**โครงงานศึกษา**

**เรื่อง**

**การจำแนกประเภทดอกไม้จากภาพโดยใช้แบบจำลอง VGG16**

**(Flower Image Classification with VGG16)**

**เสนอ**

**รศ. ดร. อรฉัตร จิตต์โสภักตร์**

**จัดทำโดย**

**ร.ท.หญิง จิตรา เรืองรุ่งโสม รหัสนักศึกษา 62601144**

**โครงงานนี้เป็นส่วนหนึ่งของวิชา เรื่องคัดสรรทางด้านการประมวลผลภาพดิจิตอล (01197704)**

**หลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมป้องกันประเทศ**

**คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง**

**ภาคการศึกษาที่ 2 ปีการศึกษา 2562**

**สารบัญ**

[**บทที่ 1 บทนำ** 1](#_Toc39476961)

[1.1 ความเป็นมาของโครงงาน 1](#_Toc39476962)

[1.2 วัตถุประสงค์ของโครงงาน 1](#_Toc39476963)

[1.3 ขอบเขตการศึกษา 1](#_Toc39476964)

[**บทที่ 2 ทบทวนวรรณกรรม** 2](#_Toc39476965)

[2.1 โครงข่ายประสาทเทียมแบบสังวัตนาการ (Convolutional Neural Network: CNN) 2](#_Toc39476966)

[2.2 การถ่ายโอนเรียนรู้ (Transfer Learning) และ Data Augmentation 3](#_Toc39476967)

[2.3 แบบจำลอง VGG16 3](#_Toc39476968)

[2.4 ตัวอย่างงานวิจัยที่เกี่ยวข้อง 3](#_Toc39476969)

[**บทที่ 3 วิธีดำเนินการ** 4](#_Toc39476970)

[3.1 ชุดข้อมูล Flowers Recongnition 4](#_Toc39476971)

[3.2 เครื่องมือที่ใช้งาน 4](#_Toc39476972)

[3.3 เกณฑ์ในการประเมิน 4](#_Toc39476973)

[3.4 วิธีดำเนินการ 4](#_Toc39476974)

[**บทที่ 4 ผลการศึกษา** 7](#_Toc39476975)

[**บทที่ 5 ผลสรุป** 15](#_Toc39476976)

[5.1 สรุปและอภิปรายผลการศึกษา 15](#_Toc39476977)

[5.2 ประโยชน์ที่ได้รับ 15](#_Toc39476978)

[5.3 ข้อเสนอแนะ 15](#_Toc39476979)

[5.4 บรรณานุกรม 15](#_Toc39476980)

# บทที่ 1

**บทนำ**

## 1.1 ความเป็นมาของโครงงาน

ดอกไม้เป็นพืชที่พบได้ทั่วไปในทุกหนทุกแห่ง มีประโยชน์ในหลายๆด้าน เช่น เป็นอาหารให้สัตว์ ใช้เป็นสมุนไพร ใช้ในการตกแต่ง เป็นต้น ขึ้นอยู่กับชนิดและสายพันธุ์ของดอกไม้ที่นำไปใช้ การระบุชนิดของดอกไม้จึงถือเป็นงานหนึ่งที่ได้รับความสนใจ โดยเฉพาะการระบุชนิดดอกไม้ได้อัตโนมัติจากภาพ โดยผู้ใช้ไม่จำเป็นต้องมีความรู้หรือรู้จักดอกไม้ประเภทนั้นมาก่อน เนื่องจากดอกไม้นั้นมีหลายประเภท และในการจำแนกดอกไม้บางประเภทนั้นอาจต้องอาศัยผู้เชี่ยวชาญในการจำแนก การพัฒนาระบบรู้จำดอกไม้โดยใช้คอมพิวเตอร์จึงได้ถูกพัฒนาขึ้นเพื่อเป็นประโยชน์ในจำแนกชนิดดอกไม้เพื่อการนำไปใช้ในงานด้านต่างๆ ต่อไป

การจำแนกข้อมูล (Classification) เป็นการวิเคราะห์ข้อมูลรูปแบบหนึ่ง ที่ทำการสร้างแบบจำลองหรือตัวจำแนกข้อมูล (model/classifier) เพื่อทำนายหมวดหมู่ของข้อมูล (category/class) ซึ่งในปัจจุบัน การจำแนกข้อมูลจากรูปภาพนิยมทำด้วยวิธีที่เรียกว่า “โครงข่ายประสาทเทียมแบบสังวัตนาการ” (Convolutional Neural Network: CNN) ซึ่งเป็นหนึ่งในวิธีการเรียนรู้เครื่องจักรที่มนุษย์ไม่จำเป็นต้องสกัดคุณลักษณะข้อมูลเองดังเช่นที่ผ่านมา แต่สามารถใช้คอมพิวเตอร์สกัดคุณลักษณะและนำเข้าพร้อมทั้งจำแนกได้อย่างอัตโนมัติ ด้วยความสะดวกแต่ให้ผลลัพธ์ที่ดี วิธีนี้จึงเป็นที่นิยมในการนำมาใช้งานที่หลากหลาย รวมถึงงานจำแนกดอกไม้

โครงงานนี้ จึงนำเสนอการใช้ CNN มาใช้ในการสร้างแบบจำลองเพื่อจำแนกภาพดอกไม้ 5 ชนิด โดยใช้วิธีถ่ายโอนเรียนรู้ (Transfer Learning) และใช้แบบจำลอง VGG16 เป็นแบบจำลองฐาน โดยเป็นการศึกษาโค้ดตัวอย่างจาก [1] โครงงานแสดงวิธีการนำเข้าข้อมูล การใช้งานแบบจำลองเพื่อฝึกฝนการรู้จำข้อมูล และประสิทธิภาพความแม่นยำในการทำนายภาพ

## 1.2 วัตถุประสงค์ของโครงงาน

เพื่อศึกษาและสร้างตัวจำแนกดอกไม้ 5 ชนิด

## 1.3 ขอบเขตการศึกษา

ศึกษาการจำแนกดอกไม้จากโค้ดตัวอย่าง [1] และบางส่วนจาก [2] โดยใช้แบบจำลอง VGG16 เป็น pre-trained model บนชุดข้อมูล Flowers Recoginition [3]

