Advanced Computer Vision

Annotation/Object detection/Segmentation

221006

고 우 영

https://github.com/airobotlab/lecture_5_yolov5

Requirement

- **Requirement**
  - 카메라에서 실시간으로 영상정보 획득 from 천장 (실내, 조명/위치 고정)
  - **4fps**(1/4초) 객체인식
  - 객체인식 위치정보 전송 to PLC
- **개발 전 고려사항**
  - 입력 이미지 사이즈 (3000x4000?, 2000x1000?, 640x640?, 224x224?)
  - 어떤 machine?
    - 노트북, 미니컴(Rpi, JetsonNano, Jetson orin), 서버, Cloud
    - GPU 사용 가능 여부
    - 재학습 여부(Continual learning or re-training)
  - Machine 사양에 따른 모델 선정
    - 크고 성능이 좋은 모델, 작고 빠른 모델, 모델 최적화 필요 여부 등
  - Python? C?

입력크기/속도

4 fps -> 250 ms/image 이하

Nano

YOLOv5n

4 MB_{FP16}
6.3 ms_{V100}
28.4 mAP_{COCO}

Small

YOLOv5s

14 MB_{FP16}
6.4 ms_{V100}
37.2 mAP_{COCO}

Medium

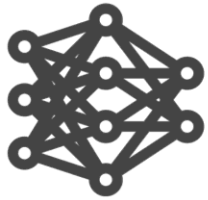
YOLOv5m

41 MB_{FP16}
8.2 ms_{V100}
45.2 mAP_{COCO}

Large

YOLOv5l

89 MB_{FP16}
10.1 ms_{V100}
48.8 mAP_{COCO}



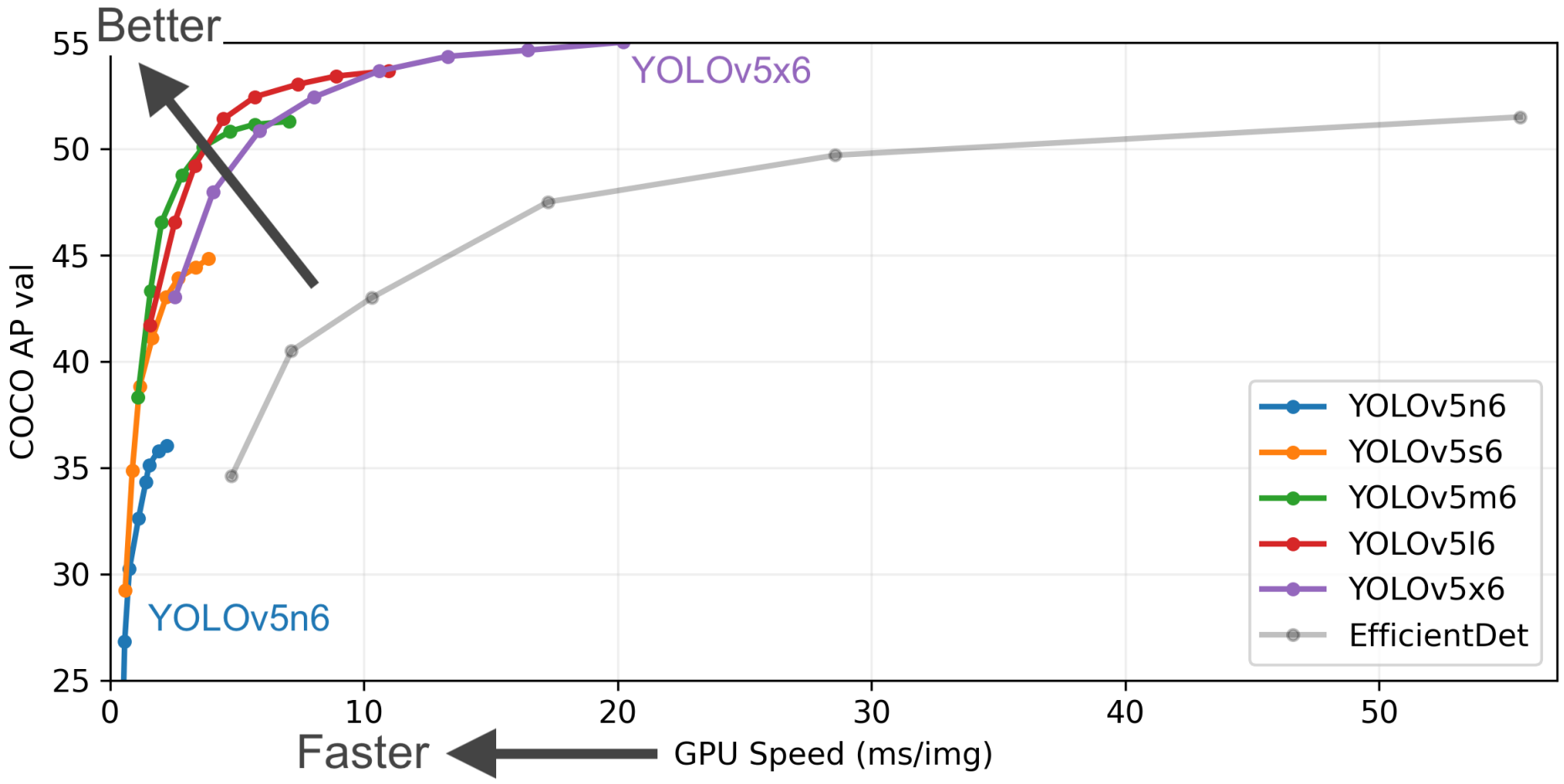
XLarge

YOLOv5x

166 MB_{FP16}
12.1 ms_{V100}
50.7 mAP_{COCO}

입력크기/속도

4 fps -> 250 ms/image 이하



입력크기/속도

4 fps -> 250 ms/image 이하

Model	size (pixels)	mAP^{val} 0.5:0.95	mAP^{val} 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	**45**	**6.3**	**0.6**	**1.9**	**4.5**
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6 + TTA	1280 1536	55.0 **55.8**	72.7 **72.7**	3136 -	26.2 -	19.4 -	140.7 -	209.8 -

NVIDIA TRT Optimization

4 fps -> 250 ms/image 이하

- **만약 더 빠르게 하고 싶으면?**
  - 그래프/Floating point 최적화
- **만약 python을 못쓰는 환경이면?**
  - 언어 변환 필요
  - Embedded board: C언어
  - 웹: Java or C#
- **속도+언어 둘다 해결하고 싶다?**
  - ONNX 포팅
  - TRT에서 구동

NVIDIA TRT Optimization

4 fps -> 250 ms/image 이하

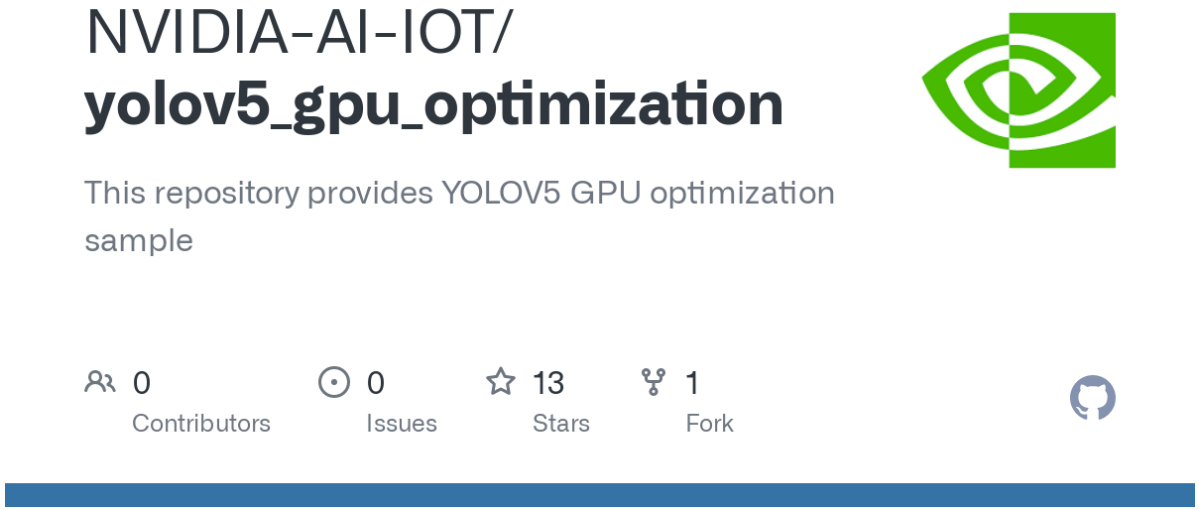
```
# 모델에 대한 입력값
x = torch.randn(batch_size, 1, 224, 224, requires_grad=True)
torch_out = torch_model(x)

# 모델 변환
torch.onnx.export(torch_model,               # 실행될 모델
                  x,                         # 모델 입력값 (튜플 또는 여러 입력값들도 가능)
                  "super_resolution.onnx",   # 모델 저장 경로 (파일 또는 파일과 유사한 객체 모두 가능)
                  export_params=True,        # 모델 파일 안에 학습된 모델 가중치를 저장할지의 여부
                  opset_version=10,          # 모델을 변환할 때 사용할 ONNX 버전
                  do_constant_folding=True,  # 최적화시 상수폴딩을 사용할지의 여부
                  input_names = ['input'],   # 모델의 입력값을 가리키는 이름
                  output_names = ['output'], # 모델의 출력값을 가리키는 이름
                  dynamic_axes={'input' : {0 : 'batch_size'},    # 가변적인 길이를 가진 차원
                                'output' : {0 : 'batch_size'}})
```

- **속도+언어 둘다 해결하고 싶다?**
  - ONNX 포팅
  - TRT에서 구동

NVIDIA TRT Optimization

4 fps -> 250 ms/image 이하



https://github.com/NVIDIA-AI-IOT/yolov5_gpu_optimization

NVIDIA TRT Optimization

4 fps -> 250 ms/image 이하

NVIDIA TRT Optimization 후 속도 **2**배 증가, 정확도 큰 차이 없음

Model	size (pixels)	mAP^{val} 0.5:0.95	mAP^{val} 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed	params	FLOPs
YOLOv5n	640	28.0	45.7	**45**	**6.3**			
YOLOv5s	640	37.4	56.8	98	6.4			
YOLOv5m	640	45.4	64.1	224	8.2			
YOLOv5l	640	49.0	67.3	430	10.1			
YOLOv5x	640	50.7	68.9	766	12.1			
YOLOv5n6	1280	36.0	54.4	153	8.1			
YOLOv5s6	1280	44.8	63.7	385	8.2			
YOLOv5m6	1280	51.3	69.3	887	11.1			
YOLOv5l6	1280	53.7	71.3	1784	15.8			
YOLOv5x6 + TTA	1280 1536	55.0 **55.8**	72.7 **72.7**	3136 -	26.2 -			

Model	Input Size	precision	FPS bs=32	FPS bs= 1	mAP@0.5
yolov5n	640	FP16	1295	448	45.9%
yolov5s	640	FP16	917	378	57.1%
yolov5m	640	FP16	614	282	64%
yolov5l	640	FP16	416	202	67.3%
yolov5x	640	FP16	231	135	68.5%
yolov5n6	1280	FP16	341	160	54.2%
yolov5s6	1280	FP16	261	139	63.2%
yolov5m6	1280	FP16	155	99	68.8%
yolov5l6	1280	FP16	106	68	70.7%
yolov5x6	1280	FP16	60	45	71.9%

개발 순서

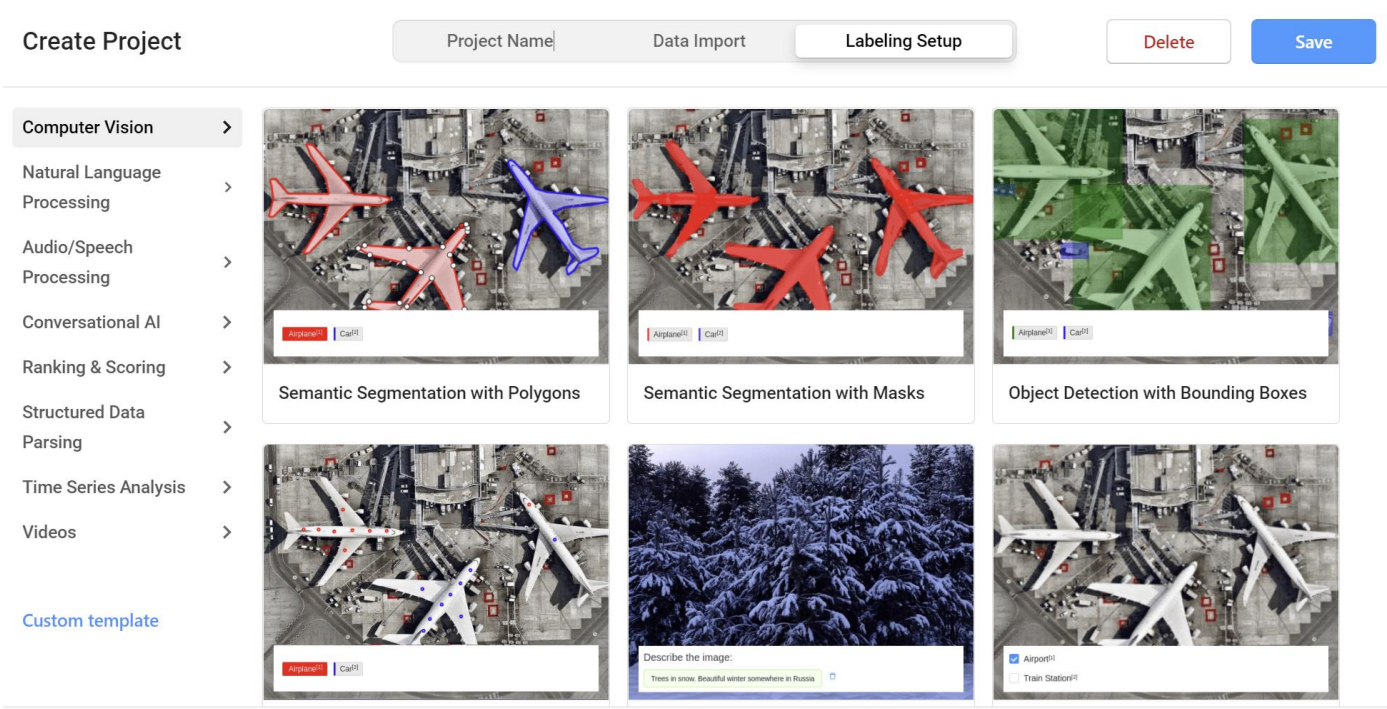
- **1) 데이터 Annotation**
  - Label-studio
- **2) Yolov5 학습**
  - https://github.com/ultralytics/yolov5
- **3) 학습모델을 이용한 추론**

Annotation

LabelStudio

Label Studio

- https://labelstud.io/
- **# Install the package**
  - **pip install -U label-studio**
- **# Launch it!**
  - **label-studio**



Label Studio

- https://labelstud.io/
- **# Install the package**
  - **pip install -U label-studio**
- **# Launch it!**
  - **label-studio**

Output preview

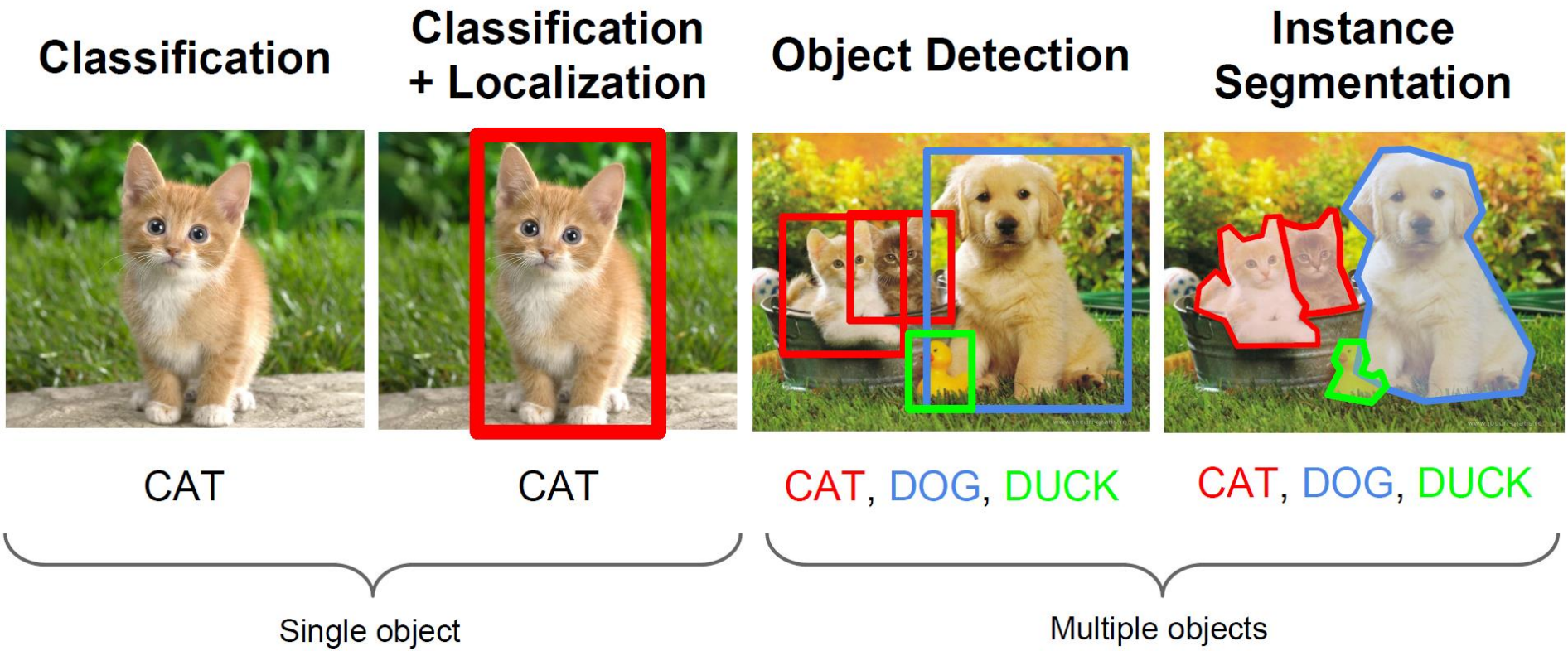

```
[
    {
        "original_width": 600,
        "original_height": 403,
        "image_rotation": 0,
        "value": {
            "x": 24.666666666666668,
            "y": 49.62779156327544,
            "width": 31.666666666666668,
            "height": 39.702233250620345,
            "rotation": 0,
            "rectanglelabels": [
                "Airplane"
            ]
        },
        "id": "yytSY7KW36",
        "from_name": "label",
        "to_name": "image",
        "type": "rectanglelabels"
    },
    {
        "original_width": 600,
        "original_height": 403,
        "image_rotation": 0,
        "value": {
            "x": 62.666666666666664,
            "y": 65.50868486352357,
            "width": 8.833333333333334,
            "height": 24.317617866004962,
            "rotation": 0,
            "rectanglelabels": [
                "Car"
            ]
        },
        "id": "ICNn84tJme",
        "from_name": "label",
        "to_name": "image",
        "type": "rectanglelabels"
    }
]
```

Object Detection

History/yolov5/DETR

Image Understanding

- 이미지에서 물체(object)를 탐지한 후, 해당 물체의 클래스와 위치를 예측하는 task

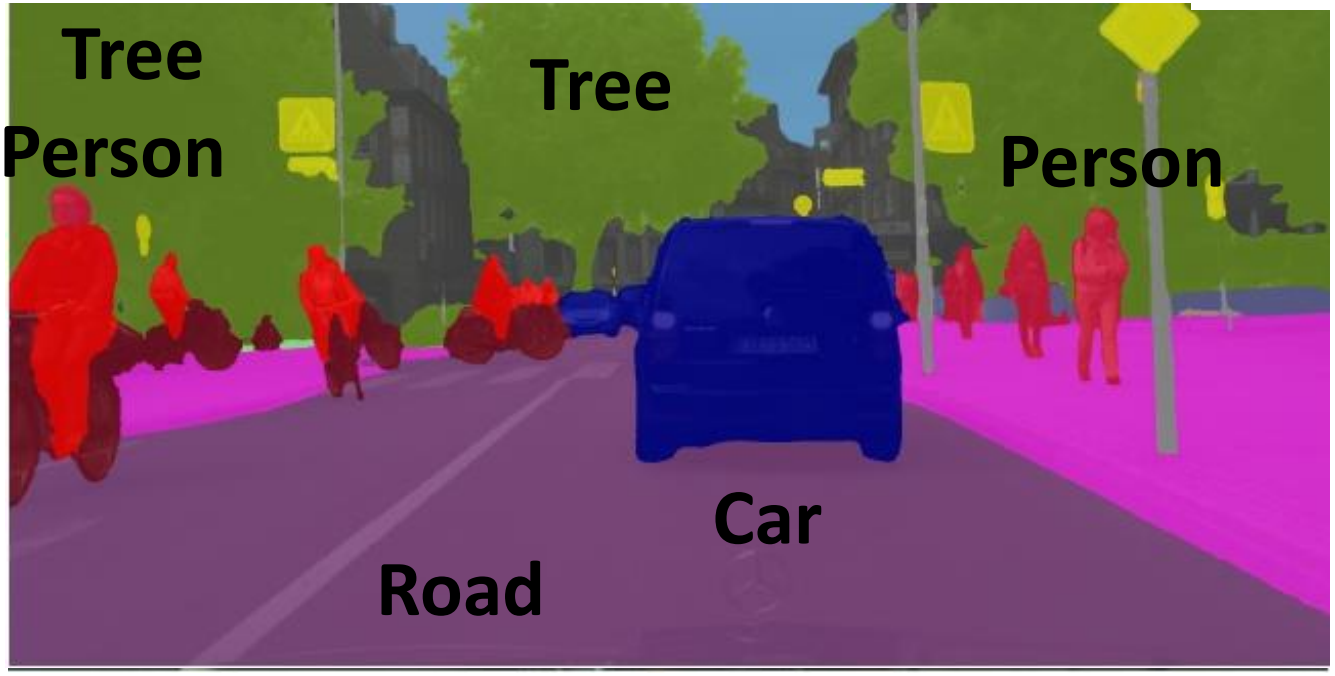


https://hoya012.github.io/blog/Tutorials-of-Object-Detection-Using-Deep-Learning-the-application-of-object-detection/


person.95
person.72
person.97
person.86
person.81
person.88
person.94
person.89
person.87
person.99
person.77
person.96
person.91
chair.98
chair.96
dining table.96
dining table.75
cup.75
cup.71
chair.99
chair.98
chair.95
dining table.78
chair.92
wine glass.80
chair.85
cup.71
chair.87
chair.97
chair.94
wine glass.91
wine glass.93
cup.96
cup.91
wine glass.83

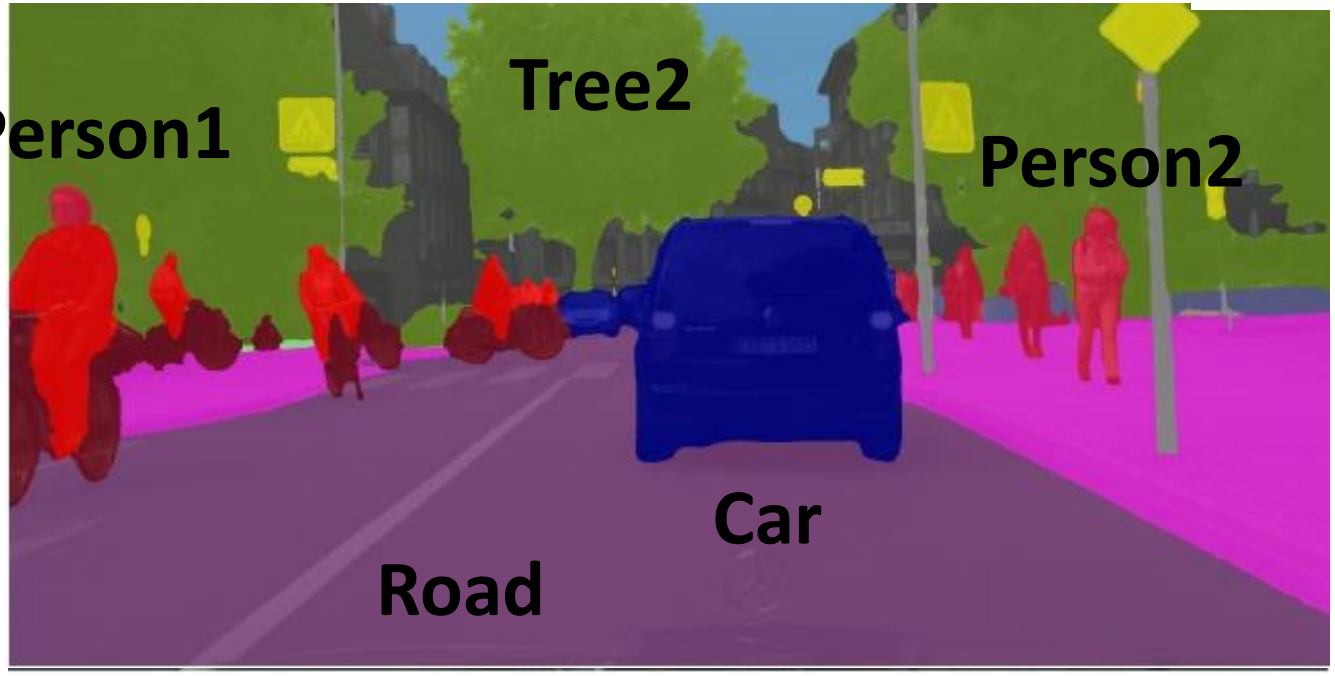
Image Understanding

Semantic Segmentation



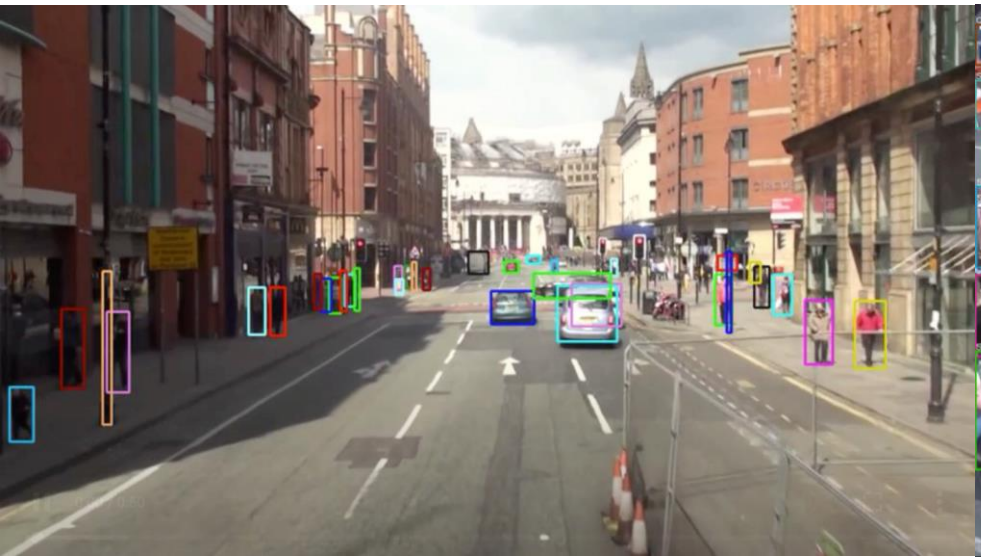
TUM, Computer Vision 3: Detection, Segmentation and Tracking
https://kr.123rf.com/%EC%9D%BC%EB%9F%AC%EC%8A%A4%ED%8A%B8/%EC%B9%B4%EB%A9%94%EB%9D%BC.html

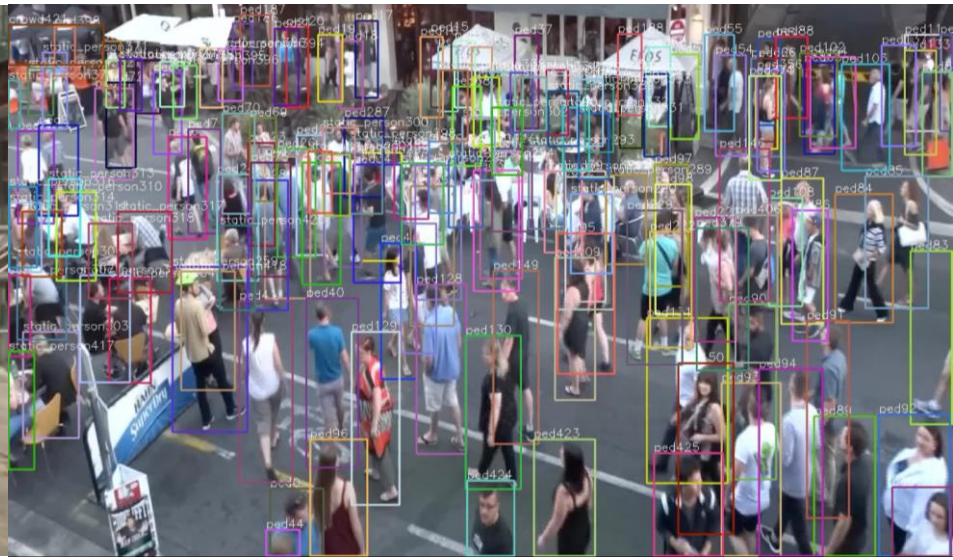
Instance Segmentation



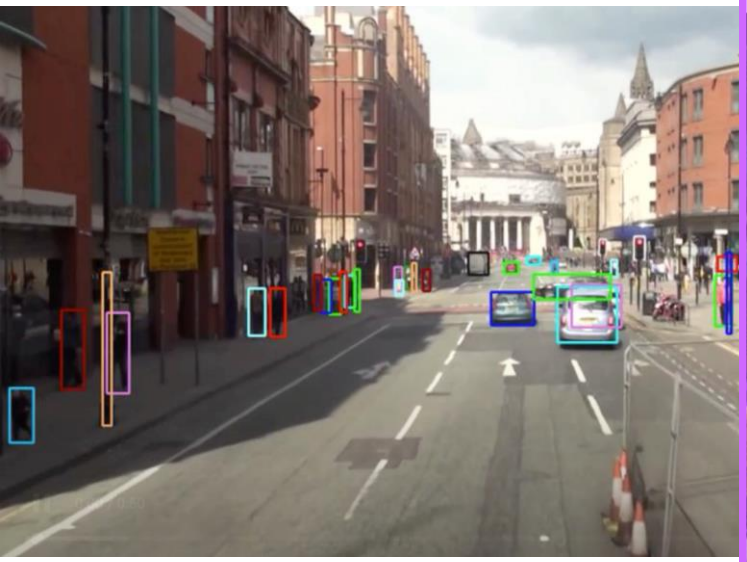
TUM, Computer Vision 3: Detection, Segmentation and Tracking
https://kr.123rf.com/%EC%9D%BC%EB%9F%AC%EC%8A%A4%ED%8A%B8/%EC%B9%B4%EB%A9%94%EB%9D%BC.html

Video Understanding

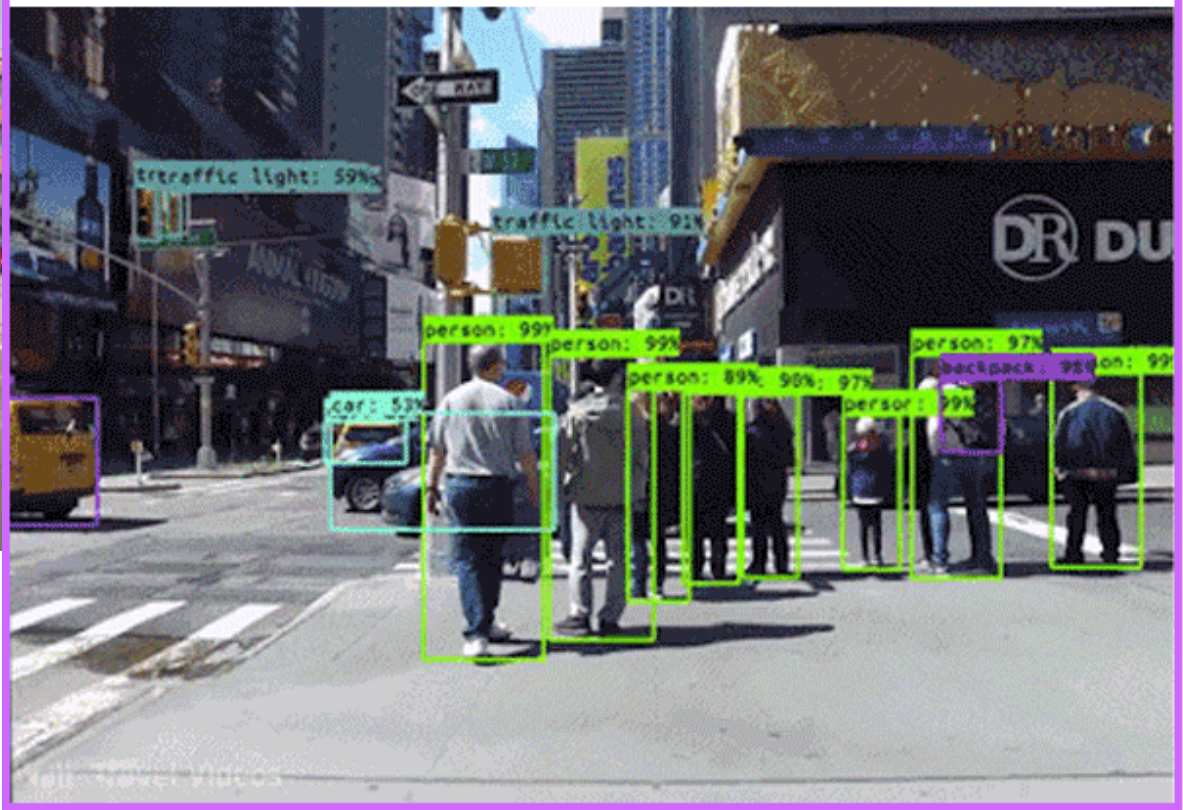




Video Understanding



자율주행



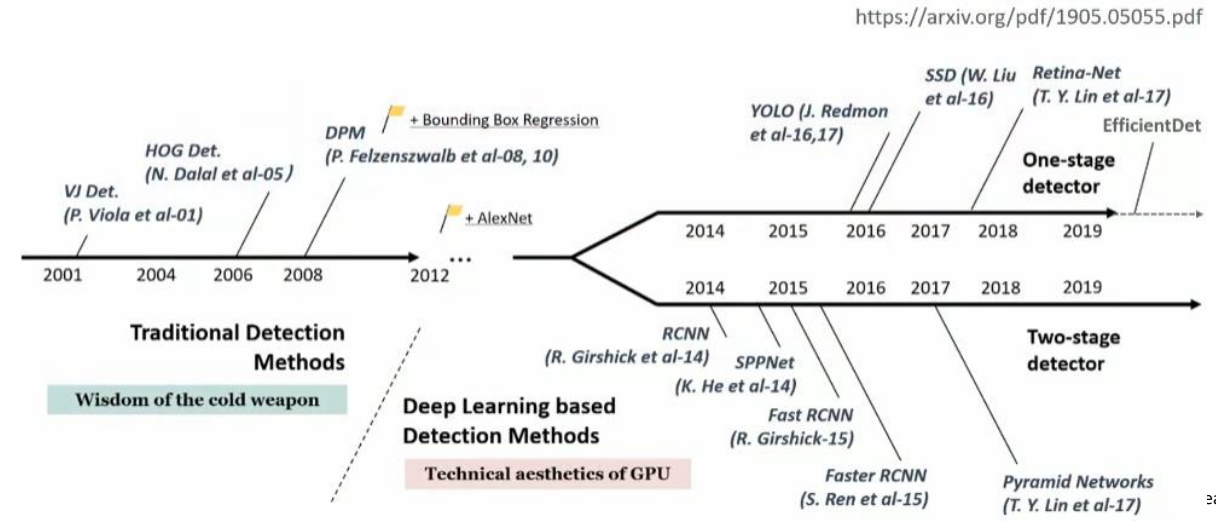
출처 : MEDIUM

Image Understanding

1) CNN
2) 이미지 분류 고급
3) Object Detection
4) Object Tracking
5) Segmentation

Object Detection

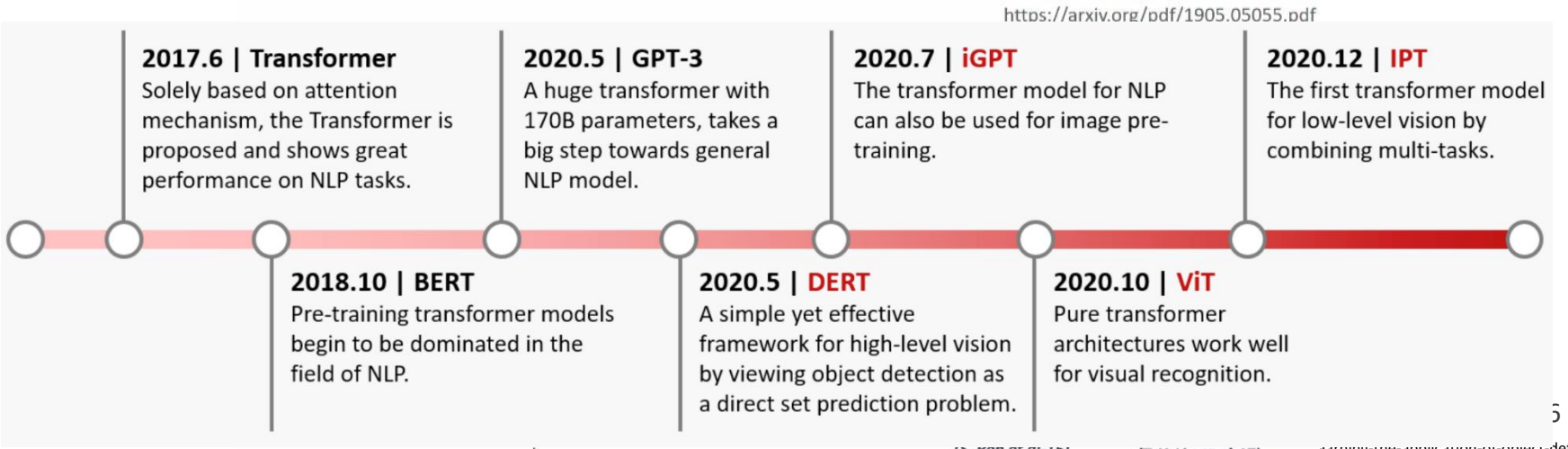
- 이미지에서 물체(object)를 탐지한 후, 해당 물체의 클래스와 위치를 예측하는 task
- Type
  - detector 앞단에 후보 지역을 추천하는 RPN(Region Proposal Network) 존재 여부
  - 1-stage detector : YOLO
    - 추론 속도가 빨라 real-time task
  - 2-stage detector: Faster R-CNN
    - 성능이 좋지만 느림



earning-the-application-of-object-detection/

Object Detection

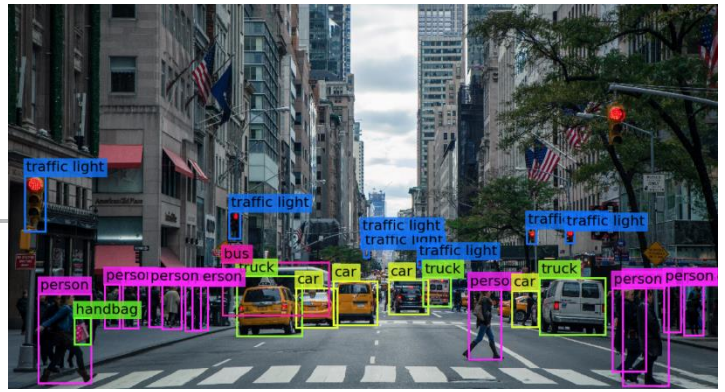
- 이미지에서 물체(object)를 탐지한 후, 해당 물체의 클래스와 위치를 예측하는 task
- Type
  - detector 앞단에 후보 지역을 추천하는 RPN(Region Proposal Network) 존재 여부
  - 1-stage detector : YOLO
    - 추론 속도가 빨라 real-time task
  - 2-stage detector: Faster R-CNN
    - 성능이 좋지만 느림

