

code review

[Inference]

git 구성

<https://github.com/RSNAPOC2021/poc1/tree/main>

RSNAPOC2021 / poc1 Private Watch 0 Fork

Code Issues Pull requests Actions Projects Wiki Security Insights Setting

main ▾ 2 branches 0 tags Go to file Add file ▾ Code ▾

leekate	inference folder commit	92fbacb 18 minutes ago	35 commits
Yolo_EfficientNet_	ROI crop 코드 추가	7 days ago	
Yolo_EfficientNet_Inference	inference folder commit	18 minutes ago	
README.md	README update	7 days ago	

Yolo_EfficientNet_Inference 구성 및 진행 순서

[폴더]

dataset: 프로젝트에 사용할 모든 데이터

models: yolo models

utils: yolo utils

weights: train 결과물 pt파일

[이미지]

black.png: 이미지 crop, resize용

Yolo_EfficientNet_Inference 구성 및 진행 순서

[코드]

data_gen.py:

dataset 폴더에서 필요한 데이터를 필요한 폴더로 복사하는 코드

bounding_box_crop.py:

yolo inference 결과물로 만들어진 바운딩박스를 자르고, efficientnet inference에 필요한 사이즈로 resize하는 코드

Yolo_Efficient_inference_image.py:

이미지 하나씩 yolo-crop-resize-efficientnet 진행

+ 최종 csv 파일 작성 코드

A	B	C	D	E	F
img_name	prediction	Negative for Pneumonia	Typical Appearance	Indeterminate Appearance	Atypical Appearance
00000001_000-crop.png	Negative for Pneumonia	0.90145284	0.03508554	0.048705816	0.016322895
00000001_001-crop.png	Negative for Pneumonia	0.9341554	0.014215147	0.019472033	0.03215734
00000001_002-crop.png	Typical Appearance	0.018052375	0.8054181	0.15171266	0.024816832
00000002_000-crop.png	Typical Appearance	0.09750793	0.808047	0.07355589	0.02088925
00000003_000-crop.png	Negative for Pneumonia	0.9130746	0.01371222	0.052219454	0.020993756
00000003_001-crop.png	Negative for Pneumonia	0.9482369	0.01862332	0.022300016	0.010839734
00000003_002-crop.png	Negative for Pneumonia	0.9800847	0.005151562	0.009058088	0.005705531
00000003_003-crop.png	Negative for Pneumonia	0.81564146	0.036714032	0.13130932	0.014535208
00000003_004-crop.png	Negative for Pneumonia	0.48281038	0.4494023	0.04791225	0.019875022
00000003_005-crop.png	Negative for Pneumonia	0.8498921	0.025677934	0.09607066	0.02835936
00000003_006-crop.png	Negative for Pneumonia	0.807123	0.06913641	0.11888384	0.004856698
00000003_007-crop.png	Negative for Pneumonia	0.9735791	0.003419584	0.0068040861	0.014960443
00000004_000-crop.png	Negative for Pneumonia	0.8542615	0.005879921	0.1350534	0.004805235
00000005_000-crop.png	Negative for Pneumonia	0.9981834	0.000297072	0.000641035	0.000878488
00000005_001-crop.png	Negative for Pneumonia	0.7394481	0.16618563	0.064970076	0.029396232
00000005_002-crop.png	Negative for Pneumonia	0.42280686	0.34325278	0.16437617	0.069564186
00000005_003-crop.png	Negative for Pneumonia	0.42760745	0.7995102	0.100567766	0.39187378
00000005_004-crop.png	Negative for Pneumonia	0.98389775	0.001731803	0.006476831	0.007893628
00000005_005-crop.png	Negative for Pneumonia	0.94005185	0.00074124	0.00113879	0.058068164
00000005_006-crop.png	Negative for Pneumonia	0.8543313	0.02069701	0.012718328	0.11225335
00000005_007-crop.png	Negative for Pneumonia	0.9027193	0.002138512	0.046842758	0.048299365

Yolo_EfficientNet_Inference 구성 및 진행 순서

[기타]

`data_total.py`: 데이터셋이 여러 폴더로 나뉘어져 있을 때 한 폴더로 합치는 코드

`check_shape.py`: `resize` 전후 `shape`을 확인하는 코드

`efficient_model.py`: efficientnet train을 진행하는 model 코드

`except_data_check.py`: Inference 결과물 csv에서 빠진 데이터가 있는지 확인하는 코드

`Yolo_Efficient_inference_one.py`:

이미지 하나 yolo-crop-resize-efficientnet 코드 + 최종 csv 파일 작성 코드

`Yolo_Efficient_inference_function.py`:

이미지 전체가 yolo-crop-resize-efficientnet + 최종 csv 파일 작성 코드

Yolo_EfficientNet_Inference : main code review

data_gen.py: dataset 폴더에서 필요한 데이터를 필요한 폴더로 복사하는 코드

```
1 import os
2 import shutil
3
4 data_path = "D:/NIH-git/poc1/Yolo_EfficientNet_Inference/dataset/TOTAL"
5 data_list = os.listdir(data_path)
6
7 #dst = "D:/NIH-git/poc1/Yolo_EfficientNet_Inference/dataset/effi/"
8 dst = "D:/NIH-git/poc1/Yolo_EfficientNet_Inference/dataset/yolo/"
9 #dst = "D:/NIH-git/poc1/Yolo_EfficientNet_Inference/dataset/yolo_effi/"
10
11 for i in range(1000):
12     img_path = data_path + data_list[i]
13     print(img_path)
14     print(dst+data_list[i])
15     print()
16     shutil.copyfile(img_path, dst+data_list[i])
```


Yolo_EfficientNet_Inference : main code review

bounding_box_crop.py:

yolo inference 결과물로 만들어진 바운딩박스를 자르고, efficientnet inference에 필요한 사이즈로 resize하는 코드

```
1 import os
2 import numpy as np
3 import cv2
4
5 #파일내 이미지 하나 crop and resize
6 def crop_merge_one(img):
7     #print("\ncrop_merge_one")
8     #print(img)
9     back_dir = './black.png'
10    back = cv2.imread(back_dir)
11    #print(back.shape)
12
13    try:
14        if img.endswith("yolo.png"):
15            name = img[:-9]
```

yolo inference의 결과물로 만들어진 이미지 불러오기

*yolo inference의 결과물은 000.png -> 000-yolo.png로 저장

yolo inference의 결과물로 만들어진 이미지 원본이름 찾기

Yolo_EfficientNet_Inference : main code review

bounding_box_crop.py:

yolo inference 결과물로 만들어진 바운딩박스를 자르고, efficientnet inference에 필요한 사이즈로 resize하는 코드

```
17     # bounding box 위치 찾기
18     image = cv2.imread(img)
19     image = cv2.resize(image, dsize=(380, 380), interpolation=cv2.INTER_AREA)
20     image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
21     image_gray = cv2.resize(image_gray, dsize=(380, 380), interpolation=cv2.INTER_AREA)

22
23     image_hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
24     rect_img = cv2.inRange(image_hsv, (30, 100, 0), (90, 255, 255))
25     cor = np.where(rect_img > 0)

26
27     cor0_set = set(cor[0])
28     cor0_list = list(cor0_set)
29     cor1_set = set(cor[1])      rgb -> hsv로 수정해
30     cor1_list = list(cor1_set)  바운딩박스를 찾아내고 모든 좌표를 저장하는 코드
31
```

*RGB 이미지에서 색 정보를 검출하기 위해서는 R, G, B 세가지 속성을 모두 참고해야 하는데,

HSV 이미지에서는 H(Hue)가 일정한 범위를 갖는 순수한 색 정보를 가지고 있기 때문에 RGB 이미지보다 쉽게 색을 분류할 수 있다.

