

Hard Hat Detection with YOLOv3

Abstract

최근 산업 재해 발생률은 건설업과 제조업에서 대부분을 차지한다. 이러한 재해를 줄이기 위해서는 안전모 착용을 관리 및 감독하는 것이 중요한데, 감독자의 수동 관리는 그 한계가 존재한다. 따라서 우리는 Deep Learning Technology를 현장에 적용하여 문제의 한계를 극복하고, 앞으로 나아가야 할 방향에 대하여 제시하고자 한다. 특히 실시간으로 감지해야 하는 task의 특징으로 인해 YOLO v3를 활용하여 Dataset preprocessing부터 Training, 그리고 Evaluation까지의 A to Z 과정을 소개한다. 더불어 실험을 통해 얻은 인사이트와 평가를 바탕으로, 모델의 성능 개선 방향과 실제 현장에 적용하기 위한 방향에 대하여 논의하고자 한다.

1. Introduction

내외전기통신저널 기사[1]에 따르면, 지난 2018년 한 해의 산업재해로 인한 경제적손실추정액은 약 25조2000억원이며, 재해자수는 10만명을 넘어서었다. 실제로 매해 발생하는 경제적손실과 재해자수는 최근 5년간 비슷한 수준으로 유지되고 있으므로, 근본적으로 산업재해를 저감하기 위한 노력이 필요하다.

파이썬 툴을 활용하여 한국산업안전보건공단의 데이터[2]를 가지고 2018년 한 해의 산업 재해 발생률을 분석한 결과, 아래의 <그림1>과 같은 양상을 보임을 확인할 수 있다.



그림1. 2018년 산업별 산업재해자 수

전체 산업군 가운데 건설업(47%), 제조업(46%)이 압도적으로 비율이 높은 것을 확인할 수 있다. 두 산업군의 사고 발생 원인을 분석한 결과는 아래 <그림2>에서 확인할 수 있다.



그림2. 건설업, 제조업 산업재해 원인

건설업의 경우 (물체가)떨어짐, (재해자가)넘어짐, (재해자가)물체에 맞음이, 제조업의 경우에는 (물체에)끼임, 절단/베임/찔림, (재해자가)넘어짐, (물체가)떨어짐 등이 각 분야의 재해자 발생 원인의 절반을 넘는 비율을 차지하고 있다. 특히 앞서 언급한 재해자 발생의 원인들은 모두 두부(頭部)의 상해를 유발할 가능성이 높은 항목으로 구성되어 있음을 확인할 수 있다. 실제로 고용노동부의 2018년 사망재해 분석 결과를 살펴보면 사망자의 상해 부위 중 두부의 손상이 압도적으로 큰 비율을 차지한다.[3] 이러한 사실들로 미루어 봤을 때, 만약 재해자가 안전모를 착용했다면 경미한 부상으로 그칠 수 있었던 사고가 안전모를 미착용 함으로써 사망사고로 이르게 되었을 소지가 매우 높다. 따라서 건설업과 제조업 현장의 작업자를 대상으로 안전모 착용 여부의 관리를 강화한다면 두부의 손상으로 하여금 일어나는 산업재해의 저감을 예상할 수 있다.

현재 국내에서는 작업자의 안전을 보장하기 위해 산업안전보건법[4]을 법제화하여 사업주와 근로자의 안전의무를 제정하고, 사업주는 사업장을 실질적으로 관리하는 관리감독자 및 안전관리자를 의무적으로 두게하여 현장작업자의 보호구 착용을 관리하고 있다. 그러나 매해 줄지 않고 비슷한 수준으로 유지되는 재해자의 수로 미루어 보았을 때, 소수의 관리감독자 및 안전관리자만으로 모든 작업현장의 작업자를 관리하는 것은 분명히 물리적, 인적 한계가 있다. 따라서 인적자원에 구애받지 않으면서 현장 전체의 안전모 착용 여부를 단속할 수 있는 자동화된 관리시스템이 필요하다.

이러한 이유로, 시시각각 변할 수 있는 작업자의 안전모 착용 여부를 시공간적, 인적 자원의 한계를 받지 않고 탐지하는 방법을 제시한다. 이를 위해서는 사람의 눈을 대신할 Vision 시스템이 필수적이며, 보다 자동화된 방식을 위해서는 Deep Learning Vision이 필요하다. Deep Learning Vision을 이용해 관리자 없이 자동화된 시스템만으로 작업자의 안전모 착용 여부를 확인하고 경고하는 시스템을 만들어 현장에 도입함으로써 현장 작업자의 안전을 보장하고, 더 나아가 우리나라 산업실태를 개선하고자 한다.

한편, 현재 다양한 산업군에서 Deep learning 기술을 산업에 적용하여 생산성과 작업 효율을 향상시키고, 작업자의 안전 보장을 꾀하고 있음은 아래의 사례를 통해 확인할 수 있다.

- 1) 수행 중 사고 발생 위험이 높아 재해자가 발생할 여지가 있는 콘크리트의 균열 및 박락의 육안점검을 대체할 이미지에 기반한 Deep Learning 콘크리트 탐지 기술 연구논문[5]
- 2) 도로 및 교량과 같은 SOC건물의 안전점검을 영상에 기반한 Deep Learning기술로 대체함으로써 객관적이고 자동화된 점검시스템을 개발하고자 하는 사례 논문[6]
- 3) 포스코 기업에서는 영상분석에 Deep Learning 알고리즘을 탑재하여 제철소 현장에 특화된 CCTV인프라를 구축했다. 제철소에 특화된 설비,재료,조업 등의 정보를 학습시켜 CCTV를 통해 설비 번호를 영상에서 자동으로 인식하고 추적해 설비 효율화를 높이고, 전수검사 또한 가능하게 만들었다.[7]

2. Related Work

2.1. Deep Learning Vision

지난 몇 년간 Deep Learning은 Computer Vision부터 Natural Language Processing(NLP)을 넘어 GAN, Reinforcement Learning 등 다양한 분야에서 수많은 문제들을 해결하는데에 사용되어왔다. 특히 Computer Vision 분야에서의 Deep Learning의 기여는 상당한데, 이는 인간의 감각 중에서 가장 많이 사용하는 감각이 시각일 뿐더러 시각에 기반한 단순 업무들이 많았던 탓이다. 때문에 학계에서도 Vision과 관련하여 가장 많은 논문들이 쏟아져나오고 있으며, 업계에서도 자율 주행차, 비전 검사를 통한 불량품 추출, 얼굴 인식 등 다양한 분야에서 가치를 창출하고 있다. 이러한 Deep Learning Vision 분야는 적용하고자 하는 분야의 목적에 따라 다음과 같이 세부적인 task로 분류할 수 있다.

