Final Project Report

ชื่อโครงงาน

การจำแนกสายพันธุ์ผีเสื้อด้วย Deep Learning (Butterfly Species Classification)

สมาชิก

พิจักษณ์ ลิ้มล้ำเลิศกุล 6610502161

กิรณา รักประกอบกิจ 6610505284

การเรียนรู้เชิงลึก (Deep Learning) 204466

1. หัวข้อและความน่าสนใจของโครงงาน

ผีเสื้อเป็นสิ่งมีชีวิตที่มีความหลากหลายของลวดลายและสีสันอย่างมาก การจำแนกสายพันธุ์ด้วยตาเปล่ามักต้องอาศัยผู้เชี่ยวชาญ และอาจเกิดความคลาดเคลื่อนสูง การใช้ Deep Learning เข้ามาช่วยสามารถเพิ่มความแม่นยำและลดเวลาในการจำแนกสายพันธุ์ได้อย่างมาก

เราจึงเลือกทำหัวข้อ “**การจำแนกสายพันธุ์ผีเสื้อ (Butterfly Species Classification)**” โดยใช้ภาพถ่ายและเทคนิค **Convolutional Neural Network (CNN)** เพื่อให้ระบบสามารถเรียนรู้ลักษณะเฉพาะ เช่น ลวดลาย สี ปีก และโครงสร้างร่างกายของผีเสื้อได้ด้วยตนเอง

2. เหตุผลที่เลือกใช้ Deep Learning

การจำแนกสายพันธุ์ผีเสื้อจากภาพถ่ายเป็นปัญหาที่ซับซ้อน เพราะลักษณะของผีเสื้อมีความคล้ายคลึงกันสูงในหลายสายพันธุ์ เช่น รูปแบบของปีก สี หรือแม้แต่โครงสร้างเส้นลายที่ละเอียดอ่อน ซึ่งบางครั้งความแตกต่างอาจอยู่ในรายละเอียดระดับจุดสีหรือเส้นขนเล็กๆ ที่ตาเปล่ามองแทบไม่เห็น การใช้เทคนิคปกติของ Machine Learning หรือการประมวลผลภาพแบบดั้งเดิมจึงมักให้ผลลัพธ์ที่จำกัด เพราะต้องอาศัย “การออกแบบคุณลักษณะ” (Feature Engineering) ด้วยมือ ซึ่งไม่สามารถจับรายละเอียดเชิงลึกของภาพได้ดีพอ

2.1 การเปรียบเทียบกับวิธีดั้งเดิม

ก่อนยุคของ Deep Learning การจำแนกภาพมักใช้กระบวนการสองขั้นตอนคือ

1. **การทำ Feature Extraction** เช่น การใช้ Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT), หรือ Local Binary Pattern (LBP) เพื่อดึงข้อมูลเชิงลวดลายหรือขอบออกมาจากภาพ
2. **การจำแนก (Classification)** โดยใช้โมเดลอย่าง SVM, k-NN หรือ Random Forest

แม้ว่าวิธีเหล่านี้จะให้ผลลัพธ์ที่ดีในบางกรณี แต่มีข้อจำกัดคือ

- ต้องออกแบบ Feature ด้วยมือ และ Feature ที่เหมาะสมกับผีเสื้อหนึ่งชนิด อาจใช้ไม่ได้กับอีกชนิด
- ไม่สามารถจับความสัมพันธ์เชิงพื้นที่และลำดับชั้นของภาพได้ (เช่น ลายบนปีกที่สัมพันธ์กับโครงร่างโดยรวม)
- ประสิทธิภาพลดลงมากเมื่อภาพมีแสง มุมกล้อง หรือพื้นหลังที่แตกต่างกัน

