Object Detection Fast R-CNN

1) 폴더 안 파일 리스트 가져오기

```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/global-wheat-detection/sample_submission.csv
/kaggle/input/global-wheat-detection/train.csv
/kaggle/input/global-wheat-detection/test/796707dd7.jpg
/kaggle/input/global-wheat-detection/test/2fd875eaa.jpg
/kaggle/input/global-wheat-detection/test/cc3532ff6.jpg
/kaggle/input/global-wheat-detection/test/53f253011.jpg
/kaggle/input/global-wheat-detection/test/f5a1f0358.jpg
/kaggle/input/global-wheat-detection/test/51f1be19e.jpg
/kaggle/input/global-wheat-detection/test/aac893a91.jpg
/kaggle/input/global-wheat-detection/test/cb8d261a3.jpg
/kaggle/input/global-wheat-detection/test/51b3e36ab.jpg
/kaggle/input/global-wheat-detection/test/348a992bb.jpg
/kaggle/input/global-wheat-detection/train/944c60a15.jpg
/kaggle/input/global-wheat-detection/train/dd5dd0234.jpg
/kaggle/input/global-wheat-detection/train/72f8aaa4f.jpg
/kaggle/input/global-wheat-detection/train/69595016d.jpg
/kaggle/input/global-wheat-detection/train/28b8ba0aa.jpg
/kaggle/input/global-wheat-detection/train/2d635716b.jpg
/kaggle/input/global-wheat-detection/train/1f548e2f2.jpg
/kaggle/input/global-wheat-detection/train/47b0d504c.jpg
/kaggle/input/global-wheat-detection/train/e34336025.jpg
/kaggle/input/global-wheat-detection/train/2d990708e.jpg
```

/kaggle/input/global-wheat-detection/train 사진 : 3422

/kaggle/input/global-wheat-detection/test 사진 : 10

파일, 폴더 형태

kaggle/input - global-wheat-detection
- sample_submission.csv
- train.csv
- train
- test
- fastrcnn.path

2) csv 불러오기

```
import cv2
import os
import re
import torch
import torchvision
```

```
from torchvision import transforms
from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
from torchvision.models.detection import FasterRCNN
from torchvision.models.detection.rpn import AnchorGenerator
from torch.utils.data import DataLoader, Dataset
from torch.utils.data.sampler import SequentialSampler
from matplotlib import pyplot as plt

WEIGHTS_FILE = '/kaggle/input/fasterrcnn/fasterrcnn_resnet50_fpn_best.pth'
train_df = pd.read_csv("/kaggle/input/global-wheat-detection/train.csv")
submit = pd.read_csv("/kaggle/input/global-wheat-detection/sample_submission.csv")
```

```
train_df.head()
```

	image_id	width	height	bbox	source
0	b6ab77fd7	1024	1024	[834.0, 222.0, 56.0, 36.0]	usask_1
1	b6ab77fd7	1024	1024	[226.0, 548.0, 130.0, 58.0]	usask_1
2	b6ab77fd7	1024	1024	[377.0, 504.0, 74.0, 160.0]	usask_1
3	b6ab77fd7	1024	1024	[834.0, 95.0, 109.0, 107.0]	usask_1
4	b6ab77fd7	1024	1024	[26.0, 144.0, 124.0, 117.0]	usask_1

train_df 가 가지고 있는 columns : image_id, width, height, bbox, source
우리는 image_id와 bbox만 쓸 거라서 나머지 열 제거 후 진행

```
train_df=train_df.drop(columns=['width','height','source'])
```

위 그림을 보면 image_id가 겹치는 게 많은데, 같은 그림 안에 다른 bbox를 그렸다. 한 이미지 안에 가장 많은 bbox와 적은 bbox를 알아보면 다음과 같다.

```
train_df['image_id'].nunique()
(train_df['image_id'].value_counts()).max()
(train_df['image_id'].value_counts()). min()



3373
116
1
```