Yolo_EfficientNet_Inference : main code review

bounding_box_crop.py:

yolo inference 결과물로 만들어진 바운딩박스를 자르고, efficientnet inference에 필요한 사이즈로 resize하는 코드

```
32
33     # yolo bounding box 없으면
34     if (len(cor0_list) ==0 & len(cor1_list)==0):
35         back_resize = image
36         cv2.imshow('crop', back_resize)
37         cv2.imwrite(name + "-crop.png", back_resize)
38         #print(name + "-crop.png")
39
```

Yolo_EfficientNet_Inference : main code review

bounding_box_crop.py:

yolo inference 결과물로 만들어진 바운딩박스를 자르고, efficientnet inference에 필요한 사이즈로 resize하는 코드

```
43
44     else:
45         cor_list = []
46         y = []
47         for n in range(len(cor0_list)-1):
48             if cor0_list[n] + 2 > cor0_list[n+1]:
49                 y.append(cor0_list[n])
50                 #print("X",x)
51                 y1 = min(y)
52                 y2 = max(y)
53
54                 x = []
55                 for m in range(len(cor1_list)-1):
56                     if cor1_list[m] + 2 > cor1_list[m+1]:
57                         x.append(cor1_list[m])
58                         #print("y",y)
59                         x1 = min(x)
60                         x2 = max(x)
61
62                         box_org = [x1,x2,y1,y2]
63                         cor_list.append(box_org)
64
65                         mask = cor_list[0]
66                         x1 = mask[0]
67                         x2 = mask[1]
68                         y1 = mask[2]
69                         y2 = mask[3]
70                         h = mask[3] - mask[2]
71                         w = mask[1] - mask[0]
72                         #print(h)
73                         #print(w)
```

바운딩박스가 있다면 (=Negative for Pneumonia),
모든 좌표를 비교해 바운딩 박스의 최종 좌표를
구하는 코드

Yolo_EfficientNet_Inference : main code review

bounding_box_crop.py:

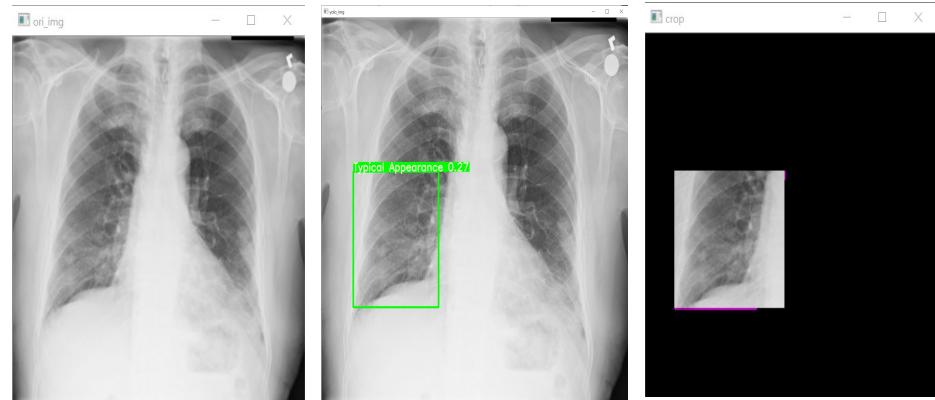
yolo inference 결과물로 만들어진 바운딩박스를 자르고, efficientnet inference에 필요한 사이즈로 resize하는 코드

```
74          # merge_img
75      if h > w:
76          ori_img = name + ".png"
77          ori_img = cv2.imread(ori_img)
78          ori_img = cv2.resize(ori_img, dsize=(380, 380), interpolation=cv2.INTER_AREA)
79          #cv2.imshow('ori_img', ori_img)
80
81          back_resize = cv2.resize(back, dsize=(w, h), interpolation=cv2.INTER_AREA)
82          ori_imgcopy = ori_img.copy()
83          image[y1:y2, x1:x2] = back_resize
84          crop = cv2.subtract(ori_imgcopy, image)
85          #cv2.imshow('crop', crop)
86          #cv2.waitKey(0)
87          cv2.imwrite(name + "-crop.png", crop)
88          #print(name + "-crop.png")
89
```

Yolo_EfficientNet_Inference : main code review

bounding_box_crop.py: yolo inference 결과물로 만들어진 바운딩박스를 자르고, efficientnet inference에 필요한 사이즈로 resize하는 코드

```
90         elif h < w:
91             ori_img = name + ".png"
92             ori_img = cv2.imread(ori_img)
93             ori_img = cv2.resize(ori_img, dsize=(380, 380), interpolation=cv2.INTER_AREA)
94             #cv2.imshow('ori_img', ori_img)
95
96             back_resize = cv2.resize(back, dsize=(w, h), interpolation=cv2.INTER_AREA)
97             ori_imgcopy = ori_img.copy()
98             image[y1:y2,x1:x2] = back_resize
99             #cv2.imshow('image', image)
100            crop = cv2.subtract(ori_imgcopy, image)
101            #cv2.imshow('crop', crop)
102            #cv2.waitKey(0)
103            cv2.imwrite(name + "-crop.png", crop)
104            #print(name + "-crop.png")
105
106        elif h == w:
107            ori_img = name + ".png"
108            ori_img = cv2.imread(ori_img)
109            ori_img = cv2.resize(ori_img, dsize=(380, 380), interpolation=cv2.INTER_AREA)
110
111            back_resize = cv2.resize(back, dsize=(w, h), interpolation=cv2.INTER_AREA)
112            ori_imgcopy = ori_img.copy()
113            image[y1:y2,x1:x2] = back_resize
114            crop = cv2.subtract(ori_imgcopy, image)
115            #cv2.imshow('crop', crop)
116            cv2.imwrite(name + "-crop.png", crop)
117            #print(name + "-crop.png")
118
119    except:
120        pass
121
122
123     return name
```



00000002_000

00000002_000-yolo

00000002_000-crop

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py: 이미지 하나씩 yolo-crop-resize-efficientnet 진행

```
1 import os
2 import albumentations as A
3 import cv2
4 import glob
5 import torch.nn.functional as F
6 import time
7 from albumentations.pytorch import ToTensorV2
8 import imutils
9 import pandas as pd
10 import argparse
11 import time
12 from pathlib import Path
13 import cv2
14 import torch
15 import torch.backends.cudnn as cudnn
16 from numpy import random
17
18 from efficient_model import Classifier
19 from models.experimental import attempt_load
20 from utils.datasets import LoadStreams, LoadImages
21 from utils.general import check_img_size, check_requirements, check_imshow, \
22     non_max_suppression, apply_classifier, scale_coords, xyxy2xywh, strip_optimizer, set_logging, increment_path
23 from utils.plots import plot_one_box, plot_one_box_only
24 from utils.torch_utils import select_device, load_classifier, time_synchronized
25 from bounding_box_crop import crop, merge_one
26
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo: 이미지 하나씩 yolo-crop-resize-efficientnet 진행