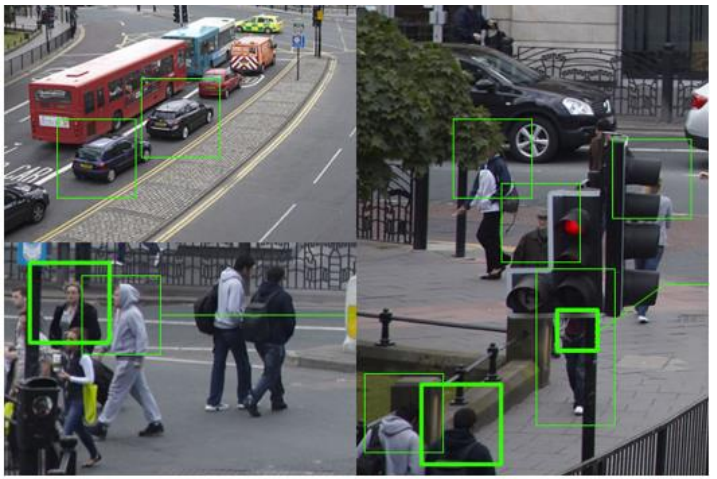


(T. Y. Lin et al-17)

OD application

- 자율주행 자동차
- CCTV Surveillance
- OCR
- Aerial Image 분석
- 신체인식
- 제조업
- 스포츠 경기 분석
- 무인점포





Text json

1종대형 1종보통 1종소형 특수(대형견인 소형견인,구난)2종보통 2종소형 원동기

면허번호

11-95-123456-72

성명

클로바

주민등록번호

990602-1234567

http://mrkim.cloudy.so/board_DJpE79/174208
https://hoya012.github.io/blog/Tutorials-of-Object-Detection-Using-Deep-Learning-the-application-of-object-detection/
http://www.itdaily.kr/news/articleView.html?idxno=202118

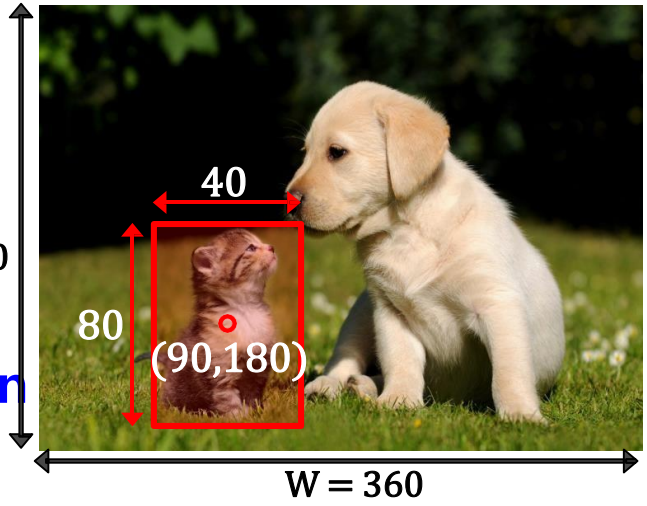
Object Detection

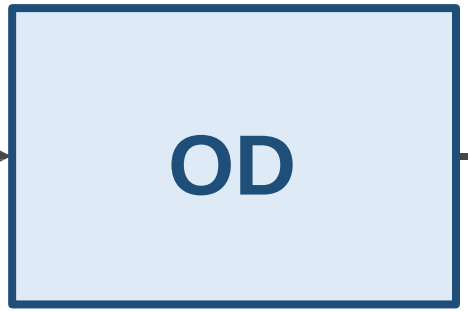
- **Task**
  - Find Bounding Box: **Regression**
  - Classify the Bounding Box: **Classification**

Object Detection

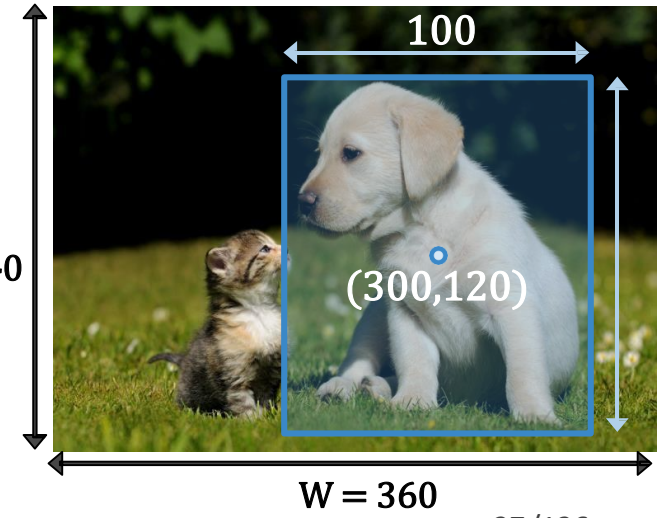

```
{
    "original_width": 600,
    "original_height": 403,
    "image_rotation": 0,
    "value": {
        "x": 24.666666666666668,
        "y": 49.62779156327544,
        "width": 31.666666666666668,
        "height": 39.702233250620345,
        "rotation": 0,
        "rectanglelabels": [
            "Airplane"
        ]
    },
    "id": "yytSY7KW36",
    "from_name": "label",
    "to_name": "image",
    "type": "rectanglelabels"
},
```


- **Task**
  - Find Bounding Box: **Regression**
    - (X, Y, W, H)
  - Classify the Bounding Box: **Classification**



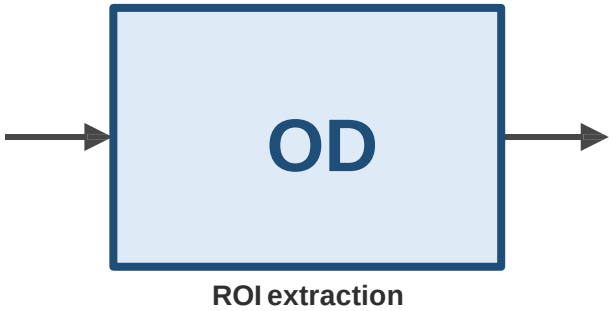


ROI extraction



Object Detection

- **Task**
  - Find Bounding Box: **Regression**
    - (X, Y, W, H)
    - Each image nees a different number of outputs



CAT: (x, y, w, h) **4**

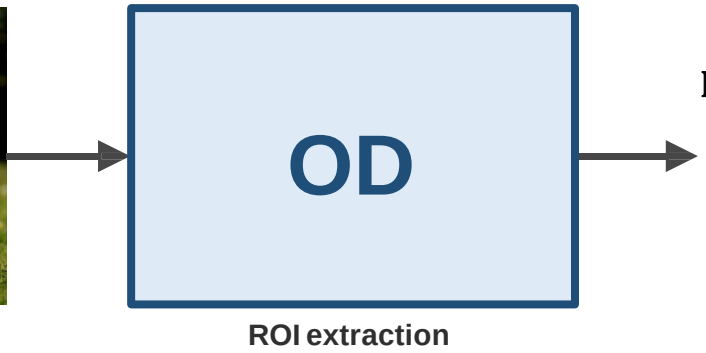
DOG: (x, y, w, h)
DOG: (x, y, w, h) **16**
CAT: (x, y, w, h)

DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

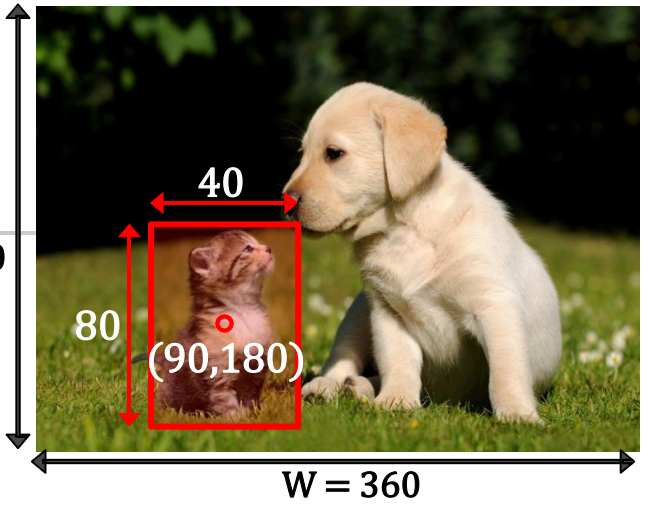
Many!

Object Detection

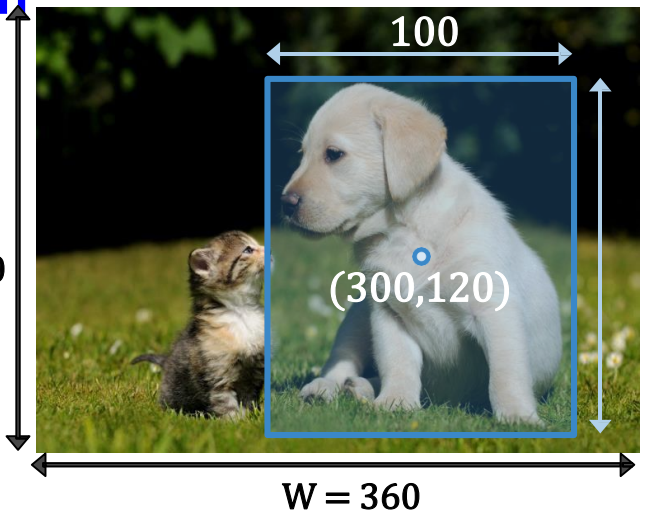
- **Task**
  - Find Bounding Box: **Regression**
    - (X, Y, W, H)
  - Classify the Bounding Box: **Classification**
    - Dog or Cat



ROI extraction



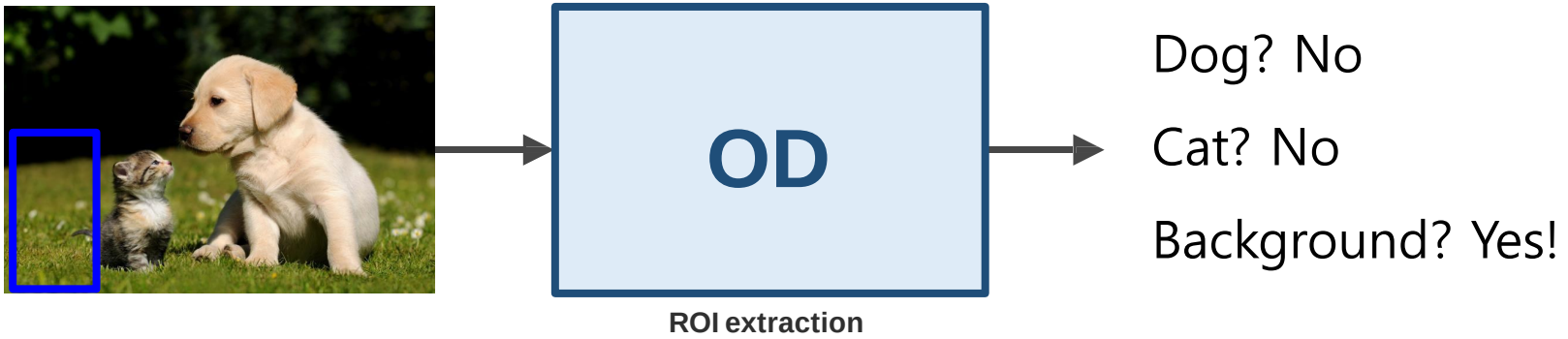
- Cat
  - Label: [1, 0]
  - Box: $[\frac{90}{360}, \frac{180}{240}, \frac{40}{360}, \frac{80}{240}]$



- Dog
  - Label: [0, 1]
  - Box: $[\frac{300}{360}, \frac{120}{240}, \frac{100}{360}, \frac{200}{240}]$

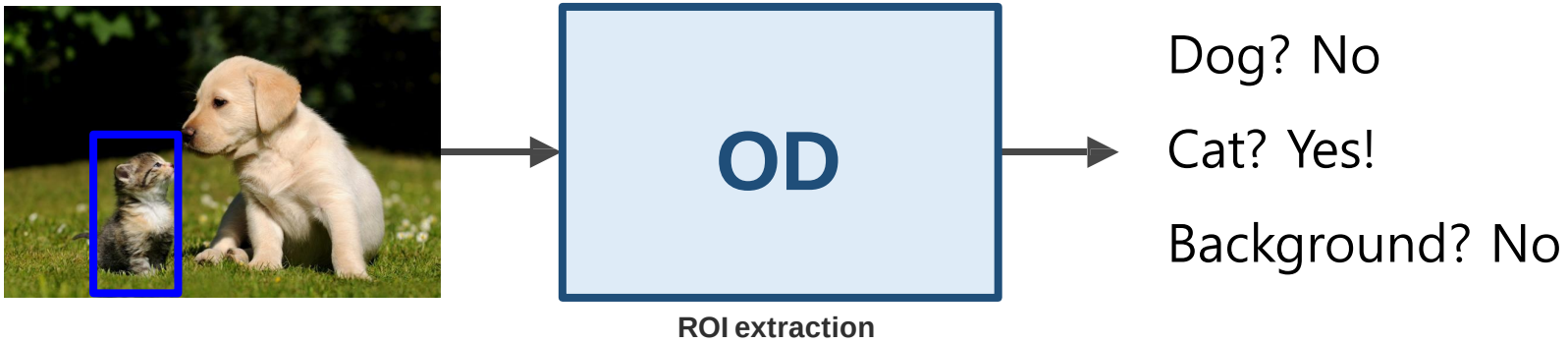
Object Detection

- **Task**
  - Find Bounding Box: **Regression**
    - (X, Y, W, H)
  - Classify the Bounding Box: **Classification**
    - Dog or Cat
    - Apply CNN to image



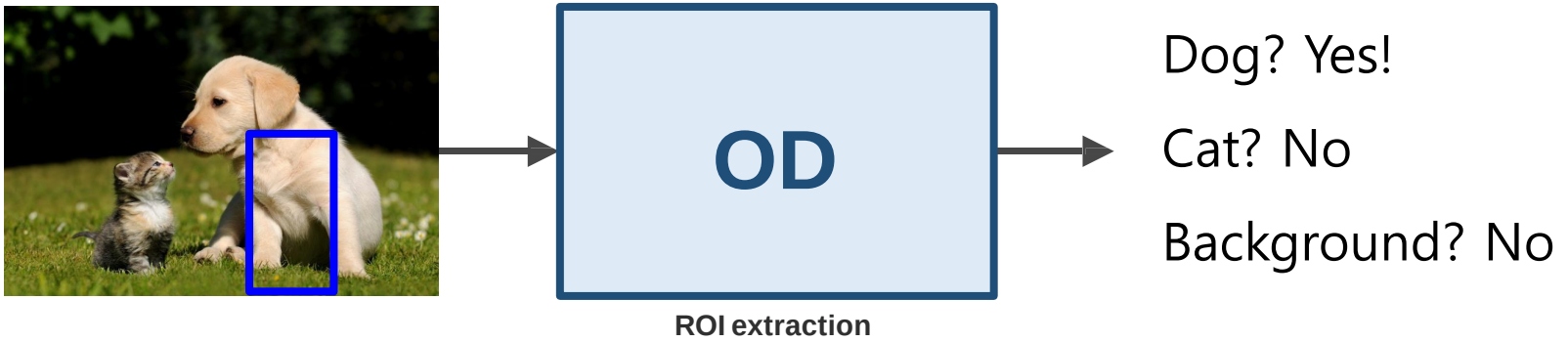
Object Detection

- **Task**
  - Find Bounding Box: **Regression**
    - (X, Y, W, H)
  - Classify the Bounding Box: **Classification**
    - Dog or Cat
    - Apply CNN to image



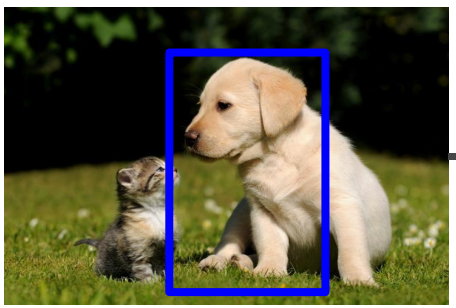
Object Detection

- **Task**
  - Find Bounding Box: **Regression**
    - (X, Y, W, H)
  - Classify the Bounding Box: **Classification**
    - Dog or Cat
    - Apply CNN to image



Object Detection

- Sliding window + Box regression + Classification

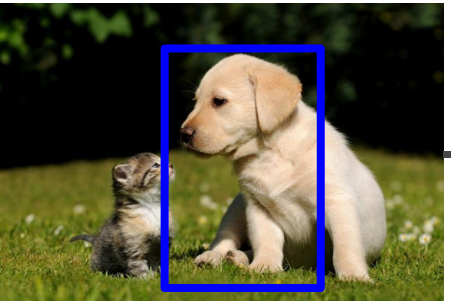


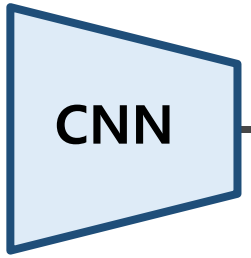
Dog? Yes!

Cat? No

Background? No

ROI extraction





Feature map
(5x5x1024)

Traditional Object Detection

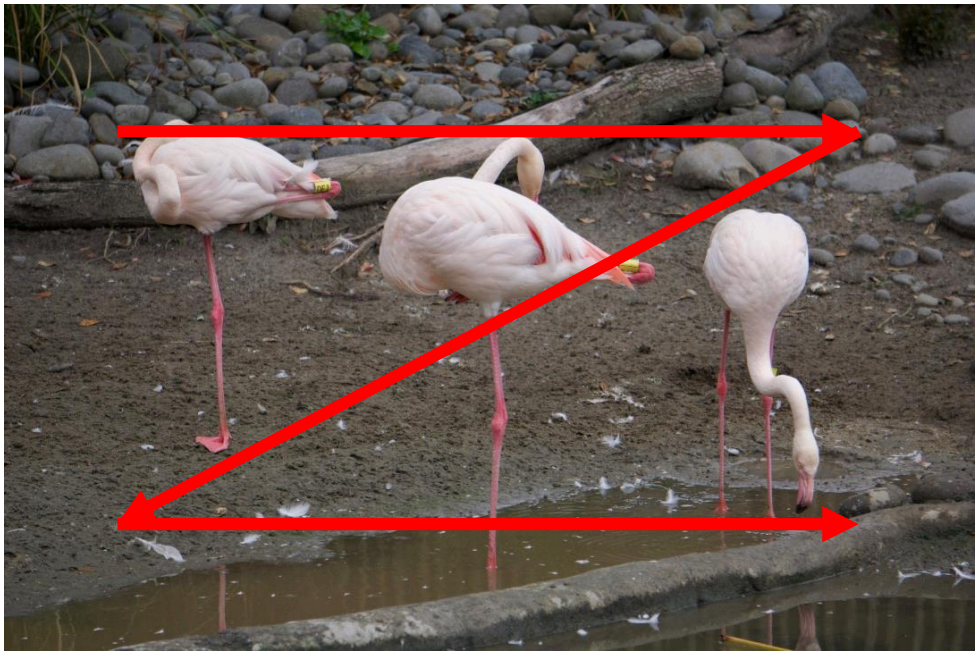
- **1. Templete matching + Sliding window**

Image

Template

Traditional Object Detection

- **1. Templete matching + Sliding window**

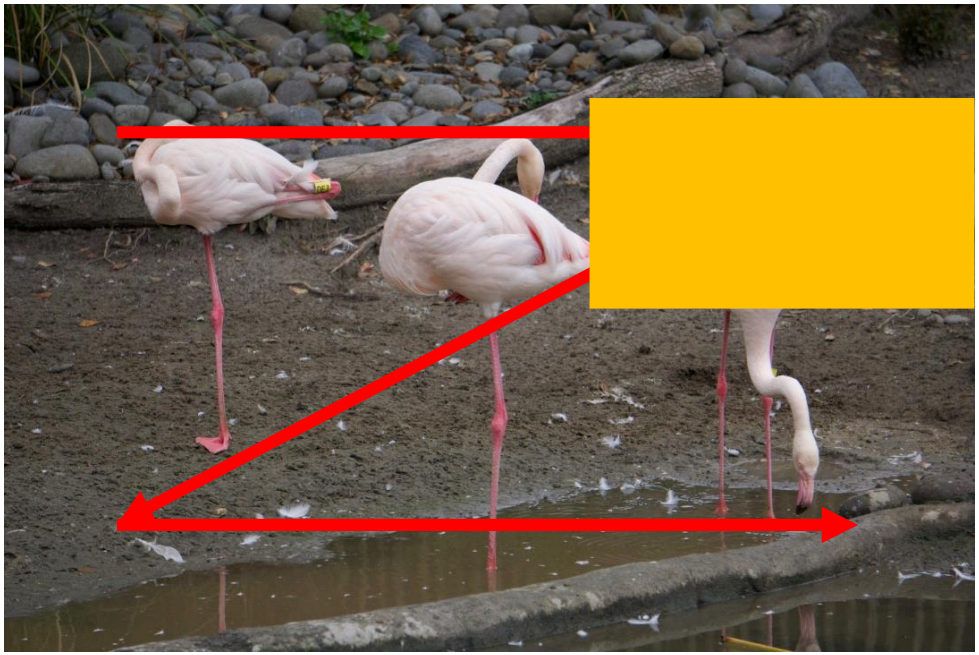


Image

Template

Traditional Object Detection

- **1. Templete matching + Sliding window**
  - Occlusion: need whole object
  - Detect given instance. Not a object class



Image

Template

Traditional Object Detection

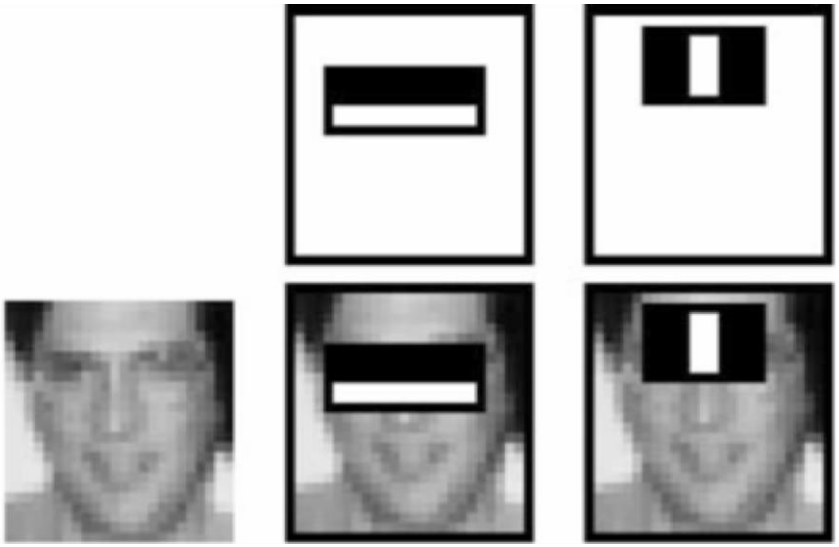
- **1. Templete matching + Sliding window**
  - Occlusion: need whole object
  - Detect given instance. Not a object class

Appearance
& shape changes

Pose changes

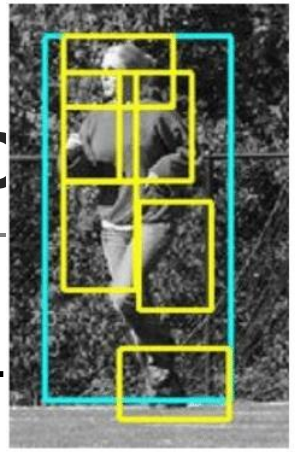
Traditional Object Detection

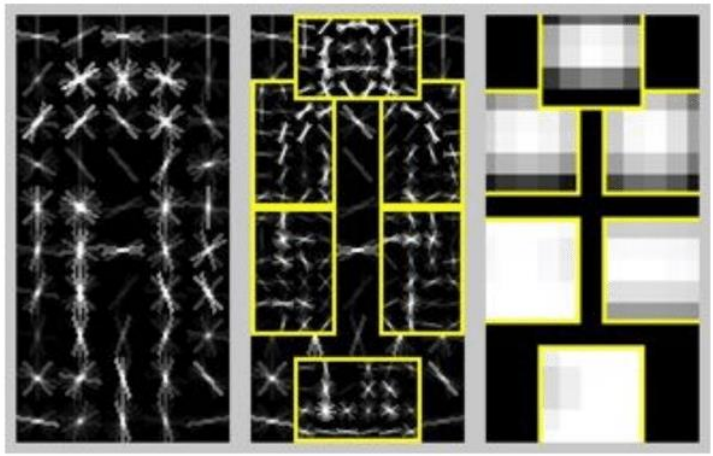
- **2. Feature extraction + classification**
  - Learning multiple weak learners to build a strong classifier
  - Make many small decisions and combine them for a stronger final decision



Haar features

Traditional Object D

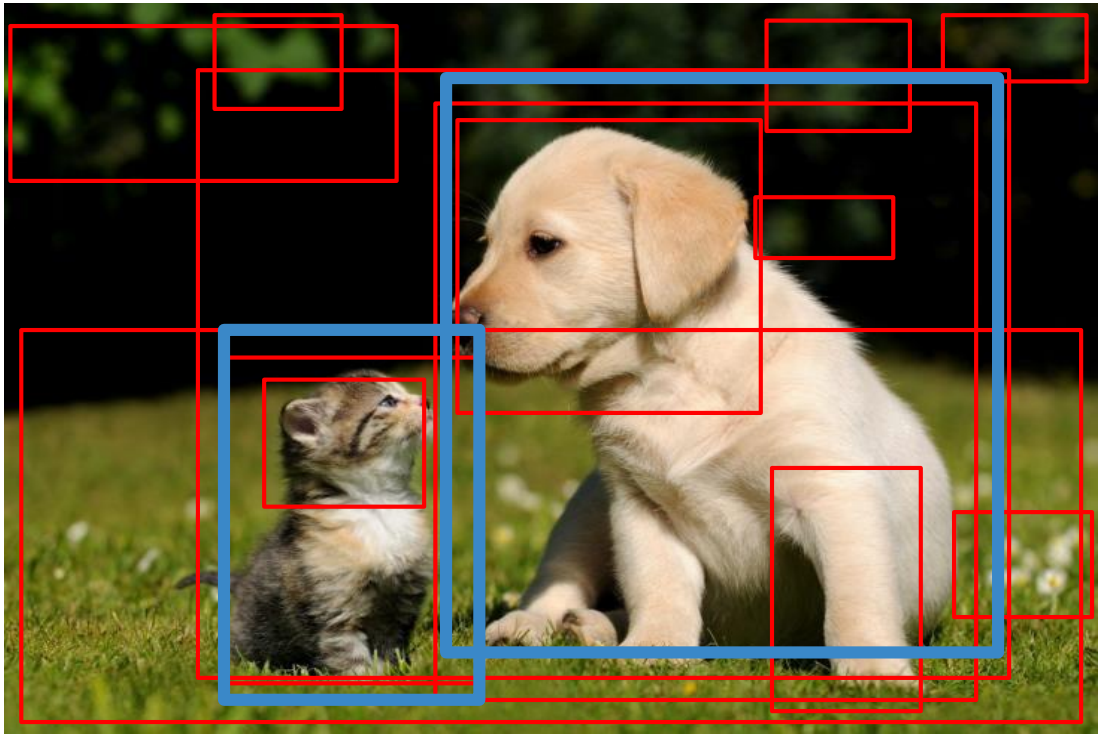


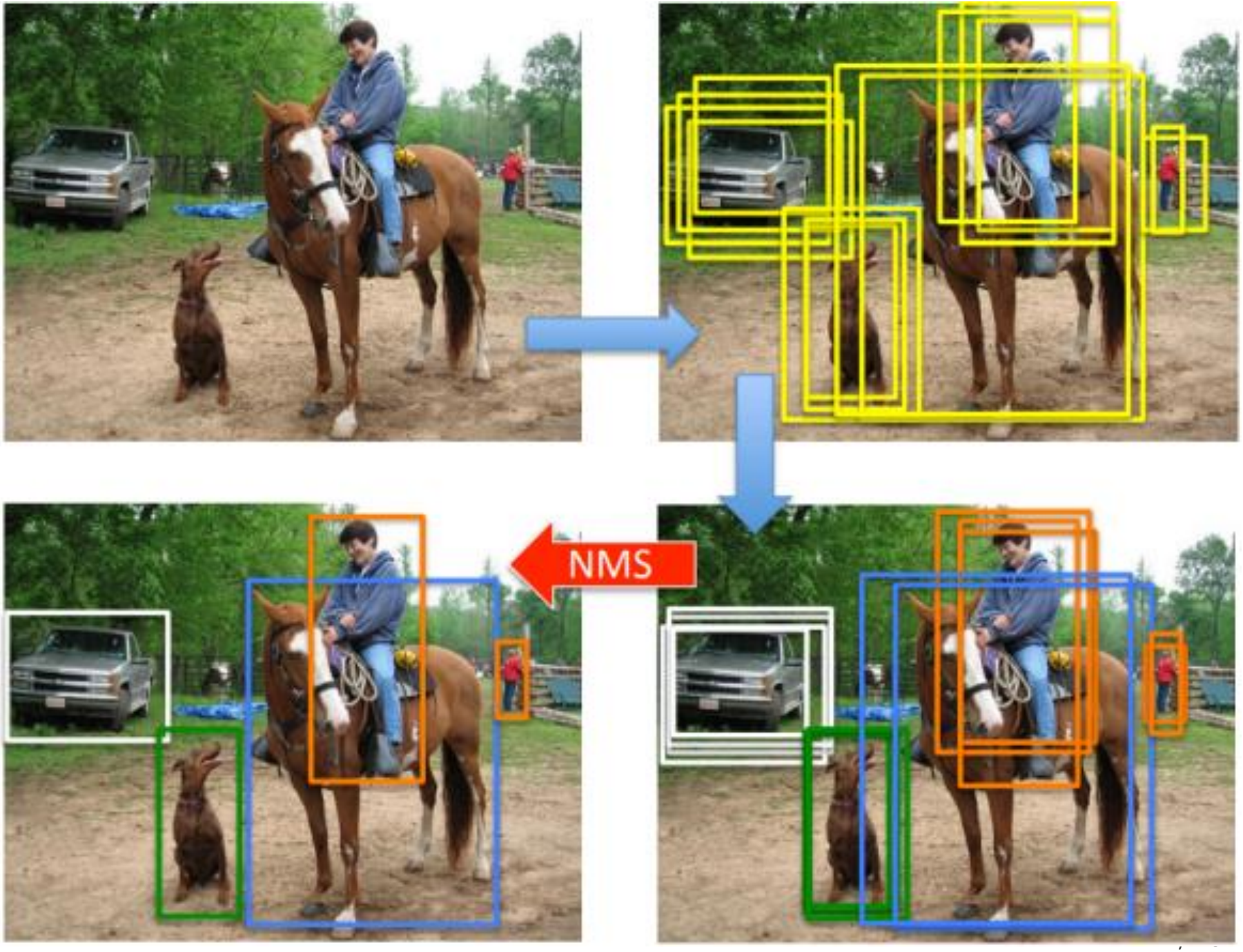


- **2. Feature extraction +**
  - Step 1: Select your Haar-like features
  - Step 2: Integral image for fast feature evaluation
    - I can evaluate which parts of the image have highest cross-correlation with my feature (template)
  - Step 3: AdaBoost for to find weak learner
    - I cannot possibly evaluate all features at test time for all image locations
    - Learn the best set of weak learners
    - Our final classifier is the linear combination of all weak learners

Object Proposal

- **NMS (Non-Maximum Suppression)**
  - We need a generic, class-agnostic objectness measure
  - Many candidate object proposals (ROI, Region of Interest)

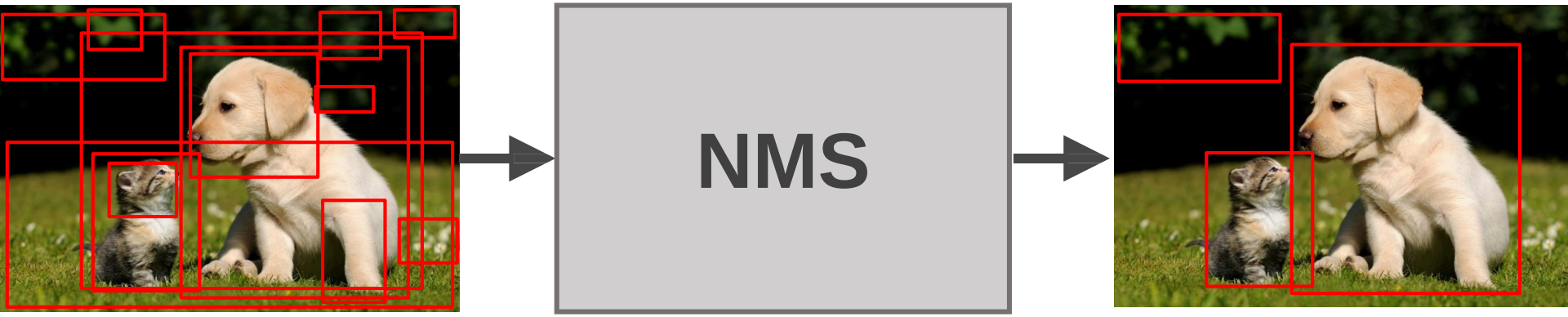




https://medium.com/@pedroazevedo6/object-detection-state-of-the-art-2022-ad750e0f6003

Object Proposal

- **NMS (Non-Maximum Suppression)**
  - We need a generic, class-agnostic objectness measure
  - Many candidate object proposals (ROI, Region of Interest)
  - Many boxes trying to explain one object
  - We need a method to keep only the **"best" boxes**



Object Proposal

- **NMS**

Algorithm 1 Non-Max Suppression

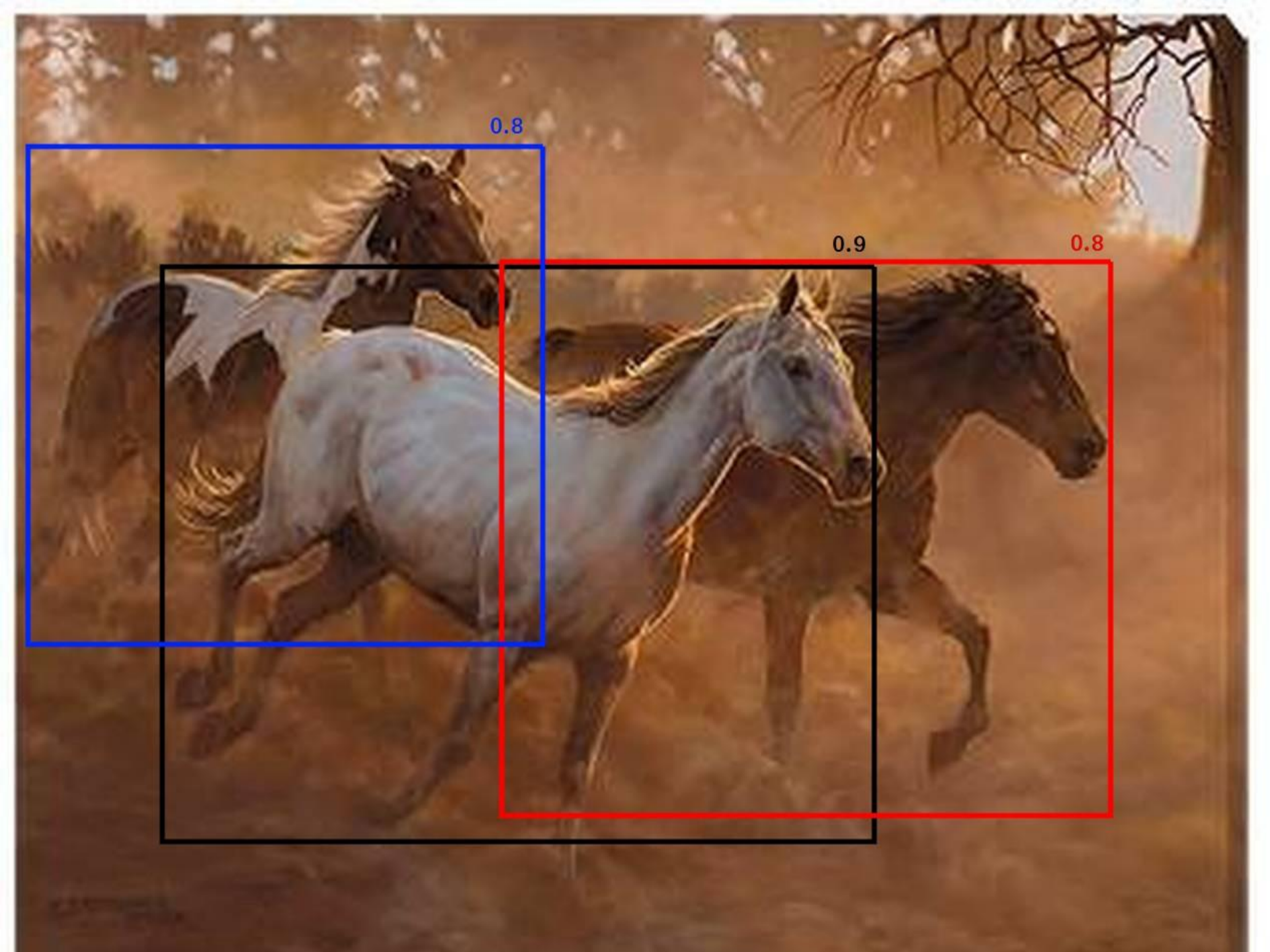
```
procedure NMS(B, c)
    B_nms ← ∅                                   Initialize empty set
    for b_i ∈ B do                              => Iterate over all the boxes
        discard ← False                         Take boolean variable and set it as false. This variable indicates whether b(i) should be kept or discarded
        for b_j ∈ B do                          Start another loop to compare with b(i)
            if same(b_i, b_j) > λ_nms then      If both boxes having same IOU
                if score(c, b_j) > score(c, b_i) then
                    discard ← True              Compare the scores. If score of b(i) is less than that of b(j), b(i) should be discarded, so set the flag to True.
        if not discard then                     Once b(i) is compared with all other boxes and still the discarded flag is False, then b(i) should be considered. So add it to the final list.
            B_nms ← B_nms ∪ b_i
    return B_nms                                Do the same procedure for remaining boxes and return the final list
```

