รายละเอียดผลการทดลอง

6610450871 นายชนพัฒน์ โชติกุลรัตน์ หมู่ 200

Model = 1.CNN , 2.yolov8n-cls
Dataset = Plant (จำแนกภาพดอกพีโอนีกับดอกกุหลาบ) (Binary Classification)

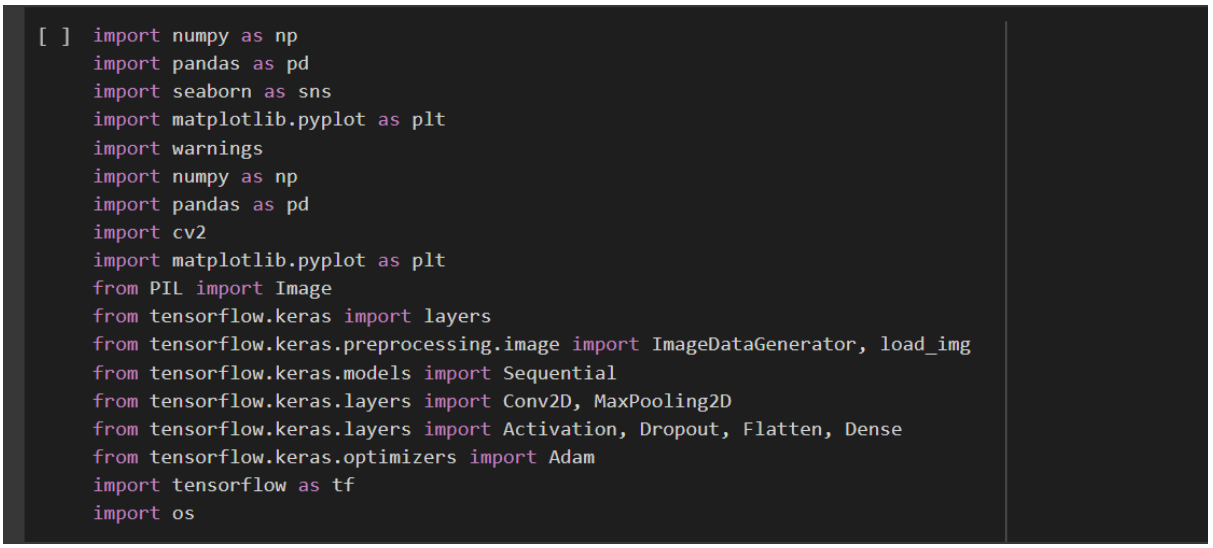
1 Model CNN

1.ดาวน์โหลดไฟล์ ZIP จาก Google Drive, แตกไฟล์ไปที่โฟลเดอร์ `/content` และลบไฟล์ ZIP ที่ดาวน์โหลดมาทิ้งหลังจากใช้งานเสร็จ.

```
import zipfile
import os
import gdown
!gdown "https://drive.google.com/uc?id=1QLWyL4x_4ijgMJNj7Qoe4hfEr3eg5BZt"

zip_filename = '/content/Plant.zip'
with zipfile.ZipFile(zip_filename, 'r') as zip_ref:
    zip_ref.extractall('/content')
os.remove(zip_filename)
```

```
Downloading...
From (original): https://drive.google.com/uc?id=1QLWyL4x_4ijgMJNj7Qoe4hfEr3eg5BZt
From (redirected): https://drive.google.com/uc?id=1QLWyL4x_4ijgMJNj7Qoe4hfEr3eg5BZt&confirm=t&uuid=3d76d066-7b2c-4a9b-984b-63607b481b2c
To: /content/Plant.zip
100% 220M/220M [00:03<00:00, 57.8MB/s]
```

2. เตรียมความพร้อมสำหรับการทำงานกับข้อมูลภาพ การสร้างและฝึกโมเดล Convolutional Neural Network (CNN) โดยใช้ไลบรารีต่างๆ



```
[ ] import numpy as np
    import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
    import warnings
    import numpy as np
    import pandas as pd
    import cv2
    import matplotlib.pyplot as plt
    from PIL import Image
    from tensorflow.keras import layers
    from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Conv2D, MaxPooling2D
    from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense
    from tensorflow.keras.optimizers import Adam
    import tensorflow as tf
    import os
```