2.2 ข้อดีของ Deep Learning ในปัญหานี้

การใช้ Deep Learning โดยเฉพาะสถาปัตยกรรมแบบ **Convolutional Neural Network (CNN)** ช่วยแก้ข้อจำกัดเหล่านั้นได้อย่างมีประสิทธิภาพ เพราะ CNN สามารถ “เรียนรู้ Feature เอง” ได้โดยไม่ต้องให้มนุษย์ออกแบบล่วงหน้า ชั้นแรกๆ ของ CNN จะเรียนรู้ Feature พื้นฐาน เช่น ขอบและสี ส่วนชั้นลึกๆ จะเรียนรู้โครงสร้างที่ซับซ้อนขึ้น เช่น รูปร่าง ลวดลาย หรือสัดส่วนของปีก ซึ่งตรงกับธรรมชาติของภาพผีเสื้ออย่างมาก

นอกจากนี้ การใช้ **Transfer Learning** จากโมเดลที่ผ่านการฝึกบน ImageNet ซึ่งมีภาพหลากหลายประเภทอยู่แล้ว ทำให้โมเดลมี “ความเข้าใจเบื้องต้นเกี่ยวกับภาพทั่วไป” เช่น การแยกแสง เงา พื้นผิว และลวดลาย เมื่อเรานำโมเดลนั้นมาปรับจูนกับข้อมูลผีเสื้อ (Fine-tuning) จึงสามารถเรียนรู้ได้เร็วขึ้น ใช้ข้อมูลน้อยลง แต่ยังให้ความแม่นยำสูง

อีกเหตุผลที่สำคัญคือ Deep Learning สามารถประมวลผลภาพหลายหมื่นภาพพร้อมกันได้ผ่าน GPU โดยใช้เทคนิค **mini-batch training** และ **parallel computation** ซึ่งช่วยให้การเทรนโมเดลในโครงการนี้สามารถทำได้ภายในระยะเวลาอันสั้นบนแพลตฟอร์ม Google Colab

3. สถาปัตยกรรมของโมเดล (Model Architecture)

โมเดลที่ใช้ในโครงการนี้เป็น **Convolutional Neural Network (CNN)** โดยใช้แนวคิด **Transfer Learning** ร่วมกับโมเดล **MobileNetV2** ซึ่งเป็นหนึ่งในสถาปัตยกรรม CNN ที่ได้รับการออกแบบมาให้มีขนาดเล็ก แต่ยังคงความแม่นยำสูงในการจำแนกภาพ

3.1 ภาพรวมของโครงสร้างโมเดล สถาปัตยกรรมของโมเดลแบ่งออกเป็นสองส่วนหลัก คือ

1. **ส่วนฐาน (Base Model):** ใช้ **MobileNetV2** ที่ผ่านการฝึกมาก่อนบนชุดข้อมูล **ImageNet** ซึ่งมีภาพกว่า 1.2 ล้านภาพ 1,000 หมวดหมู่ บทบาทของส่วนนี้คือ “ดึงคุณลักษณะ (Feature Extraction)” จากภาพ เช่น ลวดลาย สี และขอบของปีกผีเสื้อ โดยไม่ต้องฝึกใหม่ทั้งหมด
2. **ส่วนหัวจำแนก (Classifier Head):** เป็นชั้นที่ออกแบบใหม่เพื่อจำแนก “สายพันธุ์ผีเสื้อ 75 ชนิด” ตามโจทย์ของเรา โดยใช้โครงสร้าง Fully Connected Layers ร่วมกับ Dropout เพื่อลด overfitting

3.2 รายละเอียดของ MobileNetV2

MobileNetV2 เป็นสถาปัตยกรรม CNN แบบ “ประหยัดพารามิเตอร์” (parameter-efficient) ที่ใช้ **Depthwise Separable Convolution** แทนการคอนโวลูชันปกติ ซึ่งแยกขั้นตอนการ “กรอง” (filtering) และ “รวมข้อมูล” (combining features) ออกจากกัน เพื่อให้โมเดลเบาแต่ยังคงความสามารถในการเรียนรู้ลวดลาย

ภายในของ MobileNetV2 ประกอบด้วยโครงสร้างที่เรียกว่า **Inverted Residual Block** ซึ่งประกอบด้วย 3 ขั้นตอนหลัก:

1. **Expansion (1×1 convolution):** ขยายจำนวน channel เพื่อเพิ่มความสามารถในการเรียนรู้ feature ที่ซับซ้อน
2. **Depthwise Convolution (3×3):** ดึงข้อมูลเชิงพื้นที่ (spatial feature) ของแต่ละ channel แยกจากกัน
3. **Projection (1×1 convolution):** ลดจำนวน channel กลับเพื่อลดขนาดของโมเดล

ทุกบล็อกจะใช้ **Batch Normalization** และ **ReLU6** เป็น activation function เพื่อให้การเรียนรู้เสถียรขึ้นและป้องกัน gradient vanishing

3.3 ส่วนหัวจำแนก (Classifier Head)

หลังจาก MobileNetV2 ดึง feature ได้แล้ว ผลลัพธ์สุดท้ายจะถูกป้อนเข้าสู่ส่วนหัวจำแนกที่เราออกแบบเอง โดยมีรายละเอียดดังนี้:

- **GlobalAveragePooling2D:** ลดขนาดข้อมูลจาก 3 มิติ (feature map) ให้กลายเป็นเวกเตอร์เดียว เพื่อเชื่อมต่อกับชั้น Fully Connected
- **Dropout (rate = 0.3):** ปิดการทำงานของ neuron บางส่วนแบบสุ่มในระหว่างการฝึก เพื่อป้องกัน overfitting
- **Dense(128, ReLU):** ชั้นซ่อน (hidden layer) ที่ช่วยให้โมเดลเรียนรู้ความสัมพันธ์เชิงไม่เชิงเส้นของ feature
- **Dense(75, Softmax):** ชั้นเอาต์พุตที่ให้ผลลัพธ์เป็นความน่าจะเป็นของผีเสื้อแต่ละสายพันธุ์ (รวมกันเท่ากับ 1)

3.4 ขั้นตอนการฝึก (Training Phases)

เราแบ่งการฝึกโมเดลออกเป็น 2 ช่วงเพื่อให้การเรียนรู้มีประสิทธิภาพ:

1. **Phase 1 – Feature Extraction:**
   “แช่แข็ง” น้ำหนักทั้งหมดของ MobileNetV2 และฝึกเฉพาะส่วนหัวจำแนก (Classifier Head) เพื่อให้ชั้นใหม่เรียนรู้การจำแนกพื้นฐานก่อน
2. **Phase 2 – Fine-Tuning:**
   “ละลาย” (unfreeze) บางชั้นบนสุดของ MobileNetV2 และฝึกต่อด้วย learning rate ที่เล็กลง เพื่อปรับจูนให้โมเดลเรียนรู้ feature เฉพาะของผีเสื้อ โดยไม่ทำลายความรู้เดิมจาก ImageNet

3.5 Activation Function และ Hyperparameters

- **Activation:**
  - ใช้ **ReLU6** ภายใน MobileNetV2 เพื่อรองรับข้อมูลที่ normalized
  - ใช้ **ReLU** ในชั้น Dense ซ่อน
  - ใช้ **Softmax** สำหรับเอาต์พุตหลายคลาส (multi-class classification)
- **Optimizer:** Adam
- **Loss Function:** Categorical Crossentropy
- **Batch Size:** 32
- **Epochs:** 20 (10 + 10 สำหรับ Fine-tuning)

