



รายงาน

เรื่อง Eucalyptus Disease Classification (การจำแนกโรคของใบยุคาลิปตัส)

เสนอ

ดร.ธนีญา สัตยพานิช

อาจารย์ประจำภาควิชาชีวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ศรีราชา

จัดทำโดย

นางสาวอินธิรา สุทธิสันธ์ รหัสประจำตัวนิสิต 6430301117

นางสาวอิสริยา ทองแพง รหัสประจำตัวนิสิต 6430301125

คณะวิศวกรรมศาสตร์ศรีราชา ภาควิชาชีวกรรมคอมพิวเตอร์ (T12)

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา

การเรียนรู้ของเครื่อง (Machine Learning) รหัสวิชา 03603462-60

ภาคเรียนที่ 1 ปีการศึกษา 2567

มหาวิทยาลัยเกษตรศาสตร์ วิทยาเขตศรีราชา

ที่มาและความสำคัญของปัญหา

ยุคอาลีปตัส เป็นพืชเศรษฐกิจสำคัญของประเทศไทย เนื่องจากสามารถปลูกได้ในพื้นที่ที่มีคุณภาพดินต่ำ และใช้ประโยชน์ในหลายอุตสาหกรรม เช่น อุตสาหกรรมกระดาษ ไม้ และพลังงานชีมวล ต้นยุคอาลีปตัสมีการเติบโตที่รวดเร็วและสามารถเก็บเกี่ยวได้ภายใน 4-7 ปี ทำให้เป็นทางเลือกที่นิยมในการปลูกพืชเชิงพาณิชย์

การเกิดโรคที่ใบของยุคอาลีปตัส เช่น โรคใบจุด ใบไหม้ ใบเหลือง ใบหยิก จากเชื้อราหรือขี้น้อยกับสภาพแวดล้อมต่างๆ เป็นปัญหาสำคัญที่กระทบต่อการเจริญเติบโตและผลผลิต การวินิจฉัยโรคแบบดั้งเดิม จำเป็นต้องพึ่งพาผู้เชี่ยวชาญซึ่งใช้เวลาและทรัพยากรมาก ซึ่งโรคที่ส่งผลต่อใบโดยตรงทำให้ใบเสียหายและลดพื้นที่การสังเคราะห์แสง ต้นจะสูญเสียความสามารถในการสังเคราะห์อาหาร ทำให้การเจริญเติบโตช้าลง หากเป็นโรคในช่วงเวลาสั้น ๆ และได้รับการรักษาหรือป้องกันอย่างรวดเร็ว จะสามารถฟื้นตัวและเจริญเติบโตได้ แต่ถ้าเป็นเวลานาน อาจทำให้ต้นสูญเสียความสามารถในการฟื้นฟูตัวเองและการเจริญเติบโตถูกชะลอลงอย่างถาวร

การตรวจจับและจำแนกโรคของใบยุคอาลีปตัสอย่างรวดเร็วและแม่นยำเป็นสิ่งสำคัญในการป้องกันการเกิดโรคของใบพืชและลดความเสียหายต่อผลผลิต หากไม่ได้รับจัดการโรค อาจส่งผลกระทบต่อผลผลิตทางเศรษฐกิจ และความยั่งยืนของอุตสาหกรรมที่เกี่ยวข้อง

จากแนวคิด ผลการศึกษาและปัญหาดังกล่าว ผู้ศึกษาจึงพัฒนาโปรแกรมนี้ขึ้น โดยให้ความสำคัญในการพัฒนา **Machine Learning** เพื่อจำแนกโรคจากภาพถ่ายใบพืช ซึ่งจะช่วยให้การวินิจฉัยโรค ลดต้นทุนการใช้แรงงานผู้เชี่ยวชาญ แก้ไขปัญหาได้อย่างตรงจุด อีกทั้งยังช่วยให้เกษตรกรสามารถใช้สารเคมีป้องกันและรักษาได้ในปริมาณที่เหมาะสม และสามารถจัดการโรคได้อย่างมีประสิทธิภาพ ทำให้เพิ่มผลผลิตและความมั่นคงทางเศรษฐกิจในระยะยาว

ขั้นตอนการเก็บรวบรวมและจำแนกชุดข้อมูล

วิธีการเก็บรวบรวมข้อมูลเพื่อใช้ในการพัฒนาโมเดล Machine Learning สำหรับการจำแนกรोคนองใบยูคาลิปตัสเป็นขั้นตอนสำคัญที่ต้องมีการวางแผนและดำเนินการอย่างละเอียด ดังนี้

1. การเก็บภาพถ่ายใบยูคาลิปตัส (Image Collection) ไปถ่ายภาพใบยูคาลิปตัสจากแหล่งเพาะปลูกที่มีการพบโรค เพื่อให้ได้ตัวอย่างที่หลากหลายครอบคลุมทั้งใบที่เป็นโรคและใบที่ไม่เป็นโรค ด้วยตนเอง
2. Data Cleaning ตรวจสอบและคัดกรองข้อมูลที่ไม่สมบูรณ์ เช่น ภาพที่ไม่ชัด, ภาพที่ไม่สอดคล้องกับโรคที่ระบุ หรือภาพที่มีการบันทึกผิดพลาด
 - จากจำนวนชุดข้อมูลรูปภาพก่อนการคัดกรองเท่ากับ 250 ภาพ คัดกรองแล้วเหลือ 100 ภาพ
3. การจัดหมวดหมู่โรค (Labeling the Data) แบ่งหมวดหมู่ข้อมูลเป็นใบที่มีอาการโรคแต่ละชนิด จากการรวบรวมข้อมูลที่ได้มาทั้งหมด สามารถจำแนกได้ 5 คลาส ได้แก่ ใบจุด ใบหยิก ใบเหลือง ใบไหม้ และใบสุขภาพดี จึงได้คลาสละ 20 รูป
4. การจัดหมวดหมู่โรค (Labeling the Data) แบ่งหมวดหมู่ข้อมูลเป็นใบที่มีอาการโรคแต่ละชนิด จากการรวบรวมข้อมูลที่ได้มาทั้งหมด ได้คลาสละ 20 รูป สามารถจำแนกได้ 5 คลาส ได้แก่



