# **2023 제3회 K-water AI경진대회 모델 설명서**

**팀 명 : 조안나 맛있다**

**2023년** 11**월 13일**

1. **라이브러리 및 데이터 (Library & Data)**

* Ubuntu: Pytorch '2.1.0+cu121'
* Ultralytics 및 나머지는 requirements.txt에 첨부
* training data의 경우 image로 되어있고, 참고 데이터는 동영상의 형태로 되어있음

1. **데이터 전처리 (Data Cleansing & Pre-Processing)**

* Ultralytics/ultralytics/data/augment.py에 구현되어있음
* Mosaic
  1. Mosaic 증강을 통해 모델에 다양한 상황을 부여할 수 있음
* Mixup
  1. Mixup을 통해 객체 간 분류를 과하게 하는 것 방지
* Custom Cut Out
  1. 기존의 Cutout의 경우, Albumentations를 이용해 구현함
     + 이미지 내 랜덤한 위치에서 0픽셀을 적용하게 됨
     + 실험 결과 객체 / 배경에 대한 분류를 더 못하게 되는 상황이 발생함.
     + ‘객체 안에서만 Cutout을 적용하면 좋겠다’ 라는 아이디어에서 착안한 Custom Cutout 개발
  2. bounding box에 대응되는 이미지 영역 일부를 0픽셀로 만듦으로써 모델이 과하게 분류하는 것을 방지함
* Random Flip
  1. 수조에 찍히는 이미지들은 고정이므로 좌우 Flip만을 적용
* Random HSV
  1. 시간대가 다양하기 때문에 HSV값을 변경하는 증강 추가함
* Blur & MedianBlur & ToGray & CLAHE
  1. 이미지가 뿌옇거나 시간대가 밤이어서 회색조인 상황 등이 존재하기 때문에, 확률적으로 Blur, Median Blur, ToGray, CLAHE 등의 증강을 넣어줌

|  |  |
| --- | --- |
| |  | | --- | | *def* v8\_transforms(dataset, imgsz, hyp, stretch=False):  “””Convert images to a size suitable for YOLOv8 training.”””  pre\_transform = *Compose([*  Mosaic(dataset, imgsz = imgsz, p = hyp.mosaic),  CustomCutout(p = 1.0),  CopyPaste(p = hyp.copy\_paste),  RandomPerspective(  degrees = hyp.degrees,  translate = hyp.translate,  scale = hyp.scale,  shear = hyp.shear,  perspective = hyp.perspective,  pre\_transform = None if stretch else LetterBox(new\_shape = (imgsz, imgsz))  flip\_idx = dataset.data.get(‘flip\_idx’, [])# for keypoints augmentation  *if* dataset.use\_keypoints:  Kpt\_shape = dataset.data.get(‘kpt\_shape’, None)  *if*  len(flip\_idx) == 0 and hyp.fliplr > 0.0:  hyp.fliplr = 0.0  LOGGER.warning(“WARNING ⚠️ No ‘flip\_idx’ array defined in data.yaml, setting augmentation ‘fliplr=0.0”)  *elif* flip\_idx and (len(flip\_idx) != kpt\_shape[0]):  raise ValueError(f’daata.yaml flip\_idx={flip\_idx} length must be equal to kpt\_shape[0]={kpt\_shape[0]}’)  return Compose([  pre\_transform,  MixUp(dataset, pre\_transform = pre\_transform, p = hyp.mixup),  Albumentations1(p=1.0),  RandomHSV(hgain=hyp.hsv\_h, sgain=hyp.hsv\_s, vgain=hyp.hsv\_v),  RandomFlip(direction=’vertical’, p=hyp.flipud),  RandomFlip(direction=’horizontal’, p =hyp.fliplr, flip\_idx = flip\_idx)]) | |

1. **탐색적 자료 분석 (Exploratory Data Analysis)**

* 배경 이미지 : 객체 존재 이미지 = 95000 : 5000으로 극단적인 상황
* 객체처럼 보이지만 배경으로 간주하는 Noise 데이터가 많이 존재함 (좌측 그림)
  1. 이러한 배경의 학습 필요성 존재함 (training data 전부 사용의 근거)
* Class Imbalance 문제가 존재함 (우측 그림)
  1. 객체 검출에서 불균형 상황 해소를 위해 등장한 Focal Loss 적용의 근거

|  |
| --- |
| *.* |

1. **변수 선택 및 모델 구축 (Feature Engineering & Initial Modeling)**

* Ultralytics에서 개발한 YOLOv8을 선정함.
  1. Yolov5를 개선하여 만든 모델
  2. 특히 C2f 모듈을 통해 특징은 더 잘 추출하면서, 파라미터 수를 줄임
  3. 실시간성 및 성능 모두 확보 가능함
* 그 중에서 Open Image V7로 사전학습된 YOLOv8\_m 모델을 선정함.
  1. Fish class가 존재하기에 어종 특징 파악을 더 잘할 수 있음
  2. 추가적으로, 앞쪽 3개의 Layer를 Freeze함으로써 Fine-tuning 시 파라미터 줄임
  3. Label smoothing을 적용하여 모델이 하나의 클래스에 과도하게 확신하는 것을 방지하고자 함.

