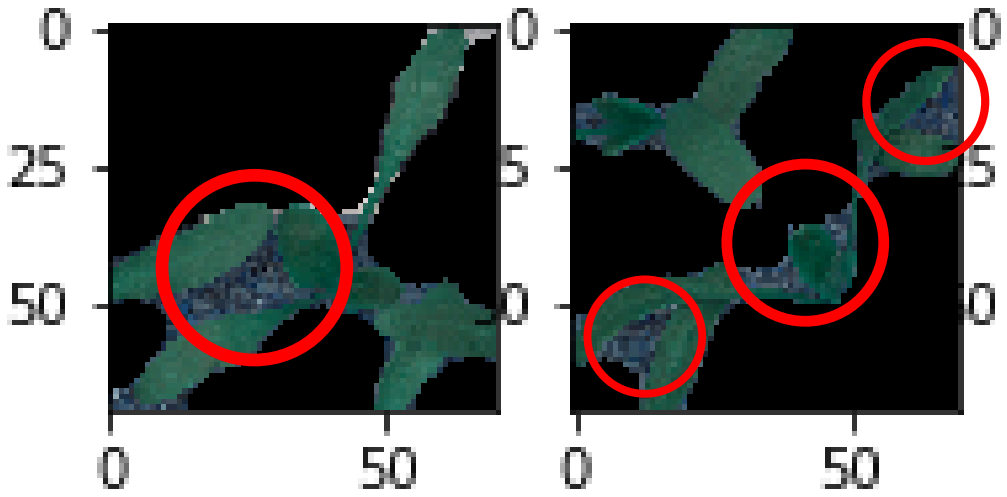
ในการใช้งานจริง ๆ ตัว Model จะสามารถแยกได้ว่าพืชต้นกล้าชนิดไหนคืออะไร

Project

Development process

Development process

ปัญหาที่พบในการทำ คือ Background ของภาพบางรูปถูกตัดไม่หมด ซึ่งเกิดจากการใช้หลัก Morphological แบบ closing ในการประมวลผลเพื่อลด Noise ที่อาจจะอยู่บนใบไม้ให้หายไป แต่ก็แลกมากลับการทำให้สีของ Background ถูกเกลี่ยให้สีใกล้กับ Range สีของใบไม้ที่กำหนดไว้ ซึ่งการที่ได้ภาพที่ไม่ใช่ใบไม้ทั้งหมด อาจทำให้ความแม่นยำของ Model ที่ Trained ลดลง