ใบจุด



ใบหยิก



ใบเหลือง



ใบไหม้



ใบสุขภาพดี

Libraries Utilized for Model Training

- TensorFlow เป็นไลบรารีที่เน้นการสร้างและฝึกโมเดล เหมาะสำหรับข้อมูลที่ซับซ้อน และรองรับการประมวลผลแบบขนาดบัน GPU และ TPU เพื่อเพิ่มประสิทธิภาพในการฝึกโมเดลขนาดใหญ่
- tensorflow.keras เป็นการนำเข้า Keras ซึ่งเป็น API สำหรับการสร้างและฝึกโมเดล Deep Learning ที่รวมอยู่ใน TensorFlow รวมถึงคลาสที่เกี่ยวข้องกับการสร้างและการจัดการโมเดล เช่น Sequential, Model และการนำเข้าฟังก์ชันและคลาสสำหรับสร้างเลเยอร์ต่าง ๆ ในโมเดล เช่น Dense, Conv2D, MaxPooling2D
- Matplotlib เป็นห้องสมุดสำหรับการสร้างกราฟและการแสดงผลข้อมูลใน Python

ขั้นตอนการสร้างโมเดล

- การประมวลผลล่วงหน้า (Preprocessing) การเตรียมข้อมูลภาพเพื่อให้สามารถนำเข้าโมเดลได้อย่างมีประสิทธิภาพเป็นขั้นตอนสำคัญ ประกอบด้วย
 - การปรับขนาดภาพ (Resizing) 256x256พิกเซล
 - การปรับค่าพิกเซลของภาพ (normalization) ให้เป็นช่วง 0-1 โดยการหารค่าพิกเซลด้วย 255
 - การเพิ่มข้อมูล (Data Augmentation) ใช้ฟังก์ชัน Random Flip ในการพลิกภาพแบบสุ่ม ทั้งในแนวอนและแนวตั้ง(horizontal_and_vertical) ในแต่ละครั้งที่มีการเรียกใช้ฟังก์ชัน Random Flip
 - การแบ่งข้อมูลเพื่อการฝึกโมเดลและทดสอบ (Data Splitting) โดยแบ่งข้อมูลออกเป็น ชุดฝึก (Training Set) 80% ชุดทดสอบ (Testing Set) 10% และชุดข้อมูลสำหรับตรวจสอบ (Validation Set) 10%
- การเลือกประเภทโมเดล (Model Type Selection) การทดลองในครั้งนี้ใช้โมเดลประเภท Convolutional Neural Networks (CNN) ในการสร้างโมเดลทดลอง
 - Convolutional Layers ที่ใช้ได้แก่ Conv2D 3 เลเยอร์ ประกอบไปด้วย
 - เลเยอร์ที่ 1 มี 32 kernel ขนาด 3x3 และมีการรักษาขนาดของข้อมูลด้วยการเติมข้อมูล (padding)
 - เลเยอร์ที่ 2 และ 3 มี 64 kernel ขนาด 3x3 และมีการรักษาขนาดของข้อมูลด้วยการเติมข้อมูล(padding)

2) Activation Functions

- ReLU ใช้เป็นฟังก์ชันการเปิดใช้งานใน Convolutional Layers ทั้งสามเลเยอร์ และใน Dense Layer แรกช่วยให้โมเดลเรียนรู้ได้เร็วขึ้นและมีประสิทธิภาพสูงขึ้นในงานที่มีความซับซ้อน
- Softmax ใช้เป็นฟังก์ชันการเปิดใช้งานในเลเยอร์สุดท้ายนั่นคือ Dense Layer ช่วยให้การตีความผลลัพธ์ง่ายขึ้นในการจำแนกประเภทหลายคลาส

- 3) Pooling Layers ใช้ Max Pooling layer 3 เลเยอร์ เพื่อทำหน้าที่ลดขนาดของ Feature Map โดยเลือกค่าสูงสุดในบริเวณเล็กๆ ของ Feature Map จากขั้นตอนโน้มูลชั้น ทำให้ลดจำนวนพารามิเตอร์ในโมเดลและลดความซับซ้อน อีกทั้งยังช่วยลด overfitting ทำให้โมเดลทนทานต่อการเปลี่ยนแปลง
- 4) Fully Connected Layers ใช้ Flatten Layer 2 เลเยอร์ แปลงข้อมูลจากรูปแบบ 2 มิติ เช่น ข้อมูลภาพ ให้เป็นเวกเตอร์ 1 มิติ เพื่อป้อนเข้าสู่เลเยอร์ Dense โดยใช้ Dense Layer แรก เป็น เลเยอร์ Fully Connected ที่มีนิวรอน 64 ตัว และใช้ฟังก์ชัน ReLU สร้างความไม่เชิงเส้นและเพิ่มความซับซ้อนในการเรียนรู้
- 5) Output Layer ใช้ Dense Layer สุดท้ายทำหน้าที่ในการจำแนกประเภทของข้อมูลออกเป็น n คลาส โดยใช้ฟังก์ชัน Softmax ซึ่งจะเปลี่ยนค่าเออต์พุตเป็นความน่าจะเป็นสำหรับแต่ละคลาส

3. Hyperparameters ประกอบไปด้วย

- 1) Batch Size คือ จำนวนตัวอย่างข้อมูลที่ใช้ในการอัปเดตหน้ากินแต่ละรอบ
ในการทดลองใช้ Batch Size = 4
- 2) Number of Epochs จำนวนรอบการฝึกที่โมเดลจะฝึกผ่านข้อมูลทั้งหมดในชุดข้อมูล
ในการทดลองใช้ Epochs = 25
- 3) Optimizer เป็นอัลกอริธึมที่ใช้ในการปรับหน้ากินของโมเดล
ในการทดลองใช้ Adam
- 4) Conv2D Layers จะมีผลต่อการเรียนรู้คุณลักษณะจากข้อมูลภาพ
ในการทดลองใช้ Conv2D Layers = 3
- 5) Activation Function เป็นฟังก์ชันในการเปิดใช้งานช่วยให้โมเดลสามารถเรียนรู้และจับลักษณะเฉพาะที่ซับซ้อนได้
ในการทดลองใช้ ReLU และ Softmax
- 6) Data Augmentation ใช้ Random Flip ฟังก์ชัน ในการพลิกภาพแบบสุ่ม ทั้งในแนวตั้งและแนวตั้ง(horizontal_and_vertical) ในแต่ละครั้งที่มีการเรียกใช้ฟังก์ชัน Random Flip

4. การวัดผลลัพธ์ (Evaluation) ประเมินผลลัพธ์โดยใช้ accuracy และ loss รวมถึงการแสดงผลด้วยกราฟเพื่อเข้าใจพัฒนาการและประสิทธิภาพของโมเดล

การทดลอง

ตารางการศึกษาผลกระทบของ Batch Size

สมมติฐาน :

