YOLO를 활용한 마스크 미착용 탐지

※ 딥러닝의 활용

통계학과 김영석
통계학과 이용헌

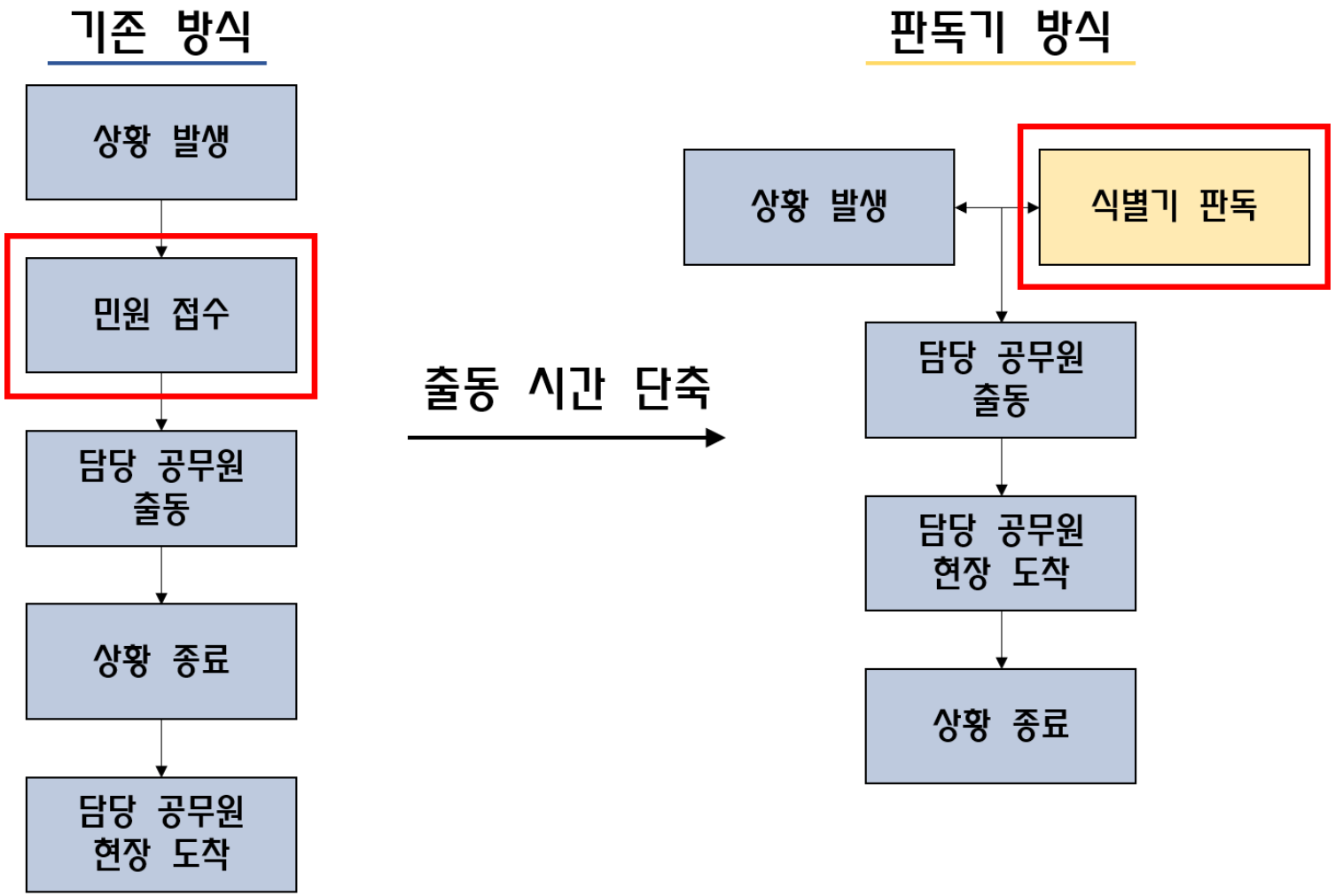
Contents

1. Intro

연구배경

- 마스크 착용으로 코로나 바이러스 비말 감염 확률 감소
- 마스크 착용 의무화 조치 후 민원 급증
- 관련 공무원이 현장을 나가더라도 이미 신고 대상자가 자리를 떠나 상황 종료

상황발생조치


기존 방식
상황 발생
민원 접수
담당 공무원 출동
상황 종료
담당 공무원 현장 도착
출동 시간 단축
판독기 방식
상황 발생
식별기 판독
담당 공무원 출동
담당 공무원 현장 도착
상황 종료

YOLO

YOLO

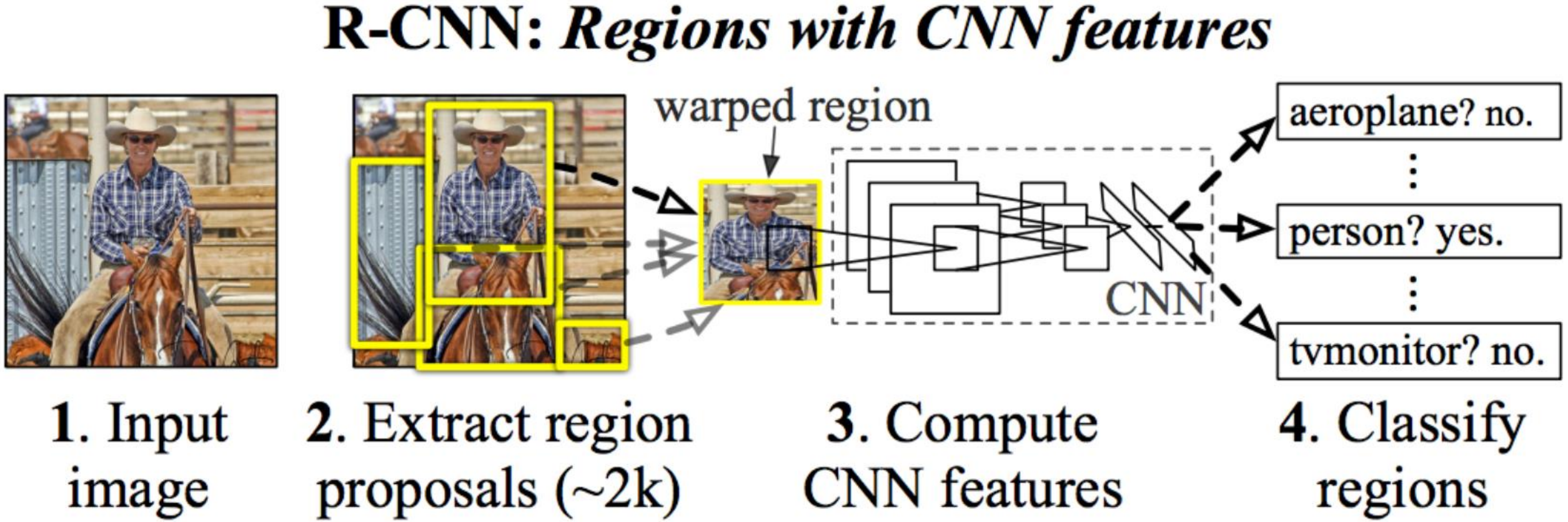
You Only Look Once

이미지를 한 번 보는 것만으로
Object의 종류와 위치를 추측하는 딥러닝 기반의 물체 인식 알고리즘

YOLO

이미지를 한 번만 본다?