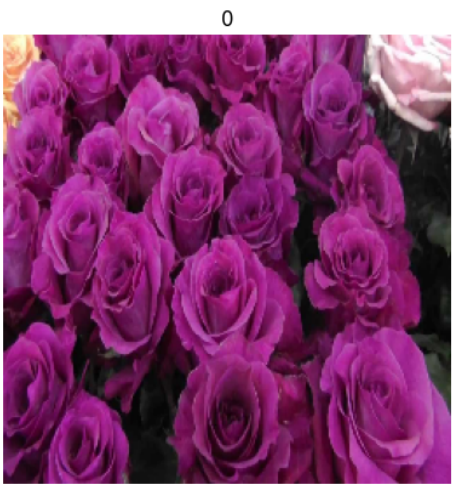
3. โหลดรูปภาพดอกไม้จาก directory ที่กำหนด กำหนด label ให้กับรูปภาพ (peonies = 1, rose = 0) และจัดเก็บข้อมูลรูปภาพและ label ไว้ใน pandas DataFrame เพื่อนำไปใช้ในการ train โมเดล Machine Learning ต่อไป โดยรูปภาพจะถูกปรับขนาดและ normalized ก่อนจัดเก็บ

```python
import os
import pandas as pd
from PIL import Image
import numpy as np

def load_images_from_paths(image_paths, target_size=(256, 256, 3)):
    images = []
    for img_path in image_paths:
        img = image.load_img(img_path, target_size=target_size)
        img_array = image.img_to_array(img)
        img_array = img_array / 255.0
        images.append(img_array)
    return np.array(images)

def load_and_label_images(dataset_dir, image_size=(256, 256,3)):
    image_list = []
    label_list = []

    for img_name in os.listdir(dataset_dir):
        img_path = os.path.join(dataset_dir, img_name)

        if "peonies" in img_name.lower():
            image_list.append(img_path)
            label_list.append(1)  # Peonies label
        elif "rose" in img_name.lower():
            image_list.append(img_path)
            label_list.append(0)  # Rose label

    return image_list, label_list
```

```python
def create_dataframe(image_list, label_list):
    data = {'image': image_list, 'label': label_list}
    df = pd.DataFrame(data)
    return df

dataset = "/content/Plant"
image_size=(256, 256,3)

image_list, label_list = load_and_label_images(dataset, image_size)

df = create_dataframe(image_list, label_list)

print(df.head())
```

```
                                   image  label
0  /content/Plant/garden_roses_00039.jpg      0
1       /content/Plant/peonies_00013.jpg      1
2  /content/Plant/garden_roses_00016.jpg      0
3       /content/Plant/peonies_00052.jpg      1
4       /content/Plant/peonies_00057.jpg      1
```

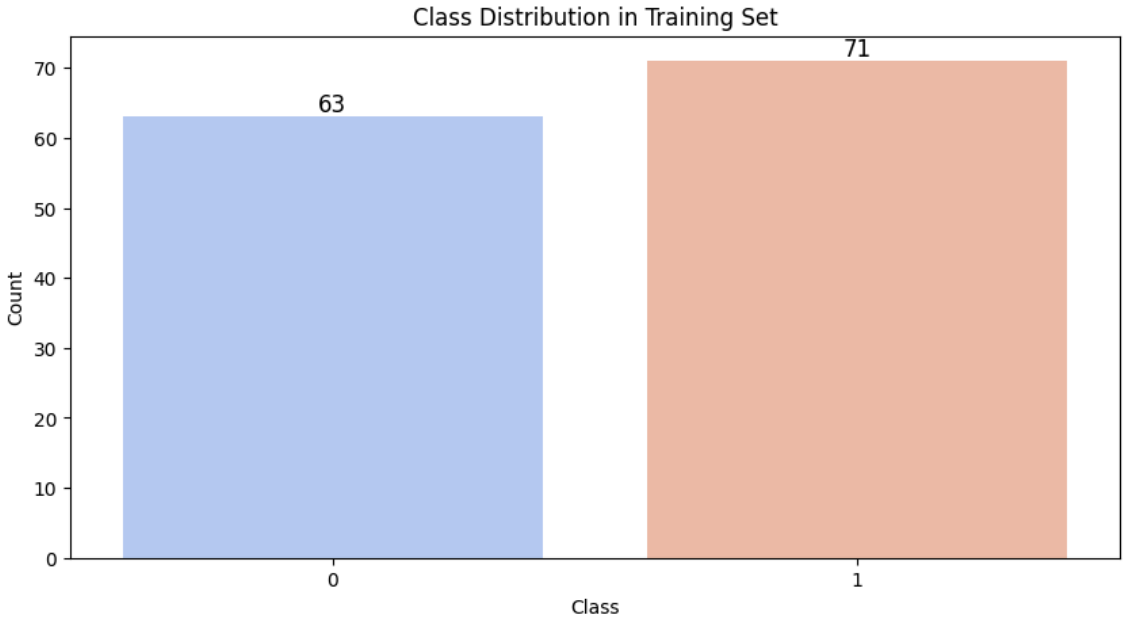
4 . display ภาพของทั้ง 2 class



5. แบ่ง train, val โดยให้ val size = 0.1 , random_state = 1

```
import numpy as np
from sklearn.model_selection import train_test_split

x = df.image
y = df.label
x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=0.1, random_state=1)
```

