

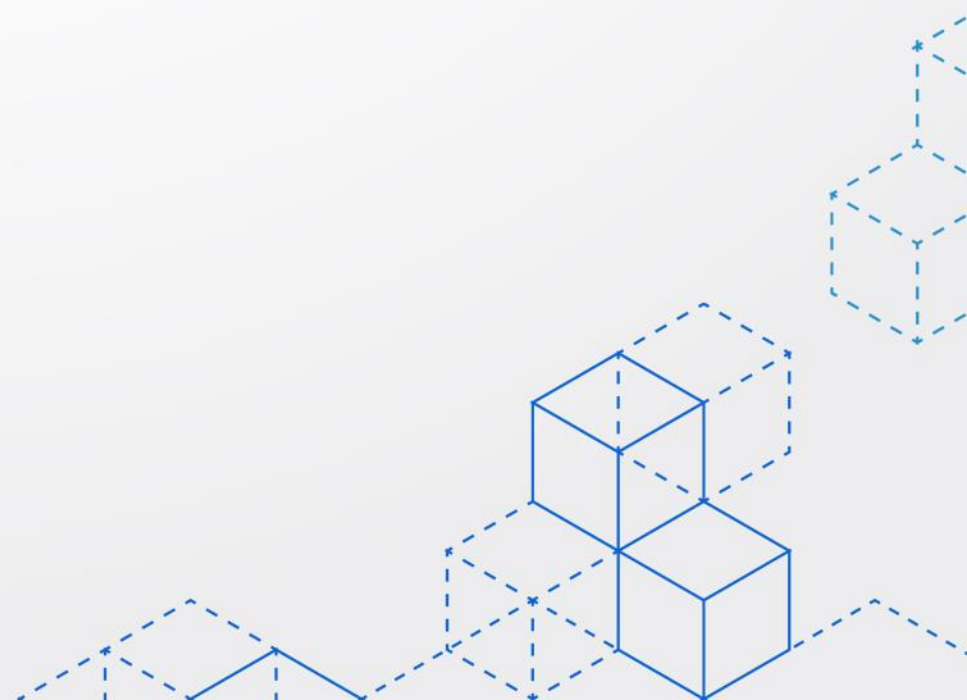


YOLO 를 제외한 1-Stage Detector

산업인공지능학과
2021254013 유대건
2021254001 이용규



- 목 차 -

- 개요
 - 1-Stage Detector, 2-Stage Detector
 - SSD, RetinaNet, FCOS
 - Performance
 - Q & A
- 

01. 개요

❑ Classification

- Single Object 에 대해 Object 의 클래스를 분류하는 문제.

❑ Classification + Localization

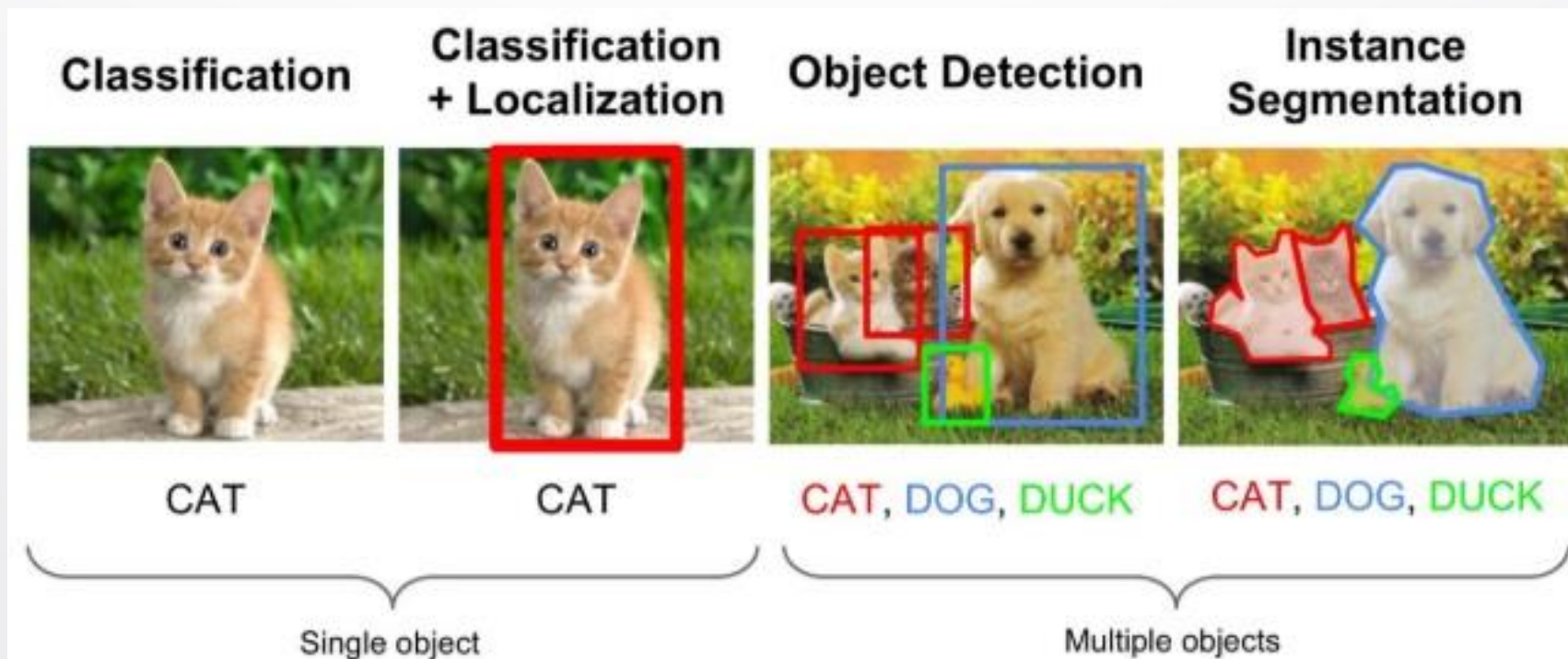
- Single Object 에 대해서 Object 의 위치를 Bounding Box 로 찾고 클래스를 분류하는 문제.