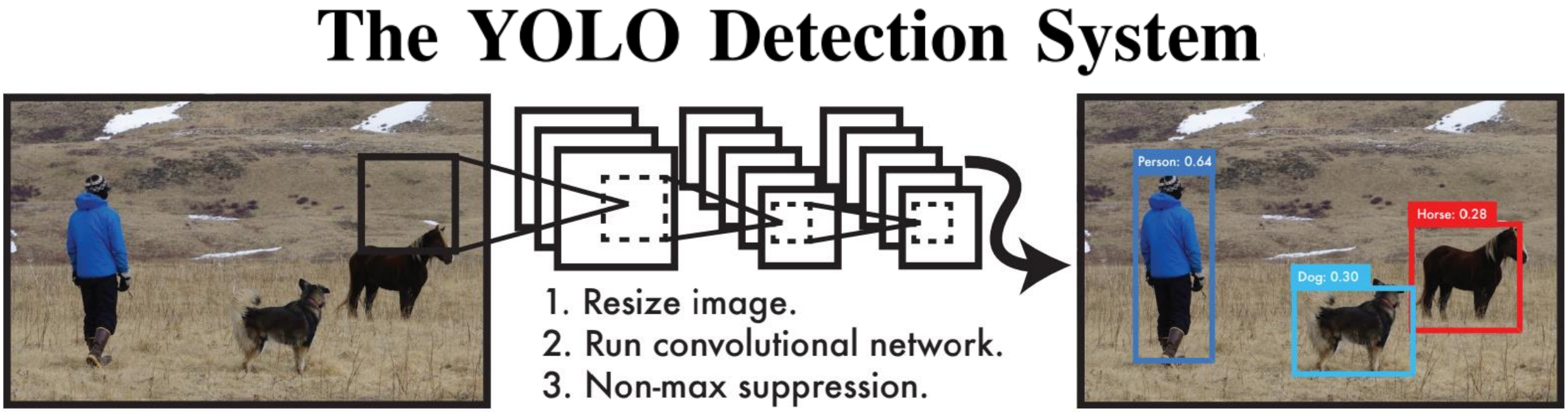
YOLO vs R-CNN



Selective Algorithm을 이용
이미지를 최대 2000개의 리전 생성
서브 이미지 추출

약 2천개의 이미지가
CNN을 통과

YOLO vs R-CNN



이미지를 CNN에 단 한 번 통과

바운딩 박스와 Class 분류 확률 출력

YOLO vs R-CNN

Real-time Object Detection: 실시간 객체 탐지

Fast R-CNN : 0.5 FPS

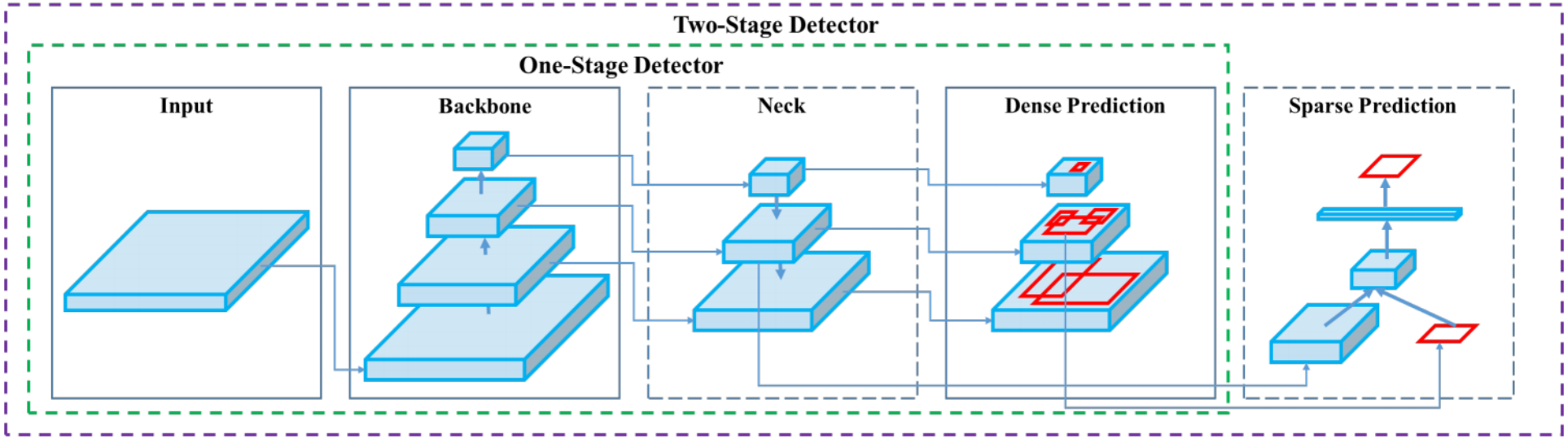
Faster R-CNN : 5 FPS

YOLOv1 : 45 FPS

YOLOv5 : 140 FPS

YOLOv4 논문 리뷰

Architecture



Input: { Image, Patches, Image Pyramid, … }

Backbone: { VGG16 [68], ResNet-50 [26], ResNeXt-101 [86], Darknet53 [63], … }

Neck: { FPN [44], PANet [49], Bi-FPN [77], … }

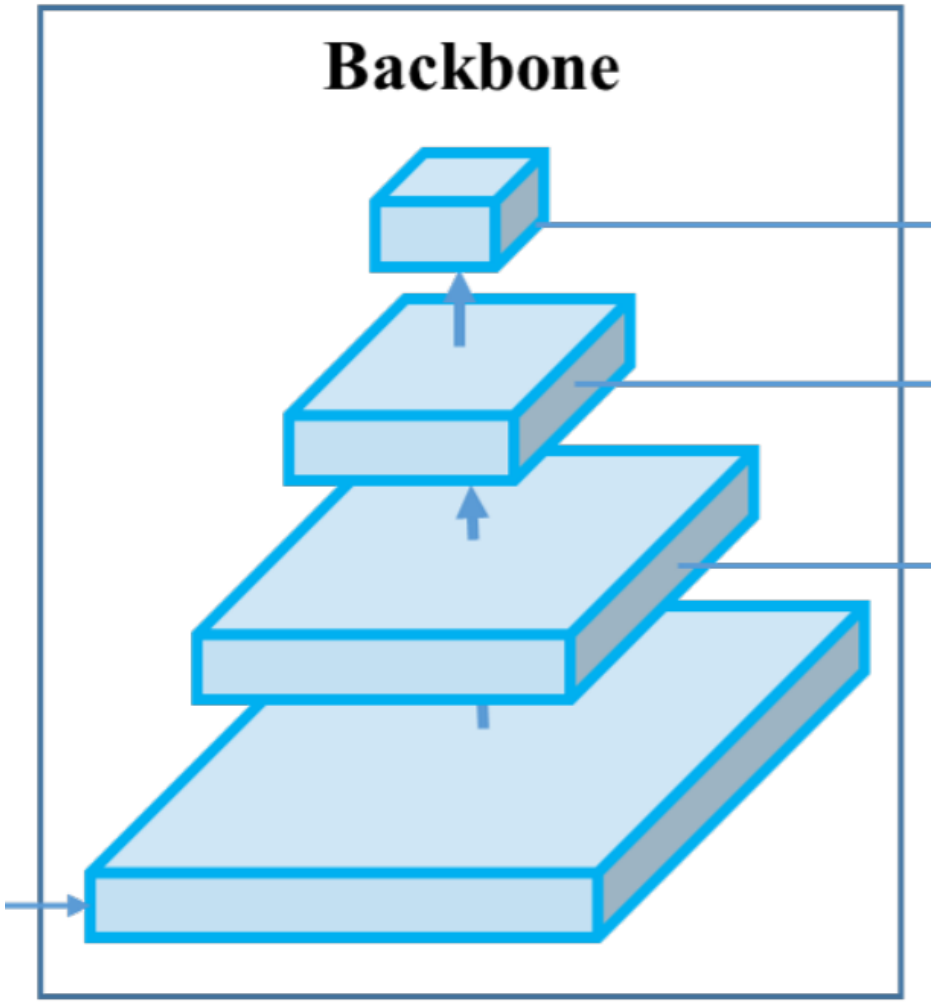
Head:

Dense Prediction: { RPN [64], YOLO [61, 62, 63], SSD [50], RetinaNet [45], FCOS [78], … }