6.show class distribution in training set



7 . สร้าง CNN ที่มี 2 convolutional block (แต่ละ block มี Conv2D, MaxPool2D และ Dropout) ตามด้วย fully connected layer โดย layer สุดท้ายใช้ sigmoid activation สำหรับ binary classification โมเดลนี้คอมไพล์ด้วย Adam optimizer และ binary cross-entropy loss function และวัดผลด้วย accuracy

```python
from sklearn.metrics import confusion_matrix
import itertools

from keras.utils import to_categorical # convert to one-hot-encoding
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import RMSprop,Adam
from keras.callbacks import ReduceLROnPlateau

model = Sequential()
#
model.add(Conv2D(filters = 8, kernel_size = (5,5),padding = 'Same',
                 activation ='relu', input_shape = (256,256,3)))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))
#
model.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.25))
# fully connected
model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(1, activation = "sigmoid"))

model.compile(Adam() , loss = "binary_crossentropy", metrics=["accuracy"])
model.summary()
```

Model: "sequential_7"

Layer (type)	Output Shape	Param #
conv2d_16 (Conv2D)	(None, 256, 256, 8)	608
max_pooling2d_16 (MaxPooling2D)	(None, 128, 128, 8)	0
dropout_23 (Dropout)	(None, 128, 128, 8)	0
conv2d_17 (Conv2D)	(None, 128, 128, 16)	1,168
max_pooling2d_17 (MaxPooling2D)	(None, 64, 64, 16)	0
dropout_24 (Dropout)	(None, 64, 64, 16)	0
flatten_7 (Flatten)	(None, 65536)	0
dense_14 (Dense)	(None, 256)	16,777,472
dropout_25 (Dropout)	(None, 256)	0
dense_15 (Dense)	(None, 1)	257

Total params: 16,779,505 (64.01 MB)
Trainable params: 16,779,505 (64.01 MB)
Non-trainable params: 0 (0.00 B)

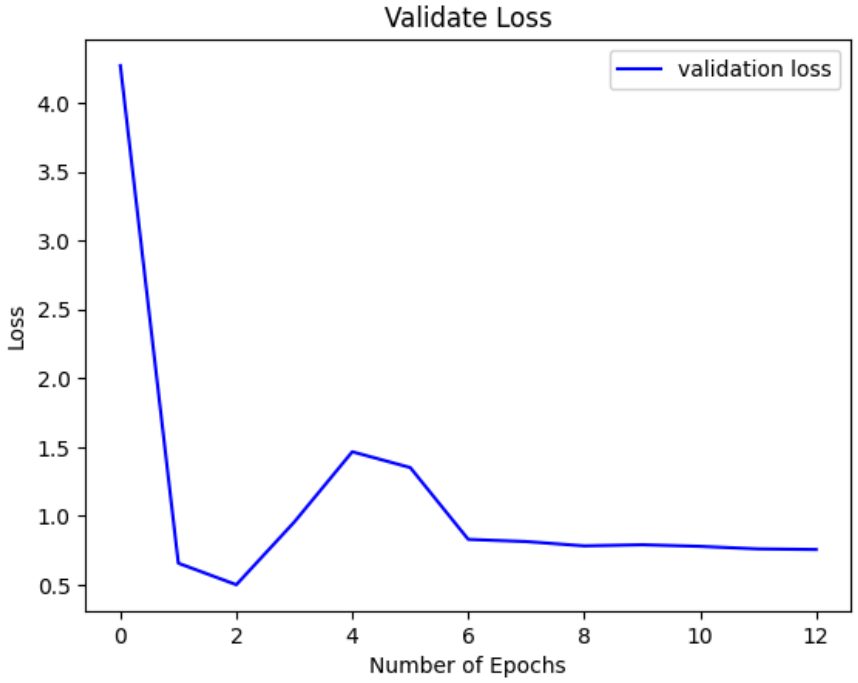
8.เตรียมความพร้อมสำหรับการ train โมเดล โดยมีการจัดการกับ imbalanced dataset ด้วย class weights เพิ่มความหลากหลายของข้อมูลด้วย image augmentation และตั้งค่า callbacks สำหรับควบคุมการ train เช่น การหยุด train เมื่อ overfitting และการปรับ learning rate ซึ่งจะช่วยให้โมเดลมีประสิทธิภาพที่ดีขึ้น