# บทที่ 2

**ทบทวนวรรณกรรม**

## 2.1 โครงข่ายประสาทเทียมแบบสังวัตนาการ (Convolutional Neural Network: CNN)

CNN เป็นโครงข่ายประสาทเทียมที่จำลองการมองเห็นของมนุษย์ที่มองพื้นที่เป็นที่ย่อยๆ และนำกลุ่มของพื้นที่ย่อยๆมาผสานกัน เพื่อดูว่าสิ่งที่เห็นอยู่เป็นอะไรกันแน่ การมองพื้นที่ย่อยของมนุษย์จะมีการแยกคุณลักษณะ (feature) ของพื้นที่ย่อยนั้น เช่น ลายเส้น และการตัดกันของสี ซึ่งการที่มนุษย์รู้ว่าพื้นที่ตรงนี้เป็นเส้นตรงหรือสีตัดกัน เพราะมนุษย์ดูทั้งจุดที่สนใจและบริเวณรอบๆ ประกอบกัน

การแยกคุณลักษณะจากภาพของ CNN ใช้หลักการเดียวกันกับคอนโวลูชันเชิงพื้นที่ (Spatial Convolution) กล่าวคือเป็นการคำนวณจากการนำตัวกรอง (filter หรือ kernel) ของคุณลักษณะหนึ่งๆ ที่ต้องการไปทาบบนพิกเซลของภาพแล้วทำการดอทเมตริกซ์เพื่อดึงค่าคุณลักษณะที่สนใจ เลื่อนตัวกรองและทำซ้ำไปเรื่อยๆ จนครบทุกพิกเซล ก็จะได้ผังคุณลักษณะ (feature map) ออกมา

โครงสร้างของ CNN มีลักษณะเป็นชั้นโครงข่ายซ้อนกัน โดยมีองค์ประกอบหลักๆ 5 อย่าง คือ



รูปที่ 1 ภาพรวมโครงสร้างของ CNN

1. ชั้น Convolution ทำการกรองโดยการดอทให้เป็นผังคุณลักษณะ สามารถนำหลายตัวกรองมาผ่านข้อมูลเพื่อกรองข้อมูลที่มีคุณลักษณะต่างกัน
2. ชั้น Activation เป็นฟังก์ชันการตัดสินใจในการส่งต่อผลลัพธ์ของแต่ละนิวรอน ReLU เป็นฟังก์ชัน non-linear activation แบบหนึ่งที่นิยมใช้
3. ชั้น Pooling ใช้เพื่อลดขนาดมิติข้อมูลเพื่อการคำนวณที่เร็วขึ้น ตัวที่นิยมใช้คือ Max Pooling ตัวกรองที่ใช้ในการเลือกค่าที่มากที่สุดในบริเวณเมทริกซ์เดียวกันของแต่ละเมทริกซ์เพื่อดึงคุณลักษณะเด่นออกมา
4. ชั้น Flatten ทำให้ feature map ที่เป็นเมตริกซ์หลายมิติ ให้เหลือมิติเดียว เพื่อสะดวกในการวิเคราะห์ข้อมูล
5. ชั้น Fully-Connected เป็นตัวตัดสินใจความน่าจะเป็นของ feature จากชั้นก่อนว่าตรงกับคลาสใดและส่งต่อค่าน้ำหนักไปยังชั้นถัดไปด้วย จนได้ผลลัพธ์เป็นคลาสในชั้นสุดท้าย ฟังก์ชันที่นิยมใช้ในชั้นสุดท้ายคือ softmax activation จะให้ผลลัพธ์ความน่าจะเป็นของแต่ละคลาสรวมกันทุกคลาสในรูปค่าความน่าจะเป็นไม่เกิน 1

## 2.2 การถ่ายโอนเรียนรู้ (Transfer Learning) และ Data Augmentation

แม้ว่า CNN จะสามารถทำงานได้ดีกว่าการสกัดคุณลักษณะจากรูปภาพแบบดั้งเดิม แต่ยังมีข้อจำกัดบางประการ คือ การฝึกแบบจำลองให้ได้ความแม่นยำสูง ต้องใช้ข้อมูลฝึกขนาดใหญ่ ยิ่งมีจำนวนมาก เครื่องก็จะสามารถเรียนรู้ได้มากและทำนายได้แม่นยำขึ้น และชั้นโครงข่ายที่ซับซ้อนก็จะใช้เวลาในการฝึกนานยิ่งขึ้นด้วย ซึ่งได้มีการนำเสนอวิธีรับมือกับปัญหานี้ 2 แนวทาง คือ 1) การทำ Data Augmentation เพื่อเพิ่มจำนวนรูปภาพ กล่าวคือ เป็นการทำซ้ำรูปโดยปรับให้แตกต่างจากเดิม เช่น การหมุนรูปในองศาอื่น การตัดรูป การปรับแสงสว่าง เป็นต้น 2) การทำ Transfer Learning คือการใช้แบบจำลองที่มีการปรับค่าน้ำหนักจากการฝึกกับชุดข้อมูลขนาดใหญ่มาแล้ว มีค่าน้ำหนักที่คลอบคลุมวัตถุทั่วไปหลากหลายชนิด แล้วสามารถนำมาปรับใช้กับงานเฉพาะทางโดยไม่ต้องทำการฝึกข้อมูลใหม่ทั้งหมด ซึ่งปัจจุบัน มีงานวิจัยที่พัฒนาโครงสร้างแบบจำลอง CNN ที่สามารถนำมาใช้เป็นแบบจำลองฐานในการทำ transfer learning มากมาย

## 2.3 แบบจำลอง VGG16

VGGNet (2014) เป็นแบบจำลองหนึ่งที่นิยมใช้เป็น pre-trained model เนื่องจากเป็นโครงสร้างทีไม่ซับซ้อน แต่ได้ผลลลัพธ์ความแม่นยำที่ค่อนข้างดี แบบจำลองที่มีจำนวนชั้นมากขึ้น และลดขนาดของตัวกรองลง โดยเป็นชั้น Conv 2 ชั้นคู่กับชั้น Max Pooling ทบไปเรื่อยๆ และมี FC 3 ชั้น เลขด้านหลังชื่อแบบจำลองระบุจำนวนชั้นที่ใช้ เช่น VGG16, VGG19 เป็นต้น