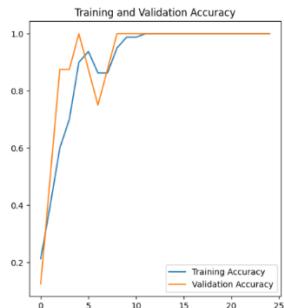
- Batch Size ขนาดเล็กเกินไป จะทำให้การอัปเดตน้ำหนักไม่เดลละเอียดและมีความผันผวนสูงกว่า ส่งผลให้การฝึกมีความเร็วในการคุณเวอร์เจนซ์ที่ช้าลงแต่แม่นยำมากขึ้น
- Batch Size ขนาดใหญ่เกินไป จะทำให้การคุณเวอร์เจนซ์เร็วขึ้น แต่ประสิทธิภาพอาจลดลงเนื่องจาก การอัปเดตน้ำหนักไม่ละเอียดพอ

ตัวแปรควบคุม : Number of Epochs, Optimizer, Conv2D Layers, Activation Function, Data Augmentation

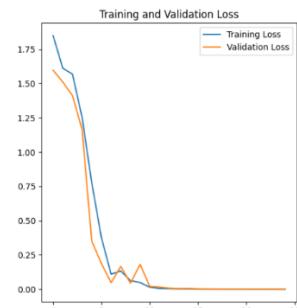
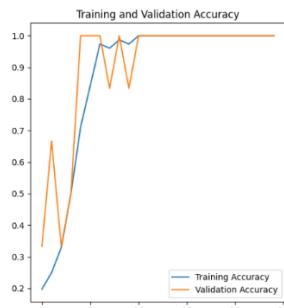
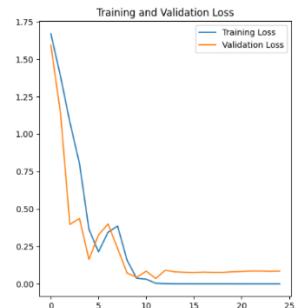
วัตถุประสงค์ : ศึกษาผลกระทบของ Batch Size ที่ต่างกันต่อประสิทธิภาพของโมเดล

Experiment	Batch Size	Epochs	Optimizer	Conv2D Layers	Activation Function	Data Augmentation Parameters	Results
1	4	25	Adam	3	ReLU	None	Accuracy : 1.0000 loss : 0.0082
2	6	25	Adam	3	ReLU	None	Accuracy : 0.9722 loss : 0.0436
3	10	25	Adam	3	ReLU	None	Accuracy : 0.6000 loss: 3.4626

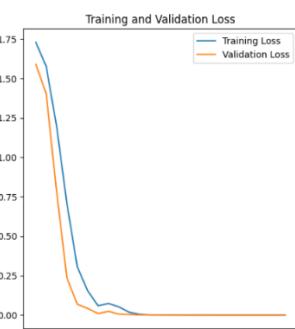
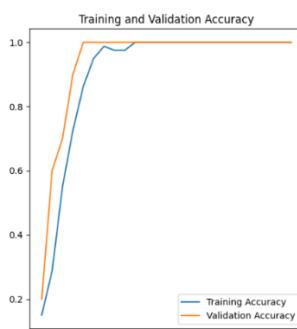
การทดลองที่ 1 Batch Size = 4



การทดลองที่ 2 Batch Size = 6



การทดลองที่ 3 Batch Size = 10



จากการทดลองตารางการศึกษาผลกราฟของ Batch Size

ได้ทำการทดลองทั้งหมด 3 ครั้ง โดยในแต่ละครั้งมีการเปลี่ยนแปลงขนาดของ Batch Size = 4 6 10 ตามลำดับ ใช้จำนวน Epochs เท่ากันที่ 25, Optimizer ใช้ชนิด Adam, จำนวนชั้น Conv2D เท่ากับ 3, พังก์ชัน การกรวยตันที่ใช้คือ ReLU รวมถึงไม่มีการใช้การเพิ่มข้อมูล (data augmentation) ในทุกการทดลอง

แสดงให้เห็นว่า จำนวน batch size ที่มีมากเกิดไปส่งผลกระทบต่อความแม่นยำและทำให้ลดลงอย่างเห็นได้ชัด ดังนั้น Experiment 1 ที่มี batch size = 4 จึงมีความแม่นยำสูงสุดที่ 1.0000 ซึ่งอาจบ่งบอกว่าโมเดลทำการฝึกฝนได้ดีในขนาด batch นี้ สามารถสังเกตการทดลองโดยภาพรวมได้จากการทดลองที่ 1

ตารางการศึกษาผลกระทบของ Epochs

สมมติฐาน :

- เมื่อเพิ่มจำนวน Epochs จะทำให้โมเดลมีโอกาสเรียนรู้จากข้อมูลได้มากขึ้น ส่งผลให้ค่า Accuracy ของโมเดลสูงขึ้น
- หาก Epochs มากเกินไป (เช่น 100) อาจทำให้เกิดการฝึกที่มากเกินไป (Overfitting) ส่งผลให้ประสิทธิภาพกับชุดข้อมูลทดสอบลดลง

ตัวแปรควบคุม : Batch Size, Optimizer, Conv2D Layers, Activation Function, Data Augmentation

วัตถุประสงค์ : ศึกษาผลกระทบของ Epochs ที่แตกต่างกันต่อประสิทธิภาพของโมเดล

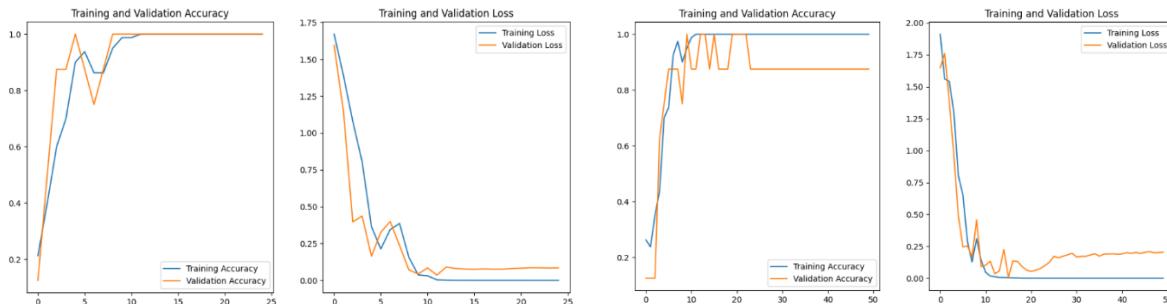
Experiment	Batch Size	Epochs	Optimizer	Conv2D Layers	Activation Function	Data Augmentation Parameters	Results
1	4	25	Adam	3	ReLU	None	Accuracy : 1.0000 loss : 0.0082
2	4	50	Adam	3	ReLU	None	Accuracy : 0.8646 loss : 0.2361
3	4	100	Adam	3	ReLU	None	Accuracy : 0.9583 loss : 0.0397

การทดลองที่ 4 Epochs = 25

(เหมือนกับตารางแรกค่าแรก เพราะค่าเท่ากัน)