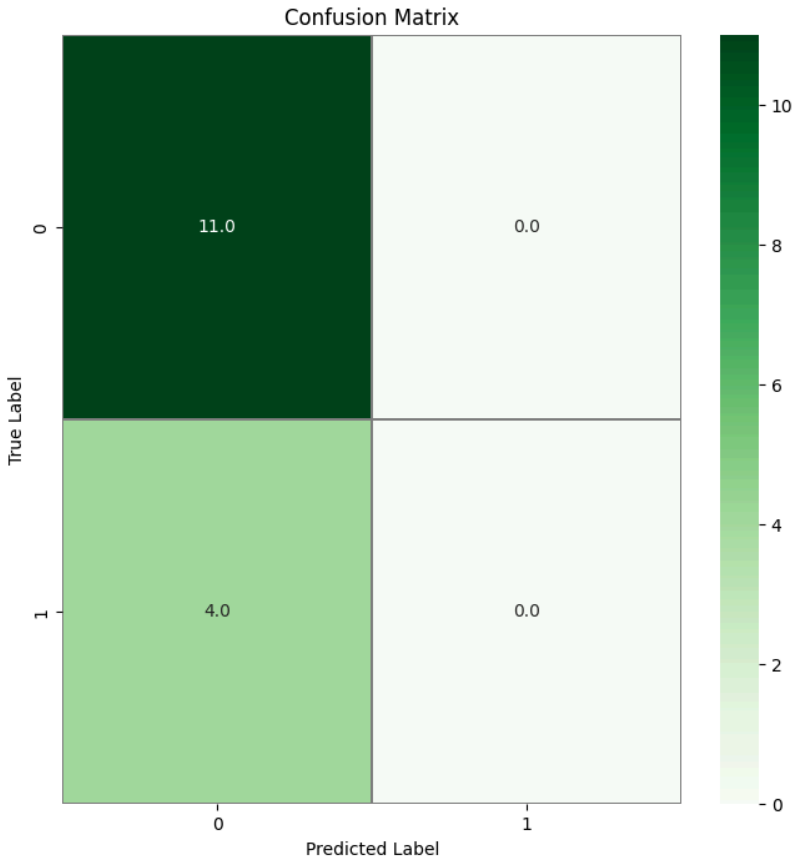
```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.utils.class_weight import compute_class_weight

class_weights = compute_class_weight(class_weight="balanced", classes=np.unique(y_train), y=y_train)
class_weight_dict = dict(enumerate(class_weights))

datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',

)

callbacks = [
    EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True),
    ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5, min_lr=1e-6)
]

epochs = 10
batch_size = 32
```

9. นำข้อมูลรูปภาพเข้าสู่กระบวนการ training โดยใช้ ImageDataGenerator ช่วยในการ augment ข้อมูล และใช้ flow ในการสร้าง batch มีการกำหนด steps_per_epoch และ validation_steps เพื่อควบคุมการ training และใช้ callbacks เพื่อ monitor การ training และปรับ learning rate ผลลัพธ์ของการ train จะถูกเก็บไว้ใน history ซึ่งสามารถนำมาวิเคราะห์เพื่อดูประสิทธิภาพของโมเดล เช่น loss และ accuracy

```python
x_train_images = load_images_from_paths(x_train,(256,256,3))
x_val_images = load_images_from_paths(x_val,(256,256,3))
datagen.fit(x_train_images)
train_generator = datagen.flow(x_train_images, y_train, batch_size=batch_size)
val_generator = datagen.flow(x_val_images, y_val, batch_size=batch_size)

history = model.fit(
    train_generator,
    epochs=50,  # Increase epochs for better training
    validation_data=val_generator,
    steps_per_epoch=len(x_train_images) // batch_size,
    validation_steps=len(x_val_images) // batch_size,
    callbacks=callbacks  # Attach callbacks
)
```