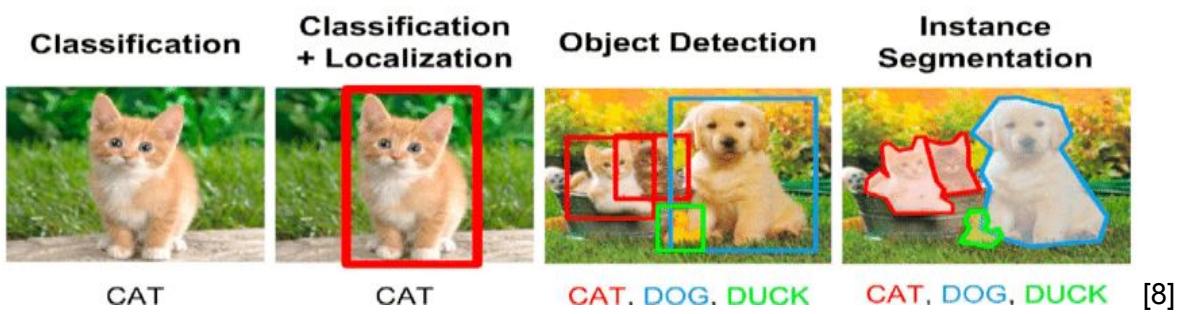


그림3. Deep learning Vision의 세부 분야

먼저 Classification은 하나의 물체(이하 Object)를 포함한 이미지를 학습하여, 임의의 이미지 안에 어떤 Object가 존재하는지를 분류하는 문제이다. 만약 Cat, Dog, Duck을 분류하는 문제를 해결하고 싶다면, 각 Object에 해당하는 이미지들과 레이블을 모아 모델을 학습할 수 있다. 두 번째는 Classification + Localization이다. 이는 기존 Classification 문제에서 한 발 더 나아가 이미지 안에 Object가 어디에 존재하는지 박스로 위치 정보(이하 Bounding Box)까지 예측하는 문제이다. 마찬가지로 이미지 안에 하나의 Object를 포함한 이미지를 전제로 한다. 해당 문제를 해결하기 위해서는 이미지, 레이블 뿐만 아니라 Bounding Box의 좌표 정보를 필요로 한다. 세 번째는 Object Detection이다. 이전 문제들은 이미지 내에 하나의 Object에 대해서만 예측을 한다. 하지만 실생활의 문제들은 이미지 또는 카메라 내에 항상 하나의 Object만 존재하도록 설정할 수는 없을 것이다. 따라서 이미지 내의 다수의 Object에 대한

Classification, 그리고 Localization을 수행하는 문제를 Object Detection 문제라고 정의한다. 마지막으로 Instance Segmentation은 Object Detection에서 더 나아가, 이미지 내의 각 Object에 대한 정확한 위치 정보까지 추출해내는 문제이다.

이 외에도 Deep Learning Vision 분야에는 Style Transfer, Everybody Dance Now, Pose Estimation, Image Resolution 등 굉장히 많은 분야가 존재하며, 이러한 학술적인 연구들을 바탕으로 기업에서는 다양한 솔루션을 개발하고 있다.

2.2. Object Detection

Object Detection은 Image 내에 존재하는 여러 개의 Object들에 대하여 Classification과 Localization을 수행하는 문제이다. 학계에서 연구되고 있는 Object Detection 관련 모델은 구조에 따라 크게 2가지로 구분할 수 있다. Object Detection은 Classification과 Localization을 모두 수행해야 하는데, 이를 두 개의 architecture로 나누어 순차적으로 수행하는 구조를 Two-Stage Detection Model이라고 정의한다. 반면에 하나의 architecture로 두 가지 task를 동시에 수행하는 구조를 One-Stage Detection Model이라고 정의한다. 모델의 구조적 특징으로 인해 일반적으로 One-Stage Detection 모델은 학습 및 추론 속도가 빠른 대신 정확도가 낮은 편이며, Two-Stage Detection Model은 정확도가 높은 대신 학습 및 추론 속도가 느린 편이다. 대표적인 모델로써 One-stage 방식의 YOLO와 Two-Stage 방식의 Faster R-CNN이 존재한다.

Object Detection 모델은 학계에서뿐만 아니라 일상생활의 문제점을 해결하는데에도 큰 기여를 하고 있다. 이를 적용하는 대표적인 사례[2]는 자율주행 자동차, CCTV, OCR(Optical Character Recognition), 신체 인식, 제품 결함 분석, 스포츠 경기 분석, 무인 점포 계산 등 무궁무진하다.

2.3. YOLO v3

YOLO는 ‘You Only Look Once’의 약자로서, One-Stage Detection 모델의 작동 원리를 한 문장으로 함축해놓았다. YOLO 모델의 구조는 Classification과 Localization에 대한 추론을 같은 matrix 내에서 처리하도록 설계되어 있다. 따라서 Classification 정보와 Localization 정보를 동시에 추론한다. 즉, 한 번에 필요한 정보를 도출하므로 You Only Look Once, YOLO라는 이름이 붙여졌다.

특히 YOLO는 Joseph Redmon이라는 저자가 처음 논문을 발표했는데, 모델에 대한 애착이 강해서 이후에도 v1, v2, v3 등의 버전을 논문으로 발표하며 지속적으로 모델의 성능을 키워나갔다. 여담이지만 저자는 자신의 모델이 군사용 목적으로 사용되는 것에 환멸을 느껴 현재는 학계에서 연구를 그만 둔 상태이다. 하지만 그 외에 YOLO에 매력을 느낀 연구자들이 지속적으로 다음 버전을 연구 중에 있으며, 공식적으로 논문으로 출판된 버전은 v4까지이다.

즉, Joseph Redmon이 연구했던 마지막 모델은 YOLO v3 [5]인데, 해당 모델에 대한 구조는 아래와 같다.

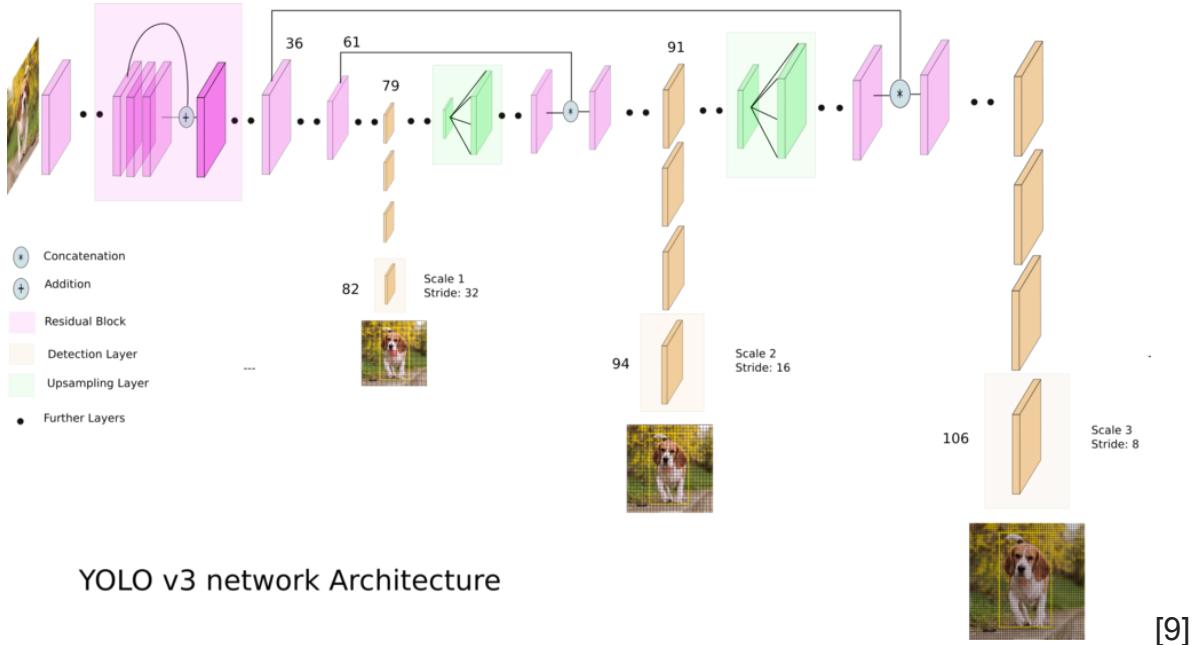


그림4. YOLO v3모델의 구조

[9]

모델은 총 106개의 layer로 구성되어있다. 모델에 이미지를 입력하면 가장 먼저 ‘feature extraction’ 역할을 수행하는 Classification Model 구조를 통과한다. 이는 이미지 내의 위치 기반으로 특징을 추출하는 CNN(Convolutional Neural Network) 구조들의 집합으로써, 각 layer들은 전체적인 특징부터 세부적인 특징까지 추출한다. 해당 Classification Model 구조는 기존의 여러 Classification Model을 그대로 사용할 수도 있지만, 해당 논문의 저자는 ‘Darknet-53’이라는 구조를 직접 만들어 Classification Model로 사용했다. Darknet-53은 초당 연산량이 빠른 것이 특징으로 GPU를 좀 더 효율적으로 사용할 수 있는 구조로 모델링되어있다.