❑ Object Detection

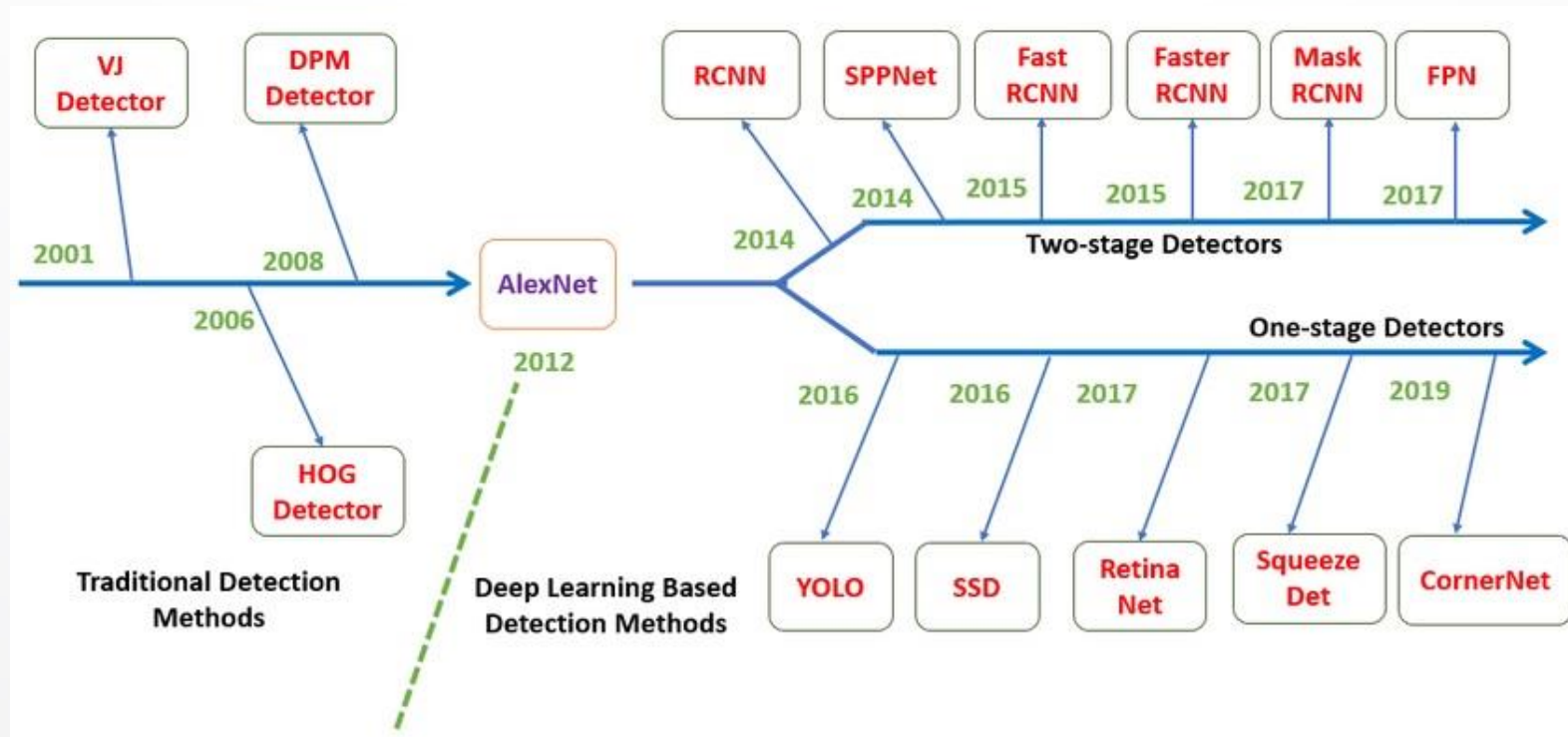
- Multiple Objects 에서 각각의 Object 에 대해 Classification + Localization 을 수행하는 것.

❑ Instance Segmentation

- Object Detection 과 유사하지만 Bounding Box 가 아닌 실제 Edge 로 Object 를 검출하는 것.



02. 2-Stage Detector, 1-Stage Detector



❑ 2-Stage Detector

- RCNN, Fast RCNN, Faster RCNN

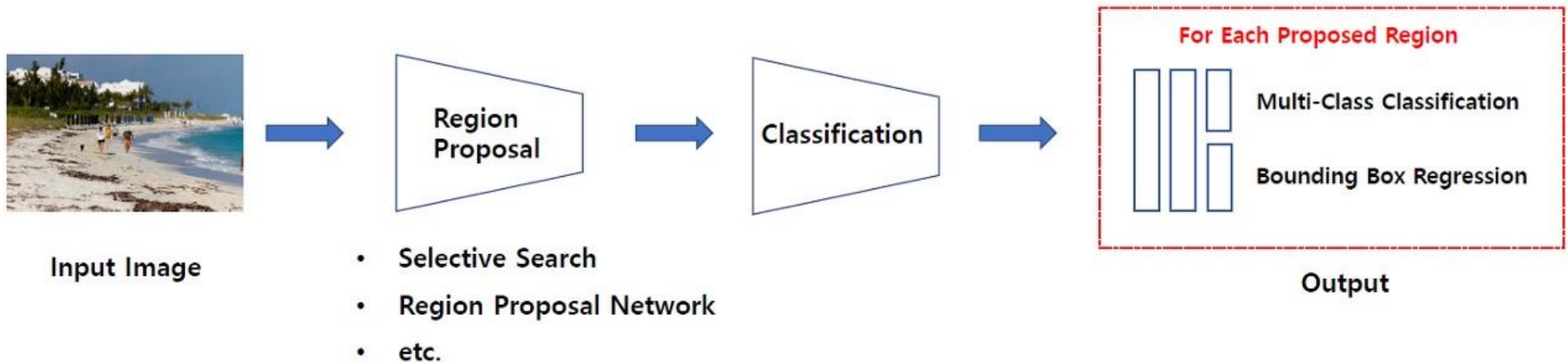
❑ 1-Stage Detector

- YOLO, SSD, Focal Loss, RetinaNet, RefineDet

02. 2-Stage Detector, 1-Stage Detector

❑ 2-Stage Detector

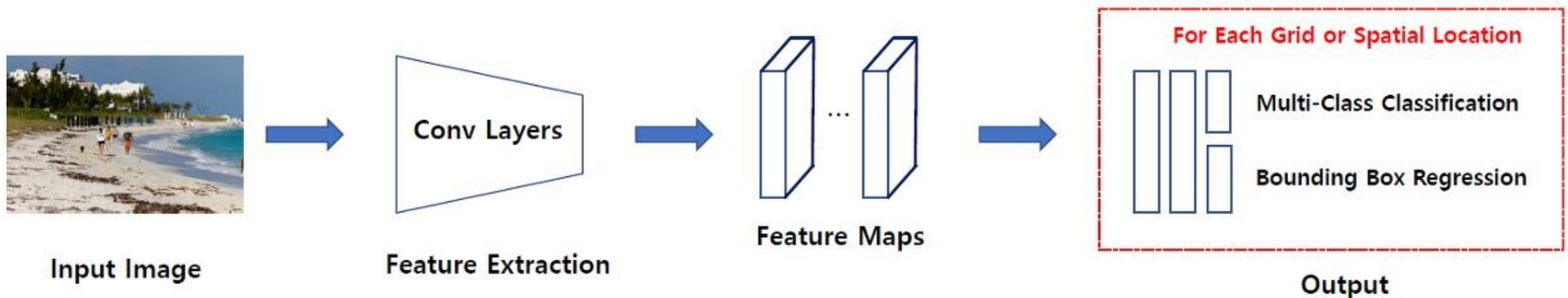
- Regional Proposal(위치정보) 과 Classification(분류) 을 순차적으로 적용
- Regional Proposal : 물체가 있을 만한 영역을 빠르게 찾아내는 알고리즘
Selective search, Edge boxes
- 두 문제를 순차적으로 행하는 방법.
- 비교적 느리지만 정확도 높음.



02. 2-Stage Detector, 1-Stage Detector

❑ 1-Stage Detector

- Localization 문제와 Classification 문제를 동시에 행하는 방법
- 비교적 빠르지만 정확도 낮음
- 전체 이미지에 대해 특징 추출, 객체 검출이 이루어짐.
- 영역을 추출하지 않고 전체 이미지를 보기 때문에 객체에 대한 맥락적 이해가 높음.