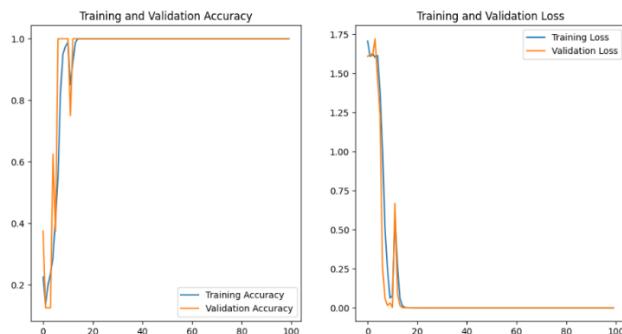
การทดลองที่ 5 Epochs = 50

เริ่มเกิดการ Overfitting



การทดลองที่ 5 Epochs = 50 เริ่มเกิดการ Overfitting : จากการที่ Training Accuracy สูงมาก แต่ Validation Accuracy ต่ำ แสดงว่าโมเดลอาจจະ overfit ข้อมูลฝึกอบรม และไม่สามารถ generalize ไปยังข้อมูลใหม่ได้

การทดลองที่ 6 Epochs = 100



จากการทดลองตารางการศึกษาผลกระทบของ Epochs

ได้ทำการทดลองทั้งหมด 3 ครั้ง โดยในแต่ละครั้งมีการเปลี่ยนแปลงขนาดของ Epochs = 25 50 100 ตามลำดับ ใช้จำนวน batch คงที่ที่ 4 ใช้ชนิด Adam, จำนวนชั้น Conv2D เท่ากับ 3, พังก์ชันการกระตุ้นที่ใช้คือ ReLU รวมถึงไม่มีการใช้การเพิ่มข้อมูล (data augmentation) ในทุกการทดลอง

แสดงให้เห็นว่าการฝึกฝนใน epochs ที่มากขึ้นสามารถปรับปรุงประสิทธิภาพของโมเดลได้หลังจากการ overfitting ในการทดลองครั้งที่ 2 สังเกตได้จากการทดลองโดยภาพรวมได้จากราฟการทดลองที่ 5 และการทดลองครั้งที่ 3 มีการปรับปรุงในความแม่นยำและลดค่า loss ซึ่งแสดงให้เห็นถึงความสำคัญของการเลือกจำนวน epochs ที่เหมาะสมในการฝึกโมเดล ดังนั้น Experiment 1 ที่มี epochs = 25 จึงมีความแม่นยำสูงสุด

ตารางการศึกษาผลกระทบของ Optimizer

สมมติฐาน :

- Optimizer แบบ Adam จะให้ผลการเรียนรู้ที่รวดเร็วและมีประสิทธิภาพมากกว่าเนื่องจากสามารถปรับความเร็วการเรียนรู้ได้อัตโนมัติ
- SGD จะให้ผลการเรียนรู้ที่ช้ากว่าและอาจต้องการจำนวน Epochs มากขึ้นในการฝึกโมเดล แต่จะมีการปรับน้ำหนักที่คงที่กว่า
- RMSprop จะมีประสิทธิภาพที่ใกล้เคียงกับ Adam แต่ในบางกรณีอาจต้องการปรับค่าพารามิเตอร์เพิ่มเติมเพื่อให้ได้ผลลัพธ์ที่ดีที่สุด

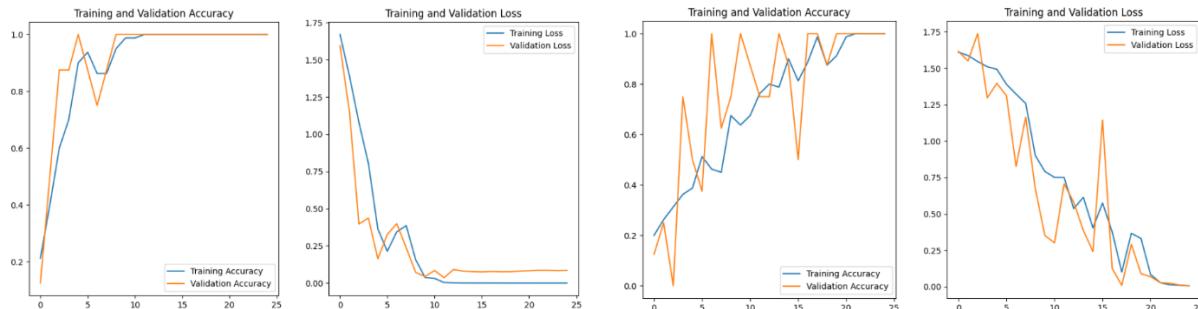
ตัวแปรควบคุม : Batch Size, Epochs, Conv2D Layers, Activation Function, Data Augmentation

วัตถุประสงค์ : ศึกษาผลกระทบของ Optimizer ที่แตกต่างกันต่อประสิทธิภาพของโมเดล

Experiment	Batch Size	Epochs	Optimizer	Conv2D Layers	Activation Function	Data Augmentation Parameters	Results
1	4	25	Adam	3	ReLU	None	Accuracy : 1.0000 loss : 0.0082
2	4	25	SGD	3	ReLU	None	Accuracy : 1.0000 loss : 0.0067
3	4	25	RMSprop	3	ReLU	None	Accuracy : 1.0000 loss : 0.0017

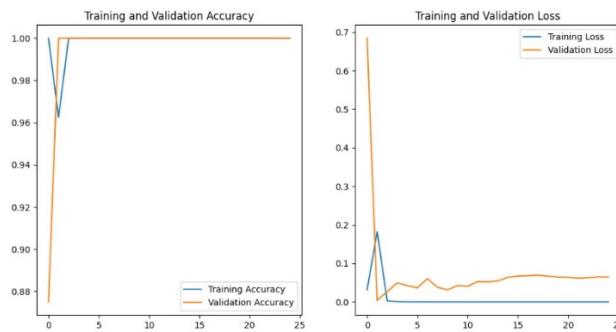
การทดลองที่ 7 Optimizer = Adam

(เหมือนกับตารางแรกค่าแรก เพราะค่าเท่ากัน)



การทดลองที่ 8 Optimizer = SGD

การทดลองที่ 9 Optimizer = RMSprop



จากการทดลองตารางการศึกษาผลลัพธ์ของ Optimizer

ได้ทำการทดลองทั้งหมด 3 ครั้ง โดยในแต่ละครั้งมีการเปลี่ยนแปลงค่าของ Optimizer = Adam, SGD (Stochastic Gradient Descent) และ RMSprop ตามลำดับ ใช้จำนวน batch คงที่ที่ 4, จำนวน epochs คงที่ที่ 25, จำนวนชั้น Conv2D เท่ากับ 3, พังก์ชันการกระตุนที่ใช้คือ ReLU รวมถึงไม่มีการใช้การเพิ่มข้อมูล (data augmentation) ในทุกการทดลอง