3) 바운딩 박스 x,y,h,w 컬럼 생성

```
train_df['x'] = -1
train_df['y'] = -1
train_df['w'] = -1
train_df['h'] = -1

def expand_bbox(x):
    r = np.array(re.findall("([0-9]+[.]?[0-9]*)", x))
    if len(r) == 0:
        r = [-1, -1, -1, -1]
    return r
train_df[['x', 'y', 'w', 'h']] = np.stack(train_df['bbox'].apply(lambda x: expand_bbo
x(x)))
train_df['x'] = train_df['x'].astype(np.float)
train_df['y'] = train_df['y'].astype(np.float)
train_df['w'] = train_df['w'].astype(np.float)
train_df['h'] = train_df['h'].astype(np.float)
```

이렇게 하면 분리가 되기는 한데 무슨 말인지 잘 모르겠어서 다음과 같이 코딩하였다.

```
for row in range(int(train_df.shape[0])):
    train_df['x'] = train_df['bbox'][row].split(',')[0][1:]
    train_df['y'] = train_df['bbox'][row].split(',')[1][1:]
    train_df['w'] = train_df['bbox'][row].split(',')[2][1:]
    train_df['h'] = train_df['bbox'][row].split(',')[3][1:-1]
train_df['x'] = train_df['x'].astype(np.float)
train_df['y'] = train_df['y'].astype(np.float)
train_df['w'] = train_df['w'].astype(np.float)
train_df['h'] = train_df['h'].astype(np.float)
```

위 코딩의 단점은 시간이 상당히 많이 걸린다는 것이다. for문으로 반복했는데 147793 행만큼 반복을 해서 시간이 엄청 오래 걸렸다.

	image_id	bbox	x	y	w	h
0	b6ab77fd7	[834.0, 222.0, 56.0, 36.0]	834.0	222.0	56.0	36.0
1	b6ab77fd7	[226.0, 548.0, 130.0, 58.0]	226.0	548.0	130.0	58.0
2	b6ab77fd7	[377.0, 504.0, 74.0, 160.0]	377.0	504.0	74.0	160.0
3	b6ab77fd7	[834.0, 95.0, 109.0, 107.0]	834.0	95.0	109.0	107.0
4	b6ab77fd7	[26.0, 144.0, 124.0, 117.0]	26.0	144.0	124.0	117.0

무튼 잘 분리된 것을 확인하였다.

4) train, validation set 나누기

```
image_ids = train_df['image_id'].unique() #3373
valid_ids = image_ids[-665:] # 665
train_ids = image_ids[:-665] # 2708

valid_df = train_df[train_df['image_id'].isin(valid_ids)]
train_df = train_df[train_df['image_id'].isin(train_ids)]
```

train_ids(80%) + valid_ids(20%) = 2708 + 665 = 3373 (image_id unique의 개수)
train_df.shape # (122787,6)
valid_df.shape # (25006,6)

