ถ้าต้องการพัฒนา AI ประมวลผลต้นไม้ป่าชายเลน เพื่อบอก ความสมบูรณ์ของต้นไม้ หรือ สร้างภาพจินตนาการ (imagine things)

**สามารถทำได้ 2 แนวทางหลักตามนี้**

**1. การวิเคราะห์ความสมบูรณ์ของต้นไม้**

ขั้นตอนหลัก :

1. **เก็บข้อมูลภาพถ่าย**
   * ใช้ โดรน หรือ กล้องมือถือ เก็บภาพต้นไม้ในป่าชายเลนจากหลายมุม
   * ถ่ายภาพระยะใกล้เพื่อดูใบ, กิ่ง, และลำต้น
   * ถ่ายภาพจากมุมสูงเพื่อดู canopy cover (การปกคลุมของต้นไม้)
2. **สร้าง Dataset**
   * จัดกลุ่มภาพตาม ระดับความสมบูรณ์ (ดี, ปานกลาง, เสื่อมโทรม)
   * เพิ่มข้อมูลเช่น สีของใบ, รูปทรงต้นไม้, ร่องรอยของโรค, การมีอยู่ของสัตว์
3. **พัฒนาโมเดล AI ด้วย Computer Vision**
   * ใช้ Deep Learning (CNN - Convolutional Neural Network)
   * โมเดลที่เหมาะสม เช่น ResNet, MobileNet, EfficientNet
   * ใช้ Pre-trained models บน TensorFlow หรือ PyTorch
4. **เทรนโมเดล & ทดสอบ**
   * ใช้เทคนิค Image Augmentation เพื่อเพิ่มความหลากหลายของข้อมูล
   * Train และปรับแต่ง hyperparameters
   * ทดสอบ accuracy ของโมเดลด้วย test dataset
5. **Deploy โมเดล**
   * ทำเป็น Web App / Mobile App ให้ผู้ใช้ถ่ายภาพแล้ว AI วิเคราะห์ผล
   * ใช้ Flask/FastAPI (สำหรับ Web) หรือ TensorFlow Lite (สำหรับ Mobile)

**2. การสร้างภาพจินตนาการ (Imagine Things)**

ถ้าต้องการให้ AI "มโนภาพ" หรือสร้างภาพต้นไม้ในป่าชายเลนที่ดูสมบูรณ์/เสื่อมโทรม สามารถใช้ Generative AI ได้

แนวทางที่ใช้ได้

1. Generative Adversarial Networks (GANs)
   * ใช้ StyleGAN, BigGAN, CycleGAN เพื่อสร้างภาพป่าชายเลน
   * ฝึกโมเดลให้สร้างภาพต้นไม้สมบูรณ์หรือเสื่อมโทรมจาก dataset
2. Stable Diffusion / DALL·E
   * ใช้ text-to-image models เช่น Stable Diffusion หรือ DALL·E
   * ป้อนคำสั่งเช่น *"A thriving mangrove tree with lush green leaves"*
3. NeRF (Neural Radiance Fields) สำหรับ 3D Visualization
   * ใช้ NeRF เพื่อสร้างป่าชายเลนแบบ 3D
   * ใช้ข้อมูลภาพถ่ายจากมุมต่างๆ มาสร้างฉากจำลอง

**เครื่องมือที่ใช้ได้**

✅ Computer Vision: OpenCV, TensorFlow, PyTorch  
✅ Generative AI: StyleGAN, Stable Diffusion, DALL·E  
✅ Data Collection: Drone imagery, Satellite images, Mobile camera  
✅ Deployment: Flask, FastAPI, TensorFlow Lite

**วิธีสร้าง Dataset สำหรับ AI วิเคราะห์ความสมบูรณ์ของต้นไม้ป่าชายเลน**

**1. การเก็บรวบรวมข้อมูล (Data Collection)**

ต้องมีภาพต้นไม้ป่าชายเลนจากหลายมุม และครอบคลุมหลายระดับของความสมบูรณ์ เช่น ต้นไม้ที่แข็งแรง, เริ่มเสื่อมโทรม, และกำลังตาย

แหล่งข้อมูลที่ใช้ได้

✅ ถ่ายภาพเอง

* ใช้ โดรน ถ่ายภาพจากมุมสูง
* ใช้ มือถือหรือกล้อง ถ่ายภาพระยะใกล้ของลำต้น, ใบ, และราก

✅ ใช้ภาพจากแหล่งข้อมูลเปิด (Open Data)