03. SSD (Single Shot Multibox Detector)

□ Overview

- SSD 이전 버전 YOLO의 문제점

1) 7X7 그리드 영역으로 나눠 Bounding Box Prediction 진행(YOLO)

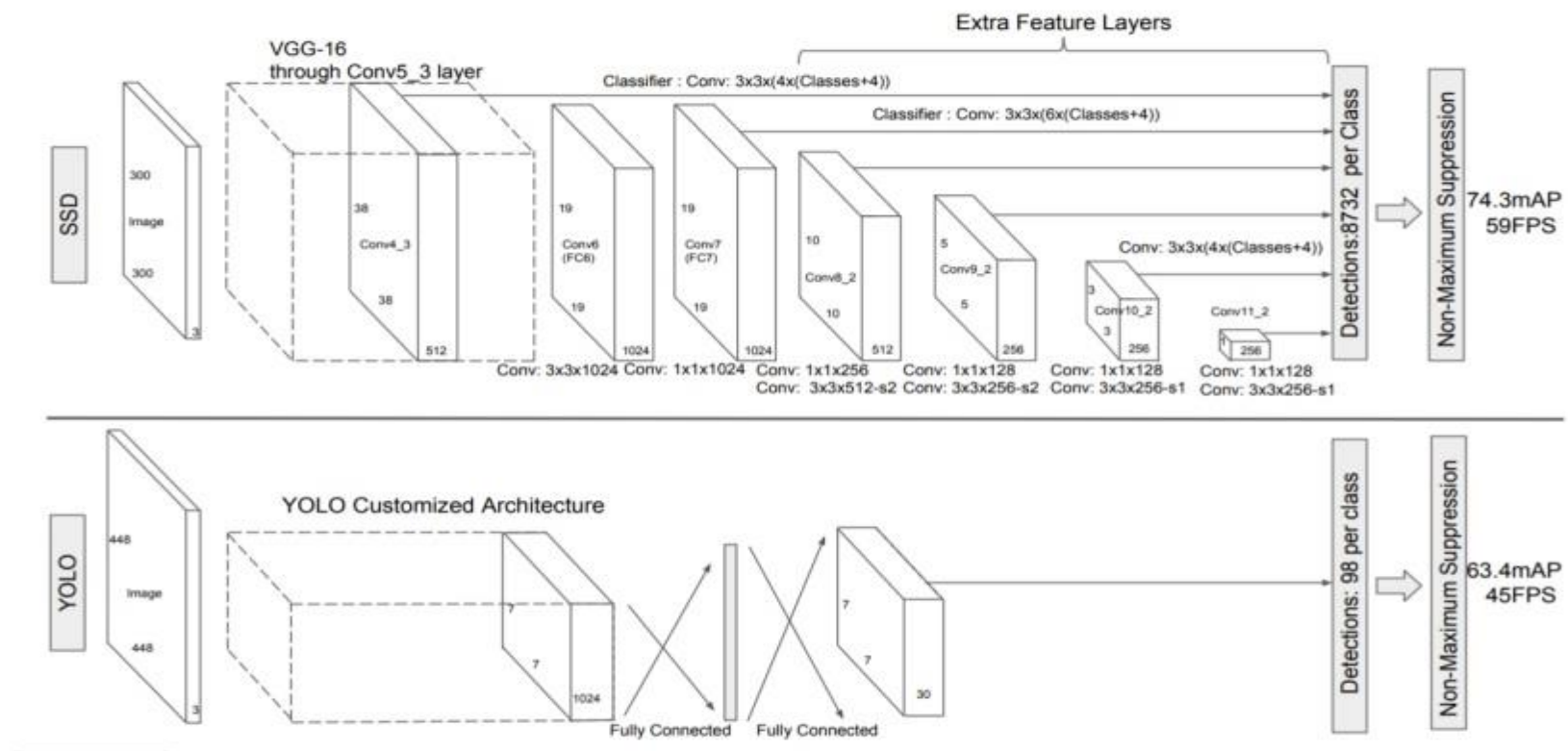
-> 그리드보다 작은 크기의 물체 검출 불가능.

-> **해결책 : Fully Convolution Network 사용**

2) 신경망을 통과하며 마지막 Feature 만 사용

-> 정확도 하락.

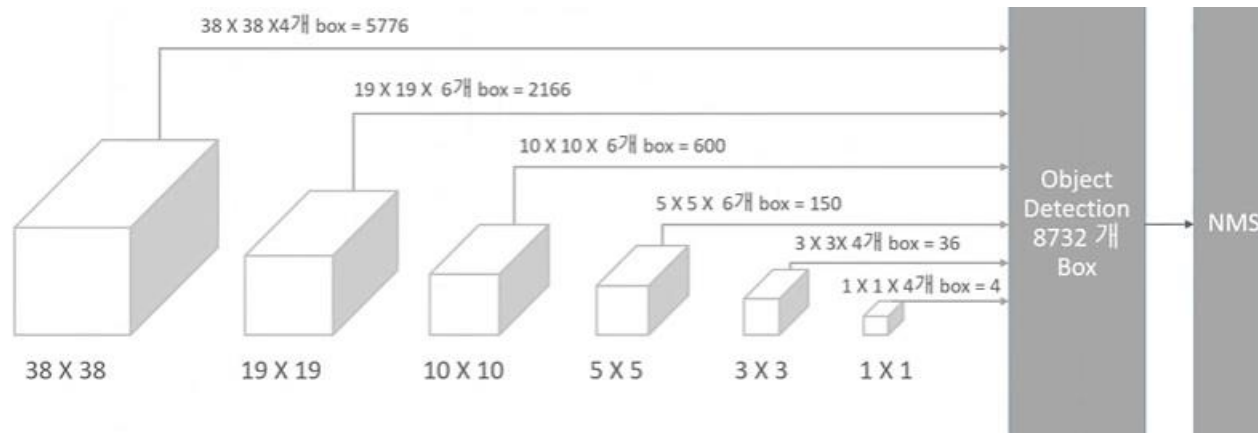
-> **해결책 : Faster RCNN의 Anchor 개념을 이용**



03. SSD (Single Shot Multibox Detector)

□ 구조

- Input 은 512x512 또는 300x300
 - Object Detection 단으로 넘어가는 Feature Map 의 크기가 38x38, 19x19, 10x10, 5x5, 3x3, 1x1 인 결과
 - Classifier : Conv, 3x3x(4x(classes+4))
 - > 3x3x(4x(classes+4)) : 3x3 필터를 사용.
 - > 3x3x(4x(classes+4)) : 4개의 Anchor Box
 - > 3x3x(4x(classes+4)) : 각 class 에 대한 Softmax 값, Bounding Box 좌표 값.
 - > 3x3x(4x(classes+4)) : Kernel 의 개수이자, Detection 결과 Output 의 Depth 값.
- 3x3 Kernel 의 Depth 는 각 Feature Map(38x38, 19x19, 10x10, 5x5, 3x3, 1x1) 의 Depth 값을 따름.
- 이렇게 Feature Map 을 통해 만들어지는 Box 개수는 다음과 같고 이것을 NMS 처리를 한다.



03. SSD (Single Shot Multibox Detector)

□ 특징

- Backbone 을 통과시키고 Feature map resolution 을 낮춰가면서 모든 Feature map 에 대해 Detection + Box regression 을 수행.
- Multi Scale Feature Layer
- Default (Anchor) Box
- Fully Connected Layer 대신 Convolution Layer 를 사용하여 속도 향상.

1. Multi Scale Feature Layer

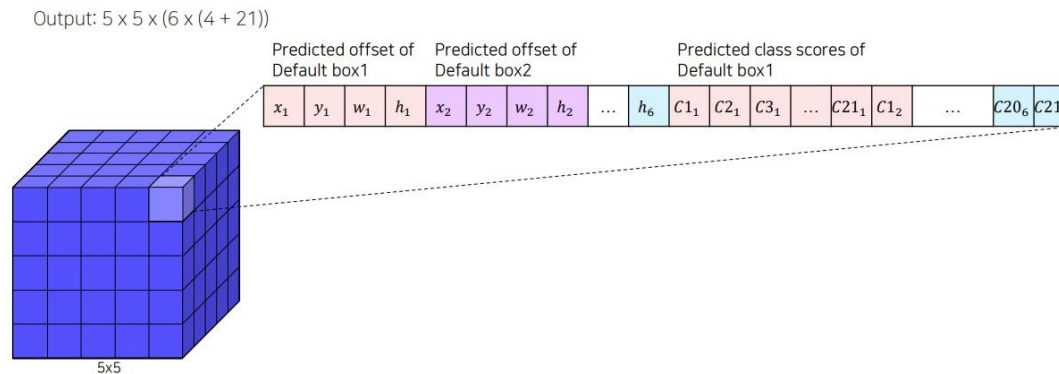
- Image Pyramid 기법을 차용하여 Object Detection 을 수행.
- Feature Map 을 convNet 에 통과시켜 크기를 줄여가며 각각 Detection 을 수행.
- 32x32 에서는 작은 Object, 4x4 에서는 큰 Object 를 Detect 가능.