แสดงให้เห็นว่าการเลือก optimizer ที่แตกต่างกันเมื่อใช้โครงสร้างโมเดลเดียวกัน โดยทุก optimizer ได้ผลลัพธ์ที่ดี (accuracy = 1.0000) แต่มีความแตกต่างในค่า loss ซึ่งชี้ให้เห็นถึงประสิทธิภาพในการฝึกของ optimizer แต่ละตัว โดย RMSprop ให้ผลลัพธ์ที่ดีที่สุด

ตารางการศึกษาผลกระทบของ Conv2D Layers

สมมติฐาน :

- การเพิ่มจำนวนชั้น Conv2D จะช่วยให้โมเดลสามารถดึงข้อมูลเชิงลึกจากภาพได้มากขึ้น ส่งผลให้ Accuracy เพิ่มขึ้น
- แต่หากเพิ่มจำนวนชั้นมากเกินไป (9 layers) อาจทำให้โมเดลมีความซับซ้อนเกินไปและเกิดการ Overfitting หรือทำให้การฝึกซัล

ตัวแปรควบคุม : Batch Size, Epochs, Optimizer, Activation Function, Data Augmentation

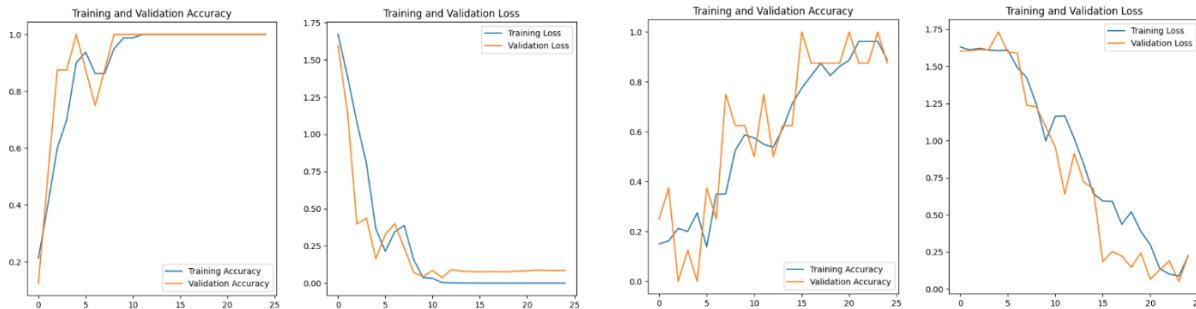
วัตถุประสงค์ : ศึกษาผลกระทบของจำนวน Conv2D Layers ที่แตกต่างกันต่อประสิทธิภาพของโมเดล

Experiment	Batch Size	Epochs	Optimizer	Conv2D Layers	Activation Function	Data Augmentation Parameters	Results
1	4	25	Adam	3	ReLU	None	Accuracy : 1.0000 loss : 0.0082
2	4	25	Adam	6	ReLU	None	Accuracy : 0.7500 loss : 0.4334
3	4	25	Adam	8	ReLU	None	Accuracy : 0.8021 loss : 0.5678

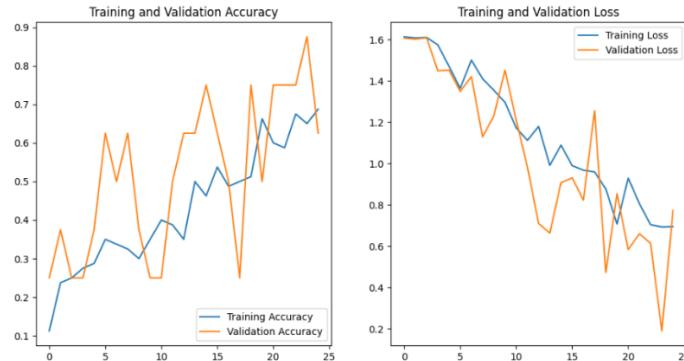
การทดลองที่ 10 Conv2D Layers = 3

การทดลองที่ 11 Conv2D Layers = 6

(เหมือนกับตารางแรกค่าแรก เพราะค่าเท่ากัน)



การทดลองที่ 12 Conv2D Layers = 8 เกิดการ Overfitting ซึ่งมีจำนวน layer มากเกินไป โดยเฉพาะเมื่อมีข้อมูลฝึกอบรมจำกัด โมเดลอาจเรียนรู้ noise ในข้อมูลแทนที่จะเรียนรู้รูปแบบที่แท้จริง ทำให้เกิด overfitting



จากการทดลองตารางการศึกษาผลกรอบของ Conv2D Layers

ได้ทำการทดลองทั้งหมด 3 ครั้ง โดยในแต่ละครั้งมีการเปลี่ยนแปลงขนาดของ Conv2D Layers = 3 6 8 ตามลำดับ ใช้จำนวน batch คงที่ที่ 4, จำนวน epochs คงที่ที่ 25, ใช้ optimizer ชนิด Adam, พังก์ชันการกระตุนที่ใช้คือ ReLU รวมถึงไม่มีการใช้การเพิ่มข้อมูล (data augmentation) ในทุกการทดลอง

แสดงให้เห็นว่าการเพิ่มจำนวน Conv2D layers ในโมเดลที่ใช้ optimizer Adam ส่งผลต่อความแม่นยำ และค่า loss ขณะที่การทดลองครั้งที่ 2 และ 3 มีการเพิ่มจำนวน layers ทำให้มีความแม่นยำลดลง และค่า loss สูงขึ้น อาจที่ให้เห็นถึงปัญหาของการ overfitting หรือโมเดลที่ซับซ้อนเกินไปสำหรับข้อมูลที่ใช้ในการฝึก ดังนั้น Experiment 1 ที่มี Conv2D Layers = 3 จึงมีความแม่นยำสูงสุด

ตารางการศึกษาผลกระทบของ Activation Function

สมมติฐาน :

- การใช้ ReLU จะทำให้การคุณเวอร์เจนซ์รวดเร็วที่สุดและเป็นที่นิยมใช้ในงาน CNN
- Sigmoid อาจให้ผลลัพธ์ที่แย่กว่า ReLU เนื่องจากปัญหาของการคุณเวอร์เจนซ์ช้า
- Tanh คาดว่าจะให้ผลลัพธ์ที่ดีกว่า Sigmoid ในการจำแนกภาพเนื่องจาก Tanh ปรับค่าให้อยู่ในช่วง -1