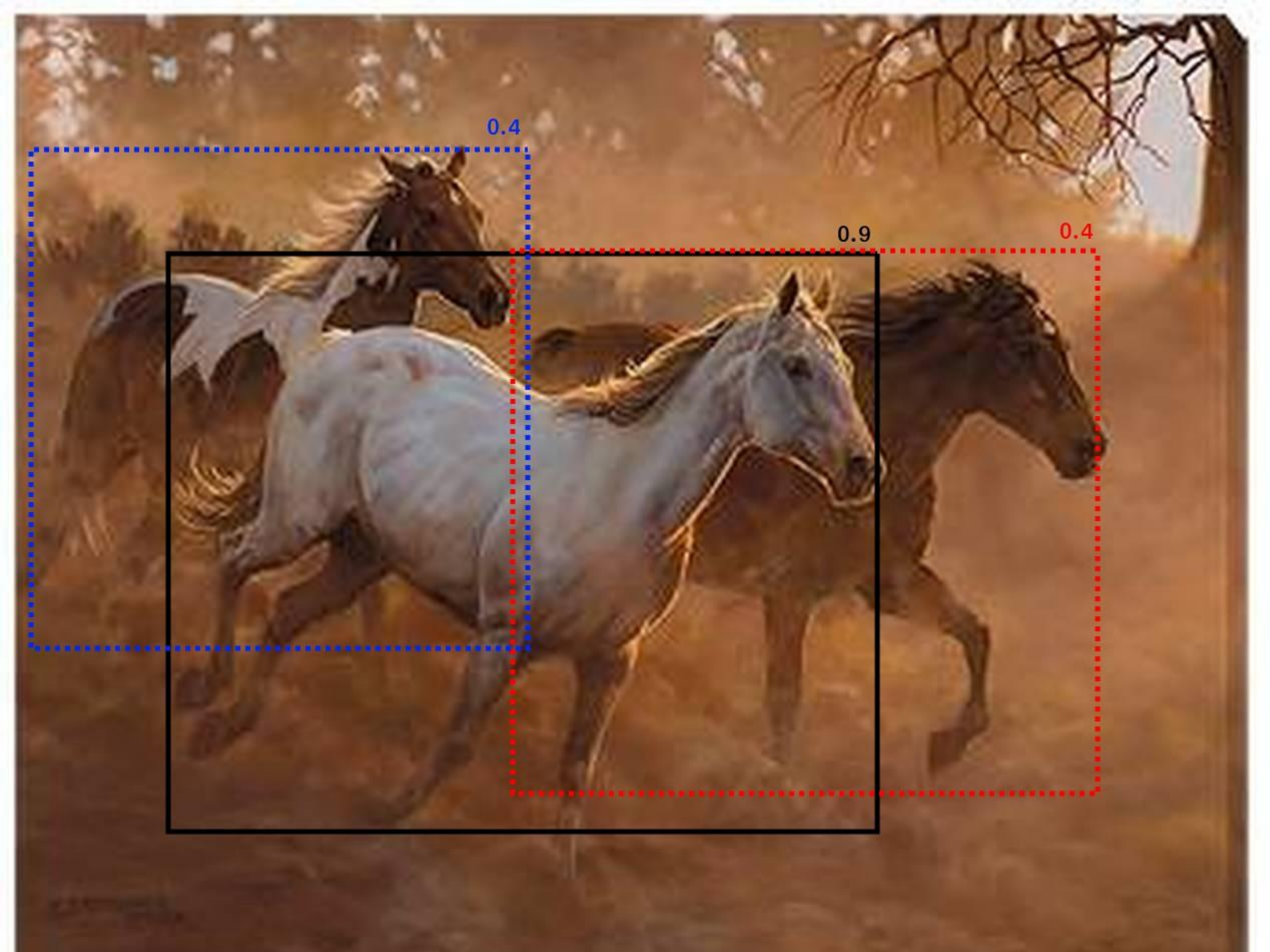
Start with anchor box i

For another box j

If they overlap

Discard box i if the score is lower than the score of j


0.8
0.9
0.8


0.4
0.9
0.4

Region Overlap

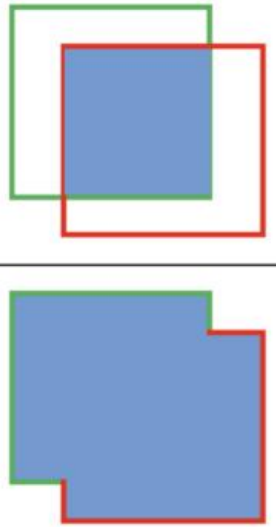
- IoU (Intersection Over Union)
- Precision
- Recall
- Average Precision
- FPS

B_p = 실제 (*Gound Truth*)

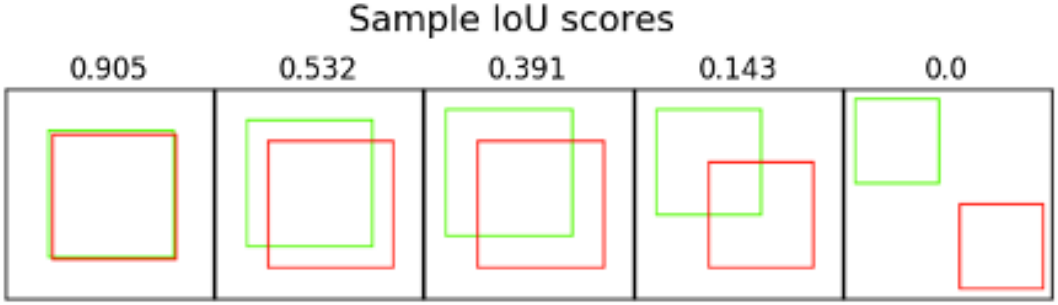
B_{gt} = 예측 (*Prediction*)

$$IOU = \frac{area\ of\ overlap}{area\ of\ union} = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$$

$$= \frac{\text{실제} \cap \text{예측 중복 영역}}{\text{실제} \cup \text{예측 전체 영역}}$$



waytoliah.com



https://hoya012.github.io/blog/Tutorials-of-Object-Detection-Using-Deep-Learning-how-to-measure-performance-of-object-detection/

Object Proposal

- **NMS**

Algorithm 1 Non-Max Suppression

```
procedure NMS(B, c)
    B_nms ← ∅                                   Initialize empty set
    for b_i ∈ B do                              => Iterate over all the boxes
        discard ← False                         Take boolean variable and set it as false. This variable indicates whether b(i) should be kept or discarded
        for b_j ∈ B do                          Start another loop to compare with b(i)
            if same(b_i, b_j) > λ_nms then      If both boxes having same IOU
                if score(c, b_j) > score(c, b_i) then
                    discard ← True              Compare the scores. If score of b(i) is less than that of b(j), b(i) should be discarded, so set the flag to True.
        if not discard then                     Once b(i) is compared with all other boxes and still the discarded flag is False, then b(i) should be considered. So add it to the final list.
            B_nms ← B_nms ∪ b_i
    return B_nms                                Do the same procedure for remaining boxes and return the final list
```

Start with anchor box i

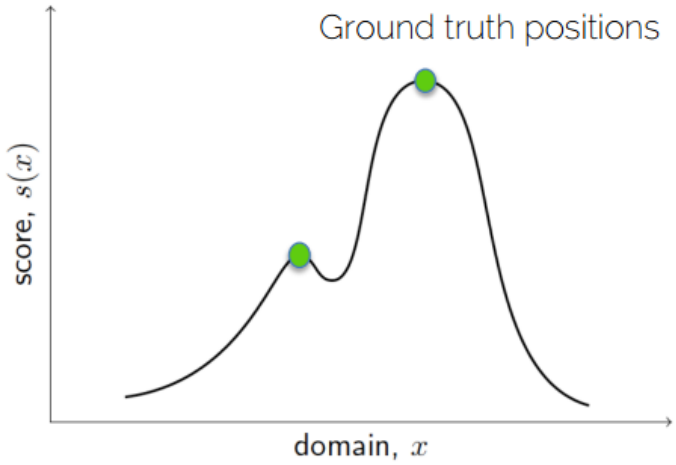
For another box j

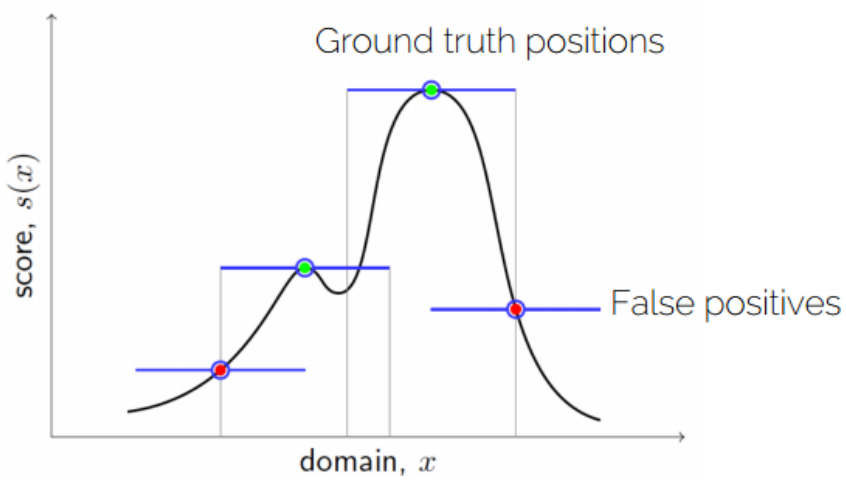
If they overlap

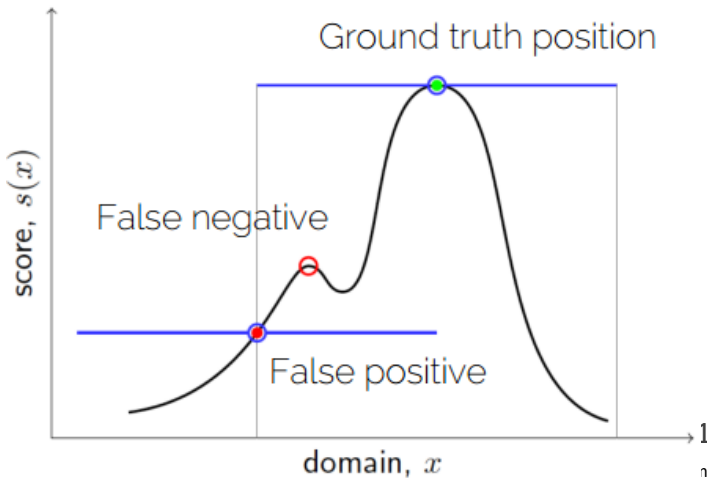
Discard box i if the score is lower than the score of j

Object Proposal

- **NMS**







OD performance

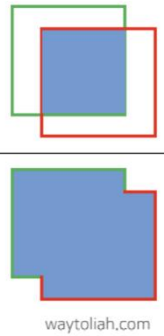
- Confidence: 검출한 것에 대해 얼마나 확신이 있는지
- IoU (Intersection Over Union)
- Precision
- Recall
- AP(Average Precision): Precision-Recall 그래프에서 아래 면적
- mAP(mean Average Precision): 각 클래스 당 AP의 평균
- mAP@0.5: mAP의 평균을 IoU>0.5로 구한 값
- mAP@0.5:0.95: 0.5~0.95 IoU threshold 값을 0.05
- 간격으로 측정한 mAP의 평균값
- FPS: 초당 몇 frame 연산을 할수 있는지

B_p = 실제 (*Gound Truth*)

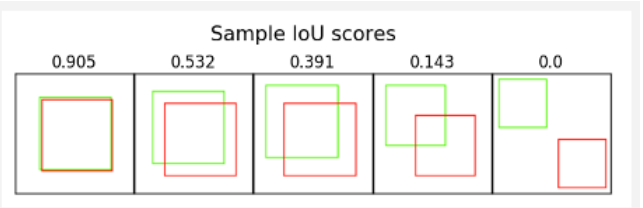
B_{gt} = 예측 (*Prediction*)

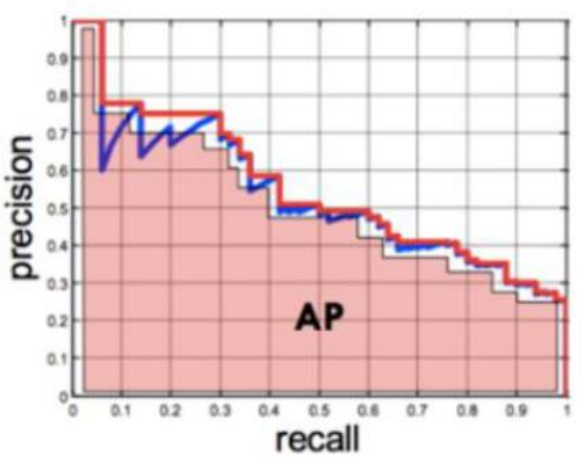
$$IOU = \frac{area\ of\ overlap}{area\ of\ union} = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$$

$$= \frac{\text{실제} \cap \text{예측 중복 영역}}{\text{실제} \cup \text{예측 전체 영역}}$$



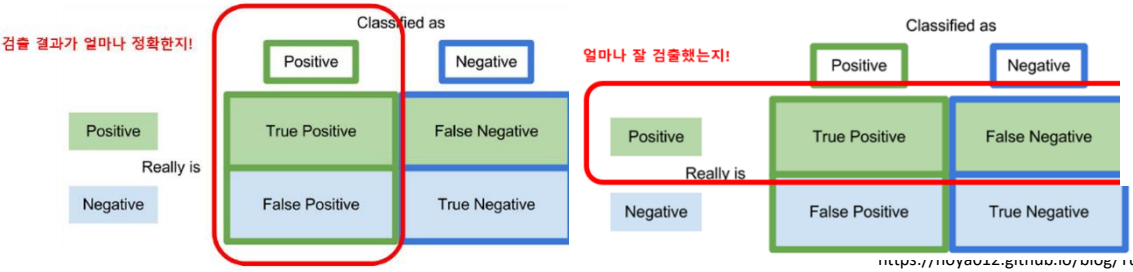
waytoliah.com





Precision = TP / (TP + FP)

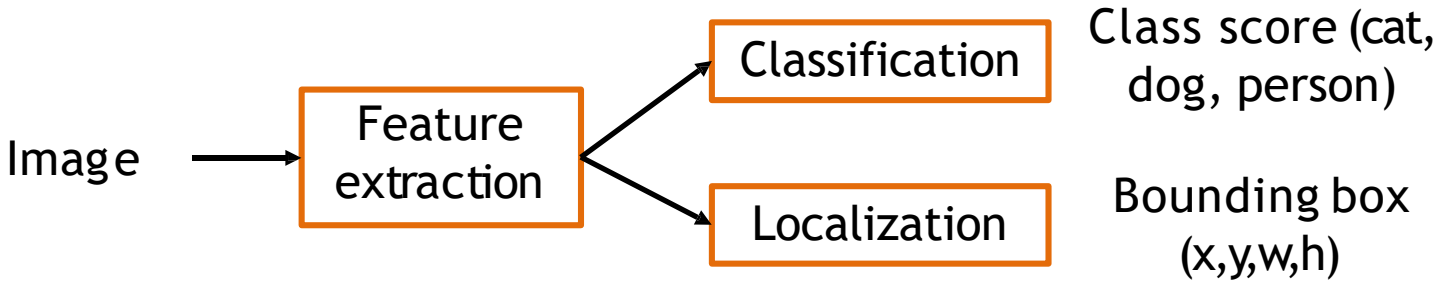
Recall = TP / (TP + FN)



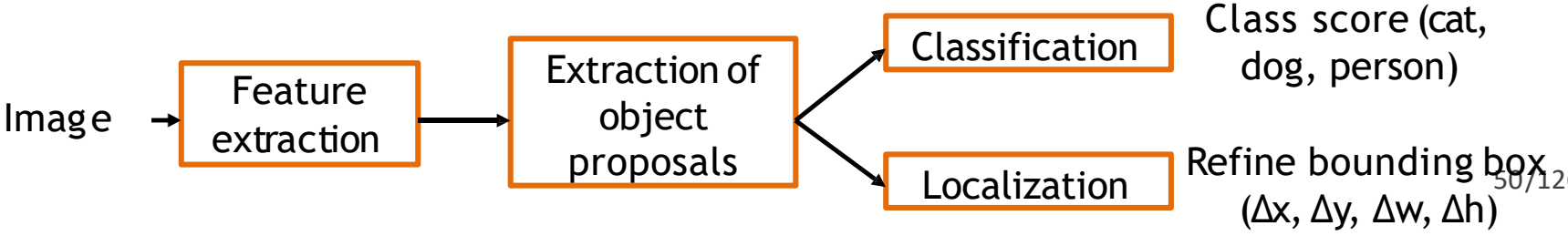
https://hoya012.github.io/blog/Tutorials-of-Object-Detection-Using-Deep-Learning-how-to-measure-performance-of-object-detection/

Object Detection Type

- **1-stage detector**
  - 추론 속도가 빨라 real-time task
  - YOLO, SSD, RetinaNet, CenterNet, CornerNet, ExtremeNet



- **2-stage detector: Faster R-CNN**
  - 성능이 좋지만 느림
  - R-CNN, Fast R-CNN, Faster R-CNN, SPP-Net, R-FCN, FPN



2-stage detector

- **1) R-CNN**

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

2-stage detector

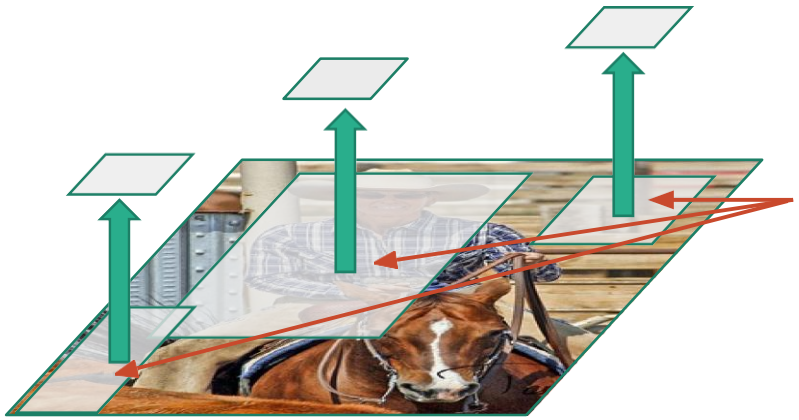
- **1) R-CNN**

RoI (Regions of Interest) ~2k

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

2-stage detector

- **1) R-CNN**



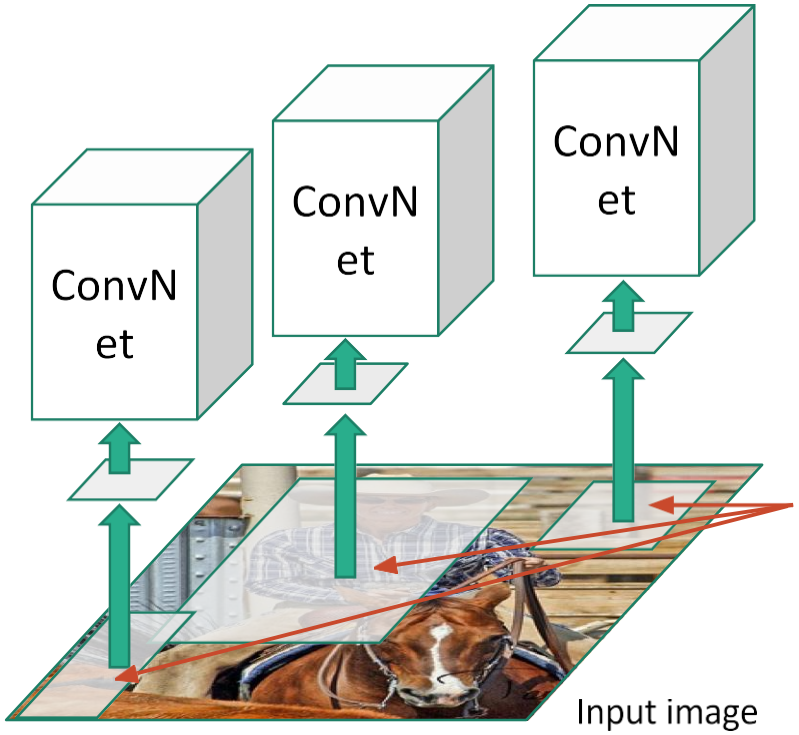
Warped image region

RoI (Regions of Interest) ~2k

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

2-stage detector

- **1) R-CNN**



Forward each region through ConvNet

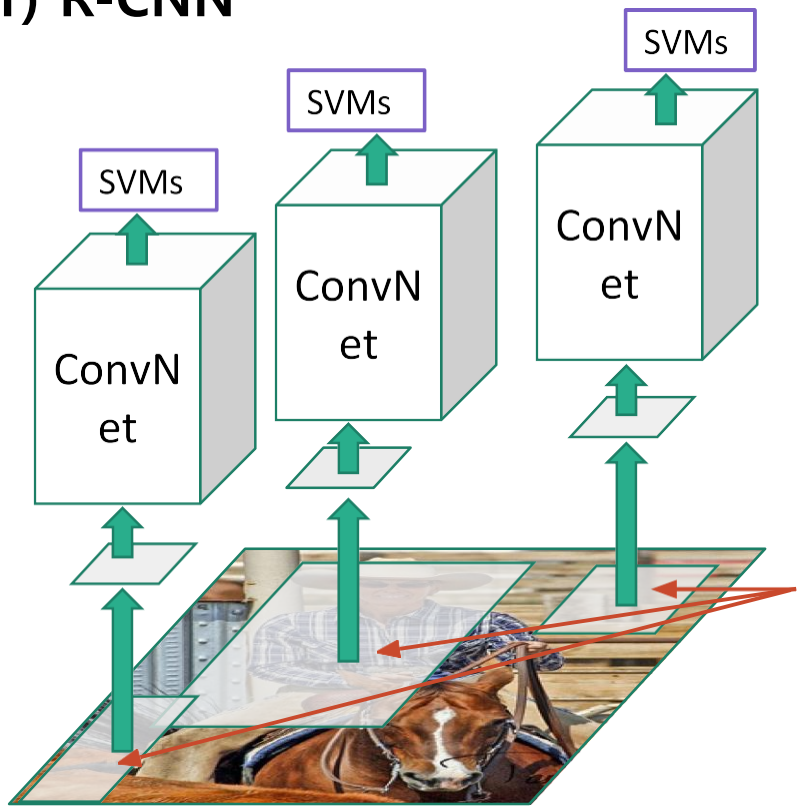
Warped image region

RoI (Regions of Interest) ~2k

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

2-stage detector

- **1) R-CNN**



Classify regions

Forward each region through ConvNet

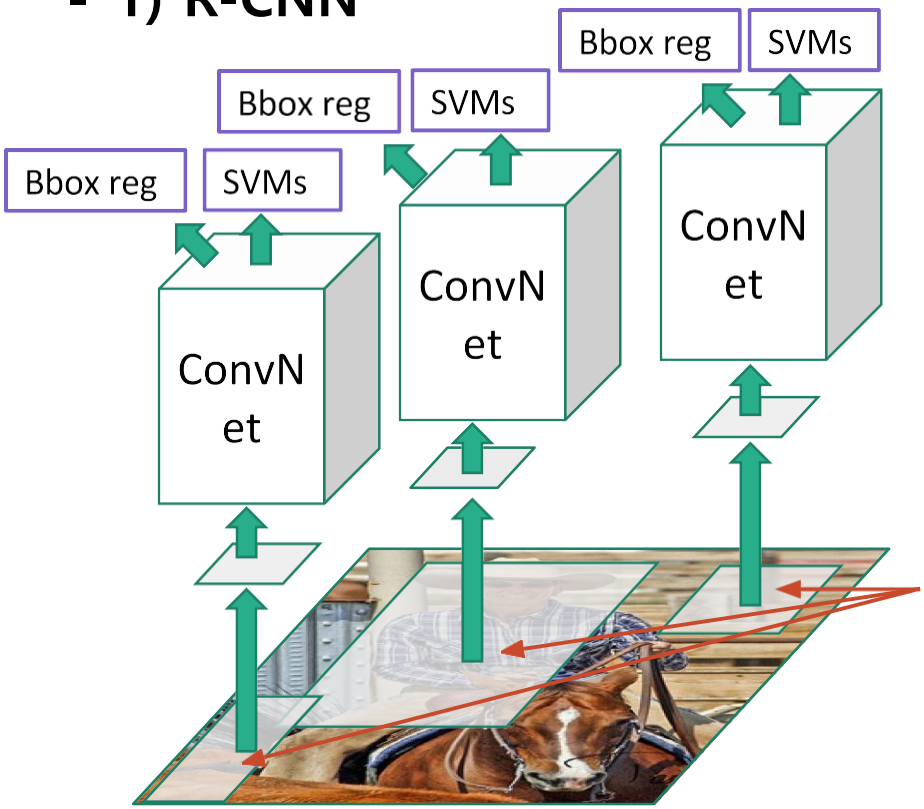
Warped image region

RoI (Regions of Interest) ~2k

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

2-stage detector

- **1) R-CNN**



Classify regions & Regression for Bbox

Forward each region through ConvNet

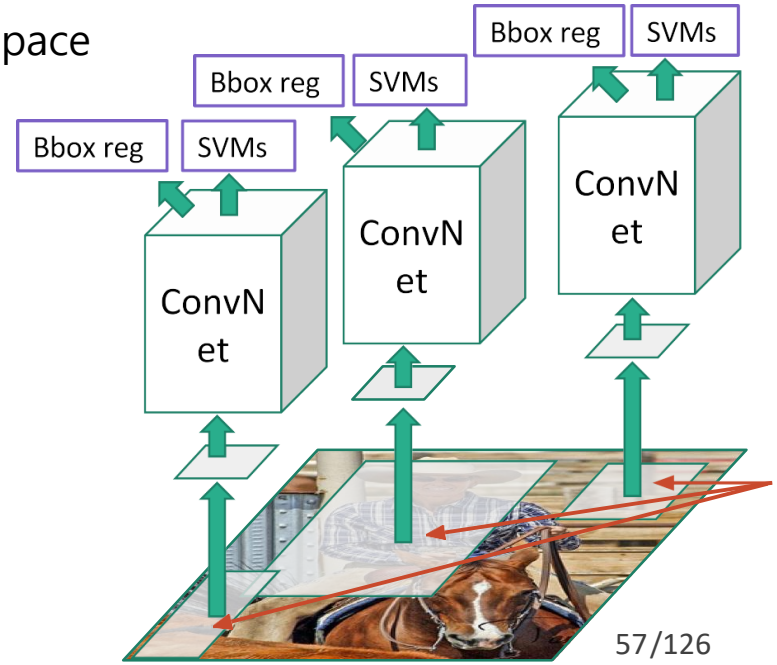
Warped image region

RoI (Regions of Interest) ~2k

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

2-stage detector

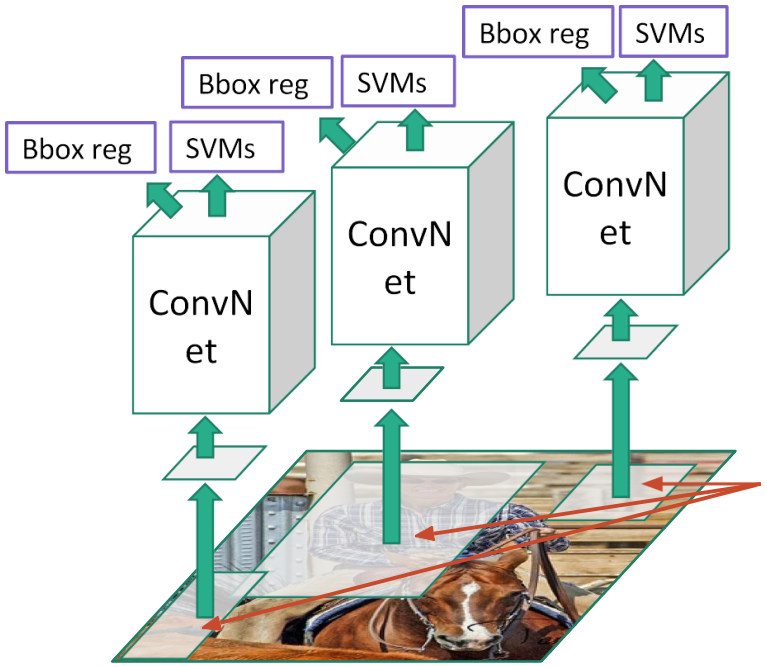
- **1) R-CNN**
  - Training objectives
    - Fine-tune network with softmax classifier (log loss)
    - Train post-hoc linear SVMs (hinge loss)
    - Train post-hoc bounding-box regressions (least squares)
  - Training is slow (84h), takes a lot of disk space
  - Inference (detection) is **slow**
    - 47s / image with VGG16



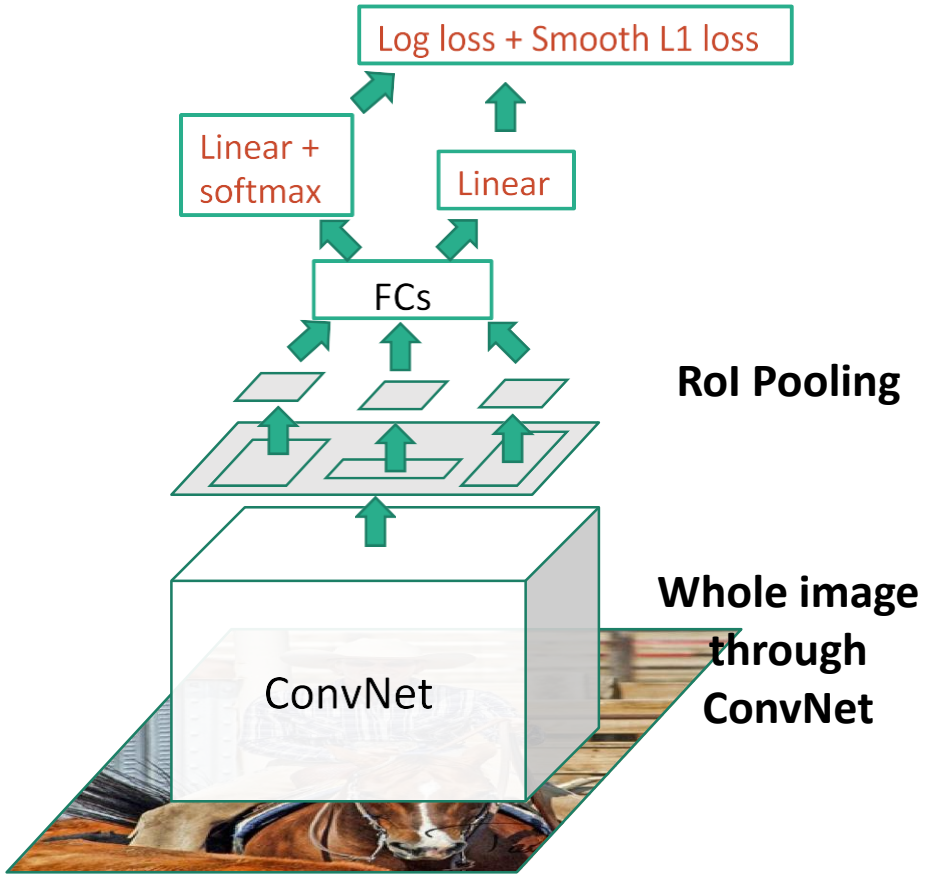
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

2-stage detector

- 2) Faster R-CNN



R-CNN

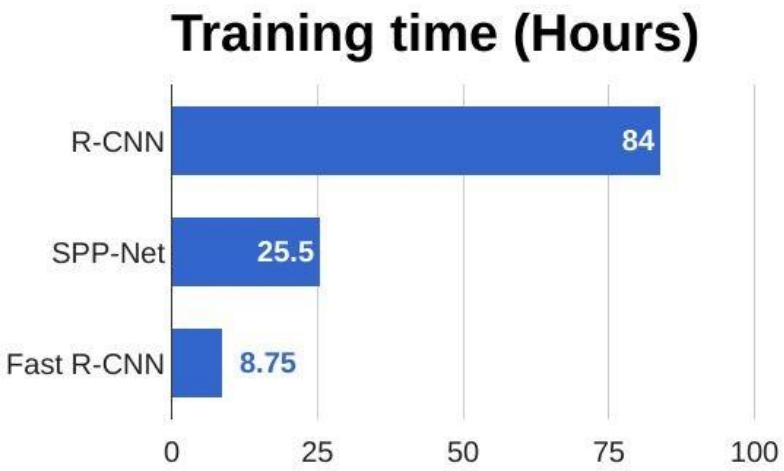


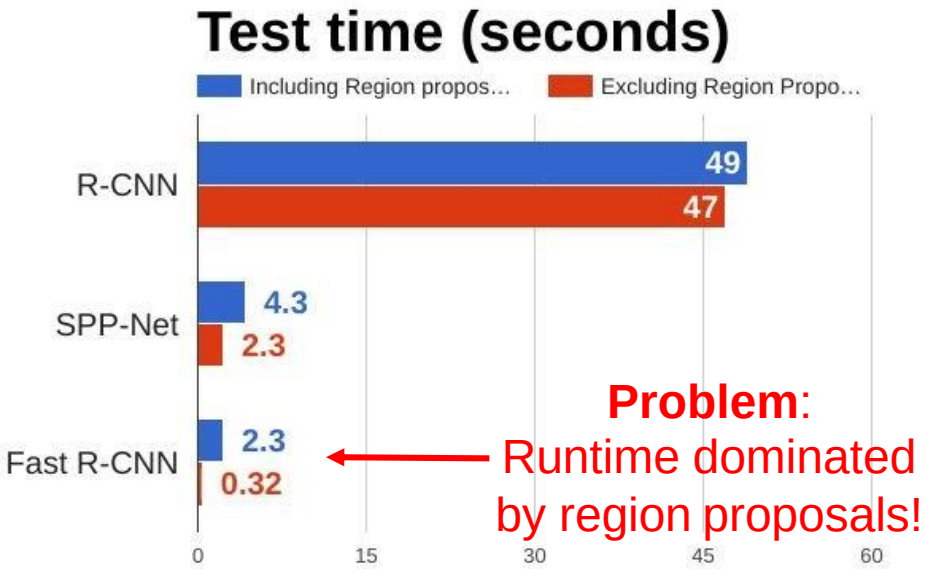
Faster R-CNN

Girshick, "Fast R-CNN", ICCV 2015.

2-stage detector

- **2) Faster R-CNN**

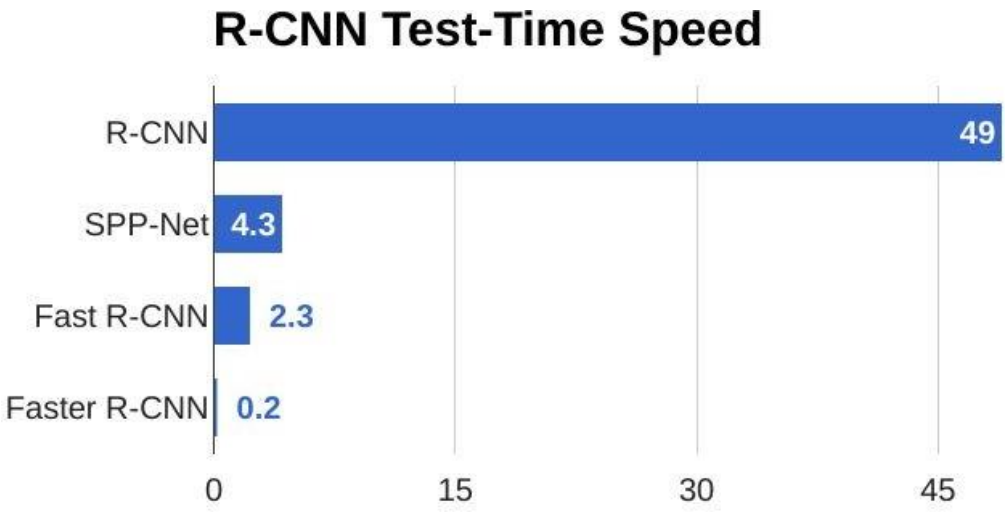


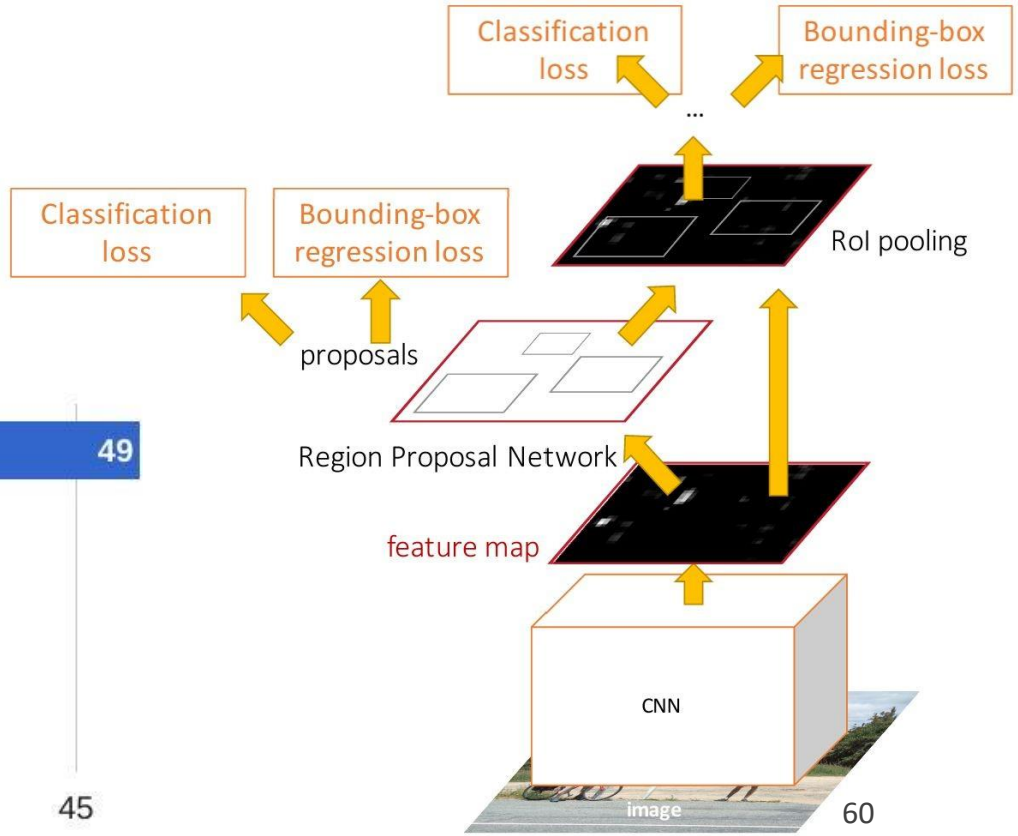


Girshick, "Fast R-CNN", ICCV 2015.