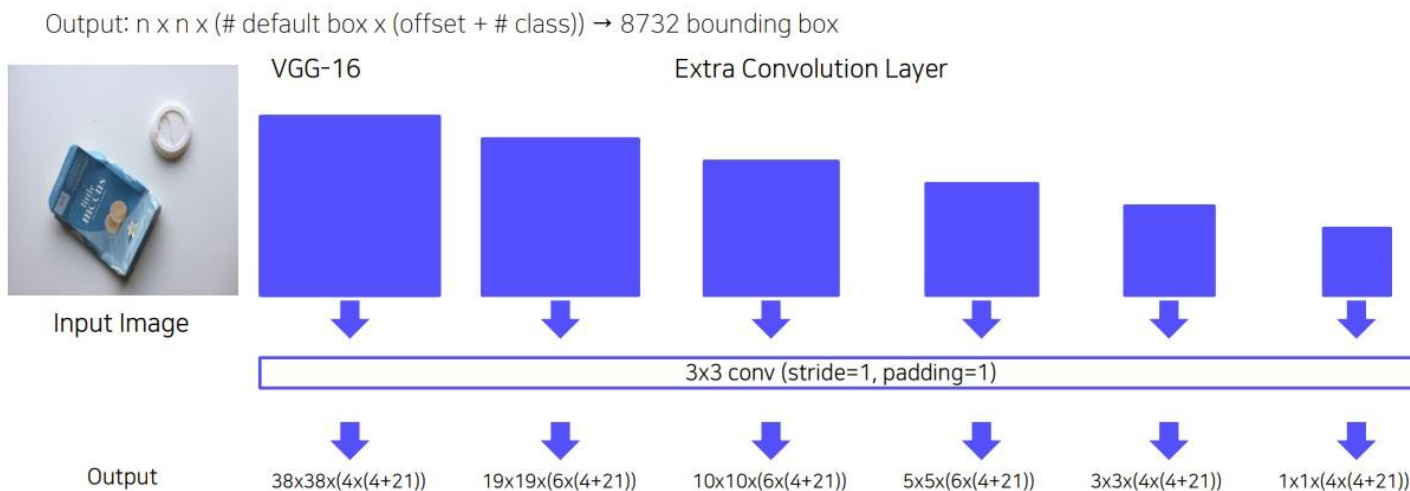
03. SSD (Single Shot Multibox Detector)

2. Default Box

- Faster RCNN 의 Anchor Box 와 유사.
- Feature map 의 각 cell 마다 서로 다른 scale, 비율을 가진 미리 정해진 box 생성.



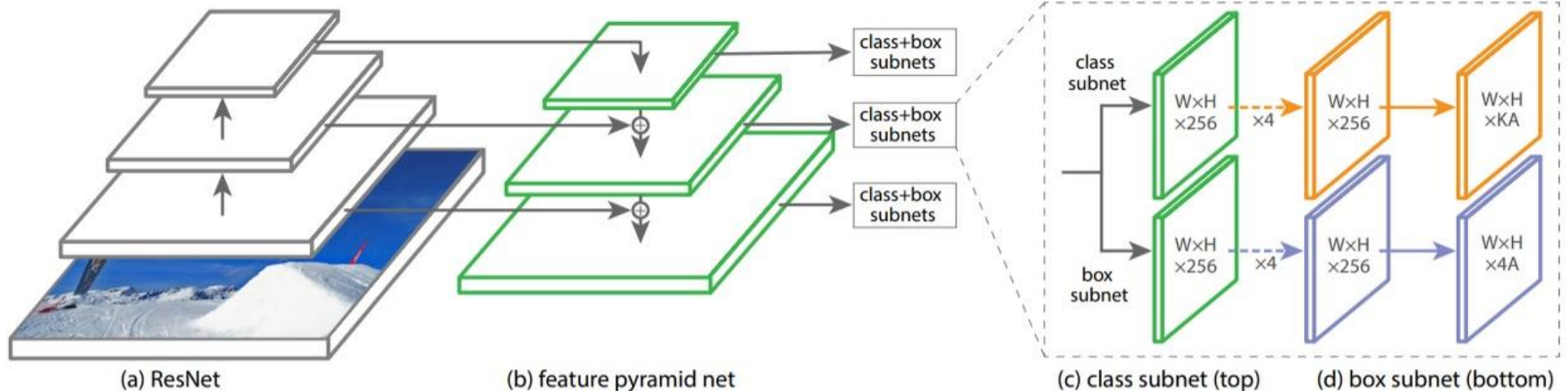
- 모델은 총 8732 개의 Default Box 를 사용.



04. RetinaNet (Focal Loss for Dense Object Detection)

□ Overview

- 기존의 1-Stage Detector 가 가지고 있는 문제점을 해결한 모델.
- 1-Stage Detector 는 RPN(Region Proposal 이 없고 Grid별로 Bounding Box 를 무조건 예측.
 - > Background 검출
- Class Imbalance
 - > Positive Sample(객체) < Negative Sample(배경)



04. RetinaNet (Focal Loss for Dense Object Detection)

□ 구조

1. Feature Pyramid by ResNet + FPN

- ResNet 구조에 FPN Backbone 을 사용, Multi-Scale Feature pyramid 를 생성, 각 pyramid level 은 256 채널.
- 각 pyramid level 에 aspect ratio 를 사용하여 Anchor 를 할당, 하나의 Anchor 에는 $K \times 4$ 의 vector 가 할당.
- Input : image
- Process : feature extraction by ResNet + FPN
- Output : feature pyramid(P5~P7)

2. Classification by Classification subnetwork

- Anchor 의 object class 를 예측하는 network.
- 각 pyramid level 에 KA (K :class, A :anchor) 개 filter 를 지닌 3×3 conv layer 가 4개로 구성된 Conv Layer 를 부착.
- classification subnet 의 출력값에 Focal Loss 를 적용.
- Input : feature pyramid(P5~P7)
- Process : classification by classification subnetwork
- Output : 5 feature maps with $K \times A$ channel

3. Bounding Box Regression by Bounding Box Regression subnetwork

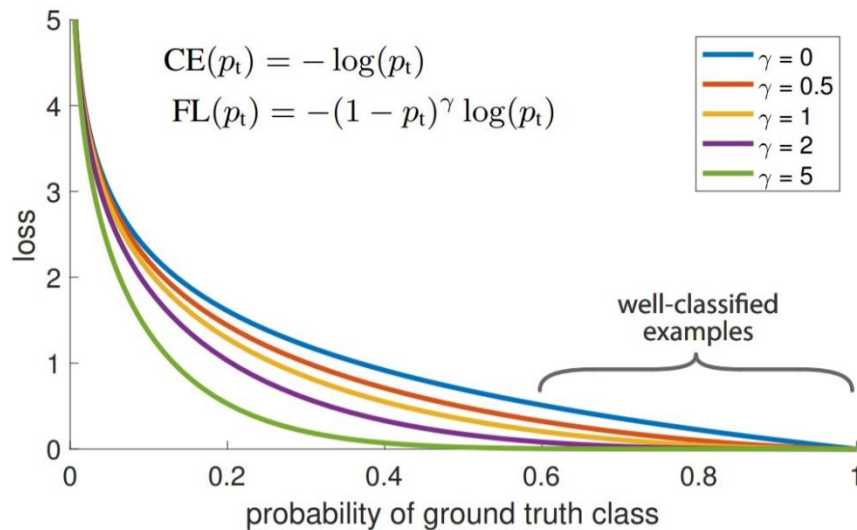
- Anchor 와 ground-truth 의 offset 을 계산하는 network.
- Classification subnet 과 동일하지만 마지막에 $4A$ 길이를 출력.
- Input : feature pyramid(P5~P7)
- Process : bounding box regression by bounding box regression subnet
- Output : 5 feature maps with $4 \times A$ channel

04. RetinaNet (Focal Loss for Dense Object Detection)

□ 특징

- Focal Loss

- > 1-Stage detector 모델에서 foreground 와 background class 사이에서 발생하는 극단적인 class imbalance 문제를 해결하는데 사용.
- > Cross Entropy(CE) 에 scaling factor 를 추가.