이후 36번 layer부터 본격적인 Classification과 Bounding Box Regression을 수행하기 위한 구조를 통과하게 된다. YOLO v3는 기존 버전들과는 달리 크게 3가지 scale에 대하여 결과를 추출한다. 82번, 94번, 106번 layer가 이에 해당하며, 이를 통해 작은 scale의 물체부터 큰 scale의 물체까지 모두 탐지할 수 있는 특징을 갖는다.

추가적으로 FPN(Feature Pyramid Network)의 구조와 비슷한 구조를 사용하기 위해 Concatenation, Addition 등의 기법을 사용하였고, 이를 통해 자칫 모델의 구조가 너무 커서 정보 손실이 될 우려를 감소시켰다.

결과를 추출하는 layer의 구조를 좀 더 구체적으로 살펴보면 아래와 같다.

Image Grid. The Red Grid is responsible for detecting the dog

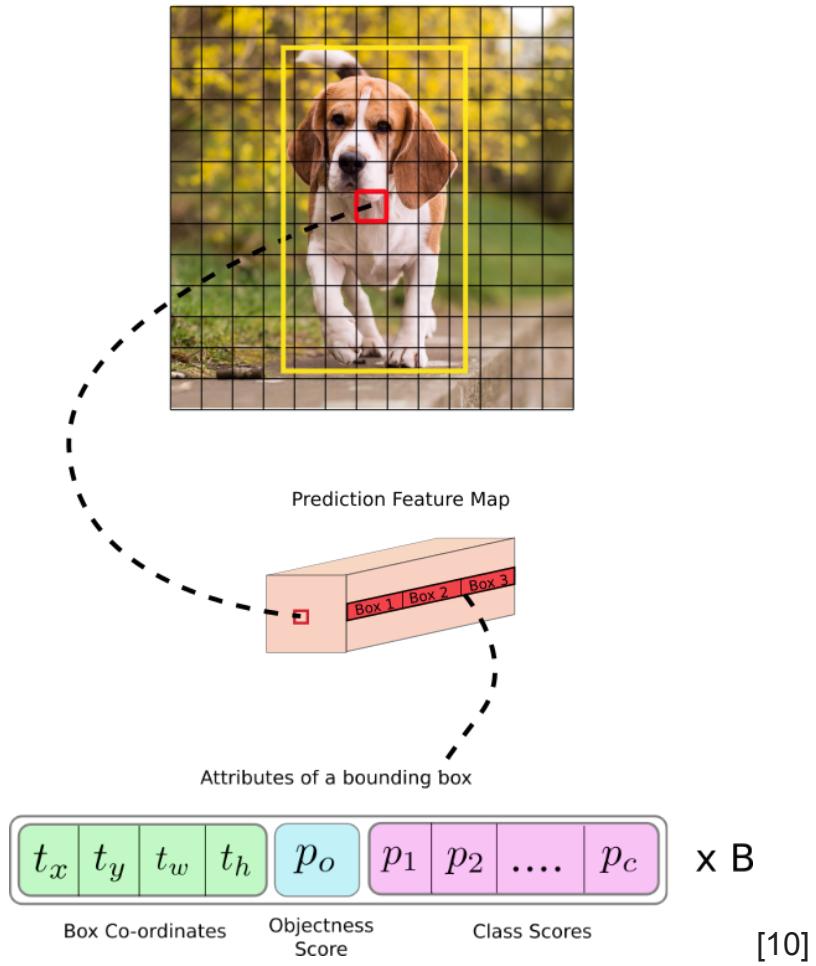


그림5. YOLO 모델의 layer 구조

output layer(82, 94, 106 layer)는 위와 같은 정보를 출력한다. 먼저 각 layer는 grid cell 형태로 나누어져 있다. 82번 layer는 13x13개, 94번 layer는 26x26개, 106번 layer는 52x52개의 grid cell로 나누어 있으며, 각 grid cell은 3개의 Bounding box를 가지고 있다(bounding box의 개수는 사용자가 직접 설정해주는 값이다). 따라서 일반적인 YOLO v3는 총 $((13 \times 13) + (26 \times 26) + (52 \times 52)) \times 3 = 10647$ 개의 bounding box를 예측할 수 있다고 볼 수 있다. 각 bounding box는 위의 형태와 같이 이루어져 있다. 먼저 bounding box 좌표를 나타내는 t_x , t_y , t_w , t_h 가 있으며, bounding box에 물체가 있을 확률을 나타내는 p_o (objectness score)가 있고, 각 class가 존재할 확률을 나타내는 p_1 , p_2 , ..., p_c (class scores) 값이 존재한다. 이러한 prediction value와 이미 가지고 있는 true value의 차이를 줄이는 방향으로 가중치를 업데이트시키는 과정이 모델 학습의 원리라고 할 수 있다.

2.4. IoU(Intersection over Union)

전통적인 Classification 문제는 전체 문제 중에 맞춘 문제의 비율로 간단하게 모델의 성능을 평가할 수 있다. 흔히 이 평가지표를 accuracy(정확도)라고 한다. 그렇다면 Object Detection 모델의 성능은 어떤 평가지표를 사용할까? Object Detection은 탐지하고자 하는 물체의 위치에 대한 Bounding Box 뿐만 아니라, 그 안에 어떤 물체가 있는지까지도 분류를 할 수

있어야 한다. 즉, Classification 문제와 Bounding Box Regression 문제를 동시에 검증하면서 모델의 성능을 평가해야만 한다. 특히 일반적으로 Bounding Box안의 물체를 제대로 Classification하지 못한다면 이는 아예 실패로 간주된다. 따라서 Classification을 제대로 수행한 Bounding Box에 대하여 그 위치를 얼마나 잘 탐지했는지를 평가해야 한다. 이는 IoU라는 평가지표를 통해 모델의 성능을 평가할 수 있다. 단어의 뜻을 잘 생각해보면 그 의미를 유추해볼 수 있는데, Intersection over Union이라는 풀네임을 가지며 교집합/합집합이라는 의미이다. 즉, 정답 Box인 GT(Ground Truth)와 예측한 Box인 Bounding Box 면적의 합을 분모로, 겹치는 면적을 분자로 하여 IoU 값을 구할 수 있다. 두 Box가 아예 분리되어 있으면 값은 0이고, 완전히 겹친다면 값은 1이다. 또한 이 값은 0과 1 사이의 범위를 가지며, 많이 겹칠수록 그 값은 1에 가까워진다. IoU가 1에 가깝다는 것은 우리가 예측한 Bounding Box가 실제 Object를 잘 탐지했다는 의미가 된다. Object Detection 분야에서는 IoU가 어떤 값 이상일 경우 Positive라고 예측하고, 그 외에는 Negative라고 예측하게 된다. 즉, 어떤 threshold 값을 가지게 되는데 일반적으로 0.5 이상이면 정답을 맞춘 것으로 평가하곤 한다.