Development process

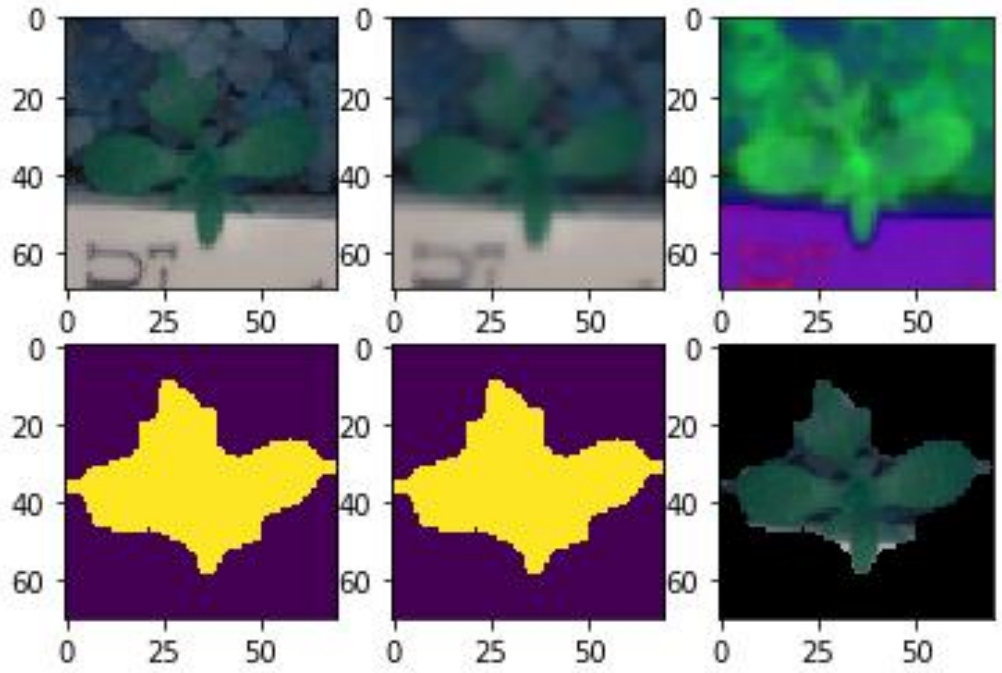
แนวคิดการแก้ปัญหา คือ พวกเราได้ลองลบการใช้ closing ออกและทำการประมวลผลเลย ซึ่งผลที่ออกมาจะเห็นได้ว่าสามารถตัดต้นไม้จาก Background ได้คมยิ่งขึ้น

```
clearTrainImg = []
examples = []; getEx = True
for img in trainImg:
    # Use gaussian blur
    blurImg = cv2.GaussianBlur(img, (5, 5), 0)

    # Convert to HSV image
    hsvImg = cv2.cvtColor(blurImg, cv2.COLOR_BGR2HSV)

    # Create mask (parameters - green color range)
    lower_green = (25, 40, 50)
    upper_green = (75, 255, 255)
    mask = cv2.inRange(hsvImg, lower_green, upper_green)
[redacted]

    # Create bool mask
    bMask = mask > 0

    # Apply the mask
    clear = np.zeros_like(img, np.uint8)  # Create empty image
    clear[bMask] = img[bMask]  # Apply boolean mask to the origin image

    clearTrainImg.append(clear)  # Append image without backgroung

    # Show examples
    if getEx:
        plt.subplot(2, 3, 1); plt.imshow(img)  # Show the original image
        plt.subplot(2, 3, 2); plt.imshow(blurImg)  # Blur image
        plt.subplot(2, 3, 3); plt.imshow(hsvImg)  # HSV image
        plt.subplot(2, 3, 4); plt.imshow(mask)  # Mask
        plt.subplot(2, 3, 5); plt.imshow(bMask)  # Boolean mask
        plt.subplot(2, 3, 6); plt.imshow(clear)  # Image without background
        getEx = False

clearTrainImg = np.asarray(clearTrainImg)
```