● 하나 문제라고 생각하는 점 : random이 아니라 순서대로 해서 잘라냄

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

5) 데이터셋 정의

```
# 클래스 정의
class WheatDataset(Dataset):

    def __init__(self, dataframe, image_dir, transforms=None,train=True):
        super().__init__()

        self.image_ids = dataframe['image_id'].unique()
        self.df = dataframe
        self.image_dir = image_dir
        self.transforms = transforms
        self.train=train # totensor

    def __len__(self) -> int:
        return self.image_ids.shape[0] #이미지 개수

    def __getitem__(self, index: int):

        image_id = self.image_ids[index]
        image = cv2.imread(f'{self.image_dir}/{image_id}.jpg', cv2.IMREAD_COLOR)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB).astype(np.float32)
        image /= 255.0 # 정규화
        if self.transforms is not None:
```

```
            image = self.transforms(image)
        if(self.train==False):  # train이 아님, test
            return image, image_id
        # train과 validation이 아님
        records = self.df[self.df['image_id'] == image_id]
        boxes = records[['x', 'y', 'w', 'h']].values
        boxes[:, 2] = boxes[:, 0] + boxes[:, 2] # x+w
        boxes[:, 3] = boxes[:, 1] + boxes[:, 3] # y+h
        boxes = torch.as_tensor(boxes, dtype=torch.float32)

        area = (boxes[:, 3] - boxes[:, 1]) * (boxes[:, 2] - boxes[:, 0])
        area = torch.as_tensor(area, dtype=torch.float32)

        # there is only one class
        labels = torch.ones((records.shape[0],), dtype=torch.int64)

        # suppose all instances are not crowd
        iscrowd = torch.zeros((records.shape[0],), dtype=torch.int64)

        target = {}
        target['boxes'] = boxes
        target['labels'] = labels
        target['image_id'] = torch.tensor([index])
        target['area'] = area
        target['iscrowd'] = iscrowd

        return image, target,image_id
```

```
train_dir = '/kaggle/input/global-wheat-detection/train'
test_dir = '/kaggle/input/global-wheat-detection/test

class Averager:        ##Return the average loss
    def __init__(self):
        self.current_total = 0.0
        self.iterations = 0.0

    def send(self, value):
        self.current_total += value
        self.iterations += 1

    @property
```

```
    def value(self):
        if self.iterations == 0:
            return 0
        else:
            return 1.0 * self.current_total / self.iterations

    def reset(self):
        self.current_total = 0.0
        self.iterations = 0.0


def collate_fn(batch):
    return tuple(zip(*batch))

train_dataset = WheatDataset(train_df, train_dir, trans,True)
valid_dataset = WheatDataset(valid_df, train_dir, trans,True)


# split the dataset in train and test set
indices = torch.randperm(len(train_dataset)).tolist()

train_data_loader = DataLoader(
    train_dataset,
    batch_size=16,
    shuffle=False,
    num_workers=4,
    collate_fn=collate_fn
)

valid_data_loader = DataLoader(
    valid_dataset,
    batch_size=8,
    shuffle=False,
    num_workers=4,
    collate_fn=collate_fn
)

device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
```

6) 확인

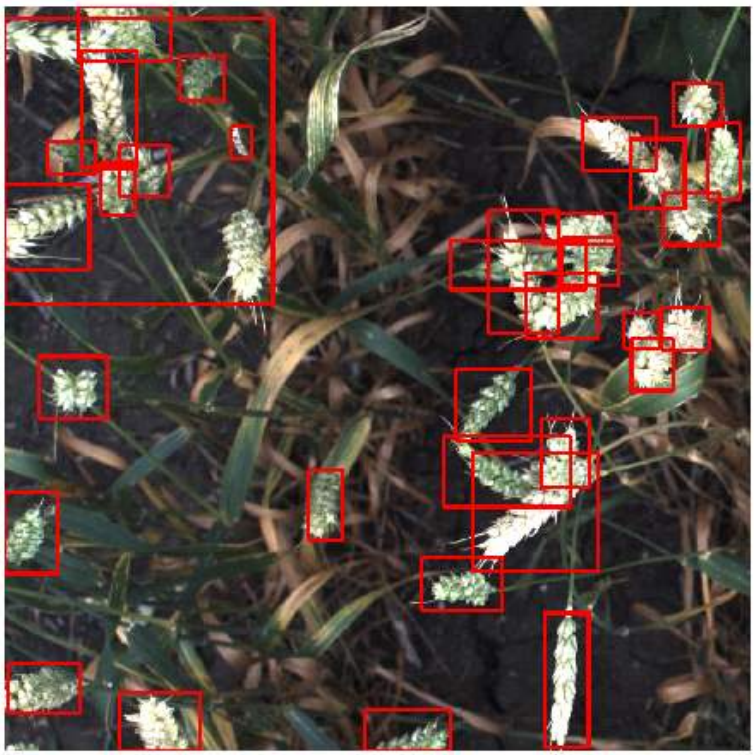
```
images, targets, image_ids = next(iter(train_data_loader)) # 데이터 로더 반복
images = list(image.to(device) for image in images)
targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

boxes = targets[4]['boxes'].cpu().numpy().astype(np.int32)
sample = images[4].permute(1,2,0).cpu().numpy()

fig, ax = plt.subplots(1, 1, figsize=(16, 8))

for box in boxes:
    cv2.rectangle(sample,
                  (box[0], box[1]),
                  (box[2], box[3]),
                  (220, 0, 0), 3)

ax.set_axis_off() #서브 플롯 축 안보이게 하기
ax.imshow(sample)
```



7) Finetuning Model

정의된 모형 : Fast R-CNN

```
# load a model; pre-trained on COCO
```

```
model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=False, pretrained_backbone=False)
```