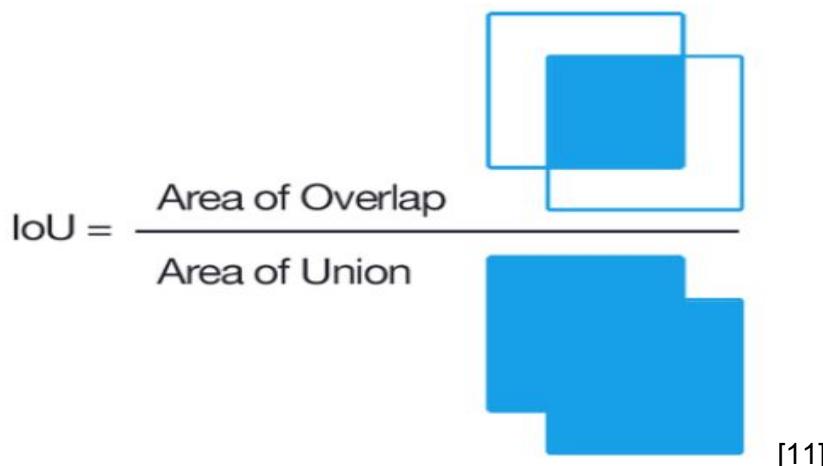


그림6. IoU 계산 예시

[11]

2.5. Precision and Recall

Precision과 Recall은 모델의 성능을 평가하는 지표로써, Confusion Matrix에서 도출되는 값이다.

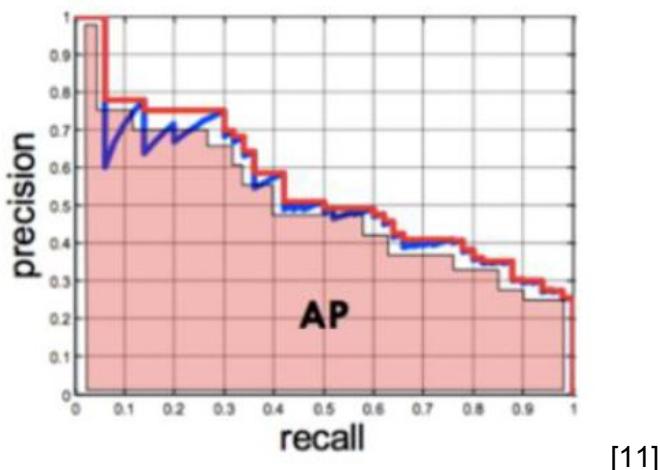
Object Detection에서 의미하는 Precision은 모델이 예측한 Bounding Box 중에서 제대로 예측한(미리 설정한 IoU threshold 기준) 개수의 비율이라고 할 수 있다. 해당 평가지표는 ‘과연 모델이 예측한 것들은 믿을만한가?’를 평가하는 것이라고 생각할 수 있다. 이 값을 올리기 위해 모델은 하나하나의 Bounding Box를 신중하게 예측해야 할 것이다.

반면에 Recall은 기준에 존재하는 전체 GT 중에서 모델이 제대로 예측한(미리 설정한 IoU threshold 기준) 개수의 비율이라고 할 수 있다. 해당 평가지표는 ‘모델은 탐지하고자 하는 물체를 얼마나 많이 맞췄는가?’를 평가하는 것이라고 생각할 수 있다. 이 값을 올리기 위해 모델은 최대한 많은 Bounding Box를 예측해내면 된다.

즉 두 값은 IoU threshold 설정값에 따라 서로 trade-off 관계를 가지게 된다. 따라서 두 값을 모두 고려해야 할 필요성이 있다.

2.6. AP(Average Precision), mAP(mean Average Precision)

Precision과 Recall은 서로 trade-off 관계이기 때문에, 두 평가지표를 동시에 고려하기 위해 AP를 주로 사용한다. IoU threshold값을 일정 단위별로 바꿔가면서 각 Precision과 Recall을 측정하고 이를 그래프로 나타낼 수 있다. 반비례 관계로 인해 x축 값(Recall)이 증가할수록 y축 값(Precision)은 감소하는 그래프를 얻는다. Precision과 Recall을 동시에 고려하기 위해서 그래프의 아래 면적을 모델의 성능 평가 지표로 사용하게 되고, 이를 Average Precision이라고 한다. 또한 각 클래스별로 AP를 모두 구해 평균을 낸 값이 mAP이며, 최종적으로 이 값을 모델의 성능 평가지표로 사용한다. mAP는 0과 1 사이의 값을 가지며, 값이 클수록 모델의 성능 또한 높다는 것을 의미한다.



[11]

그림7. Average Precision 계산 예시

3. Experiments

3.1. Experiment Environment

실험을 위해 사용한 software, hardware 및 skill은 다음과 같다.

CPU(google colab pro environment)	Intel(R) Xeon(R) CPU @ 2.30GHz
GPU(google colab pro environment)	Tesla P100
RAM(google colab pro environment)	25.51GB
OS(google colab pro environment)	Ubuntu 18.04.3 LTS
Language	Python, C
Deep Learning Framework	Tensorflow, Pytorch
Reference code	https://github.com/AlexeyAB/darknet

표1. 실험 환경

3.2. Hardhat Dataset

Dataset을 수집하는 방법은 공개된 Dataset 활용, Web Crawling, 직접 수집 등 다양한 방법이 존재한다. 시간을 절약하기 위해 Hardhat Detection 이미지에 관한 공개된 Dataset을 우선적으로 찾아보았고, 아래 github을 통해 Dataset을 확보할 수 있었다. 해당 github의 저자가 공개된 자료에 대하여 'MIT license'를 표기를 통해 사용 권한을 확인했다.

[Hardhat Dataset Github URL]

<https://github.com/njvisionpower/Safety-Helmet-Wearing-Dataset>[12]

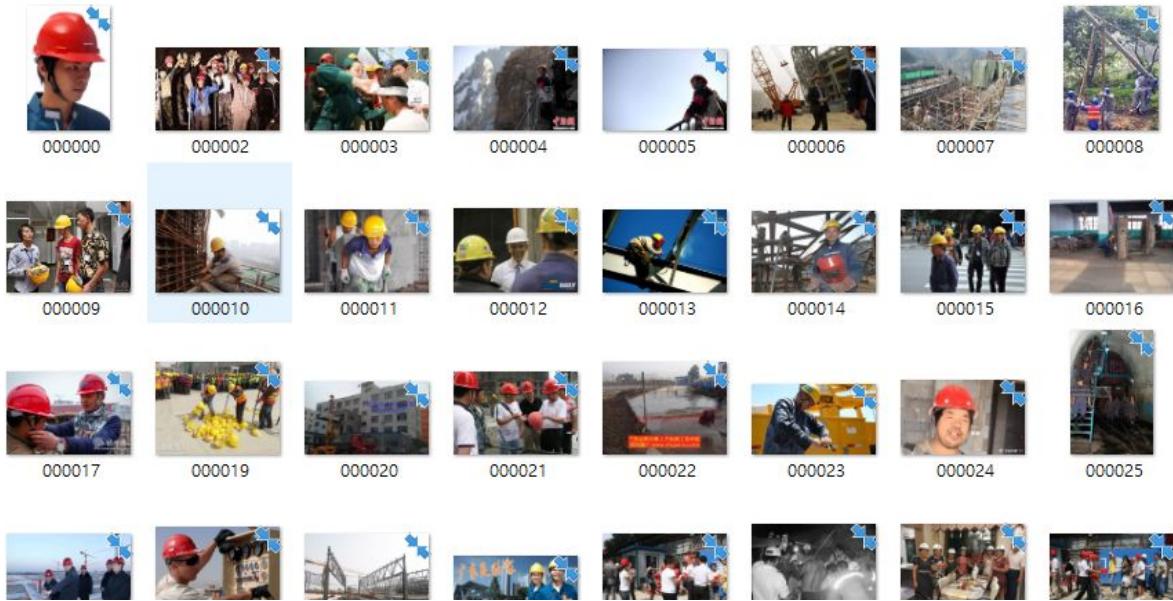


그림8. 이미지 Dataset 예시

Dataset의 구성은 다음과 같다.