* Kaggle
* Google Earth / Google Street View
* Global Forest Watch

✅ ใช้ข้อมูลจากดาวเทียม (ถ้ามี)

* Sentinel-2 (ESA)
* Landsat (NASA)

**2. การจัดกลุ่มข้อมูล (Data Labeling)**

หลังจากได้ภาพมาแล้ว ต้อง จัดหมวดหมู่ (Labeling) ตามระดับความสมบูรณ์ เช่น:

ตัวอย่างการ Labeling

| หมวดหมู่ความสมบูรณ์ | ลักษณะต้นไม้ | ตัวอย่าง Label |
| --- | --- | --- |
| ดี (Healthy) | ใบเขียวเข้ม, ลำต้นแข็งแรง | Healthy |
| ปานกลาง (Moderate) | ใบร่วงบ้าง, มีรอยเสียหาย | Moderate |
| เสื่อมโทรม (Unhealthy) | ใบร่วงเยอะ, กิ่งแห้ง | Unhealthy |
| กำลังตาย (Dying/Dead) | ไม่มีใบ, ลำต้นผุพัง | Dying |

วิธี Labeling

✅ ใช้ LabelIMG (สำหรับภาพนิ่ง)  
✅ ใช้ Roboflow (แพลตฟอร์มช่วยทำ Data Labeling)  
✅ ใช้ Python + OpenCV ทำการ Label อัตโนมัติจากสีของใบ

**3. การจัดโครงสร้าง Dataset**

ควรจัดโฟลเดอร์ข้อมูลให้เป็นระเบียบ เช่น:

bash

CopyEdit

dataset/

│── train/

│ │── healthy/ (ภาพต้นไม้ที่สมบูรณ์)

│ │── moderate/ (ภาพต้นไม้ปานกลาง)

│ │── unhealthy/ (ภาพต้นไม้เสื่อมโทรม)

│ │── dying/ (ภาพต้นไม้กำลังตาย)

│── test/

│ │── healthy/

│ │── moderate/

│ │── unhealthy/

│ │── dying/

* Train Set (80%) → ใช้สำหรับฝึกโมเดล
* Test Set (20%) → ใช้ทดสอบความแม่นยำ

**4. การเพิ่มข้อมูล (Data Augmentation)**

ถ้าข้อมูลภาพน้อย สามารถเพิ่มความหลากหลายโดยใช้ Data Augmentation เช่น:

✅ Flip & Rotation: หมุนภาพ (Rotate) หรือกลับด้าน (Flip)  
✅ Brightness & Contrast: ปรับแสงให้แตกต่างกัน  
✅ Noise: เพิ่มจุดรบกวน (Noise) เพื่อให้โมเดลแข็งแรงขึ้น  
✅ Blurring & Sharpening: ปรับความคมชัดของภาพ

ใช้ Python + OpenCV หรือ TensorFlow/Keras ทำได้ง่าย เช่น:

python

CopyEdit

import tensorflow as tf

datagen = tf.keras.preprocessing.image.ImageDataGenerator(

rotation\_range=30,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

brightness\_range=[0.5, 1.5],

horizontal\_flip=True

)

**5. การตรวจสอบคุณภาพข้อมูล (Data Validation)**

ก่อนนำข้อมูลไปฝึกโมเดล ต้องตรวจสอบ: ✅ ภาพมีคุณภาพดี (ไม่มีเบลอ)  
✅ แต่ละหมวดหมู่มีข้อมูลใกล้เคียงกัน (Balanced Data)  
✅ ไม่มีภาพซ้ำซ้อน (Duplicate Images)

ใช้ Python ช่วยตรวจสอบได้ เช่น:

python

CopyEdit

import os

from collections import Counter

dataset\_path = "dataset/train/"

categories = os.listdir(dataset\_path)

for category in categories:

count = len(os.listdir(os.path.join(dataset\_path, category)))

print(f"{category}: {count} images")

**6. การ Export เป็นไฟล์สำหรับโมเดล AI**

เมื่อจัดการข้อมูลเรียบร้อยแล้ว สามารถบันทึกในรูปแบบ CSV หรือ TFRecord เช่น:

CSV Format

css

CopyEdit

filename,label

img001.jpg,Healthy

img002.jpg,Moderate

img003.jpg,Unhealthy

img004.jpg,Dying

TFRecord (สำหรับ TensorFlow)

python

CopyEdit

import tensorflow as tf

def \_bytes\_feature(value):

return tf.train.Feature(bytes\_list=tf.train.BytesList(value=[value]))

def serialize\_example(image\_path, label):

image\_string = open(image\_path, 'rb').read()

feature = {

'image': \_bytes\_feature(image\_string),

'label': \_bytes\_feature(label.encode())

}

example\_proto = tf.train.Example(features=tf.train.Features(feature=feature))

return example\_proto.SerializeToString()

**7. นำ Dataset ไป Train โมเดล**

เมื่อ Dataset พร้อม สามารถนำไปใช้ฝึกโมเดล AI เช่น CNN (Convolutional Neural Network) ได้เลย!