COCO로 pretrained 되어진 Fast R-CNN 모형을 가져와서 모형으로 정의한다.
pretrained_backbone=False : If True, returns a model with backbone pre-trained on Imagenet
그 뒤에 출력할 마지막 층을 변경한다.

```
num_classes = 2  # 1 class (wheat), 0 background
in_features = model.roi_heads.box_predictor.cls_score.in_features
model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)

model.load_state_dict(torch.load(WEIGHTS_FILE))
#model.eval()

#x = model.to(device)
```

https://tutorials.pytorch.kr/intermediate/torchvision_tutorial.html
㉠ in_feature : 분류기에서 사용할 입력 특징의 차원 정보를 얻는다.
㉡ model.roi_heads.bax_predictor : 미리 학습된 모델의 머리 부분을 새로운 것으로 교체한다.

8) Train model

```
model.train()
model.to(device)
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.01, momentum=0.9, weight_decay=0.00001)
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=1, gamma=0.5)
#lr_scheduler = None

num_epochs = 5

loss_hist = Averager()
itr = 1

for epoch in range(num_epochs):
    loss_hist.reset() # 초기화
```

```
    for images, targets, image_ids in train_data_loader:

        images = list(image.to(device) for image in images)
        targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

        loss_dict = model(images, targets)

        losses = sum(loss for loss in loss_dict.values())
        loss_value = losses.item()

        loss_hist.send(loss_value)  #Average out the loss

        optimizer.zero_grad()
        losses.backward()
        optimizer.step()

        if itr % 50 == 0:
            print(f"Iteration #{itr} loss: {loss_value}")

        itr += 1

    # update the learning rate
    if lr_scheduler is not None:
        lr_scheduler.step()

    print(f"Epoch #{epoch} loss: {loss_hist.value}")
```

```
Iteration #50 loss: 0.7108762860298157
Iteration #100 loss: 0.559785783290863
Iteration #150 loss: 0.5037466883659363
Epoch #0 loss: 0.6033961573067833
Iteration #200 loss: 0.6382746696472168
```

9) Prediction

```
test_dataset = WheatDataset(submit,test_dir, trans,False)
test_data_loader = DataLoader( test_dataset, batch_size=8, shuffle=False)
detection_threshold = 0.45

def format_prediction_string(boxes, scores): ## Define the formate for storing prediction results
```

```
    pred_strings = []
    for j in zip(scores, boxes):
        pred_strings.append("{0:.4f} {1} {2} {3} {4}".format(j[0], j[1][0], j[1][1], j[1][2], j[1][3]))

    return " ".join(pred_strings)
## Lets make the prediction
results=[]
model.eval()

for images, image_ids in test_data_loader:

    images = list(image.to(device) for image in images)
    outputs = model(images)

    for i, image in enumerate(images):

        boxes = outputs[i]['boxes'].data.cpu().numpy()
        scores = outputs[i]['scores'].data.cpu().numpy()

        boxes = boxes[scores >= detection_threshold].astype(np.int32)
        scores = scores[scores >= detection_threshold]
        image_id = image_ids[i]

        boxes[:, 2] = boxes[:, 2] - boxes[:, 0]
        boxes[:, 3] = boxes[:, 3] - boxes[:, 1]



        result = {
            'image_id': image_id,
            'PredictionString': format_prediction_string(boxes, scores)
        }



        results.append(result)

test_df = pd.DataFrame(results, columns=['image_id', 'PredictionString'])
test_df.head()
sample = images[1].permute(1,2,0).cpu().numpy()
```

```
boxes = outputs[1]['boxes'].data.cpu().numpy()
scores = outputs[1]['scores'].data.cpu().numpy()

boxes = boxes[scores >= detection_threshold].astype(np.int32)
```

10) 시각화 및 결과 저장

```
fig, ax = plt.subplots(1, 1, figsize=(16, 8))

for box in boxes:
    cv2.rectangle(sample,
                  (box[0], box[1]),
                  (box[2], box[3]),
                  (220, 0, 0), 2)

ax.set_axis_off()
ax.imshow(sample)
test_df.to_csv('submission.csv', index=False) # 제출용
```

Reference
https://www.kaggle.com/heojihye/getting-started-with-object-detection-with-pytorch/edit