ถึง 1

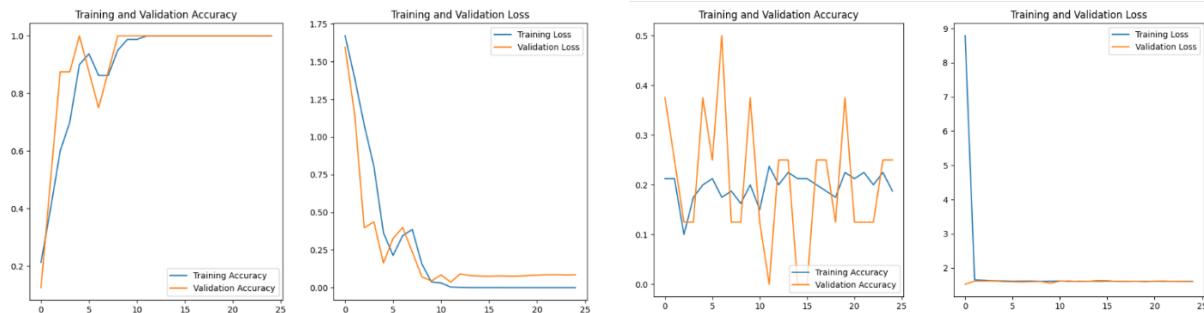
ตัวแปรควบคุม : Batch Size, Epochs, Optimizer, Conv2D Layers, Data Augmentation

วัตถุประสงค์ : ศึกษาผลกระทบของประเภทของ Activation Function ที่แตกต่างกันต่อประสิทธิภาพของโมเดล

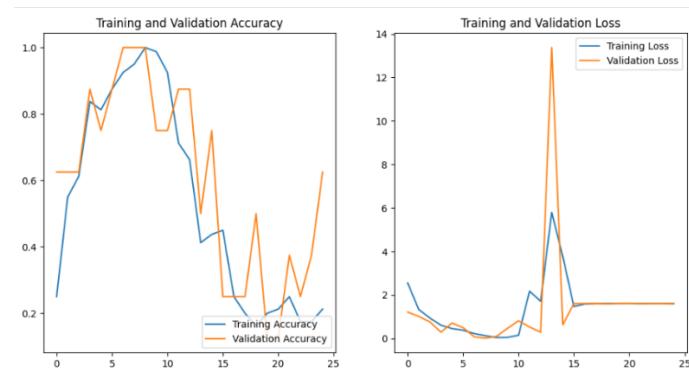
Experiment	Batch Size	Epochs	Optimizer	Conv2D Layers	Activation Function	Data Augmentation Parameters	Results
1	4	25	Adam	3	ReLU	None	Accuracy : 1.0000 loss : 0.0082
2	4	25	Adam	3	Sigmoid	None	Accuracy : 0.4167 loss : 1.5934
3	4	25	Adam	3	Tanh	None	Accuracy : 0.1354 loss : 1.6093

การทดลองที่ 13 Activation Function = ReLU การทดลองที่ 14 Activation Function = Sigmoid

(เหมือนกับตารางแรกค่าแรก เพราะค่าเท่ากัน)



การทดลองที่ 15 Activation Function = Tanh



จากการทดลองตารางการศึกษาผลกรอบของ Activation Function

ได้ทำการทดลองทั้งหมด 3 ครั้ง โดยในแต่ละครั้งมีการเปลี่ยนแปลงค่าของ Activation Function = ReLU Sigmoid Tanh ตามลำดับ ใช้จำนวน batch คงที่ที่ 4, จำนวน epochs คงที่ที่ 25, ใช้ optimizer ชนิด Adam รวมถึงไม่มีการใช้การเพิ่มข้อมูล (data augmentation) ในทุกการทดลอง

แสดงให้เห็นถึงผลของการเลือกฟังก์ชันการกระตุนที่แตกต่างกันในโมเดลที่ใช้ optimizer Adam โดยฟังก์ชัน ReLU ทำให้ได้ผลลัพธ์ที่ดีที่สุด ในขณะที่ฟังก์ชัน Sigmoid และ Tanh มีความแม่นยำต่ำมาก ทำให้เห็นถึงความสำคัญของการเลือกฟังก์ชันการกระตุนให้มีความเหมาะสมต่อโมเดลที่ใช้ในการฝึก

ตารางการศึกษาผลกระทบของ Data Augmentation Parameters

สมมติฐาน :

- การใช้ Data Augmentation จะช่วยป้องกันการ Overfitting และเพิ่มความสามารถของโมเดลในการจำแนกรูปภาพที่ไม่ได้อยู่ในชุดฝึก
- การใช้แค่ RandomFlip คาดว่าจะให้ผลลัพธ์ดีขึ้นเล็กน้อย แต่เมื่อเพิ่ม RandomRotation เข้าไป โมเดลจะสามารถเรียนรู้การเปลี่ยนแปลงของรูปภาพที่หลากหลายมากขึ้น

ตัวแปรควบคุม : Batch Size, Epochs, Optimizer, Conv2D Layers, Activation Function

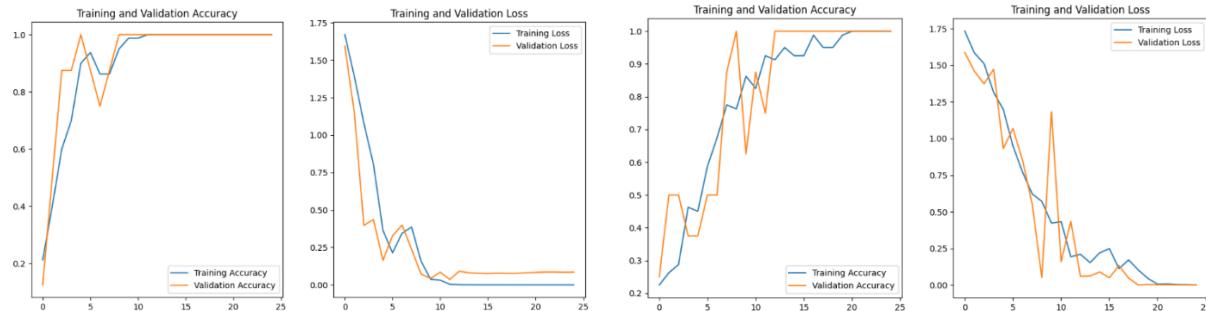
วัตถุประสงค์ : ศึกษาผลกระทบของการใช้ Data Augmentation ที่แตกต่างกันต่อประสิทธิภาพของโมเดล