```
29 def Yolo():
30     parser = argparse.ArgumentParser()
31     parser.add_argument('--weights', nargs='+', type=str, default='yolov5-V5.0_best_weight_batch16_epoch20.pt',
32                         help='model.pt path(s)')
33     parser.add_argument('--source', type=str, default='D:/NIH-git/poc1/Yolo_EfficientNet_Inference/dataset/yolo', help='source')
34     parser.add_argument('--img-size', type=int, default=640, help='inference size (pixels)')
35     parser.add_argument('--conf-thres', type=float, default=0.25, help='object confidence threshold')
36     parser.add_argument('--iou-thres', type=float, default=0.45, help='IOU threshold for NMS')
37     parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
38     parser.add_argument('--view-img', action='store_true', help='display results')
39     parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
40     parser.add_argument('--save-conf', action='store_true', help='save confidences in --save-txt labels')
41     parser.add_argument('--nosave', action='store_true', help='do not save images/videos')
42     parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --class 0, or --class 0 2 3')
43     parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic NMS')
44     parser.add_argument('--augment', action='store_true', help='augmented inference')
45     parser.add_argument('--update', action='store_true', help='update all models')
46     parser.add_argument('--project', default='result', help='save results to project/name')
47     parser.add_argument('--name', default='exp', help='save results to project/name')
48     parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')
49     opt = parser.parse_args()
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo: 이미지 하나씩 yolo-crop-resize-efficientnet 진행

```
52     source, weights, view_img, save_txt, imgsz = opt.source, opt.weights, opt.view_img, opt.save_txt, opt.img_size
53     save_img = not opt.nosave and not source.endswith('.txt') # save inference images
54     webcam = source.isnumeric() or source.endswith('.txt') or source.lower().startswith(
55         ('rtsp://', 'rtmp://', 'http://', 'https://'))
56
57     # Directories
58     #save_dir = Path(increment_path(Path(opt.project) / opt.name, exist_ok=opt.exist_ok)) # increment run
59     #(save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True) # make dir
60     save_dir = opt.source
61
62     # Initialize
63     set_logging()
64     device = select_device(opt.device)
65     half = device.type != 'cpu' # half precision only supported on CUDA
66
67     # Load model
68     model = attempt_load(weights, map_location=device) # load FP32 model
69     stride = int(model.stride.max()) # model stride
70     imgsz = check_img_size(imgsz, s=stride) # check img_size
71     if half:
72         model.half() # to FP16
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo: 이미지 하나씩 yolo-crop-resize-efficientnet 진행

```
74     # Second-stage classifier
75     classify = False
76     if classify:
77         modelc = load_classifier(name='resnet101', n=2) # initialize
78         modelc.load_state_dict(torch.load('weights/resnet101.pt', map_location=device)['model']).to(device).eval()
79
80     # Set Dataloader
81     vid_path, vid_writer = None, None
82     if webcam:
83         view_img = check_imshow()
84         cudnn.benchmark = True # set True to speed up constant image size inference
85         dataset = LoadStreams(source, img_size=imgsz, stride=stride)
86     else:
87         dataset = LoadImages(source, img_size=imgsz, stride=stride)
88
89     # Get names and colors
90     names = model.module.names if hasattr(model, 'module') else model.names
91     colors = [[random.randint(0, 255) for _ in range(3)] for _ in names]
92
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo: 이미지 하나씩 yolo-crop-resize-efficientnet 진행

```
93     # Run inference
94     if device.type != 'cpu':
95         model(torch.zeros(1, 3, imgsz, imgsz).to(device).type_as(next(model.parameters())))  # run once
96         t0 = time.time()
97     for path, img, im0s, vid_cap in dataset:
98
99         name = path.split(".")[0]
100        #print(name)
101        img = torch.from_numpy(img).to(device)
102        img = img.half() if half else img.float()  # uint8 to fp16/32
103        img /= 255.0  # 0 - 255 to 0.0 - 1.0
104        if img.ndimension() == 3:
105            img = img.unsqueeze(0)
106
107        # Inference
108        t1 = time_synchronized()
109        pred = model(img, augment=opt.augment)[0]
110
111        # Apply NMS
112        pred = non_max_suppression(pred, opt.conf_thres, opt.iou_thres, classes=opt.classes, agnostic=opt.agnostic_nms)
113        t2 = time_synchronized()
114
115        # Apply Classifier
116        if classify:
117            pred = apply_classifier(pred, modelc, img, im0s)
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo: 이미지 하나씩 yolo-crop-resize-efficientnet 진행

```
119         # Process detections
120     for i, det in enumerate(pred): # detections per image
121         if webcam: # batch_size >= 1
122             p, s, im0, frame = path[i], '%g: ' % i, im0s[i].copy(), dataset.count
123         else:
124             p, s, im0, frame = path, '', im0s, getattr(dataset, 'frame', 0)
125
126         p = Path(p) # to Path
127         #print(p.name)
128         #(save_dir / p.name).mkdir(parents=True, exist_ok=True)
129         #save_path = str(save_dir / p.name / p.name) # img.jpg
130         save_path = save_dir
131         #txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image' else f'{frame}') # img.txt
132         txt_path = str(save_dir) + ('' if dataset.mode == 'image' else f'{frame}') # img.txt
133         # s += '%gx%g ' % img.shape[2:] # print string
134         gn = torch.tensor(im0.shape)[[1, 0, 1, 0]] # normalization gain whwh
135         if len(det):
136             # Rescale boxes from img_size to im0 size
137             det[:, :4] = scale_coords(img.shape[2:], det[:, :4], im0.shape).round()
138
139             # Print results
140             for c in det[:, -1].unique():
141                 n = (det[:, -1] == c).sum() # detections per class
142                 s += f"{n} {names[int(c)]}{s' * (n > 1)}, " # add to string
143
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo: 이미지 하나씩 yolo-crop-resize-efficientnet 진행

```
144          # Write results
145          for *xyxy, conf, cls in reversed(det):
146              if save_txt: # Write to file
147                  xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist() # normalized xywh
148                  line = (cls, *xywh, conf) if opt.save_conf else (cls, *xywh) # label format
149                  with open(txt_path + '.txt', 'a') as f:
150                      f.write(( '% ' * len(line)).rstrip() % line + '\n')
151
152          if save_img or view_img: # Add bbox to image
153              label = f'{names[int(cls)]} {conf:.2f}'
154              plot_one_box(xyxy, im0, label=label, color=colors[int(cls)], line_thickness=3)
155
156          # Print time (inference + NMS)
157          print(f'{s}Done. ({t2 - t1:.3f}s)')
158
159          # Stream results
160          if view_img:
161              cv2.imshow(str(p), im0)
162              cv2.waitKey(1) # 1 millisecond
163
```

