2021 Small Object AI Challenge

TRACK 1. 소형객체 AI 모델 개발
데이터 및 코드 소개

NSML Baseline Code Guide

1.Overview

- 문제 : 고해상도 이미지 내에서 소형객체를 탐지하는 딥러닝 모델 개발

- 목표 : 최종적으로 이미지 내의 여러 객체의 bounding box좌표 (x,y,w,h)와
  해당 객체의 클래스 번호를 제출해야 함

- 대회방식 및 규칙
  1. 훈련에 pretrained 모델 (웨이트)는 쓸 수 없습니다.
  2. 제공한 train 데이터로만 훈련해야 합니다.

2.Data Information

- **정의: 이미지 1장당 소형객체(100x100 미만) 1~5개가 분포되어 있음**

이미지	17173 장
클래스	30개
해상도	2800 px x 2100 px
객체크기	10 px ~ 100 px

- **클래스 번호 : 0~29**

['SD카드','웹캠','OTP','계산기','목걸이','넥타이핀','십원','오십원','백원','오백원','미국지폐','유로지폐','태국지폐','필리핀지폐','밤','브라질너트','은행','피칸','호두','호박씨','해바라기씨','줄자','건전지','망치','못','나사못','볼트','너트','타카','베어링']

2.Data Information

- **Label 형식**

Sample.json


```
{"0.jpg": [[8.0, 1029.0, 1011.0, 146.0, 197.0], [7.0, 1044.0, 823.0, 186.0, 106.0]]
, "1.jpg": [[26.0, 1534.0, 1161.0, 78.0, 77.0], [26.0, 1423.0, 963.0, 69.0, 70.0],
[26.0, 1231.0, 1196.0, 77.0, 77.0]], "2.jpg": [[19.0, 934.0, 653.0, 165.0, 128.0],
[19.0, 1004.0, 1002.0, 92.0, 178.0], [19.0, 1002.0, 1367.0, 205.0, 115.0], [19.0,
1658.0, 1348.0, 187.0, 99.0], [19.0, 1624.0, 1034.0, 145.0, 132.0]], "3.jpg": [[3.
0, 1122.0, 612.0, 182.0, 143.0], [3.0, 1543.0, 556.0, 178.0, 101.0], [3.0, 1506.0,
902.0, 228.0, 132.0]], "4.jpg": [[22.0, 1839.0, 654.0, 117.0, 111.0], [22.0, 1229.
0, 640.0, 117.0, 107.0]], "5.jpg": [[17.0, 1795.0, 956.0, 96.0, 91.0], [17.0, 2018.
0, 905.0, 121.0, 145.0]], "6.jpg": [[3.0, 1306.0, 1562.0, 170.0, 156.0], [3.0,
1647.0, 1755.0, 169.0, 198.0], [3.0, 1078.0, 1746.0, 164.0, 197.0]], "7.jpg": [[3.
0, 1092.0, 910.0, 105.0, 104.0], [3.0, 1254.0, 1006.0, 123.0, 91.0]], "8.jpg": [[7.
0, 495.0, 459.0, 131.0, 111.0], [8.0, 635.0, 419.0, 142.0, 123.0], [8.0, 409.0,
591.0, 151.0, 130.0]], "9.jpg": [[3.0, 1561.0, 980.0, 84.0, 64.0], [3.0, 1583.0,
615.0, 80.0, 65.0], [3.0, 1551.0, 1276.0, 89.0, 93.0]], "10.jpg": [[22.0, 1092.0,
868.0, 163.0, 160.0]], "12.jpg": [[0.0, 993.0, 1083.0, 169.0, 126.0]], "13.jpg": [
```

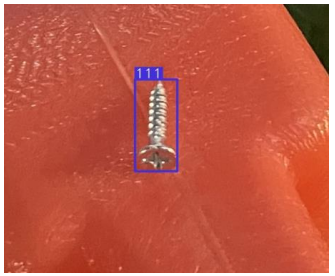

'file_name':[[cls_num, x, y, w, h]]

Ex) 0.jpg에 객체가 2개 있다.