10. plot validate loss และ cunfusion_matrix ได้ผลลัพธ์ ดังนี้





2 Model yolov8n-cls

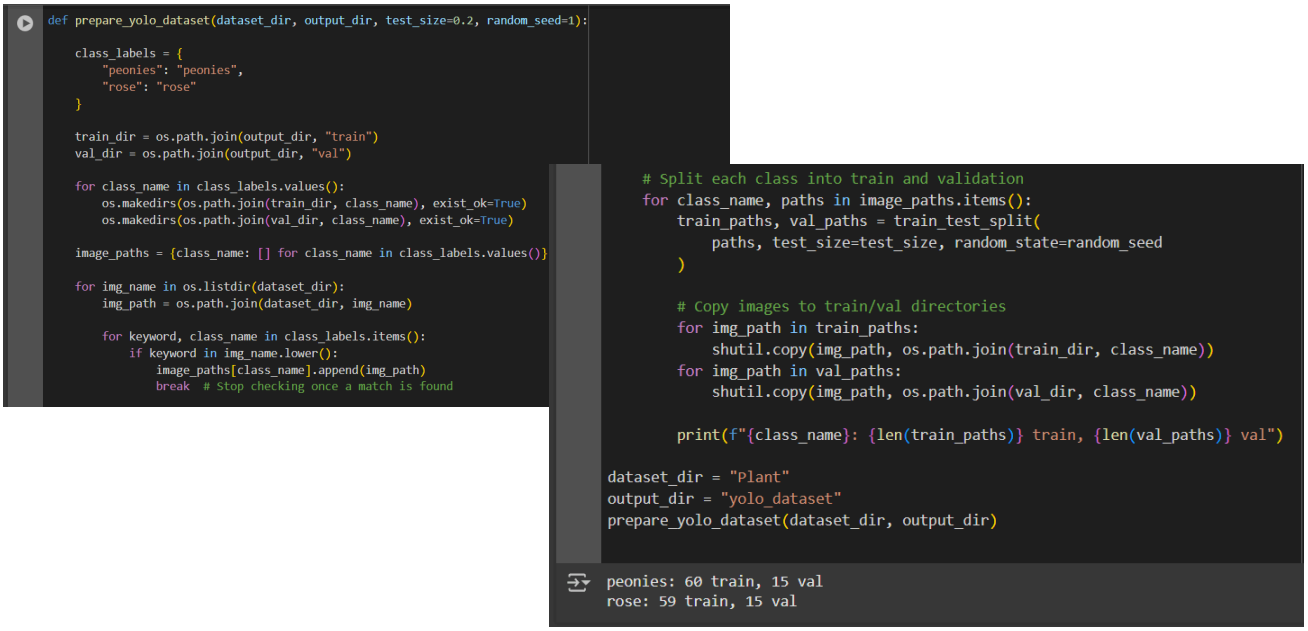
1.ดาวน์โหลดไฟล์ ZIP จาก Google Drive, แตกไฟล์ไปที่โฟลเดอร์ `/content` และลบไฟล์ ZIP ที่ดาวน์โหลดมาทิ้งหลังจากใช้งานเสร็จ.

```
import zipfile
import os
import gdown
!gdown "https://drive.google.com/uc?id=1QLWyL4x_4ijgMJNj7Qoe4hfEr3eg5BZt"

zip_filename = '/content/Plant.zip'
with zipfile.ZipFile(zip_filename, 'r') as zip_ref:
    zip_ref.extractall('/content')
os.remove(zip_filename)
```

```
Downloading...
From (original): https://drive.google.com/uc?id=1QLWyL4x_4ijgMJNj7Qoe4hfEr3eg5BZt
From (redirected): https://drive.google.com/uc?id=1QLWyL4x_4ijgMJNj7Qoe4hfEr3eg5BZt&confirm=t&uuid=3d76d066-7b2c-4a9b-984b-63607b481b2c
To: /content/Plant.zip
100% 220M/220M [00:03<00:00, 57.8MB/s]
```

2. install yolo library, และ import library ต่างๆที่ต้องใช้

3. จัดเรียงรูปภาพดอกไม้ peonies และ rose โดยแบ่งเป็น train และ validation set และจัดเก็บใน directory ที่เหมาะสม เพื่อให้ง่ายต่อการนำไปใช้เทรนโมเดล YOLO โดยรูปภาพจะถูกคัดลอกไปยัง directory ใหม่ และไม่มีการเปลี่ยนแปลงชื่อไฟล์ ทำให้ง่ายต่อการจัดการและติดตาม นอกจากนี้ ยังมีการพิมพ์จำนวนรูปภาพในแต่ละ set เพื่อให้ผู้ใช้ทราบจำนวนข้อมูลที่ใช้ในการเทรนและประเมินผลโมเดล



```
def prepare_yolo_dataset(dataset_dir, output_dir, test_size=0.2, random_seed=1):

    class_labels = {
        "peonies": "peonies",
        "rose": "rose"
    }

    train_dir = os.path.join(output_dir, "train")
    val_dir = os.path.join(output_dir, "val")

    for class_name in class_labels.values():
        os.makedirs(os.path.join(train_dir, class_name), exist_ok=True)
        os.makedirs(os.path.join(val_dir, class_name), exist_ok=True)

    image_paths = {class_name: [] for class_name in class_labels.values()}

    for img_name in os.listdir(dataset_dir):
        img_path = os.path.join(dataset_dir, img_name)

        for keyword, class_name in class_labels.items():
            if keyword in img_name.lower():
                image_paths[class_name].append(img_path)
                break  # Stop checking once a match is found

    # Split each class into train and validation
    for class_name, paths in image_paths.items():
        train_paths, val_paths = train_test_split(
            paths, test_size=test_size, random_state=random_seed
        )

        # Copy images to train/val directories
        for img_path in train_paths:
            shutil.copy(img_path, os.path.join(train_dir, class_name))
        for img_path in val_paths:
            shutil.copy(img_path, os.path.join(val_dir, class_name))

        print(f"{class_name}: {len(train_paths)} train, {len(val_paths)} val")

dataset_dir = "Plant"
output_dir = "yolo_dataset"
prepare_yolo_dataset(dataset_dir, output_dir)
```

```
peonies: 60 train, 15 val
rose: 59 train, 15 val
```

4. display ภาพของทั้ง 2 class

```
[ ] def load_images_from_paths(image_paths, target_size=(256, 256, 3)):
        images = []
        for img_path in image_paths:
            img = image.load_img(img_path, target_size=target_size)
            img_array = image.img_to_array(img)
            img_array = img_array / 255.0
            images.append(img_array)
        return np.array(images)
```

```
[ ] import matplotlib.pyplot as plt
    from keras.preprocessing import image

    dataset_path = "/content/yolo_dataset/train/peonies"

    img_filename = os.listdir(dataset_path)[0]
    img_path = os.path.join(dataset_path, img_filename)
    img = image.load_img(img_path, target_size=(256, 256,3))
    img_array = image.img_to_array(img)

    plt.imshow(img_array.astype('uint8'))
    plt.title(1)  # Display label as title
    plt.axis("off")
    plt.show()
```

