204466 การเรียนรู้เชิงลึก
Deep Learning

รายงานโครงงาน
การจำแนกภาพขยะ 4 ประเภทด้วยเทคนิค Deep Learning

(Waste Classification Using CNN)

ผู้จัดทำ
ธาดา วิยาภรณ์ 6610502081
ธยศ ขันทะยศ 6610505420

อาจารย์ผู้สอน
อาจารย์ ภารุจ รัตนวรพันธุ์

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

ภาคต้น ปีการศึกษา 2568

1. หัวข้อนี้น่าสนใจอย่างไร ทำไมถึงเลือกหัวข้อนี้มาทำเป็น final project

การแยกขยะอย่างถูกต้องเป็นหนึ่งในแนวทางสำคัญของการจัดการสิ่งแวดล้อมอย่างยั่งยืน
แต่ในความเป็นจริงยังมีข้อจำกัดจากพฤติกรรมของคนและระบบคัดแยกที่ไม่ทั่วถึง
การใช้ปัญญาประดิษฐ์ (AI) โดยเฉพาะ Deep Learning
สามารถเข้ามาช่วยเพิ่มประสิทธิภาพการจำแนกขยะได้อย่างแม่นยำและรวดเร็วโมเดลที่ได้สามารถนำไปใช้ในระบบ “ถังขยะอัจฉริยะ (Smart Bin)”
หรือระบบกล้องอัตโนมัติที่คัดแยกขยะได้ทันทีโดยไม่ต้องใช้แรงงานคน
ซึ่งสอดคล้องกับแนวคิด “Smart City” และ “Green Technology”
โดยเหตุผลที่เลือกหัวข้อนี้ยังประกอบด้วย Dataset พร้อมใช้งานจาก Kaggleทำให้สามารถเทรนโมเดลใน Google Colab
ได้โดยไม่ต้องใช้เครื่องมือพิเศษ,
สามารถช่วยลดปริมาณขยะปนเปื้อนในระบบรีไซเคิล
และเพิ่มอัตราการแยกขยะอย่างถูกต้องและโมเดลนี้สามารถพัฒนาไปสู่ระบบอัตโนมัติที่สามารถนำไปใช้ได้จริง

2. ทำไมหัวข้อนี้จึงต้องใช้ deep learning ในการแก้ปัญหา เปรียบเทียบกับการแก้ปัญหานี้ด้วยวิธีอื่นๆ วิธี deep learning มีข้อเด่น ข้อด้อยอย่างไร

การจำแนกภาพขยะเป็นปัญหาที่มีความซับซ้อนสูง เนื่องจากภาพขยะในโลกจริงมีความแตกต่างกันอย่างมาก ทั้งในด้านแสง มุมมอง สีพื้นหลัง และการเสื่อมสภาพของวัตถุ ซึ่งทำให้การแยกประเภทขยะด้วยวิธีการทางสถิติหรือ Machine Learning แบบดั้งเดิมทำได้ยาก เพื่อให้โมเดลสามารถเข้าใจรูปร่างและลักษณะของวัตถุในภาพได้โดยอัตโนมัติ จึงจำเป็นต้องใช้ Deep Learning โดยเฉพาะ Convolutional Neural Networks ซึ่งมีความสามารถในการเรียนรู้ลักษณะเชิงลึก (จากภาพโดยไม่ต้องอาศัยมนุษย์ออกแบบคุณลักษณะล่วงหน้า

<u>ข้อดีของการใช้ Deep Learning</u>

- CNN สามารถเรียนรู้ลักษณะภาพได้โดยอัตโนมัติ
- สามารถรับมือกับภาพขยะที่บิดเบือน มุมมองต่าง ๆ หรือแสงไม่สม่ำเสมอได้ดีกว่าวิธีทั่วไป
- โมเดลสามารถนำความรู้จาก dataset อื่นมาปรับใช้กับขยะประเภทใหม่ ๆ ได้อย่างรวดเร็ว
- เมื่อฝึกเสร็จแล้ว โมเดลสามารถนำไปใช้ในระบบ Smart Bin หรือกล้องตรวจจับอัตโนมัติได้ทันที

<u>ข้อจำกัดของ Deep Learning</u>

- หากข้อมูลมีน้อย โมเดลอาจ overfit และทำนายผิดพลาดเมื่อเจอภาพใหม่
- ต้องอาศัย GPU และเวลาฝึกฝนมากกว่าวิธีทั่วไป
- โมเดลมีลักษณะเป็นกล่องดำทำให้การอธิบายเหตุผลของการตัดสินใจทำได้ยาก