รูปที่ 2 ภาพรวมโครงสร้างของ CNN [5]

## 2.4 ตัวอย่างงานวิจัยที่เกี่ยวข้อง

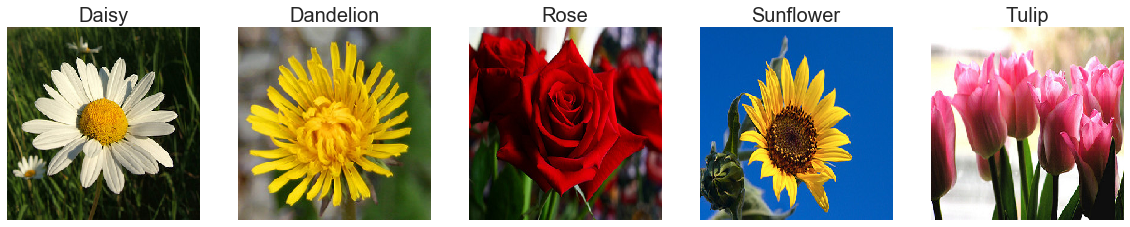
(Gurnani & Mavani, 2011) [4] เปรียบเทียบการใช้แบบจำลอง CNN 2 โครงสร้างคือ AlexNet และ GoogleNet ในการจำแนกดอกไม้ 102 ชนิดบนชุดข้อมูลดอกไม้จาก Oxford University ผลพบว่า GoogleNet ให้ผลลัพธ์ค่าความแม่นยำ Top-1 และ Top-5 ดีกว่าที่ 47.15% 69.17% แต่ก็ไม่ได้เหนือกว่า AlexNet มากนัก

# บทที่ 3

**วิธีดำเนินการ**

## 3.1 ชุดข้อมูล Flowers Recongnition

ชุดข้อมูลนี้ประกอบไปด้วยรูปดอกไม้จำนวน 4323 ภาพ ซึ่งรวบรวมจากการ 3 เว็บไซต์สืบค้นภาพ ได้แก่ flickr, google images, และ yandex images แบ่งเป็นคลาสดอกไม้ 5 ชนิดได้แก่ daisy, dandelion, rose, sunflower, และ tulip แบ่งตามโฟลเดอร์ แต่ละคลาสจะมีจำนวนรูปประมาณ 800 ภาพ ความละเอียด 320x240 พิกเซลโดยประมาณ มีความกว้างยาวและสัดส่วนรูปภาพต่างกันออกไป



รูปที่ 3 ตัวอย่างดอกไม้ในแต่ละคลาส

## 3.2 เครื่องมือที่ใช้งาน

ใช้ไลบรารี่ Keras เป็นเครื่องมือในการนำเข้าแบบจำลอง เขียนโปรแกรมด้วย Jupyter Notebook ประมวลผลการฝึกและทดสอบบน Intel(R) Core(TM) i5-7300 CPU @ 2.60GHz 2.71 GHz RAM 8 GB

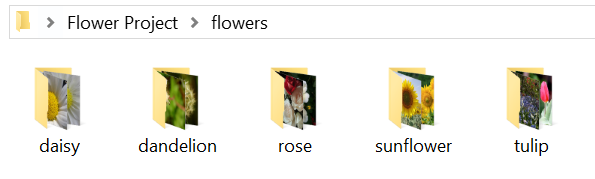
## 3.3 เกณฑ์ในการประเมิน

เกณฑ์ในการประเมินประสิทธิภาพการใช้งานตัวจำแนกเบย์อย่างง่ายที่เลือกใช้ คือร้อยละความแม่นยำในการทำนาย (Accuracy) คำนวณจาก จำนวนที่ทายถูกหารด้วยจำนวนทั้งหมด (%) โดยมีค่าอุดมคติคือ 100%

## 3.4 วิธีดำเนินการ

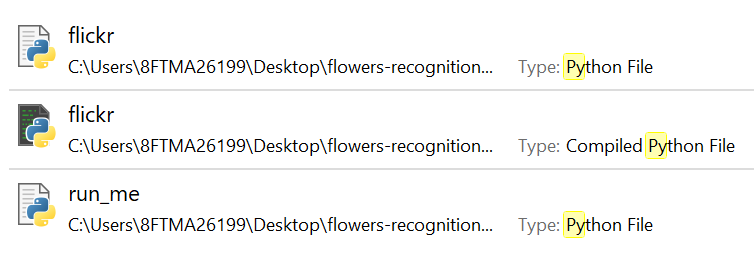
**1. Data Cleaning**

1.1 เมื่อดาวน์โหลดข้อมูลมา ให้แตกไฟล์ zip จัดโฟลเดอร์ให้อยู่ในลำดับขั้น โฟลเดอร์หลัก > 5 โฟลเดอร์ย่อยของคลาส



รูปที่ 4 โครงสร้างโฟลเดอร์ย่อยตามคลาส

1.2 ลบ 3 ไฟล์นี้ที่นามสกุล .py ซึ่งเป็นไฟล์ที่ไม่ใช่รูปภาพปะปนมาใน dataset



รูปที่ 5 ไฟล์ขยะที่ต้องลบ

**2. Data Preparation**

2.1 สำหรับรูปภาพ อ่านไฟล์รูปภาพมาในรูปแบบข้อมูลอาร์เรย์ ด้วยคำสั่ง cv2.imread() และย่อขนาดรูปให้ตรงกับขนาดที่เป็นอินพุตของแบบจำลองที่ใช้ด้วยคำสั่ง cv2.resize() แล้วแปลงเป็น numpy array ด้วยคำสั่ง np.array() จากนั้นหาร 255 เพื่อทำการ normalize ค่าใน numpy array จาก 0 ถึง 255 ให้อยู่ในช่วง 0 ถึง 1 เพื่อให้คำนวณได้รวดเร็ว