ตัวอย่างการโหลด Dataset ไปใช้กับโมเดล TensorFlow:

python

CopyEdit

train\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory(

"dataset/train",

image\_size=(224, 224),

batch\_size=32

)

**สรุป**

✅ เก็บภาพต้นไม้จากโดรน/กล้อง หรือใช้ Open Data  
✅ จัดหมวดหมู่ (Labeling) เป็น Healthy, Moderate, Unhealthy, Dying  
✅ จัดโครงสร้างโฟลเดอร์ให้ง่ายต่อการใช้งาน  
✅ เพิ่มข้อมูลด้วย Data Augmentation  
✅ ตรวจสอบคุณภาพข้อมูลก่อนนำไป Train  
✅ บันทึกเป็น CSV หรือ TFRecord เพื่อใช้กับโมเดล AI

**วิธีพัฒนาโมเดล AI ด้วย Computer Vision เพื่อวิเคราะห์ความสมบูรณ์ของต้นไม้ป่าชายเลน**

การพัฒนาโมเดล AI ด้วย Computer Vision มี 5 ขั้นตอนหลัก:

1. เตรียมข้อมูล (Data Preparation)
2. สร้างและฝึกโมเดล (Model Training)
3. ทดสอบความแม่นยำของโมเดล (Model Evaluation)
4. Deploy โมเดล (Model Deployment)
5. ปรับปรุงโมเดลให้ดีขึ้น (Model Optimization)

**1. เตรียมข้อมูล (Data Preparation)**

ก่อนเทรนโมเดล เราต้องมี Dataset ที่มีภาพของต้นไม้ป่าชายเลน พร้อม Label (Healthy, Moderate, Unhealthy, Dying) ซึ่งได้จากขั้นตอนที่แล้ว

โหลด Dataset เข้า Python

python

CopyEdit

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# โหลด dataset จาก directory

train\_datagen = ImageDataGenerator(rescale=1./255, validation\_split=0.2)

train\_ds = train\_datagen.flow\_from\_directory(

'dataset/train',

target\_size=(224, 224),

batch\_size=32,

class\_mode='categorical',

subset='training'

)

val\_ds = train\_datagen.flow\_from\_directory(

'dataset/train',

target\_size=(224, 224),

batch\_size=32,

class\_mode='categorical',

subset='validation'

)

**2. สร้างและฝึกโมเดล (Model Training)**

ใช้ Convolutional Neural Network (CNN) เช่น MobileNetV2 หรือ ResNet50 ซึ่งเหมาะกับการแยกประเภทของต้นไม้

สร้างโมเดล CNN

python

CopyEdit

from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense, Flatten, Dropout

# โหลดโมเดล Pretrained (MobileNetV2)

base\_model = MobileNetV2(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

# Freeze layers ของ MobileNetV2

for layer in base\_model.layers:

layer.trainable = False

# เพิ่ม Layer เพื่อแยกประเภทต้นไม้

x = Flatten()(base\_model.output)

x = Dense(128, activation='relu')(x)

x = Dropout(0.3)(x)

x = Dense(4, activation='softmax')(x) # 4 classes: Healthy, Moderate, Unhealthy, Dying

model = Model(inputs=base\_model.input, outputs=x)

# คอมไพล์โมเดล

model.compile(optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy'])

# แสดงโครงสร้างโมเดล

model.summary()

**3. ฝึกโมเดล (Train Model)**

เมื่อสร้างโมเดลเสร็จแล้วให้ทำการฝึกโมเดล (Training) โดยใช้ข้อมูลที่เตรียมไว้

python

CopyEdit

history = model.fit(

train\_ds,

validation\_data=val\_ds,

epochs=20, # จำนวนรอบที่ให้โมเดลเรียนรู้

batch\_size=32

)

📌 หมายเหตุ:

* ถ้าข้อมูลมีจำนวนน้อย สามารถเพิ่ม Data Augmentation เช่น Flip, Rotate, Brightness
* ถ้า GPU ไม่แรงพอ แนะนำให้ใช้ Google Colab เพื่อเทรนโมเดลได้เร็วขึ้น

**4. ทดสอบความแม่นยำของโมเดล (Model Evaluation)**

หลังจากฝึกโมเดลเสร็จแล้ว ต้องทดสอบความแม่นยำกับข้อมูลที่ไม่เคยเห็นมาก่อน

python

CopyEdit

test\_loss, test\_acc = model.evaluate(val\_ds)

print(f"Test Accuracy: {test\_acc \* 100:.2f}%")

สามารถ plot กราฟดู accuracy และ loss ได้:

python

CopyEdit

import matplotlib.pyplot as plt

# Plot Accuracy

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.title('Model Accuracy')

plt.show()

**5. ทำนายผลภาพใหม่ (Prediction)**

ลองให้ AI วิเคราะห์ภาพที่ไม่เคยเห็นมาก่อน

python

CopyEdit

import numpy as np

from tensorflow.keras.preprocessing import image

# โหลดภาพที่ต้องการทดสอบ

img\_path = 'dataset/test/healthy/sample.jpg'

img = image.load\_img(img\_path, target\_size=(224, 224))

img\_array = image.img\_to\_array(img) / 255.0 # Normalize

img\_array = np.expand\_dims(img\_array, axis=0) # เพิ่ม Batch Dimension

# ให้โมเดลทำนายผล

predictions = model.predict(img\_array)

predicted\_class = np.argmax(predictions)

class\_labels = ['Healthy', 'Moderate', 'Unhealthy', 'Dying']

print(f"AI คาดการณ์ว่า: {class\_labels[predicted\_class]}")

**6. Deploy โมเดลเป็น Web API หรือ Mobile App**

เมื่อโมเดลพร้อมใช้งาน สามารถทำให้ใช้งานผ่าน Web หรือ Mobile ได้

Deploy เป็น Web API ด้วย Flask

python

CopyEdit

from flask import Flask, request, jsonify

import tensorflow as tf

import numpy as np

from tensorflow.keras.preprocessing import image

app = Flask(\_\_name\_\_)

# โหลดโมเดล

model = tf.keras.models.load\_model("mangrove\_model.h5")

# คำจำกัดความของ label

class\_labels = ['Healthy', 'Moderate', 'Unhealthy', 'Dying']

@app.route('/predict', methods=['POST'])

def predict():

file = request.files['file']

img = image.load\_img(file, target\_size=(224, 224))

img\_array = image.img\_to\_array(img) / 255.0

img\_array = np.expand\_dims(img\_array, axis=0)

predictions = model.predict(img\_array)

predicted\_class = class\_labels[np.argmax(predictions)]

return jsonify({'prediction': predicted\_class})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

* ใช้ Flask API รองรับการอัปโหลดภาพผ่านเว็บ
* ส่งภาพให้ AI วิเคราะห์ และคืนค่าประเภทของต้นไม้

**7. ปรับปรุงโมเดลให้ดีขึ้น (Model Optimization)**

หลังจาก Deploy แล้ว ต้องปรับปรุงโมเดลเพื่อเพิ่มความแม่นยำ เช่น:

✅ Fine-tune โมเดล

* Unfreeze บาง Layer ของ MobileNetV2 แล้วฝึกเพิ่ม

python

CopyEdit

for layer in base\_model.layers[-10:]:

layer.trainable = True

✅ ลองใช้โมเดลอื่นๆ เช่น ResNet50, EfficientNet

✅ ใช้ Transfer Learning กับโมเดลที่มีข้อมูลป่าชายเลน

✅ เพิ่ม Dataset โดยใช้ข้อมูลจากโดรน หรือ Satellite

**สรุปขั้นตอนการพัฒนา AI วิเคราะห์ต้นไม้ป่าชายเลน**

1. เตรียม Dataset → รวมภาพต้นไม้, Label เป็น Healthy, Moderate, Unhealthy, Dying
2. สร้างและเทรนโมเดล CNN → ใช้ MobileNetV2 หรือ ResNet50
3. ทดสอบโมเดล → เช็ค accuracy และดูผลทำนาย
4. Deploy เป็น Web API หรือ Mobile App
5. ปรับปรุงโมเดลให้ดีขึ้น → ใช้ Fine-tuning และเพิ่ม Dataset