Workshop II: Object Detection

Assignment Workshop 02

1. ให้นักศึกษาทำโจทย์ของไฟล์ Workshop_II-02_ObjectDetection.ipynb ให้เรียบร้อยพร้อมทั้งมาตอบคำถามต่อไปนี้

1.1 อธิบายวิธีการเตรียมมข้อมูล (Data Preparation) ของนักศึกษาที่ใช้ในการจัดการข้อมูลก่อนที่จะนำเทรนโมเดลครั้งนี้ สามารถแคปภาพหน้าจอแล้วมาอธิบายได้

```python
class SyntheticDectectionDataset(Dataset):
    def __init__(self, images_path, annotation_files, transform=None):
        self.coco = COCO(annotation_files)
        self.images_path = images_path
        self.transform = transform
        self.image_ids = list(self.coco.imgs.keys())
```

สร้างคลาส `SyntheticDectectionDataset`:

- **โหลดข้อมูล `images` และ `annotations` จากไฟล์ JSON โดยใช้ `pycocotools.COCO`**
- **ดึงข้อมูล `image` ในรูปแบบ grayscale และ normalize ให้มีค่าอยู่ในช่วง [0, 1]**
- **แปลง bounding box ให้อยู่ในรูปแบบ normalized (0 ถึง 1) ตามขนาดภาพ**
- **เก็บข้อมูล bounding boxes และ labels ไว้ใน dictionary**

```python
# Set paths
train_dir = "/content/train"
test_dir = "/content/test"
val_dir = "/content/val"

train_anns = "/content/train.json"
test_anns = "/content/test.json"
val_anns = "/content/val.json"

train_transform = transforms.Compose([
    transforms.ToTensor()
])

test_transform = transforms.Compose([
    transforms.ToTensor()])

val_transform = transforms.Compose([
    transforms.ToTensor()])
```

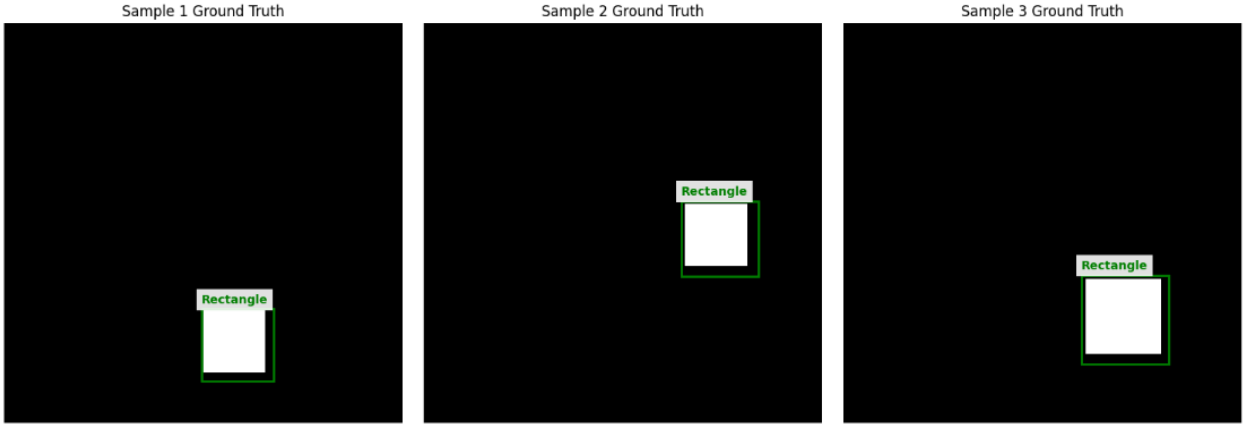
โหลด dataset ตาม path และกำหนดวิธีการ transforms

```python
# Create datasets
train_dataset = SyntheticDectectionDataset(train_dir, train_anns, train_transform)
test_dataset = SyntheticDectectionDataset(test_dir, test_anns, test_transform)
val_dataset = SyntheticDectectionDataset(val_dir, val_anns, val_transform)


# Create dataloaders
train_loader = DataLoader(train_dataset, batch_size=4, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=4, shuffle=False)
val_loader = DataLoader(val_dataset, batch_size=4, shuffle=False)
```

สร้าง Dataloader:

- **ใช้ `DataLoader` สำหรับการโหลดข้อมูลแบบ batch**
- **กำหนด batch size เท่ากับ 4 และสุ่มลำดับข้อมูล (`shuffle=True`) ในชุด Train**



สำรวจข้อมูล:

ใช้ฟังก์ชัน plot_samples_from_dataloader เพื่อแสดงตัวอย่างภาพและ bounding box ของข้อมูล Ground Truth พร้อม label

1.2 ทำการสร้างและปรับแต่งโมเดลตามโครงสร้างที่กำหนด (สามารถปรับโครงสร้างได้ตามความคิดสร้างสรรค์ พร้อมอธิบายการออกแบบโมเดล) จากนั้นทำการเทรนโมเดล โดยเก็บค่า Training Loss และ Validation Loss นำมาสร้างกราฟ พร้อมวิเคราะห์ว่าโมเดลมีการ Overfitting หรือ Underfitting หรือไม่ พร้อมอธิบายเหตุผล