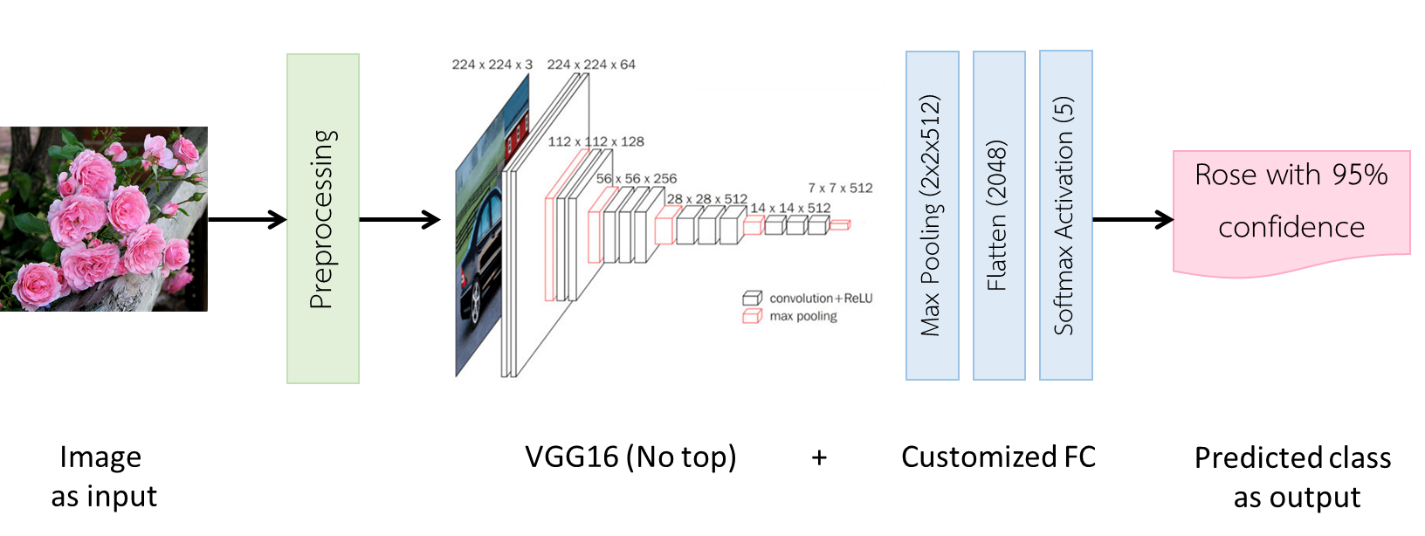
2.2 สำหรับผลเฉลยคลาส แปลงให้อยู่โดยแมพให้ตรงกับดัชนีที่เป็นตัวเลข 0 ถึง n-1 เรียงตามลำดับตัวอักษร จะได้ว่า daisy = 0, dandelion = 1, rose = 2, sunflower = 3 และ tulip = 4 จะได้เวคเตอร์ค่าหมายเลขคลาสของแต่ละภาพ แปลงเป็น numpy array แล้วใช้ LabelBinarizer() แปลงให้อยู่ในรูปอาร์เรย์หลายคอลัมน์ที่แสดงผลแบบไบนารี่ เช่น ผลเฉลายคือหมายเลข [3] ซึ่งก็คือ sunflower จะแปลงได้เป็น [0,0,0,1,0] แทนผลเฉลยในแต่ละภาพ เป็นต้น

2.3 แบ่งข้อมูลสำหรับฝึกและทดสอบด้วยสัดส่วน 80:20 จะได้รูปสำหรับฝึกจำนวน 3,458 ภาพ และรูปสำหรับทดสอบ 865 ภาพ

2.3 ทำ Data Augmentation มีการเปลี่ยนแปลงจะรูปต้นฉบบเล็กน้อย เช่น ขยับรูปไปทางขวา เพื่อเพิ่มจำนวนภาพที่ใช้ในการฝึก สามารถกำหนดความเปลี่ยนแปลงที่จะให้เกิดขึ้นได้ โดยผลการทำซ้ำและปรับเปลี่ยนรูปจะเกิดขึ้นตอนขณะทำการฝึกข้อมูลให้แบบจำลอง พร้อมนำข้อมูลรวมเข้าไปฝึกด้วย

**3. Model Design**

ออกแบบจำลองที่ใช้ในโครงงานนี้ ใช้วิธี Transfer Learning โดยใช้ VGG16 เป็นแบบจำลองฐานที่ตัดชั้น Fully-connected อกแล้วใส่ชั้น Max Pooling แผ่นกรองขนาด 2x2, strides = 2 ตามด้วยชั้น Flatten เพื่อคลี่ปรับอาร์เรย์ให้เป็นเวคเตอร์ แล้วใช้ตัวติดสินใจด้วยฟังก์ชัน softmax 5 คลาสผลลัพธ์



รูปที่ 6 โครงสร้างการจำแนกคลาสดอกไม้

**4. Training and Testing**

นำข้อมูลมาฝึกและทดสอบ โดยกำหนด batch\_size = 64, epochs = 10

**5. Performance Evaluation**

เกณฑ์วัดประสิทธิภาพหลักของโครงงานนี้คือค่าความแม่นยำ (Accuracy) ของชุดข้อมูลทดสอบ นอกจากนี้ ยังสามารถแสดงค่าประสิทธิภาพเกณฑ์อื่นๆ เพิ่มเติมได้

5.1 ค่า Accuracy และ Loss ของข้อมูลทดสอบ ด้วย model.evaluate() และแสดงกราฟที่ค่าเหล่านั้นผันไปตามรอบ epoch

5.2 Classification Report ประกอบด้วยค่า Precision, Recall, และ F1 แยกแต่ละคลาส

5.3 Confusion Matrix บอกจำนวนทายถูกและผิดในแต่ละคลาส

5.4 ตัวอย่างรูปภาพที่ทายถูกและผิด อย่างละ 10 ภาพ พร้อมค่าความน่าจะเป็นของการทำนาย

# บทที่ 4

**ผลการศึกษา**

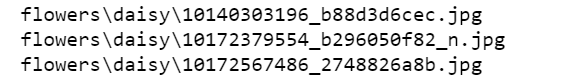
ในหัวข้อนี้ จะนำเสนอการศึกษาขั้นตอนการเขียนโค้ดที่ศึกษาจาก [1] และผลลัพธ์จากการทดสอบกับชุดข้อมูล [3] โดยใช้ Jupyter Notebook เป็นเครื่องมือทำ

1. เรียกใช้ไลบรารี่ที่จำเป็น

|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sns  import keras  from keras.models import Sequential, Model  from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout, MaxPooling2D  from keras.preprocessing.image import ImageDataGenerator  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import classification\_report,confusion\_matrix  from keras.applications import MobileNet  import cv2  import os  import random  import tensorflow as tf |

