지능화 캡스톤 프로젝트-10

[프로젝트 #2]
YOLO를 이용한 안전모 검출

2022. 05. 11

도규원, 김현용

충북대학교 산업인공지능학과

공지사항

- 출석관리 철저

  - 채팅창에 “20학번 OOO 출석”으로 1일 2회 기록 : 입장(19시)과 퇴장(21시~)

  - 20학번 : 프로젝트(60%) + 주제발표(10%) + **출석(30%)**

항목		비율	내용	비고
발표 평가 (총 100%)	1차 발표	30%	**Project #1. CNN을 이용한 불량검출**	조 편성
	2차 발표	40%	**Project #2. YOLO를 이용한 안전모 검출**	
주제발표 평가		10%	조별 발표	조별 발표 주제 선정
출 석		20%	총 수업시간의 ¾ 미달시 F. (총 수업시간 : 60시간, 15시간 이상 미출석시 F)	

→ 5일 이상 결석 시 F 처리

조편성 결과(4/27 현재)

20 학번 조편성

조	성명	발표여부
20-1	전일우, 이효중, 박성범	O
20-2	임동민, 신정환, 안건호	
20-3	최원희, 손의걸	O
20-4	윤재웅, 김성웅	O
20-5	강윤구, 김병근	O
20-6	박민우	O
20-7	원형일, 장민우	
20-8	고정재, 유용주	O

총 8조 / 17명

21 학번 조편성

조	성명	발표여부
21-1	이용규, 유대건	
21-2	김대훈	O
21-3	최준혁, 이지연	O
21-4	김상순, 정수현	O
21-5	방창현, 정원용	
21-6	우상진, 김준태	
21-7	봉은정, 김원우	
21-8	이충현, 이지호	
21-9	정준영, 윤범희	

총 9조 / 17명

조별 발표 주제 (4/27 현재)

주차	날짜	대면	발표 주제	발표조
7	4/20	O	Project #2 : YOLO를 이용한 안전모 검출	김현용, 도규원
8	4/27	X	검출과 분할 – Object Detection vs. Segmentation, Mask-RCNN	21-3, 21-4
9	5/4	X	AI특강1(19:00~20:30) : 산업지능화를 통한 비즈니스	초청강사
10	5/11	X	RCNN계열–**R-CNN, fast/faster R-CNN의 비교(구조, 성능)** 1-Stage detector – **YOLO 외 SSD, RetinaNet 등**	21-6, 21-9 20-7, 21-1
11	5/18	X	**YOLOv5 사용법** - 각종 파라미터 설정, 학습/검증/추론 방법 등 주석 - **레이블링 파일형식(xml, json, yolov5) 변환 코드** 평가지표- **mAP(예제, 코드)** AI특강2(21:00~22:00) : 심층강화학습 개론	20-2, 21-7 21-5 21-8 초청강사
12	5/25	O	중간점검(19~20시, 개신동 E9-105): 5분 발표 (20학번, 21학번) 장소 이동 후 간담회: 학과장님 및 교수님	모든 조
13	6/1	x	[휴무 대보강] 동영상 강의 시청	이광연, 김재영
14	6/8	x	AI특강3(19:00~20:00) : 미정 최종점검(20학번, 21학번) 테스트 데이터 공개 → 검출결과 제출 (→ mAP 피드백)	초청강사 모든 조
15	6/15	O	프로젝트 #2 발표평가	모든 조

오늘 수업진행

<table>
<tr><th>구분</th><th>시간</th><th>20학번</th><th>21학번</th></tr>
<tr><td rowspan="2">전체수업</td><td>19:00~19:10</td><td colspan="2">수업 안내 / 출석체크</td></tr>
<tr><td>19:10~19:50</td><td colspan="2">주제발표
21-6조, 21-9조 : RCNN 계열
20-7조, 21-1조 : YOLO 계열</td></tr>
<tr><td rowspan="2">분반수업</td><td rowspan="2">20:00~21:00</td><td rowspan="2">소회의실1 / 논문방
(이광연, 윤성철, 김재영)
- 포트폴리오 제출방법
- 논문 샘플 공유
- 논문발표 연습일정</td><td>소회의실2 / 프로젝트방
(김현용, 도규원)
- AI메이커톤 사례#2
- 각 조 진행점검</td></tr>
<tr><td>조별 토의 (30분)</td></tr>
<tr><td>전체수업</td><td>21:00~</td><td colspan="2">수업 마무리 / (전체방에서) 출석체크</td></tr>
</table>

질의응답

Q & A

LAB 423

제 2회 AI 메이커톤

+AI MAKEATHON

충북대학교 빅데이터전공 최우석
충북대학교 빅데이터전공 이주연
충북대학교 빅데이터전공 유연주

CONTENTS

1-1. 배경

인공지능 기술을 적용(+AI)하여 산업 현장의 문제를 해결

출처 : 안전보건공단 블로그

안전모를 착용한 사람과 착용하지 않은 사람을

AI 알고리즘을 적용하여 보다 정확히 판별,

나아가 산업 현장의 안전사고 예방에 기여

1-2. 문제 정의 및 방법론

문제 정의 : “영상 속에서 작업자의 안전모 착용상태에 대해 Object Detection을 수행하여 Class와 Bounding box 정보 도출”