Sparse Prediction: { Faster R-CNN [64], R-FCN [9], … }

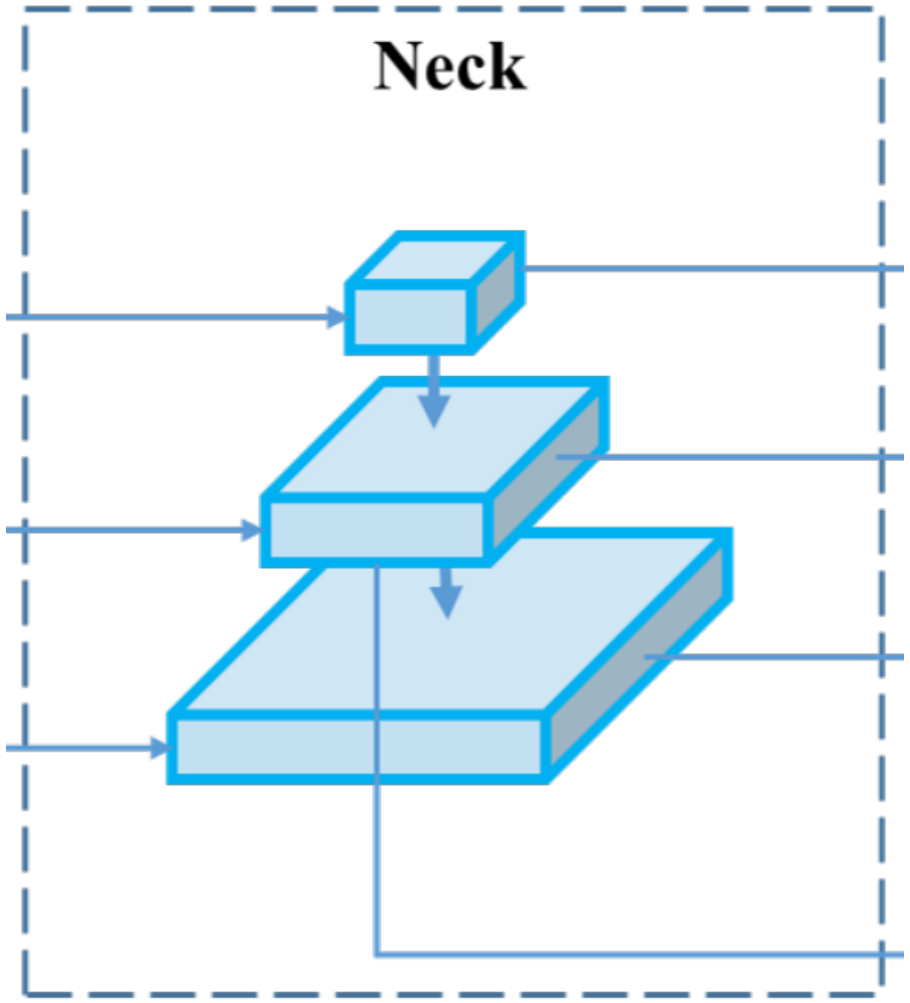
Figure 2: Object detector.

YOLOv4 논문 리뷰



Backbone은
input image를 feature map으로 변형

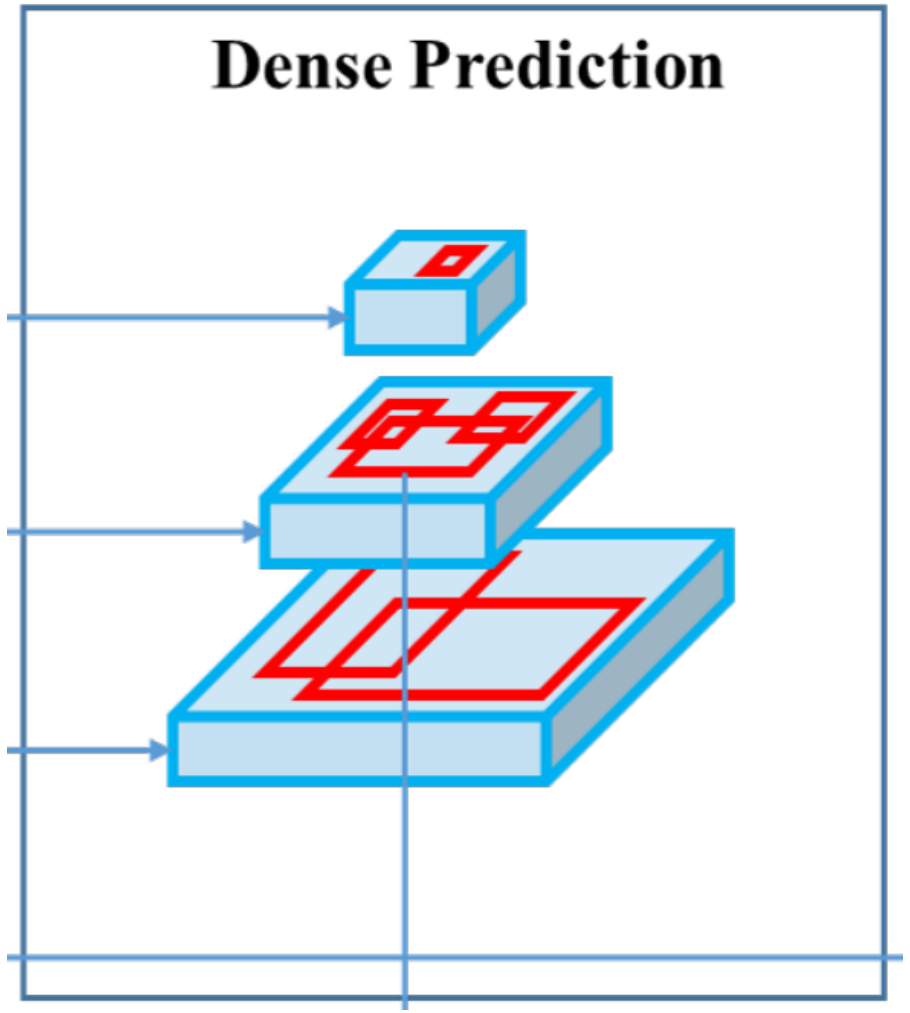
YOLOv4 논문 리뷰



Neck은
Backbone과 Head를 연결하는 부분

Feature map을
1. Refinement(정제)
2. Reconfiguration(재구성)

YOLOv4 논문 리뷰



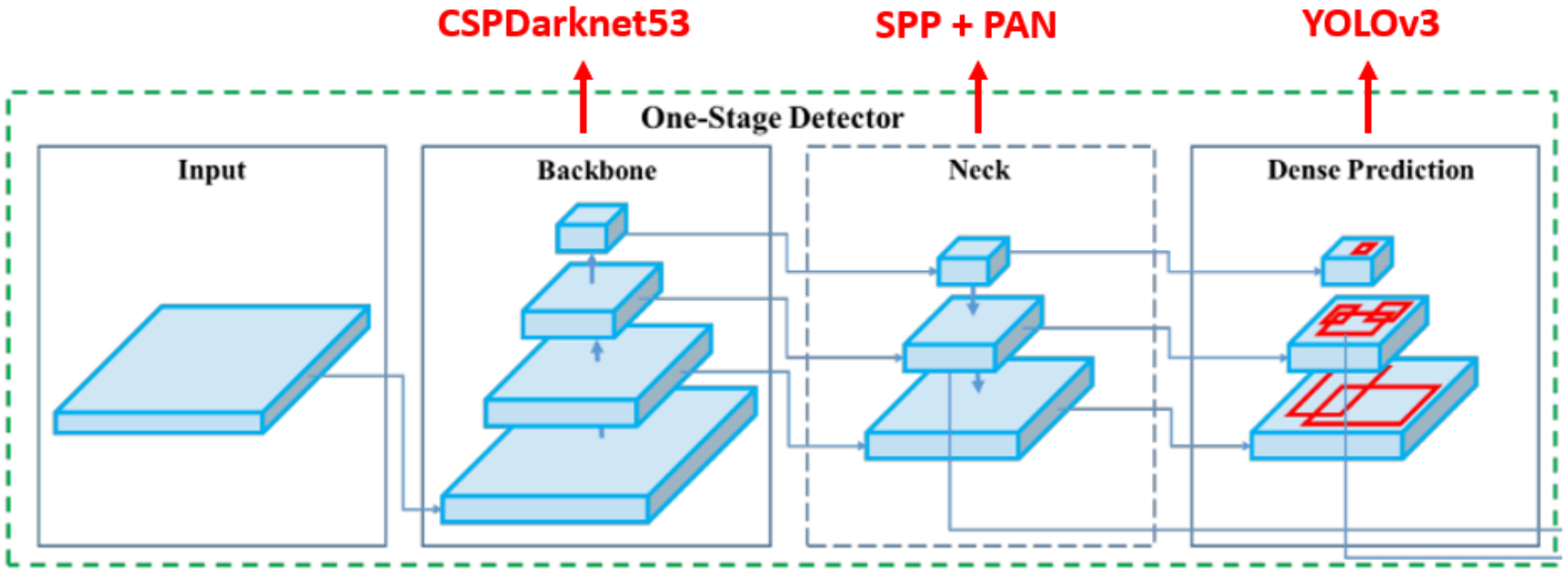
Head는
Backbone에서 추출한 feature map의 location 작업
Predict classes와 bounding boxes 작업