		Probability(p_t)	
		High	Low
Class	$Y=1$	Down-weight	Concentrate
	$Y=-1$	Concentrate	Down-weight

1) p_t 와 Modulating factor 와의 관계

example이 잘못 분류되고, p_t 가 작으면, modulating factor는 1과 가까워지며, loss는 영향을 받지 않습니다. 반대로 p_t 값이 크면 modulating factor는 0에 가까워지고, 잘 분류된 example의 loss는 down-weight됩니다.

2) focusing parameter γ 의 역할

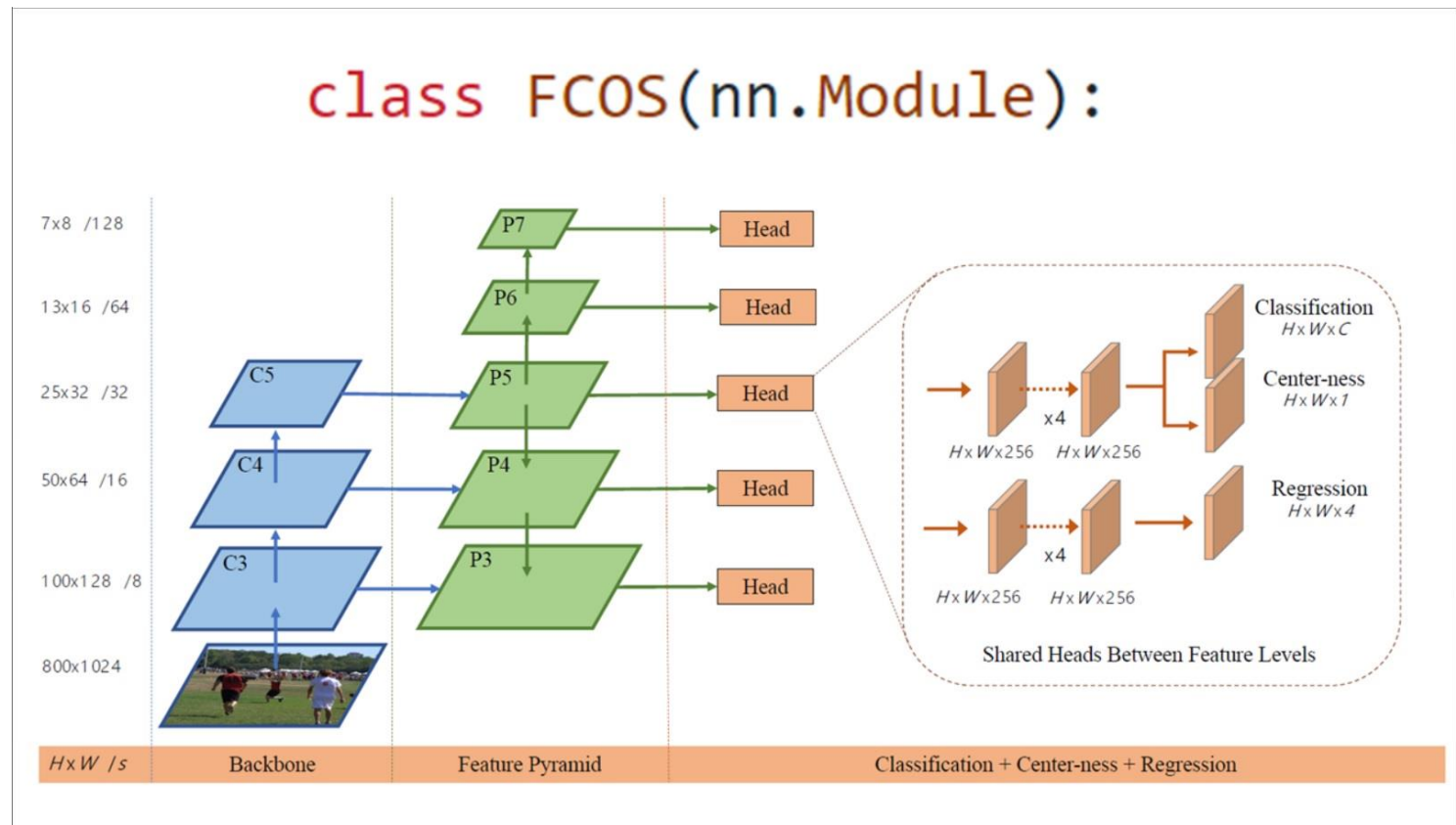
focusing parameter γ 는 easy example을 down-weight하는 정도를 부드럽게 조정합니다.

$\gamma=0$ 인 경우, focal loss는 CE와 같으며, γ 가 상승할수록 modulating factor의 영향력이 커지게 됩니다.

05. FCOS (Fully Convolutional One Stage Object Detection)

□ Overview

- Anchor-based detector 에서 발생하는 단점을 개선하기 위해 제안.
- Anchor Box 설계 형태에 따라 모델의 성능에 영향.
- Anchor Box 크기와 다른 ground-truth 를 검출하기 어려움.
- 많은 수의 Anchor Box 가 negative 로 할당되므로 class imbalance 문제 발생.
- Anchor Box 와 ground-truth 와의 iou 를 계산해야 하므로 계산과정 복잡



05. FCOS (Fully Convolutional One Stage Object Detection)

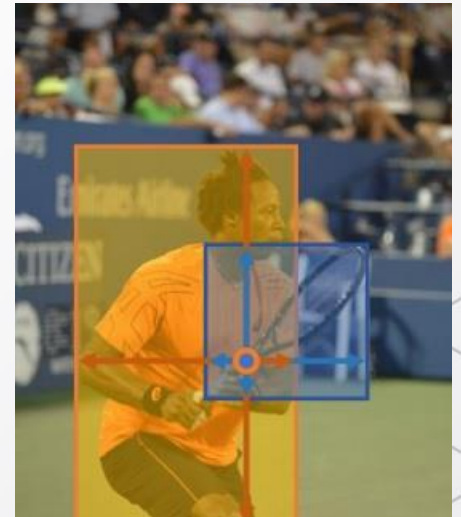
□ 구조

1. Fully Convolutional One-Stage Object Detection

- 중심점(x,y)부터 예측한 Bounding Box 의 경계까지의 거리를 예측.
- (x,y)가 gt(ground-truth) 범위 안에 존재하고 class 가 동일하면 positive 로 간주.
- (x,y)가 여러 gt 안에 존재한다면 ambiguous sample 로 간주하며 multi-level prediction with FPN 에서 처리

2. Multi-Level Prediction with FPN for FCOS

- C3, C4, C5 에 1x1 convolution 을 통해 feature 를 얻고 P5->P4->P3 의 top-down 방향으로 연결.
- P6, P7 은 P5 부터 시작하여 stride 2를 차례로 적용하여 P5->P6->P7 순서로 생성.
- P3=8, P4=16, P5=32, P6=64, P7=128 에 해당하는 stride 크기를 가지는 feature pyramid 를 형성.
- 오른쪽 그림처럼 bounding box 가 overlap 된 경우 작은 박스는 P3, 큰 박스는 P6 에서 검출