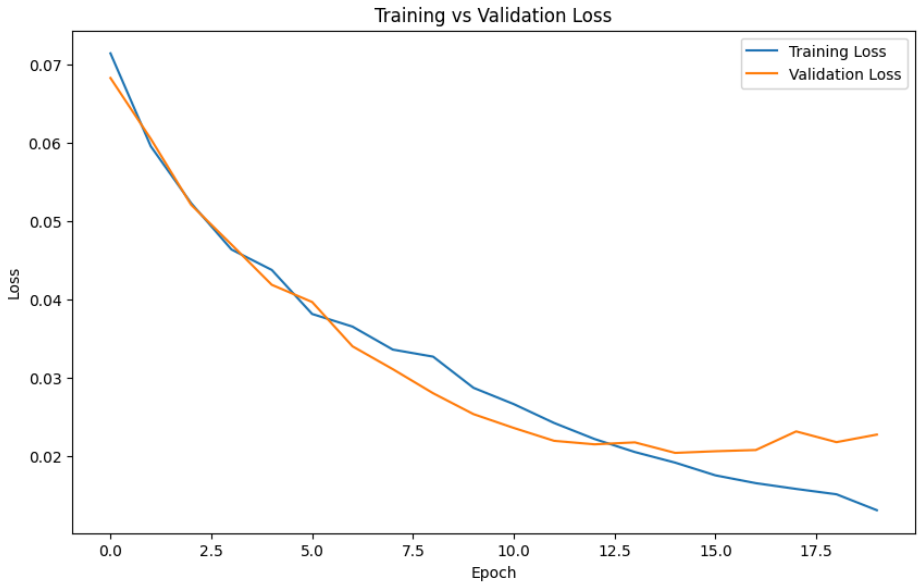
```python
self.conv1 = nn.Conv2d(1, 32, kernel_size=3, padding=1)
self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
self.fc1 = nn.Linear(16 * 16 * 64, 128)
self.fc2 = nn.Linear(128, 5) # class, x, y, w, h
```

โมเดลที่สร้างขึ้น (SimpleObjectDetector) เป็น Convolutional Neural Network (CNN) ธรรมดา ที่มีโครงสร้าง

- **มี 2 ชั้น Convolution (`conv1`, `conv2`) ตามด้วย MaxPooling**
- **Fully Connected Layer (`fc1`, `fc2`) สำหรับ output: `[class, x, y, w, h]`**
- **Activation Functions: ReLU และ Sigmoid**

Layer (type)	Output Shape
Conv2d-1	[-1, 32, 64, 64]
MaxPool2d-2	[-1, 32, 32, 32]
Conv2d-3	[-1, 64, 32, 32]
MaxPool2d-4	[-1, 64, 16, 16]
Linear-5	[-1, 128]
Linear-6	[-1, 5]

ใช้ Mean Squared Error (MSE) เป็น Loss Function ในการ Train
ใช้ Adam เป็น Optimizer



จากกราฟ Training vs Validation Loss:

- **ค่า Loss ลดลงอย่างต่อเนื่อง แสดงว่าโมเดลกำลังเรียนรู้ได้ดี**

- **Validation Loss ต่ำกว่า Training Loss ในหลายช่วง แสดงว่าไม่มี overfitting เกิดขึ้น**
- **กราฟ Loss เริ่มเข้าสู่จุด Plateau หลัง Epoch ที่ 15-20 บ่งชี้ว่าโมเดลเริ่ม convergence**

1.3 ทำการวัดผลของโมเดลด้วยการใช้คะแนน IOU, Precision, Recall และ F1-Score

```
# Create test function for predict test_loader
def test_model(model, test_loader):
    model.eval()  # Set the model to evaluation mode
    predictions = []
    ground_truths = []
```

ประเมินผลโมเดลด้วย test_model

```
return predictions, ground_truths
```

เก็บค่าการทำนาย (`predictions`) และ Ground Truth (`ground_truths`) ไว้สำหรับการวิเคราะห์ภายหลัง

```
# IoU function to compute intersection over union
def compute_iou(pred_bbox, gt_bbox):
    # Calculate intersection coordinates
    x1 = max(pred_bbox[0], gt_bbox[0])
    y1 = max(pred_bbox[1], gt_bbox[1])
    x2 = min(pred_bbox[0] + pred_bbox[2], gt_bbox[0] + gt_bbox[2])
    y2 = min(pred_bbox[1] + pred_bbox[3], gt_bbox[1] + gt_bbox[3])
```

```
# Calculate IoU
iou = intersection_area / union_area if union_area > 0 else 0.0
return iou
```