2-stage detector

- **3) Faster R-CNN**
  - Make CNN do proposals!
  - Insert Region Proposal Network (RPN) to predict proposals from features
    - RPN classification (object/non-object)
    - RPN regression (anchor -> proposal)
    - Fast R-CNN classification (type of object)
    - Fast R-CNN regression (proposal -> box)



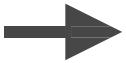


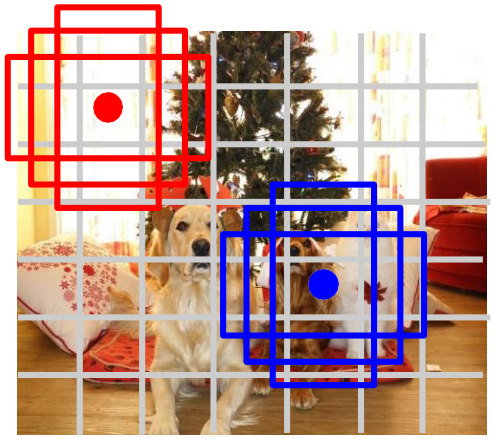
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

1-stage detector

- **1) YOLO, SSD in reach grid cell**
  - Regress from each base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
  - Predict scores for each of C classes (including background as a class)
  - Output: 7x7x(5xB+C)

Input image 3 x H x W





Divide image into grid 7x7
base boxes centered at
each grid cell, B=3

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

YOLO and SSD

- **PROS:**
  - **Very fast**
  - **End-to-end trainable and fully convolutional**
  - **SSD detects more objects than YOLO**
- **CONS:**
  - **Performance is not as good as two-stage detectors**
  - **Difficulty with small objects**

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

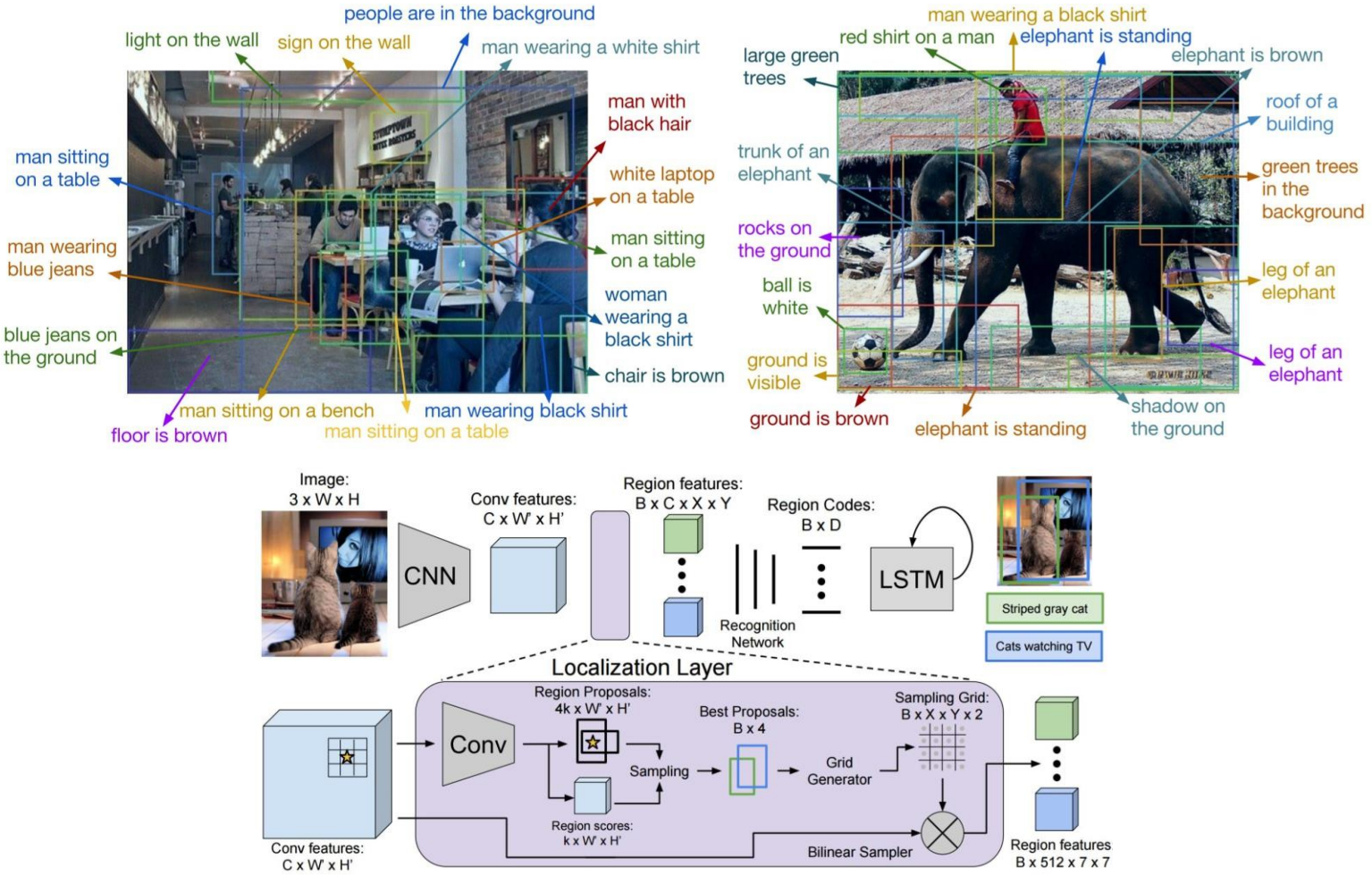
OD variables

- **Base Network**
  - VGG, ResNet, Inception v1/2/3, MobileNet
- **Architecture**
  - YOLO, SSD, FaSTER R-CNN
- **Size**
  - Input image size, Region proposal
- **Takeaways**
  - Faster R-CNN is slower but more accurate
  - YOLO/SSD is faster but not as accurate

Girshick, "Fast R-CNN", ICCV 2015.

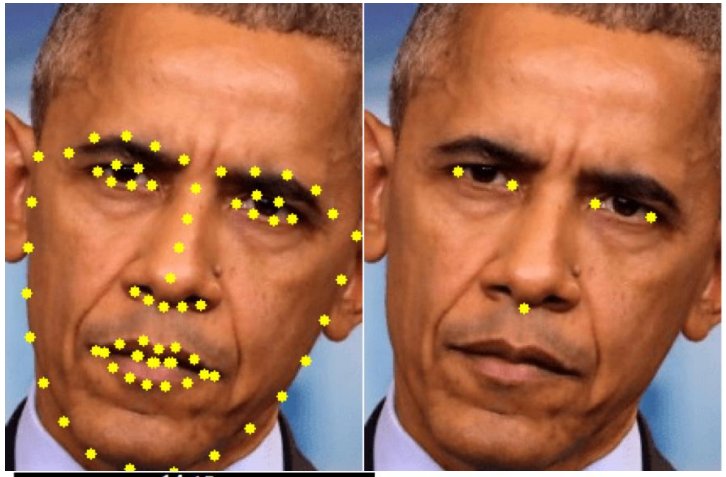
OD Application

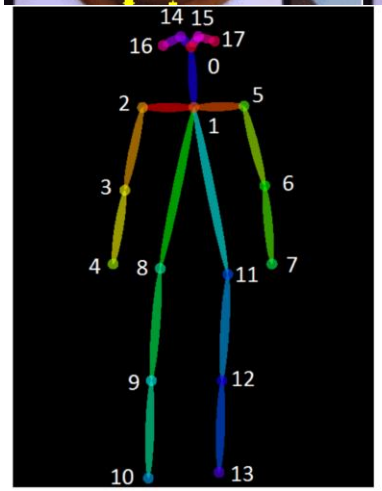
- **Object Detection + Captioning**

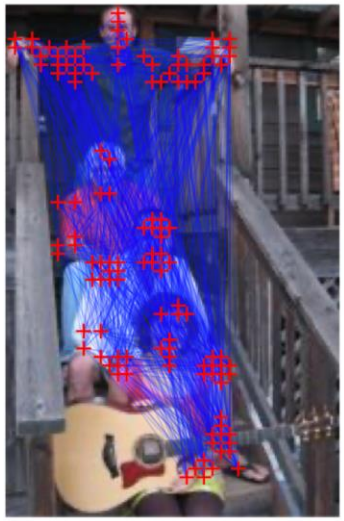


OD Application

- **Detection beyond 2D boxes**

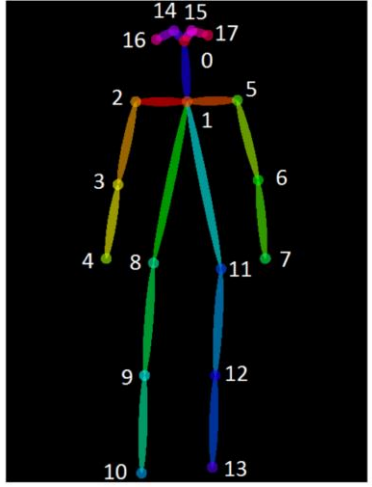






OD Application

- **Pose Estimation data**
  - **COCO dataset**
    - 12만(19GB)
    - 17 key points
    - 여러 사람에 대한 데이터셋
  - **MPII dataset**
    - 2.5만
    - 16 key points
    - 한명에 대한 데이터셋

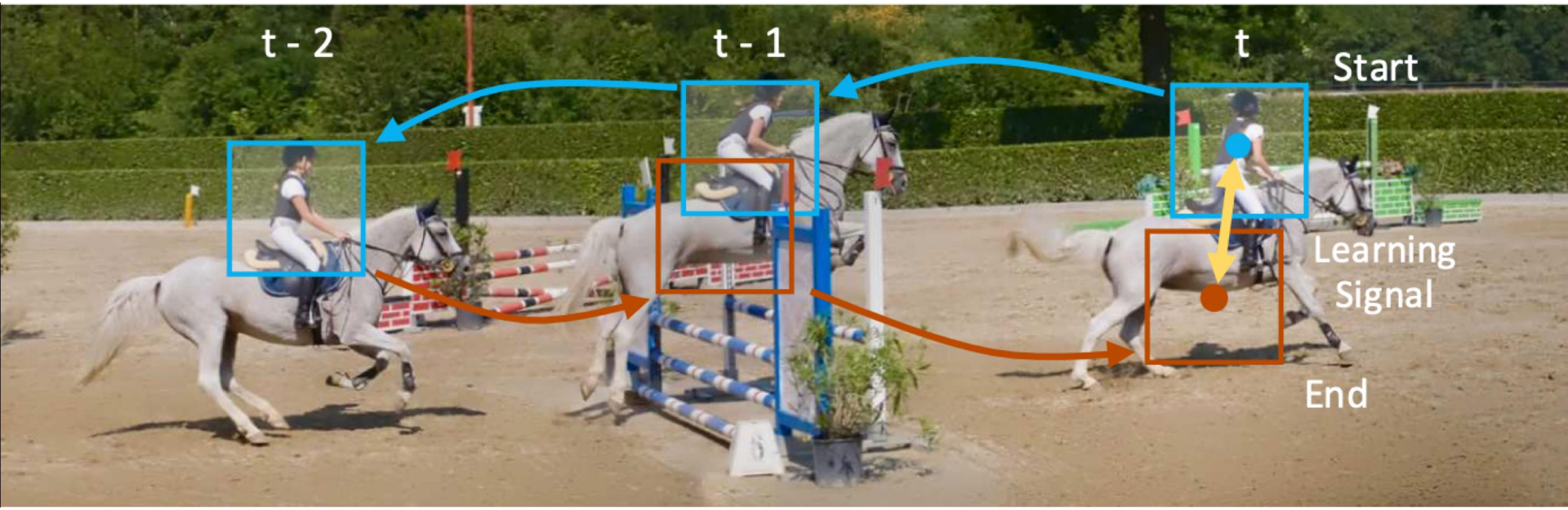


COCO KeyPoints

MPII KeyPoints

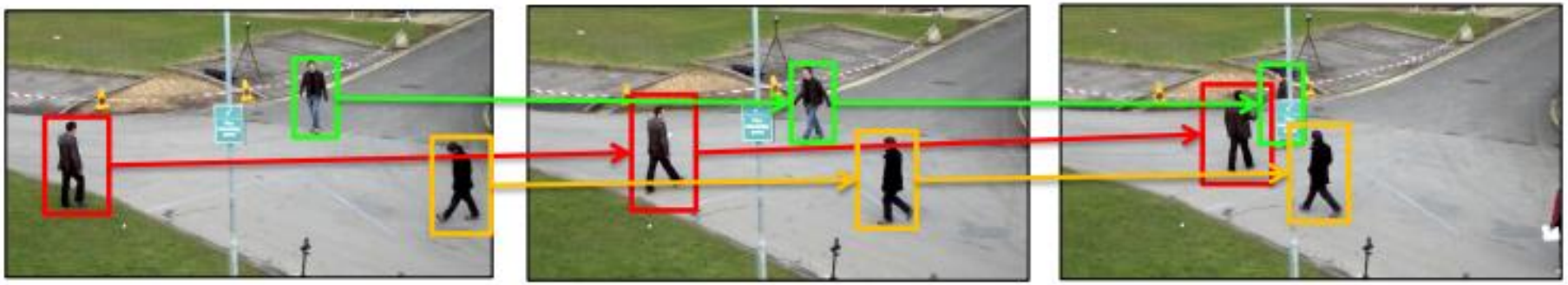
OD Application

- **Object Tracking(Single)**



OD Application

- **Object Tracking(Multi)**
  - **Graph based method**



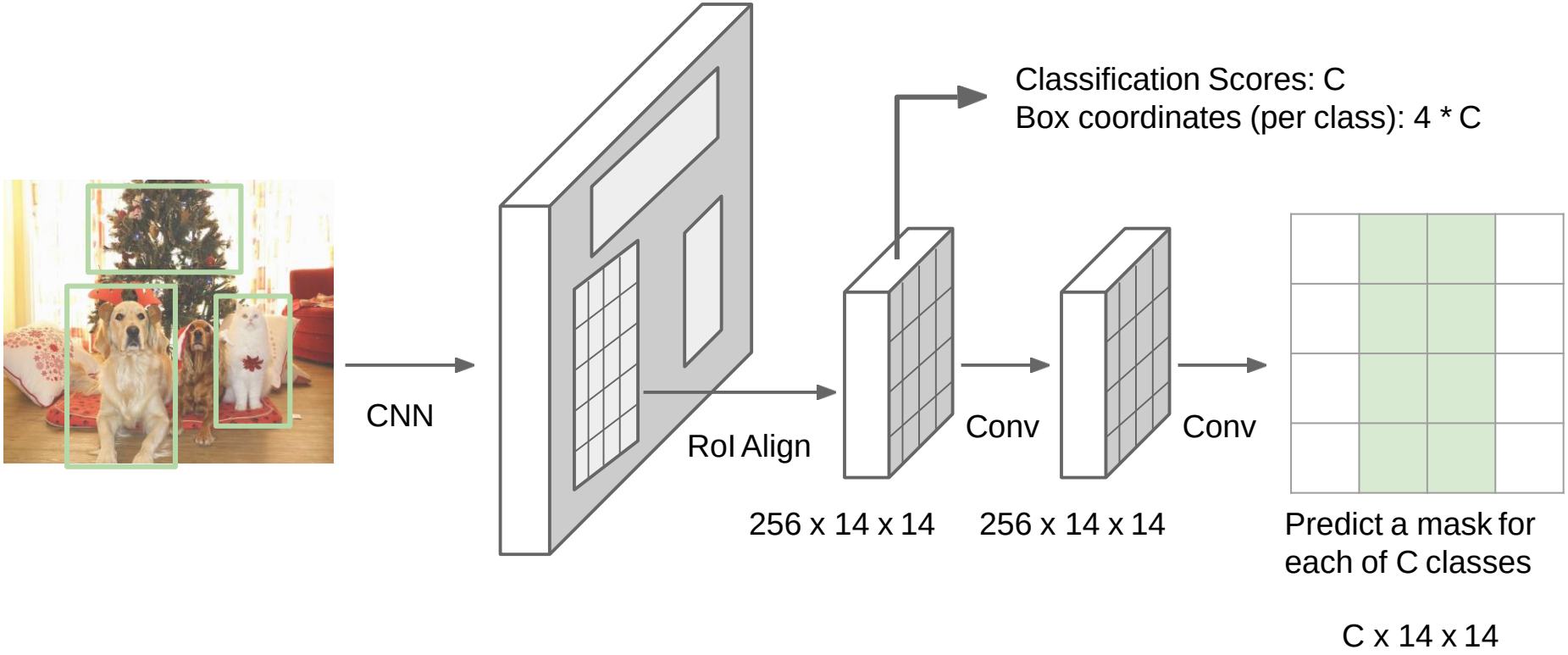
OD Application

- **Object Tracking(Multi)**
  - **DeepSort**

DeepSort Demo

OD Application

- **Segmentation**



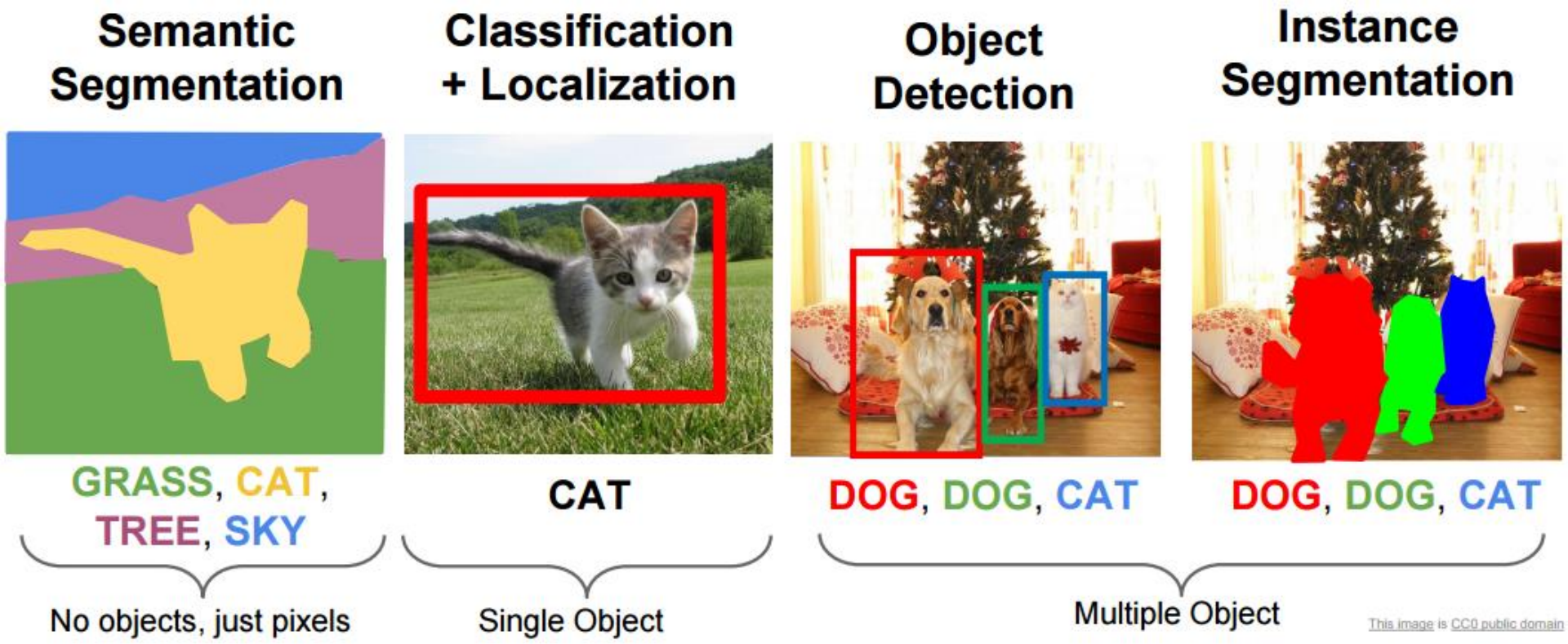
He et al, “Mask R-CNN”, arXiv 2017

OD Application

- **Segmentation**

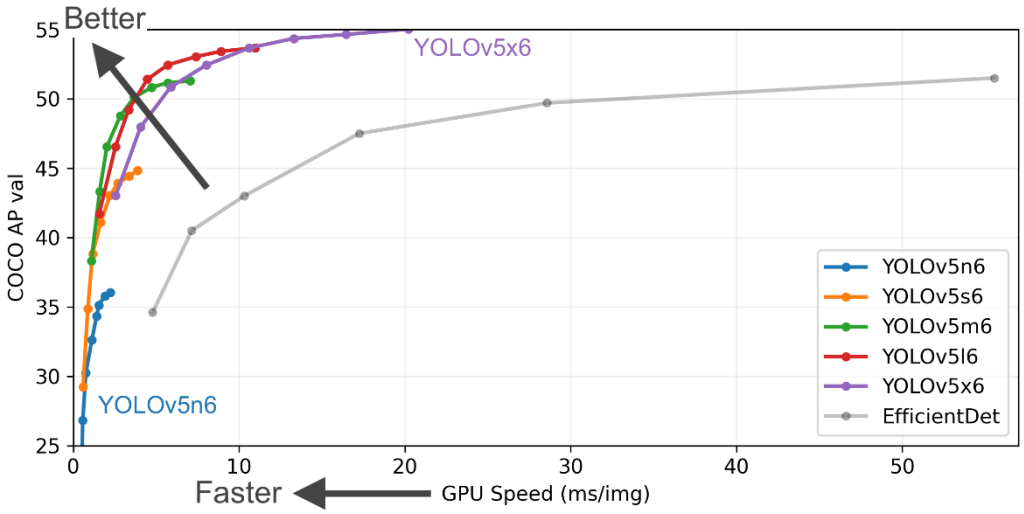
He et al, “Mask R-CNN”, arXiv 2017

Recap


Semantic Segmentation
Classification + Localization
Object Detection
Instance Segmentation
GRASS, CAT, TREE, SKY
CAT
DOG, DOG, CAT
DOG, DOG, CAT
No objects, just pixels
Single Object
Multiple Object
This image is CC0 public domain

OD Practice

- **Yolov5[URL]**
  - https://github.com/ultralytics/yolov5



Model	size (pixels)	mAP^{val} 0.5:0.95	mAP^{val} 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	**45**	**6.3**	**0.6**	**1.9**	**4.5**
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6 + TTA	1280 1536	55.0 **55.8**	72.7 **72.7**	3136 -	26.2 -	19.4 -	140.7 -	209.8 -

OD Practice

- **Yolov5[URL]**
  - **Colab 실습: https://bit.ly/3CECu5n**
  - https://github.com/airobotlab/lecture_5_yolov5
  - 1_train_yolo_colab_220509.ipynb
    - mydaya 폴더 안에 데이터셋 등록(image&label)

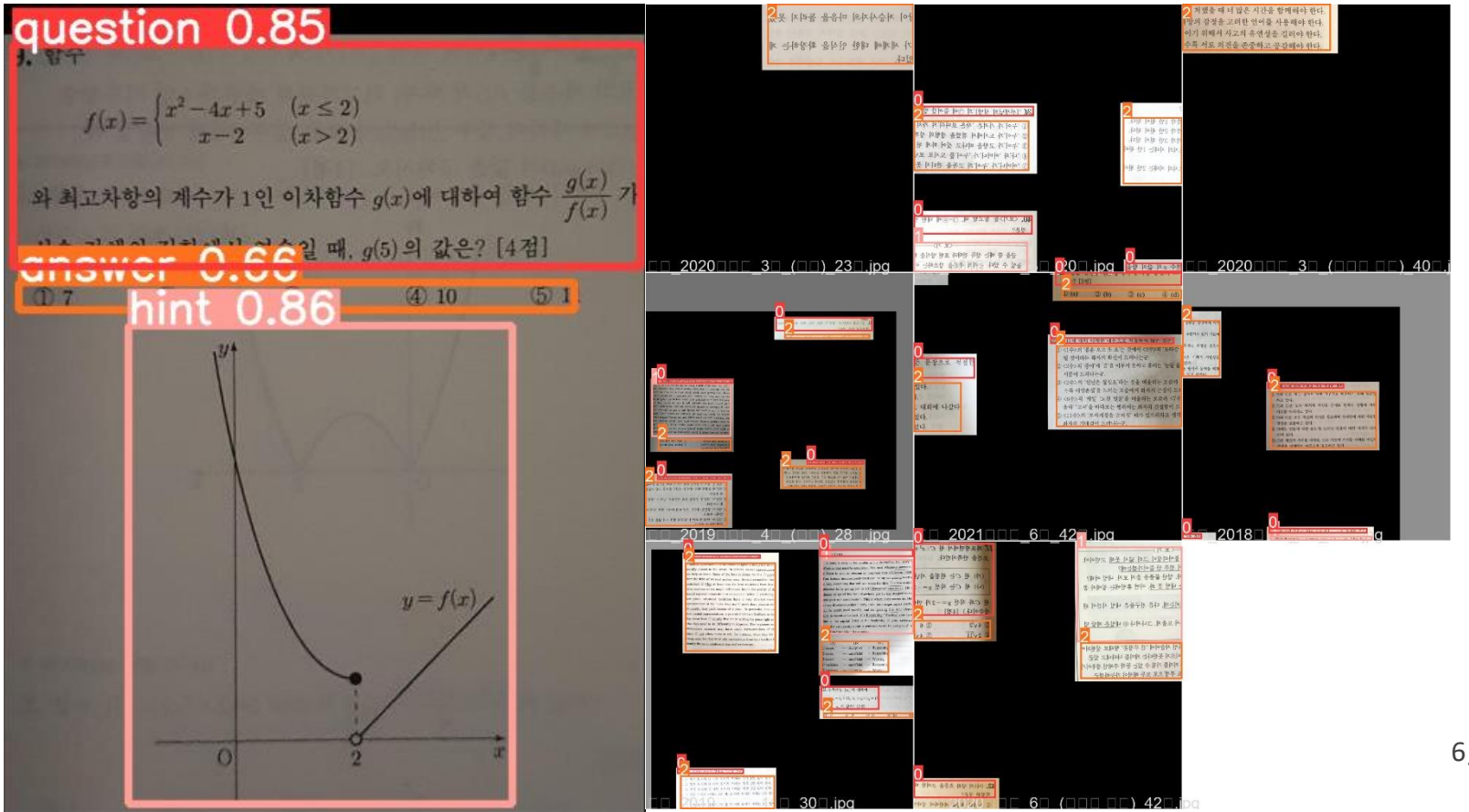
```
# 데이터 경로와 class 지정, 중요!!
import yaml

data_yaml = dict(
    train = './mydata/images/test/',
    val = './mydata/images/small/',
    nc = 3,
    names = ['question', 'hint', 'answer']
)

# Note that I am creating the file in the yolov5/data/ directory.
with open('data/mydata.yaml', 'w') as outfile:
    yaml.dump(data_yaml, outfile, default_flow_style=True)
```

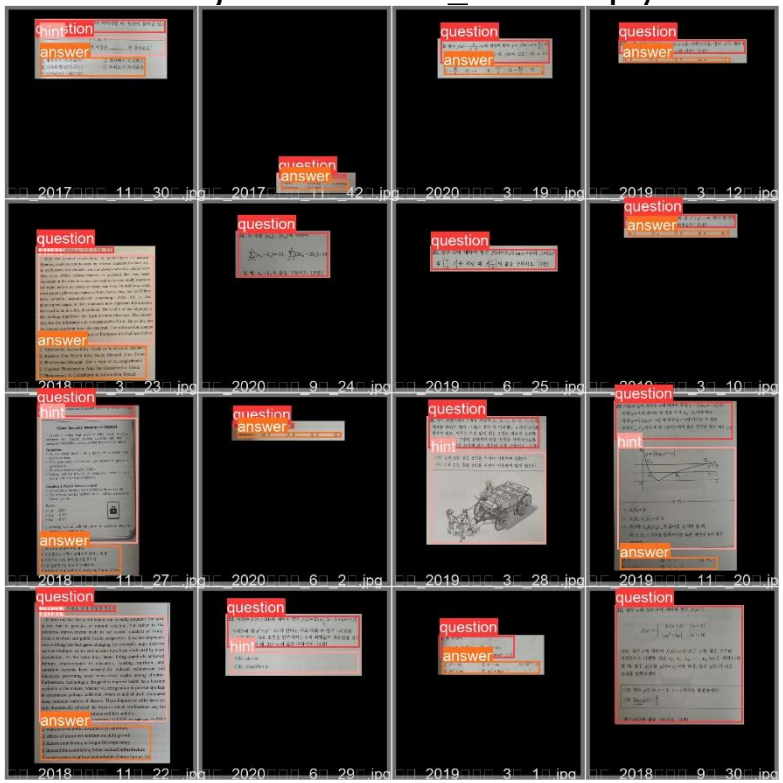
OD Practice

- **Yolov5[URL]**
  - https://github.com/airobotlab/lecture_5_yolov5
  - 1_train_yolo_colab_220509.ipynb



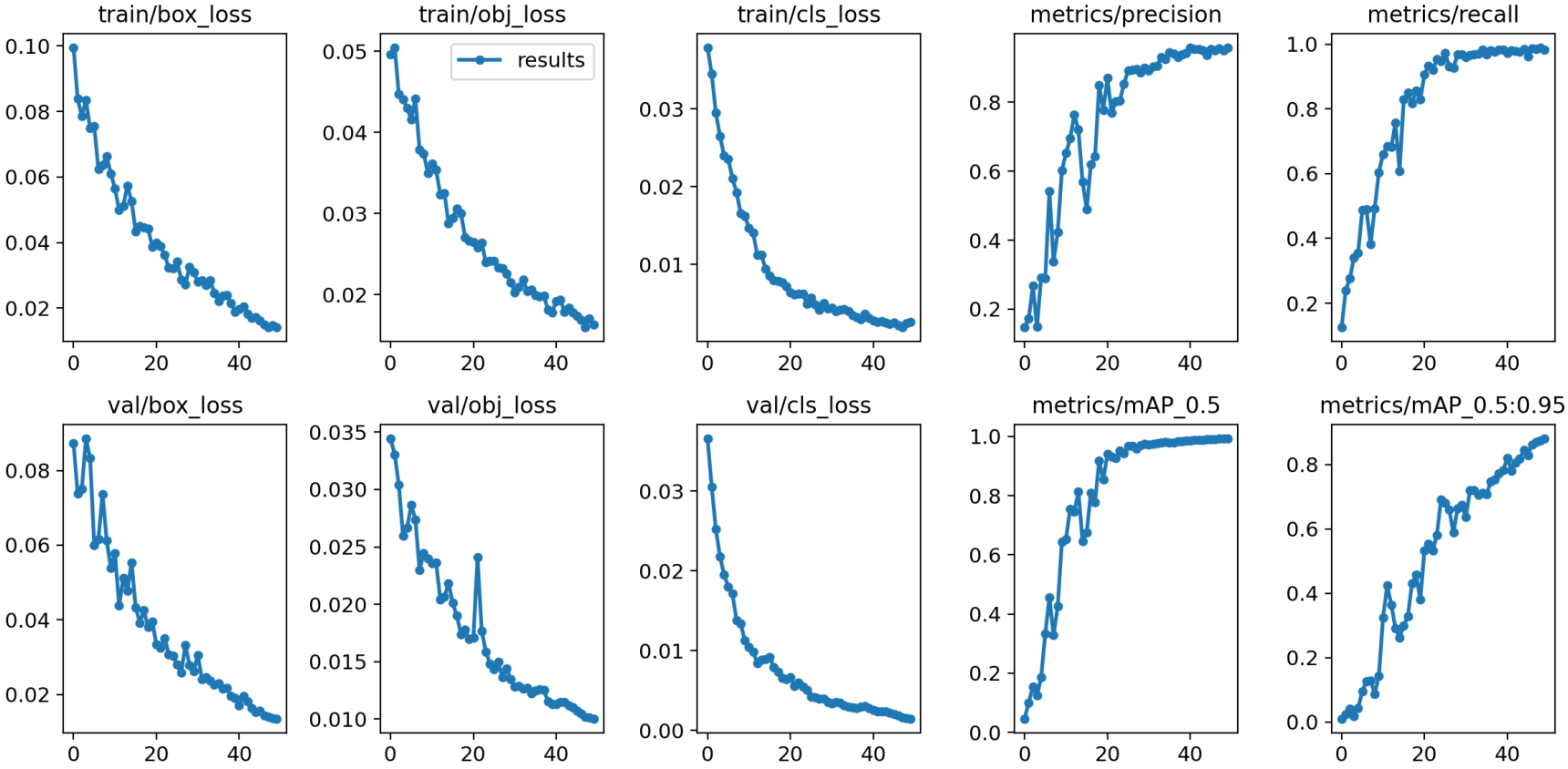
OD Practice

- **Yolov5[URL]**
  - https://github.com/airobotlab/lecture_5_yolov5
  - 1_train_yolo_colab_220509.ipynb
  - segmentation_colab.ipynb



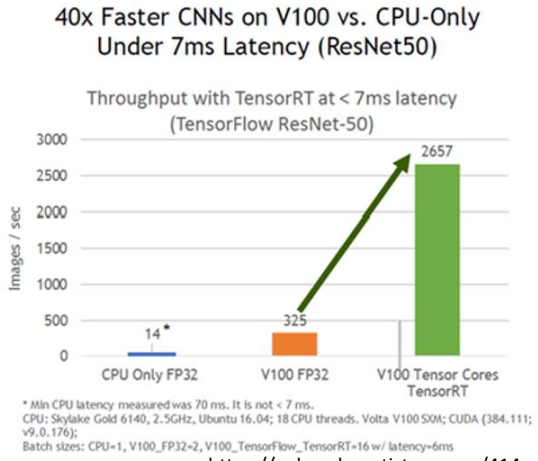
OD Practice

- **Yolov5[URL]**
  - https://github.com/airobotlab/lecture_5_yolov5
  - 1_train_yolo_colab_220509.ipynb

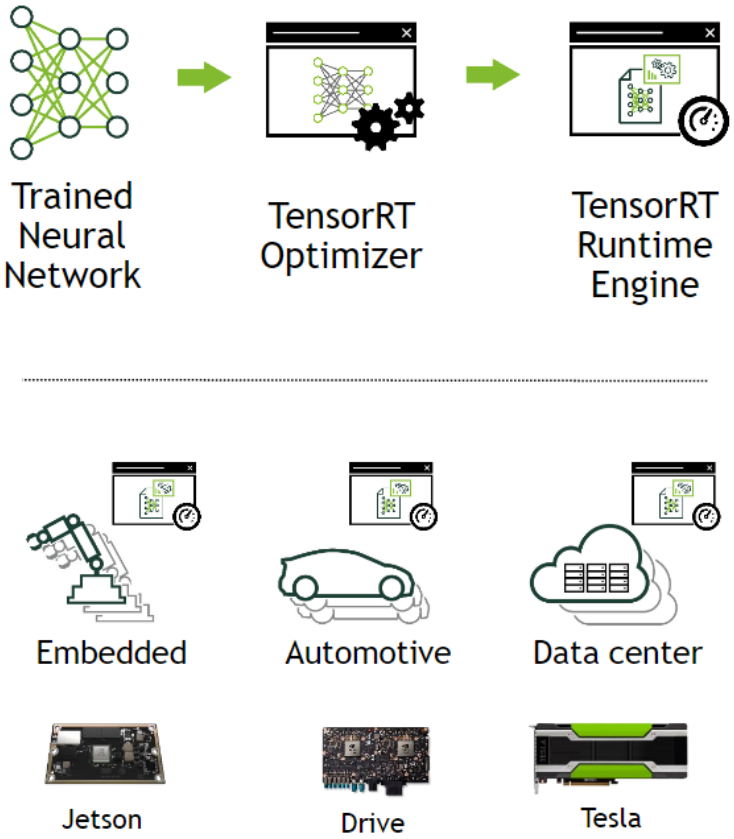


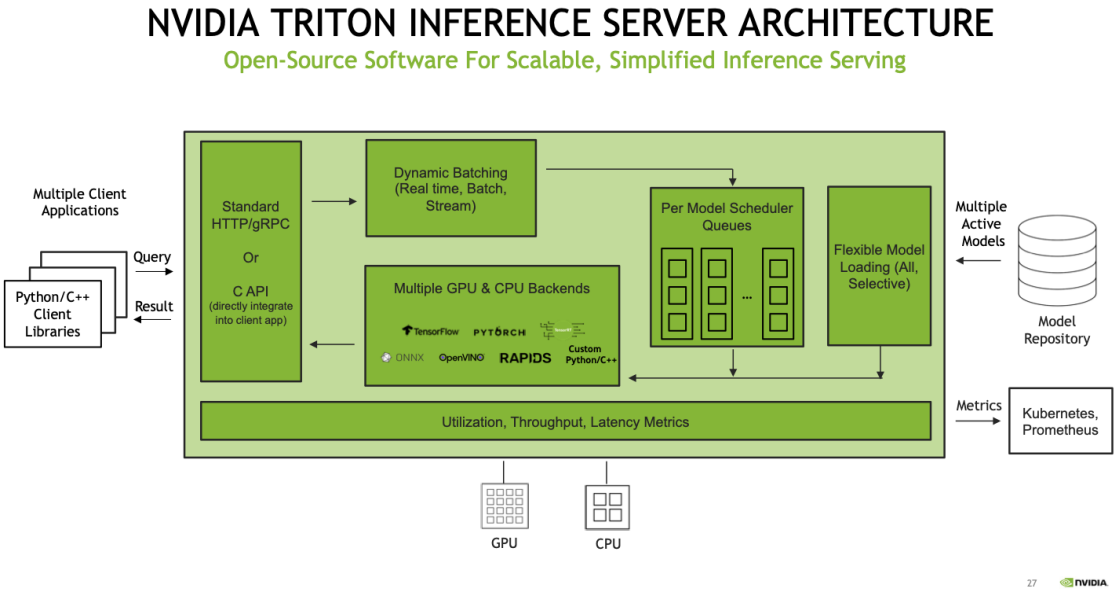
TensorRT

- **TensorRT**
  - 학습된 딥러닝 모델을 최적화
  - NVIDIA GPU 상에서의 추론 속도를 수배 ~ 수십배 까지 향상
  - 딥러닝 서비스를 개선하는데 도움을 줄 수 있는 모델 최적화 엔진
  - Caffe, Pytorch, TensorFlow 모델을 NVIDIA GPU 플랫폼(TESLA T4 , JETSON TX2, TESLA V100)에 아름답게 싣는 것
  - NVIDIA GPU 연산에 적합한 최적화 기법들을 이용하여 모델을 최적화하는 Optimizer 와 다양한 GPU에서 모델 연산을 수행하는 Runtime Engine 을 포함
    - 양자화 및 정밀도 캘리브레이션
    - 그래프 최적화
    - 커널 자동 튜닝
    - 동적 텐서 메모리 및 멀티 스트림 실행



TensorRT





https://dataintegration.info/deploy-fast-and-scalable-ai-with-nvidia-triton-inference-server-in-amazon-

Pose estimation

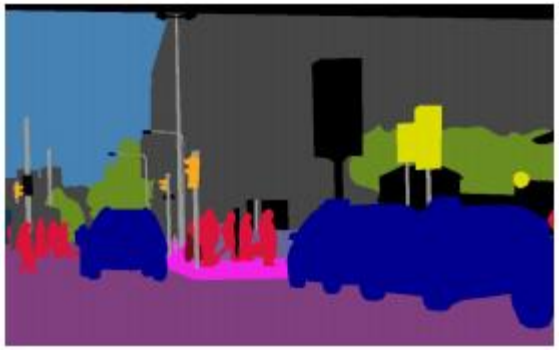
Skeleton detection with yolov7

Pose estimation 실습

- Code: https://github.com/airobotlab/colab_pose_detection.git
- Colab 실습: https://bit.ly/3CBUAER
- run 220922_pose_detection_yolov7_colab.ipynb on colab