3. อธิบายสถาปัตยกรรม deep learning ที่ใช้ (feedforward NN CNN RNN GAN หรือ VAE) วาดรูปแสดงจำนวนโหนด weight bias รวมถึงการเชื่อมต่อ และ activation function ต่างๆให้ชัดเจน

ชนิด: CNN (Convolutional Neural Network) เหมาะที่สุดกับงานภาพนิ่ง เพราะเรียนรู้ฟิลเตอร์เชิงพื้นที่ (ขอบ ลวดลาย รูปทรง) ได้อัตโนมัติ

โครงสร้าง: Input → Stem(Conv+BN+ReLU) → 5×ConvBlocks(Conv+BN+ReLU ×2 + MaxPool + Dropout2d) → Head(AdaptiveAvgPool → Flatten → Dropout → Linear → ReLU → Dropout → Linear → Softmax-inference)
อินพุต 224×224×3; ช่องสัญญาณเพิ่ม 32→64→128→256→384→512 แล้วสรุปเป็นเวกเตอร์ก่อนทำนาย 12 คลาส

Activation functions:

- ReLU หลังคอนโวลูชันทุกชั้น และใน head (ชั้น 512→256) เพื่อเพิ่มไม่เชิงเส้นและช่วยให้เทรนเร็ว
- Softmax ที่เอาต์พุตตอนอนุมาน แปลง logits เป็นความน่าจะเป็นรายคลาส

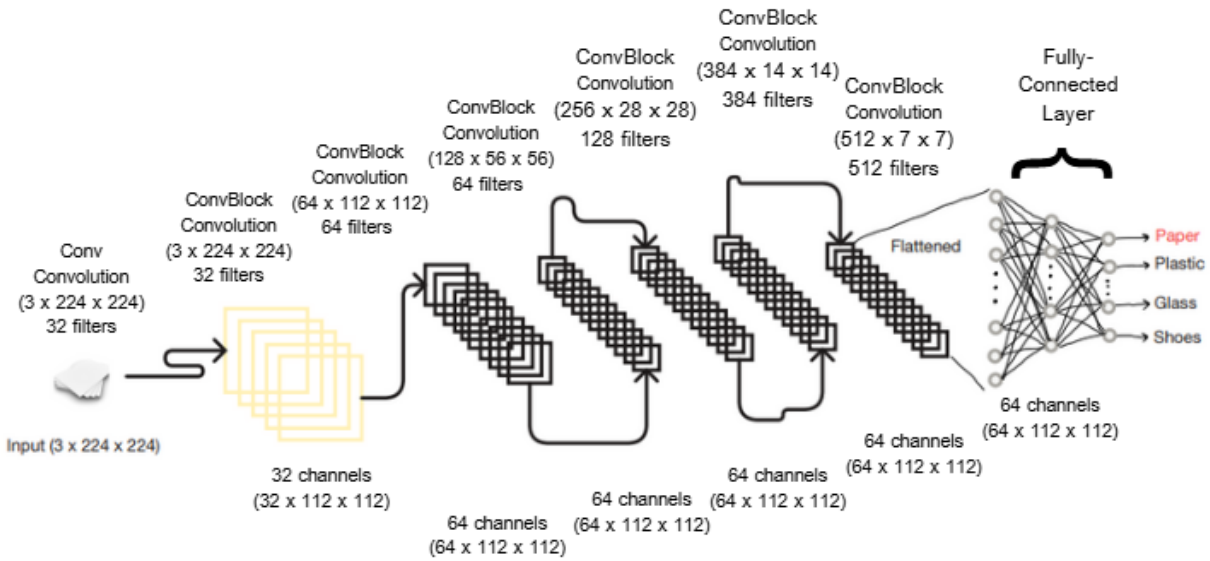
Regularization:

- BatchNorm ทุกคอนโวลูชัน → ทำให้กระจายตัวของค่าแอคติเวชันนิ่งขึ้น
- Dropout2d ในบล็อก และ Dropout ใน head → ลด overfitting
- MaxPool(2×2) ในแต่ละบล็อก → ลดขนาดเชิงพื้นที่ คงฟีเจอร์เด่น

ไดอะแกรม & ตารางพารามิเตอร์:

ชั้น (Layer)	ชนิด (Type)	จำนวน Channels	ขนาด Feature Map (H × W)	คำนวณ (224x224)
INPUT	Image	3	224 x 224	3 x 224 x 224
STEM	Conv (3x3), BN, ReLU	32	224 x 224	32 x 224 x 224
BLOCK 1	ConvBlock (32 --> 64)	64	112 x 112	64 x 112 x 112
BLOCK 2	ConvBlock (64 --> 128)	128	56 x 56	128 x 56 x 56
BLOCK 3	ConvBlock (128 --> 256)	256	28 x 28	256 x 28 x 28
BLOCK 4	ConvBlock (256 --> 384)	384	14 x14	384 x 14 x 14
BLOCK 5	ConvBlock (384 --> 512)	512	7 x 7	512 x 7 x 7

HEAD	AdaptiveAvgPool(1)	512	1 x 1	512 x 1 x 1
FLATTEN	Flatten	512	1	512
FC 1	Linear (512 --> 256), Dropout, ReLU	256	1	256
FC 2	Linear (256 --> 12)	12	1	num_classes



Mengenal Convolutional Neural Network

4. อธิบายโค้ด PyTorch หรือ TensorFlow รวมไปถึงโค้ดส่วนอื่นๆที่ใช้ในการเชื่อมต่อกับโมเดลอื่นๆ อย่างชัดเจน ส่วนไหนจัดการกับข้อมูล ส่วนไหนสร้างโมเดล ส่วนไหน train ฯลฯ

ในส่วนของการอธิบายโค้ดทั้งหมดได้ทำการอธิบายไว้ในลิงค์ด้านล่างนี้

https://github.com/tadashi404/final_project

5. อธิบายวิธีในการ train ตัว deep learning network ที่เลือกมาใช้ รวมไปถึงอธิบาย dataset ที่เกี่ยวข้องและแหล่งที่มา

วิธีในการ Train โมเดล Deep Learning Network
โมเดลนี้ใช้แนวคิด Convolutional Neural Network (CNN) ส่วนหลักๆ จะประกอบไปด้วย

- โครงสร้างภายใน
  - Stem Layer
    เป็นเลเยอร์เริ่มต้นที่รับภาพ RGB (3 channel)
    แล้วแปลงให้เป็น feature map ขนาด 32 channels ผ่าน Conv2D + BatchNorm + ReLU
  - Convolutional Blocks 5 ชั้น
    แต่ละบล็อกมี 2 ชั้น Conv2D + BatchNorm + ReLU พร้อม **MaxPooling** และ **Dropout2D** เพื่อลด overfitting
    ขนาด channel จะเพิ่มขึ้นเรื่อย ๆ:
    32 → 64 → 128 → 256 → 384 → 512
    ซึ่งช่วยให้โมเดลเรียนรู้ feature ที่ซับซ้อนขึ้นตามลำดับ
  - Head Layer

    เป็น fully connected layer สำหรับจำแนกคลาส โดยใช้

    - AdaptiveAvgPool2d(1) → ลด spatial dimension เหลือ 1×1
    - Linear(512→256) + ReLU + Dropout
    - Linear(256→num_classes) เพื่อทำนายผลแต่ละคลาส

- การเตรียมข้อมูล (Data Loader)

  ในโค้ดใช้ train_loader, val_loader, test_loader ซึ่งมักมาจาก torch.utils.data.DataLoader โดยโหลดข้อมูลภาพ (image dataset) ที่แบ่งเป็น 3 ส่วน:

  - **train_loader**: ใช้ฝึกโมเดล (training)
  - **val_loader**: ใช้ประเมินระหว่างฝึก เพื่อปรับ learning rate และเลือก best model
  - **test_loader**: ใช้ทดสอบประสิทธิภาพสุดท้ายของโมเดล

- การคำนวณน้ำหนักคลาส (Class Weights)

  เพื่อแก้ปัญหา **class imbalance** มีการคำนวณ “น้ำหนัก” ของแต่ละคลาส:

```
def compute_class_weights(loader, n_classes):
    counts = torch.zeros(n_classes)
    for _, y in loader:
        for c in range(n_classes):
            counts[c] += (y == c).sum()
    w = 1.0 / torch.clamp(counts, min=1.0)
    w = w / w.sum() * n_classes
    return w

use_class_weights = True
if use_class_weights:
    cls_w = compute_class_weights(train_loader, num_classes).to(device)
    criterion = nn.CrossEntropyLoss(weight=cls_w)
    print("⚖️ class weights:", cls_w.cpu().numpy())
else:
    criterion = nn.CrossEntropyLoss()
```