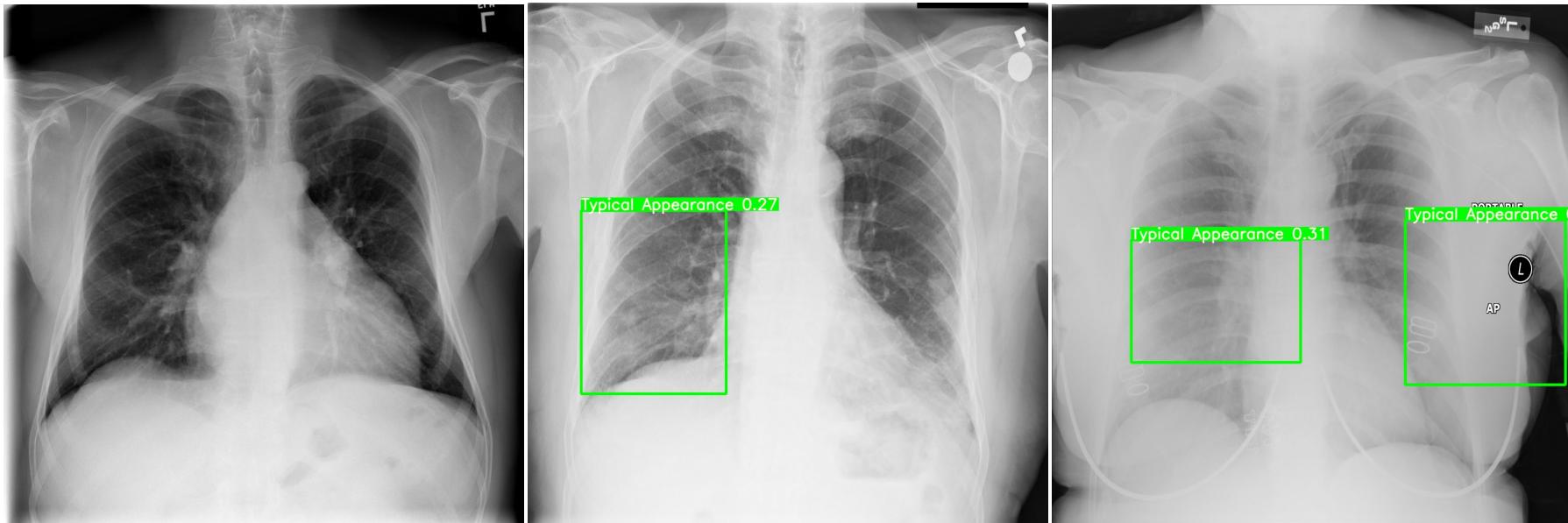
Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo: 이미지 하나씩 yolo-crop-resize-efficientnet 진행

```
159     # Stream results
160     if view_img:
161         cv2.imshow(str(p), im0)
162         cv2.waitKey(1) # 1 millisecond
163
164     # Save results (image with detections)
165     #cv2.imwrite(save_path, img) #원본
166     if save_img:
167         if dataset.mode == 'image':
168             #cv2.imwrite(save_path, im0)
169             #print(name + ".yolo.png")
170             cv2.imwrite(name + "-yolo.png", im0)
171
172     else: # 'video' or 'stream'
173         if vid_path != save_path: # new video
174             vid_path = save_path
175             if isinstance(vid_writer, cv2.VideoWriter):
176                 vid_writer.release() # release previous video writer
177             if vid_cap: # video
178                 fps = vid_cap.get(cv2.CAP_PROP_FPS)
179                 w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
180                 h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
181             else: # stream
182                 fps, w, h = 30, im0.shape[1], im0.shape[0]
183                 save_path += '.mp4'
184             vid_writer = cv2.VideoWriter(save_path, cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
185             vid_writer.write(im0)
186
187     if save_txt or save_img:
188         s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}" if save_txt else ''
189         print(f"Results saved to {save_dir}{s}")
190
191     print(f'Done. ({time.time() - t0:.3f}s)')
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo: 이미지 하나씩 yolo-crop-resize-efficientnet 진행



Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - EfficientNet: 이미지 하나씩 yolo-crop-resize-efficientnet 진행

```
199 def EfficientNet():
200     parser = argparse.ArgumentParser(description='PyTorch Classification')
201     parser.add_argument('--model_name', type=str, default="efficientnet_b4", metavar='S', help='model name')
202     parser.add_argument('--model_path', type=str, default=".//weights/efficientnet_b4_fold1_best_accuracy.pt", metavar='S',
203                         help='model path')
204     parser.add_argument('--num_classes', type=int, default=4, metavar='N', help='num classes')
205     parser.add_argument('--dataset_dir', type=str, default="D:/NIH-yolo-efficientnet/KAGGLE-IMAGE/archive/TEST_EFFICIENT",
206                         metavar='S',
207                         help='model path')
208     opt = parser.parse_args()
209
210     device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
211
212     model = Classifier(opt)
213     model.load_state_dict(torch.load(opt.model_path))
214     model.to(device)
215     model.eval()
216
217     source_path = opt.dataset_dir
218     #print(source_path)
219     source_path_list = os.listdir(source_path)
220     #print(source_path_list)
221     data_folder_image = []
222     image_list = []
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - EfficientNet: 이미지 하나씩 yolo-crop-resize-efficientnet 진행

```
222     image_list = []
223     for i in range_(len(source_path_list)):
224         data_image = source_path + "/" + source_path_list[i] #폴더
225         print(data_image)
226         #if data_image.endswith("crop.png"):
227         #    image_list.append(data_image)
228         image_list.append(data_image)
229
230     #print(image_list)
231     dur_sum=0
232     csv_list = []
233     for idx, img_path in enumerate(image_list):
234         transform = A.Compose([
235             A.Resize(height=380, width=380),
236             A.Normalize(
237                 mean=[0.485, 0.456, 0.406],
238                 std=[0.229, 0.224, 0.225]),
239             ToTensorV2()
240         ])
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - EfficientNet: 이미지 하나씩 yolo-crop-resize-efficientnet 진행

```
241         image0 = cv2.imread(img_path)
242         img_path0 = os.path.basename(img_path)
243         start_time = time.time()
244         image = cv2.cvtColor(image0, cv2.COLOR_BGR2RGB)
245         image = transform(image=image)['image']
246         image = image.unsqueeze(0)
247         input = image.to(device)
248         with torch.no_grad():
249             pred = model(input)
250             prob = F.softmax(pred, dim=1)
251             prob = prob.cpu().data.numpy()[0]
252             end_time = time.time()
253             dur = end_time - start_time
254             class_id = prob.argmax()
255             fps = 1 / dur
256             if idx==0: # 최초 frame은 느리므로 평균에서 제외
257                 avg_fps = 0
258             else:
259                 dur_sum += dur
260                 avg_fps = idx_/dur_sum
```

Yolo_EfficientNet_Inference : main code review

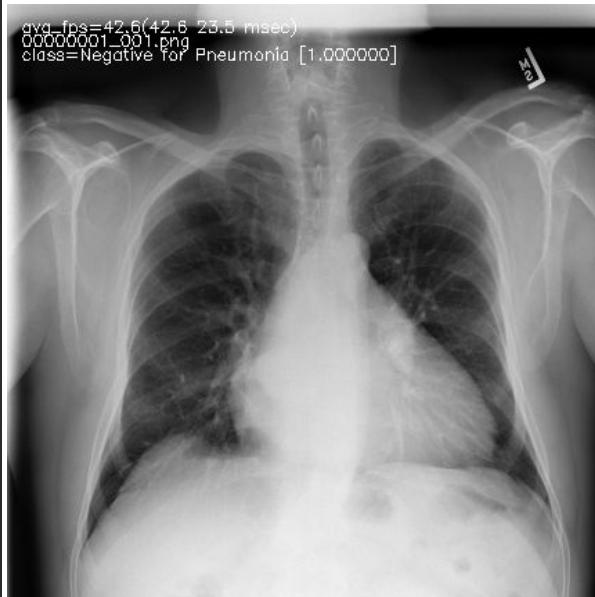
Yolo_Efficient_inference_image.py - EfficientNet: 이미지 하나씩 yolo-crop-resize-efficientnet 진행