2. เรียกดูรายชื่อไฟล์ในโฟลเดอร์ย่อย

|  |
| --- |
| for dirname, \_, filenames in os.walk('flowers'):  for filename in filenames:  print(os.path.join(dirname, filename)) |



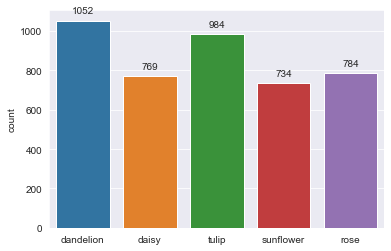
รูปที่ 7 ตัวอย่างพาธไฟล์

3. กำหนดตัวแปร labels สำหรับชนิดดอกไม้ และขนาดรูปที่จะย่อ 150 pixels สร้างเมธอดชื่อ get\_data สำหรับวนลูปอ่านรูปภาพมาในรูปแบบ array จากนั้นย่อขนาดรูปเป็น 150x150 pixels ตามที่กำหนด แล้วใส่ข้อมูลภาพพร้อมตัวเลขดัชนีที่แทนชื่อคลาสลงในตัวแปรอาร์เรย์ data เมื่อวนลูปจนครบทุกไฟล์ในทุกโฟล์แล้ว แปลง data ให้อยู่ในรูป numpy array แล้วส่งตัวแปรนี้เป็นค่ากลับไป (data.shape = (4323, 2))

|  |
| --- |
| labels = ['dandelion', 'daisy','tulip','sunflower','rose']  img\_size = 150  def get\_data(data\_dir):  data = []  for label in labels:  path = os.path.join(data\_dir, label)  class\_num = labels.index(label)  for img in os.listdir(path):  try:  img\_arr = cv2.imread(os.path.join(path, img), cv2.IMREAD\_COLOR)  resized\_arr = cv2.resize(img\_arr, (img\_size, img\_size))  data.append([resized\_arr, class\_num])  except Exception as e:  print(e)  return np.array(data)  data = get\_data("flowers") |

4. แสดงแผนภูมิแท่งของจำนวนรูปในแต่ละคลาส

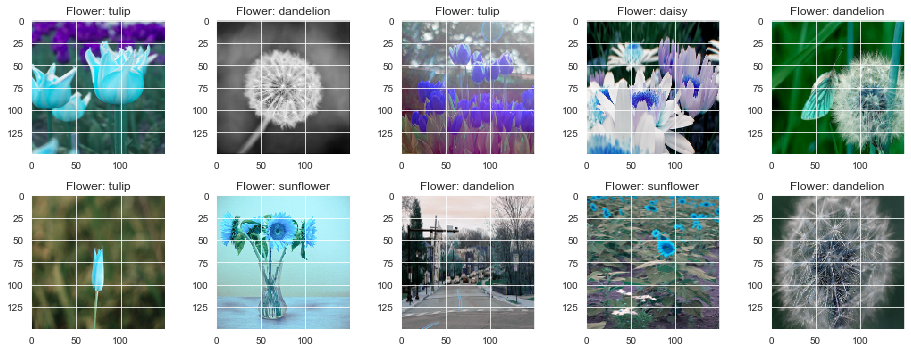
|  |
| --- |
| l = []  for i in data:  l.append(labels[i[1]])  sns.set\_style('darkgrid')  splot = sns.countplot(l)  for p in splot.patches:  splot.annotate(p.get\_height(), (p.get\_x() + p.get\_width() / 2.,  p.get\_height()), ha = 'center', va = 'center', xytext = (0, 10),  textcoords = 'offset points') |



รูปที่ 8 แผนภูมิแท่งจำนวนรูปในแต่ละคลาส

5. แสดงตัวอย่างรูปภาพดอกไม้ที่ใช้ รูปจะแสดงโทนสีไม่เหมือนรูปภาพจริง เนื่องจากตอนที่ใช้เมธอด get\_data ใช้ opencv เป็นตัวอ่านข้อมูล ซึ่งข้อมูลจะถูกอ่านในรูปแบบสี BGR

|  |
| --- |
| fig,ax = plt.subplots(2,5  fig.set\_size\_inches(13,5)  for i in range(2):  for j in range (5):  l=random.randint(0,len(data))  ax[i,j].imshow(data[l][0])  ax[i,j].set\_title('Flower: '+labels[data[l][1]])  plt.tight\_layout() |



รูปที่ 9 ตัวอย่างรูปภาพที่ใช้

6. แยกข้อมูลจากตัวแปร data ออกเป็น 2 ตัวแปร คือ x (ลิสท์ข้อมูลอารเรย์ทั้ง 4323 ภาพ) และ y (ลิสท์หมายเลขแทนคลาส)

|  |
| --- |
| x = []  y = []  for feature, label in data:  x.append(feature)  y.append(label) |

7. แปลง x และ y เป็น numpy array แล้วหาร x ด้วย 255 เพื่อทำการ grayscale normalization ลดผลกระทบจากในเรื่องแสงสีที่แตกต่างกันในแต่ละรูปภาพ นอกจากนี้ เมื่อค่าอยู่ในช่วง [0,1] ยังเป็นการทำให้ CNN คำนวณได้เร็วขึ้นด้วยกว่าค่าในช่วง [0, 255] อีกด้วย reshape เพื่อปรับจากอาร์เรย์ 1 มิติเป็น 3 มิติ (x.shape = (4323, 150, 150, 3), y.shape = (4323, ))

|  |
| --- |
| x = np.array(x) / 255  x = x.reshape(-1, img\_size, img\_size, 3)  y = np.array(y) |

8. ทำให้ค่า y แต่ละอันเป็นอาร์เรย์มิติ 1x5 (y.shape = (4323, 5))

|  |
| --- |
| from sklearn.preprocessing import LabelBinarizer  label\_binarizer = LabelBinarizer()  y = label\_binarizer.fit\_transform(y) |