3. Center-ness

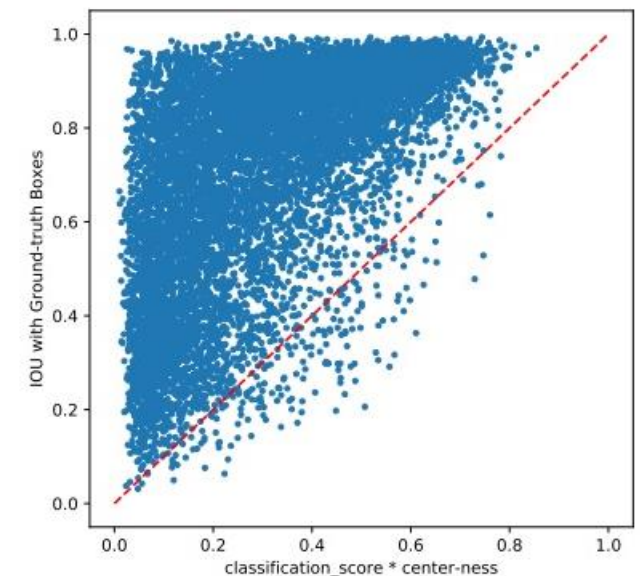
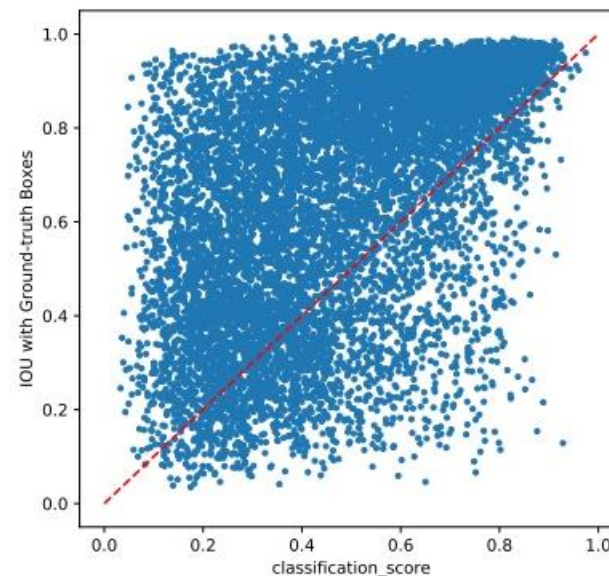
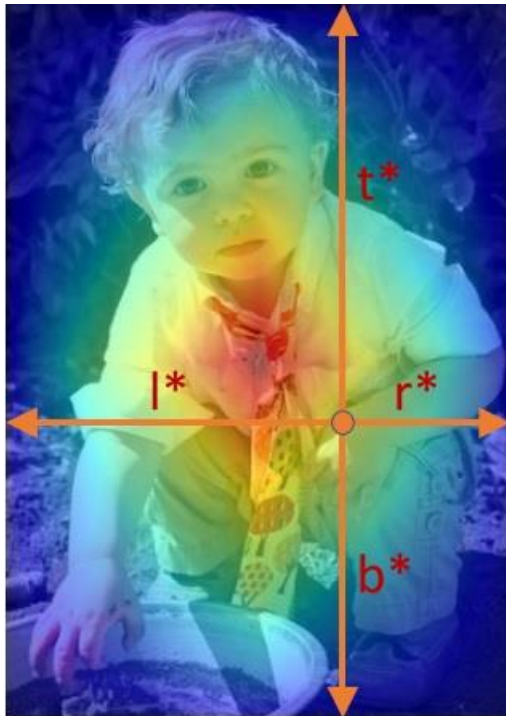
- low-quality predicted bounding box 를 제거하기 위해 사용.
- center-ness 는 중심점과의 거리를 정규화.
- 중심점과 가까우면 1에 가까운 값, 중심점과 멀리 존재하면 0 에 가까운 값.
- center-ness 는 class score 와 곱하여 중심점에서 멀리 떨어진 bounding box 는 class score 값이 낮기 때문에 nms 에서 제거.

05. FCOS (Fully Convolutional One Stage Object Detection)

□ 특징

- 중심점에서 Bounding Box 경계까지의 거리 예측.
- Multi-level prediction with FPN.
- Center-ness
- Anchor Box 를 사용하지 않는 1-Stage Detector

□ Center-ness



center-ness 는 class score 와 곱하여 사용합니다. 중심점과 멀리 떨어진 bounding box 는 class score 값이 낮아지기 때문에 nms 에서 걸러지게 됩니다.

06. Performance

Method	Backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
Two-stage methods:							
Faster R-CNN w/ FPN [14]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [11]	Inception-ResNet-v2 [27]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w/ TDM [25]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
One-stage methods:							
YOLOv2 [22]	DarkNet-19 [22]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [18]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [5]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [15]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
CornerNet [13]	Hourglass-104	40.5	56.5	43.1	19.4	42.7	53.9
FSAF [34]	ResNeXt-64x4d-101-FPN	42.9	63.8	46.3	26.6	46.2	52.7
FCOS	ResNet-101-FPN	41.5	60.7	45.0	24.4	44.8	51.6
FCOS	HRNet-W32-51 [26]	42.0	60.4	45.3	25.4	45.0	51.0
FCOS	ResNeXt-32x8d-101-FPN	42.7	62.2	46.1	26.0	45.6	52.6
FCOS	ResNeXt-64x4d-101-FPN	43.2	62.8	46.6	26.5	46.2	53.3
FCOS w/ improvements	ResNeXt-64x4d-101-FPN	44.7	64.1	48.4	27.6	47.5	55.6

07. 인식 결과 비교 (FasterRCNN+InceptionResNet V2 vs ssd + mobilenet V2)

```
import tensorflow_hub as hub

def run_detector(detector, path):
    img = load_img(path)

    converted_img = tf.image.convert_image_dtype(img, tf.float32)[tf.newaxis, ...]
    start_time = time.time()
    result = detector(converted_img)
    end_time = time.time()

    result = {key:value.numpy() for key,value in result.items()}

    print("Found %d objects." % len(result["detection_scores"]))
    print("Inference time: ", end_time-start_time)

    image_with_boxes = draw_boxes(
        img.numpy(), result["detection_boxes"],
        result["detection_class_entities"], result["detection_scores"])

    display_image(image_with_boxes)

#FasterRCNN+InceptionResNet V2: 높은 정확성
#ssd + mobilenet V2: 작고 빠름
module_handle = "https://tfhub.dev/google/openimages_v4/ssd/mobilenet_v2/1" #ssd + mobilenet V2
#module_handle = "https://tfhub.dev/google/faster_rcnn/openimages_v4/inception_resnet_v2/1" #FasterRCNN+InceptionResNet V2

detector = hub.load(module_handle).signatures['default']

run_detector(detector, downloaded_image_path)
```


07. 인식 결과 비교 (FasterRCNN+InceptionResNet V2 vs ssd + mobilenet V2)

```
def detect_img(image_url):  
    start_time = time.time()  
    image_path = download_and_resize_image(image_url, 640, 480)  
    run_detector(detector, image_path)  
    end_time = time.time()  
    print("Inference time:", end_time-start_time)  
  
detect_img(image_urls[0])  
detect_img(image_urls[1])  
detect_img(image_urls[2])
```