```
280
281     msg_img = "%s" % (
282         img_path0)
283     msg_fps = "avg_fps=%f(%f %f msec)" % (
284         avg_fps, fps, dur * 1000)
285     msg_class = "class=%s [%f]" % (
286         class_name, max_prob)
287
288     image1 = cv2.resize(image0, dsize=(0, 0), fx=0.42, fy=0.42)
289     image1_copy = image1.copy()
290     cv2.putText(image1_copy, msg_fps, (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1)
291     cv2.putText(image1_copy, msg_img, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1)
292     cv2.putText(image1_copy, msg_class, (10, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1)
293
294     effi_img_path = img_path[:-3]
295     cv2.imwrite(effi_img_path + "-efficientnet.png", image1_copy)
296     cv2.imshow("Efficientnet", image1_copy)
297
298     key=cv2.waitKey(1)
299     if key==27:
300         break
301     elif key==32:
302         cv2.waitKey(0)
303
304
305     col_name = ["img_name", "prediction", "Negative for Pneumonia", "Typical Appearance",
306                 "Indeterminate Appearance", "Atypical Appearance"]
307     csv_df = pd.DataFrame(csv_list, columns=col_name)
308     csv_df.to_csv("./efficientnet_classification.csv", index=None) # 수정; classify 결과 csv파일 이름
309
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - EfficientNet: 이미지 하나씩 yolo-crop-resize-efficientnet 진행

```
avg_fps=43.2(39.4 25.4 msec) 00000057_001.png class=0 [1.000000 0.000000 0.000000 0.000000]
avg_fps=43.2(42.6 23.5 msec) 00000057_002.png class=0 [0.997245 0.000000 0.000000 0.002755]
avg_fps=43.2(44.5 22.5 msec) 00000057_003.png class=0 [1.000000 0.000000 0.000000 0.000000]
avg_fps=43.3(46.5 21.5 msec) 00000057_004.png class=0 [1.000000 0.000000 0.000000 0.000000]
avg_fps=43.3(44.5 22.5 msec) 00000058_000.png class=0 [0.951103 0.000000 0.000001 0.048896]
avg_fps=43.3(44.5 22.5 msec) 00000059_000.png class=0 [1.000000 0.000000 0.000000 0.000000]
avg_fps=43.3(42.6 23.5 msec) 00000059_001.png class=0 [1.000000 0.000000 0.000000 0.000000]
avg_fps=43.3(44.5 22.5 msec) 00000060_000.png class=0 [1.000000 0.000000 0.000000 0.000000]
avg_fps=43.3(40.9 24.4 msec) 00000061_000.png class=0 [1.000000 0.000000 0.000000 0.000000]
avg_fps=43.3(46.5 21.5 msec) 00000061_001.png class=0 [0.999655 0.000000 0.000001 0.000344]
avg_fps=43.3(44.5 22.5 msec) 00000061_002.png class=0 [0.922137 0.000017 0.000024 0.077822]
avg_fps=43.3(44.5 22.5 msec) 00000061_003.png class=0 [0.687298 0.000065 0.000008 0.312629]
avg_fps=43.3(40.9 24.4 msec) 00000061_004.png class=0 [0.997711 0.000003 0.000000 0.002286]
avg_fps=43.2(28.4 35.2 msec) 00000061_005.png class=0 [0.999858 0.000000 0.000000 0.000141]
avg_fps=43.2(44.5 22.5 msec) 00000061_006.png class=0 [0.998894 0.000000 0.000000 0.001106]
avg_fps=43.2(39.4 25.4 msec) 00000061_007.png class=0 [1.000000 0.000000 0.000000 0.000000]
avg_fps=43.2(44.5 22.5 msec) 00000061_008.png class=0 [1.000000 0.000000 0.000000 0.000000]
```



Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - EfficientNet: 이미지 하나씩 yolo-crop-resize-efficientnet 진행

	A	B	C	D	E	F
1	img_name	prediction	Negative for Pneumonia	Typical Appearance	Indeterminate Appearance	Atypical Appearance
2	00000001_000.png	Negative for Pneumonia		1	2.34E-27	2.32E-25
3	00000001_001.png	Negative for Pneumonia		1	2.21E-26	5.48E-30
4	00000001_002.png	Negative for Pneumonia		1	1.13E-28	7.88E-28
5	00000002_000.png	Negative for Pneumonia		1	6.32E-23	7.05E-23
6	00000003_000.png	Negative for Pneumonia		1	5.04E-19	4.69E-19
7	00000003_001.png	Negative for Pneumonia		1	5.41E-27	5.09E-29
8	00000003_002.png	Negative for Pneumonia		1	1.99E-23	9.68E-23
9	00000003_003.png	Negative for Pneumonia		1	1.36E-27	6.68E-27
10	00000003_004.png	Negative for Pneumonia		1	2.19E-25	8.31E-29
11	00000003_005.png	Negative for Pneumonia		1	5.77E-21	4.32E-22
12	00000003_006.png	Negative for Pneumonia		1	4.37E-20	1.94E-24
13	00000003_007.png	Negative for Pneumonia		1	4.83E-19	2.45E-21
14	00000004_000.png	Negative for Pneumonia		1	1.26E-29	6.66E-31
15	00000005_000.png	Negative for Pneumonia		1	2.19E-26	3.85E-28
16	00000005_001.png	Negative for Pneumonia		1	6.60E-44	0
17	00000005_002.png	Negative for Pneumonia		1	0	0
18	00000005_003.png	Negative for Pneumonia		1	2.34E-36	1.89E-41
19	00000005_004.png	Negative for Pneumonia		1	2.49E-26	9.08E-35
20	00000005_005.png	Negative for Pneumonia		1	1.18E-22	1.92E-21
21	00000005_006.png	Negative for Pneumonia		1	3.06E-18	4.28E-20
22	00000005_007.png	Negative for Pneumonia		1	1.73E-20	1.21E-17
23	00000006_000.png	Negative for Pneumonia		1	9.65E-30	7.22E-30
24	00000007_000.png	Negative for Pneumonia		1	3.15E-34	2.83E-38
25	00000008_000.png	Negative for Pneumonia	0.9999641	1.28E-10	4.74E-11	3.58E-05
26	00000008_001.png	Negative for Pneumonia	0.99999964	6.01E-15	4.63E-15	3.36E-07
27	00000008_002.png	Negative for Pneumonia		1	4.42E-18	8.57E-18
28	00000009_000.png	Negative for Pneumonia	0.9999999	1.15E-14	1.33E-13	1.13E-07
29	00000010_000.png	Negative for Pneumonia		1	5.24E-15	5.73E-13
30	00000011_000.png	Negative for Pneumonia		1	2.13E-14	1.42E-14
31	00000011_001.png	Negative for Pneumonia		1	2.97E-18	1.63E-16
32	00000011_002.png	Negative for Pneumonia		1	6.95E-14	2.10E-13