คำนวณ IoU (Intersection over Union) ระหว่าง Bounding Box ที่โมเดลทำนาย (`pred_bbox`) กับ Ground Truth (`gt_bbox`)

```
def calculate_performance(predictions, ground_truths, iou_threshold=0.5):
    TP = 0
    FP = 0
    FN = 0
    iou_values = []
```

```
precision = TP / (TP + FP) if (TP + FP) > 0 else 0.0
recall = TP / (TP + FN) if (TP + FN) > 0 else 0.0
f1_score = 2 * (precision * recall) / (precision + recall) if (precision + recall) > 0 else 0.0
mean_iou = np.mean(iou_values)

return precision, recall, f1_score, mean_iou
```

คำนวณ Mean IOU, Precision, Recall และ F1-Score:
เมื่อ IoU >= Threshold (เช่น 0.5) ถือว่าการทำนายเป็นบวก (Positive)

```
Precision: 1.0000
Recall: 0.7333
F1-Score: 0.8462
Mean IoU: 0.5609
```

```
ious = [compute_iou(pred[:4], gt[:4]) for pred, gt in zip(predictions, ground_truths)]
```

สำรวจ Distribution ของ IoU:

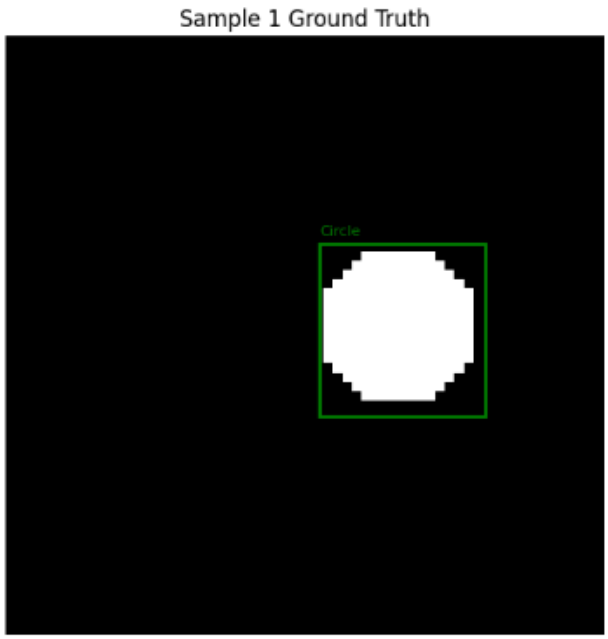
```
IoU Distribution: [0.0, 0.05025739, 0.0, 0.0, 0.5442371, 0.0, 0.3056161, 0.0, 0.16802335, 0.0, 0.0, 0.0, 0.3787551, 0.23775522, 0.15072472]
```

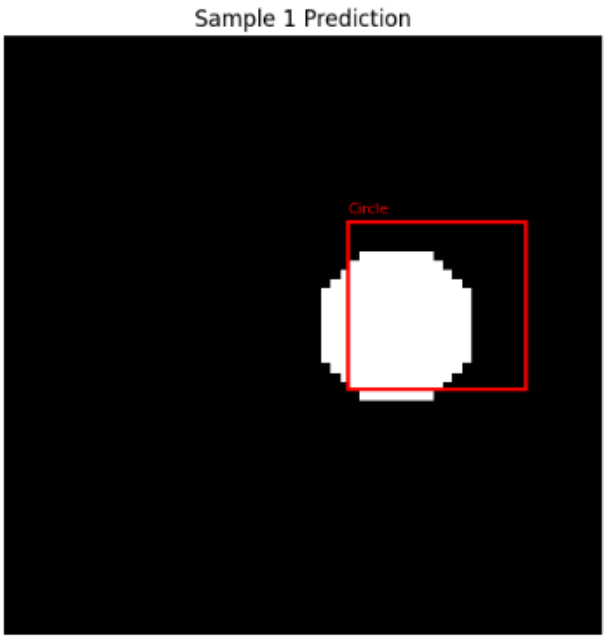
1.4 ทำการแสดงผลของการทำนายด้วยโมเดลเทียบระหว่างภาพ bounding box ของ labels และ predictions

```
image = images[i].cpu().numpy().squeeze()
gt_bboxes = targets['bboxes'][i].cpu().numpy()
gt_labels = targets['labels'][i].cpu().numpy()
pred_bbox = outputs[i, 1:].cpu().detach().numpy()
pred_class = outputs[i, 0].cpu().detach().numpy()
```

Extract Predictions and Ground Truth:

- **Bounding Boxes และ Labels ของ Ground Truth มาจาก `targets`**
- **ผลทำนาย Bounding Boxes (`outputs[i, 1:]`) และ Class (`outputs[i, 0]`) มาจากโมเดล**





Plot Ground Truth and Predictions:

- **Ground Truth: ใช้สีเขียว (`g`) สำหรับ Bounding Box และ Label**
- **Predictions: ใช้สีแดง (`r`) พร้อมแสดงผล Label ที่ทำนาย (คลาส `Circle` มีค่า < 0.5 และ `Rectangle` ≥ 0.5)**