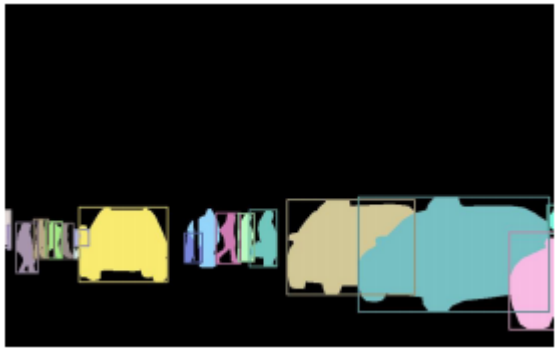
Segmentation

Semantic/Instance/Panoptic segmentation



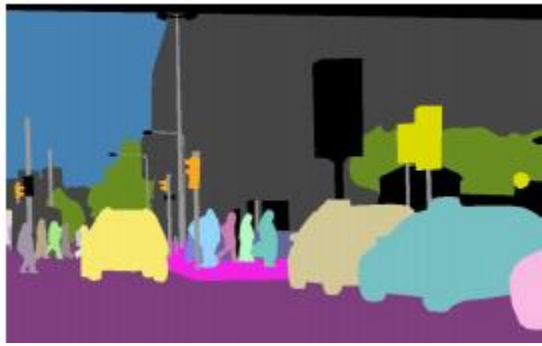
Semantic Segmentation

+



Instance Segmentation

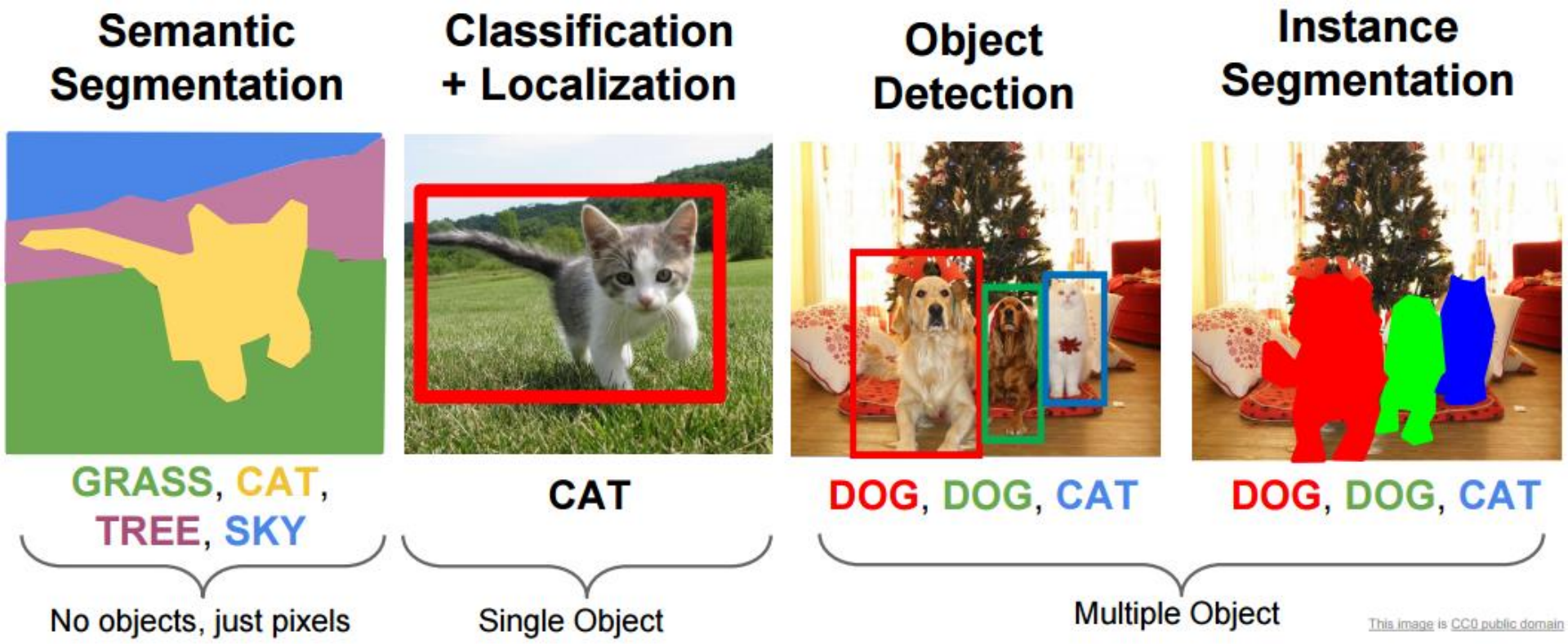
=



Panoptic Segmentation

https://github.com/airobotlab/lecture_5_yolov5

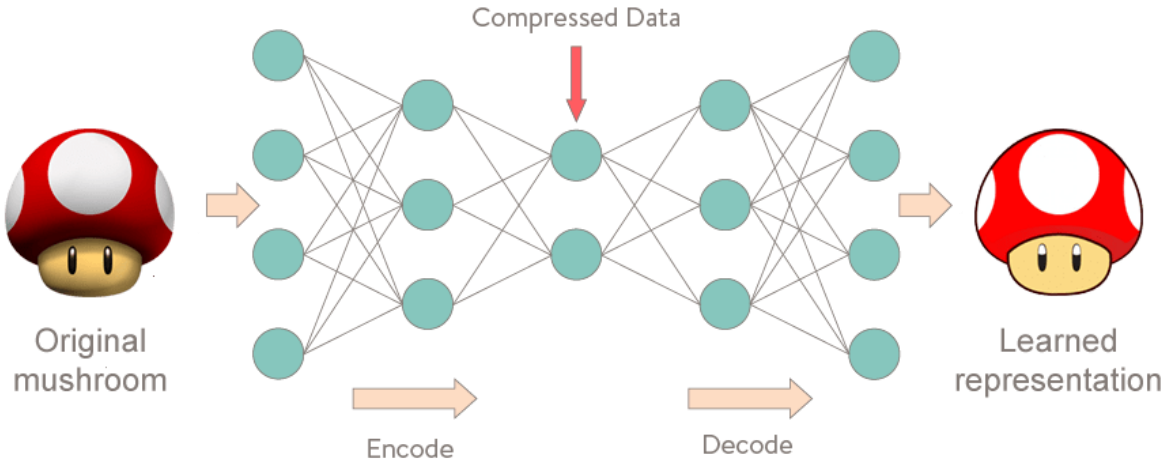
Segmentation

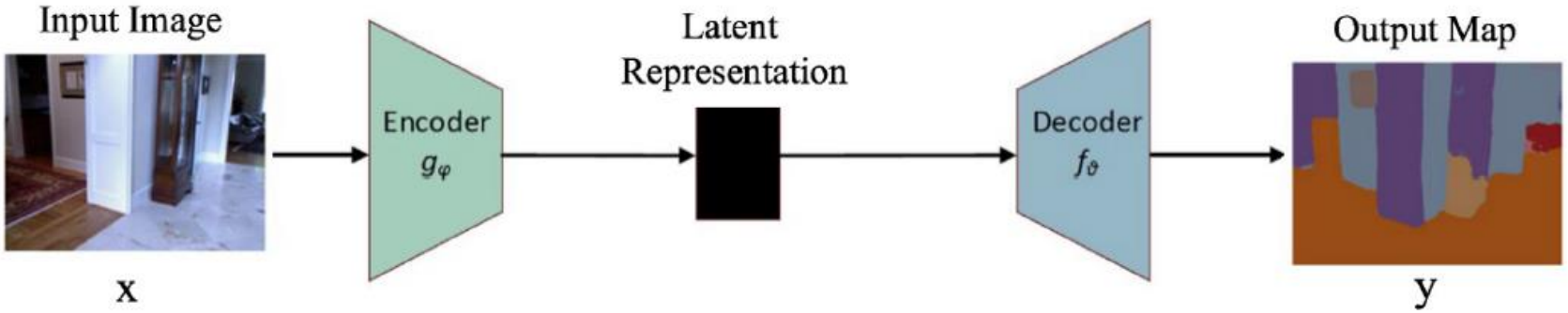


https://medium.com/@ashika.pobbathi/image-segmentation-using-deep-learning-models-and-datasets-98609c02a2b

Encoder-Decoder & Auto-Encoder Model

- **Encoder to extract input embedding vector**
- **Decoder make a output from embedding vector**





https://medium.com/@ashika.pobbathi/image-segmentation-using-deep-learning-models-and-datasets-98609c02a2b

Segmentation

- **2D Image Dataset**
  - **PASCAL Visual Object Classes (VOC), 21 classes**
  - **PASCAL Context, 59 classes**
  - **Microsoft Common Objects in Context, 91 classes, 328,000 images**
  - **Cityscapes, 30 classes grouped into 8 categories**

gtFine_trainvaltest.zip (241MB) [md5]

fine annotations for train and val sets (3475 annotated images) and dummy annotations (ignore regions) for the test set (1525 images)

gtCoarse.zip (1.3GB) [md5]

coarse annotations for train and val set (3475 annotated images) and train_extra (19998 annotated images)

leftImg8bit_trainvaltest.zip (11GB) [md5]

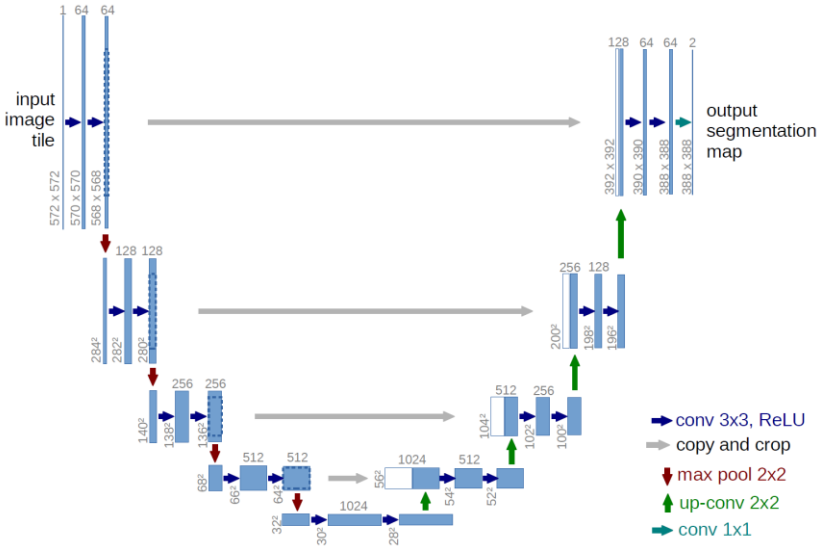
left 8-bit images - train, val, and test sets (5000 images)

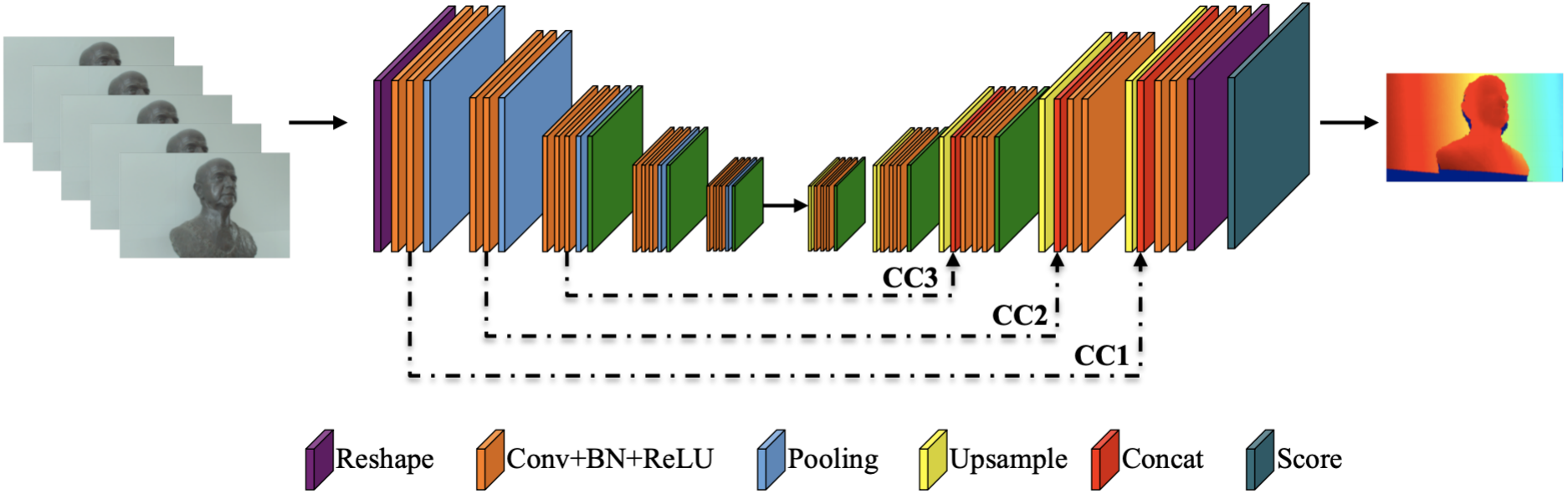
leftImg8bit_trainextra.zip (44GB) [md5]

left 8-bit images - trainextra set (19998 images, note that the image "troisdorf_000000_000073_leftImg8bit.png" is corrupt/black)

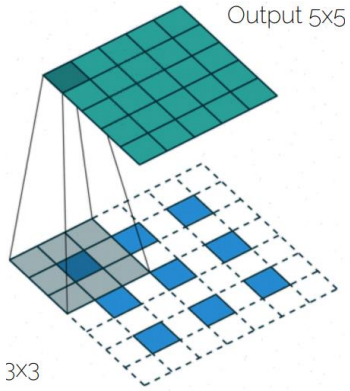
https://www.cityscapes-dataset.com/download

Unet

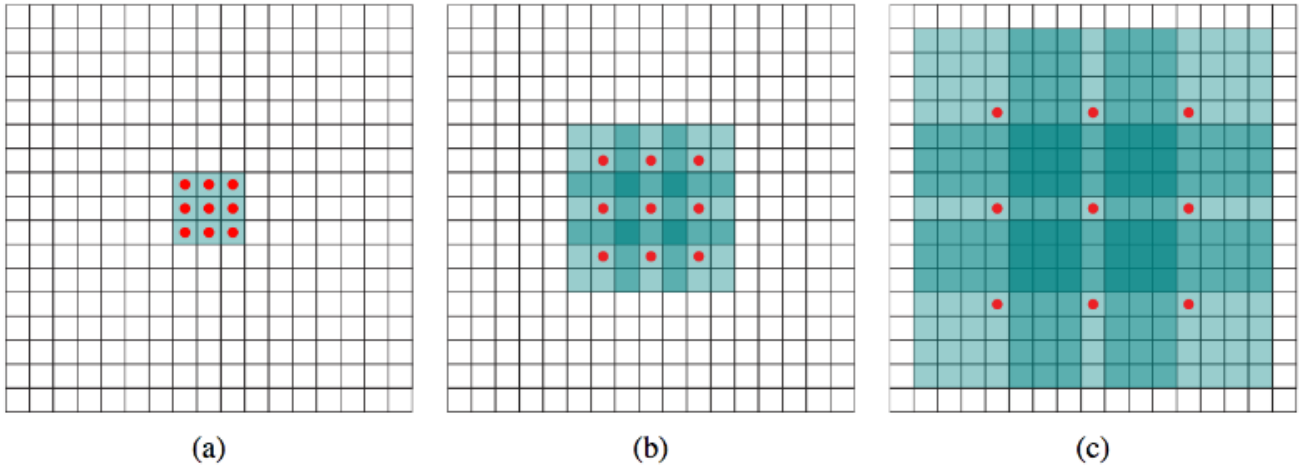

input image tile
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1


CC3
CC2
CC1
Reshape
Conv+BN+ReLU
Pooling
Upsample
Concat
Score

Upsampling



Transposed Convolution



Dilated Convolutions 2D

Segmentation dataset format

- Same size, input image – mask with label
- Multi class classification

Input

segmented

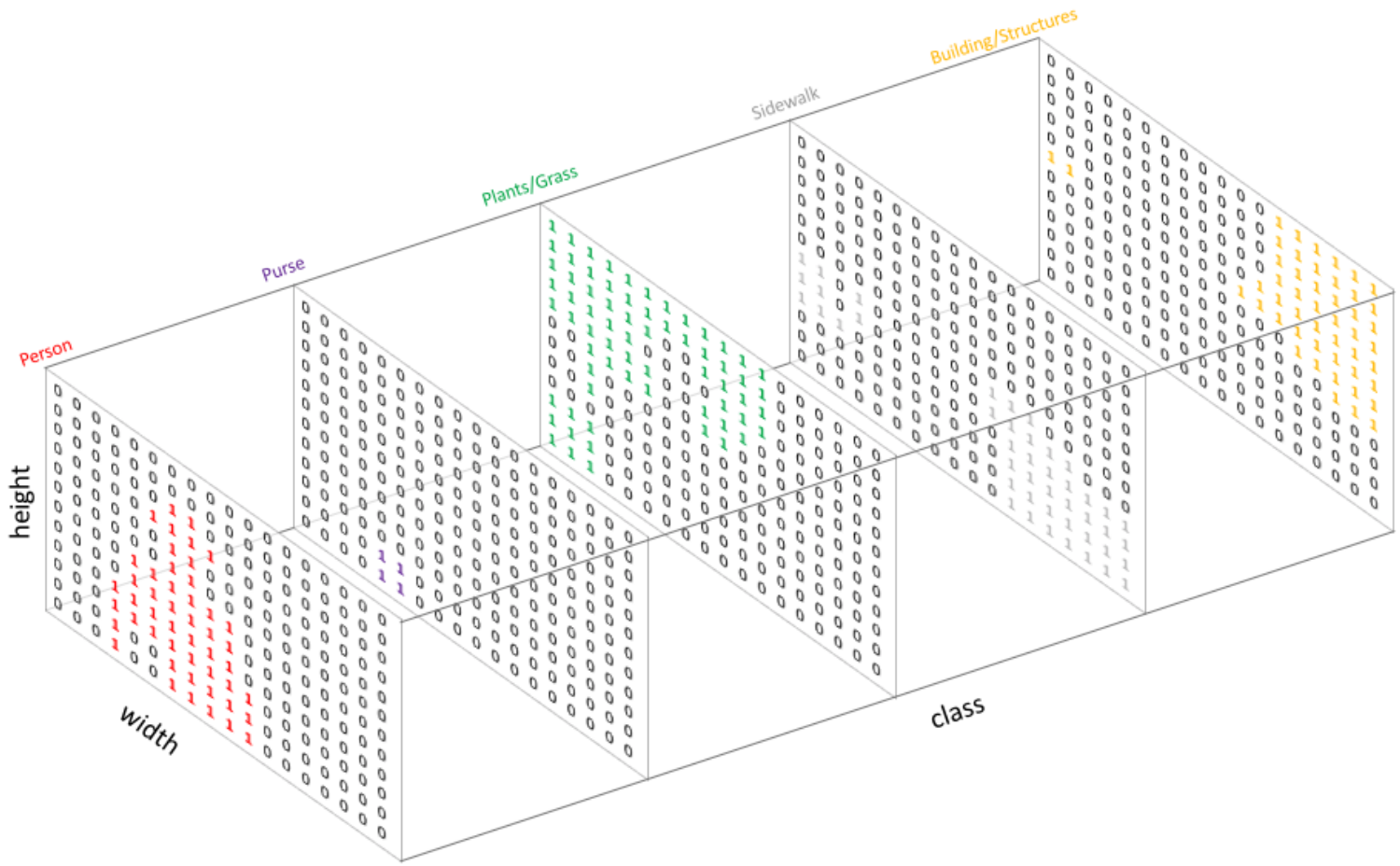
1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

```
3 3 3 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5
3 3 3 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5
3 3 3 3 3 3 1 1 3 3 3 3 5 5 5 5 5 5
3 3 3 3 3 1 1 1 1 3 3 3 5 5 5 5 5 5
3 3 3 3 3 3 1 1 3 3 3 5 5 5 5 5 5 5
5 5 3 3 3 3 1 1 3 3 5 5 5 5 5 5 5 5
4 4 3 4 1 1 1 1 1 1 4 4 4 5 5 5 5 5
4 4 3 4 1 1 1 1 1 1 4 4 4 4 4 5 5 5
4 4 4 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4
3 3 3 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4
3 3 3 1 2 2 1 1 1 1 1 4 4 4 4 4 4 4
3 3 3 1 2 2 1 1 1 1 1 4 4 4 4 4 4 4
```

Semantic Labels

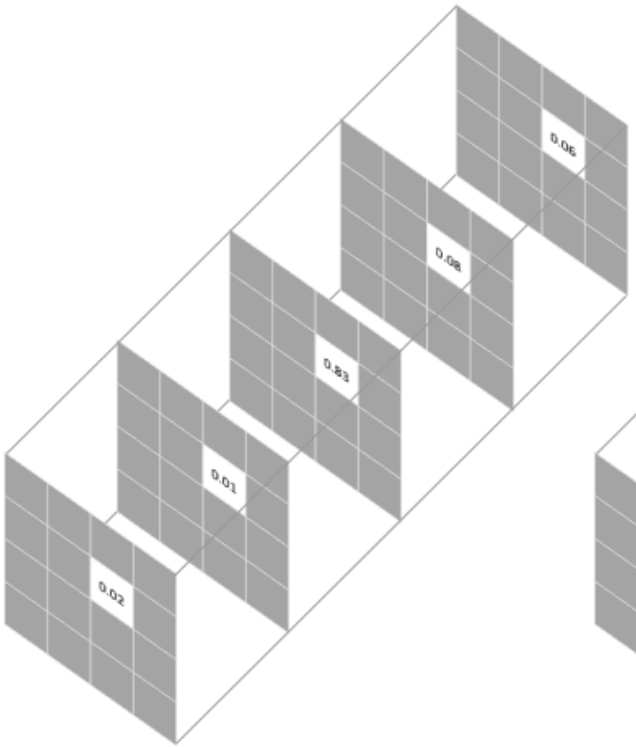
Object Detection

- **Multi-label classification**

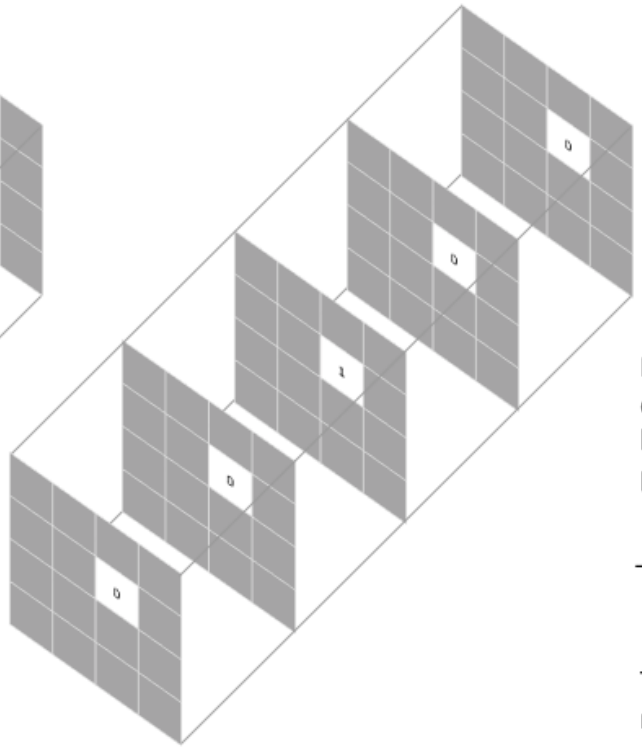


Object Detection

- **Multi-label classification**



Prediction for a selected pixel



Target for the corresponding pixel

Pixel-wise loss is calculated as the log loss, summed over all possible classes

$$-\sum_{classes} y_{true} \log\left(y_{pred}\right)$$

This scoring is repeated over all **pixels** and averaged

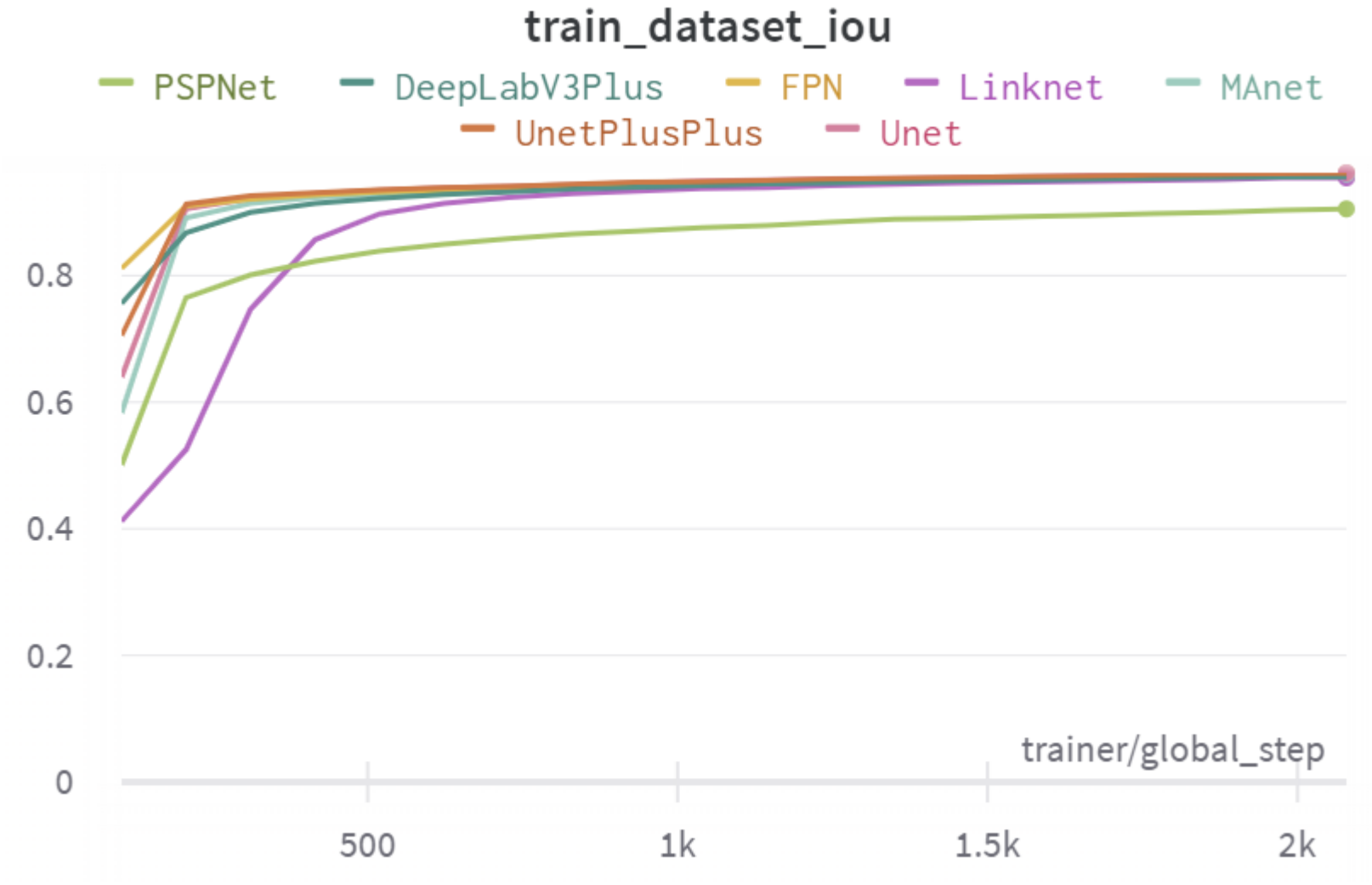
Experiment

- **Simple Oxford Pet Dataset(3x256x256)**

	Test	Valid	Traintime (min) /batch	Iteration /s	Parameters(M)	CPU 100 image inference (s)	Batch
PSPNet	87.83	87.57	1:30	4.67	28.4	**10.3**	32
DeepLabV3	91.46	91.12	**23:49**	2.88	33.1	40.8	**16**
DeepLabV3+	91.34	91.19	**5:07**	3.43	29.5	30.2	32
FPN	91.71	91.64	1:22	4.63	30.2	34.1	32
Linknet	91.65	91.42	1:22	4.74	28.7	35.0	32
MAnet	92.02	91.35	1:32	4.10	38.2	36.4	32
Unet	92.05	**91.88**	1:35	4.27	31.2	34.8	32
Unet++	91.75	91.78	1:48	3.71	31.9	40.0	32

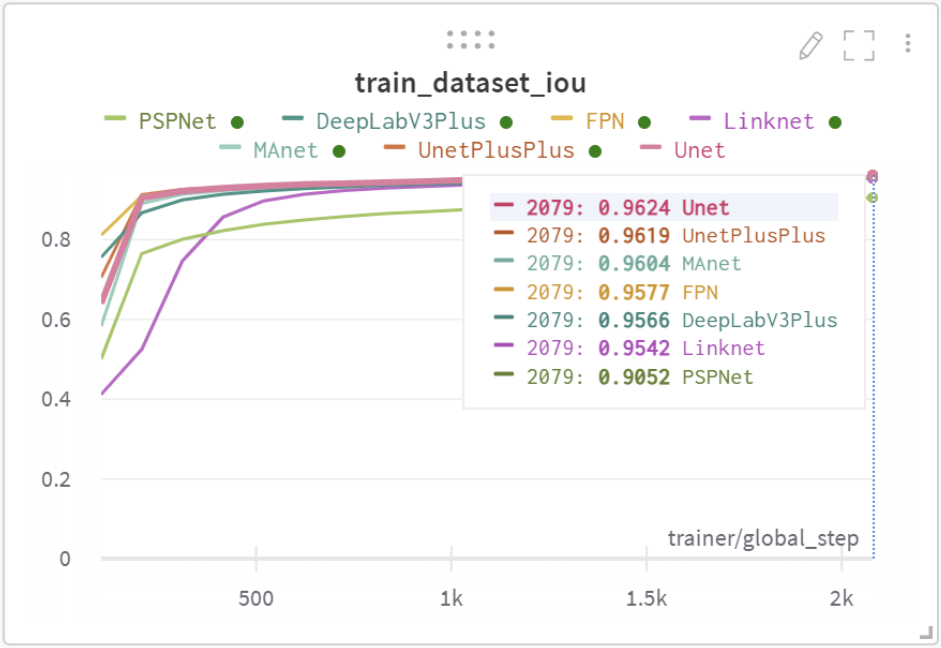
Experiment

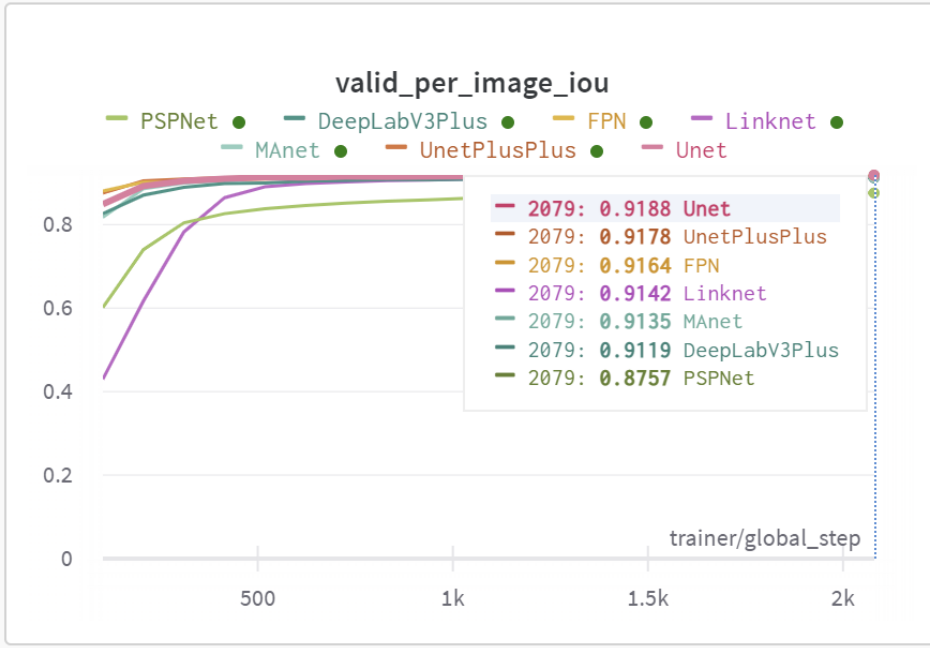
- **Simple Oxford Pet Dataset(3x256x256)**



Experiment

- **Simple Oxford Pet Dataset(3x256x256)**





Segmentation 실습

- Code: https://github.com/airobotlab/lecture_5_yolov5.git
- Colab 실습: https://bit.ly/3T2dYAw
- run segmentation_colab_220916.ipynb on colab

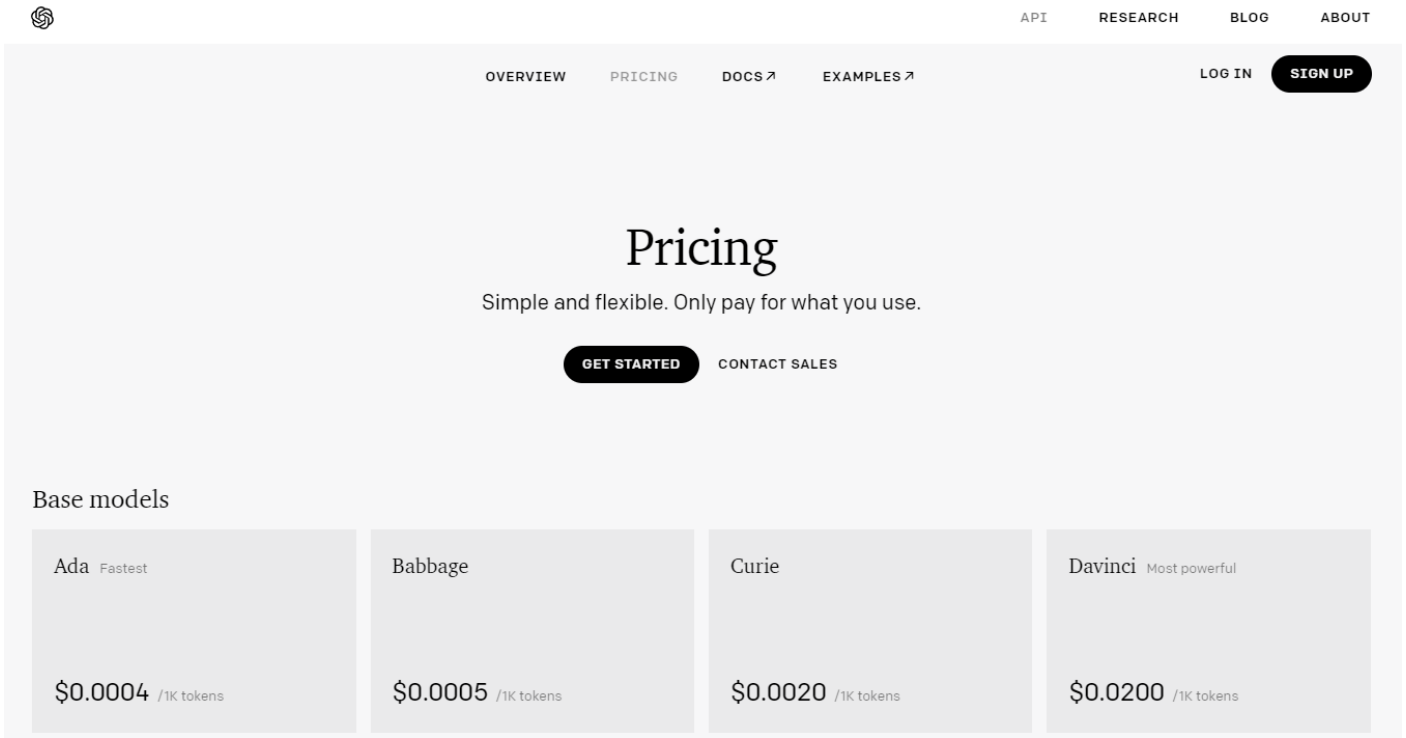
Image generation

Text2image generation/Stable diccusion

https://replicate.com/stability-ai/stable-diffusion/examples

Text2Image generation

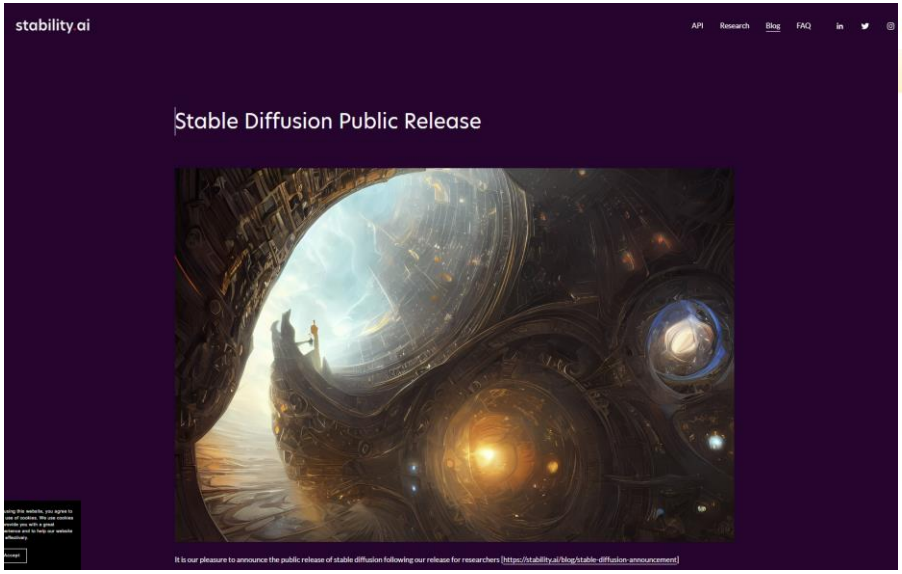
- **Dalle-2: https://openai.com/dall-e-2/**

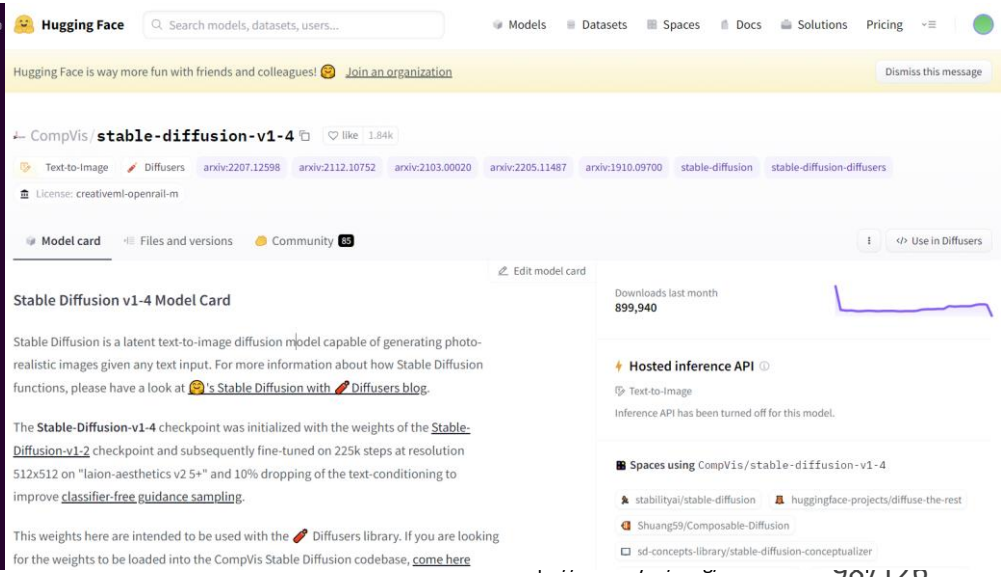


https://openai.com/api/pricing/

Text2Image generation

- **Dalle-2: https://openai.com/dall-e-2/**
- **Stable diffusion (CVPR)**
  - **github: https://github.com/CompVis/stable-diffusion**
  - **huggingface: https://huggingface.co/CompVis/stable-diffusion-v1-4**

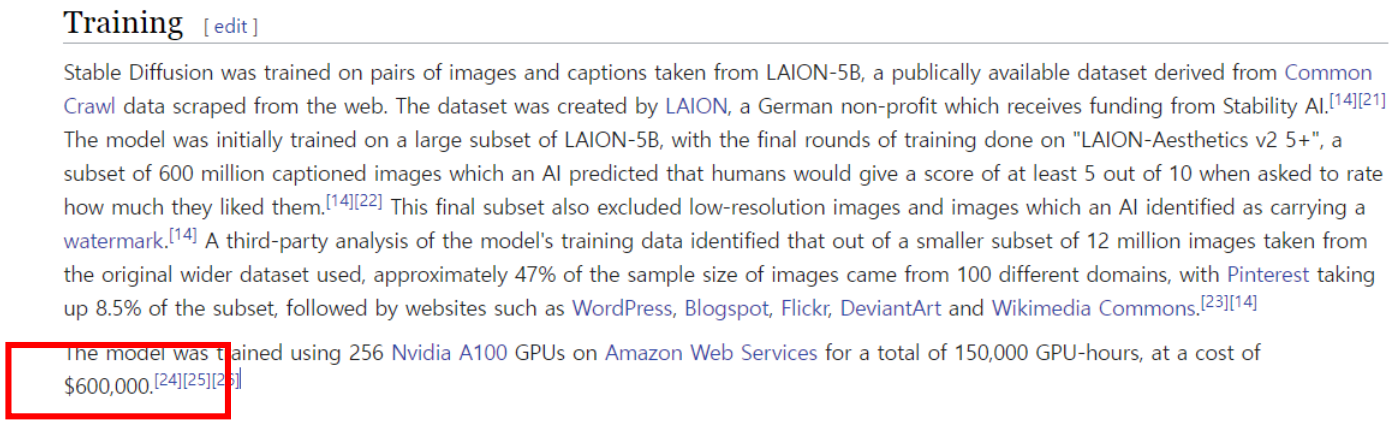




https://huggingface.co/CompVis/stable-diffusion-v1-4

Text2Image generation

- **Stable diffusion (CVPR)**
  - **github:** https://github.com/CompVis/stable-diffusion
  - **huggingface:** https://huggingface.co/CompVis/stable-diffusion-v1-4
  - **키워드:** https://lexica.art/
  - **가이드북:** https://dallery.gallery/the-dalle-2-prompt-book/
  - **실습1:** https://bit.ly/3SVko4a
  - **실습2:** https://bit.ly/3V0eXmx
  - **설명:** https://velog.io/@hewas1230/StableDiffusion