Head 구분
- Dense Prediction(One stage):
Predict + Bounding boxes

- Sparse Prediction(Two stage):
Predict | Bounding boxes

YOLOv4 논문 리뷰

Architecture



YOLOv4 = YOLOv3 + CSPDarknet53 + SPP + PAN + BoF + BoS

↓

Path Aggregation Network

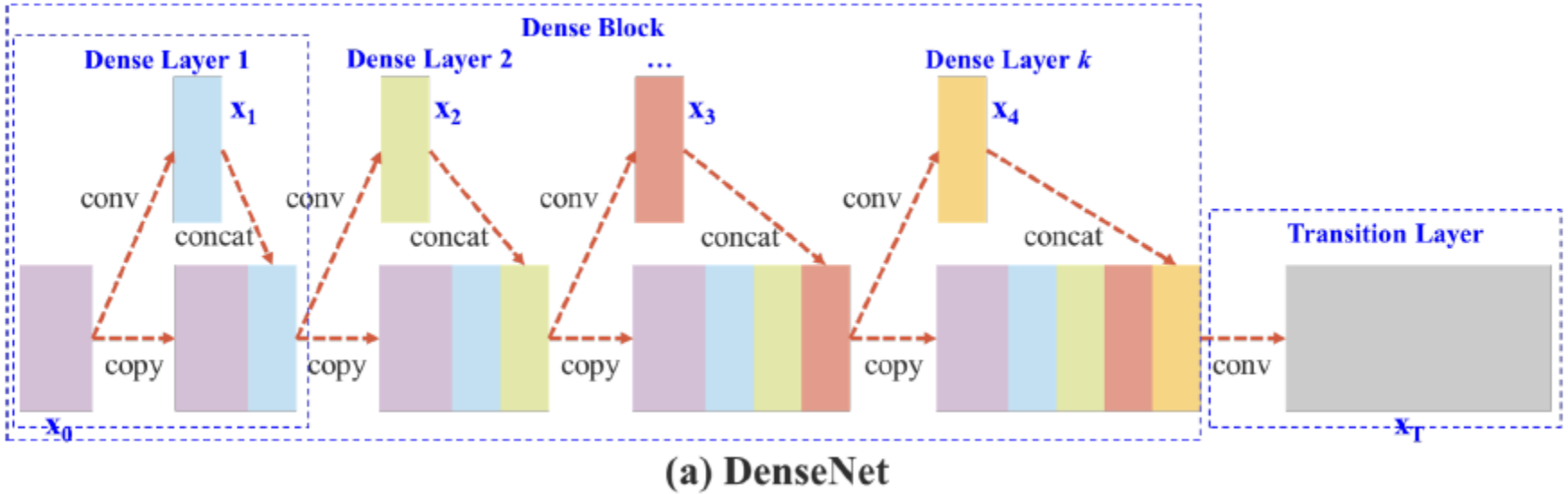
YOLOv4 논문 리뷰 Bag Of Freebies(BOF) & Bag of Specials(BOS)

BOF와 BOS는 특정 Block의 하나이기보다는 Backbone과 Detector 등에 포함된다.

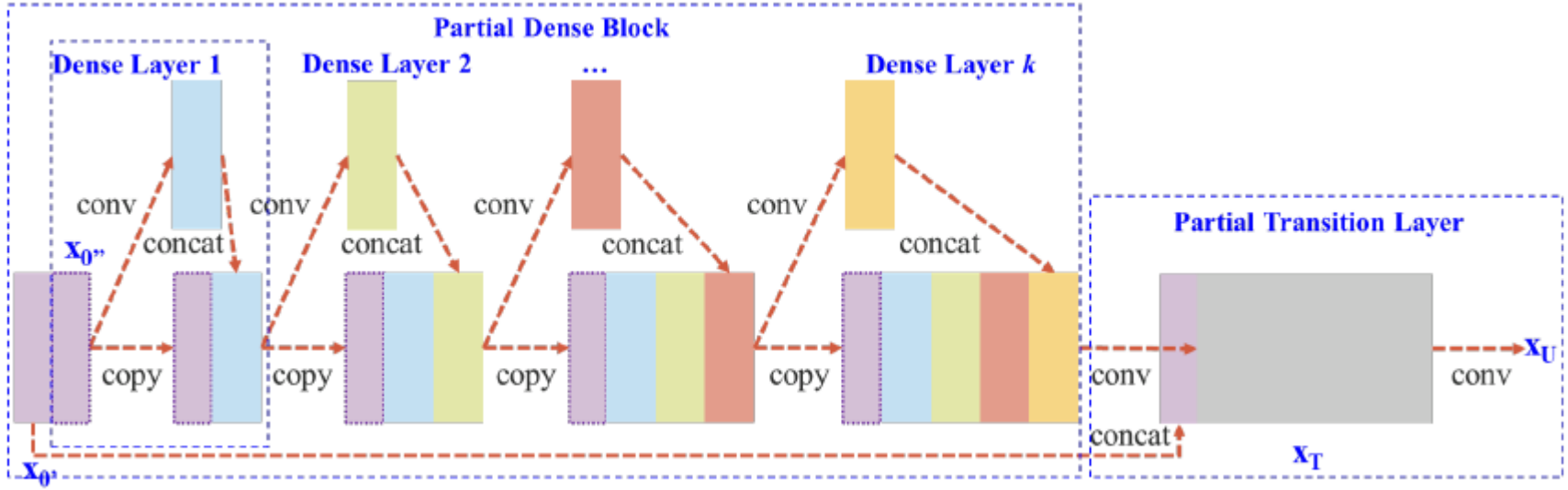
- BOF는 Inference cost의 변화 없이 성능 향상(better accuracy)을 꾀할 수 있는 딥러닝 기법이다.
- 대표적으로 데이터 증강(CutMix, Mosaic 등)과 BBox(Bounding Box) Regression의 손실 함수(IOU loss, CIOU loss 등)이 있다.

- BOS는 BOF의 반대로 Inference cost가 조금 상승하지만 성능 향상이 되는 딥러닝 기법이다.
- 대표적으로 enhance receptive filed(SPP, ASPP, RFB), feature integration(skip-connection, hyper-column, Bi-FPN) 그리고 최적의 activation function(P-ReLU, ReLU, Mish 등)이 있다.