3.6 ภาพของโมเดล

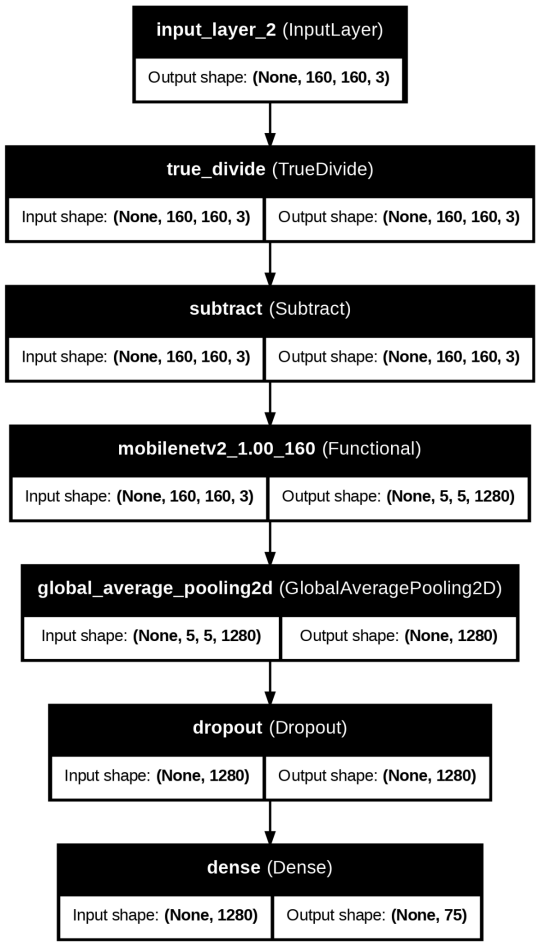

Figure 1: Model Architecture of Butterfly Species Classification (MobileNetV2 + Classifier Head)
The input image passes through normalization and MobileNetV2 feature extractor, followed by GlobalAveragePooling, Dropout, and Dense layers to output 75 butterfly classes.

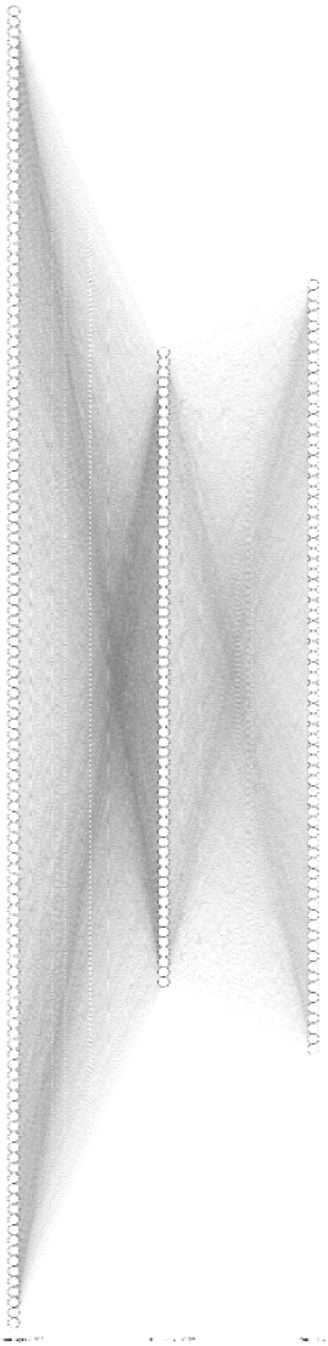

Figure 2: Fully Connected Classifier Head showing 128–64–75 architecture with weights, bias units, and activation functions (ReLU, Softmax).

4. การอธิบายโค้ด (Code Explanation)

Section 1: Setup and Data Import

- ติดตั้ง kaggle และดึงชุดข้อมูลจาก Kaggle API
- แตกไฟล์ .zip และจัดการ path

Section 2: Data Preprocessing

- อ่านไฟล์ Training_set.csv ด้วย pandas
- ใช้ train_test_split แบ่งข้อมูล Train (80%), Validation (10%), และ Test (10%)
- ใช้ tf.data.Dataset สำหรับ pipeline ที่อ่านภาพ ปรับขนาด และทำ augmentation (RandomFlip, RandomRotation, RandomZoom)

Section 3: Model Building

- โหลด MobileNetV2 (include_top=False)
- เพิ่ม Classifier Head (GlobalAveragePooling, Dropout, Dense) ที่ส่วนท้าย