JPEGImages 7581

```
|--- Origin 1608  
|--- Part2 1633  
|--- PartA 1999  
|--- PartB 2341
```

Annotations 7581

```
|--- Origin 1608  
|--- Part2 1633  
|--- PartA 1999  
|--- PartB 2341
```

JPEGImages는 안전모를 착용한 사람과 미착용한 사람에 대한 Image(.jpg file) 7581장을 포함하고 있으며, Annotations는 각 Image 안에 있는 사람들의 얼굴 위치와 안전모 착용 여부에 대한 정보가 담겨 있는 Annotation(.xml file) 7581장을 포함하고 있다.

Dataset에 대하여 조금 더 자세하게 설명하면 다음과 같다. 각 .jpg file에 대하여 동일한 이름을 갖는 .xml file이 1대1 대응으로 존재한다. 특히 .xml 파일은 오른쪽 아래와 같은 내용을

담고 있는데, file name, Image size, Image 내에 존재하는 Object 별 label 및 위치 좌표 등이 담겨 있다. 예를 들어 Dataset 내의 000178.jpg 이미지와 대응하는 xml file은 000178.xml file인데, 해당 파일에 표기되어 있는 object가 Image 내의 object를 하나씩 지칭한다. 각각의 object는 object의 이름(name), 위치 좌표(bndbox = xmin, ymin, xmax, ymax)의 정보를 포함한다. 또한 각 Object의 name은 'hat' 또는 'person'라는 class로 labeling 되어있으며, 'hat'은 안전모를 착용한 사람을 의미하고, 'person'은 안전모를 착용하지 않은 사람을 의미한다. 예로 설명한 000178.jpg와 000178.xml은 아래와 같다.

000178.jpg



000178.xml

```

<object>
  <name>hat</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>
    <xmin>367</xmin>
    <ymin>460</ymin>
    <xmax>544</xmax>
    <ymax>670</ymax>
  </bndbox>
</object>
<object>
  <name>hat</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>
    <xmin>511</xmin>
    <ymin>346</ymin>
    <xmax>763</xmax>
    <ymax>587</ymax>
  </bndbox>
</object>
<object>
  <name>person</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>
    <xmin>180</xmin>
    <ymin>576</ymin>
    <xmax>322</xmax>
    <ymax>740</ymax>
  </bndbox>
</object>

```

그림9. 동일한 이름을 갖는 jpg와 xml파일 예시

먼저, 해당 Dataset을 사용하기 위해 preprocessing을 해주는 과정을 거쳤다. Image의 수가 7581장인데, 현재 가지고 있는 Google Colab Pro 자원으로 모든 Image를 학습하기에는 시간적, 자원적으로 문제가 발생한다. 특히 Google Colab Pro는 학습 시간이 24시간 이상 지속될 경우 세션이 종료되는 이슈가 존재한다. 따라서 주어진 자원에서 최대한으로 Dataset을 활용하기 위해 Origin 1608장을 Train Dataset으로, Part2 100장을 Test Dataset으로 선정했다.

모델에 학습시키기 위해 .xml 파일을 .txt 파일로 변환하는 과정에서 지속적인 error가 발생했는데, 디버깅 결과 00377.xml 파일 내에 object 몇 개가 'dog'로 labeling 되어 있었다. 실제 Image를 통해 강아지를 나타내는 object는 존재하지 않았음을 확인했고, 따라서 Bad data로 간주하여 해당 Image와 Annotation file은 학습에서 제외시켰다.



그림10. 삭제한 Bad data

따라서 Train Dataset 1607 Images, Test Dataset 100 Images로 데이터셋을 구성했다.

3.3. Transfer Learning

우리의 목표는 기존에 학습되어져 있는 YOLOv3를 우리가 해결하고자 하는 문제에 맞게 재구성하여 학습시키는 것이다. 흔히 이를 transfer learning이라고 한다. 우리는 Google Colab 환경에서 학습할 수 있는 모델이 필요했기에, 'AlexeyAB'가 C language로 구현한 github를 활용했다.[13]

그림11. 'AlexeyAB'의 github 화면

최근 학계에서는 딥러닝 모델을 논문으로 발표할 때 해당 모델을 구현한 code와 pre-trained weights까지 함께 공개하는 추세이다. YOLO v3 또한 구현 code 뿐만 아니라 pre-trained

weights가 함께 공개되어 있다. 여기서 pre-trained weights란, 모델이 특정 Dataset을 학습한 weights를 저장해서, 필요로 할 때 바로 사용할 수 있도록 해 준 weights를 의미한다. 해당 weights를 모델에 적용하면, 랜덤으로 초기화시키는 weights에 비해 학습 수렴 속도도 빠를 뿐더러 더 좋은 퍼포먼스를 내는 것으로 알려져 있다. 학계에서는 논문으로 발표한 Object Detection Model을 평가할 때 일반적으로 COCO dataset에 대한 성능을 기준으로 평가를 하는데, YOLO v3 또한 COCO dataset을 학습한 pre-trained weights를 제공한다(참고로 COCO dataset이란, 일상생활에서 볼 수 있는 80여 가지의 클래스에 대한 classification 정보와 bounding box 정보를 담고 있는 dataset이다). 모델을 transfer learning하기 전에, 학습 과정 없이 pre-trained weights를 적용하여 안전모를 탐지할 수 있는지 간단한 실험을 수행했다.



그림12. transfer learning과정 이전의 탐지 예시

간단한 실험 결과는 위 <그림12>와 같다. COCO dataset으로 pre-trained된 YOLOv3는 주어진 이미지에서 'person'과 'book'만을 탐지했다. 이러한 결과가 도출되는 이유는 우리가 사용한 pre-trained weights가 COCO dataset에 존재하는 class만을 탐지할 수 있는 능력을 가지고 있기 때문이다. 즉, COCO dataset 안에는 'person'이나 'book'과 같은 class가 존재하지만, 'hardhat'이라는 class는 존재하지 않기 때문이다. 따라서 우리가 해결하고자 하는 문제에 맞게 모델을 transfer learning하는 과정이 필요한 것이다.
모델을 Hardhat Dataset에 transfer learning하기 위해서는 몇 가지 설정들이 필요하다. 설정해주어야 할 파일들은 다음과 같다.

- .cfg file
- obj.data
- obj.names
- train.txt
- test.txt

1) .cfg file

yolov3.cfg 파일을 복사하여 해결하고자 하는 문제에 맞게 수정한 후, 'yolov3_Hardhat.cfg'로 다시 저장했다. 수정한 사항들은 다음과 같다.

- max_batches = 10000
- steps = 8000, 9000
- classes 2 (in the three YOLO layers)
- filters = 21 (in the three convolutional layers before the YOLO layers)

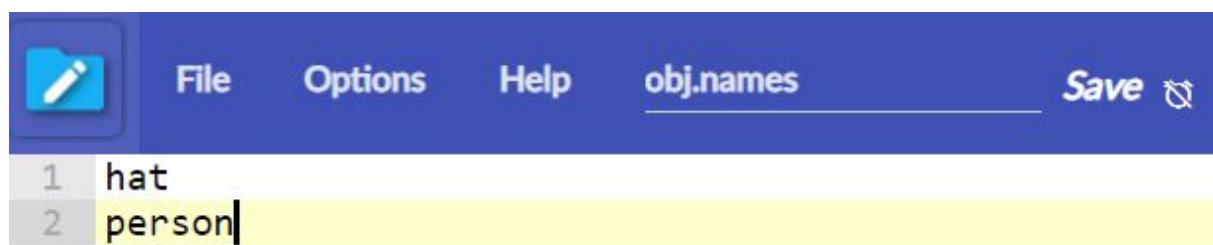
```
598
599 [convolutional]
600 size=1
601 stride=1
602 pad=1
603 filters=21
604 activation=linear
605
606
607 [yolo]
608 mask = 6,7,8
609 anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326
610 classes=2
611 num=9
612 jitter=.3
613 ignore_thresh = .7
614 truth_thresh = 1
615 random=1
```

그림13. cfg 파일의 수정 예시

이 때 filters=21인 이유는 YOLO v3 모델의 작동 원리 때문이다. (x, y, w, h, p0, c1, c2)으로 구성된 Bounding Box가 3개 존재하기 때문에 $7 \times 3 = 21$ 개의 filters를 갖는다. 참고로 c1은 hat, c2는 person을 의미한다.