YOLO-CNN 복합모델 적용

YOLO 결과 예시 — CNN 결과 예시 — Hard_hat_workers1513.txt

label	confidence	xmin	ymin	xmax	ymax
head	0.999952	159	217	147	210
helmet	1.000000	309	411	99	228
helmet	0.999985	380	415	120	216

2-1. 활용 데이터 소개

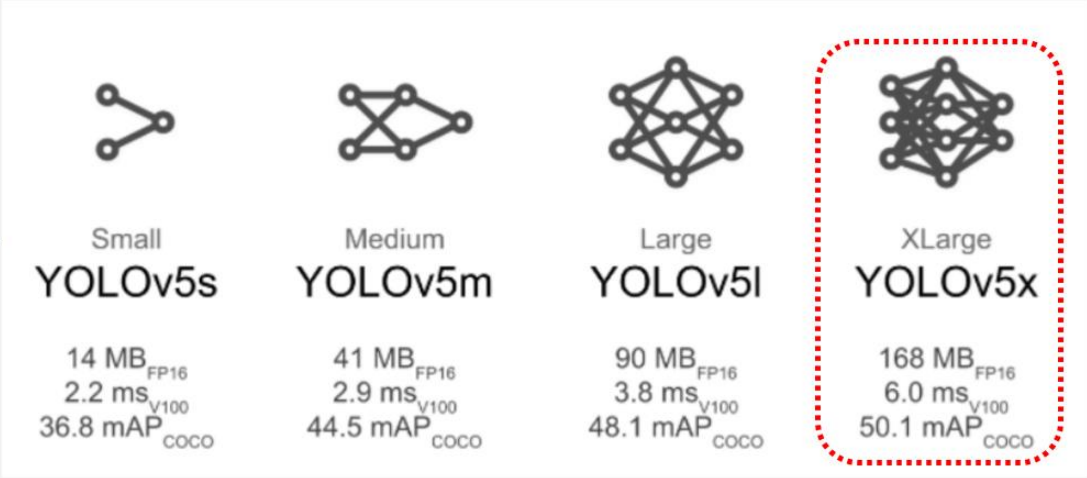
총 12,041개 이미지 활용

	데이터 형태	데이터 활용 방법
경진대회 제공 데이터	총 **5,000**개 / .png(416x416) / .xml	CNN 모델 학습 / YOLO-CNN 모델 평가
Roboflow	총 **7,041**개 / .jpg / .xml	CNN 모델 학습

* Roboflow 데이터 : https://public.roboflow.com/object-detection/hard-hat-workers

2-2. 알고리즘 소개 (1)

YOLOv5x



출처 : https://ropiens.tistory.com/44

YOLO란, 객체 탐지를 단일 회귀로 재구성하여 이미지에 대한 Bounding Box로 영역을 설정하고,
영역 내 물체의 종류나 유무를 판단하여 분류와 객체 탐지를 한꺼번에 수행하는 방법론이다.

s, m, l, x 순으로 모델의 깊이가 깊어지며 이에 따라 **정확도가 높아진다**.

본 경진대회에서는 정확도가 가장 높은 모델인 YOLOv5x를 사용하였다.

2-2. 알고리즘 소개 (2)