YOLOv4 논문 리뷰 Backbone(CSP DenseNet)



(a) DenseNet



(b) Cross Stage Partial DenseNet

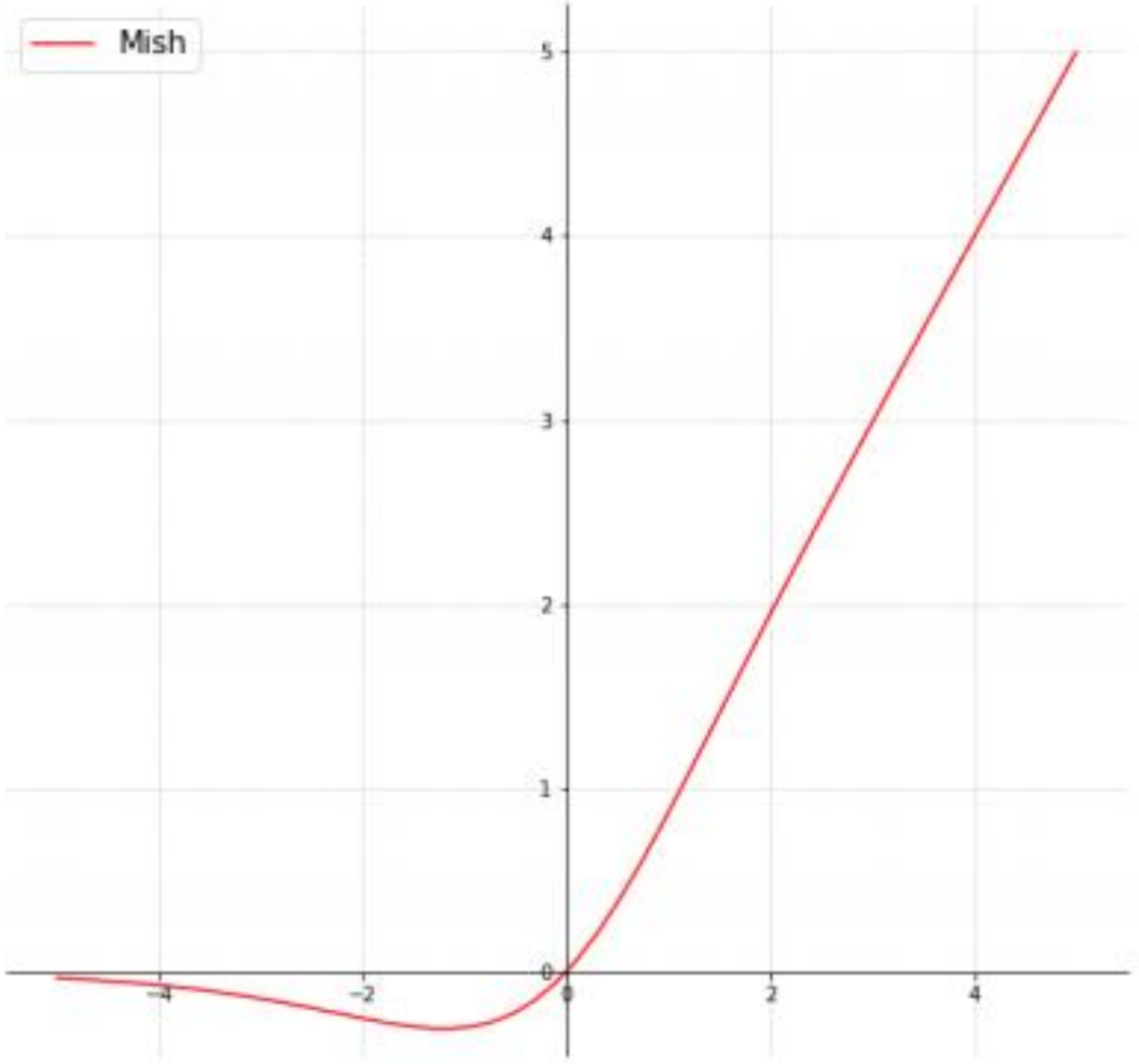
DenseNet과 CSP의 차이점

input X0: X0`, X0`` 분리
X0``: 기존 DenseNet
X0`: Transition Layer

Weight 갱신 시
다른 Layer의 기울기 정보가
중복되는 것을 방지

Input size 감소:
Inference Cost 감소
Memory Cost 감소

YOLOv4 논문 리뷰 Backbone(Mish Activation)



Mish Activation

$$f(x) = x \times \tanh(softplus(x))$$
$$softplus(x) = \ln(1 + e^{x})$$

- 일반적으로 0에 가까운 기울기로 인해 훈련 속도가 급격히 느려지는 포화 문제 방지
- ReLU보다 Mish의 Gradient가 좀 더 Smooth함

- 강한 Regularization 효과로 Overfitting 문제 방지

YOLOv4 논문 리뷰 Neck(Spatial Pyramid Polling(SPP)-Net)

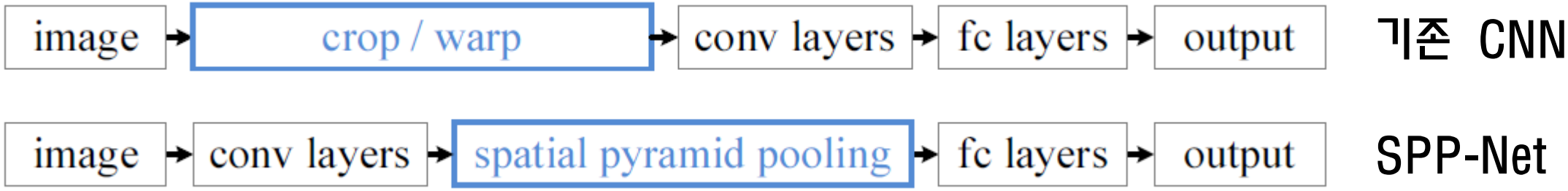
- 기존 CNN

Image 영역 수천개를 Crop/Warp로 왜곡시켜 fixed sized된 image를 CNN에 넣음
- 속도가 매우 느림

- SPP-Net

한번만 CNN을 수행 후 SPP-Net에서 fixed Image 출력
- 속도가 매우 빠름

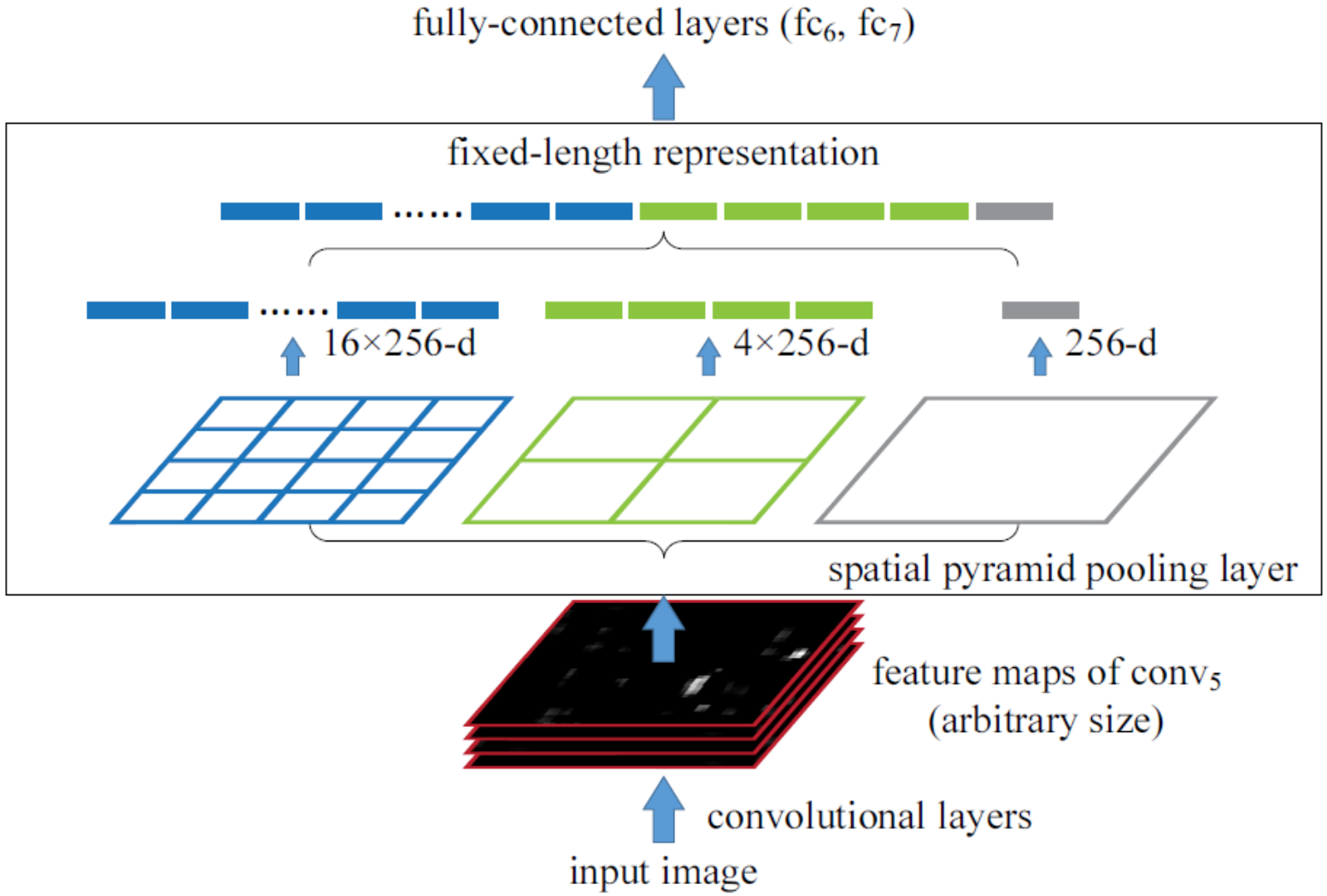
YOLOv4 논문 리뷰 Neck(Spatial Pyramid Polling(SPP)-Net)