2) obj.names and obj.data

obj.names 파일을 만들어서 class의 종류를 입력한다. 또한 obj.data 파일을 만들어서 class의 수, obj.names 파일의 경로, dataset의 경로, backup 폴더 경로를 입력한다. 모델을 학습한 후의 weights는 설정한 backup 폴더에 저장된다.
생성한 두 개의 파일은 data 폴더에 넣어주었다.



```

1 classes = 2
2 train = data/train.txt
3 valid = data/test.txt
4 names = data/obj.names
5 backup = /mydrive/Project/YOLOv3/darknet/backup/

```

그림14. obj.names와 obj.data 파일 생성 예시

3) train.txt and test.txt

마지막으로 모델을 학습할 이미지의 경로를 담고 있는 train.txt 파일과 모델을 평가할 이미지의 경로를 담고 있는 test.txt 파일을 생성한다.

```

train.txt content:
1 data/obj/000000.jpg
2 data/obj/000002.jpg
3 data/obj/000003.jpg
4 data/obj/000004.jpg
5 data/obj/000005.jpg
6 data/obj/000006.jpg
7 data/obj/000007.jpg
8 data/obj/000008.jpg
9 data/obj/000009.jpg
10 data/obj/000010.jpg
11 data/obj/000011.jpg
12 data/obj/000012.jpg
13 data/obj/000013.jpg
14 data/obj/000014.jpg
15 data/obj/000015.jpg
16 data/obj/000016.jpg
17 data/obj/000017.jpg
18 data/obj/000019.jpg
19 data/obj/000020.jpg
20 data/obj/000021.jpg
21 data/obj/000022.jpg
22 data/obj/000023.jpg
23 data/obj/000024.jpg
24 data/obj/000025.jpg
25 data/obj/000026.jpg
26 data/obj/000027.jpg
27 data/obj/000028.jpg
28 data/obj/000029.jpg
29 data/obj/000030.jpg
30 data/obj/000031.jpg
31 data/obj/000032.jpg
32 data/obj/000033.jpg
33 data/obj/000034.jpg
34 data/obj/000035.jpg
35 data/obj/000036.jpg
36 data/obj/000037.jpg
37 data/obj/000038.jpg
38 data/obj/000039.jpg

test.txt content:
1 data/obj/part2_000000.jpg
2 data/obj/part2_000001.jpg
3 data/obj/part2_000002.jpg
4 data/obj/part2_000004.jpg
5 data/obj/part2_000005.jpg
6 data/obj/part2_000006.jpg
7 data/obj/part2_000007.jpg
8 data/obj/part2_000008.jpg
9 data/obj/part2_000010.jpg
10 data/obj/part2_000011.jpg
11 data/obj/part2_000013.jpg
12 data/obj/part2_000014.jpg
13 data/obj/part2_000015.jpg
14 data/obj/part2_000017.jpg
15 data/obj/part2_000018.jpg
16 data/obj/part2_000021.jpg
17 data/obj/part2_000022.jpg
18 data/obj/part2_000025.jpg
19 data/obj/part2_000026.jpg
20 data/obj/part2_000028.jpg
21 data/obj/part2_000029.jpg
22 data/obj/part2_000030.jpg
23 data/obj/part2_000031.jpg
24 data/obj/part2_000035.jpg
25 data/obj/part2_000036.jpg
26 data/obj/part2_000041.jpg
27 data/obj/part2_000042.jpg
28 data/obj/part2_000043.jpg
29 data/obj/part2_000044.jpg
30 data/obj/part2_000045.jpg
31 data/obj/part2_000049.jpg
32 data/obj/part2_000054.jpg
33 data/obj/part2_000054.jpg

```

그림15. train.txt와 test.txt 파일 생성 예시

위에서 설정한 내용들을 모두 수행한 뒤 학습을 진행했고, 약 15시간에 걸쳐 모델 학습을 마쳤다. backup 폴더에는 1000 epoch 별 weights와 모든 학습을 끝낸 후의 final weights가 저장되어 있다. 또한 아래 그림을 통해 모델이 학습하면서 출력한 loss curve를 확인할 수 있다. 이는 epoch 별 average loss를 보여준다. github 저자는 모델의 성능을 확인하는 간단한 기준으로 loss 값이 2.0 이하가 나오면 좋은 성능을 기대할 수 있다는 가이드를 제시한다. 우리가 수행한 학습의 loss curve는 1000 epoch 근처에서 이미 2.0 이하의 loss 값을 얻을 수 있었기 때문에 좋은 성능을 기대할 수 있었다.

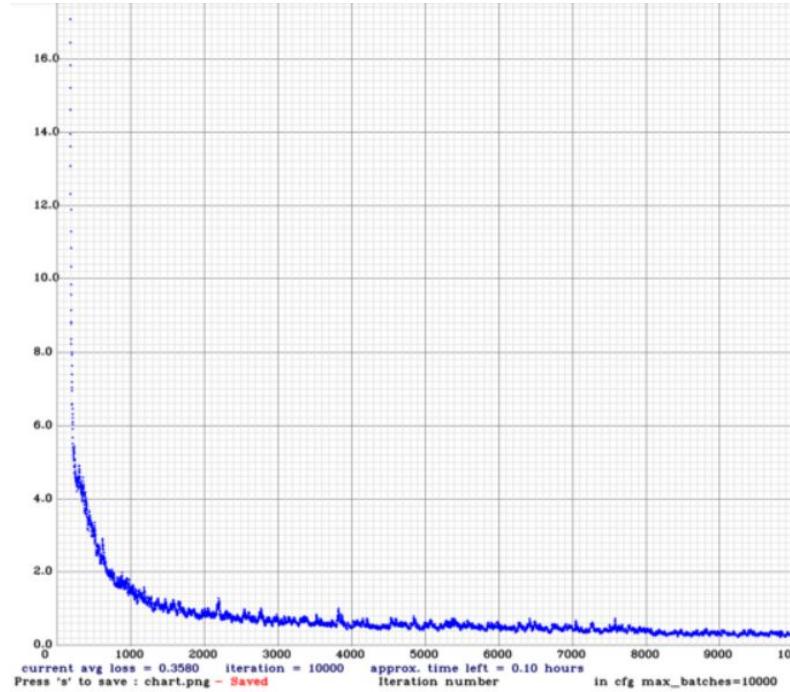


그림16. 실제 수행한 학습의 loss curve

3.4. Result & Evaluation



그림17. transfer learning과정 후의 탐지 예시

처음 사용했던 이미지를 다시 모델에 input으로 사용한 결과는 위 <그림17>과 같다. person과 book을 예측하던 모델은 이제 hat과 person(no hat)을 예측한 모델의 구조와 가중치로 학습한

상태이다. 각 사람에 대하여 안전모 착용 여부를 정확히 탐지한다는 것을 확인했다. 이 외의 몇 가지 이미지를 input으로 사용하면서 모델의 효용성을 평가했다.



그림18. 일반적인 사진의 탐지(1)



그림19. 일반적인 사진의 탐지(2)

먼저 일반적인 사진을 적용했을 때의 결과는 <그림18>과 <그림19>에서 확인할 수 있다. 안전모를 착용한 사람을 hat으로, 착용하지 않은 사람은 person으로 탐지하는 것을 확인할 수 있다. 이렇듯 일반적인 사진에는 모델의 성능이 잘 나타난다.