VGGNet

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

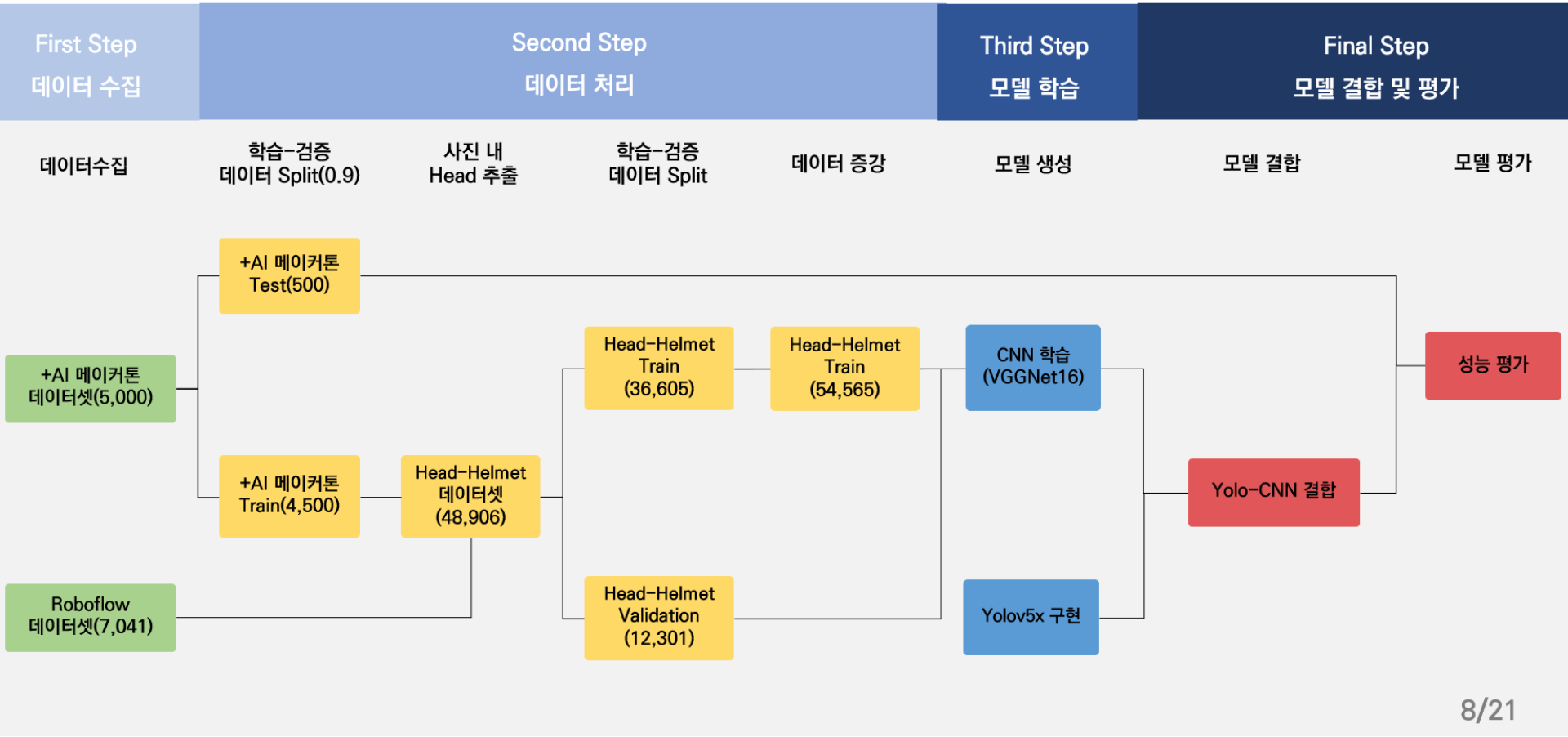
VGGNet의 종류

VGGNet은 2014년에 등장한 CNN 알고리즘으로,

간단한 CNN층으로 **빠르며 정확한 이미지 분류 성능**을 보인다.

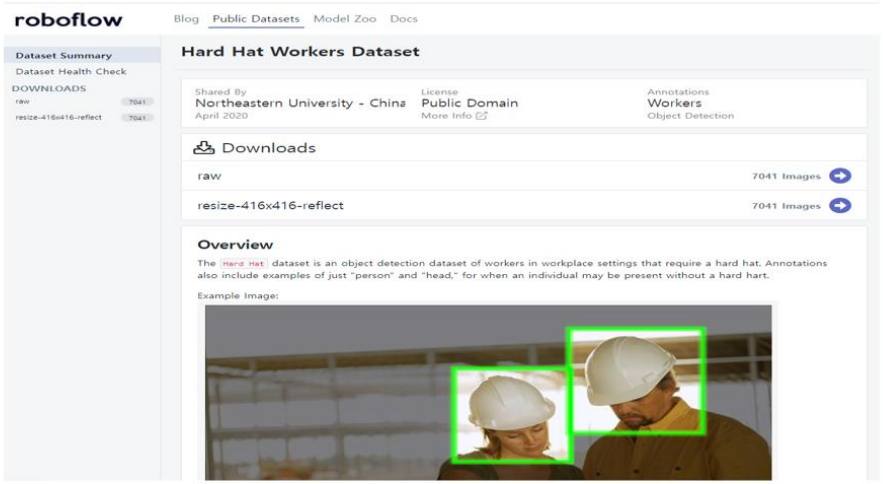
본 경진대회에서는 **16버전**을 사용하였다.

2-3. 분석 프로세스 로드맵



3-1. 데이터 분석 수행

First Step) 데이터 수집



- 충분한 양의 데이터로 모델을 학습 시키기 위해 +AI 메이커톤 데이터(5,000건)와 roboflow 데이터(총 7,041건)를 함께 수집/활용

데이터수집 ✓

학습-검증 데이터 Split(0.9)

사진 내 Head 추출

학습-검증 데이터 Split

데이터 증강

모델 생성

모델 결합

모델 평가

+AI 메이커톤 데이터셋(5,000)

+AI 메이커톤 Test(500)

+AI 메이커톤 Train(4,500)

Head-Helmet 데이터셋 (48,906)

Head-Helmet Train (36,605)

Head-Helmet Validation (12,301)

Head-Helmet Train (54,565)

CNN 학습 (VGGNet16)

Yolov5X 구현

Yolo-CNN 결합

성능 평가

Roboflow 데이터셋(7,041)