Training [edit]

Stable Diffusion was trained on pairs of images and captions taken from LAION-5B, a publically available dataset derived from Common Crawl data scraped from the web. The dataset was created by LAION, a German non-profit which receives funding from Stability AI.[14][21] The model was initially trained on a large subset of LAION-5B, with the final rounds of training done on "LAION-Aesthetics v2 5+", a subset of 600 million captioned images which an AI predicted that humans would give a score of at least 5 out of 10 when asked to rate how much they liked them.[14][22] This final subset also excluded low-resolution images and images which an AI identified as carrying a watermark.[14] A third-party analysis of the model's training data identified that out of a smaller subset of 12 million images taken from the original wider dataset used, approximately 47% of the sample size of images came from 100 different domains, with Pinterest taking up 8.5% of the subset, followed by websites such as WordPress, Blogspot, Flickr, DeviantArt and Wikimedia Commons.[23][14]

The model was trained using 256 Nvidia A100 GPUs on Amazon Web Services for a total of 150,000 GPU-hours, at a cost of $600,000.[24][25][26]

ble-diffusion-public-release
mpVis/stable-diffusion-v1-4

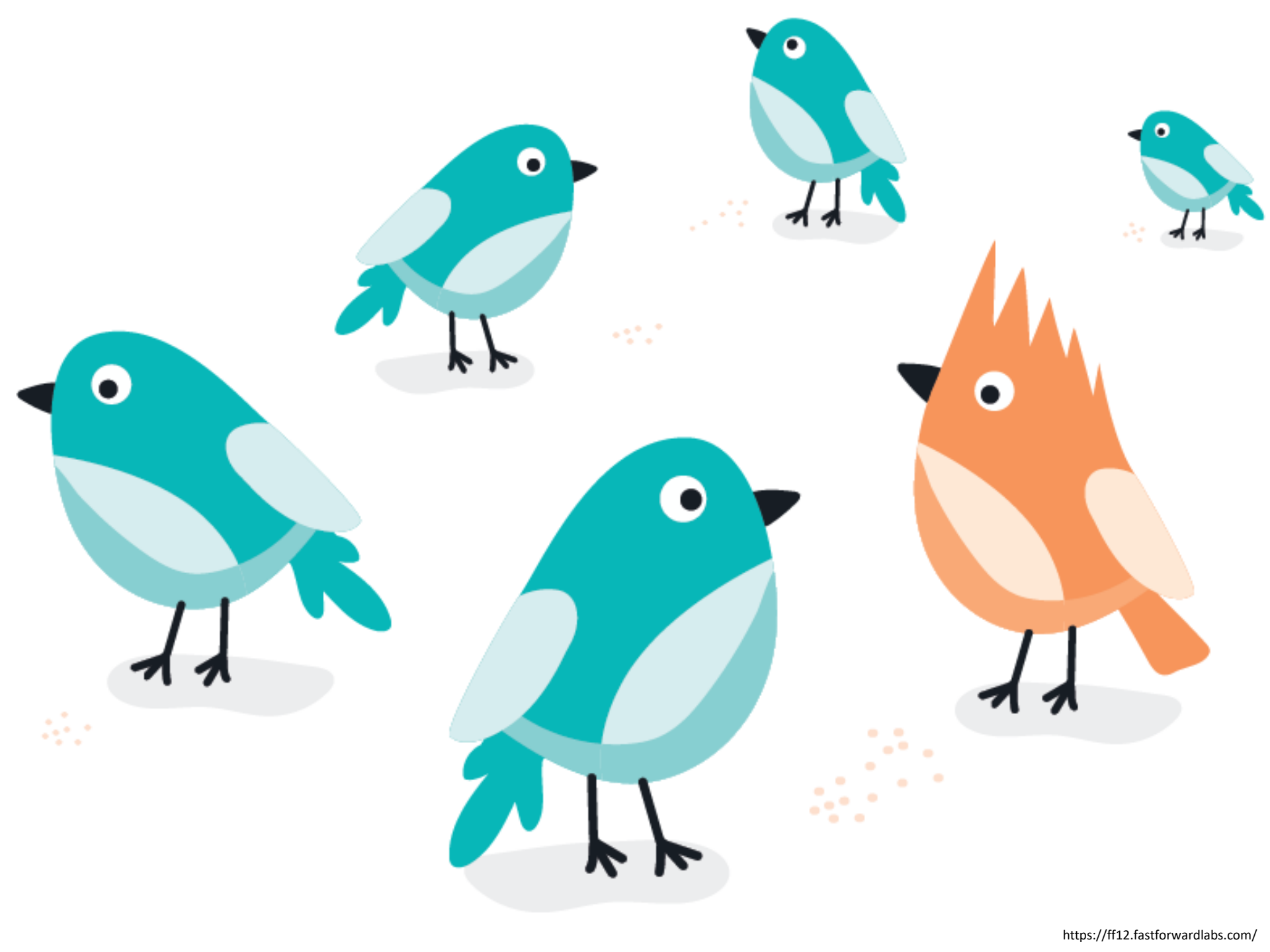
Anomaly Detection

Time series data/Image data

Time Series Anomaly Detection

GAN/Transformer/GRU

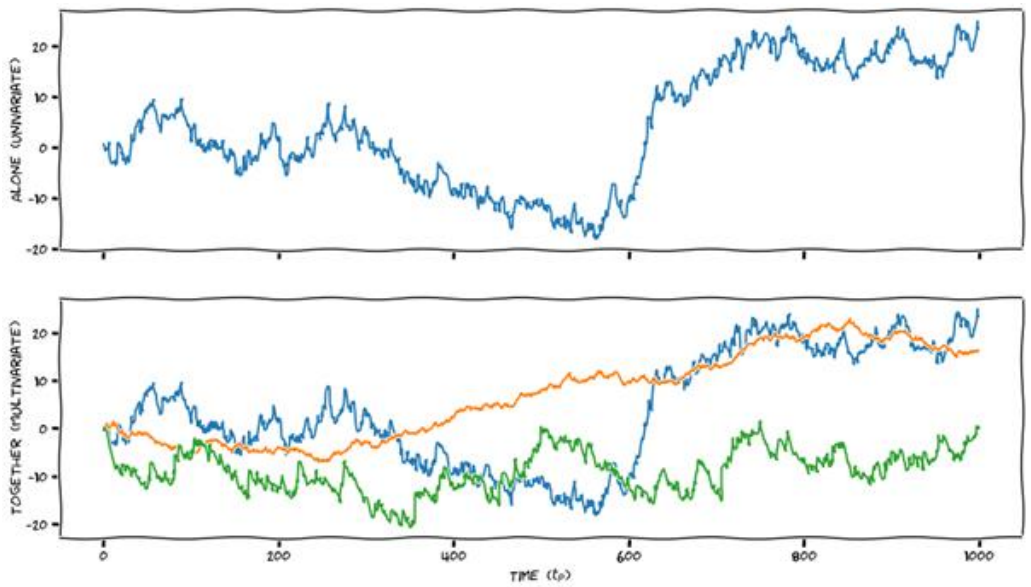
https://zindi.africa/learn/introduction-to-anomaly-detection-using-machine-learning-with-a-case-study



https://ff12.fastforwardlabs.com/

Time series types

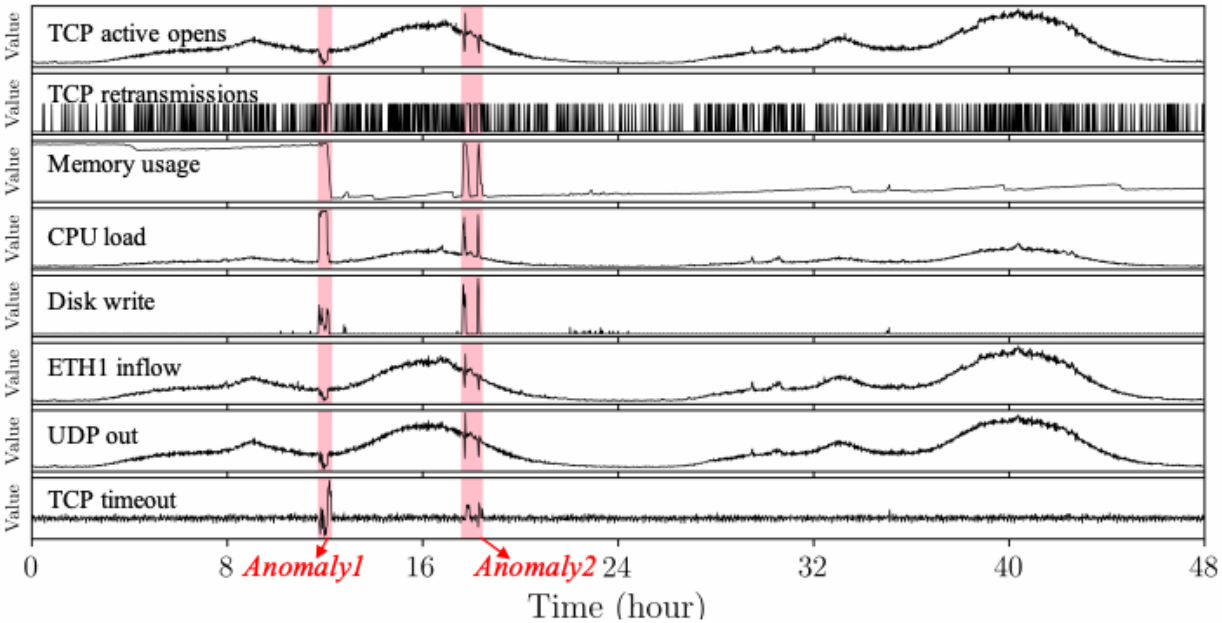
- **Univariate**
  - 단변량 시계열 데이터
  - 주식가격, 유가, 전력 수요
- **Multivariate**
  - 다변량 시계열 데이터
  - 공정 센서 데이터



https://medium.com/geekculture/vector-auto-regression-for-multivariate-time-series-forecasting-9334d29591f3

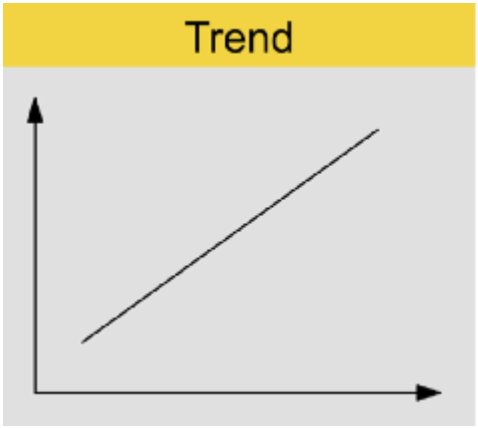
Multivariate Time series Anomaly Detection

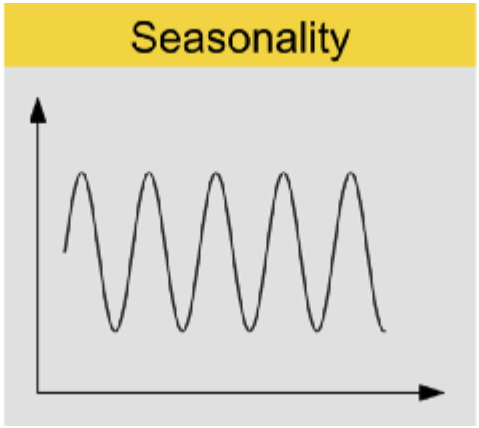
- **Dataset**
  - m개의 변수로 이루어진 t시점 m차원 벡터가 총 시간 T 만큼 존재
- **Task**
  - 길이가 K인 time window $W_t = \{x_{t-k}, \dots, x_{t-1}, x_t\}$를 입력으로 t 시점의 normal/abnormal 여부를 예측

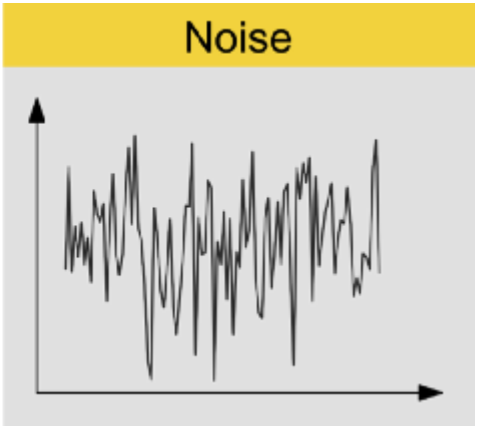


Time series data components

- **Trend**
- **Seasonality**
- **Noise**



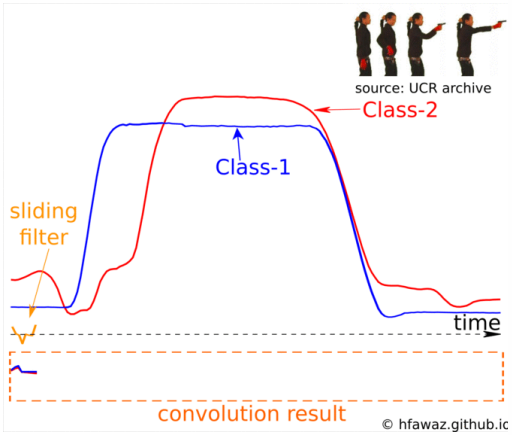


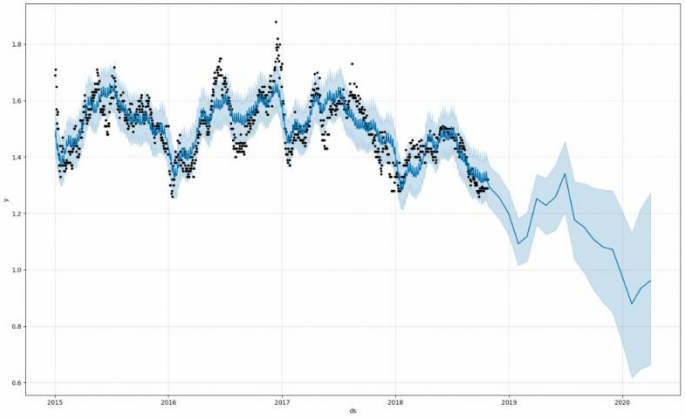


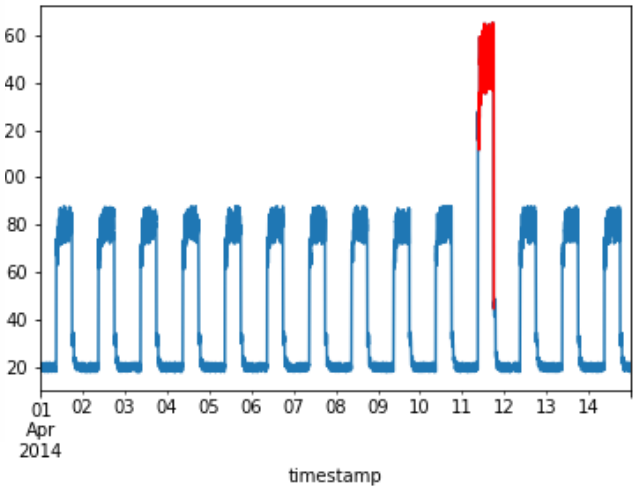
https://towardsdatascience.com/a-comprehensive-beginners-guide-to-the-diverse-field-of-anomaly-detection-8c818d153995

Time series tasks

- 1) Classification
- 2) Forecasting
- 3) Anomaly Detection



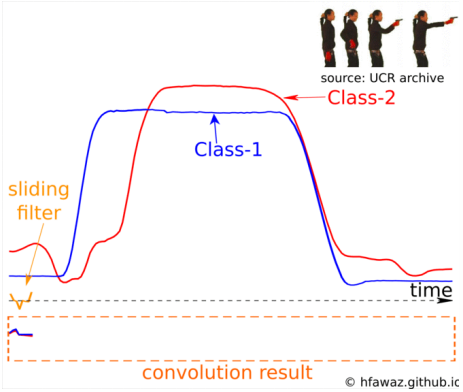


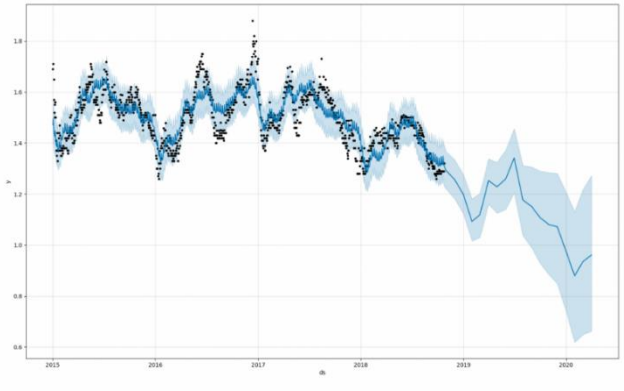


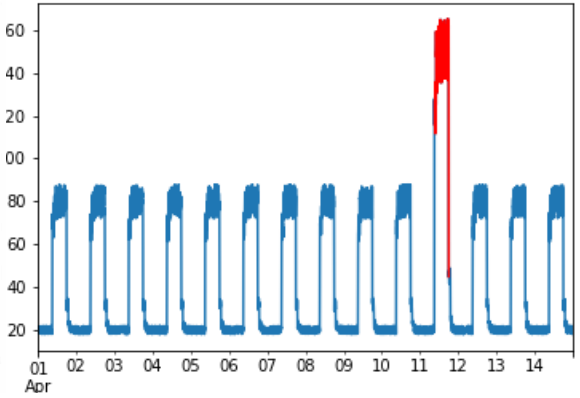
https://medium.com/@hassanismailfawaz/deep-learning-for-time-series-classification-a-brief-overview-73b58767ed0f
https://keras.io/examples/timeseries/timeseries_anomaly_detection/
https://syllepsis.live/2019/10/08/time-series-forecasting-a-quick-reference/

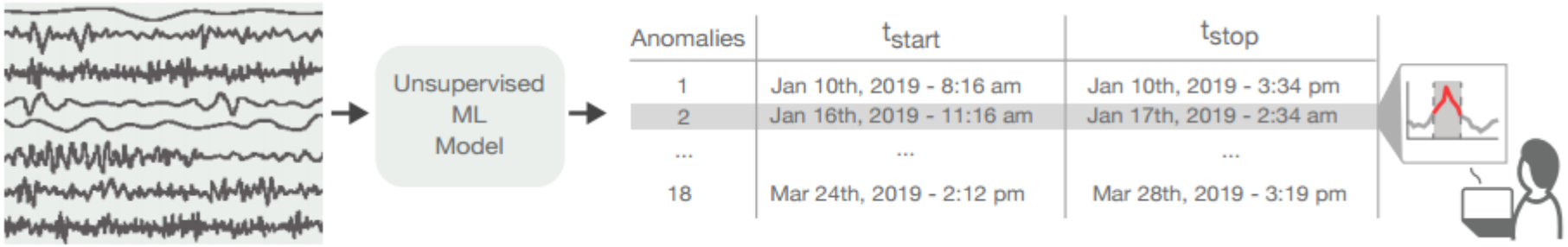
Time Series Tasks

- 1) Classification
- 2) Forecasting
- 3) Anomaly Detection
  - 과거 data를 기반으로 t시점에서의 abnormal 여부 예측



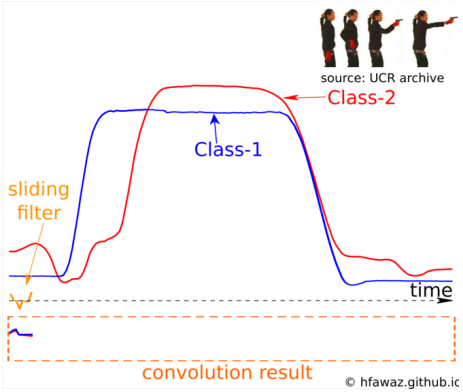


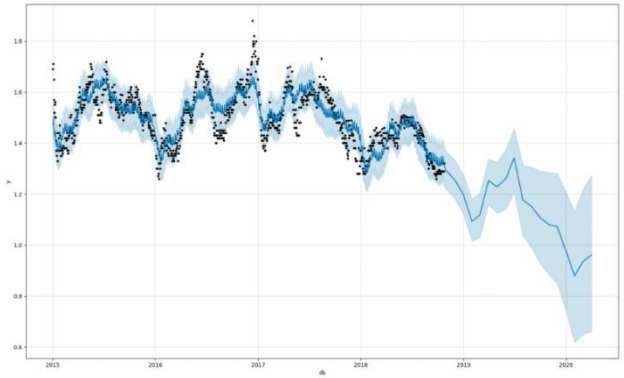


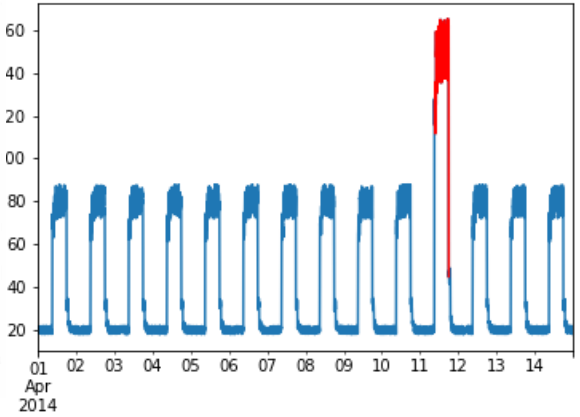


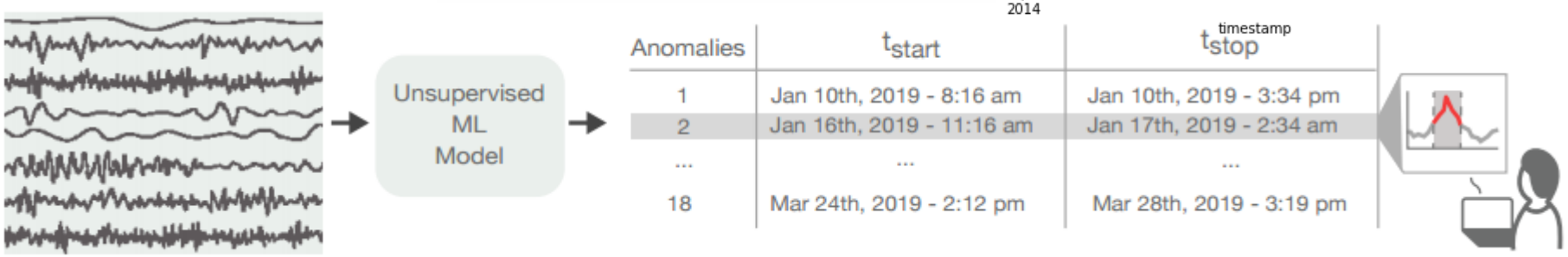
Time Series Tasks

- **3) Anomaly Detection**
  - 과거 data를 기반으로 t시점에서의 abnormal 여부 예측
  - 제조 공정 과정에서 불량품이나 기계 고장을 탐지
  - System Security 분야에서 보안이 위협받는 상황을 판별









Anomaly

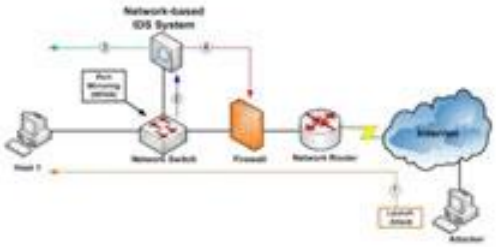
- **Anomaly Detection**
  - 정상 데이터와 본질적으로 다름
- **Novelty Detection**
  - 정상 데이터와 본질적으로 같지만 유형이 다름
- **Outlier Detection**
  - outlier: 일반적인 데이터 생성 매커니즘을 위배해서 만들어진 데이터
  - noise: 데이터 수집 관점에서 자연적으로 발생하는 변동성 (random error or variance)

https://yupsung.blogspot.com/2021/01/anomaly-detection-overview.html

Anomaly Detection 적용 사례

- **Anomaly Detection**
  - 정상 데이터와 본질적으로 다름

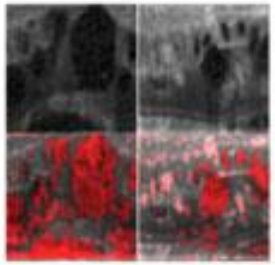
Cyber-intrusion



Fraud

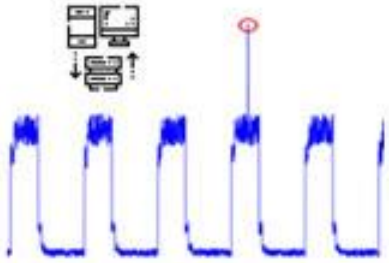
Malware

Medical

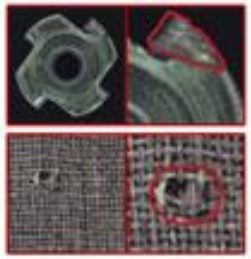


Social Network

Log file



IoT Big-Data



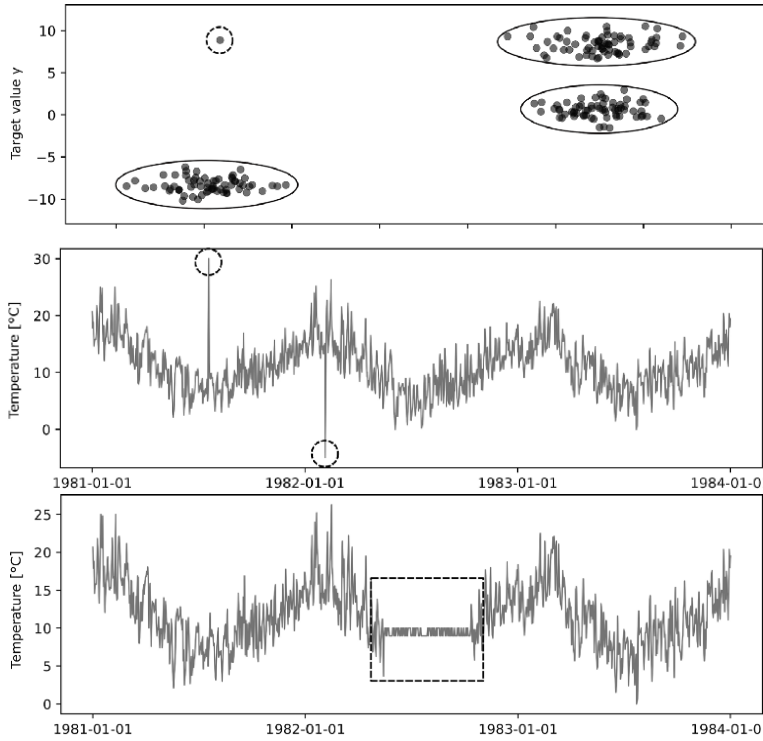
Industrial

Video Surveillance

https://arxiv.org/abs/1901.03407

https://www.cognex.com/ko-kr/blogs/deep-learning/research/anomaly-detection-overview-1-introduction-anomaly-detection

Types of Anomalies



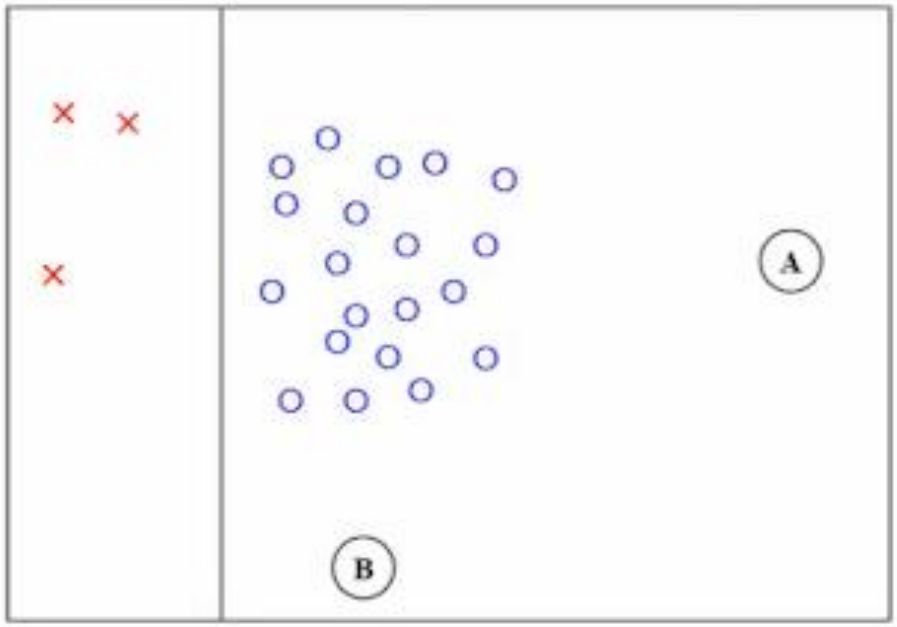
- **Point/Global Anomalies**
  - 대다수의 Data set과 완전히 다른 객체
  - ex)Credit card fraud detection
- **Contextual Anomalies**
  - 조건부(context) 이상치, 특정 조건이 충족될 때 이상치로 판단됨
  - ex)온도 29도는 우리나라에선 정상이지만, 북극에서는 특이한 케이스
- **Collective Anomalies**
  - 단일 객체들은 outlier가 아니지만 모아서 보면 편차가 심한 케이스
  - ex)DDos공격. 단일 접속 패킷을 정상, 한번에 접속이 너무 많으면 서버 다운

https://towardsdatascience.com/a-comprehensive-beginners-guide-to-the-diverse-field-of-anomaly-detection-8c818d153995
https://yupsung.blogspot.com/2021/01/anomaly-detection-overview.html

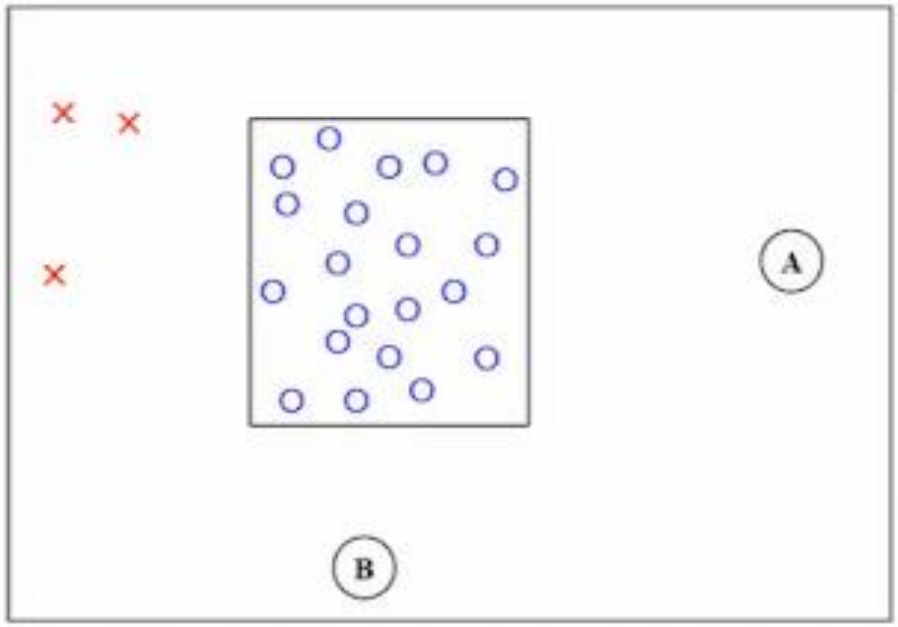
What is Anomaly?

- **데이터 생성 매커니즘**
  - 일반적인 데이터와 다른 매커니즘으로 발생한 data
- **데이터 분포**
  - Data가 발생할 확률 밀도가 매우 낮은 data

Classification vs Anomaly Detection



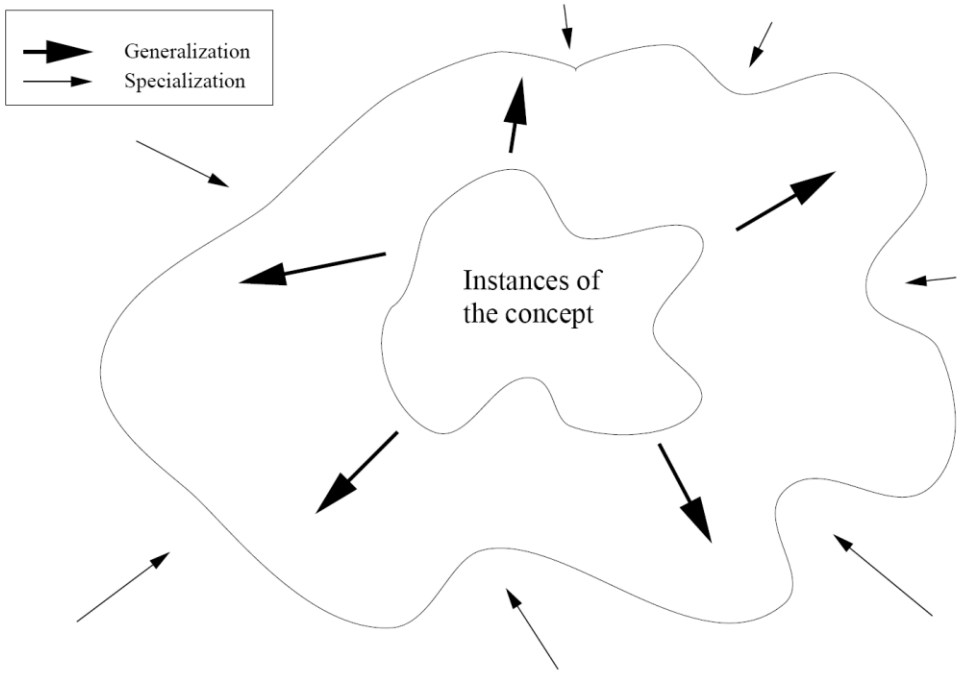
Binary classification



Anomaly detection

Generalization vs Specialization

- **Generalization: 일반화**
  - abnormal data를 normal로 오분류
- **Specialization: 구체화, 특수화**
  - normal data를 abnormal로 오분류
- **Trade off 관계. 조절 필요**

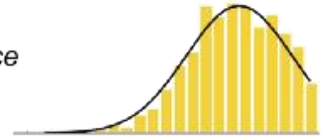


Anomaly Detection

Unsupervised outlier detection

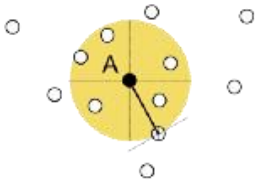
Probabilistic Methods

e.g. Robust Covariance Estimatior



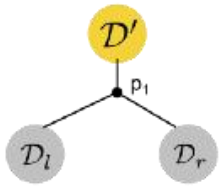
Distance and Density methods

e.g. Local Outlier Factor



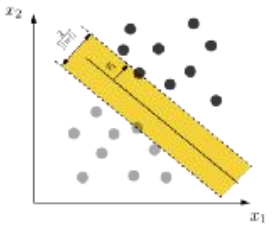
Decision Trees and Ensemble methods

e.g. Isolation Forest



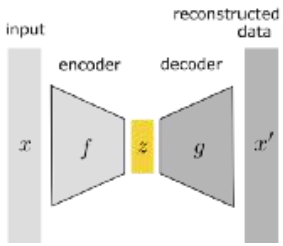
Kernel methods

e.g. One-Class SVM



Deep Learning

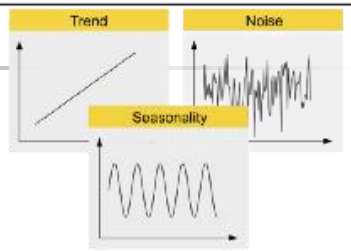
e.g. Autoencoder



Model-based approaches

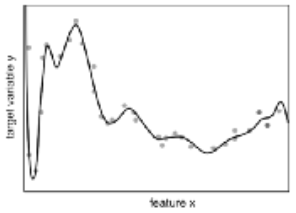
Time Series Analysis

e.g. Moving Average



Regression Analysis

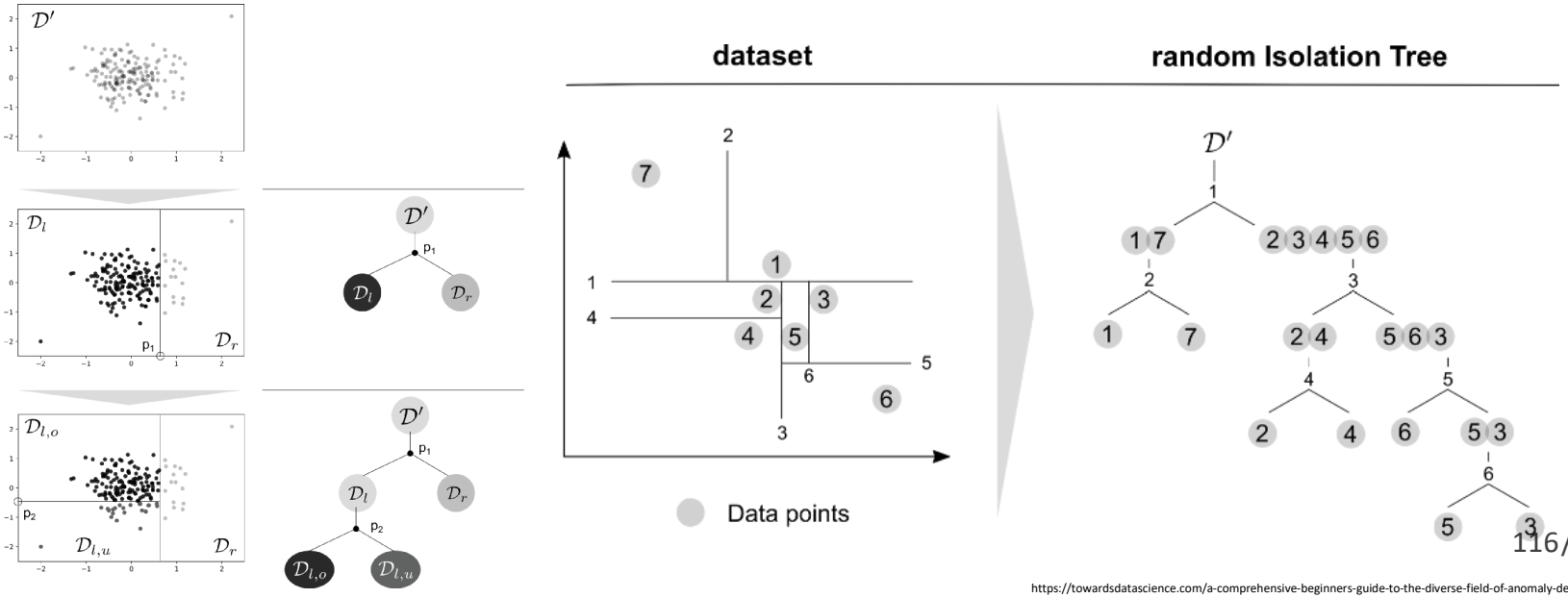
e.g. Polynomial Regression



https://towardsdatascience.com/a-comprehensive-beginners-guide-to-the-diverse-field-of-anomaly-detection-8c818d153995

Isolation Forest

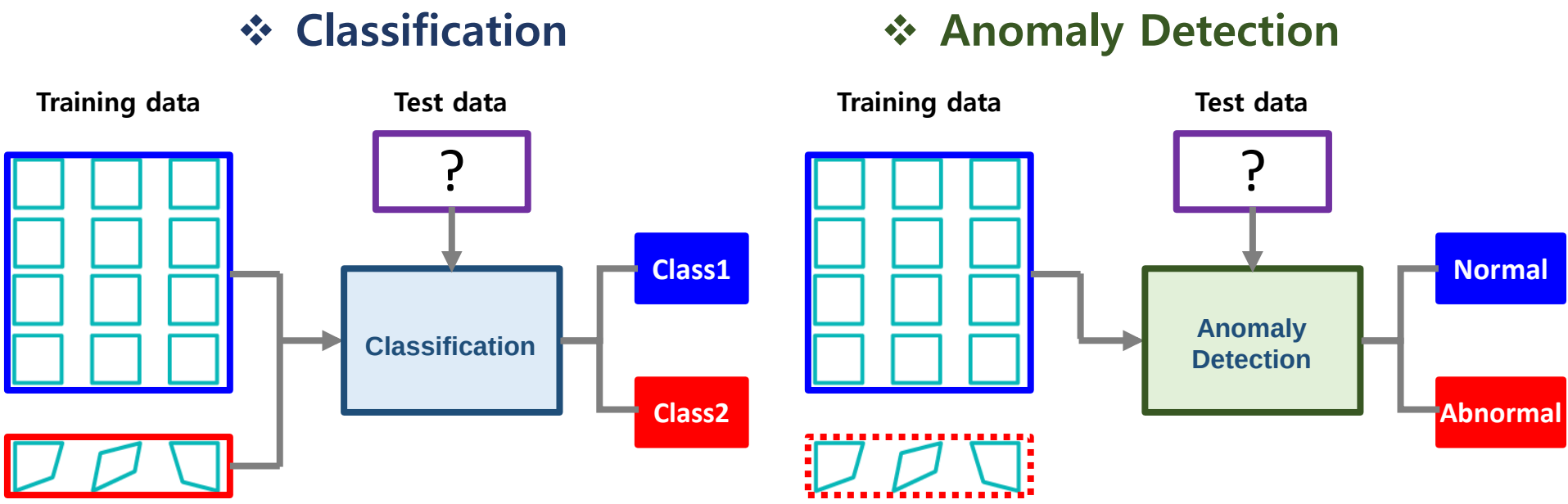
- 정상/비정상을 가르는 기준은 해당 데이터를 isolation(고립) 하는데 걸린 평균 분기 횟수로 예측
- 정상 데이터는 밀집 지역에 분포
- 비정상 데이터는 그로부터 떨어진 밀도가 낮은 지역에 분포
- 분기 횟수가 적을 수록 비정상
- 분기 횟수가 높을수록 정상



https://towardsdatascience.com/a-comprehensive-beginners-guide-to-the-diverse-field-of-anomaly-detection-8c818d153995

Classification VS Anomaly Detection

- Normal data가 Abnormal data 보다 많은 상황 전제
- Training: Only normal data만으로 모델 training
- Test: Normal+Abnomal data로 테스트



https://towardsdatascience.com/a-comprehensive-beginners-guide-to-the-diverse-field-of-anomaly-detection-8c818d153995

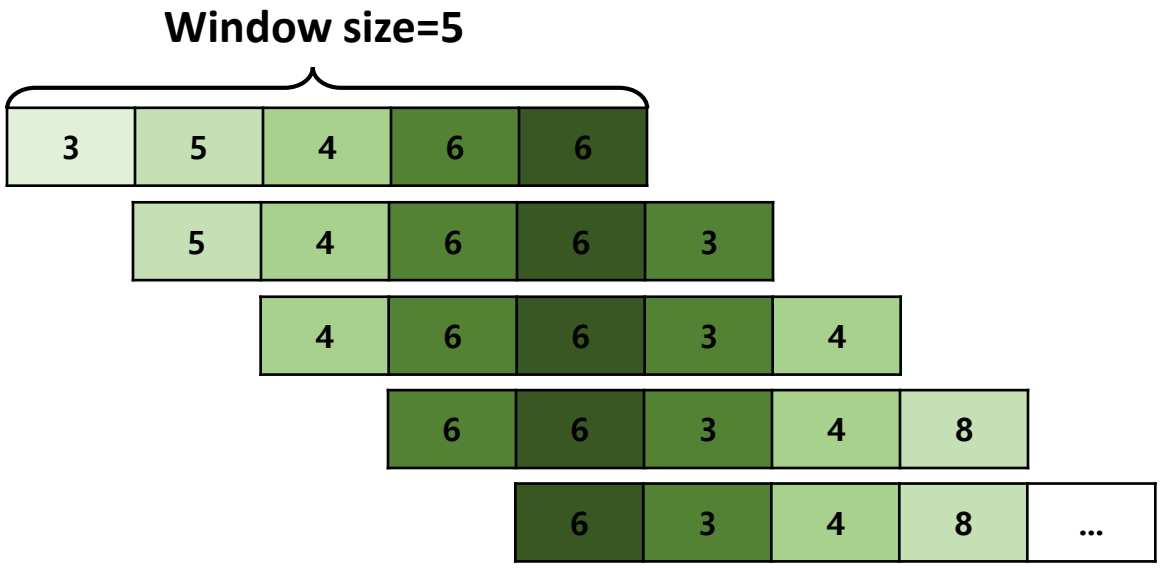
Classification VS Anomaly Detection

- **Severe data imbalance**
  - minority class samples이 조금은 있을 때 (100개 이상)
    - Classification with Oversampling
  - minority class samples이 너무 적을 때(5~10개)
    - Anomaly Detection!!