그림20. 원거리 촬영 이미지 탐지(1)



그림21. 원거리 촬영 이미지 탐지(2)

<그림20>과 <그림21>은 원거리 촬영 이미지에 대한 결과이다. 대부분 안전모를 착용한 사람을 hat으로 잘 탐지했지만, 자세히 확인해보면 몇몇 사람에 대해서 탐지력이 부족한 것을 확인할 수 있다.

위와 같은 정성적인 평가는 모델의 성능을 대략적으로 파악하는데에 도움이 된다. 추가적으로 모델의 성능을 수치적으로 평가하기 위해 처음 우리가 설정했던 Test Dataset 100장을 대상으로 정량적 평가를 수행했다. 일반적으로 Object Detection의 모델 성능 평가 지표는 'mAP'를 많이 사용하므로, 이에 대한 결과를 도출했다.

```

calculation mAP (mean average precision)...
100
detections_count = 395, unique_truth_count = 269
class_id = 0, name = hat, ap = 88.92%           (TP = 211, FP = 17)
class_id = 1, name = person, ap = 40.92%          (TP = 8, FP = 3)

for conf_thresh = 0.25, precision = 0.92, recall = 0.81, F1-score = 0.86
for conf_thresh = 0.25, TP = 219, FP = 20, FN = 50, average IoU = 78.07 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.649170, or 64.92 %
Total Detection Time: 4 Seconds

```

그림22. 모델 성능 평가

<그림22>의 빨간색으로 박스 친 부분을 유념해서 볼 필요가 있다. 먼저 'hat'(안전모를 착용한 사람) class에 대하여 88.92% AP를, 'person'(안전모를 착용하지 않은 사람) class에

대하여 40.92%의 AP를 도출했다. 따라서 이를 평균으로 계산한 mAP는 64.92%가 도출된다. Test Dataset 100장으로 대상으로 Detection 소요 시간은 총 4초가 걸렸다.

안전모 탐지는 이미지가 아닌 영상에 적용이 가능할 때 효용성이 있다. 모델 적용 환경이 대부분 작업이 진행 중인 공사 현장이기 때문이다. 따라서 동영상을 모델에 입력할 때 원하는 기댓값이 나오는지 inference하는 과정을 수행했다. 이를 위해 '다큐 3일'에서 방영한 '세종시 건설현장에서의 3일'[14] 영상의 일부분을 캡처하여 모델에 입력했다. 영상에 대한 정량적인 수치 측정 방법을 아직 구현하지 못했기 때문에, 출력 영상을 육안으로 확인하는 정성적인 평가 방법을 수행했다. 그 결과 안전모를 착용한 사람에 대하여 'hardhat'으로, 착용하지 않은 사람에 대하여 'person'으로 정확히 탐지를 한 것을 확인했다. 이를 통해 모델의 학습이 정상적으로 이루어졌고, 실시간으로 적용 가능하다는 것을 확인했다. 하지만 멀리서 찍은 영상에서의 성능, 특정 환경(먼지로 인해 뿐만 아니라 어두운 환경)에서 찍은 영상에서의 성능 등 범용성을 위한 평가를 수행하지 않았다. 이는 적용하고자 하는 환경의 영상을 사용하여 모델의 출력값을 확인해보고, 차후 보완점을 생각해야 할 것이다.

우리가 정성적으로 확인한 모델의 출력 영상은 유튜브에 업로드했고, 아래 URL을 통해 그 결과를 확인해 볼 수 있다.

URL)

<https://www.youtube.com/watch?v=jrzEf5ONquo>

4. Discussion

지금까지 우리는 데이터 분석을 통해 안전모 착용의 중요성을 파악하고, Dataset을 확보하였으며, 모델을 학습시켜 유의미한 결과를 도출했다. 이를 상용화 시켜 공사 현장에 적용하기 위해서는 모델의 성능을 더욱 개선하고, 실질적인 적용 가이드라인을 고민해 볼 필요가 있다. 따라서 프로젝트를 진행하면서 얻은 인사이트를 바탕으로, 현재 모델의 문제점을 분석하고 앞으로 나아가야 할 방향에 대하여 논의한 결과를 제시한다. 논의 내용은 모델의 성능을 향상시키는 방법에 대한 고안과 현장에 모델을 도입했을 시 발생할 수 있는 문제점과 해결 방안, 프로젝트 확장 방향 등이 있다.

4-1. 모델 성능 향상 방법 고안

먼저 모델의 성능을 기술적으로 향상시키는 방법이다. 우리가 수행한 프로젝트는 기한이 정해져있고 Google Colab에서 수행하는 등 시간적, 공간적 한계가 존재했다. 따라서 이러한 한계가 해결된다면 좀 더 많은 자원을 활용하여 모델의 성능 향상을 기대할 수 있을 것이다. 따라서 이러한 문제가 해결된다는 가정 하에 모델의 성능을 향상시키기 위해 어떠한 Experiment를 수행할 수 있는지 제시한다.

4.1.1. Class Imbalance Problem 해결

Results에서 알 수 있듯이, 우리가 학습한 모델은 hat(안전모 착용, 88.92%)에 비해 person(안전모 미착용, 40.92%)에 대한 mAP가 현저히 떨어짐을 확인했다. 즉, 안전모를

착용하지 않은 사람에 대한 탐지가 부족한 상황이다. 하지만 본래의 목적인 안전모 미착용 인원에 대한 감지를 달성하지 못하므로 이는 큰 문제가 될 수 있다.

이러한 문제가 발생하는 원인은 우리가 학습에 사용한 Dataset에 존재하는 Class Imbalance Problem 때문이다. python의 os module을 이용한 결과, 우리가 사용한 Training Image 1607장에 해당하는 Object의 비율은 아래와 같음을 확인했다.

- hardhat: 5242
- person: 1198

즉 Dataset안에 hardhat에 해당하는 Object가 압도적으로 많기 때문에, hardhat을 더 잘 예측하는 방향으로 모델의 학습이 진행되었다. 이러한 문제는 Deep Learning 모델을 학습하는데에 발생하는 일반적인 문제 중의 하나로써, 시도해볼 수 있는 해결 방법은 다음과 같다.

첫째, Dataset을 재구축한다. 우리는 Original 카테고리에 한정하여 Training Dataset을 구축했다. 따라서 person(안전모 미착용) Object가 많은 PartB 카테고리의 Dataset을 섞는 방법이 필요하다. 이러한 방법을 통해 hat과 person의 비율을 어느 정도 맞춰줄 수 있을 것이다.

둘째, Loss Function을 바꾼다. YOLOv3는 일반적으로 Dataset을 학습할 때 'Binary Cross Entropy function'을 사용한다. 이는 Dataset의 비율이 일정할 때 사용하는 방식으로, 모든 class에 대하여 동일한 가중치를 부여한다. 하지만 현재 모델이 가지고 있는 문제는 class의 비율이 달라서 많은 쪽으로 학습이 더 이루어지기 때문에, loss function을 변경하여 이를 고의적으로 조정해주는 방법이 필요하다. 예를 들어, 학습이 잘 안 이루어지는 class에 더 높은 가중치를 부여하여 학습의 방향성을 맞춰주는 방법이 있다. 'A survey of loss functions for semantic segmentation' 논문[15]에 따르면, Class Imbalance Problem에 적용할 수 있는 여러 loss function이 제시되어 있다. 대표적인 예로써, Balanced Cross-Entropy, Focal Loss 등이 있다.

4.1.2. Small Object Detection에 대한 성능 향상

정성적으로 모델의 성능을 평가한 내용을 살펴보면, 우리의 모델은 작은 Object에 대하여 탐지력이 다소 떨어진다는 것을 확인했다. 이는 모델이 작은 Object에 대한 학습이 부족한 상황이기 때문에, 마찬가지로 Dataset의 변화가 필요하다.