3-1. 데이터 분석 수행

Second Step) 학습-검증 데이터 Split (0.9)

```python
from sklearn.model_selection import train_test_split

file_names = ["hard_hat_workers"+str(i)+".png" for i in range(5000)]

train, test = train_test_split(file_names, test_size=0.1, shuffle=True) #랜덤분할
train = sorted(train, key = lambda x : int(x.split('workers')[-1].replace('.png', '')))
test = sorted(test, key = lambda x : int(x.split('workers')[-1].replace('.png', '')))

image_p = 'D:/메이커톤/images/'

for i in train:
    image_path = image_p + i
    image = Image.open(image_path)
    image.save('D:/메이커톤/images/train/' + i)

for i in test:
    image_path = image_p + i
    image = Image.open(image_path)
    image.save('D:/메이커톤/images/test/' + i)
```

- YOLO-CNN 결합 모델의 성능을 평가하기 위해 Sklearn의 train_test_split 함수를 활용하여 5,000개의 데이터 중 10%를 평가데이터로 분리

1.사람추출(AI메이커톤).py

2.사람추출(Hardhat).py

3-1. 데이터 분석 수행

Second Step) 사진 내 Head 추출

```
#각 사진에서 person box crop.
for person in range(1, len(persons_number)+1):
    obj = tree.find('./object['+str(person)+']')

    #helmet image crop & numberfing
    if obj.find('name').text == 'helmet':
        name = "hard_hat_workers"+str(person-1)
        file_name.append(image_path)
        p_number.append(person)
        label_box.append(obj.find('name').text)

        xmin, ymin, xmax, ymax = int(obj.find('bndbox').find('xmin').text), int(obj.
        bounding_box = [xmin, ymin, xmax, ymax]
        image = Image.open(image_path)
        image = image.crop(bounding_box) # left top right bottom
        image = image.resize((224,224))
        image.save('./test/helmet' + str(helmet)+'.png')

        helmet = helmet + 1
```

- +AI 메이커톤과 roboflow 데이터 모두 annotations을 **XML 형식**으로 제공
- Python의 xml.etree.ElementTree 패키지를 활용하여 정보 추출
- 사진마다 Object-bounding_box 정보를 활용하여 **사람의 머리 부분을 .png형식으로 crop**

(이미지가 Head 인지 Helmet인지 분류하는 모델을 학습하기 위함)

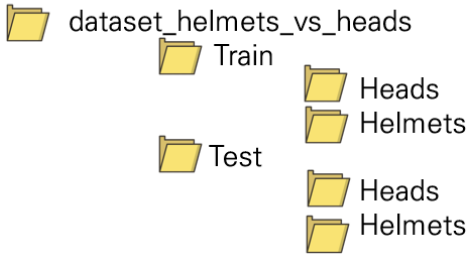
3.train-test-extract2.py

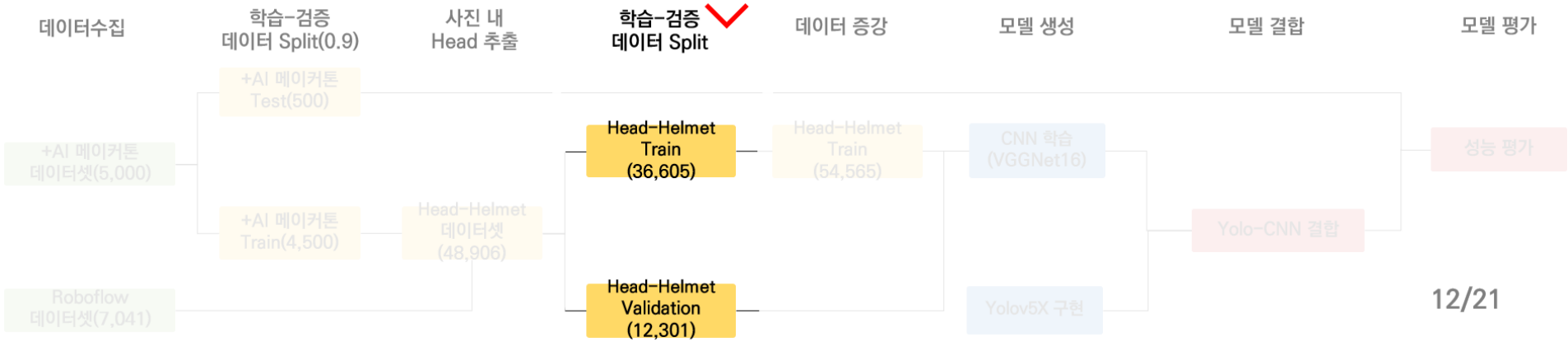
3-1. 데이터 분석 수행

Second Step) 학습-검증 데이터 Split

```python
# create directories
dataset_home = 'dataset_helmets_vs_heads/' #폴더명 지정
subdirs = ['train/', 'test/']

for subdir in subdirs:
    # create label subdirectories
    labeldirs = ['helmets/', 'heads/']
    for labldir in labeldirs:
        newdir = dataset_home + subdir + labldir
        makedirs(newdir, exist_ok=True)

# copy training dataset images into subdirectories
src_directory = 'train/'
for file in listdir(src_directory):
    print(file)
    src = src_directory + '/' + file
    dst_dir = 'train/'
    if random() < 0.25: # define ratio of pictures to use for validation
        dst_dir = 'test/'
    if file.startswith('helmet'):
        dst = dataset_home + dst_dir + 'helmets/'  + file
        copyfile(src, dst)
    elif file.startswith('head'):
        dst = dataset_home + dst_dir + 'heads/'  + file
        copyfile(src, dst)
```