  จะทำให้โมเดลไม่ลำเอียงไปหาคลาสที่มีข้อมูลเยอะเกินไป

- การฝึกโมเดล (Training Loop)

  การ Train ดำเนินการภายในลูปหลัก

```
for epoch in range(1, EPOCHS+1):
    t0 = time.time()
    tr_loss, tr_acc = run_epoch(train_loader, train_mode=True)
    val_loss, val_acc = run_epoch(val_loader, train_mode=False)
```

โดยฟังก์ชัน run_epoch() จะทำหน้าที่:

- อ่าน batch ของภาพจาก DataLoader
- ส่งเข้าโมเดลเพื่อคำนวณผลลัพธ์และ loss
- ทำ backpropagation เพื่ออัปเดตน้ำหนักของโมเดล (เฉพาะตอน train)
- ใช้ torch.cuda.amp เพื่อรองรับ mixed precision training ช่วยให้ฝึกได้เร็วขึ้นบน GPU

นอกจากนี้โมเดลมีระบบ

- Early Stopping: หยุดฝึกเมื่อ val accuracy ไม่ดีขึ้นในช่วง patience
- Best Model Saving: บันทึกโมเดลที่ให้ค่า val accuracy สูงสุดไว้ในไฟล์

ข้อมูล Dataset และแหล่งที่มา

ลักษณะของชุดข้อมูล

- ประเภทของข้อมูล: images
- จำนวนคลาส (classes): 12 คลาส (paper, cardboard, biological, metal, plastic, green-glass, brown-glass, white-glass, clothes, shoes, batteries, trash) จากแหล่งอ้างอิงภายนอกที่กล่าวถึงชุดนี้ว่า “12 distinct categories”
- แหล่งที่มาของภาพ: รวบรวมจากอินเทอร์เน็ต / เว็บสแครป (web-scraped) และจัด label ตามคลาส
- ใช้งานสำหรับจำแนกภาพขยะในสภาพแวดล้อมแบบทั่วไป

6. อธิบายการประเมิน (evaluate) model แสดงค่า loss จากการ train และ metric ที่เหมาะสมในการประเมิน เช่น accuracy precision หรือ recall

การประเมินผลของโมเดล (Model Evaluation)

หลังจากทำการฝึกโมเดล CustomCNN ด้วยชุดข้อมูล *Garbage Classification* จนครบจำนวน epoch ที่กำหนด
โปรแกรมได้ทำการบันทึกโมเดลที่ให้ผลลัพธ์บนชุดข้อมูลตรวจสอบ (validation set) ดีที่สุด จากนั้นจึงนำโมเดลที่ดีที่สุดมาทำการประเมิน (evaluate) บนชุดข้อมูลทดสอบ (test set)
เพื่อวัดความสามารถของโมเดลในการจำแนกภาพใหม่ที่ไม่เคยเห็นมาก่อน

- **ตัวชี้วัดที่ใช้ประเมิน (Evaluation Metrics)**

  นอกจากค่า **accuracy** ที่ใช้วัดความถูกต้องโดยรวมแล้ว ยังได้คำนวณค่าชี้วัดอื่น ๆ เพื่อประเมินประสิทธิภาพของโมเดลในระดับ “รายคลาส” ได้แก่