9. แบ่งข้อมูลที่ใช้ train และ test อัตราส่วน 80:20

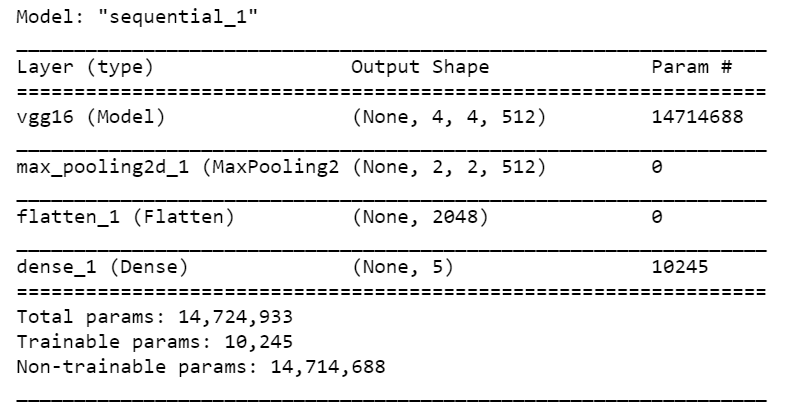
|  |
| --- |
| x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=0) |

10. ทำ data augmentation เพื่อลดปัญหาที่เกิดจาก overfitting และจำนวนรูปที่ไม่เท่ากันในแต่ละคลาส

|  |
| --- |
| datagen = ImageDataGenerator(  featurewise\_center=False, # set input mean to 0 over the dataset  samplewise\_center=False, # set each sample mean to 0  featurewise\_std\_normalization=False, # divide inputs by std of the dataset  samplewise\_std\_normalization=False, # divide each input by its std  zca\_whitening=False, # apply ZCA whitening  rotation\_range = 30, # randomly rotate images in the range (degrees, 0-180)  zoom\_range = 0.2, # Randomly zoom image  width\_shift\_range = 0.1, # randomly shift images horizontally (fraction of total width)  height\_shift\_range = 0.1, # randomly shift images vertically (fraction of total height)  horizontal\_flip = False, # randomly flip images  vertical\_flip=False) # randomly flip images  datagen.fit(x\_train) |

11. นำเข้าและปรับแต่งเพิ่มชั้นในแบบจำลอง VGG16 คอมไพล์โมเดล แสดงสรุปชั้นและจำนวนพารามิเตอร์

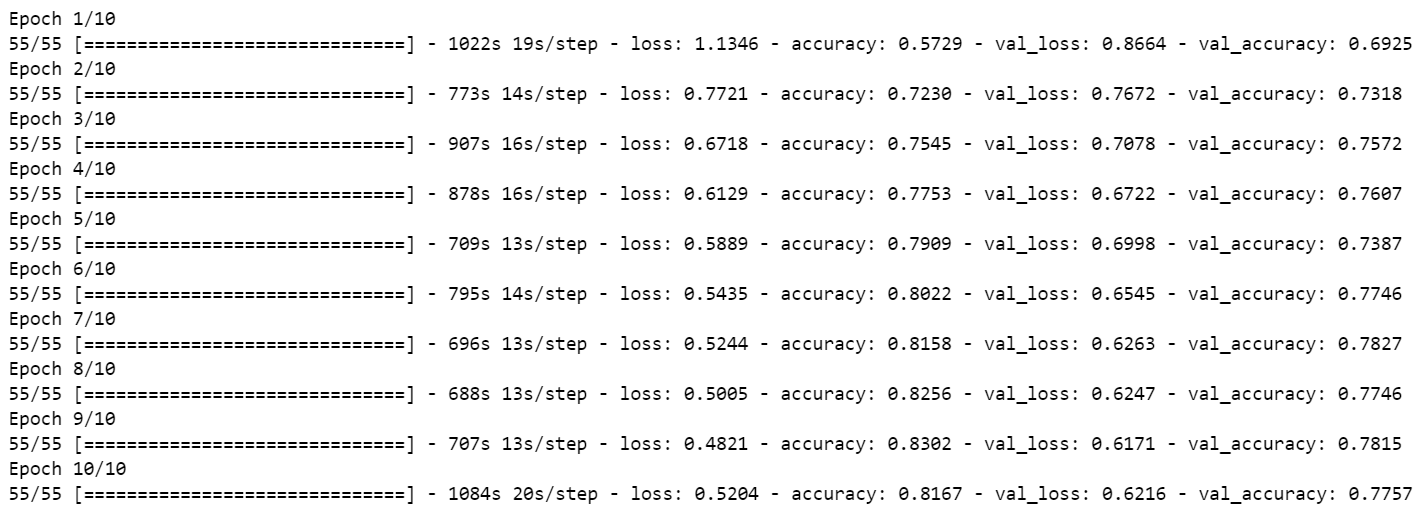
|  |
| --- |
| pre\_trained\_model = VGG16(input\_shape=(150,150,3), include\_top=False, weights="imagenet")  pre\_trained\_model.trainable = False    model = Sequential([  pre\_trained\_model,  MaxPool2D((2,2), strides = 2),  Flatten(),  Dense(5 , activation='softmax')])  model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])  model.summary() |



รูปที่ 10 โครงสร้างแบบจำลอง VGG16 ที่ปรับแต่ง

12. ฝึกข้อมูลกับแบบจำลอง ในขณะเทรนจะมีการสร้าง data augmentation ขณะฝึกแล้วนำไปเข้าฝึกเลยไปด้วย

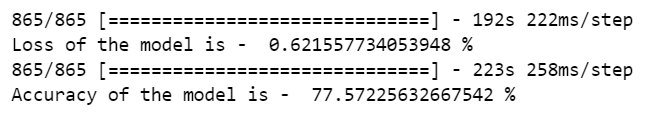
|  |
| --- |
| history = model.fit(datagen.flow(x\_train, y\_train, batch\_size = 64),  epochs = 10 , validation\_data = (x\_test, y\_test)) |