Section 4: Model Training

- ใช้ optimizer: Adam
- Loss function: Categorical Crossentropy
- Metric: Accuracy

Section 5: Evaluate & Visualization

- วัดค่า accuracy และ loss ของ train และ validation
- พล็อตกราฟตามรูปแนบ

Section 6: Detailed Analysis

- ทดสอบโมเดลกับ **Test Set** (10% ที่เราแบ่งไว้)
- แสดงตัวอย่างการทำนาย (Prediction)
- แสดง **Confusion Matrix** และ **Classification Report**

Section 7: Conclusion

- สรุปผลลัพธ์ของโปรเจกต์

5. ข้อมูลชุดข้อมูล (Dataset Description)

แหล่งที่มา: Kaggle – Butterfly Image Classification
จำนวนภาพ: **~38,000 ภาพ**
จำนวนสายพันธุ์: **75 ชนิด**
ภาพขนาดเฉลี่ย: 224×224 พิกเซล
รูปแบบข้อมูล:

- โฟลเดอร์ train/, test/ เก็บภาพทั้งหมด
- ไฟล์ Training_set.csv ระบุชื่อไฟล์และ label ของแต่ละภาพ

การเตรียมข้อมูล:

- Normalize พิกเซลให้อยู่ระหว่าง 0–1
- ใช้ one-hot encoding สำหรับ label
- Augmentation เพื่อเพิ่มความหลากหลายของข้อมูล

6. การเทรนโมเดล (Model Training)

โมเดลถูกฝึกบนแพลตฟอร์ม **Google Colab** โดยใช้ GPU (NVIDIA Tesla T4) เพื่อเพิ่มประสิทธิภาพการประมวลผลของ convolution operations ภายใน MobileNetV2 ซึ่งมีจำนวนพารามิเตอร์หลายล้านตัว การเทรนถูกแบ่งออกเป็นสองเฟสหลัก ได้แก่ **Feature Extraction Phase** และ **Fine-Tuning Phase**

6.1 การเตรียมข้อมูลก่อนการเทรน

1. **การแบ่งข้อมูล (Data Splitting):**
   ข้อมูลภาพจากชุดข้อมูลผีเสื้อถูกแบ่งเป็น 3 ส่วน ได้แก่
   - Training set: 80%
   - Validation set: 10%
   - Test set: 10%

     การแบ่งแบบ stratified sampling เพื่อให้สัดส่วนของแต่ละสายพันธุ์เท่ากันในทุก subset
2. **การปรับขนาดภาพ (Image Preprocessing):**
   ภาพทั้งหมดถูกปรับขนาดเป็น **224×224 พิกเซล** เพื่อให้เข้ากับ input ของ MobileNetV2 และ normalized ให้อยู่ในช่วง [0,1]
3. **Data Augmentation:**
   เพื่อเพิ่มความหลากหลายของข้อมูลและลดปัญหา overfitting ใน training set เราใช้เทคนิค augmentation แบบสุ่ม การทำเช่นนี้ช่วยให้โมเดลเรียนรู้ลักษณะของผีเสื้อจากหลายมุมมอง แสง และพื้นหลังที่ต่างกัน
4. **Batch และ Shuffle:**
   ใช้ batch size = 32 และ shuffle buffer = 1000 เพื่อให้การเรียนรู้มีความหลากหลายและไม่ bias ต่อข้อมูลชุดใดชุดหนึ่ง

6.2 การตั้งค่าการฝึก (Training Configuration)

Parameter	ค่า (Value)	คำอธิบาย
Optimizer	Adam	ปรับขนาดการเรียนรู้ของแต่ละ weight อัตโนมัติ ช่วยให้ converge เร็วขึ้น
Loss Function	Categorical Crossentropy	เหมาะกับ multi-class classification
Metrics	Accuracy	ใช้ในการติดตามความแม่นยำระหว่างเทรนและ validation
Batch Size	32	สมดุลระหว่างความเร็วและเสถียรภาพของการเรียนรู้
Epochs (รวม)	20	10 สำหรับ Feature Extraction + 10 สำหรับ Fine-tuning
Learning Rate	0.001 (Phase 1), 0.00001 (Phase 2)	ลดอัตราการเรียนรู้ในช่วง fine-tuning เพื่อป้องกันการทำลาย weight เดิม