|  |  |
| --- | --- |
| YOLOv8_structure_v1.6   |  | | --- | | *class* C3(nn.Module):  # CSP Bottleneck with 3 convolutions  *def* \_\_init\_\_(self, c1, c2, n=1, shortcut=True, g=1, e=0.5): # ch\_in, ch\_out, number, shortcut, groups, expansion  *super*().\_\_init\_\_()  c\_ = *int*(c2 \* e) # hidden channels  self.cv1 = Conv(c1, c\_, 1, 1)  self.cv2 = Conv(c1, c\_, 1, 1)  self.cv3 = Conv(2 \* c\_, c2, 1) # optional act=FReLU(c2)  self.m = nn.Sequential(\*(Bottleneck(c\_, c\_, shortcut, g, e=1.0) for \_ in *range*(n)))  *def* forward(self, x):  return self.cv3(torch.cat((self.m(self.cv1(x)), self.cv2(x)), 1))  *class* C2f(nn.Module):  # CSP Bottleneck with 2 convolutions  *def* \_\_init\_\_(self, c1, c2, n=1, shortcut=False, g=1, e=0.5): # ch\_in, ch\_out, number, shortcut, groups, expansion  *super*().\_\_init\_\_()  self.c = *int*(c2 \* e) # hidden channels  self.cv1 = Conv(c1, 2 \* self.c, 1, 1)  self.cv2 = Conv((2 + n) \* self.c, c2, 1) # optional act=FReLU(c2)  self.m = nn.ModuleList(Bottleneck(self.c, self.c, shortcut, g, k=((3, 3), (3, 3)), e=1.0) for \_ in *range*(n))  *def* forward(self, x):  y = *list*(self.cv1(x).split((self.c, self.c), 1))  y.extend(m(y[-1]) for m in self.m)  return self.cv2(torch.cat(y, 1)) | |

1. **모델 학습 및 검증 (Model Tuning & Evaluation)**

* 데이터를 학습과 검증으로 나누고 두 가지 측면에서 모델의 성능을 검증함
  1. validation data에 대해 loss가 잘 떨어지고, mAP가 올라가는지
  2. CAM(Class Activation Mapping)을 이용하여, 모델이 우리가 원하는 객체를 잘 보고 있는지 시각화 (위에서 아래 그림으로 가면서 개선되는 모습)
* 참고 데이터의 경우, 동영상의 형태로 되어있었는데, 이를 프레임 단위로 이미지화하고, training data를 모두 학습한 가장 좋은 모델(epoch60.pt)로 inference하여, pseudo-labeling을 생성함.
* 또한, 데이터를 점차 늘려가며 실험을 진행함
  1. 객체가 있는 이미지만 사용
  2. 객체가 있는 이미지 : 배경 이미지 = 1:1이 되도록 배경 추가하여 실험
  3. 2번 상황에서 참고 데이터셋으로 받은 동영상 psuedo labeling 후 객체 이미지만 추가하여 재학습
  4. Training data 모두 사용
  5. 4번 상황에서 참고 데이터셋으로 받은 동영상 pseudo labeling 후 객체 이미지만 추가하여 재학습

→ 1에서 5로 갈수록 성능이 점진적으로 향상되는 것 확인

* 최종적으로, training data + pseudo-labeling을 이용하여 모델을 학습(epoch90.pt)하였을 때, 가장 좋은 성능을 보여주는 것을 확인함.

|  |  |
| --- | --- |
| |  | | --- | | **# ttach 설치**  **# !pip install ttach 설치**  **import ultralytics**  **from ultralytics import YOLO**  **import warnings**  **warnings.filterwarnings('ignore')**  **warnings.simplefilter('ignore')**  **import torch**  **import cv2**  **import numpy as np**  **import matplotlib.pyplot as plt**  **import requests**  **import torchvision.transforms as transforms**  **from PIL import Image**  **import io**  **plt.rcParams["figure.figsize"] = [3.0, 3.0]**  **from yolo\_cam.eigen\_cam import EigenCAM**  **from yolo\_cam.utils.image import show\_cam\_on\_image, scale\_cam\_image**  **from glob import glob**  **model7 = YOLO('models/46last.pt') # 모델 path 넣어주기**  **target\_layers7 =[model7.model.model[-4]] # conv layer 넣어주기**  **img = cv2.imread('/notebooks/datasets/train/images/train\_40000.png') # train image path 넣어주기**  **img = cv2.resize(img, (640, 640))**  **rgb\_img = img.copy()**  **img = np.float32(img) / 255**  **cam = EigenCAM(model7, target\_layers7,task='od')**  **grayscale\_cam = cam(rgb\_img)[0, :, :]**  **cam\_image = show\_cam\_on\_image(img, grayscale\_cam, use\_rgb=True)**  **im = cv2.cvtColor(rgb\_img, cv2.COLOR\_RGB2BGR)**  **Image.fromarray(np.hstack((im, cam\_image)))** | |

1. **결과 및 결언 (Conclusion & Discussion)**

* Public과 Private 모두에서 강건한 모델을 만들었음.
* 특히, Custom Cut Out을 적용하여 라벨 한쪽만 학습하여 편향되는 모델을 기피할 수 있었음. 이 부분이 모델의 강건함을 만들었다고 생각함.
* 따로 Ensemble을 적용하지 않고 빠른 YOLO를 통해 실시간 검출 가능.
* 현재 Train 라벨의 경우 물고기의 일부만 보일 때 라벨이 존재하지 않는데 라벨링이 추가적으로 수행된다면 모델의 성능이 향상될 수 있음.
* 시간이 더 주어지면 객체 검출 Ensemble 방법인 WBF(Weighted Boxes Fusion)을 통해 더욱 Robust한 모델을 만들 수 있음.