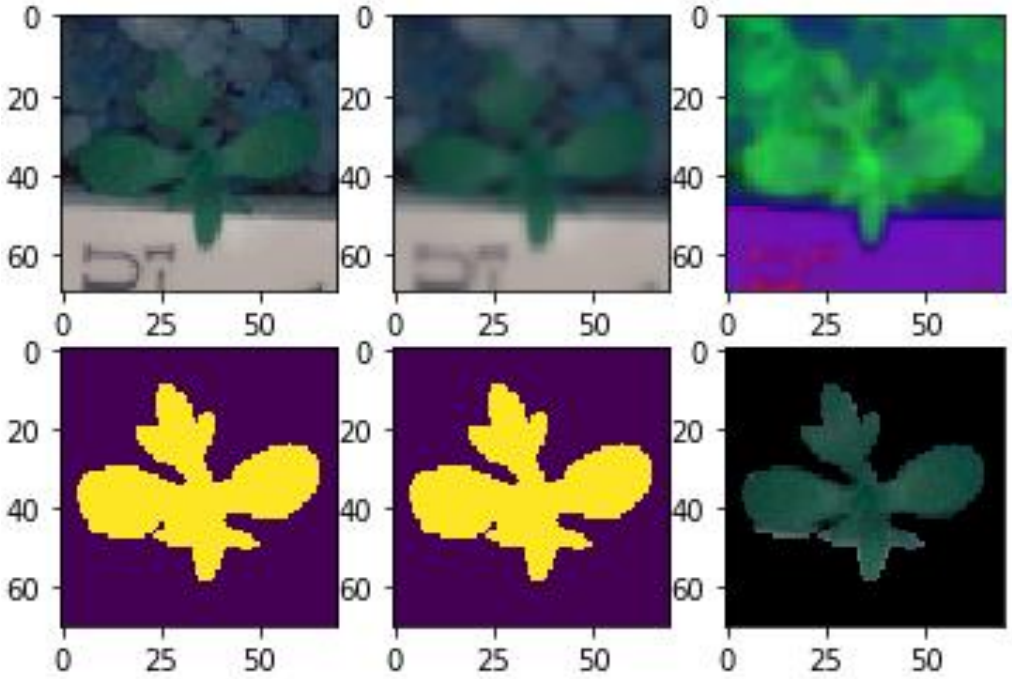
Project

Development process

ก่อนการแก้ปัญหา



หลังการแก้ปัญหา



Development process

อีกแนวทางการแก้ปัญหา คือ ต้องมีตัวกำหนด threshold ของรูปภาพและเทียบกับค่าสีของรูปภาพ ว่าถ้าหากเกินจุดนี้ ให้ทำการทำ Boolean Mask เพื่อตัด Background ออกจากสิ่งที่สนใจ ซึ่งข้อดี คือ เราจะสามารถกรองรูปภาพได้ว่ารูปไหนเข้าเกณฑ์ที่สามารถนำไปใช้งานได้ รูปไหนไม่เหมาะไปใช้งานได้ แต่ข้อเสีย คือ จะมีบางรูปเท่านั้นที่ถูกตัดให้คม ไม่สามารถตัดทุกรูปได้

Development process

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

clearTrainImg = []
examples = []
getEx = True

def calculate_probabilities(hsvImg, lower_range, upper_range):
    # Create a mask based on the provided HSV range
    mask = cv2.inRange(hsvImg, lower_range, upper_range)

    # Calculate the probability of each pixel belonging to the plant (foreground)
    pixel_count = np.count_nonzero(mask)
    total_pixels = hsvImg.shape[0] * hsvImg.shape[1]
    plant_probability = pixel_count / total_pixels

    return plant_probability, mask

for img in trainImg:
    # Use Gaussian blur
    blurImg = cv2.GaussianBlur(img, (5, 5), 0)
```

```python
    # Convert to HSV image
    hsvImg = cv2.cvtColor(blurImg, cv2.COLOR_BGR2HSV)

    # Define lower and upper bounds for the plant color (adjust these ranges)
    lower_green = (25, 40, 40)
    upper_green = (75, 255, 255)

    # Calculate the probabilities for the plant and the background
    plant_probability, mask = calculate_probabilities(hsvImg, lower_green, upper_green)

    # Choose a threshold for separating plant from background based on probability
    # You may need to experiment with this threshold value
    probability_threshold = 0.5  # Adjust as needed


    if plant_probability > probability_threshold:
        # Create a boolean mask
        bMask = mask > 0

        # Apply the mask to the original image
        clear = np.zeros_like(img, np.uint8)
        clear[bMask] = img[bMask]

        clearTrainImg.append(clear)

    # Show examples
    if getEx:
        plt.subplot(2, 3, 1); plt.imshow(img)  # Show the original image
        plt.subplot(2, 3, 2); plt.imshow(blurImg)  # Blur image
        plt.subplot(2, 3, 3); plt.imshow(hsvImg)  # HSV image
        plt.subplot(2, 3, 4); plt.imshow(mask)  # Mask
        plt.subplot(2, 3, 5); plt.imshow(bMask)  # Boolean mask
        plt.subplot(2, 3, 6); plt.imshow(clear)  # Image without background
        print(plant_probability);
        getEx = False