- 추출한 'Head-Helmet 데이터셋'을 Train-Test로 분리함
- CNN 학습에 필요한 데이터 입출력 구조를 Image Data Generator 패키지에 적절하도록 아래와 같은 폴더 구성으로 분리





4. augmentation.py

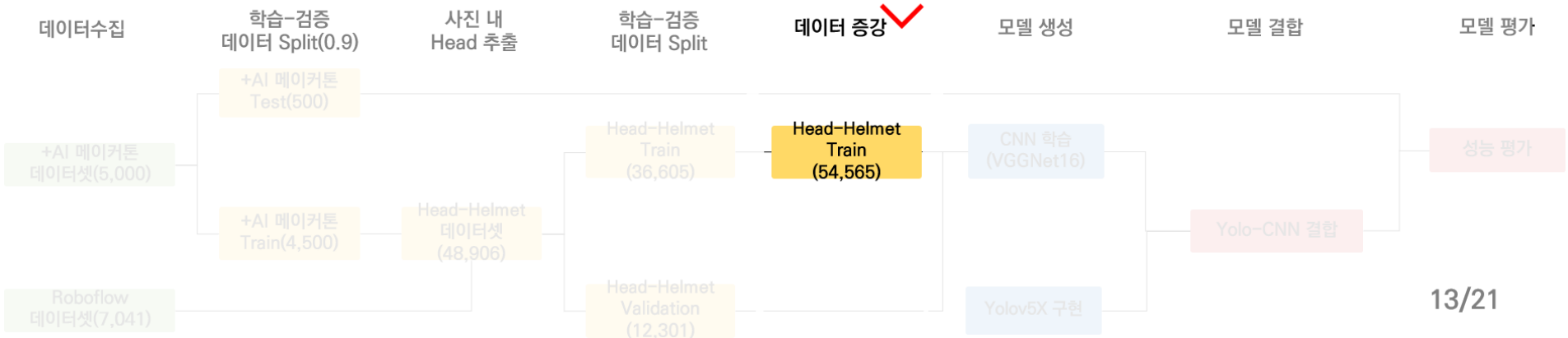
3-1. 데이터 분석 수행

Second Step) 데이터 증강

```python
aug = ImageDataGenerator(
    rotation_range=5,
    width_shift_range=0.12,
    height_shift_range=0.12,
    shear_range=0.1,
    # zoom_range=0.2,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode="nearest",
```

```python
image_directory = image_p + i + '\\' + str(j)  #save_path (output)
image_directories = sorted(glob(image_directory + '\\*'))
image_directories = image_directories[:-1]
image_directories = sorted(image_directories, key = lambda x : int((x.replace

for img in image_directories:
    print(img) #image_path
    image = load_img(img) #이미지로드
    image = np.expand_dims(image, axis=0)

    imageGen = aug.flow(
        image,
        batch_size=1,
        save_to_dir=image_directory, #저장장소
        # save_prefix=args["prefix"],
        save_format="jpg",
)

    total = 0

    for image in imageGen:
        total += 1
        if total == 3: #증강시키고 싶은 횟수
            break
```

- Head-Helmet 데이터셋을 확인해본 결과, Head(8,918개), Helmet(27,687개)로 **클래스 불균형** 확인
- 따라서 augmentation 방법을 통해 Head 이미지를 3배 증강시킴
- 증강 후, **Head 26,878개, Helmet 27,687개**로 데이터 확정



3-1. 데이터 분석 수행

Third Step) CNN 모델 생성 (1)