□ 사용 모듈

1. FasterRCNN+InceptionResNet V2:

- https://tfhub.dev/google/faster_rcnn/openimages_v4/inception_resnet_v2/1

2. ssd + mobilenet V2

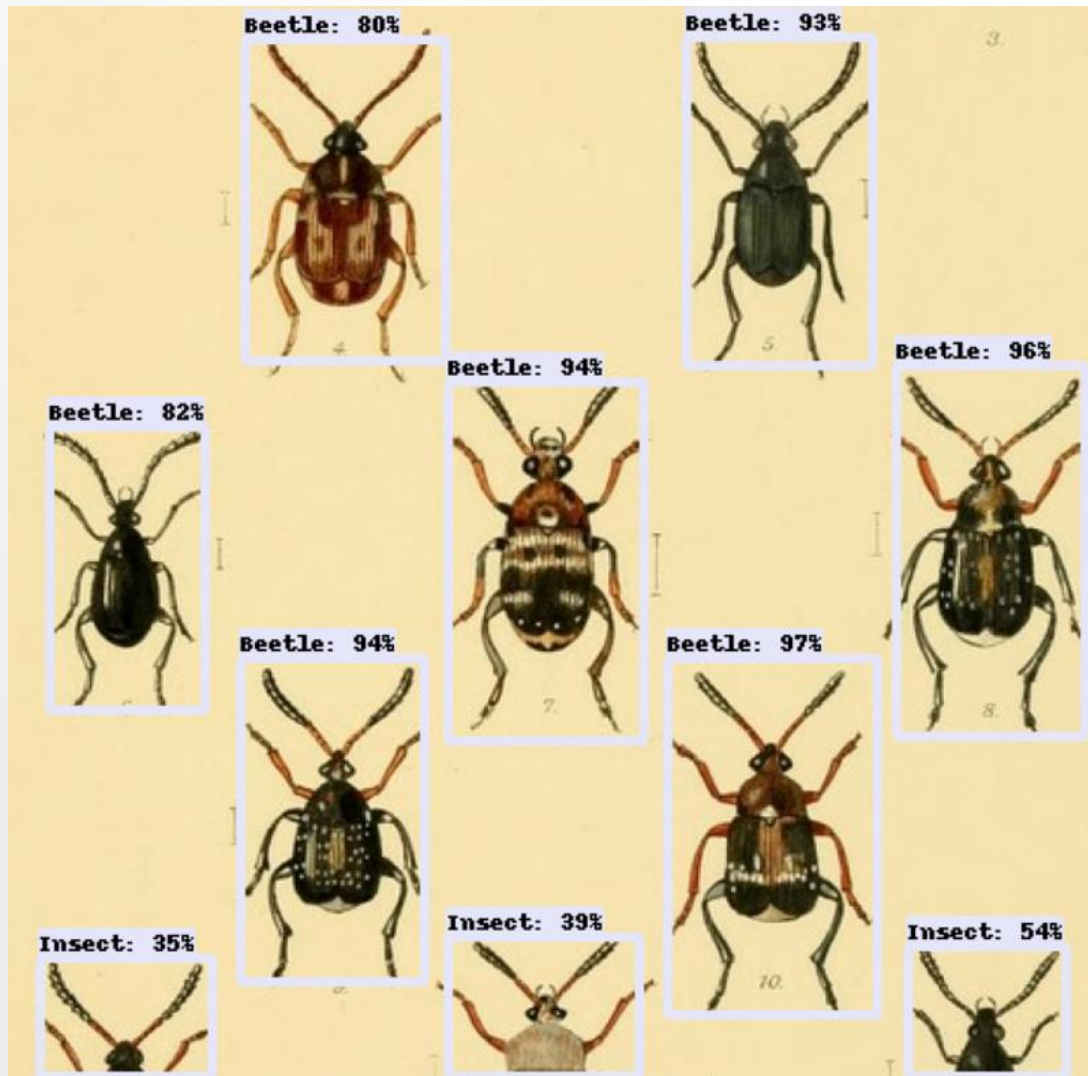
- https://tfhub.dev/google/openimages_v4/ssd/mobilenet_v2/1

3. 테스트 환경

- OS : Windows 10
- GPU : GeForce RTX 2060
- CPU : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz
- RAM : 16.0GB

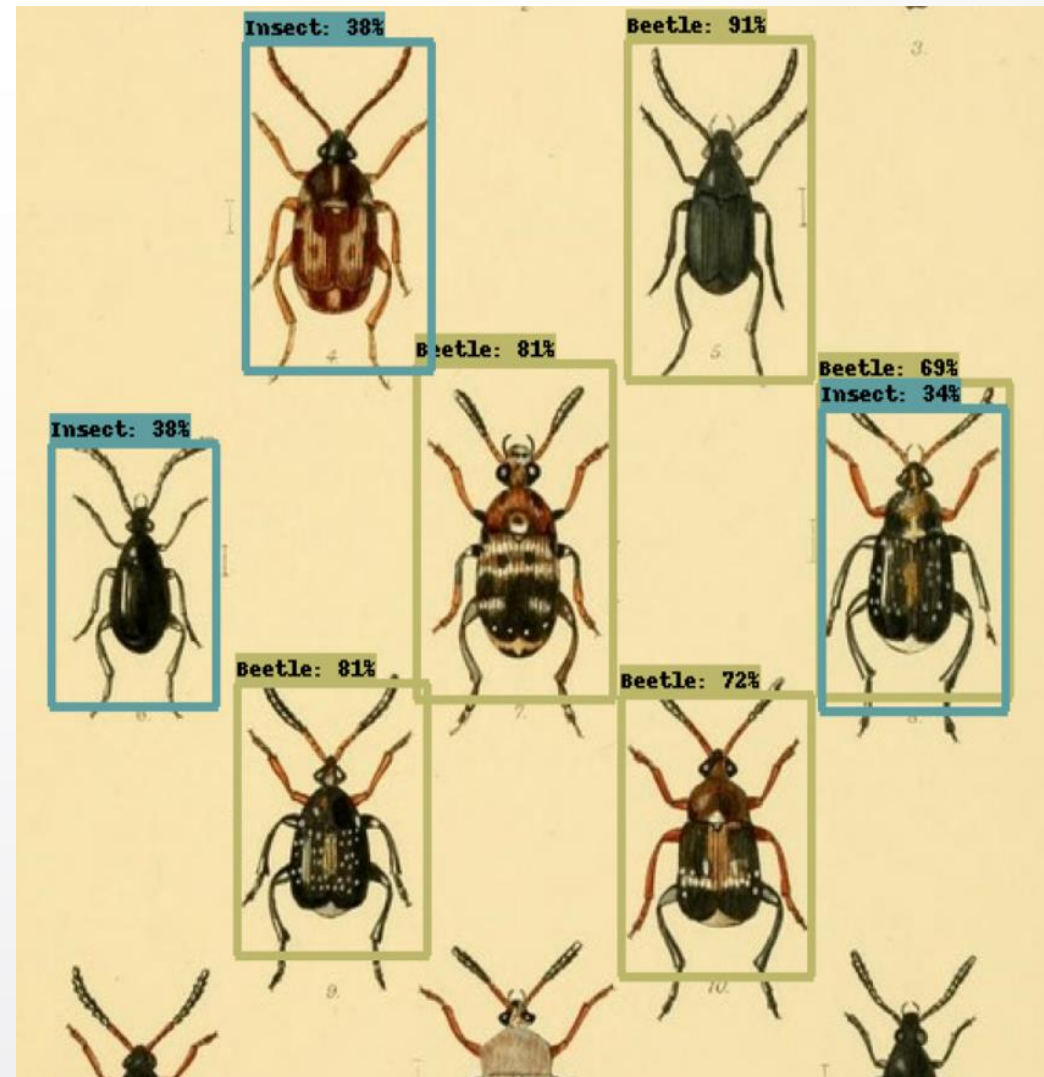
07. 인식 결과 비교 (FasterRCNN+InceptionResNet V2 vs ssd + mobilenet V2)