Fully-Connected(FC) layers는 fixed-size/length를 가진 input이 필요

- 기존 CNN은 fc layers의 직전 conv layers에 fixed된 input이 필요해 crop/warp를 시행 후 conv layers의 input으로 사용
- SPP는 fc layers 직전 Spatial Pyramid Pooling이 input size에 관계없이 fixed된 output을 생성하기 때문에 conv layers가 input size로부터 자유롭다.

YOLOv4 논문 리뷰 Neck(Spatial Pyramid Polling(SPP)-Net)



YOLOv4 논문 리뷰 Neck(Path Aggregation Network(PANet))

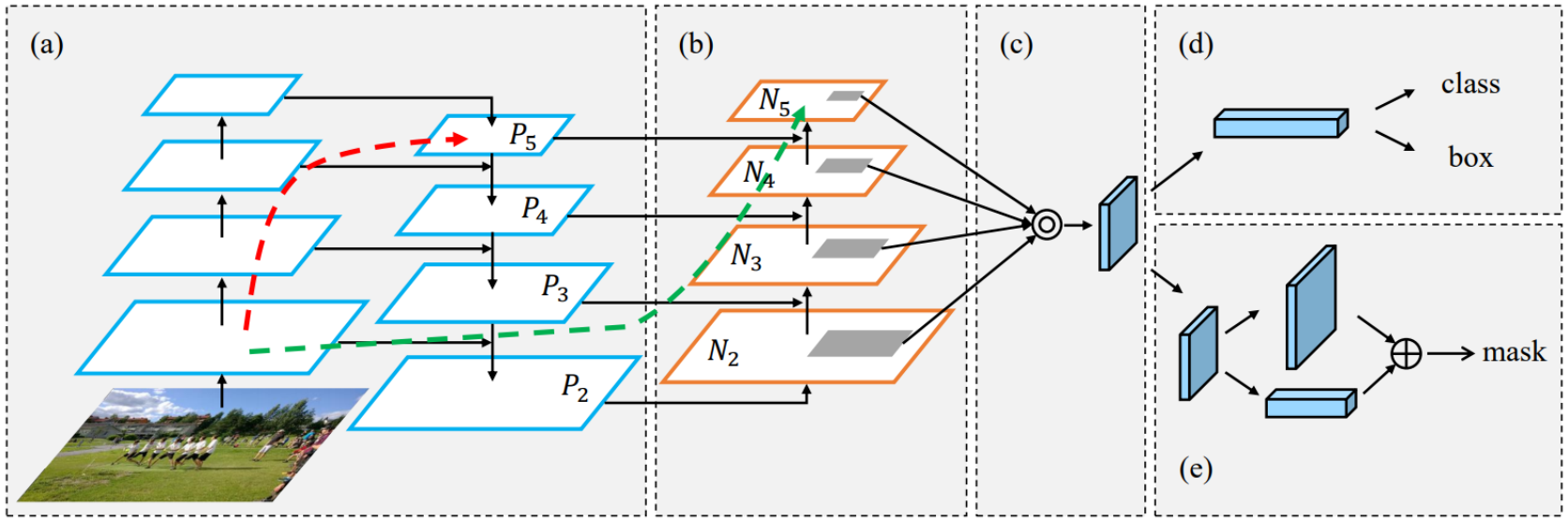


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature maps in (a) and (b) for brevity.

PANet은 기존 FPN Backbone의 한계점인 low-level to topmost features에 많은 과정(붉은 점선)을 거쳐 Propagate가 원활하게 이뤄지지 못하는 단점을 새로운 피라미드 구조를 만들어 Shortcut(녹색 점선) 하나와 몇 개의 과정만 거치면 된다.

이를 통해 원활한 Information Flow가 가능하다.

프로젝트 목표

- YOLO를 활용한 마스크 미착용 감지 모델 생성
- 실제 착용 사진 및 미착용 사진 촬영 후 감지 능력 확인
- 비디오에서 감지할 수 있는 모델 생성
- CCTV(실시간 비디오)에서 마스크 미착용 실시간 감지

2. Design

개발환경

OS – Ubuntu 18.04

Python - 3.7

OpenCV – 4.4.2

Yolo – Yolov4

GPU – GTX1070TI x 2

CUDA – 10.1

CUDNN – 7.6.4

DATASET

선행 연구 데이터

총 887장

명확한 데이터

구글 크롤링 데이터

총 756장

명확하지 못한 데이터

어떤 데이터로 학습하면 더 좋은 성능을 보일까?