< 이미지 예시 >

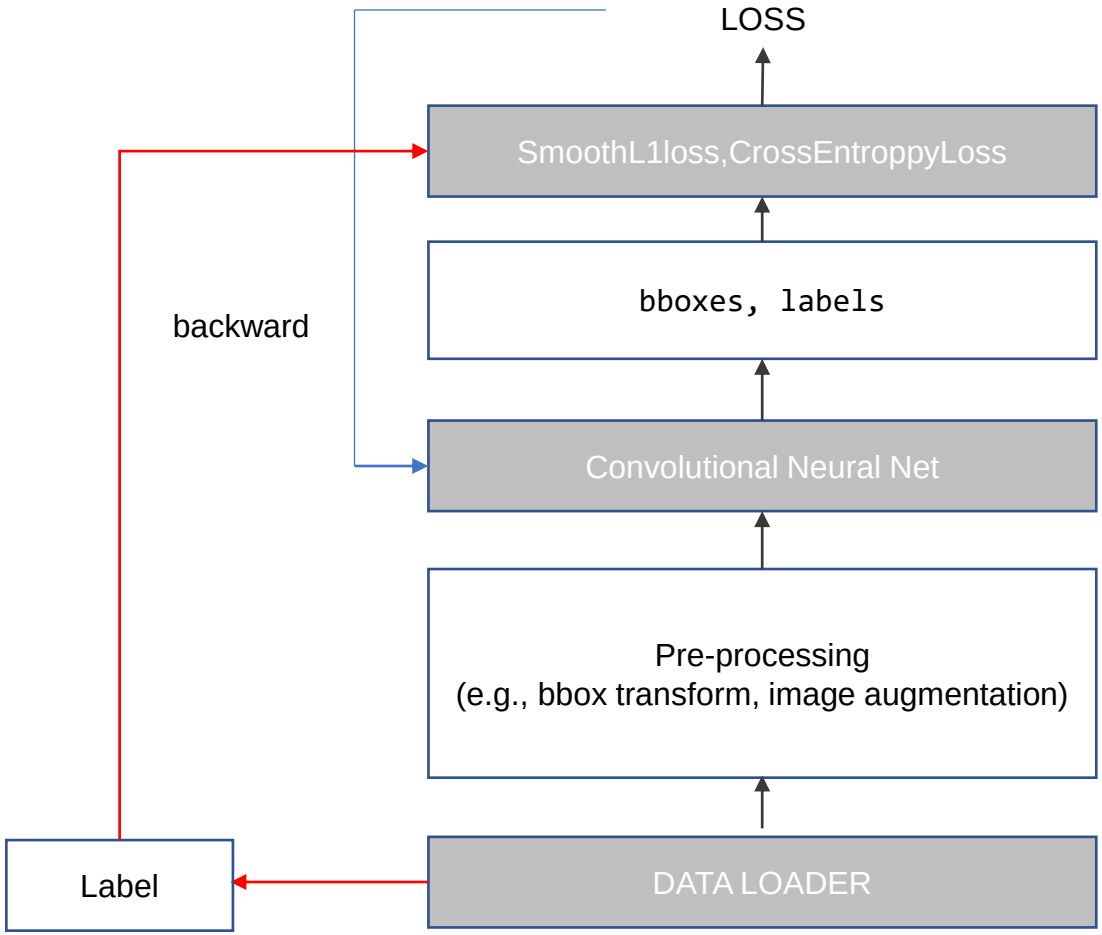
< Sample image >

< 확대 >



3.Baseline Code

- **Process**



3.Baseline Code

- **Main.py**

```
# nsml
import nsml
from nsml import DATASET_PATH
```

①

②

```
def bind_model(model):
    def save(dir_path, **kwargs):
        checkpoint = {
            "model": model.state_dict()}
        torch.save(checkpoint, os.path.join(dir_path, 'model.pt'))
        print("model saved!")

    def load(dir_path):
        checkpoint = torch.load(os.path.join(dir_path, 'model.pt'))
        model.load_state_dict(checkpoint["model"])
        print('model loaded!')
```

- **① 데이터셋 경로입니다.**

학습용 이미지의 경로는

DATASET_PATH + “train/train_data” 입니다.

- **② 학습 된 모델을 저장하고 불러오는 함수입니다.**

nsml.save(epoch) 를 통해 매 epoch마다 모델의

파라미터를 저장합니다.

저장 된 모델은 nsml submit 시 불러와 사용합니다.