1

0

5. เริ่มต้นด้วยการตรวจสอบว่าระบบมี GPU ที่รองรับ CUDA หรือไม่ ถ้ามี ก็จะแสดงชื่อ GPU จากนั้น จะนำเข้าไลบรารีที่จำเป็นสำหรับการทำงานกับโมเดล YOLO ได้แก่ Ultralytics (YOLO), Pillow (รูปภาพ), และ Matplotlib (แสดงผล) โค้ดนี้เป็นส่วนหนึ่งของการเตรียมความพร้อมก่อนที่จะเริ่มใช้งานโมเดล YOLO

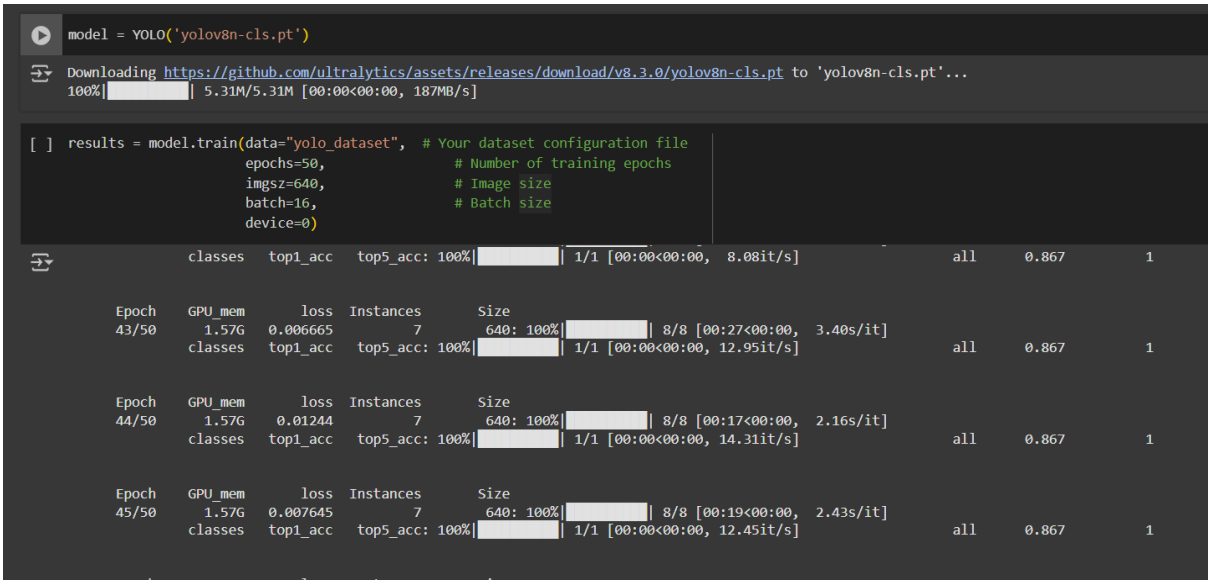
```
import torch

print("Is CUDA available:", torch.cuda.is_available())
print("GPU device:", torch.cuda.get_device_name(0) if torch.cuda.is_available() else "None")
```

```
Is CUDA available: True
GPU device: Tesla T4
```

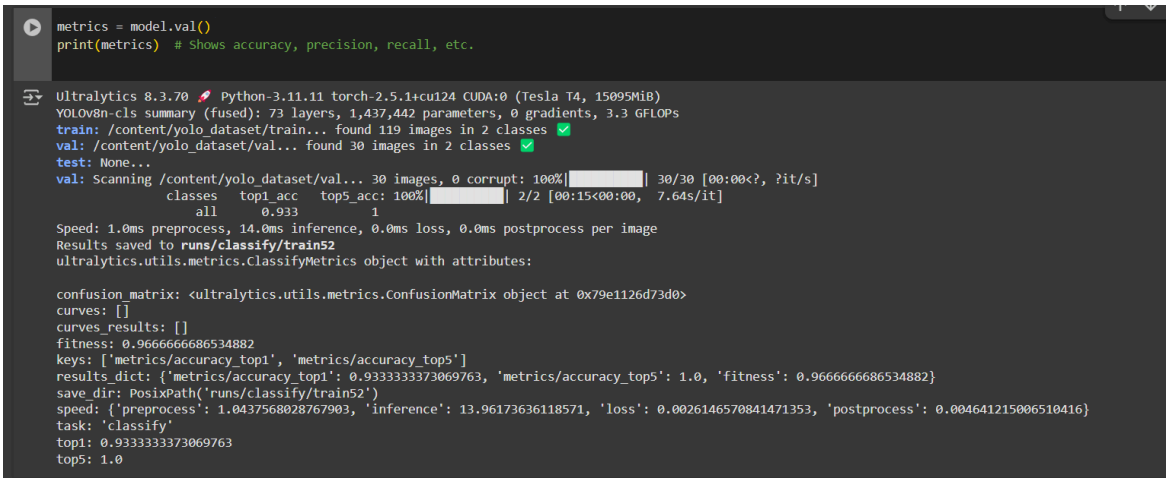
```
[ ] from ultralytics import YOLO
    from PIL import Image
    import matplotlib.pyplot as plt
```

```
Creating new Ultralytics Settings v0.0.6 file ✅
View Ultralytics Settings with 'yolo settings' or at '/root/.config/Ultralytics/settings.json'
Update Settings with 'yolo settings key=value', i.e. 'yolo settings runs_dir=path/to/dir'. For help see https://docs.ultralyt
```