DATASET

TRAIN

선행 연구 데이터

767장

VS

웹 크롤링 데이터

656장

VS

(선행+크롤링)통합 데이터

328장 + 385장

TEST

선행 연구 데이터

120장

+

웹 크롤링 데이터

100장

총 220장

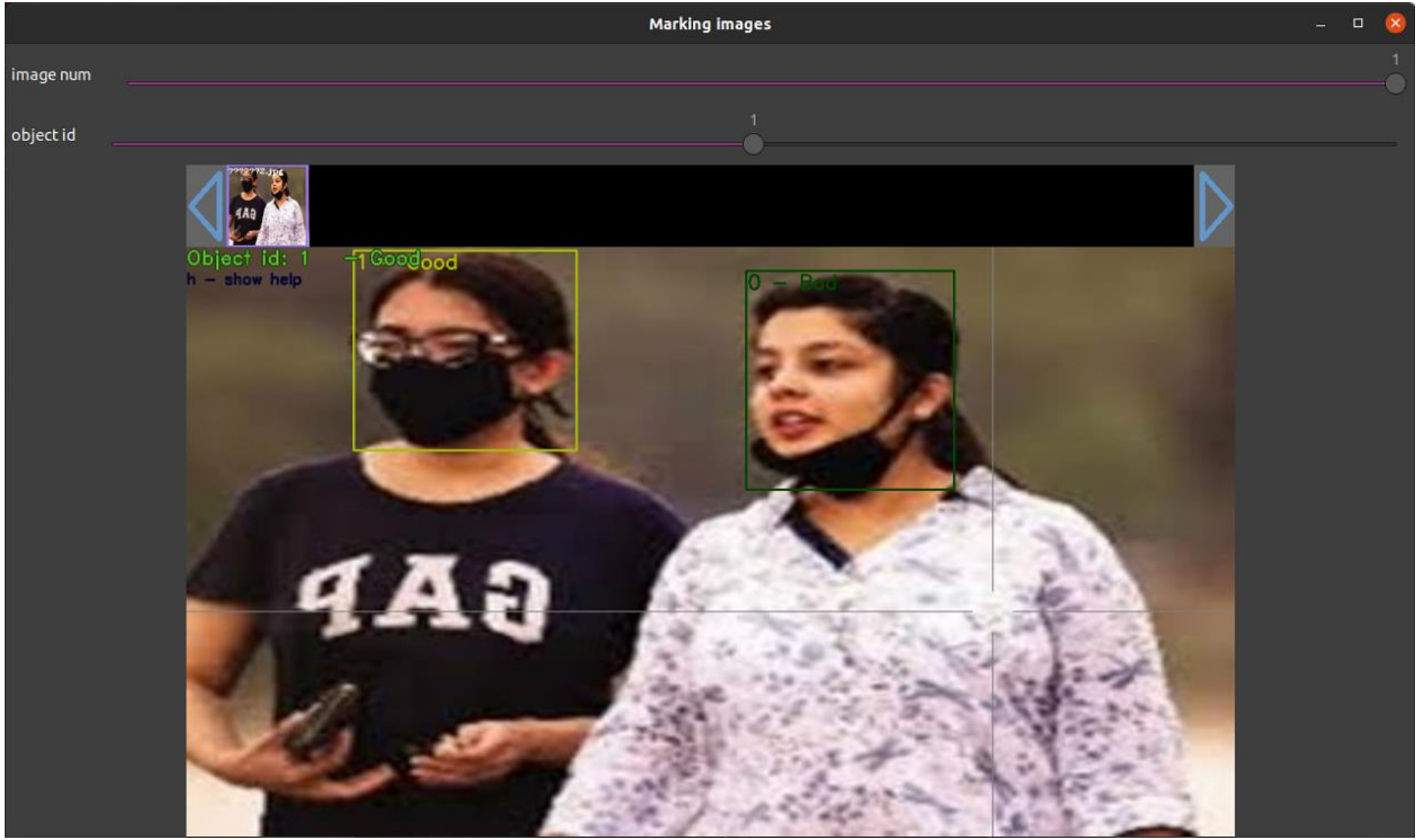
VRAM ISSUE

- GTX1070TI – VRAM 8GB
- 이미지 데이터 파일 별 용량 차이로 batch_size 한계
- Input image resize 필요

IMAGE RESIZE & PADDING

- 원본 비율 고정, 608x608로 나머지 공간 black padding

LABELING – YOLO_MARK

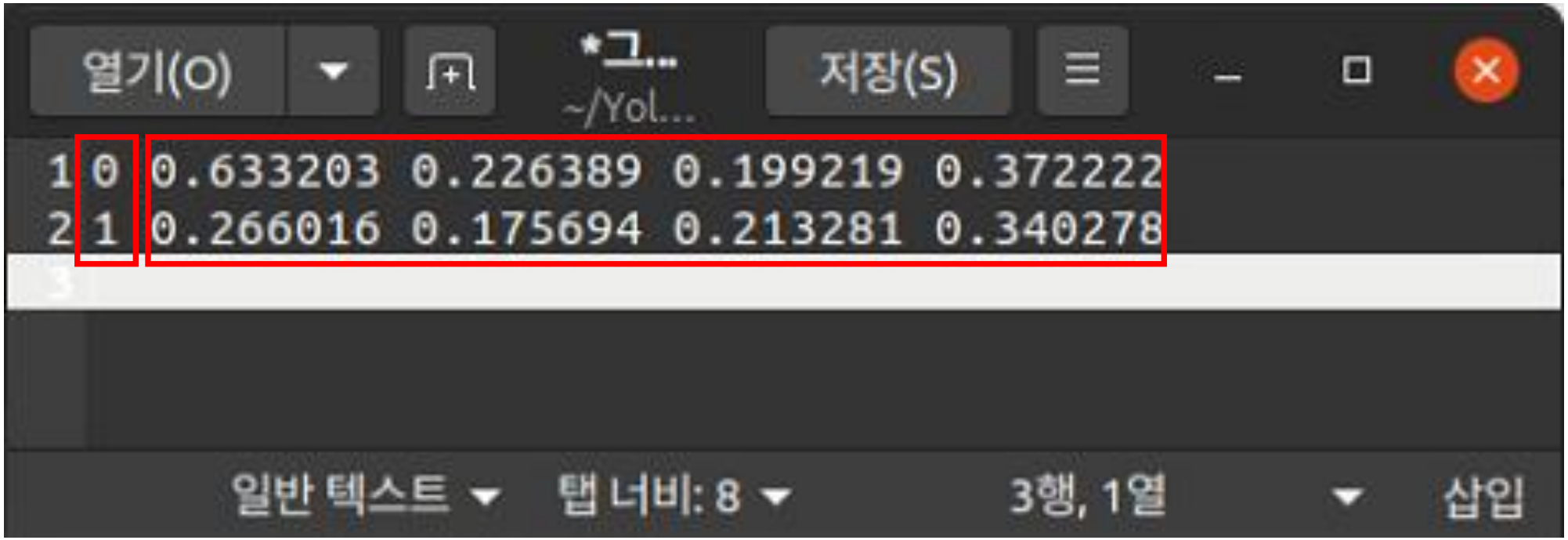


LABELING – YOLO_MARK

Class, X, Y, Width, Height

X, Y : Boxing한 사각형의 중심 좌표

Width, Height : 사각형의 가로, 세로 길이



LABELING 후 Object 수

	Train Good	Train Bad	Train Merge	Test
Number of Images	767	656	713	220
Mask	2,923	2,549	2,547	947
No Mask	894	655	823	220
Total	3,817	3,204	3,370	1,167

DARKNET

$ git clone github.com/AlexeyAB/darknet

- Makefile option 수정

GPU=1	GPU 사용
CUDNN=1	신경망 성능 향상
CUDNN_HALF=1	
OPENCV=1	OPENCV 사용
AVX=0	AVX INTEL CPU 성능 개선
OPENMP=0	다중코어 사용
LIBSO=1	.so 파일 생성
ZED_CAMERA=0	
ZED_CAMERA_v2_8=0	

DARKNET

- Cfg 파일 수정 (yolo 설정 파일)

- [net]
- batch=16
- subdivisions=16
- width=608
- height=608
- channels=3
- momentum=0.949
- decay=0.0005
- angle=0
- saturation = 1.5
- exposure = 1.5
- hue=.1
- learning_rate=0.00261
- burn_in=1000
- max_batches=4000
- policy=steps
- steps=3200,3600
- scales=.1,.1

- 수정된 값 설명

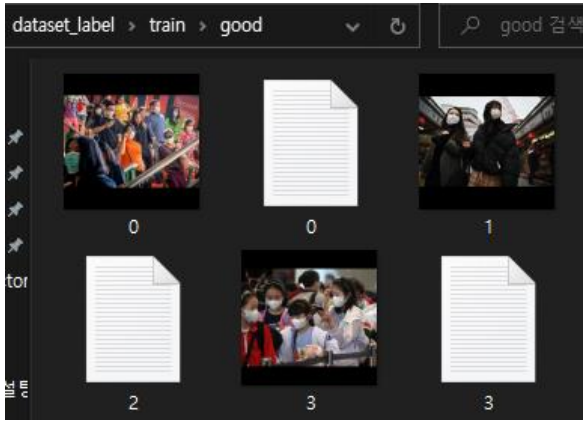
- batch: 배치사이즈
- width 및 height: 기본값은 416이지만 608로 변경하여 학습 할 경우 해상도 향상으로 정확도가 좋아질 수 있으나 Out Of Memory ERROR가 발생할 수 있음.
- max_batches: 언제까지 iteration을 돌건지 설정하는 값으로 본인 데이터의 클래스 개수 * 2000을 제시한다.