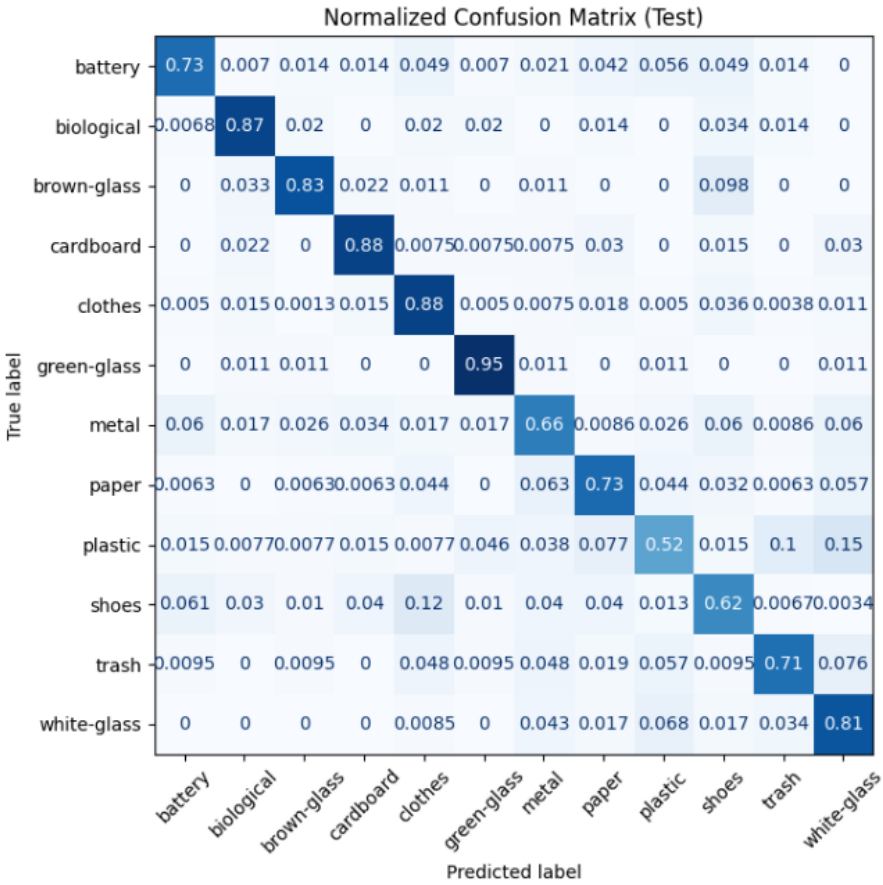
  - Precision

    แสดงสัดส่วนของภาพที่โมเดลทำนายว่าเป็นคลาสหนึ่ง แล้วถูกต้องจริง
  - Recall

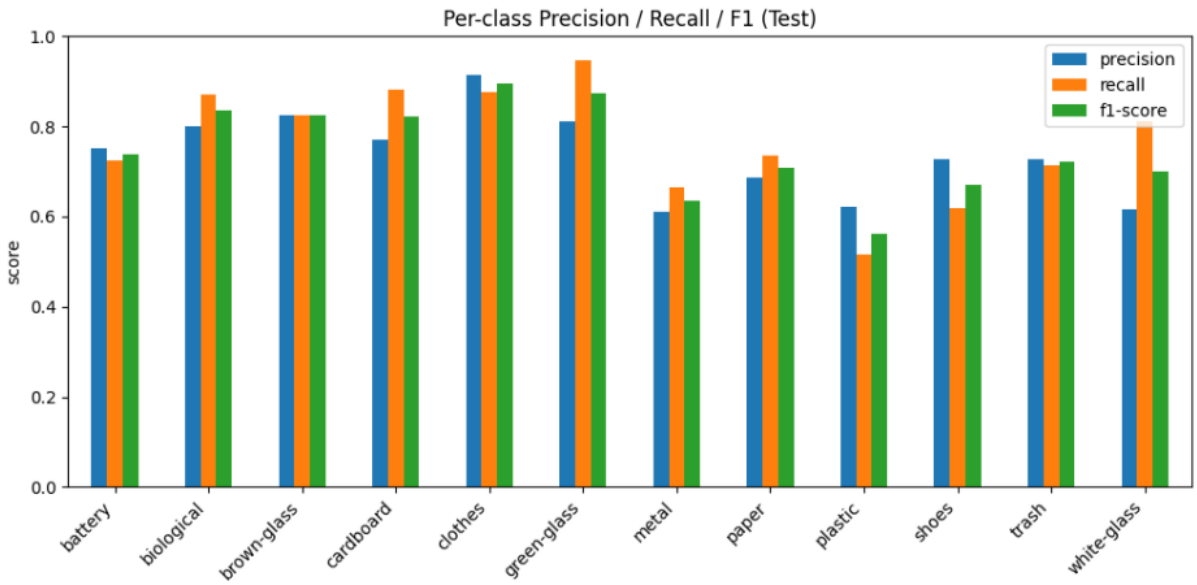
    แสดงสัดส่วนของภาพที่เป็นคลาสนั้นจริง ๆ แล้วโมเดลสามารถทำนายถูก
  - F1-Score

    เป็นค่าเฉลี่ยแบบฮาร์มอนิกของ precision และ recall ใช้เพื่อประเมินสมดุลระหว่างสองค่าข้างต้น