clearTrainImg = np.asarray(clearTrainImg)
```

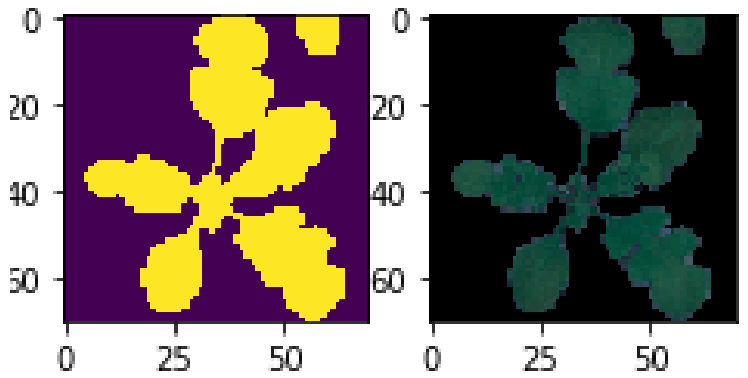
Development process

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

clearTrainImg = []
examples = []
getEx = True

def calculate_probabilities(hsvImg, lower_range, upper_range):
    # Create a mask based on the provided HSV range
    mask = cv2.inRange(hsvImg, lower_range, upper_range)

    # Calculate the probability of each pixel belonging to the plant (foreground)
    pixel_count = np.count_nonzero(mask)
    total_pixels = hsvImg.shape[0] * hsvImg.shape[1]
    plant_probability = pixel_count / total_pixels

    return plant_probability, mask

for img in trainImg:
    # Use Gaussian blur
    blurImg = cv2.GaussianBlur(img, (5, 5), 0)
```

```python
    # Convert to HSV image
    hsvImg = cv2.cvtColor(blurImg, cv2.COLOR_BGR2HSV)

    # Define lower and upper bounds for the plant color (adjust these ranges)
    lower_green = (25, 40, 40)
    upper_green = (75, 255, 255)

    # Calculate the probabilities for the plant and the background
    plant_probability, mask = calculate_probabilities(hsvImg, lower_green, upper_green)

    # Choose a threshold for separating plant from background based on probability
    # You may need to experiment with this threshold value
    probability_threshold = 0.5  # Adjust as needed


    if plant_probability > probability_threshold:
        # Create a boolean mask
        bMask = mask > 0

        # Apply the mask to the original image
        clear = np.zeros_like(img, np.uint8)
        clear[bMask] = img[bMask]

        clearTrainImg.append(clear)

    # Show examples
    if getEx:
        plt.subplot(2, 3, 1); plt.imshow(img)  # Show the original image
        plt.subplot(2, 3, 2); plt.imshow(blurImg)  # Blur image
        plt.subplot(2, 3, 3); plt.imshow(hsvImg)  # HSV image
        plt.subplot(2, 3, 4); plt.imshow(mask)  # Mask
        plt.subplot(2, 3, 5); plt.imshow(bMask)  # Boolean mask
        plt.subplot(2, 3, 6); plt.imshow(clear)  # Image without background
        print(plant_probability);
        getEx = False

clearTrainImg = np.asarray(clearTrainImg)
```

Development process

การแก้ปัญหาในวิธีการมี threshold เป็นตัวอ้างอิง



Project

Further Development

รวมถึงเราสามารถนำไปต่อยอดเพื่อดูโรคของพืชได้ด้วย และอื่น ๆ เป็นต้น

สำหรับการพัฒนานำโปรเจคนี้ไปต่อยอด

สามารถนำไปทำการตรวจจับวัชพืชที่จะที่แย่งสารอาหารจากต้นอ่อนทำให้สารอาหารของต้นอ่อนไม่เพียงพอจนทำให้ตาย ซึ่งหลักการพัฒนาคร่าวๆ เช่น Detect วัชพืช ด้วยวิธี Canny หรือการเช็คสีของพืช และทำการ Train Model ให้จดจำความต่างระหว่างพืชปกติกับพืชที่มีวัชพืช เป็นต้น