DARKNET

- 데이터 셋 경로 파일 수정

```
├── custom_data/
    ├── dataset_label
    ├── obj.data
    ├── obj.names
    ├── test.txt
    └── train.txt
```

- dataset_label: 모든 데이터 셋과 각 사진들의 label에 대한 txt파일을 저장
- obj.data: 전체적인 파일의 경로들을 저장
- obj.names: classes name을 저장(no mask / mask)
- test.txt: test에서 활용할 데이터 셋의 경로를 저장
- train.txt train에서 활용할 데이터 셋의 경로를 저장



dataset_label

obj.data

```
classes= 2
train  = /home/fadi/darknet/custom_data/train_bad.txt
valid  = /home/fadi/darknet/custom_data/test.txt
names = /home/fadi/darknet/custom_data/obj.names
backup = backup/train_bad_weights
```

test/train.txt

```
/home/fadi/darknet/custom_data/dataset_label/train/good/0.jpg
/home/fadi/darknet/custom_data/dataset_label/train/good/1.jpeg
/home/fadi/darknet/custom_data/dataset_label/train/good/10.jpeg
/home/fadi/darknet/custom_data/dataset_label/train/good/100.jpg
/home/fadi/darknet/custom_data/dataset_label/train/good/101.jpeg
```

MODEL LEARNING

yolov4 weights file download from **github.com/AlexeyAB/darknet**

- Transfer Learning(전이학습)

-> 기존 weights를 초기 값으로 신경망 생성 후 데이터 셋을 통한 학습

-> 학습에 필요한 데이터가 충분하지 않기 때문에 전이학습을 통한 문제 해결

- 학습의 경우 python 환경에서 기존 yolov4의 weight값 변환을 통해 tensorflow에서 구현하고자 했으나 알 수 없는 문제로 인해 predict 결과가 모두 no mask로 나와 정상적인 학습이 됐다고 보기 어렵다.
- (C를 기반으로 학습된 yolov4의 weight를 python으로 변환하는 과정에서 문제가 발생한 게 아닐까 하는 생각)
- 따라서 darknet에서 C언어를 기반으로 제작된 train파일을 terminal에서 실행하여 학습 후 predict를 진행했다.

MODEL Training

- 연구목적

- cctv 같은 작고 많은 사람이 나타나 식별이 어려운 영상에서 탐지하는 알고리즘

- Model1: 선행연구에서 사용한 비교적 detection이 쉬운 데이터(Train Good)로 학습

- Model2: 구글 크롤링한 detection이 쉽지 않은 데이터(Train Bad)로 학습

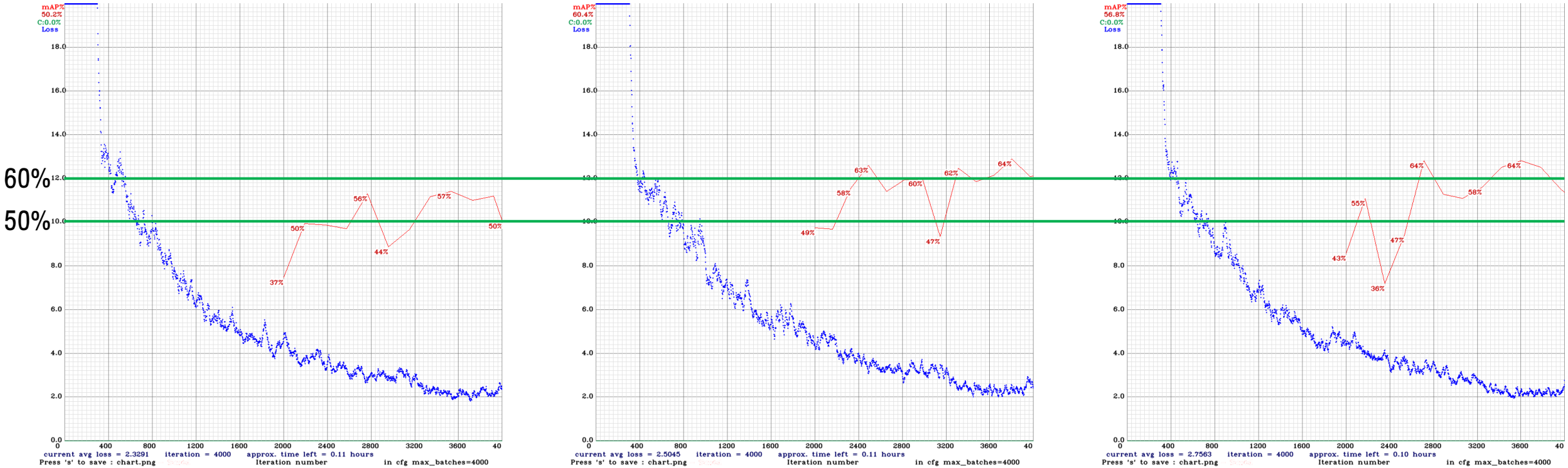
- Model3: 선행연구 데이터와 크롤링 데이터를 절반씩 섞은 데이터(Train Merge)로 학습

MODEL 비교

- mAP50(mean Average Precision; IOU Threshold = 0.50)

-> TP / (TP + FP) = AP를 여러 obs에 대해 mean을 구한 값

- Yolo v4의 MS COCO 데이터 mAP = 43.5%



Train_Good: best mAP = 57%

Train_Bad: best mAP = 64%

Train_Merge: best mAP = 64%

MODEL 비교

- No Mask AP & Mask AP

-> No Mask의 TP / (TP + FP)와 Mask의 TP / (TP + FP)

No Mask	Mask
• Train Good: 46.31%	• Train Good: 67.55%
• Train Bad: 51.68%	• Train Bad: 77.05%
• Train Merge: 54.54%	• Train Merge: 73.54%

- No Mask에서는 Merge 데이터, Mask에서는 Bad 데이터의 AP가 각각 높았다.
- 하지만 No Mask의 Detection이 주 목적이라는 부분이 고려되어야 한다.

MODEL 비교

- FPS(Frame per Second)

-> 초당 몇 장의 이미지 처리가 가능한지를 나타내는 지표로 모델의 예측 속도를 평가할 수 있다.

- 중간 해상도 영상과 저해상도 영상에 대해 Predict한 FPS 결과의 평균을 계산

- Train Good: 23.15 (FPS)

- Train Bad: 22.95 (FPS)

- Train Merge: 23 (FPS)

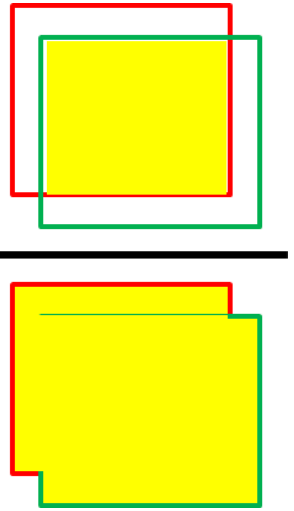
- 모델에 따른 차이가 거의 없다.

MODEL 비교

- Average IOU

-> 예측된 Bounding Box와 True Bounding Box의 중첩된 면적을 합한 면적으로 나눠준 값이다.

$$IoU = \frac{area(B_{gt} \cap B_p)}{area(B_{gt} \cup B_p)} =$$



- Train Good: 48.42%
- Train Bad: 51.86%
- Train Merge: 51.83%

- IOU는 Bad 데이터가 가장 높으나, Bad와 Merge가 거의 비슷하다.

MODEL Selection

- Evaluate Table

Value	Train Good	Train Bad	Train Merge
No_Mask_AP	46.31%	51.68%	54.54%
Mask_AP	67.55%	77.05%	73.54%
mAP50	56.93%	64.36%	64.04%
FPS	23.15	22.95	23
Average IOU	48.42%	51.86%	51.83%

- No_mask_AP와 Mask_AP를 제외한 나머지 값들은 Bad와 Merge가 거의 일치
- 따라서 연구 목적인 No_Mask_AP가 가장 높은 Merge 데이터로 학습한 모델 선택

3. Result

테스트 데이터에서 검출


no mask: 0.99
no mask: 0.91
no mask: 0.98
no mask: 0.98
mask: 0.97
no mask: 0.94
mask: 0.99
no mask: 0.97

동영상에서 검출

프로젝트 목표 달성

✓ YOLO를 활용한 마스크 미착용 감지 모델 생성

✓ 실제 착용 사진 및 미착용 사진 촬영 후 감지 능력 확인

✓ 비디오에서 감지할 수 있는 모델 생성

- CCTV(실시간 비디오)에서 마스크 미착용 실시간 감지

보완하고 싶은 점

- tensorflow를 활용한 학습을 시도하고 싶다.
- 앱과 연동된 프로그램 개발을 시도하고 싶다.
- 좀 더 여유 있는 VRAM 확보가 가능할 경우 이미지 resize 없이도 충분히 빠른 속도로 학습이 가능할 것으로 예상된다.

Github Upload

- https://github.com/kys159/Yolo

Q & A