FasterRCNN



인식 시간 : 3.449643611907959

SSD



인식 시간 : 0.3375563621520996

07. 인식 결과 비교 (FasterRCNN+InceptionResNet V2 vs ssd + mobilenet V2)

FasterRCNN



인식 시간 : 1.5846748352050781

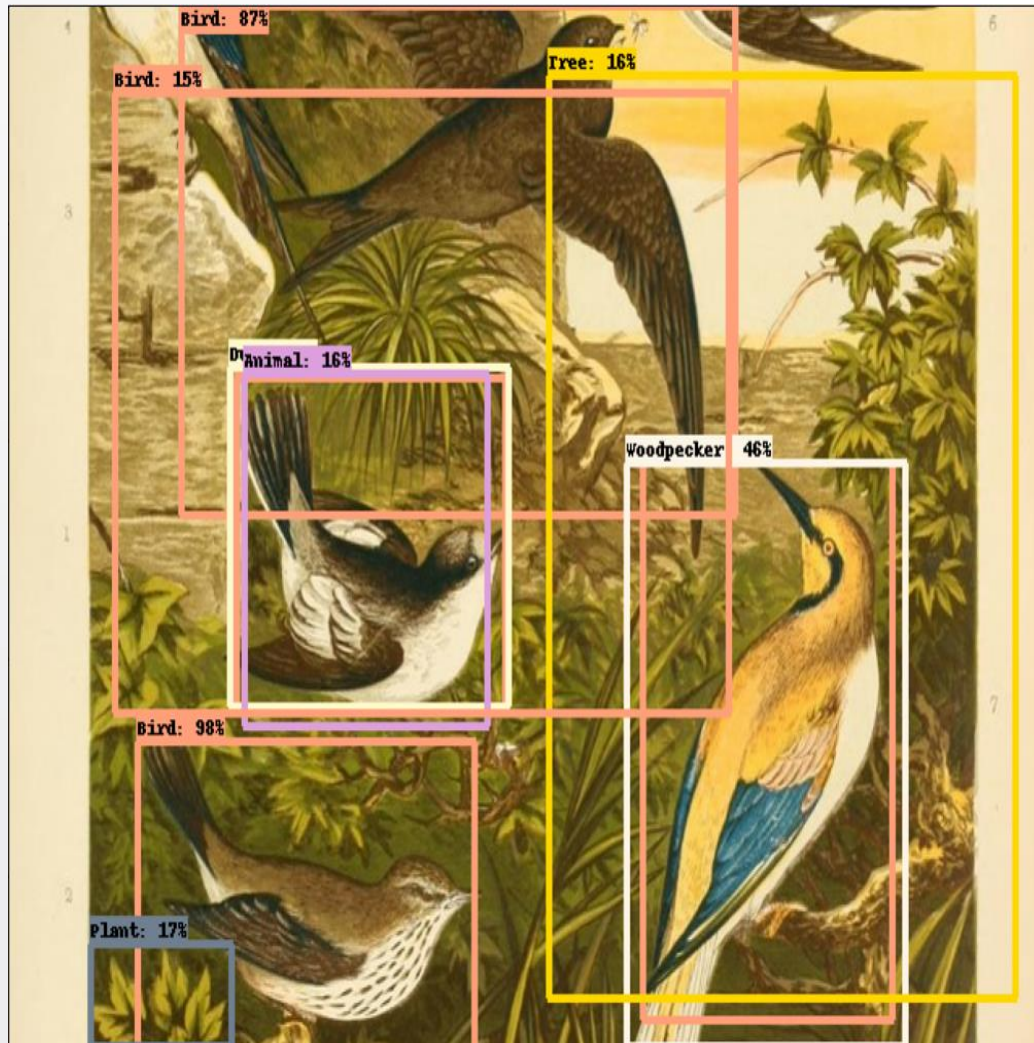
SSD



인식 시간 : 0.32801294326782227

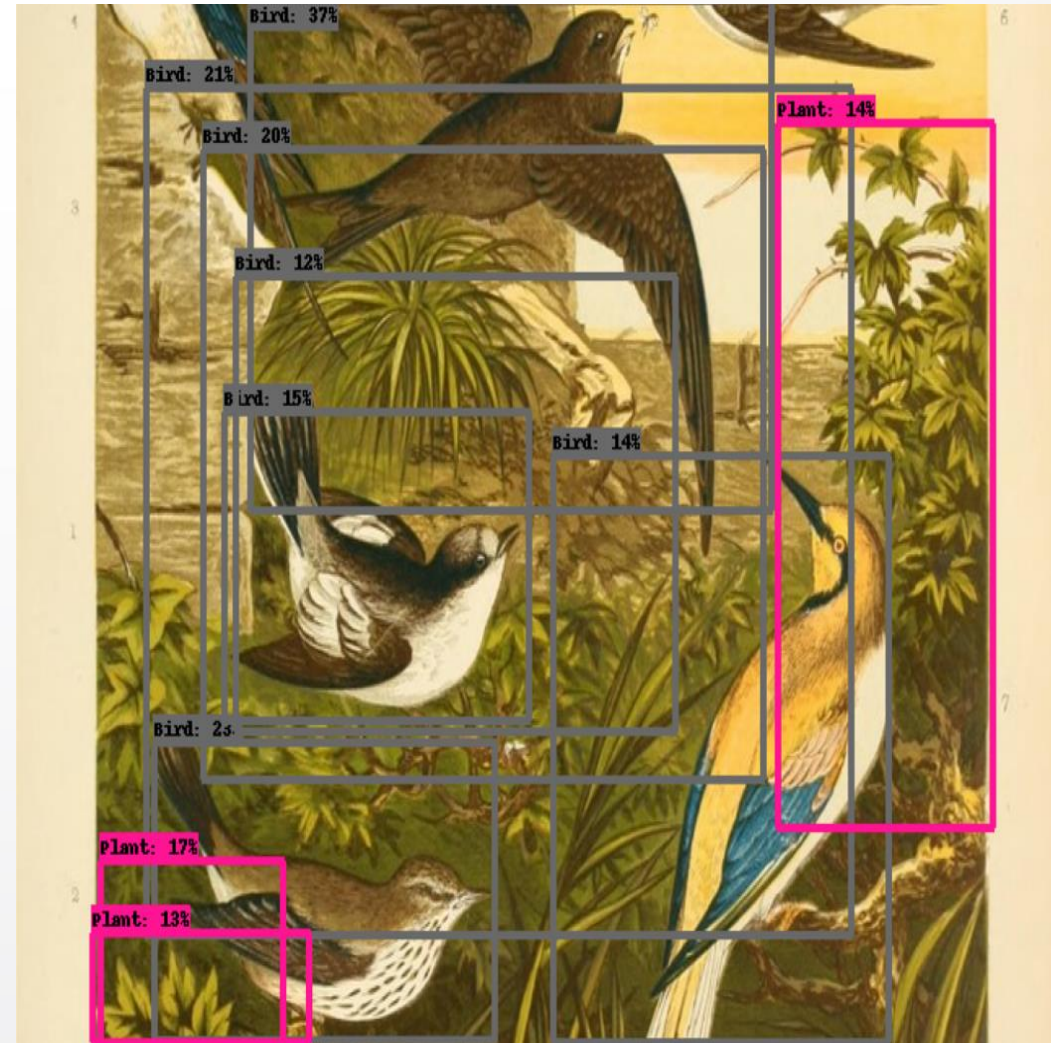
07. 인식 결과 비교 (FasterRCNN+InceptionResNet V2 vs ssd + mobilenet V2)

FasterRCNN



인식 시간 1.585092544555664

SSD



인식 시간 : 0.35207653045654297

Q & A



- End -