3.Baseline Code

- **Main.py**

```
③ # data_loader에서 인자 받음
  def infer(test_img_path_list):

      반환 형식 준수해야 정상적으로 score가 기록됩니다.
      {'file_name':[[cls_num, x, y, w, h, conf]]}
      '''
      result_dict = {}

      # for baseline model ==============================
      import torchvision.transforms as transforms
      from PIL import Image
      from tqdm import tqdm

      infer_transforms = transforms.Compose([
              transforms.Resize((300,300)),
              transforms.ToTensor(),
              transforms.Normalize(mean=[0.5], std=[0.5])
          ])
      dboxes = generate_dboxes()
      # inference시 박스 좌표로 후처리하는 모듈
      encoder = Encoder(dboxes)

      model.cuda()
      model.eval()

      for _, file_path in enumerate(tqdm(test_img_path_list)):
          file_name = file_path.split("/")[-1]
          img = Image.open(file_path)
          width, height = img.size

          img = test_preprocessing(img, infer_transforms)
          img = img.cuda()
          detections = []

          with torch.no_grad():
              ploc, plabel = model(img)
              ploc, plabel = ploc.float().detach().cpu(), plabel.float().detach().cpu()

              try:
                  result = encoder.decode_batch(ploc, plabel, 0.5, 100)[0]
              except:
                  print("No object detected : ", file_name)
                  continue

              loc, label, prob = [r.numpy() for r in result]
              for loc_, label_, prob_ in zip(loc, label, prob):
                  try:
                      '''
                      결과 기록 형식 준수해야 함
                      pred_cls, x, y, w, h, confidence
                      '''
                      detections.append([
                          int(label_)-1,
                          float( loc_[0] * width ),
                          float( loc_[1] * height ),
                          float( (loc_[2] - loc_[0]) * width ),
                          float( (loc_[3] - loc_[1]) * height ),
                          float( prob_ )
                          ])
                  except:
                      continue

          result_dict[file_name] = detections # 반환 형식 준수해야 함
④     return result_dict

  # DO NOT CHANGE: They are reserved for nsml
  nsml.bind(save=save, load=load, infer=infer)
```

- **③ infer()**

Infer()는 nsml submit 명령어 이용 시 pause가 걸려 test data_loader로 부터 인자를

받아옵니다. (train data_loader와 다르며 볼 수 없음)

따라서 참가자는 반환 형식을 동일하게 맞춰 점수가 측정될 수 있도록

합니다.

- **④ nsml submit 시 반환되는 output 입니다.**

모델에 맞추어 자유롭게 작성하시면 되나,

Output 형식은 반환 형식 준수해야 정상적으로 score가 기록됩니다.

`{'file_name':[[cls_num, x, y, w, h, conf]]}` 를 사용합니다.

Ex)

{'2133.jpg': [[19, 2040.0216579437256, 912.6190602779388, 810.861349105835, 720.7046627998352, 0.116291344165802], [19, 1852.2767066955566, 286.19741946458817, 567.3027038574219, 394.56382244825363, 0.12294214963912964]]}

3.Baseline Code

- **Main.py**

```
def get_args():
    parser = ArgumentParser(description="NSML BASELINE")
    parser.add_argument("--epochs", type=int, default=30, help="number of total epochs to run")
    parser.add_argument("--batch-size", type=int, default=8, help="number of samples for each iteration")
    parser.add_argument("--lr", type=float, default=0.001, help="initial learning rate")
    parser.add_argument("--nms-threshold", type=float, default=0.5)
    parser.add_argument("--num-workers", type=int, default=0)
①
    # DONOTCHANGE: They are reserved for nsml
    parser.add_argument("--pause", type=int, default=0)
    parser.add_argument('--mode', type=str, default='train', help='submit일때 test로 설정됩니다.')
    parser.add_argument('--iteration', type=str, default='0', help='fork 명령어를 입력할때의 체크포인트로 설정됩니다.')
    args = parser.parse_args()
    return args
```

- **① nsml submit 시 필요한 인자입니다.**

박스 안의 인자는 수정하시면 안됩니다.

ArgumentParser() 를 통해 parser를 생성해 add_argument를 이용해 입력 받고자 하는 인자의 조건을 설정합니다.

parser.parse_arg()를 통해 인자들을 파싱하여 args에 저장합니다.