5. classification_learning.py
6. classification_inference.py

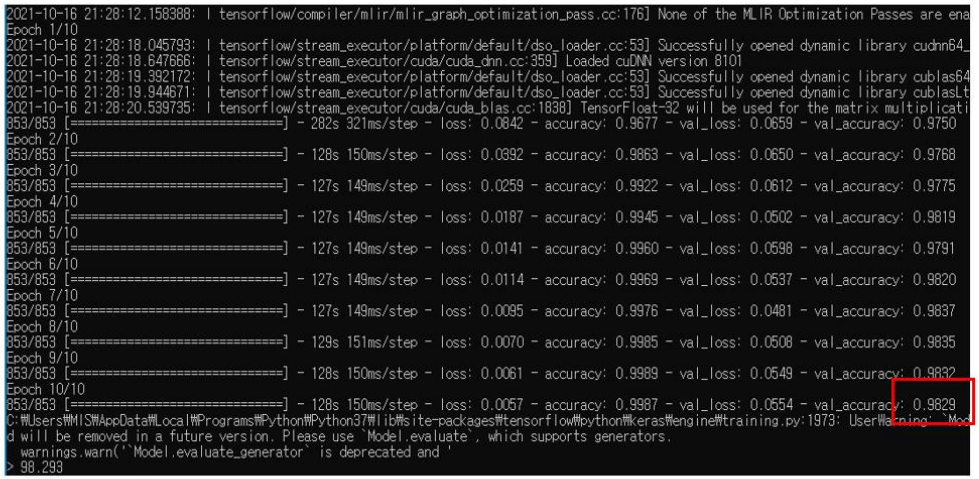
```python
def define_model():
    model = VGG16(include_top=False, input_shape=(224, 224, 3))
    # model = DenseNet121(include_top=False, input_shape=(224, 224, 3))
    # model = EfficientNetB0(include_top=False, input_shape=(224, 224, 3))
    # model = EfficientNetB4(include_top=False, input_shape=(224, 224, 3))

    # 전이학습을 하기 위해, CNN층의 학습 차단
    for layer in model.layers:
        layer.trainable = False

    # classification을 위한 FC층
    flat1 = Flatten()(model.layers[-1].output)
    class1 = Dense(128, activation='relu', kernel_initializer='he_uniform')(flat1)
    output = Dense(1, activation='sigmoid')(class1)

    # 모델 정의
    model = Model(inputs=model.inputs, outputs=output)
```

- Head-Helmet 이미지 분류를 위해 CNN 기반의 Model 생성
- 모델의 성능을 고도화하기 위해 **전이학습** 실시
- Dense층을 1로 설정하여, Confidence가 0과 가까우면 Head, 1과 가까우면 Helmet이라고 볼 수 있음
- 임계값은 0.5로 설정 (0.5보다 크면 Helmet, 작으면 Head Labeling)

5. classification_learning.py
6. classification_inference.py

3-1. 데이터 분석 수행

Third Step) CNN 모델 생성 (2)



- CNN 모델의 분류 성능은 98.29%로 확인됨

데이터수집 | 학습-검증 데이터 Split(0.9) | 사진 내 Head 추출 | 학습-검증 데이터 Split | 데이터 증강 | **모델 생성** | 모델 결합 | 모델 평가

+AI 메이커톤 Test(500)
+AI 메이커톤 데이터셋(5,000)
+AI 메이커톤 Train(4,500)
Roboflow 데이터셋(7,041)
Head-Helmet 데이터셋 (48,906)
Head-Helmet Train (36,605)
Head-Helmet Validation (12,301)
Head-Helmet Train (54,565)
CNN 학습 (VGGNet16)
Yolov5X 구현
Yolo-CNN 결합
성능 평가

run_yolov5x.py

3-1. 데이터 분석 수행

Third Step) YOLOv5x 모델 생성 (1)

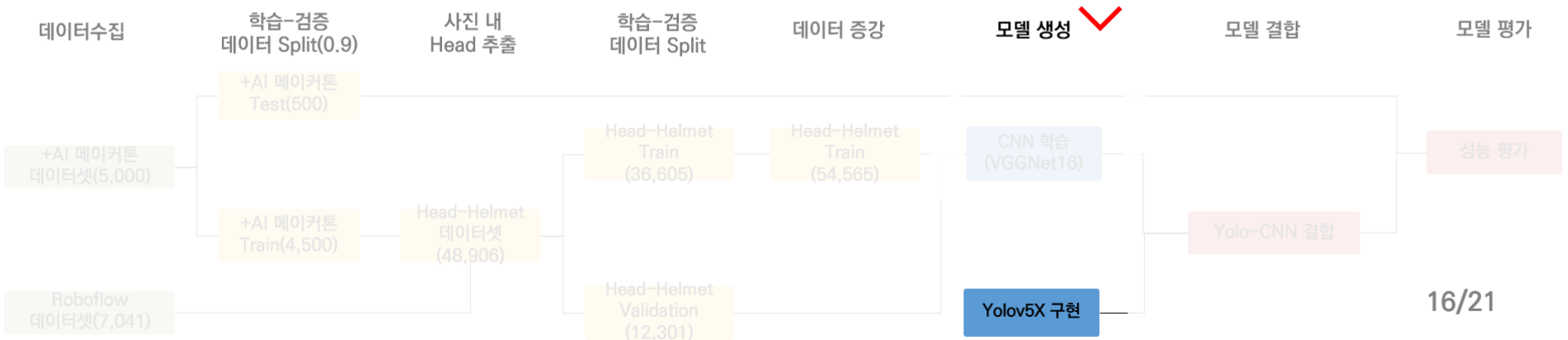
```
14  model_size = (640,640)
15  path_objname = r"obj.names"
16  path_weights = r"YOLOV5X/human_head.pt"
17  score = 0.45 #0.25
18  nms = 0.80
19  gpu = True
```