Experiment	Batch Size	Epochs	Optimizer	Conv2D Layers	Activation Function	Data Augmentation Parameters	Results
1	4	25	Adam	3	ReLU	None	Accuracy : 1.0000 loss : 0.0082
2	4	25	Adam	3	ReLU	RandomFlip : horizontal and vertical	Accuracy : 1.0000 loss : 0.0017
3	4	25	Adam	3	ReLU	RandomFlip : horizontal and vertical, RandomRotation : 0.2	Accuracy : 0.5521 loss : 0.9358

การทดลองที่ 16 Data Augmentation Parameters การทดลองที่ 17 Data Augmentation Parameters

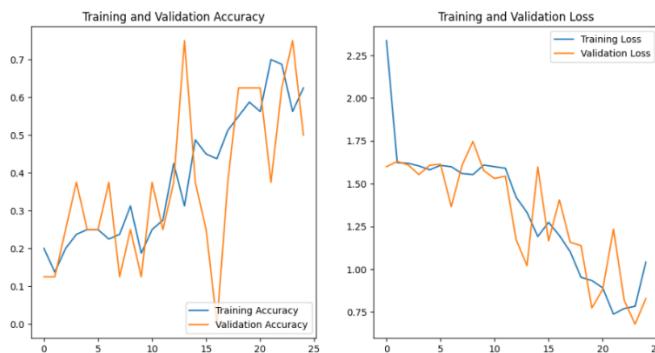
= None การทดลองที่ (เหมือนกับตารางแรกค่าแรก)

= RandomFlip : horizontal and vertical



การทดลองที่ 18 Data Augmentation Parameters = RandomFlip : horizontal and vertical,

RandomRotation : 0.2



จากการทดลองตารางการศึกษาผลกระทำของ Data Augmentation Parameters

ได้ทำการทดลองทั้งหมด 3 ครั้ง โดยในแต่ละครั้งมีการเปลี่ยนแปลงค่าของ Data Augmentation

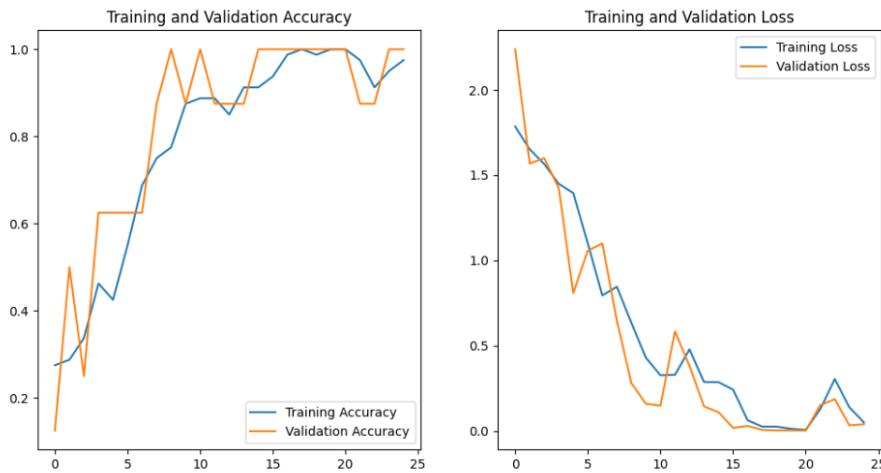
Parameters = None RandomFlip : horizontal and vertical และ RandomFlip : horizontal and vertical, RandomRotation : 0.2 ตามลำดับ ใช้จำนวน batch คงที่ที่ 4, จำนวน epochs คงที่ที่ 25 รวมถึงใช้ optimizer ชนิด Adam ในทุกการทดลอง

แสดงให้เห็นถึงผลของการเพิ่มข้อมูลในโมเดลที่ใช้ optimizer Adam โดยในการทดลองครั้งที่ 1 และครั้งที่ 2 โมเดลแสดงผลลัพธ์ที่ดีมาก แต่เมื่อมีการเพิ่มข้อมูลที่ซับซ้อนมากขึ้นในการทดลองครั้งที่ 3 ความแม่นยำลดลงอย่างเห็นได้ชัด ซึ่งอาจบ่งบอกว่าโมเดลเริ่มสับสนหรือไม่สามารถจัดการกับการเปลี่ยนแปลงที่เกิดขึ้นได้ดี

ตารางสรุปผลการทดลอง

Experiment	Batch Size	Epochs	Optimizer	Conv2D Layers	Activation Function	Data Augmentation Parameters
1	Accuracy : 1.0000 loss : 0.0082	Accuracy : 1.0000 loss : 0.0082	Accuracy : 1.0000 loss : 0.0082	Accuracy : 1.0000 loss : 0.0082	Accuracy : 1.0000 loss : 0.0082	Accuracy : 1.0000 loss : 0.0082
2	Accuracy : 0.9722 loss : 0.0436	Accuracy : 0.8646 loss : 0.2361	Accuracy : 1.0000 loss : 0.0067	Accuracy : 0.7500 loss : 0.4334	Accuracy : 0.4167 loss : 1.5934	Accuracy : 1.0000 loss : 0.0017
3	Accuracy : 0.6000 loss: 3.4626	Accuracy : 0.9583 loss : 0.0397	Accuracy : 1.0000 loss : 0.0017	Accuracy : 0.8021 loss : 0.5678	Accuracy : 0.1354 loss : 1.6093	Accuracy : 0.5521 loss : 0.9358

Best Case



accuracy: 1.0000 - loss: 0.0287

กราฟนี้เป็นกราฟที่ได้จากการทดลอง แสดงผลลัพธ์ที่มีความเสถียร รวมถึงค่า accuracy และ loss กรณีที่ดีที่สุด
ขนาดและจำนวนตัวแปรควบคุมที่ใช้

Batch Size = 4 , Epochs = 25 , Optimizer = Adam , Conv2D Layers = 3 , Activation Function =
ReLU และ Data Augmentation Parameters = RandomFlip : horizontal and vertical

ความแม่นยำในการฝึกอบรมและการตรวจสอบ (Training and Validation Accuracy)

แนวโน้ม

- ความแม่นยำในการฝึกอบรม เพิ่มขึ้นจากประมาณ 0.2 จนถึง 1.0 ในรอบที่ 20 และให้เห็นว่าโมเดลสามารถเรียนรู้จากข้อมูลได้อย่างมีประสิทธิภาพ
- ความแม่นยำในการตรวจสอบ มีแนวโน้มที่เพิ่มขึ้น เช่น กัน จาก 0.2 ไปสู่ 1.0 และให้เห็นว่าโมเดลทำงานได้ดีในข้อมูลใหม่

การเปรียบเทียบ

ความแม่นยำในการฝึกอบรมที่สูงกว่าความแม่นยำในการตรวจสอบเล็กน้อยอาจบ่งชี้ถึงการ overfitting แต่การที่ทั้งสองค่ามีความใกล้เคียงกันที่ 1.0 แสดงว่าโมเดลมีความสามารถในการทั่วไป (generalization) ที่ดี