6. โหลดโมเดล YOLOv8n-cls ที่เทรนไว้แล้ว (หรือถ้ายังไม่มี ก็อาจจะดาวน์โหลดหรือเทรนเอง) จากนั้น จะทำการเทรนโมเดลเพิ่มเติมโดยใช้ dataset ที่กำหนดไว้ใน `yolo_dataset` โดยมีการตั้งค่าต่างๆ เช่น จำนวน epoch ขนาดรูปภาพ batch size และอุปกรณ์ที่ใช้ในการเทรน ผลลัพธ์ของการเทรน เช่น ค่า loss ค่า metrics ต่างๆ จะถูกเก็บไว้ในตัวแปร `results` ซึ่งสามารถนำมาใช้เพื่อวิเคราะห์ผลการเทรน หรือนำโมเดลที่เทรนแล้วไปใช้งานต่อไปได้



7. ผลลัพธ์แสดงให้เห็นว่าโมเดล YOLOv8n-cls ที่เทรนมา มี Top-1 accuracy 93.3% และ Top-5 accuracy 100% บนข้อมูล validation ซึ่งถือว่ามีประสิทธิภาพที่ดี นอกจากนี้ ยังแสดงข้อมูลอื่นๆ เช่น ความเร็วในการประมวลผล confusion matrix และค่า fitness ซึ่งเป็นประโยชน์ในการวิเคราะห์และปรับปรุงโมเดลต่อไป



8. ทดลอง predict รูปได้

```
# Run prediction for the 'peonies' folder
results = model.predict("/content/yolo_dataset/val/peonies")

print(results)
```

```
image 1/15 /content/yolo_dataset/val/peonies/peonies_00013.jpg: 640x640 peonies 1.00, rose 0.00, 4.6ms
image 2/15 /content/yolo_dataset/val/peonies/peonies_00015.jpg: 640x640 peonies 1.00, rose 0.00, 4.8ms
image 3/15 /content/yolo_dataset/val/peonies/peonies_00020.jpg: 640x640 peonies 1.00, rose 0.00, 6.2ms
image 4/15 /content/yolo_dataset/val/peonies/peonies_00028.jpg: 640x640 peonies 0.83, rose 0.17, 4.4ms
image 5/15 /content/yolo_dataset/val/peonies/peonies_00032.jpg: 640x640 peonies 1.00, rose 0.00, 14.7ms
image 6/15 /content/yolo_dataset/val/peonies/peonies_00034.jpg: 640x640 peonies 1.00, rose 0.00, 16.7ms
image 7/15 /content/yolo_dataset/val/peonies/peonies_00037.jpg: 640x640 peonies 1.00, rose 0.00, 6.9ms
image 8/15 /content/yolo_dataset/val/peonies/peonies_00044.jpg: 640x640 peonies 1.00, rose 0.00, 13.5ms
image 9/15 /content/yolo_dataset/val/peonies/peonies_00046.jpg: 640x640 peonies 1.00, rose 0.00, 5.5ms
image 10/15 /content/yolo_dataset/val/peonies/peonies_00049.jpg: 640x640 peonies 1.00, rose 0.00, 6.0ms
image 11/15 /content/yolo_dataset/val/peonies/peonies_00053.jpg: 640x640 peonies 1.00, rose 0.00, 5.3ms
image 12/15 /content/yolo_dataset/val/peonies/peonies_00060.jpg: 640x640 rose 0.99, peonies 0.01, 3.5ms
image 13/15 /content/yolo_dataset/val/peonies/peonies_00063.jpg: 640x640 peonies 1.00, rose 0.00, 3.5ms
image 14/15 /content/yolo_dataset/val/peonies/peonies_00072.jpg: 640x640 peonies 1.00, rose 0.00, 3.5ms
image 15/15 /content/yolo_dataset/val/peonies/peonies_00080.jpg: 640x640 peonies 0.69, rose 0.31, 3.5ms
Speed: 62.4ms preprocess, 6.9ms inference, 0.1ms postprocess per image at shape (1, 3, 640, 640)
[ultralytics.engine.results.Results object with attributes:
```