รูปที่ 11 ผลการฝึกข้อมูล epoch 10 รอบ

13. ประเมินค่า Loss และ Accuracy กับชุดข้อมูลทดสอบ

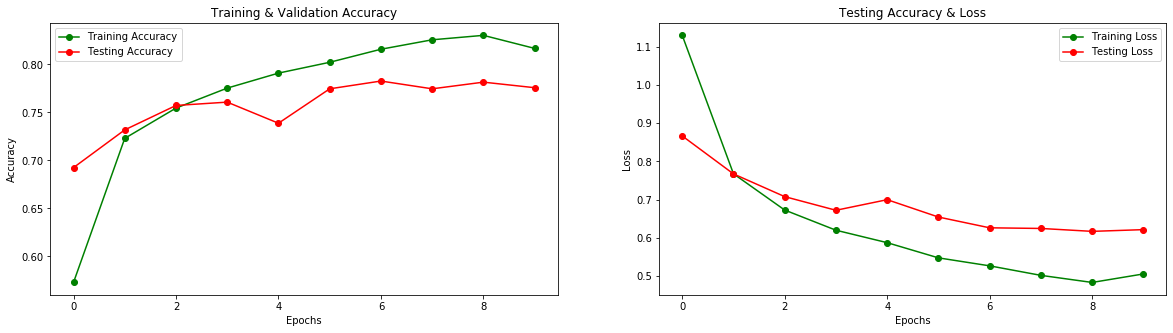
|  |
| --- |
| print("Loss of the model is - " , model.evaluate(x\_test,y\_test)[0], "%")  print("Accuracy of the model is - " , model.evaluate(x\_test,y\_test)[1]\*100 , "%") |



รูปที่ 12 ผลค่า Loss และ Accuracy

14. แสดงกราฟ Training & Validation Accuracy และ Testing Accuracy & Loss ใน Epoch 10 รอบ

|  |
| --- |
| epochs = [i for i in range(10)]  fig , ax = plt.subplots(1,2)  train\_acc = history.history['accuracy']  train\_loss = history.history['loss']  val\_acc = history.history['val\_accuracy']  val\_loss = history.history['val\_loss']  fig.set\_size\_inches(20,5)  ax[0].plot(epochs , train\_acc , 'go-' , label = 'Training Accuracy')  ax[0].plot(epochs , val\_acc , 'ro-' , label = 'Testing Accuracy')  ax[0].set\_title('Training & Validation Accuracy')  ax[0].legend()  ax[0].set\_xlabel("Epochs")  ax[0].set\_ylabel("Accuracy")  ax[1].plot(epochs , train\_loss , 'g-o' , label = 'Training Loss')  ax[1].plot(epochs , val\_loss , 'r-o' , label = 'Testing Loss')  ax[1].set\_title('Testing Accuracy & Loss')  ax[1].legend()  ax[1].set\_xlabel("Epochs")  ax[1].set\_ylabel("Loss")  plt.show() |



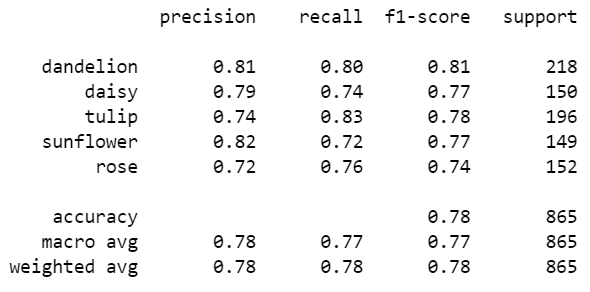
รูปที่ 13 Training & Validation Accuracy และ Testing Accuracy & Loss ใน Epoch 10 รอบ

15. ตัวแปร pred\_cls เป็นอาร์เรย์เก็บค่าหมายเลขคลาสที่ทำนาย ตัวแปร pred\_prop เก็บค่าความน่าจะเป็นในแต่ละคลาสของแต่ละรูปภาพ ตัวแปร y\_test\_inv เป็นอาร์เรย์เก็บค่าหมายเลขคลาสเฉลย

|  |
| --- |
| pred\_cls = model.predict\_classes(x\_test)  pred\_prop = model.predict(x\_test)  y\_test\_inv = label\_binarizer.inverse\_transform(y\_test) |

16. แสดง Classification Report

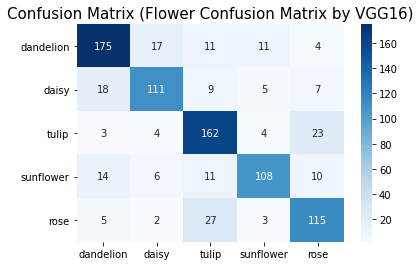
|  |
| --- |
| print(classification\_report(y\_test\_inv, pred\_cls, target\_names = labels)) |



รูปที่ 14 Classification Report

17. แสดง Confusion Matrix

|  |
| --- |
| cm = confusion\_matrix(y\_test\_inv, pred\_cls)  cm = pd.DataFrame(cm, index = labels, columns = labels)  sns.heatmap(cm,cmap= "Blues", annot = True, fmt='' , xticklabels = labels , yticklabels = labels) |



รูปที่ 15 Confusion Matrix

16. เก็บตัวอย่างรูปภาพที่ทำนายถูกและผิดอย่างละ 10 ภาพ เพื่อจะนำมาแสดง

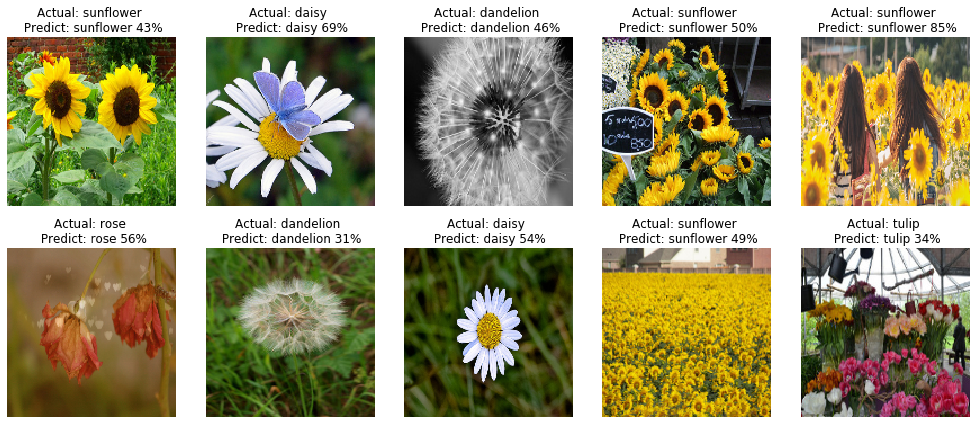
|  |
| --- |
| i=0  prop\_class=[]  mis\_class=[]  for i in range(len(y\_test\_inv)):  if(y\_test\_inv[i] == pred\_cls[i]):  prop\_class.append(i)  if(len(prop\_class) == 10):  break  for i in range(len(y\_test\_inv)):  if(y\_test\_inv[i] != pred\_cls[i]):  mis\_class.append(i)  if(len(mis\_class) == 10):  break |