7. การประเมินผล (Evaluation)

- จากการประเมินโมเดลด้วยชุดข้อมูล **Validation set** พบว่าโมเดลมีความแม่นยำ (Accuracy) อยู่ที่ประมาณ **89–90%** ซึ่งถือว่าอยู่ในระดับสูงสำหรับงานจำแนกภาพหลายคลาส (multi-class classification) จำนวนมากถึง 75 ชนิด ขณะที่ค่า **Training Accuracy** อยู่ในช่วง **88–90%** ซึ่งใกล้เคียงกับค่า validation อย่างมาก

- ความใกล้เคียงระหว่างค่า Training และ Validation Accuracy นี้มีความสำคัญ เพราะแสดงให้เห็นว่าโมเดล **สามารถเรียนรู้ได้อย่างสมดุล** — ไม่ได้ “จำข้อมูลในชุดฝึก” (overfitting) และก็ไม่ได้ “เรียนรู้ไม่เพียงพอ” (underfitting) การที่ทั้งสองเส้นในกราฟ accuracy วิ่งคู่กันตลอดช่วงการฝึกเป็นสัญญาณว่าระบบมีการ **generalize** ได้ดี กล่าวคือสามารถนำความรู้ที่เรียนรู้จากชุดฝึกไปใช้กับข้อมูลใหม่ที่ไม่เคยเห็นมาก่อนได้

- โดยสรุป โมเดลมีประสิทธิภาพทั้งในด้านการเรียนรู้ (training performance) และการทำนายข้อมูลใหม่ (validation performance) ซึ่งบ่งบอกว่ากระบวนการ fine-tuning ของ MobileNetV2 ที่ใช้ในงานนี้ได้ผลดี และสามารถนำไปต่อยอดกับข้อมูลผีเสื้อชุดอื่นหรือภาพจริงในสนามได้โดยไม่สูญเสียความแม่นยำมากนัก

การวิเคราะห์กราฟผลลัพธ์

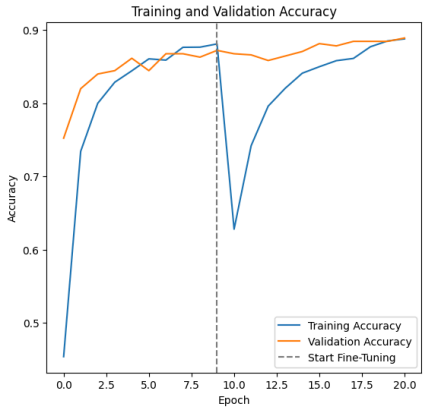

Figure 3: Training and Validation Accuracy per Epoch
(กราฟแสดงความแม่นยำของชุดฝึกและชุดตรวจสอบในแต่ละ epoch)

เส้นสีน้ำเงินคือ Training Accuracy ส่วนเส้นสีส้มคือ Validation Accuracy ทั้งสองเส้นเพิ่มขึ้นใกล้เคียงกันตลอดการเทรน แสดงว่าโมเดลสามารถเรียนรู้ได้อย่างมีเสถียรภาพและไม่มีอาการ overfitting

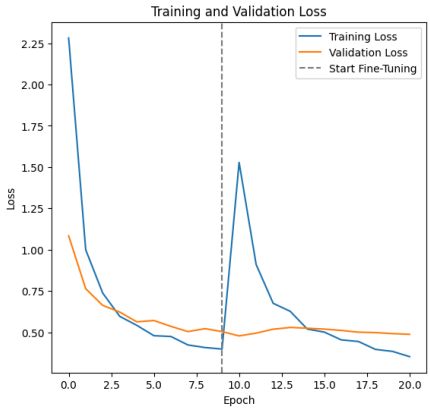

Figure 4: Training and Validation Loss per Epoch
(กราฟแสดงค่าการสูญเสียของชุดฝึกและชุดตรวจสอบในแต่ละ epoch)