predict ถูกต้อง 14 จาก 15 ภาพ (Peonies)

```
[ ] # Run prediction for the 'peonies' folder
    results = model.predict("/content/yolo_dataset/val/rose")

    print(results)
```

```
image 1/15 /content/yolo_dataset/val/rose/garden_roses_00003.jpg: 640x640 rose 0.99, peonies 0.01, 6.0ms
image 2/15 /content/yolo_dataset/val/rose/garden_roses_00006.jpg: 640x640 rose 0.90, peonies 0.10, 5.9ms
image 3/15 /content/yolo_dataset/val/rose/garden_roses_00011.jpg: 640x640 rose 1.00, peonies 0.00, 3.8ms
image 4/15 /content/yolo_dataset/val/rose/garden_roses_00015.jpg: 640x640 rose 0.99, peonies 0.01, 3.8ms
image 5/15 /content/yolo_dataset/val/rose/garden_roses_00016.jpg: 640x640 rose 1.00, peonies 0.00, 3.5ms
image 6/15 /content/yolo_dataset/val/rose/garden_roses_00017.jpg: 640x640 rose 0.87, peonies 0.13, 3.5ms
image 7/15 /content/yolo_dataset/val/rose/garden_roses_00019.jpg: 640x640 rose 1.00, peonies 0.00, 3.9ms
image 8/15 /content/yolo_dataset/val/rose/garden_roses_00034.jpg: 640x640 peonies 1.00, rose 0.00, 3.6ms
image 9/15 /content/yolo_dataset/val/rose/garden_roses_00038.jpg: 640x640 rose 1.00, peonies 0.00, 3.5ms
image 10/15 /content/yolo_dataset/val/rose/garden_roses_00040.jpg: 640x640 rose 1.00, peonies 0.00, 3.5ms
image 11/15 /content/yolo_dataset/val/rose/garden_roses_00041.jpg: 640x640 rose 1.00, peonies 0.00, 3.5ms
image 12/15 /content/yolo_dataset/val/rose/garden_roses_00061.jpg: 640x640 rose 1.00, peonies 0.00, 3.5ms
image 13/15 /content/yolo_dataset/val/rose/garden_roses_00075.jpg: 640x640 rose 0.74, peonies 0.26, 3.5ms
image 14/15 /content/yolo_dataset/val/rose/garden_roses_00083.jpg: 640x640 rose 1.00, peonies 0.00, 3.6ms
image 15/15 /content/yolo_dataset/val/rose/garden_roses_00085.jpg: 640x640 rose 1.00, peonies 0.00, 3.5ms
Speed: 455.6ms preprocess, 3.9ms inference, 0.1ms postprocess per image at shape (1, 3, 640, 640)
[ultralytics.engine.results.Results object with attributes:
```

predict ถูกต้อง 14 จาก 15 ภาพ (Rose)

สรุป

จากการเปรียบเทียบผลลัพธ์ของโมเดล CNN (Convolutional Neural Network) และ YOLOv8n-cls (You Only Look Once version 8 nano-classification) พบว่าโมเดล YOLOv8n-cls ให้ผลลัพธ์ที่ดีกว่า CNN

เหตุผลที่ YOLOv8n-cls ให้ผลลัพธ์ที่ดีกว่า

- **สถาปัตยกรรมโมเดล**: YOLOv8n-cls เป็นโมเดลตรวจจับวัตถุแบบ Single-stage ซึ่งออกแบบมาเพื่อประมวลผลภาพทั้งภาพในครั้งเดียว ทำให้สามารถตรวจจับวัตถุได้รวดเร็วและแม่นยำกว่า CNN ซึ่งมักใช้การเลื่อนหน้าต่าง (sliding window) เพื่อวิเคราะห์ภาพทีละส่วน
- **การเรียนรู้ลักษณะเด่น**: YOLOv8n-cls ได้รับการออกแบบมาเพื่อเรียนรู้ลักษณะเด่นของวัตถุโดยตรงจากภาพ ทำให้สามารถจำแนกประเภทของวัตถุได้แม่นยำกว่า CNN ซึ่งอาจต้องใช้กระบวนการ Feature extraction ที่ซับซ้อนกว่า
- **การปรับแต่งโมเดล**: YOLOv8n-cls มีพารามิเตอร์ที่สามารถปรับแต่งได้หลากหลาย ทำให้สามารถปรับโมเดลให้เหมาะสมกับงานเฉพาะได้ง่ายกว่า CNN
- **ความเร็วในการประมวลผล**: YOLOv8n-cls มีความเร็วในการประมวลผลที่สูงกว่า CNN มาก ทำให้เหมาะสำหรับงานที่ต้องการการประมวลผลแบบเรียลไทม์