17. กำหนดเมธอดแปลงรูปโหมด BGR เป็น RGB ไว้แสดงสีรูปปกติ

|  |
| --- |
| def cvtRGB(img):  return cv2.cvtColor(img.copy(), cv2.COLOR\_BGR2RGB) |

18. แสดงรูปที่ทำนายถูก

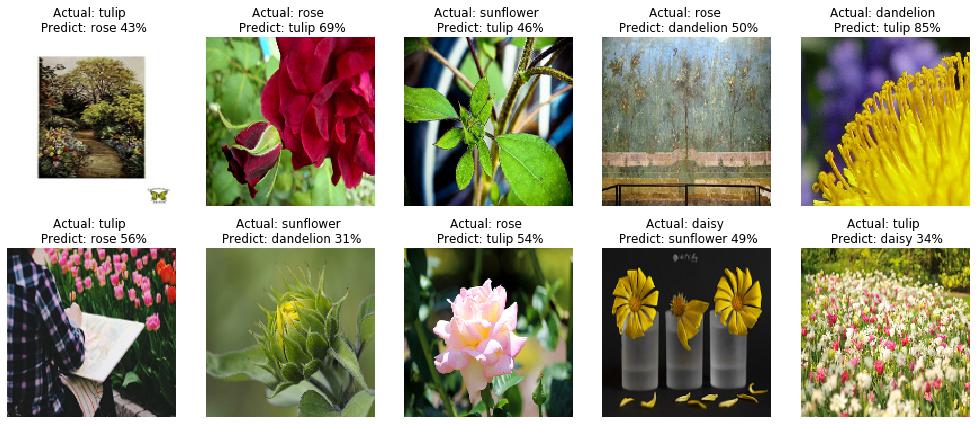
|  |
| --- |
| count=0  fig,ax=plt.subplots(2,5)  fig.set\_size\_inches(14,6)  for i in range (2):  for j in range (5):  img = cvtRGB(x\_test[prop\_class[count]].astype('float32'))  actual = labels[y\_test\_inv[prop\_class[count]]]  predict = labels[pred\_cls[prop\_class[count]]]  prob = round(np.max(pred\_prop[mis\_class[count]]),2)\*100  ax[i,j].imshow(img)  ax[i,j].set\_title('Actual: %s \n Predict: %s %d%%' % (actual, predict, prob))  ax[i,j].set\_axis\_off()  plt.tight\_layout()  count+=1 |



รูปที่ 16 ตัวอย่างรูปที่ทำนายถูก

19. แสดงรูปที่ทำนายผิด

|  |
| --- |
| count=0  fig,ax=plt.subplots(2,5)  fig.set\_size\_inches(14,6)  for i in range (2):  for j in range (5):  img = cvtRGB(x\_test[mis\_class[count]].astype('float32'))  actual = labels[y\_test\_inv[mis\_class[count]]]  predict = labels[pred\_cls[mis\_class[count]]]  prob = round(np.max(pred\_prop[mis\_class[count]]),2)\*100  ax[i,j].imshow(img)  ax[i,j].set\_title('Actual: %s \n Predict: %s %d%%' % (actual, predict, prob))  ax[i,j].set\_axis\_off()  plt.tight\_layout()  count+=1 |



รูปที่ 17 ตัวอย่างรูปที่ทำนายผิด

# บทที่ 5

**สรุปผล**

## 5.1 สรุปและอภิปรายผลการศึกษา

โครงงานนี้นำเสนอการจำแนกดอกไม้ 5 ประเภทจากภาพ โดยใช้แบบจำลอง VGG16 เป็น pre-trained model บนชุดข้อมูล Flowers Recoginition ผลการทดสอบข้อมูลรูปจำนวน 865 ภาพ จากข้อมูลใช้ฝึก 3,458 ภาพ พบว่ามีค่าความแม่นยำที่ 77.57% ซึ่งผู้จัดทำโครงงานถือว่าอยู่เกณฑ์ที่สามารถยอมรับได้สำหรับการใช้งานทั่วไป แต่อาจไม่เหมาะกับงานที่ต้องการความแม่นยำในระดับสูงมาก ข้อดีของการใช้งานตัวจำแนกดอกไม้ที่จัดทำขึ้นนี้คือ สามารถทำตามได้ตามขั้นตอน และการทำใช้วิธี Transfer Learning ทำให้ย่นระยะเวลาการฝึกข้อมูลโดยอาศัยค่าน้ำหนักที่ปรับมาแล้วร่วมกับการเทรนชุดข้อมูลดอกไม้ ทำให้ไม่ต้องเทรนใหม่ทั้งหมดอีกด้วย

## 5.2 ประโยชน์ที่ได้รับ

1) ได้สร้างตัวจำแนกประเภทดอกไม้ที่สามารถนำไปใช้งานได้

2) ได้ศึกษา CNN และวิธีการเขียนโค้ด

3) ได้รับความรู้เพื่อไปประยุกต์ใช้หลักการจำแนกกับงานอื่นๆ ต่อไป

## 5.3 ข้อเสนอแนะ

1) ทดลองใช้แบบจำลองอื่นเป็น Pre-train model เช่น InceptionV3, ResNet50 เป็นต้น

2) เพิ่มประเภทดอกไม้ให้หลากหลายขึ้น

## 5.4 บรรณานุกรม

1. Madz2000. CNN's using VGG16 (83% Accuracy). Retrived May 2, 2020, from <https://www.kaggle.com/madz2000/cnn-s-using-vgg16-83-accuracy>
2. Yinghan Xu. Three models for Kaggle’s “Flowers Recognition” Dataset. Retrived May 2, 2020, from <https://medium.com/@rockyxu399/three-models-for-kaggles-flowers-recognition-dataset-bc2ff732cf4e>
3. Mamaev, A. (2018). Flowers Recognition. Retrived May 1, 2020, from <https://www.kaggle.com/alxmamaev/flowers-recognition>
4. Gurnani, A., & Mavani, V. (2017). Flower categorization using deep convolutional neural networks. arXiv: 1708.03763.
5. Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv: 1409.1556.