"사전학습모델 Human_head.pt
(가중치 파일)를 이용하였음"

```
37  if __name__ == "__main__":
38      for image in file_list:
39          ### yolo_v5 run and save box lotation
40          image_path = path_dir + "/" + image
41          INPUT = cv2.imread(image_path)
42          img = np.array(INPUT)
43          img, cnn_boxes = yolomodel.getObject(img, 0.25, 0.55, drawBox=True, char_type='Chinese')
44          cv2.imwrite('./yolo-result/'+image+'.png',img) #박스 시각화
45          # print(cnn_boxes)
46          for box in cnn_boxes:
47              image_paths.append(image_path)
48              image_boxes.append(box)
49
50  dataset["image_path"] = image_paths
51  dataset["image_box"] = image_boxes
52
53  dataset.to_excel("./yolo-result/yolo_result.xlsx", index=False)
```

Yolov5x 모델에 이미지를 넣으면
Output 위치 정보를 저장하도록 코드 구현

- YOLOv5x 코드 구현
- 평가방법 중 머리의 위치도 정확도 계산에 반영된다는 점을 고려하여, 직접 학습하는 대신 **사전학습 모델을 이용**하여 정확도를 높이고자 함
- 사전 학습 모델인 YOLO_person.pt를 이용
- **YOLO_person.pt**는 Head(사람머리)와 Person(사람전체) **2개의 Class를 탐지**할 수 있도록 학습됨



construction_YOLO.py

3-1. 데이터 분석 수행

Third Step) YOLOv5x 모델 생성 (2)

```
6   class opencvYOLO:
7       def __init__(self, mtype='darknet', imgsize=(416,416), objnames="coco.names", \
8                    weights="yolov3.weights", darknetcfg="yolov3.cfg", score=0.25, nms=0.6, gpu=False):
9           self.mtype = mtype
10          self.imgsize = imgsize
11          self.score = score
12          self.nms = nms
13
14          self.inpWidth = self.imgsize[0]
15          self.inpHeight = self.imgsize[1]
16          self.classes = None
17          with open(objnames, 'rt') as f:
18              self.classes = f.read().rstrip('\n').split('\n')
```

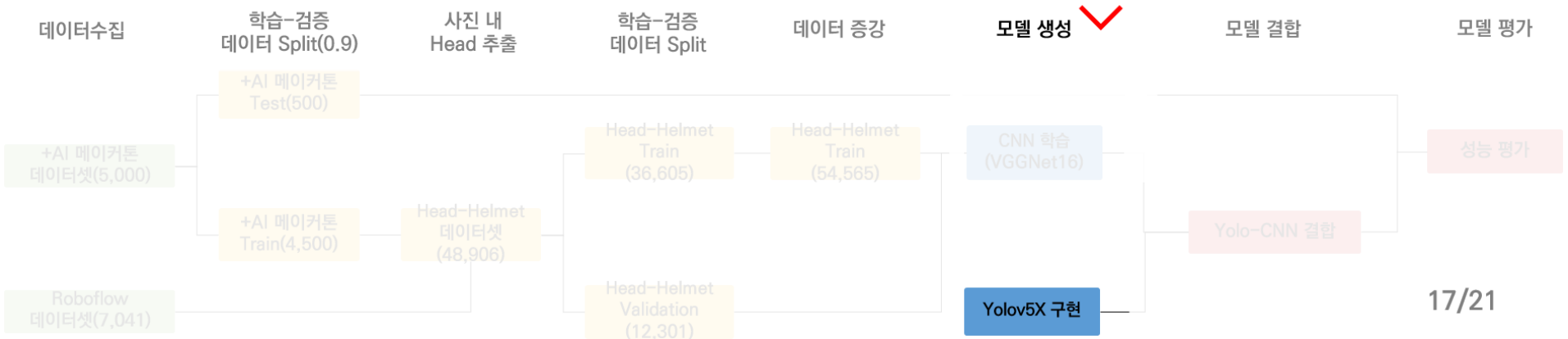
"Yolo Class 구현"

```
58              for p in predictions:
59                  if p['class'] == 0: #사람머리만 등장하게함
```

"Class=0(Head)만 출력하게 함"

```
79                          cnn_box.append((xmin, ymin, xmax, ymax))
```

- construction_YOLO.py에 Yolo Class 설계
- 사전에 학습된 Weight(.pt)를 이용하여 사진을 넣으면 사람의 얼굴(Head)만 탐지되도록 코드 수정
- Output 결과물과 박스 위치를 return 받음