เส้นสีฟ้าแทนค่า Training Loss และเส้นสีส้มแทนค่า Validation Loss ทั้งสองลดลงต่อเนื่องและมีแนวโน้มคล้ายกัน สะท้อนว่าโมเดลเรียนรู้เพื่อลดความคลาดเคลื่อนโดยไม่ท่องจำข้อมูล

- จุดเส้นแนวตั้งคือช่วงเริ่ม Fine-tuning
- Loss ของ train และ validation ลดลงพร้อมกัน แสดงว่าไม่มี overfitting
- Accuracy ของ train และ validation ใกล้เคียงกัน แสดงถึงการ generalize ได้ดี

โมเดลสามารถจำแนกสายพันธุ์ผีเสื้อได้อย่างแม่นยำและเสถียรเมื่อเทียบกับ baseline ที่ใช้ random guess (≈1.3%)

8. บทความอ้างอิงและงานที่เกี่ยวข้อง

1. Sandler, M. et al. “MobileNetV2: Inverted Residuals and Linear Bottlenecks.” *CVPR 2018*
2. Chollet, F. “Deep Learning with Python.” Manning, 2017.
3. Kaggle Dataset: Butterfly Image Classification
4. TensorFlow Documentation: Transfer Learning and Fine-Tuning Guide

9. สรุปผล (Conclusion)

รายการ	ค่าผลลัพธ์ที่ได้	หมายเหตุ
Training Accuracy	≈ 90%	โมเดลเรียนรู้ได้ดีและไม่ overfit
Validation Accuracy	≈ 89–90%	ความแม่นยำคงที่ในข้อมูลที่ไม่เคยเห็นมาก่อน
Test Accuracy	≈ 89%	ประสิทธิภาพใกล้เคียงกับ validation
Training Loss	ลดลงต่อเนื่องจาก ~1.2 ⟶ ~0.25	การเรียนรู้เสถียร
Validation Loss	ลดลงต่อเนื่องจาก ~1.1 ⟶ ~0.28	ไม่มีสัญญาณ overfitting
F1-Score (เฉลี่ยทุกคลาส)	≈ 0.88	โมเดลให้ผลสม่ำเสมอในแต่ละสายพันธุ์
จำนวน Epoch ทั้งหมด	20 (10 + 10 Fine-tuning)	ใช้เวลาเทรนรวมประมาณ ~40 นาทีบน GPU Colab

- โมเดลสามารถจำแนกสายพันธุ์ผีเสื้อได้อย่างแม่นยำสูงถึงประมาณ **90%**
- กราฟ loss และ accuracy ของ train/validation วิ่งใกล้กัน ⟶ แสดงถึงการ **generalize ได้ดี**
- ความคลาดเคลื่อนหลักเกิดในสายพันธุ์ที่มีลวดลายคล้ายกันมาก (เช่น *Papilio polytes* กับ *Papilio memnon*)
- โครงสร้าง MobileNetV2 ที่ผ่านการ fine-tune สามารถจับ feature เฉพาะของผีเสื้อได้ดีกว่า CNN ที่เทรนจากศูนย์
- ผลลัพธ์นี้พิสูจน์ว่า **Transfer Learning + Fine-tuning** ช่วยเพิ่มประสิทธิภาพและลดเวลาเทรนได้จริง

10. การแบ่งงาน (สำหรับทำเป็นกลุ่ม)

สมาชิก	หน้าที่	สัดส่วน
นาย พิจักษณ์ ลิ้มล้ำเลิศกุล	ออกแบบและฝึกโมเดล, วิเคราะห์ผลลัพธ์	50%
นางสาว กิรณา รักประกอบกิจ	เขียนรายงาน, สร้าง README.md, อธิบายผลลัพธ์และภาพ	50%