การสูญเสียในการฝึกอบรมและการตรวจสอบ (Training and Validation Loss)

แนวโน้ม

- การสูญเสียในการฝึกอบรม แสดงการลดลงอย่างต่อเนื่องจากประมาณ 2.0 ถึง 0.03 ภายในรอบที่ 20 แสดงว่าโมเดลกำลังปรับปรุงความแม่นยำในการคาดการณ์
- การสูญเสียในการตรวจสอบ ลดลงจาก 2.0 ถึง 0.02 แสดงให้เห็นว่าการคาดการณ์ที่ดีขึ้นทั้งในชุดข้อมูลฝึกอบรมและตรวจสอบ

การเปรียบเทียบ

การสูญเสียในการตรวจสอบสูงกว่าการสูญเสียในการฝึกอบรมเล็กน้อย ซึ่งเป็นร่องปกติ แต่ค่าสูญเสียต่ำทั้งสองชุดบ่งชี้ว่าไม่มีปัญหาการ overfitting ที่สำคัญ

ดังนั้น การลดลงของการสูญเสียที่รวดเร็วแสดงว่าโมเดลกำลังเรียนรู้อย่างมีประสิทธิภาพโดยไม่มีการรบกวนจากฟิเจอร์ที่ไม่เกี่ยวข้อง ปัจจัยเหล่านี้ ทำให้โมเดลได้รับการฝึกอบรมอย่างดี มีการสร้างความรู้ที่ดี และมีการ overfitting ที่น้อย ทำให้เป็นตัวอย่างที่ยอดเยี่ยมในกระบวนการฝึกอบรมโมเดล



หลักการวิเคราะห์ผลการทดลอง

1. Loss : คือค่าความสูญเสียที่คำนวณจากชุดข้อมูลฝึก (training set) ในแต่ละ epoch ซึ่งปัจบุกถึงความสามารถของโมเดลในการเรียนรู้จากข้อมูลฝึก โดยแสดงถึงความคลาดเคลื่อนระหว่างค่าคาดการณ์ และค่าจริงในชุดข้อมูลฝึก
2. Accuracy : คือความแม่นยำของโมเดลในการคาดการณ์ข้อมูลในชุดฝึก (training set) ในแต่ละ epoch ซึ่งแสดงถึงสัดส่วนของข้อมูลที่โมเดลคาดการณ์ถูกต้องจากชุดข้อมูลฝึก
3. Val Loss : คือค่าความสูญเสียที่คำนวณจากชุดข้อมูลทดสอบ/วัดผล (validation set) ในแต่ละ epoch โดยชุดข้อมูล validation เป็นข้อมูลที่โมเดลไม่เคยเห็นมาก่อนในระหว่างการฝึกฝน ดังนั้น val_loss จะบอกถึงความสามารถของโมเดลในการทำนายผลลัพธ์สำหรับข้อมูลใหม่ หาก val_loss สูงกว่าหรือไม่ลดลงเมื่อเทียบกับ loss ของการฝึกฝน แสดงว่าโมเดลอาจกำลัง overfitting ซึ่งหมายความว่าโมเดลเรียนรู้ข้อมูลฝึกได้ดีแต่ไม่สามารถทำงานได้ดีบนข้อมูลใหม่
4. Val Loss : คือค่าความสูญเสียที่คำนวณจากชุดข้อมูล validation ในแต่ละ epoch ซึ่งเป็นข้อมูลที่โมเดลไม่เคยเห็นมาก่อน ระบุความสามารถของโมเดลในการทำนายผลลัพธ์สำหรับข้อมูลใหม่ หาก val_loss สูงกว่าหรือไม่ลดลงเมื่อเทียบกับ training loss แสดงว่าโมเดลอาจกำลัง overfitting หมายถึงเรียนรู้ข้อมูลฝึกได้ดี แต่ไม่สามารถทำงานได้ดีบนข้อมูลใหม่

อภิรายผล สรุปและข้อเสนอแนะ

การอภิรายผล

1. ประสิทธิภาพของโมเดล โมเดลมีประสิทธิภาพดี เนื่องจากมีการปรับ hyperparameters ต่างๆให้เหมาะสมโดยผลลัพธ์คือ accuracy=1.0000 และ loss=0.0082 แต่เมื่อมีการปรับ hyperparameters ต่างๆที่ไม่เหมาะสม โมเดลเกิดความซับซ้อนสำหรับข้อมูลที่ใช้ในการฝึก ทำให้ประสิทธิภาพลดลงได้ถึง accuracy=0.5521 และ loss=0.9358 แสดงให้เห็นว่าโมเดลไม่สามารถรับมือกับการเปลี่ยนแปลงข้อมูลที่ซับซ้อนได้ดี
2. ข้อจำกัดที่พบ เกิดจากการมีข้อมูลน้อยแล้ว ปรับ hyperparameters บางอย่างทำให้โมเดลอาจ overfit ต่อข้อมูลเดิมและไม่สามารถรับมือกับการเพิ่มข้อมูลได้ดี นอกจากนี้ โมเดลที่ใช้อาจไม่ซับซ้อนพอในการจัดการข้อมูลภาพที่ซับซ้อน
3. การปรับปรุงในอนาคต การใช้โมเดลที่ซับซ้อนขึ้น เช่น ResNet หรือ Transfer Learning อาจช่วยปรับปรุงความแม่นยำ ต้องปรับการเพิ่มข้อมูลให้เพื่อป้องกันการ overfitting รวมถึงปรับ hyperparameters ร่วมด้วย เพื่อลดการสูญเสียและปรับปรุงความแม่นยำ

สรุป

- ผลลัพธ์ของการจำแนกภาพเป็นที่น่าพอใจในชุดข้อมูลปกติ แต่เมื่อต้องรับมือกับข้อมูลที่ซับซ้อนขึ้น โมเดลยังไม่สามารถทำงานได้อย่างเต็มที่ และมีประสิทธิภาพลดลงเมื่อมีการเพิ่มข้อมูลที่ซับซ้อน

ข้อเสนอแนะ

- สามารถนำไปเป็นแอพลิเคชันที่สามารถทดลองแบบ real time เพื่อให้มีความสะดวกต่อการใช้งานมากขึ้น
- โมเดลนี้สามารถพัฒนาให้ใช้ได้ในบริบทที่ข้อมูลไม่ซับซ้อนมาก แต่ถ้าต้องใช้งานจริงในสถานการณ์ที่มีภาพที่หลากหลาย ควรปรับปรุงโมเดลหรือใช้โมเดลที่ซับซอนกว่า