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo_EfficientNet:

```
298 def Yolo_Efficient():
299     parser = argparse.ArgumentParser()
300     parser.add_argument('--weights', nargs='+', type=str, default='yolov5-V5.0_best_weight_batch16_epoch20.pt', help='model.pt path(s)')
301     parser.add_argument('--source', type=str, default='D:/NIH-yolo-efficientnet/KAGGLE-IMAGE/archive/TEST_FULL', help='source') # file/folder, 0 for webcam
302     parser.add_argument('--img-size', type=int, default=640, help='inference size (pixels)')
303     parser.add_argument('--conf-thres', type=float, default=0.25, help='object confidence threshold')
304     parser.add_argument('--iou-thres', type=float, default=0.45, help='IOU threshold for NMS')
305     parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
306     parser.add_argument('--view-img', action='store_true', help='display results')
307     parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
308     parser.add_argument('--save-conf', action='store_true', help='save confidences in --save-txt labels')
309     parser.add_argument('--nosave', action='store_true', help='do not save images/videos')
310     parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --class 0, or --class 0 2 3')
311     parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic NMS')
312     parser.add_argument('--augment', action='store_true', help='augmented inference')
313     parser.add_argument('--update', action='store_true', help='update all models')
314     parser.add_argument('--project', default='D:/NIH-yolo-efficientnet/KAGGLE-IMAGE/archive/TEST_FULL', help='save results to project/name')
315     parser.add_argument('--name', default='exp', help='save results to project/name')
316
317     parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')
318     parser.add_argument('--model_name', type=str, default="efficientnet_b4", metavar='S', help='model name')
319     parser.add_argument('--model_path', type=str, default="efficientnet_b4.pt", metavar='S', help='model path')
320     parser.add_argument('--num_classes', type=int, default=4, metavar='N', help='num classes')
321     #parser.add_argument('--dataset_dir', type=str, default="KAGGLE-IMAGE/archive/TEST_FULL", metavar='S', help='model path')
322     opt = parser.parse_args()
323
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo_EfficientNet:

```
324     # Yolo
325     source, weights, view_img, save_txt, imgsz = opt.source, opt.weights, opt.view_img, opt.save_txt, opt.img_size
326     save_img = not opt.nosave and not source.endswith('.txt') # save inference images
327     webcam = source.isnumeric() or source.endswith('.txt') or source.lower().startswith(
328         ('rtsp://', 'rtmp://', 'http://', 'https://'))
329
330     # Directories
331     # save_dir = Path(increment_path(Path(opt.project) / opt.name, exist_ok=opt.exist_ok)) # increment run
332     # (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True) # make dir
333     save_dir = opt.project
334
335     # Initialize
336     set_logging()
337     device = select_device(opt.device)
338     half = device.type != 'cpu' # half precision only supported on CUDA
339
340     # Load model
341     model = attempt_load(weights, map_location=device) # load FP32 model
342     stride = int(model.stride.max()) # model stride
343     imgsz = check_img_size(imgsz, s=stride) # check img_size
344     if half:
345         model.half() # to FP16
346
347     # Second-stage classifier
348     classify = False
349     if classify:
350         modelc = load_classifier(name='resnet101', n=2) # initialize
351         modelc.load_state_dict(torch.load('weights/resnet101.pt', map_location=device)['model']).to(device).eval()
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo_EfficientNet:

```
353     # Set Dataloader
354     vid_path, vid_writer = None, None
355     if webcam:
356         view_img = check_imshow()
357         cudnn.benchmark = True # set True to speed up constant image size inference
358         dataset = LoadStreams(source, img_size=imgsz, stride=stride)
359     else:
360         dataset = LoadImages(source, img_size=imgsz, stride=stride)
361
362     # Get names and colors
363     names = model.module.names if hasattr(model, 'module') else model.names
364     colors = [[random.randint(0, 255) for _ in range(3)] for _ in names]
365
366     # Run inference
367     if device.type != 'cpu':
368         model(torch.zeros(1, 3, imgsz, imgsz).to(device).type_as(next(model.parameters()))). # run once
369         t0 = time.time()
370         csv_list = []
371
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo_EfficientNet:

```
372     for path, img, im0s, vid_cap in dataset:
373         ori_img = cv2.imread((path))
374         cv2.imshow("ORIGIN IMAGE", ori_img)
375         # cv2.waitKey(0) # 1 millisecond
376
377         print("\n\n=====Yolo=====")
378         # print("im0s",im0s)
379         # print("img", img)
380
381         name = path.split(".")[0]
382         # print(name)
383         img = torch.from_numpy(img).to(device)
384         img = img.half() if half else img.float() # uint8 to fp16/32
385         img /= 255.0 # 0 - 255 to 0.0 - 1.0
386         if img.ndim == 3:
387             img = img.unsqueeze(0)
388
389         # Inference
390         t1 = time_synchronized()
391         # print("img",img)
392         pred = model(img, augment=opt.augment)[0]
393
394         # Apply NMS
395         pred = non_max_suppression(pred, opt.conf_thres, opt.iou_thres, classes=opt.classes, agnostic=opt.agnostic_nms)
396         t2 = time_synchronized()
397
398         # Apply Classifier
399         if classify:
400             pred = apply_classifier(pred, modelc, img, im0s)
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo_EfficientNet:

```
402     # Process detections
403     for i, det in enumerate(pred): # detections per image
404         if webcam: # batch_size >= 1
405             p, s, im0, frame = path[i], '%g: ' % i, im0s[i].copy(), dataset.count
406         else:
407             p, s, im0, frame = path, '', im0s, getattr(dataset, 'frame', 0)
408         p = Path(p) # to Path
409         # print(p.name)
410         # (save_dir / p.name).mkdir(parents=True, exist_ok=True)
411         # save_path = str(save_dir / p.name / p.name) # img.jpg
412         save_path = save_dir
413         # txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image' else f'_{{frame}}') # img.txt
414         txt_path = str(save_dir) + ('' if dataset.mode == 'image' else f'_{{frame}}') # img.txt
415         # s += '%gx%g ' % img.shape[2:] # print string
416         gn = torch.tensor(im0.shape)[[1, 0, 1, 0]] # normalization gain whwh
417
418         if len(det):
419             # Rescale boxes from img_size to im0 size
420             det[:, :4] = scale_coords(img.shape[2:], det[:, :4], im0.shape).round()
421             # Print results
422             for c in det[:, -1].unique():
423                 n = (det[:, -1] == c).sum() # detections per class
424                 s += f'{n} {names[int(c)]}{'' * (n > 1)}' # add to string
425             # Write results
426             im0_copy_box = im0.copy()
427             im0_copy_text = im0.copy()
428             for *xyxy, conf, cls in reversed(det):
429                 if save_txt: # Write to file
430                     xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist() # normalized xywh
431                     line = (cls, *xywh, conf) if opt.save_conf else (cls, *xywh) # label format
432                     with open(txt_path + '.txt', 'a') as f:
433                         f.write((('%g ' * len(line)).rstrip() % line + '\n')
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo_EfficientNet:

```
435     if save_img or view_img: # Add bbox to image
436         label = f'{names[int(cls)]} {conf:.2f}'
437
438         img0_box = plot_one_box_only(xyxy, im0_copy_box, color=colors[int(cls)], line_thickness=3)
439         cv2.imwrite(name + "-efficientnet.png", img0_box)
440         #cv2.imshow("plot_one_box_only", img0_box)
441
442         im0_copy_text = plot_one_box(xyxy, im0_copy_text, label=label, color=colors[int(cls)], line_thickness=3)
443         cv2.imwrite(name + "-yolo.png", im0_copy_text)
444         #cv2.imshow("plot_one_box", im0_text)
445
446         '''key = cv2.waitKey(0) & 0xFF
447         if key == ord('q') or key == 27: # 'q' 이거나 'esc' 이면 종료
448             cv2.destroyAllWindows()'''
449
450         if len(det)==0:
451             cv2.imwrite(name + "-yolo.png", im0)
452             #cv2.imshow("yolo", im0)
```

바운딩박스 + 예상 레이블
바운딩박스

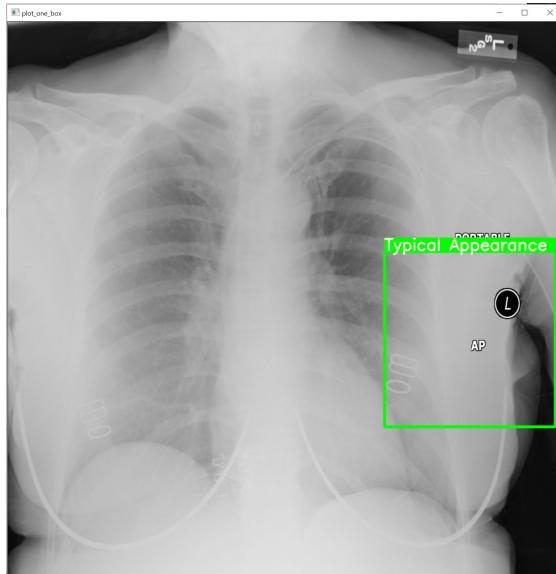
*결과물:

000.png -> 000-yolo.png로 저장

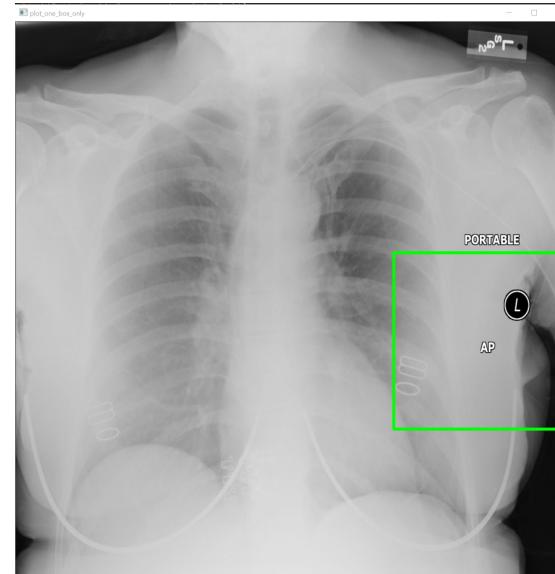
000.png -> 000-efficientnet.png로 저장

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo_EfficientNet:



바운딩박스 + 예상 레이블



바운딩박스

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo_EfficientNet:

```
453
454     # Print time (inference + NMS)
455     print(f'{s} Done. ({t2 - t1:.3f}s)')
456     # Stream results
457     if view_img:
458         cv2.imshow(str(p), im0)
459         cv2.waitKey(0) # 1 millisecond
460     # Save results (image with detections)
461     # cv2.imwrite(save_path, img) #월분
462     if save_img:
463         if dataset.mode == 'image':
464             yolo_img_show = cv2.imread(name + "-yolo.png")
465             cv2.imshow("yolo", yolo_img_show)
466
467         elif dataset.mode == 'video' or 'stream': # 'video' or 'stream'
468             if vid_path != save_path: # new video
469                 vid_path = save_path
470                 if isinstance(vid_writer, cv2.VideoWriter):
471                     vid_writer.release() # release previous video writer
472                 if vid_cap: # video
473                     fps = vid_cap.get(cv2.CAP_PROP_FPS)
474                     w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
475                     h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
476                 else: # stream
477                     fps, w, h = 30, im0.shape[1], im0.shape[0]
478                     save_path += '.mp4'
479                 vid_writer = cv2.VideoWriter(save_path, cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
480                 vid_writer.write(im0)
481
482
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo_EfficientNet:

```
483     # bounding box crop and resize
484     print("=====bounding box crop and resize=====")
485     #print(name +"-yolo.png")
486     crop_merge_one(name + "-yolo.png")
487
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo_EfficientNet:

```
488     # EfficientNet
489     print("=====EfficientNet=====\\n")
490     device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
491     model_eff = Classifier(opt)
492     model_eff.load_state_dict(torch.load(opt.model_path))
493     model_eff.to(device)
494     model_eff.eval()
495
496     image_list = []
497     data_image = name + "-crop.png" # 폴더
498
499     try:
500         #crop 실패한 이미지 기록
501         if os.path.isfile(data_image) == False:
502             value = (data_image,"","")
503             csv_list.append(value)
```

혹시 crop 이미지가 저장되지 않았다면
efficientnet의 결과물 csv에 표시
(이미지의 절대경로 기록)

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo_EfficientNet:

```
505     elif os.path.isfile(data_image):
506         image_list.append(data_image)
507
508         dur_sum = 0
509         for idx, img_path in enumerate(image_list):
510             transform = A.Compose([
511                 A.Resize(height=380, width=380),
512                 A.Normalize(
513                     mean=[0.485, 0.456, 0.406],
514                     std=[0.229, 0.224, 0.225]),
515                 ToTensorV2()
516             ])
517             image0 = cv2.imread(img_path)
518             img_path0 = os.path.basename(img_path)
519             start_time = time.time()
520             image = cv2.cvtColor(image0, cv2.COLOR_BGR2RGB)
521             image = transform(image=image)['image']
522             image = image.unsqueeze(0)
523             input = image.to(device)
524             with torch.no_grad():
525                 pred = model_eff(input)
526                 prob = F.softmax(pred, dim=1)
527                 prob = prob.cpu().data.numpy()[0]
528
529                 end_time = time.time()
530                 dur = end_time - start_time
531                 class_id = prob.argmax()
532                 max_prob = prob[int(class_id)]
533
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo_EfficientNet:

```
534         fps = 1 / dur
535     if idx == 0: # 최초 frame을 느리므로 평균에서 제외
536         avg_fps = 0
537     else:
538         dur_sum += dur
539         avg_fps = idx / dur_sum
540
541     # print("class_id", class_id)
542     # print("class_id type", type(class_id))
543     class_name = ""
544
545     if str(class_id) == "0":
546         class_name = "Negative for Pneumonia"
547     elif str(class_id) == "1":
548         class_name = "Typical Appearance"
549     elif str(class_id) == "2":
550         class_name = "Indeterminate Appearance"
551     elif str(class_id) == "3":
552         class_name = "Atypical Appearance"
553
554     value = (img_path0, class_name, prob[0], prob[1], prob[2], prob[3])
555     csv_list.append(value)
556
557     msg_img = "%s" % [
558         img_path0]
559     msg_fps = "avg_fps=%1f(%1f.%1f msec)" % (
560         avg_fps, fps, dur * 1000)
561     msg_class = "class=%s [%%.6f]" % (
562         class_name, max_prob)
```