Data setup

- **Sliding window**
  - Long sequence to small window sequence
  - Sliding window size: 5

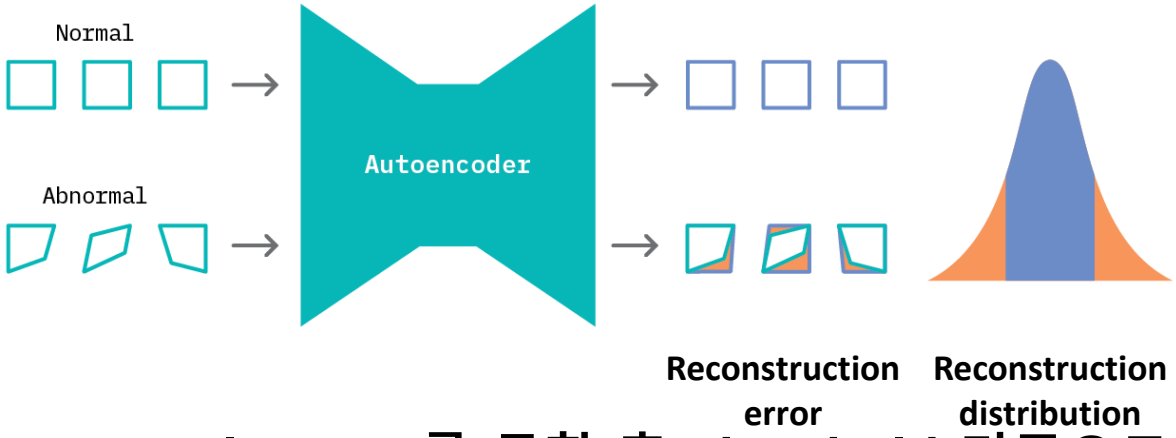
Time stamp	1	2	3	4	5	6	7	8	…
Value	3	5	4	6	6	3	4	8	…



Anomaly Detection Phases

- **Phase 1) 데이터 분포 학습**
  - Seq2Seq

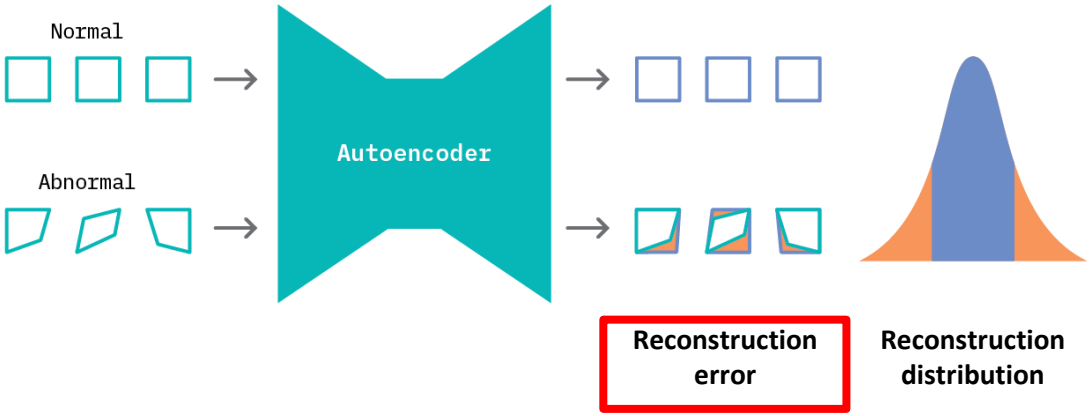
  - AutoEncoder

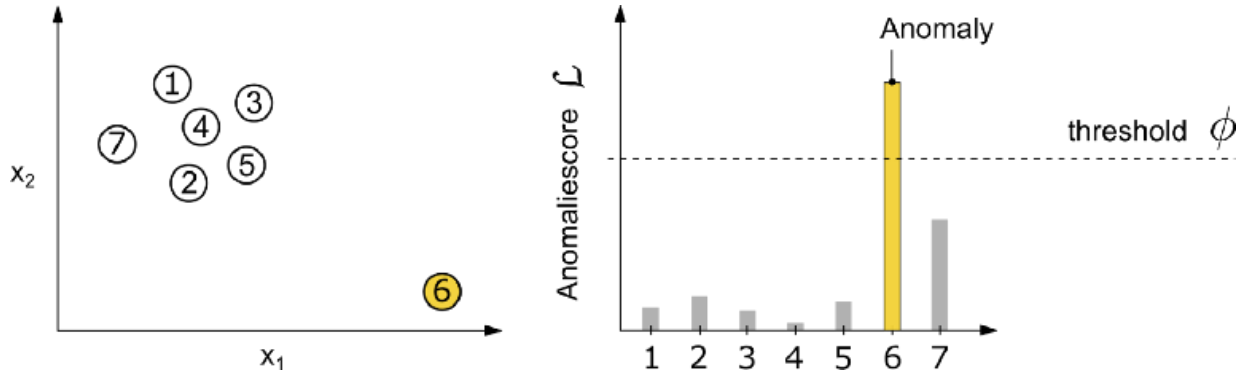


- **Phase 2) Anomaly score를 구한 후 Threshold 기준으로 판별**

Anomaly Detection Phases

- **Phase 1) 데이터 분포 학습**
- **Phase 2) Anomaly score를 구한 후 Threshold 기준으로 판별**

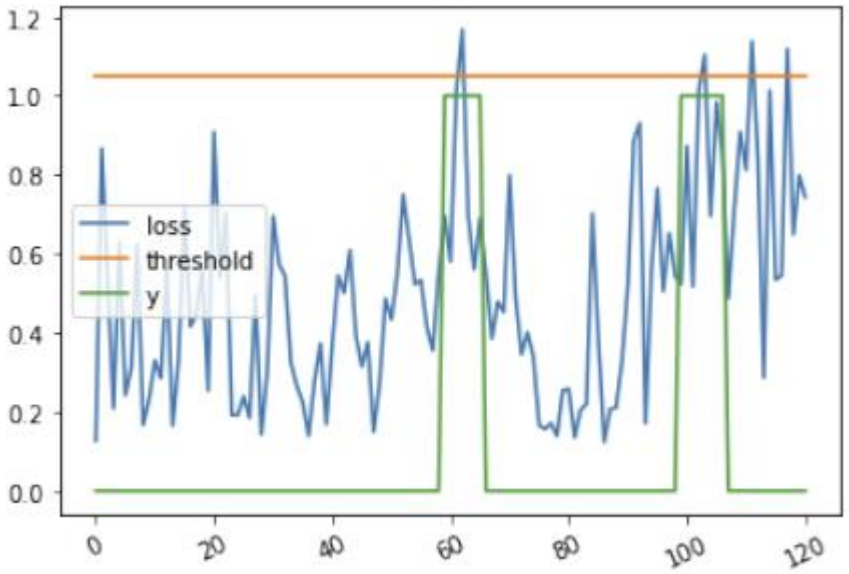


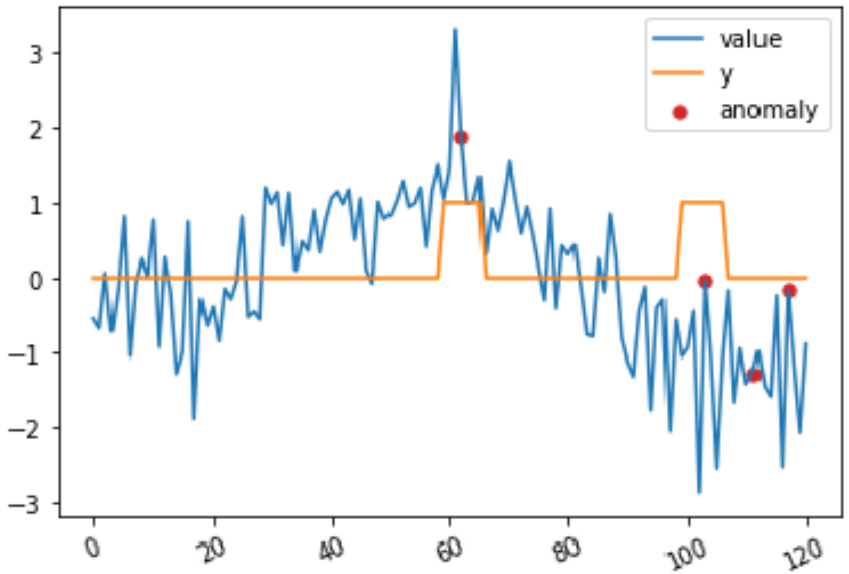


https://towardsdatascience.com/a-comprehensive-beginners-guide-to-the-diverse-field-of-anomaly-detection-8c818d153995

Anomaly Detection Phases

- **Phase 1) 데이터 분포 학습**
- **Phase 2) Anomaly score를 구한 후 Threshold 기준으로 판별**



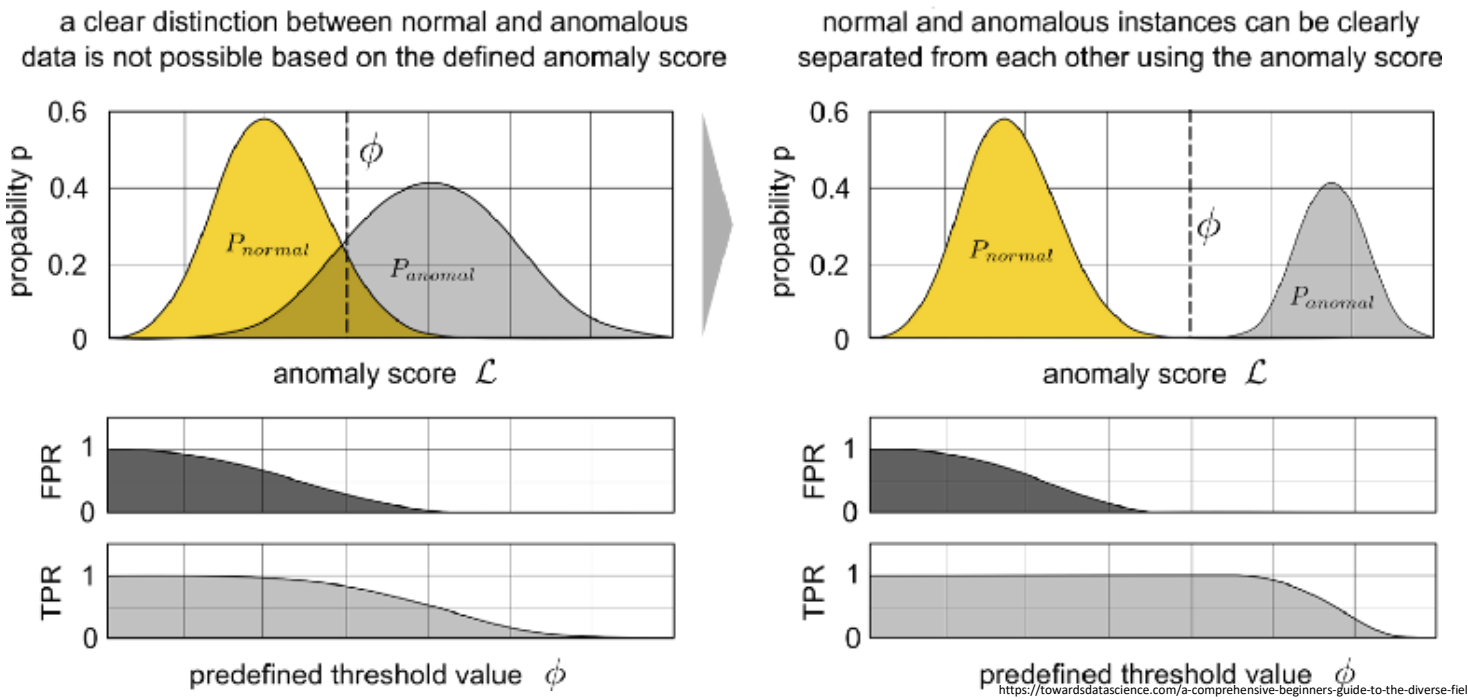


https://arxiv.org/pdf/2008.09567.pdf

Threshold

- The threshold for the anomaly score defines the sensitivity of the system
- Designation of instances as normal/anomalous based on their anomaly score and the predefined threshold value

P_{normal}	Probability distribution of normal data	TPR	True Positive Rate
P_{anomal}	Probability distribution of anomalous data	FPR	False Positive Rate



https://towardsdatascience.com/a-comprehensive-beginners-guide-to-the-diverse-field-of-anomaly-detection-8c818d153995

Challenges

- Normal과 abnormal data 사이의 경계가 모호 (Gray area)
- 연속한 데이터에서 어디를 경계로 설정할지 결정하기 어려움
- Normal과 abnormal을 명확히 구분하는 설명이 어려움
- 시간이 흘러 새로운 정상 패턴이 생겼을 때, 과거 데이터로는 비정상이라고 탐지할 확률이 높음

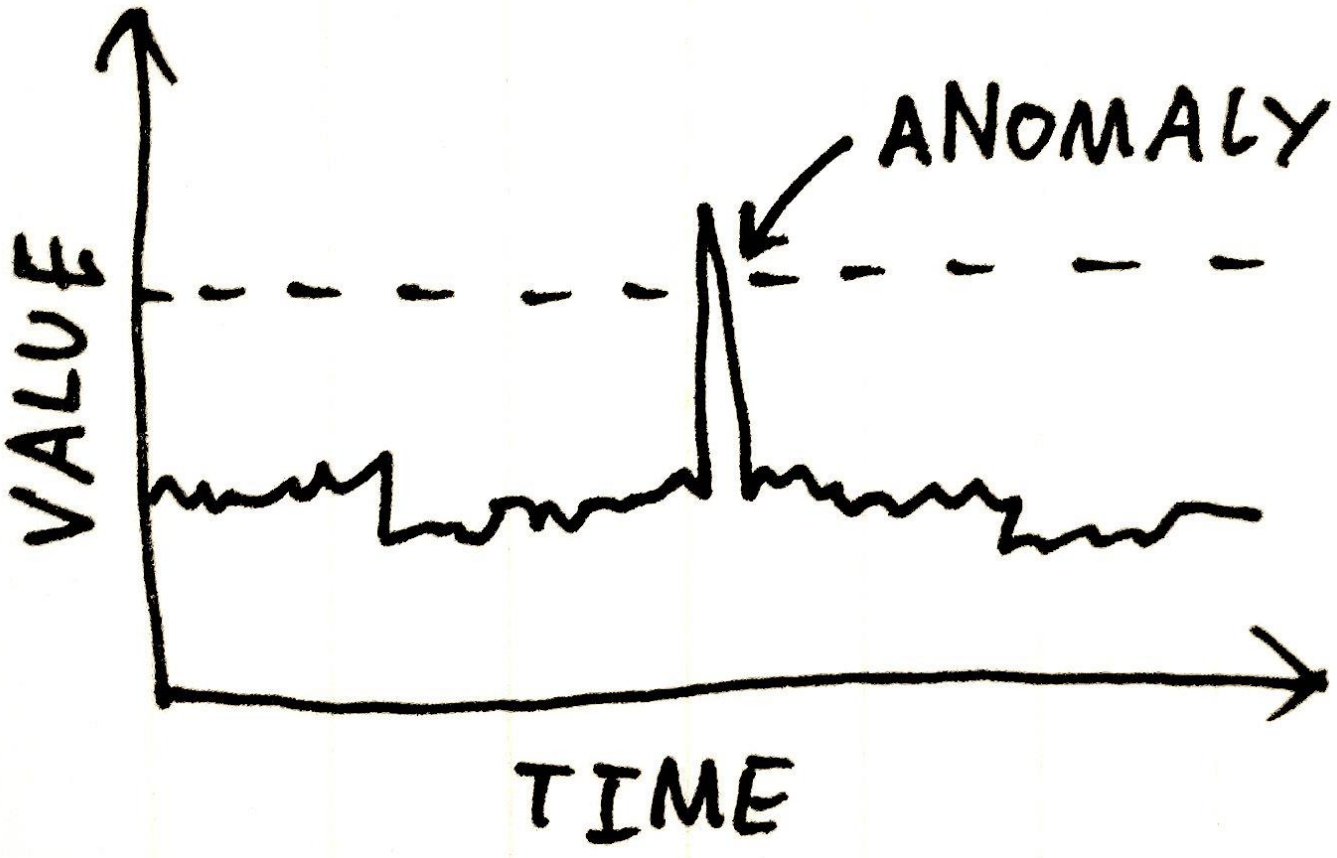
Image Anomaly Detection

AI Visual Inspection

https://zindi.africa/learn/introduction-to-anomaly-detection-using-machine-learning-with-a-case-study

Image Anomaly Detection?

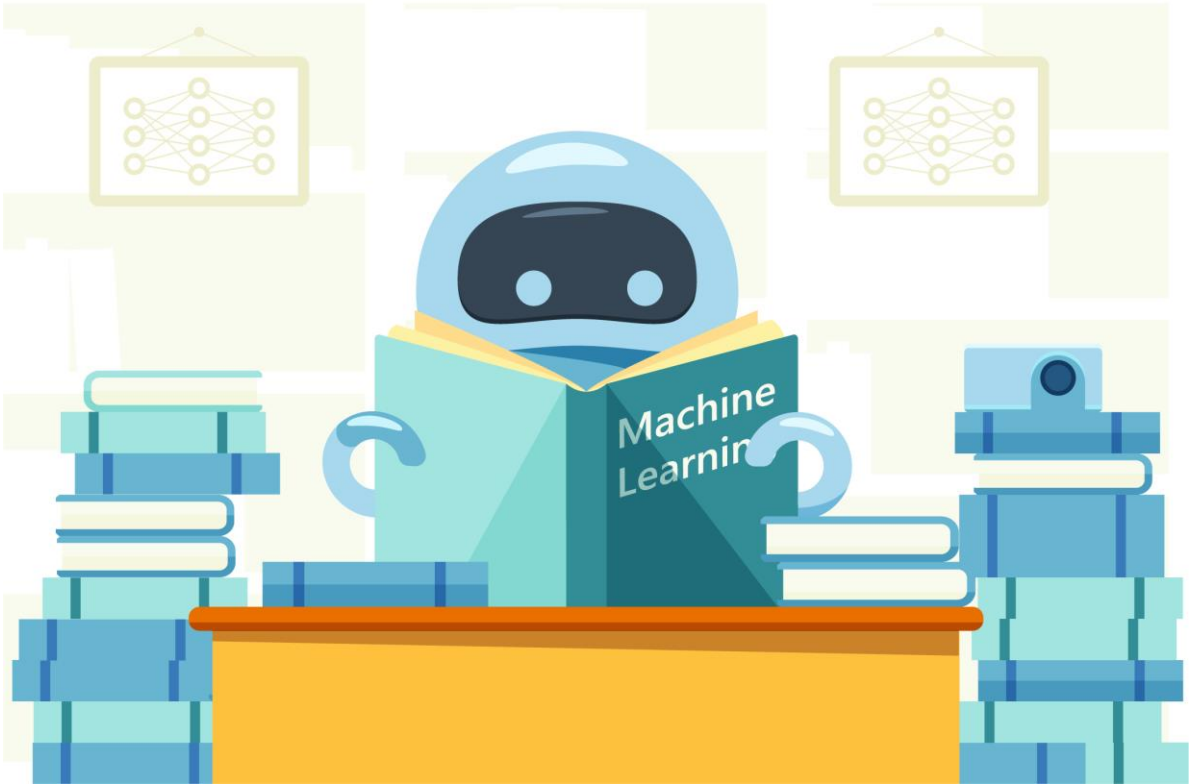
- 기존과 상이한, 정상이 아닌 데이터를 찾는 것
- 정상인지 이상인지 어떻게 알지?



인간의 이상탐지 프로세스는?

- 경험, 그리고 경험에 의한 직관을 바탕으로한 이상탐지
- 알고리즘으로?

Thanks to Machine Learning



정상의 사전적 의미

- 정상 : 특별한 변동이나 탈이없이 제대로인 상태

- Normal : usual or expected state, level, amount, etc

비정상의 사전적 의미

- 비정상 : 정상이 아님.
- 이상 : 정상적인 상태와 다름.
- Abnormal : different from what is normal or average.

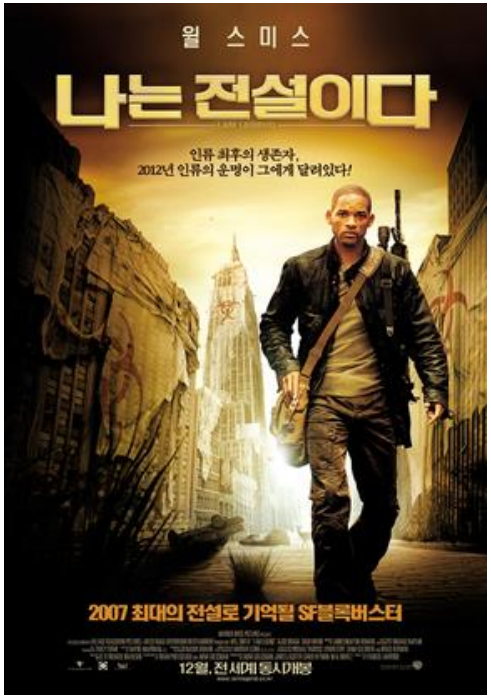
어떤 것이 정상일까요?

사람

감염된 좀비

정말인가요?

- 바이러스에 의해 대부분의 사람들이 흡혈귀로 변함.
- 살아남은 주인공은 낮동안 흡혈귀들을 사냥함.
- 지성이 있는 흡혈귀 집단의 존재 → 신인류의 등장
- 주인공 관점 : 흡혈귀 사냥
- → 흡혈귀 관점. : 잠든 사이 펼쳐지는 살인사건



I am a legend

- 최후의 사람인 주인공이 마지막으로 남긴 말 : "나는 전설이다."
- 내가 그들의 입장에서는 드라큘라와 같은 전설 속의 존재이다.
- 나만이 정상인 줄 알았지만 알고보니 나만이 비정상이었다.

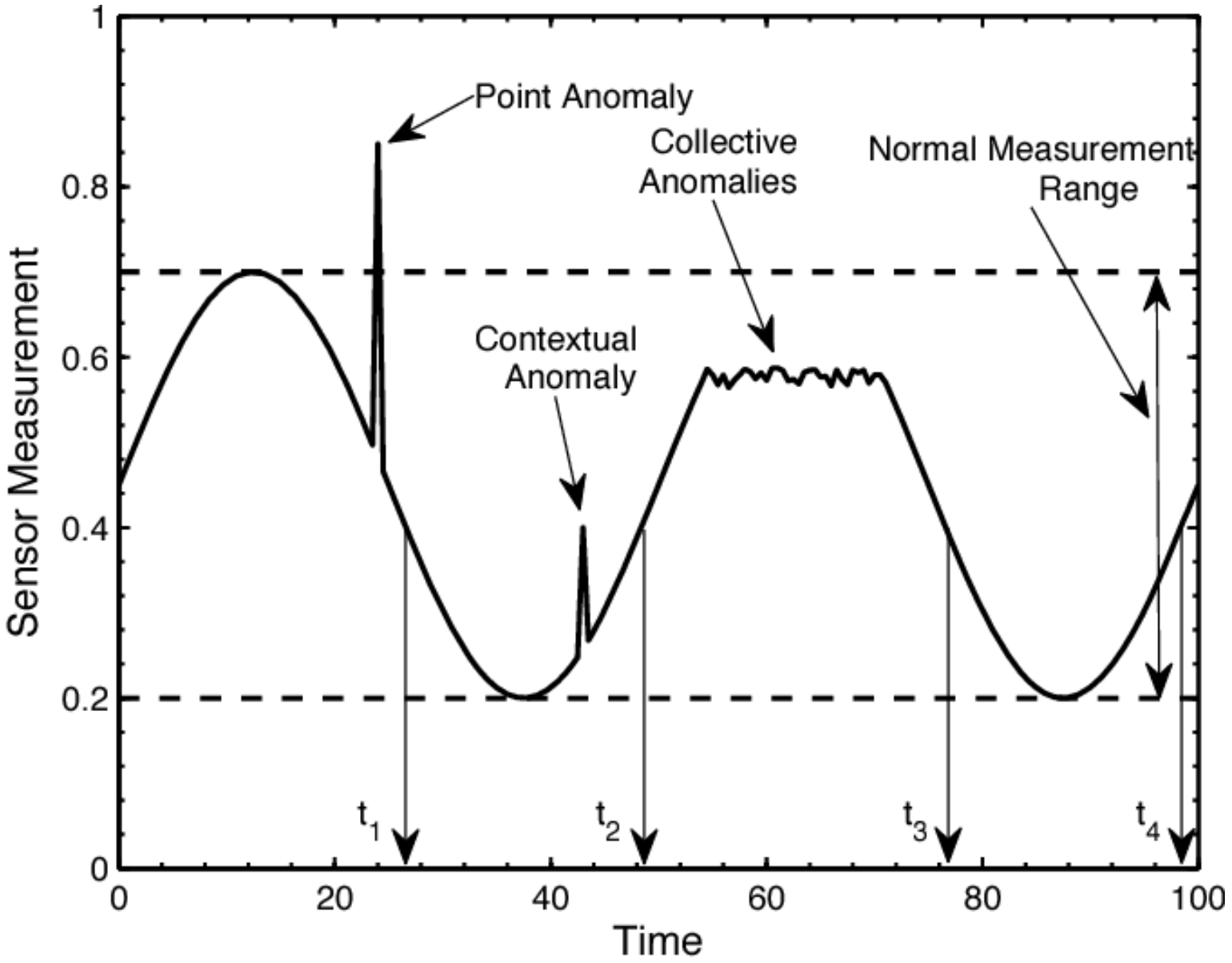
"정상이란 상대적이며 변화하는 개념"

Anomaly detection

- 정상과는 뭔가가 다르다.
  - 사전적 정의와 일맥상통
- 정량적으로 명확하게 정의되지 않는다.
  - Deviate so much
  - Occur very rarely
  - Relatively distant

Anomaly의 분류

- Point anomaly
  - 개별 데이터포인트가 다른 데이터에 비해 비정상일 때.
- Contextual (conditional) anomaly
  - 가능한 값이나 맥락 상 비정상인 경우
  - e.g., 한여름에 눈이 올 때
- Collective (group) Anomaly
  - 개별 데이터포인트는 가능한 값이나 모였을 때 비정상인 경우
  - e.g., 비가 한달 내내 오는 경우

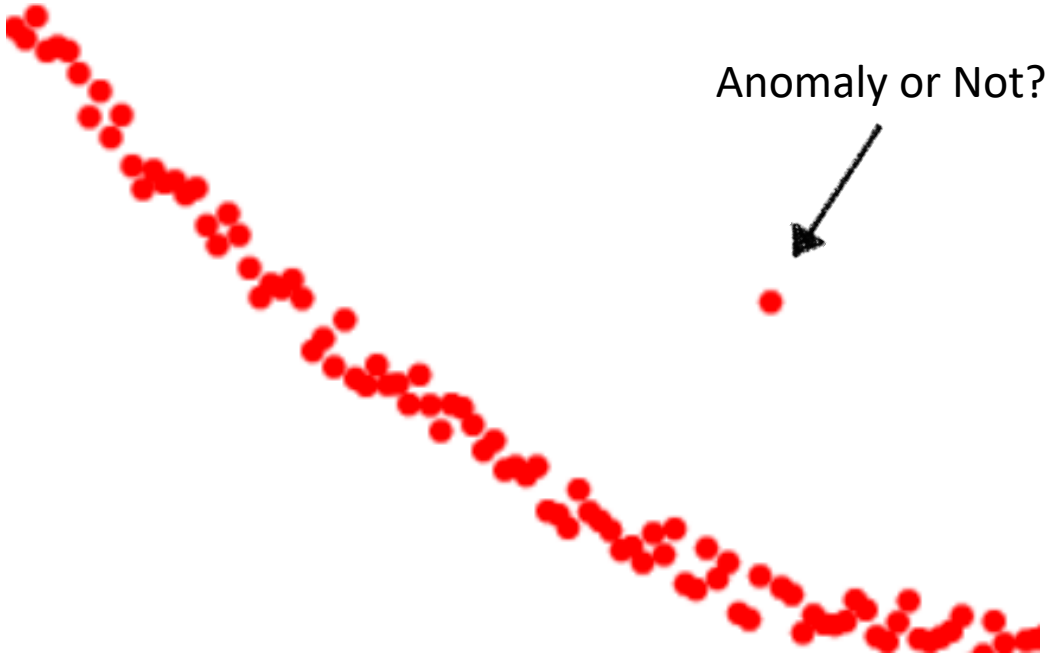


(c) Point, Contextual and Collective Anomalies [10]

어떻게 정상과 비정상을 구분할수 있을까?

- 정상과 비정상의 구분은 결국 정상과 비정상의 이진 분류 문제
- 이상의 정도를 정량화, 수치화 할 수 있다면?
  - 이상정도가 높으면 이상, 낮으면 정상..
  - Threshold을 통한 정상과 비정상의 이진 분류 수행 가능.

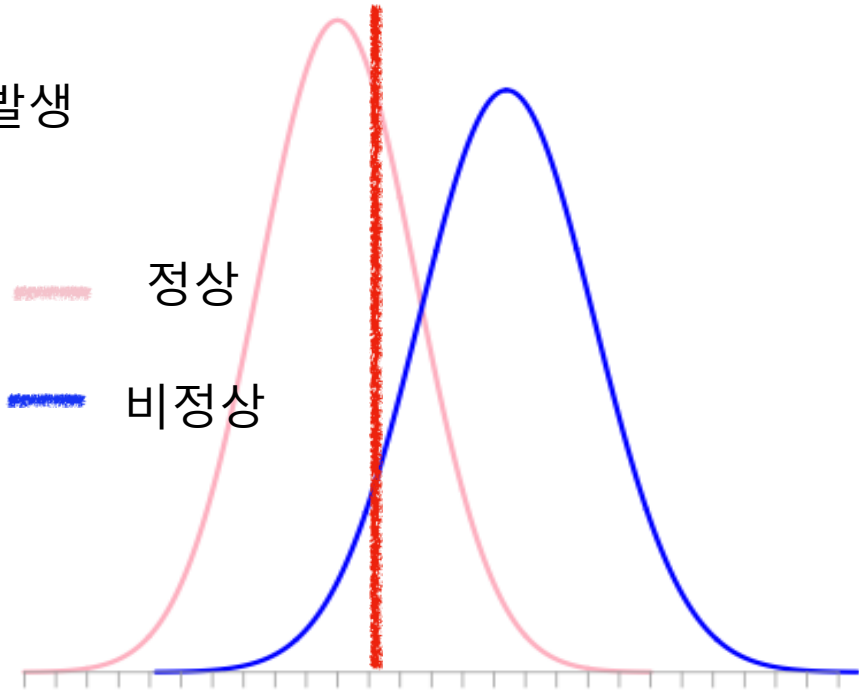
무엇을 활용 할 수 있을까?



- 확률
  - 정상의 확률 분포를 모델링
- 거리 = 정상과의 차이
  - 인접 포인트 : k-NN
  - 인접 그룹 : k-mean, DBSCAN
  - 복원값 : 오토인코더, PCA

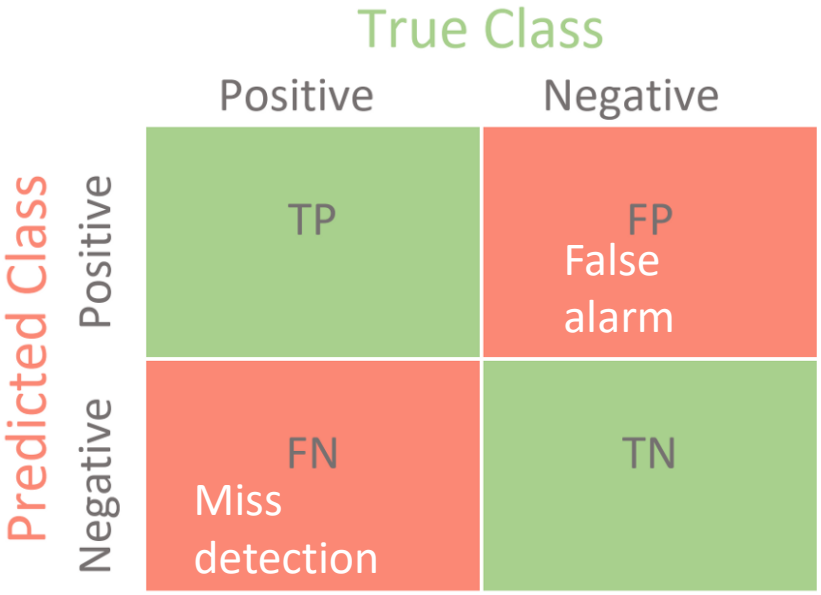
이상 점수를 얻으면?

- Threshold를 통한 이진 분류 수행
  - 이상점수 > Threshold → 비정상
  - 이상점수 < Threshold → 정상
- 고차원 벡터가 스칼라로 변환
  - → 점수 분포의 overlap
- Threshold에 따라서 불가피한 Tradeoff 발생
  - → 이상 탐지 모델의 성능과 연관



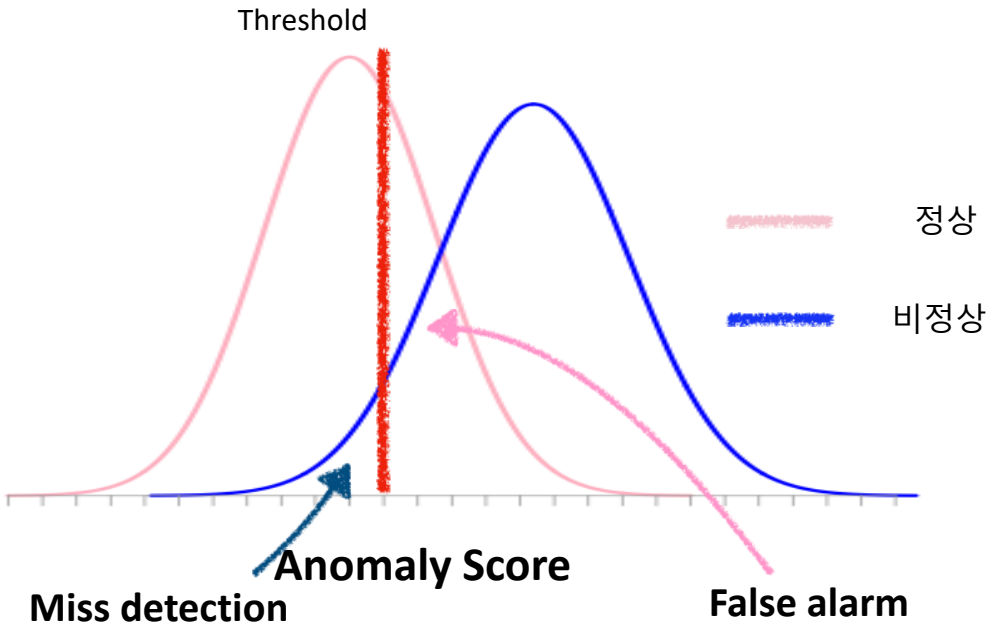
이상 탐지 모델의 평가

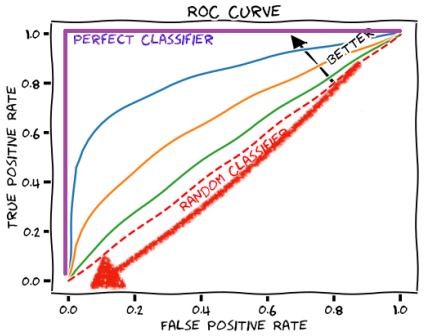
- True Positive (TP) : 실제 이상을 이상으로 분류.
- True Negative (TN): 실제 정상을 정상으로 분류.
- False Positive (FP): 실제 정상을 이상으로 분류, **False alarm**.
- False Negative (FN): 실제 이상을 정상으로 분류, **Miss detection**.