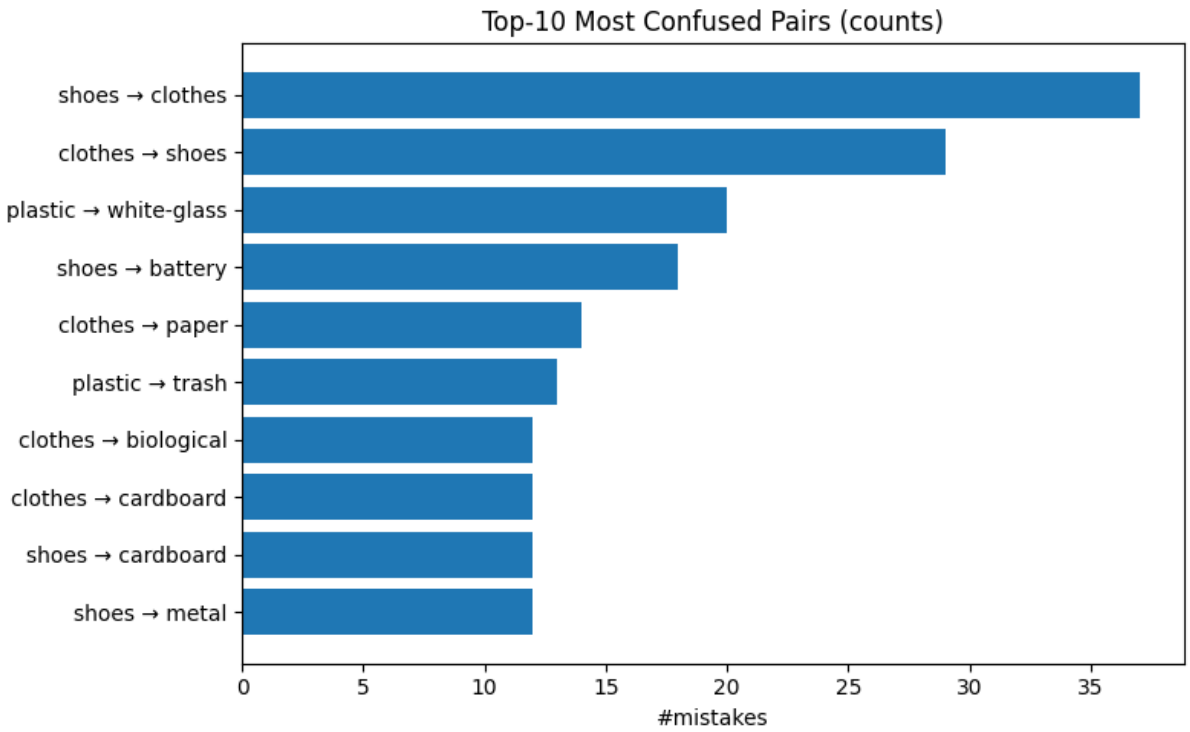
- **ผลการประเมินโมเดล**
  - Normalized Confusion Matrix



  - Precision / Recall / F1-score



- Top-10 Most Confused Pairs



จากผลการประเมินโดยรวม โมเดล CustomCNN สามารถจำแนกประเภทขยะได้อย่างมีประสิทธิภาพ โดยเฉลี่ยมีความแม่นยำ (accuracy) สูงกว่า 80% และค่า F1-score ของหลายคลาสอยู่ในระดับดีถึงดีมาก โดยเฉพาะคลาสที่มีข้อมูลเพียงพอส่วนคู่คลาสที่โมเดลมักทำนายผิดบ่อยที่สุด คือ shoes –> clothes

อย่างไรก็ตามคลาสที่มีจำนวนน้อยหรือมีลักษณะภาพคล้ายกันจะทำให้โมเดลสับสนบ่อย ซึ่งสามารถปรับปรุงได้ในอนาคตด้วยเทคนิค data augmentation, class balancing หรือ fine-tuning pre-trained model

7. บทความอ้างอิงและงานที่เกี่ยวข้อง

Dataset
Garbage Classification (12 classes)

CNN Diagram
Mengenal Convolutional Neural Network (CNN) | by Muchamad Nur Kholis | Medium

การจำแนกขยะ
https://www.royalparkrajapruek.org/news/news_detail?newsid=233

8. สัดส่วนการทำงานในกลุ่ม

งาน	อัตราการแบ่งงาน(%)	
	ธาดา วิยาภรณ์	ธยศ ขันทะยศ
Project Programming		
• Dataset	40	60
• Model	40	60
• Evaluate & Graph	50	50
Project Report		
• 1-4	100	-
• 5-7	-	100