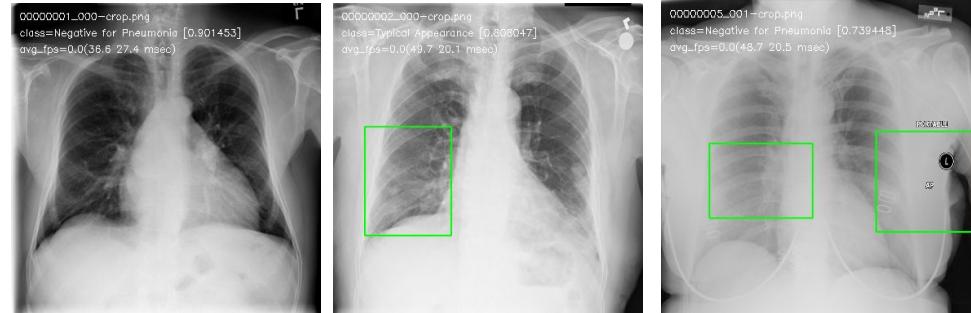
A	B	C	D	E	F
img_name	prediction	Negative for Pneumonia	Typical Appearance	Indeterminate Appearance	Atypical Appearance
2 00000001_000-crop.png	Negative for Pneumonia	0.90145284	0.033508554	0.048705816	0.016332895
3 00000001_001-crop.png	Negative for Pneumonia	0.9341554	0.014215147	0.019472033	0.03215734
4 00000001_002-crop.png	Typical Appearance	0.018052375	0.8054181	0.15171266	0.024816832
5 00000002_000-crop.png	Typical Appearance	0.09750793	0.808047	0.07355589	0.02088925
6 00000003_000-crop.png	Negative for Pneumonia	0.9130746	0.01371222	0.052219454	0.020993756
7 00000003_001-crop.png	Negative for Pneumonia	0.9482369	0.01862332	0.022300016	0.010839734
8 00000003_002-crop.png	Negative for Pneumonia	0.9800847	0.005151562	0.009058088	0.005705531
9 00000003_003-crop.png	Negative for Pneumonia	0.81564146	0.036714032	0.13310932	0.014535208
10 00000003_004-crop.png	Negative for Pneumonia	0.48281038	0.4494023	0.04791225	0.019875022
11 00000003_005-crop.png	Negative for Pneumonia	0.8498921	0.025677934	0.09607066	0.02835936
12 00000003_006-crop.png	Negative for Pneumonia	0.807123	0.06913641	0.11888384	0.004856698

index(0,1,2,3)로 기록되는데,
사용자 편의를 위해 레이블로 기록

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo_EfficientNet:

```
562     #yolo_img = name + "-yolo-only.png"
563     yolo_img = name + "-efficientnet.png"
564     yolo_img_split=yolo_img.split("\\")[-1]
565     ori_img = name + ".png"
566     ori_img_split = ori_img.split("\\")[-1]
567     images = os.listdir(source)
568     #print(images)
569     #print(yolo_img_split)
570     #print(ori_img_split)
571
572     if yolo_img_split in images:
573         yolo_img = cv2.imread(yolo_img)
574         # cv2.imshow("yolo-only", yolo_img)
575         yolo_img = cv2.resize(yolo_img, dsizes=(380, 380), fx=0.42, fy=0.42)
576         cv2.putText(yolo_img, msg_img, (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1)
577         cv2.putText(yolo_img, msg_class, (10, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1)
578         cv2.putText(yolo_img, msg_fps, (10, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1)
579         cv2.imwrite(name + "-efficientnet.png", yolo_img)
580         cv2.imshow("Efficientnet", yolo_img)
581         # cv2.waitKey(0)
582     elif ori_img_split in images:
583         ori_img = cv2.imread(ori_img)
584         # cv2.imshow("ori_img", ori_img)
585         ori_img = cv2.resize(ori_img, dsizes=(380, 380), fx=0.42, fy=0.42)
586         cv2.putText(ori_img, msg_img, (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1)
587         cv2.putText(ori_img, msg_class, (10, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1)
588         cv2.putText(ori_img, msg_fps, (10, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1)
589         cv2.imwrite(name + "-efficientnet.png", ori_img)
590         cv2.imshow("Efficientnet", ori_img)
591         # cv2.waitKey(0)
592
593     key = cv2.waitKey(1)
594     if key == 27:
595         break
596     elif key == 32:
597         cv2.waitKey(0)
```



Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - Yolo_EfficientNet:

```
598
599     col_name = ["img_name", "prediction", "Negative for Pneumonia", "Typical Appearance",
600                 "Indeterminate Appearance", "Atypical Appearance"]
601     csv_df = pd.DataFrame(csv_list, columns=col_name)
602     csv_df.to_csv("./yolo_efficientnet_classification.csv", index=None) # 수정; classify 결과 csv파일 이름
603     cv2.waitKey(1000)
604     cv2.destroyAllWindows()
605
606     except:
607         pass
608
609     if save_txt or save_img:
610         s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}" if save_txt else ''
611         print(f"Results saved to {save_dir}{s}")
612         print(f'Done. ({time.time() - t0:.3f}s)')
613         print()
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - __main__:

```
618 ► if __name__ == '__main__':
619     #check_requirements(exclude=('pycocotools', 'thop'))
620     #with torch.no_grad():
621     #    Yolo()
622
623     #EfficientNet()
624
625     Yolo_Efficient()
```

Yolo_EfficientNet_Inference : main code review

Yolo_Efficient_inference_image.py - __main__:



00000005_000



00000005_000-crop



00000005_000-efficientnet



00000005_000-yolo

A	B	C	D	E	F
img_name	prediction	Negative for Pneumonia	Typical Appearance	Indeterminate Appearance	Atypical Appearance
1 img_name	prediction	Negative for Pneumonia	Typical Appearance	Indeterminate Appearance	Atypical Appearance
2 00000001_000-crop.png	Negative for Pneumonia	0.90145284	0.033508554	0.048705816	0.016332895
3 00000001_001-crop.png	Negative for Pneumonia	0.9341554	0.014215147	0.019472033	0.03215734
4 00000001_002-crop.png	Typical Appearance	0.018052375	0.8054181	0.15171266	0.024816832
5 00000002_000-crop.png	Typical Appearance	0.09750793	0.808047	0.07355589	0.02088925
6 00000003_000-crop.png	Negative for Pneumonia	0.9130746	0.01371222	0.052219454	0.020993756
7 00000003_001-crop.png	Negative for Pneumonia	0.9482369	0.01862332	0.022300016	0.010839734
8 00000003_002-crop.png	Negative for Pneumonia	0.9800847	0.005151562	0.009058088	0.005705531
9 00000003_003-crop.png	Negative for Pneumonia	0.81564146	0.036714032	0.13310932	0.014535208
10 00000003_004-crop.png	Negative for Pneumonia	0.48281038	0.4494023	0.04791225	0.019875022
11 00000003_005-crop.png	Negative for Pneumonia	0.8498921	0.025677934	0.09607066	0.02835936
12 00000003_006-crop.png	Negative for Pneumonia	0.807123	0.06913641	0.11888384	0.004856698
13 00000003_007-crop.png	Negative for Pneumonia	0.9735791	0.003419584	0.008040861	0.014960443
14 00000004_000-crop.png	Negative for Pneumonia	0.8542615	0.005879921	0.1350534	0.004805235



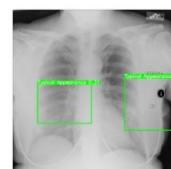
00000005_001



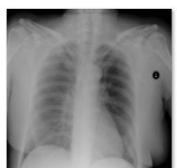
00000005_001-crop



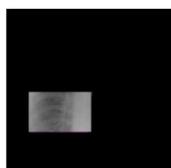
00000005_001-efficientnet



00000005_001-yolo



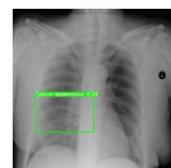
00000005_002



00000005_002-crop



00000005_002-efficientnet



00000005_002-yolo