3.Baseline Code

- **Main.py**

```
def main(opt):

    torch.manual_seed(123)
①   num_class = 30 # 순수한 데이터셋 클래스 개수

②   # baseline model
    dboxes = generate_dboxes()
    model = BASELINE_MODEL(num_classes=num_class+1) # 배경 class 포함 모델
    optimizer = torch.optim.Adam(model.parameters(), lr=opt.lr, betas=(0.937, 0.999))
    scheduler = None

    bind_model(model)

③   if opt.pause:
        nsml.paused(scope=locals())
    else:
        # loss
④       criterion = Loss(dboxes)

        # train data
        with open(os.path.join(DATASET_PATH, 'train', 'train_label'), 'r', encoding="utf-8") as f:
            train_data_dict = json.load(f)
            train_img_label = prepocessing(root_dir=os.path.join(DATASET_PATH, 'train', 'train_data'),\
                label_data=train_data_dict,output_path=cache_file_path, input_size=(300,300))

        train_params = {"batch_size": opt.batch_size,
                        "shuffle": True,
                        "drop_last": False,
                        "num_workers": opt.num_workers,
                        "collate_fn": collate_fn}

        # data loader
        train_data = Small_dataset(train_img_label, num_class, BaseTransform(dboxes))
        train_loader = DataLoader(train_data, **train_params)

        model.cuda()
        criterion.cuda()

        for epoch in range(0, opt.epochs):
            train_loss = train(model, train_loader, epoch, criterion, optimizer, scheduler)
            nsml.report(
                epoch=epoch,
                epoch_total=opt.epochs,
                batch_size=opt.batch_size,
                train_loss=train_loss)
            nsml.save(epoch)

if __name__ == "__main__":
    opt = get_args()
    main(opt)
```

- **① 데이터셋의 클래스 개수**

데이터셋의 클래스 개수에 맞게 변경하셔야 합니다.

- **② 모델 설정**

baseline 모델은 ssd(백본 + extra layers)를 이용, ssd는 배경 class 를

+1 카운팅 하여 클래스 개수를 설정합니다.

본인의 모델에 맞게 설정 합니다. NSML 에서 필요한

load(), save(), infer() 가 있는 bind_model로 감싸줍니다.

- **③ nsml submit 할 경우 실행 됩니다.**

- **④ 이미지 캐싱**

baseline model의 이미지 캐싱 방법 : 데이터셋이 고해상도 이미지이기

때문에 훈련 소요 시간을 줄이고자 훈련 시작 전에 원본을 미리 resize

를 해서 로드함.

(사용하지 않아도 무방함)

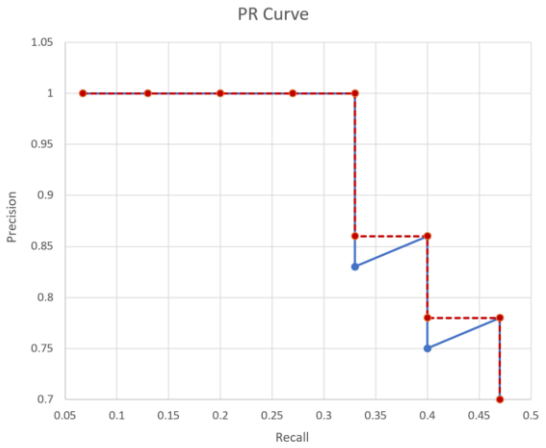
4.성능평가지표

*** 세부 조건**
_ IoU threshold : 0.5
_ max detections : 이미지 1장당 100개 (이미지당 제한 개수 초과하면 임의로 제외함)

- Mean Average Precisione (mAP)
  물체 클래스가 여러 개인 경우 각 클래스의 AP를 모두 합한 후 물체 클래스의 개수로 나눠 계산

$$\mathrm{m}AP = \frac{\sum \text{클래스의 } AP}{\text{클래스 개수}}$$

- Average Precisione (AP)
  Precision-Recall 그래프에서 그래프 선 아래 쪽의 면적으로 계산



4.성능평가지표

- Recall (sensitivity)
  재현율이란 실제 True인 것 중에서 모델이 True라고 예측

$$Recall = \frac{TP}{TP + FN}$$

- Precision (Positive Predictive Value)
  정밀도란 모델이 True라고 분류한 것 중에서 실제 True라고 예측

$$Precision = \frac{TP}{TP + FP}$$

TRACK 1. 소형객체 AI 모델 개발
데이터 및 코드 소개

NSML Baseline Code Guide