3-1. 데이터 분석 수행

run_cnn.py

Final Step) 모델 결합

```
### 2. run_yolov5x.py 결과 불러오기(이미지 경로, 바운딩 박스 좌표)및 데이터 형식 변경 ###
box_location = pd.read_excel("./yolo-result/yolo_result.xlsx")

box_location["image_path"] = box_location["image_path"].apply(lambda x : x.replace("./Demo/test/", ""))
box_location["image_box"] = box_location["image_box"].apply(lambda x : eval(x))
```

```
    '''YOLO_v5 모델 output(좌표값)을 input으로 크롭한 뒤 classification(VGG16) 모델 로드'''
    try:
        while image_p == box_location["image_path"][check]:
            if box_location["image_box"][check] != 0: #내용이 없는 경우에 빈 파일 저장

                INPUTs = INPUT.crop(box_location["image_box"][check])
                INPUTs = INPUTs.resize((224,224))
                # INPUTs.save('./crop-result/' + str(num) + '.jpg')

                INPUTs = img_to_array(INPUTs)
                INPUTs = INPUTs.reshape(1, 224, 224, 3)
                INPUTs = INPUTs.astype('float32')
                INPUTs = INPUTs - [123.68, 116.779, 103.939]

                CNN_model = load_model('./weight/VGG16_에폭10.h5')
                result = CNN_model.predict(INPUTs)

                '''predicts > 0.49이면 helmet , predict <= 0.49이면 head로 분류'''
                if result >0.49:
                    # predicts.append(f"{result[0][0]:.6f}")
                    classification.append('helmet')
```

- Yolov5x의 Output으로 나온 box에 CNN 분류 모델을 이용하여 Head-Helmet 분류 진행
- 임계값을 0.5로 설계하여 CNN 결과가 0.5보다 크면 Helmet, 작으면 Head로 분류
- YOLO-CNN 모델의 결과물은 사진 1장에 .TXT 1개로 저장하도록 구현
- +AI 메이커톤 Test 데이터를 이용하여 YOLO-CNN 모델 평가

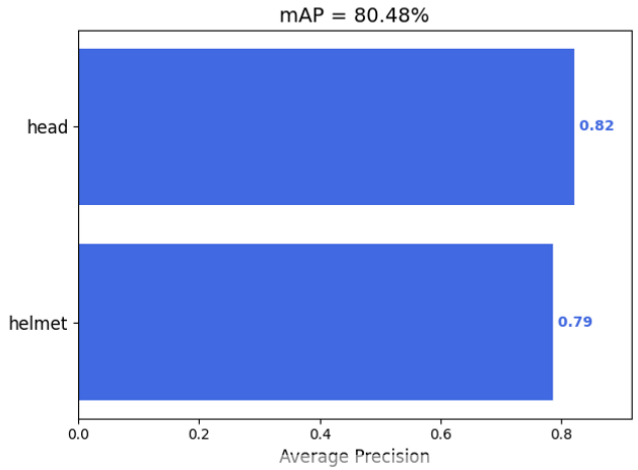
데이터수집 | 학습-검증 데이터 Split(0.9) | 사진 내 Head 추출 | 학습-검증 데이터 Split | 데이터 증강 | 모델 생성 | **모델 결합** | 모델 평가

+AI 메이커톤 데이터셋(5,000)
+AI 메이커톤 Test(500)
+AI 메이커톤 Train(4,500)
Roboflow 데이터셋(7,041)
Head-Helmet 데이터셋 (48,906)
Head-Helmet Train (36,605)
Head-Helmet Validation (12,301)
Head-Helmet Train (54,565)
CNN 학습 (VGGNet16)
Yolov5X 구현
Yolo-CNN 결합
성능 평가

mAP-calculation.py

3-1. 데이터 분석 수행

Final Step) 모델 평가



- https://github.com/Cartucho/mAP에서 제공하는 mAP code를 바탕으로 탐지&분류 성능 평가
- 총 500장의 Test 데이터를 가지고 성능을 평가한 결과, Head(82%), Helmel(79%)로 나타났으며, 최종적으로 80.48%의 성능을 보임

데이터수집 | 학습-검증 데이터 Split(0.9) | 사진 내 Head 추출 | 학습-검증 데이터 Split | 데이터 증강 | 모델 생성 | 모델 결합 | 모델 평가

+AI 메이커톤 Test(500)

+AI 메이커톤 데이터셋(5,000)

+AI 메이커톤 Train(4,500)

Roboflow 데이터셋(7,041)

Head-Helmet 데이터셋 (48,906)

Head-Helmet Train (36,605)

Head-Helmet Validation (12,301)

Head-Helmet Train (54,565)

CNN 학습 (VGGNet16)

Yolov5X 구현

Yolo-CNN 결합

성능 평가

3-2. 데이터 분석 결과

YOLO 결과 예시 — CNN 결과 예시 — Hard_hat_workers1513.txt

label	confidence	xmin	ymin	xmax	ymax
head	0.999952	159	217	147	210
helmet	1.000000	309	411	99	228
helmet	0.999985	380	415	120	216

LAB 423

감사합니다

충북대학교 빅데이터전공 최우석
충북대학교 빅데이터전공 이주연
충북대학교 빅데이터전공 유연주