첫째, Dataset을 재구축한다. 현재 Dataset은 근접 촬영이 대부분이다. 즉, 원거리 촬영에 대한 Dataset에 부족하기 때문에 작은 Object에 대한 성능이 떨어진다. 따라서 원거리 촬영에 대한 Dataset을 수집하여 Training Dataset에 포함시키는 과정이 필요하다.

둘째, Augmentation 기법을 이용한다. 이는 Deep Learning Vision 분야에서 이미지가 부족할 때 사용하는 방법 중에 하나이다. 이미지를 회전시키거나, 확대 및 축소, 좌우 및 상하 반전, 노이즈 추가 등의 영상처리 기법을 통해 기존 이미지와 비슷한 이미지들로 변형시켜 Dataset을 부풀리는 기법이라고 할 수 있다. 우리가 해당 기법 중에 '축소' 기법을 통해 문제를 해결해볼 수 있다. 작은 Object에 대한 이미지 수집이 어려울 경우, 현재 가지고 있는 이미지들을 축소시켜 마치 작은 Object처럼 만들어 주는 것이다. 이에 대한 예시는 아래와 같다.

before	after
--------	-------



그림23. Augmentation을 위한 이미지 축소 예시

<그림23>의 예시와 같이 small object 이미지를 임의로 늘려준 뒤 학습데이터에 포함시킴으로써, 모델이 small object에 대한 학습의 기회가 많아져 검출력을 높일 수 있다.

4.1.3. 근거리 촬영 및 안전모 착용 여부 탐지

먼 거리에 있어 작게 보이는 대상에 대한 탐지력이 상대적으로 떨어지기 때문에 현장의 작업자와 탐지 시스템의 카메라가 근거리에 있는 상황을 고려한다. 작업 현장으로 들어갈 수 있는 모든 출입구에 안전모 착용 여부 탐지 시스템을 설치하여 이 시스템을 지나지 않고는 들어가지 못하도록 관리한다. 그러나 출입구를 지나면 작업자가 시스템의 관리 영역을 벗어나게 되므로, 결과적으로 안전모를 쓴 상태를 유지하도록 만드는 방안에 대한 논의가 추가적으로 필요하다.

4.2. 현장에 도입되었을 때 발생할 수 있는 문제점

이 모델은 기존 현장의 인적 한계를 보완한 자동화 시스템으로 안전모를 미착용한 작업자를 스스로 탐지할 수 있다. 하지만 해당 시스템은 ‘탐지’만 가능할 뿐, 기존의 작업(안전)관리자처럼 안전모 미착용 작업자에게 주의나 경각심을 줄 수 없다는 한계가 있다. 따라서 실제 현장에서 해당 시스템으로 안전모 미착용 작업자를 ‘탐지’했다면 작업자에게 어떤 방식으로 실질적인 경고나 경각심을 줄 지 제시한다.

4.2.1. 경보음을 통한 경고

차량 앞좌석에 탑승한 사람이 안전벨트를 매지 않으면 앞좌석의 사람이 안전벨트를 맬 때까지 멈추지 않고 경보음이 울린다. 이처럼 작업 현장을 비추는 화면 안의 person 상자 수를 실시간으로 체크하여 관리 화면 안에 person 상자가 한 개라도 있다면 경보음을 계속해서 울려 현장의 작업자에게 경고한다. 안전벨트 경고음과 마찬가지로 person 상자의 수가 0이 될때까지 경고음을 울린다. 경고음은 작업 현장이라는 특성 상 일정 데시벨 이상으로 설정하여 작업자가 충분히 경각심을 가지도록 한다.

4.2.2. 벌점 시스템

안전모 착용 여부 감지 시스템에 벌점 시스템을 추가하여, 안전모 착용에 강제성을 부여한다. 안전모 앞, 뒷면에 QR 코드를 삽입하여, 안전모 미착용이 감지 되면, QR코드도 동시에 감지되도록 한다. QR 코드에 암호화 되어 등록된 작업자 이름, 연락처, QR코드를 생성한

일시를 근거로 하여, 작업자에게 패널티를 준다. 벌점으로 인한 패널티가 존재하기 때문에 오류를 최대한 줄이기 위해서 정확도가 더욱 향상된 모델이 필요하다. 또한 건설업의 경우, 일용직 작업자가 많기 때문에 정규직 작업자와 일용직 작업자의 벌점 규정을 다르게 설정해야 한다.

4.3 안전모만이 아닌 다른 보호구에 적용

안전보건규칙 제32조(보호구의 지급 등)에 따르면 사업주는 근로자에게 작업조건에 맞는 보호구를 지급하고 착용하도록 해야 할 의무가 있다. 안전모 뿐만 아니라 안전화(물체의 낙하, 충격, 물체에의 끼임, 감전의 위험이 있는 작업), 보안경(물체가 훌날릴 위험이 있는 위험이 있는 작업), 보안면(용접 시 불꽃이나 물체가 훌날릴 위험이 있는 작업) 등 작업자들은 작업 조건에 맞게 보호구를 착용해야 할 필요가 있다. 하나의 예로, 현대자동차 공장에서는 안전화 착용이 필수이다. 따라서 안전모 착용 감지 시스템이 여러 산업 현장 상황에 맞게 적재적소에 사용된다면 작업자의 재해, 사망률을 줄이는 데 도움이 될 것이다.

5. Conclusion

산업 재해를 줄이기 위한 가장 기본적인 사항은 작업자의 안전보호구 착용이다. 이를 감독자가 눈으로 직접 점검하는 것은 비효율적일 뿐만 아니라 공간적으로도 그 한계가 존재한다. 우리는 이러한 한계를 극복하기 위해 Deep Learning Technology로 문제를 해결할 수 있는지 그 가능성은 평가했다. 특히 작업자의 안전모 탐지는 실시간으로 수행해야 하기에 속도면에서 유리한 YOLO v3 모델을 Transfer Learning하였고, 그 결과 적은 실험으로도 0.64 mAP 성능을 달성할 수 있었다. 이를 통해 Deep Learning Technology의 적용 가능성을 입증했고 잠재력을 확인했다. 또한, 이에 그치지 않고 모델의 성능을 개선시키기 위한 기술과 현장에 모델을 적용시키기 위한 방안, 또한 기술의 범용적 측면을 제시하며 프로젝트의 확장 가능성을 제시했다.

이미 우리의 삶 주변에는 수많은 Deep Learning Technology가 사용되고 있다. 앞으로 우리 사회는 어떤 문제에 어떤 Deep Learning Technology를 적용할 수 있을지를 파악하고 가능성을 평가하는 안목을 기를 수 있어야 할 것이다.

6. References

- [1] [내외전기통신저널 기사](#)
- [2] [한국산업보건공단 데이터](#)
- [3] [고용노동부 - 2018년 산업재해 현황분석 책자](#)
- [4] [산업안전보건법](#)
- [5] [딥러닝을 이용한 영상 기반의 콘크리트 구조물 박락 탐지](#)
- [6] [딥러닝\(Deep Learning\) 기술을 활용한 SOC구조물 성능평가 기술 개발](#)
- [7] [날로 스마트해지는 포스코 제철소, 스마트 CCTV로 또 한번 진화한다](#)

- [8] [Machine Learning in Computer Vision](#)
- [9] [YOLOv3: An Incremental Improvement](#)
- [10] [What's new in YOLO v3?](#)
- [11] [How to measure performance of object detection](#)
- [12] [Hardhat Dataset Github URL](#)
- [13] [AlexeyAB/darknet](#)
- [14] [\[다큐3일\] '세종시 건설현장에서의 3일'](#)
- [15] [A survey of loss functions for semantic segmentation](#)