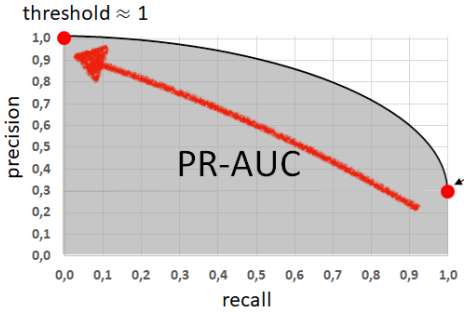
Confusion Matrix

ROC 곡선과 Precision-Recall 곡선





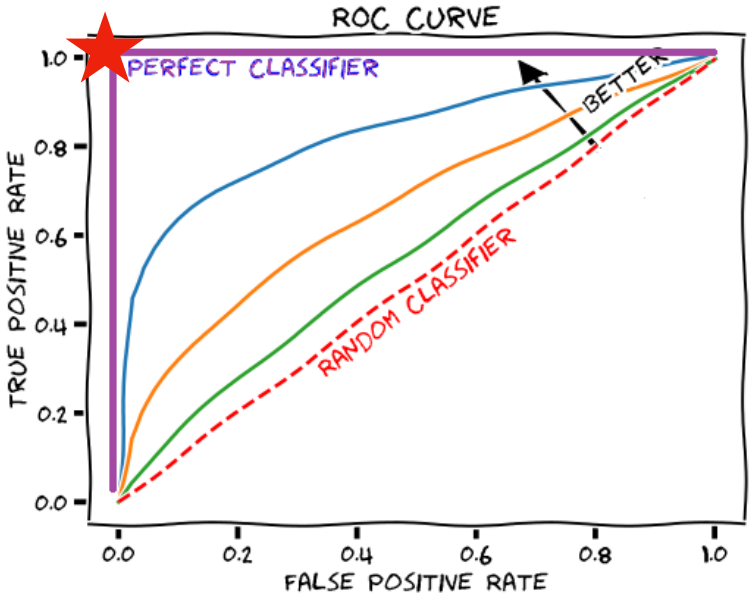
ROC



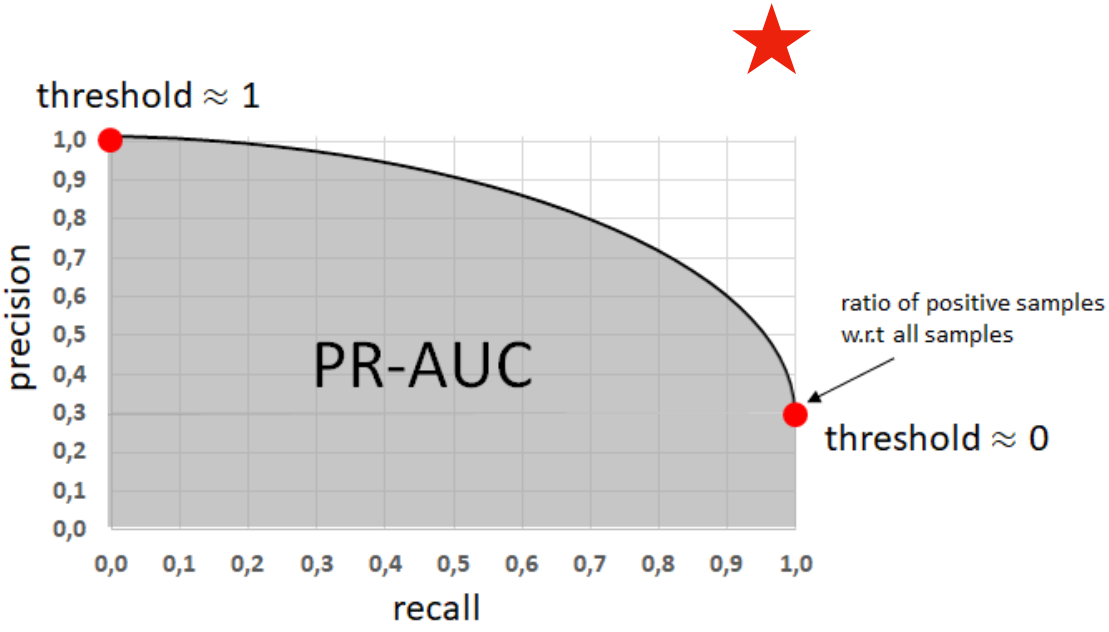
PRC

완벽한 이상탐지 모델의 성능은?

- TPR(Recall)=1, FPR = 0, Precision=1



ROC curve and AUROC (AUC)

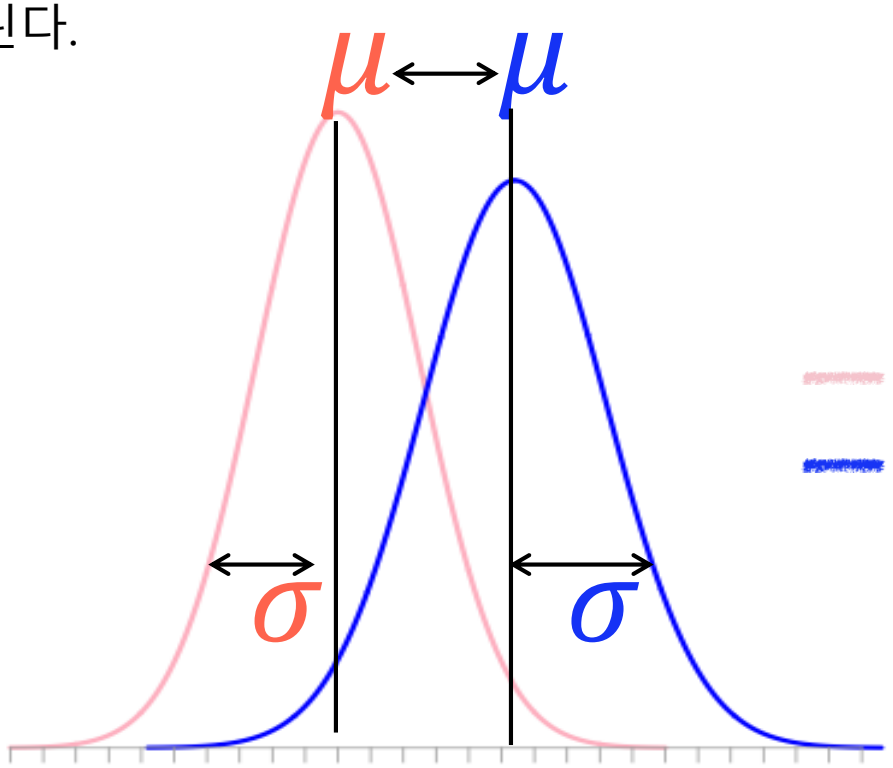


PR curve and PR-AUC

좋은 성능을 얻기위해서는?

- 특징 변환 $\phi \rightarrow$ 이상 점수 산정 τ 로 이어지는 비선형 변환
- 결과적으로 **데이터 분포를 이상점수 분포로 manipulation**

- 두 분포의 평균을 떨어뜨린다.
- 두 분포의 분산을 줄인다.



Summary

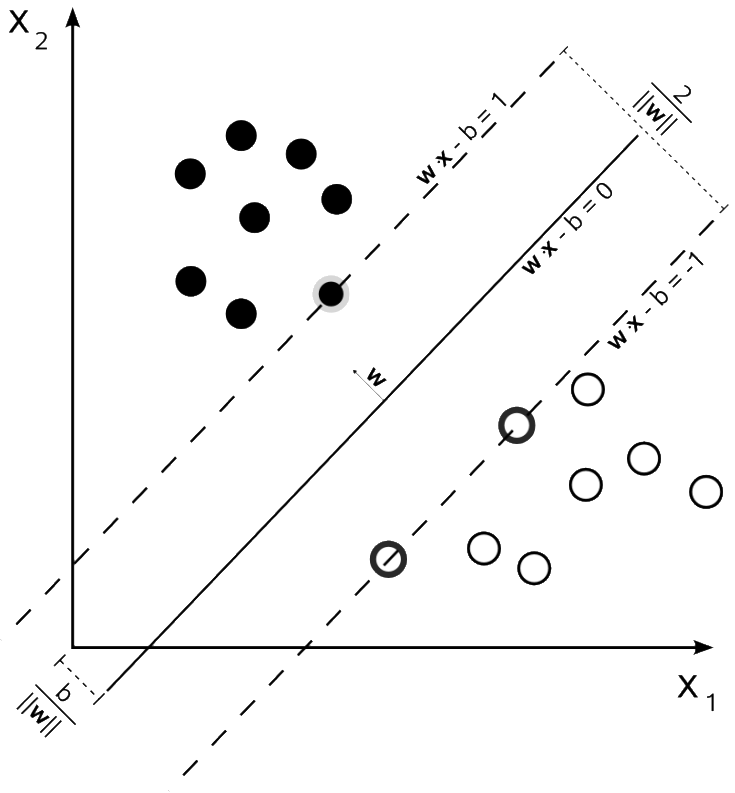
- 정상은 상대적으로 변화하는 것. 따라서 목표를 명확하게 설정하자.
  - 정상을 정의하는데 있어서 도메인 지식이 필수적.
- 현상 (관측값) 밑에 숨겨진 컨텍스트를 파악해야 이상을 잘 탐지 할 수 있음.
  - 이미지 데이터와 시계열 데이터 또한 마찬가지.
- 이상 점수의 분포와 이상 탐지의 사이의 관계를 이해 해야함.

Deeplearning based Anomaly Detection

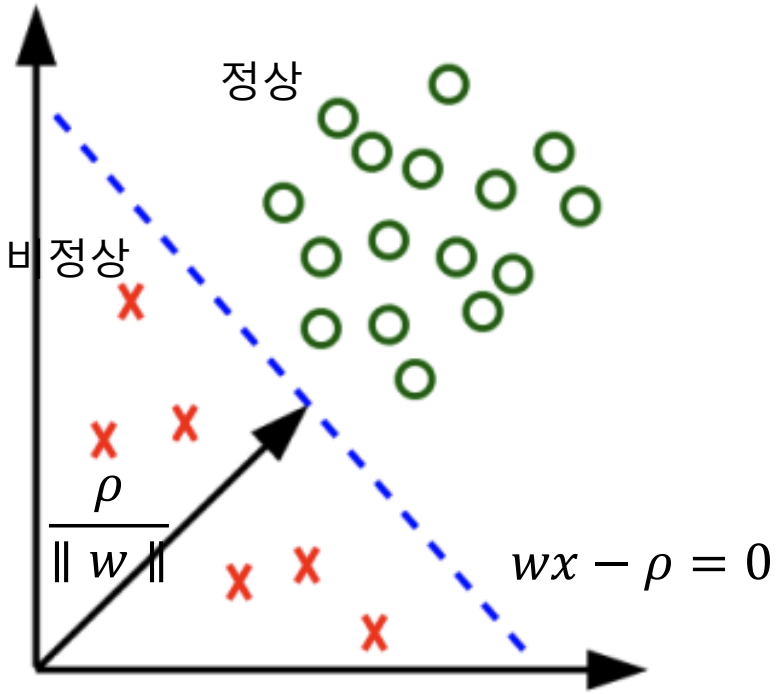
"정상 데이터는 잘하게,
비정상 데이터는 못하게"

DSVDD : SVM (Recap)

SVM



One Class-SVM



https://arxiv.org/pdf/1901.08688v1.pdf

DSVDD: SVM/SVDD to deep learning model

- DSVDD : DNN + SVDD
- Feature space 상에서 OC-SVM, SVDD 수행 → 커널 트릭을 활용
- DSVDD : Deep neural network를 통해 feature space로 mapping

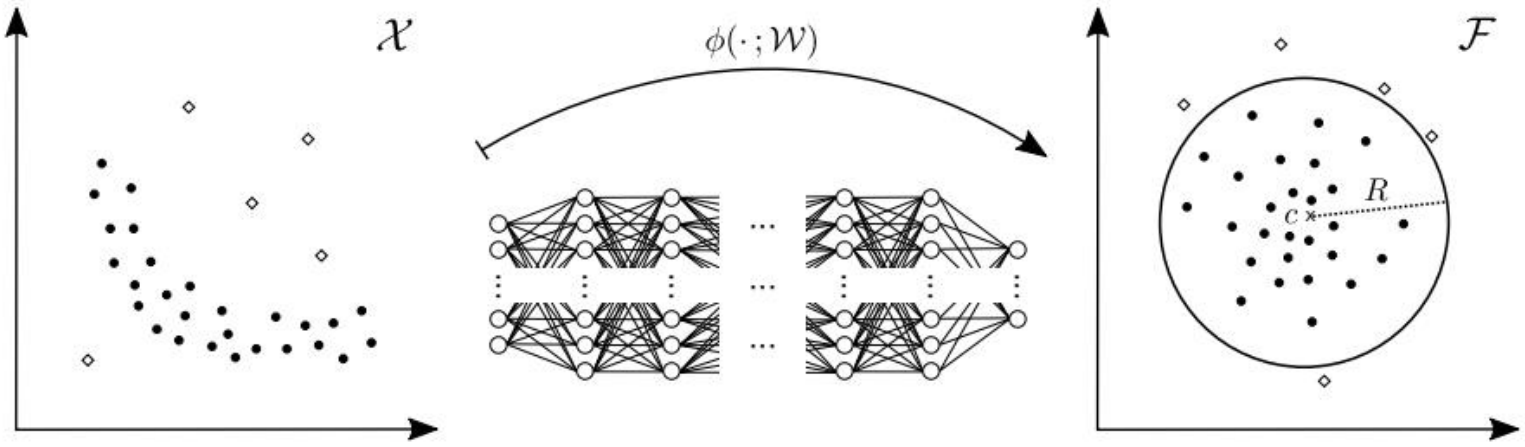
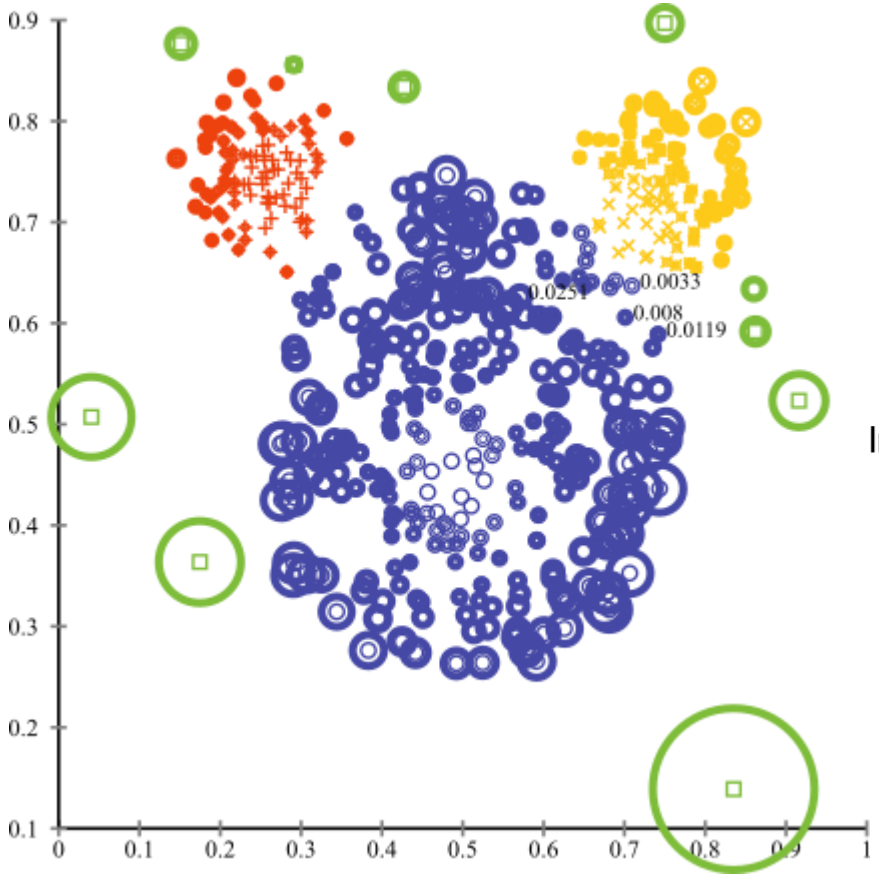
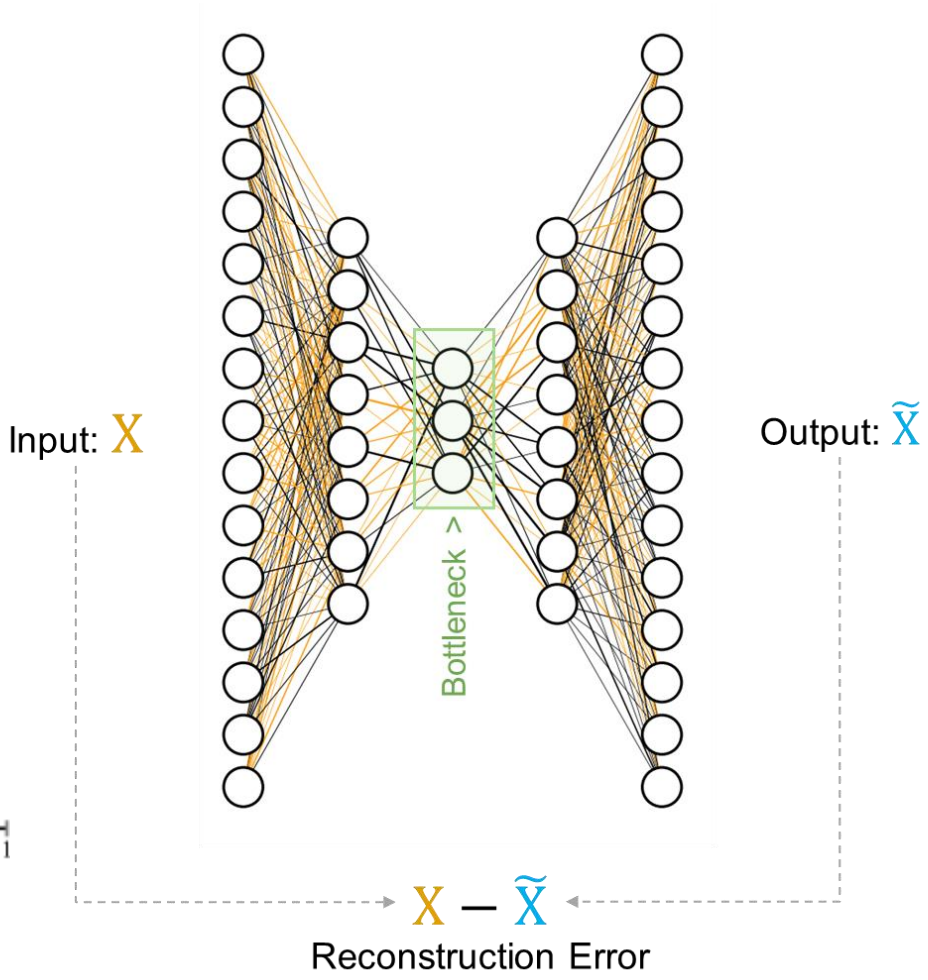


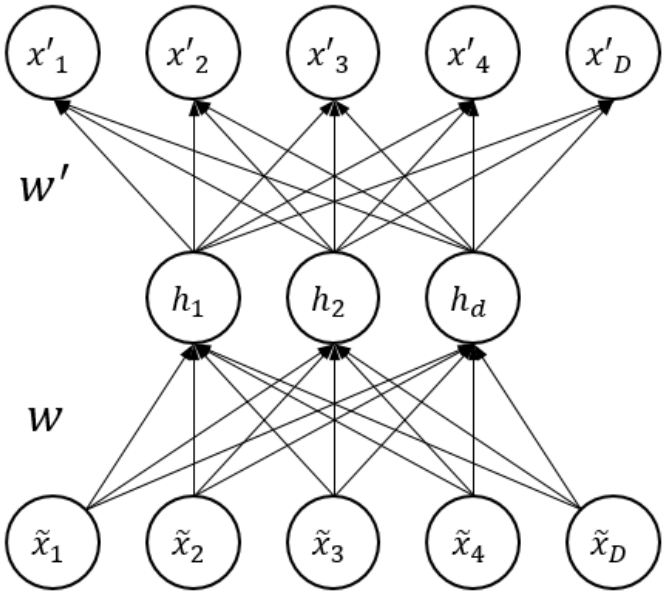
Figure 1. Deep SVDD learns a neural network transformation $\phi(\cdot;\mathcal{W})$ with weights $\mathcal{W}$ from input space $\mathcal{X} \subseteq \mathbb{R}^d$ to output space $\mathcal{F} \subseteq \mathbb{R}^p$ that attempts to map most of the data network representations into a hypersphere characterized by center $\boldsymbol{c}$ and radius R of minimum volume. Mappings of normal examples fall within, whereas mappings of anomalies fall outside the hypersphere.

Auto-Encoder





Auto-Encoder



$\tilde{x}$ = x + noise

Visual Inspection Status in Industrial Area

- **Labor Shortage**
  - Number of Workers in Visual Inspection: 1.4M in Japan (10~20%)

- **Variation in inspection result**
  - Depends on individuals
  - Human error
  - Cost in education

Challenges

- **Failure Rate： 0.01%**
- **Only 1 or 2 of fails occurs a day**

Main Business Field

Semiconductor

Display

Electronics

Car

Plastic

Steel

Paper

Medicine

Textile

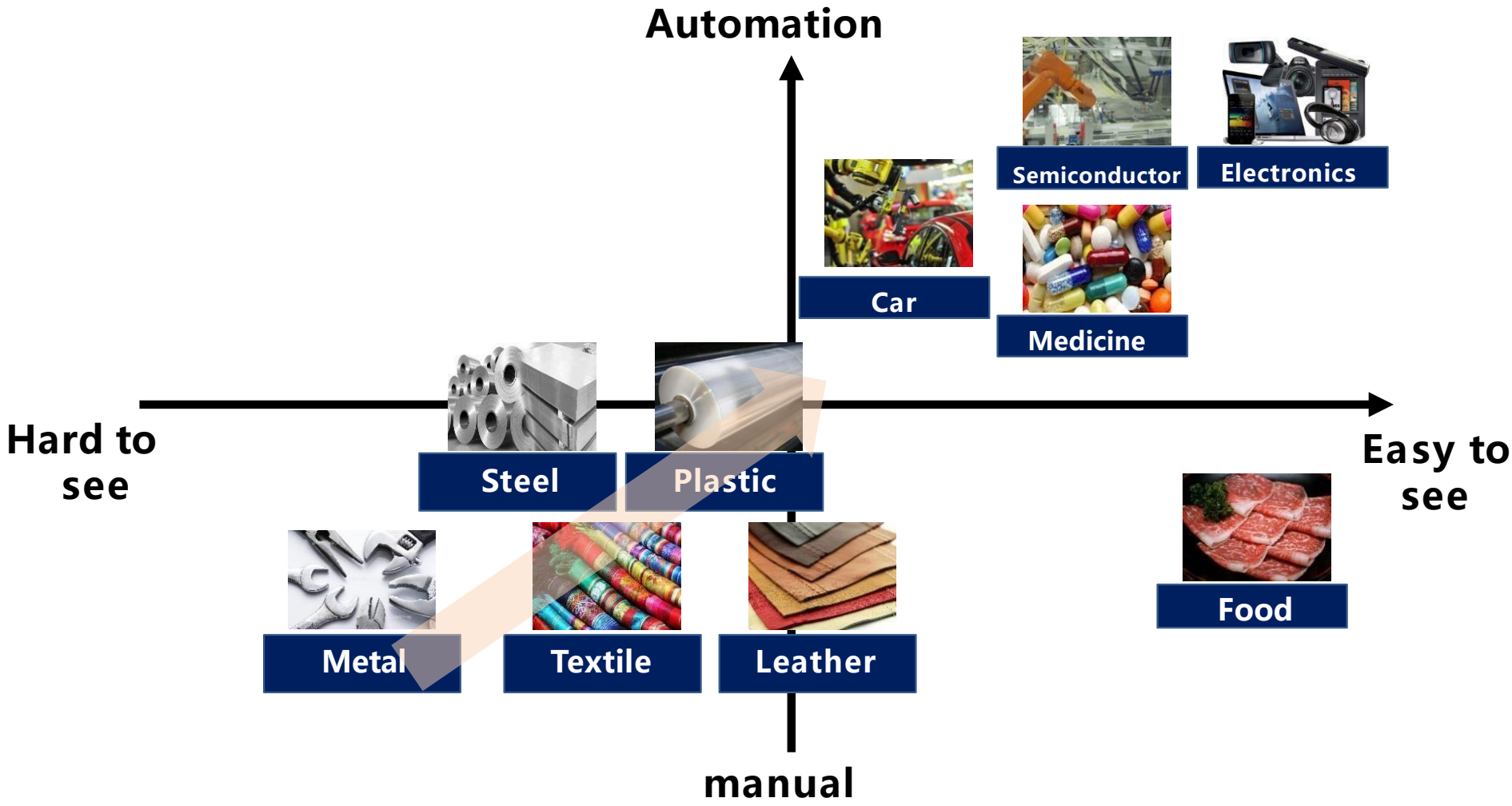
Leather

Food

Airplane

World Bank, U.S. Department of Labor

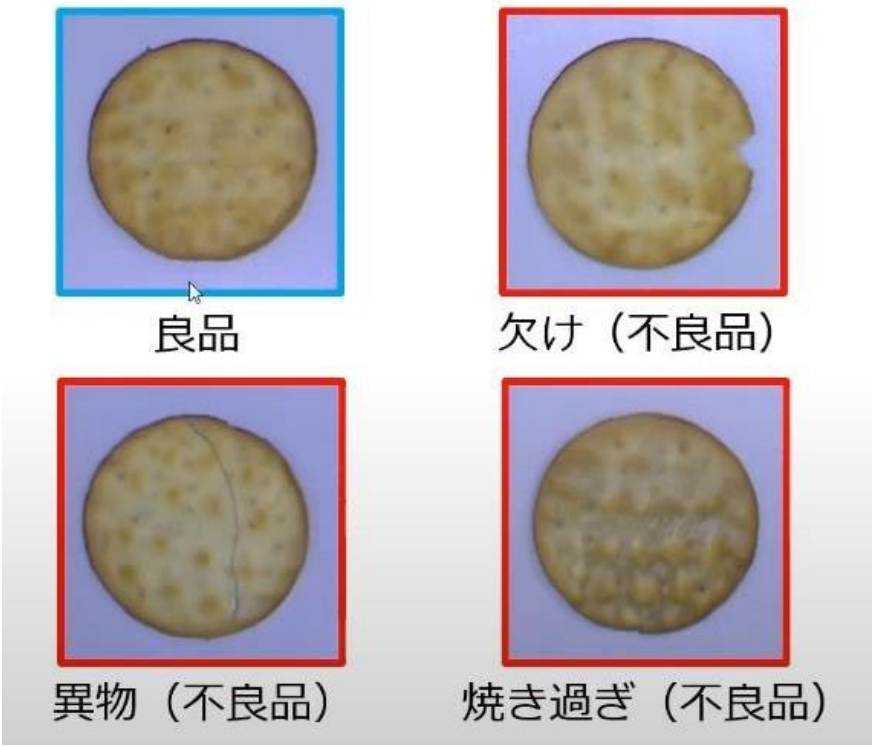
Main Business Field



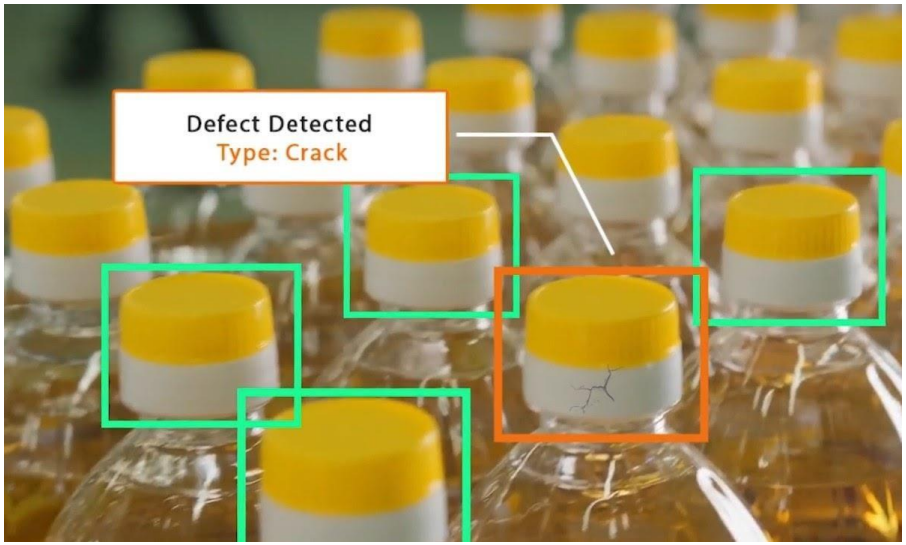
World Bank, U.S. Department of Labor

Four Main Algorithm for Visual Inspection

1.Classification



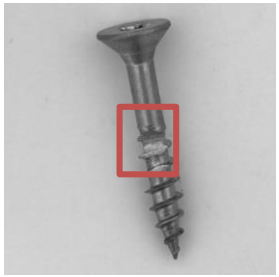
2.Object Detection

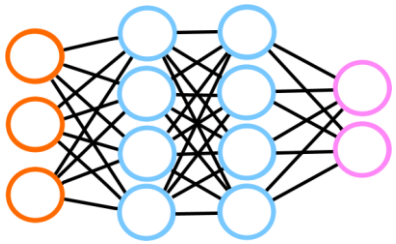


Four Main Algorithm for Visual Inspection

3.Segementation

4. Anomaly Detection

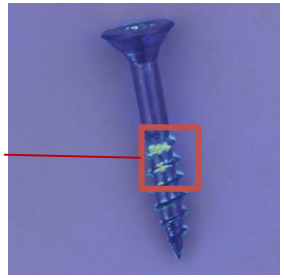




Input

Output

Features can not be generated = Defect



Anomaly map

Challenges

- Failure Rate : 0.01%
- Only 1 or 2 of fails occurs a day

- 1) Classification
- 2) Object Detection
- 3) Segmentation
- 4) Anomaly Detection

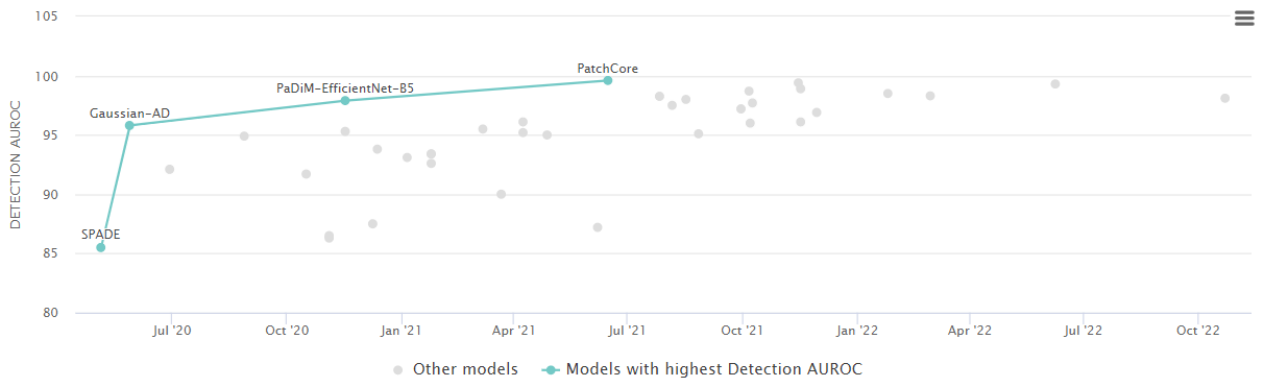
https://paperswithcode.com/sota/anomaly-detection-on-mvtec-ad

Anomaly Detection

Anomaly Detection on MVTec AD

Leaderboard Dataset

View Detection AUROC by Date for All models



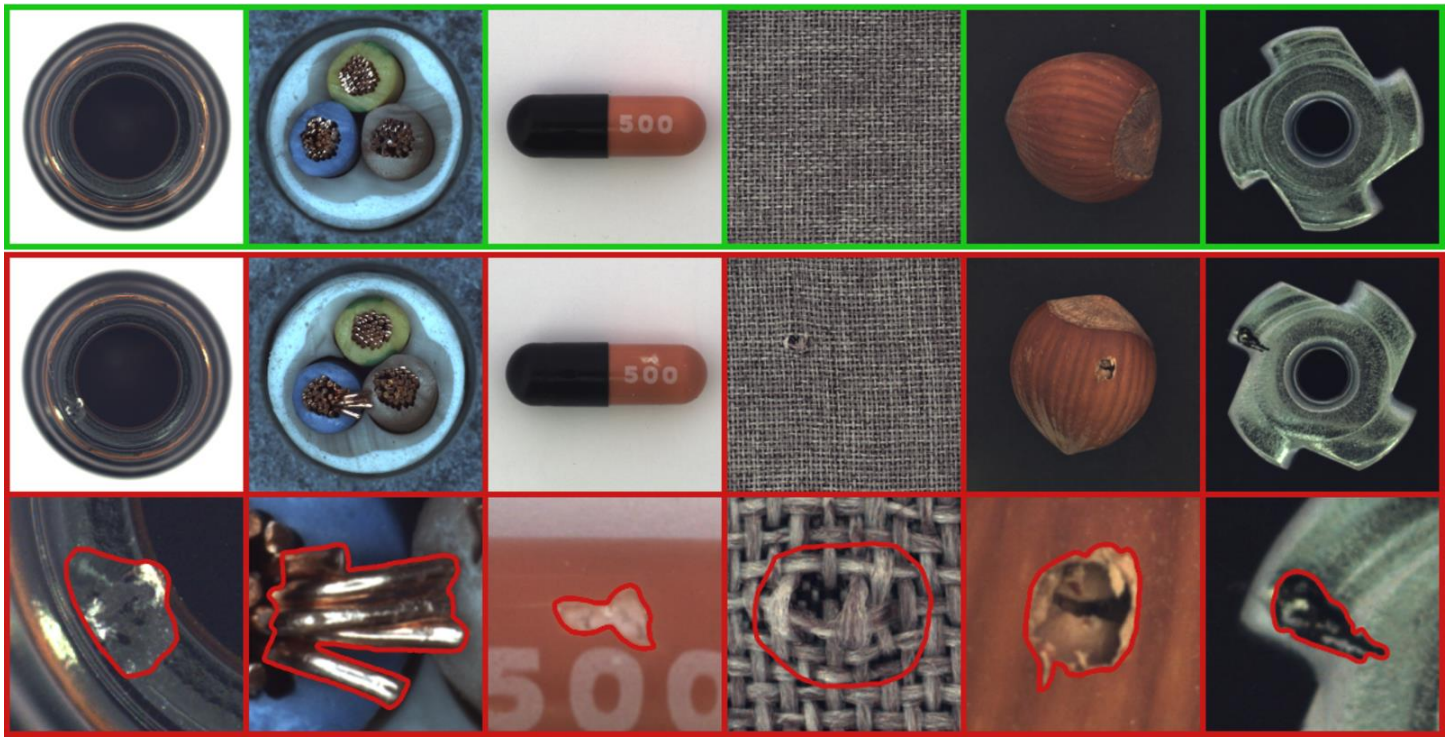
Filter: ResNet | Pre-trained on ImageNet | ImageSize256 | ResNet-50 | WideResNet | WideResNet-50-2 | EfficientNet | ImageSize256Cropped224 | ResNet-18 | Transformer | ResNet-34 | Pre-trained on CelebA | DCGAN | Unsupervised | UNet | EfficientNet-B5 | SuperPoint | D2-Net | AlexNet | VGG | HRNet | untagged

Edit Leaderboard

Rank	Model	Detection AUROC	Segmentation AUROC	Segmentation AUPRO	Extra Training Data	Paper	Code	Result	Year	Tags
1	**PatchCore**	99.6	98.2	94.9	✓	Towards Total Recall in Industrial Anomaly Detection			2021	ResNet-50, WideResNet, WideResNet-50-2
2	**Fastflow**	99.4	98.5		✓	FastFlow: Unsupervised Anomaly Detection and Localization via 2D Normalizing Flows			2021	ResNet, Transformer

Datasets : MVTec AD

- **Dataset:** https://www.mvtec.com/company/research/datasets/mvtec-ad



- Bottle (148 MB)
- Cable (481 MB)
- Capsule (385 MB)
- Carpet (705 MB)
- Grid (153 MB)
- Hazelnut (588 MB)
- Leather (500 MB)
- Metal Nut (157 MB)
- Pill (262 MB)
- Screw (186 MB)
- Tile (335 MB)
- Toothbrush (104 MB)
- Transistor (384 MB)
- Wood (474 MB)
- Zipper (152 MB)

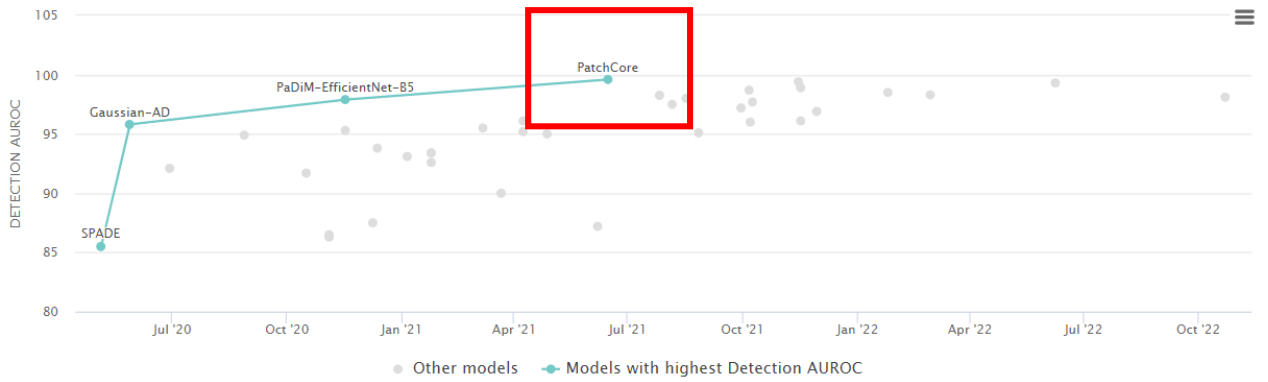
PatchCore

Anomaly Detection

Anomaly Detection on MVTec AD

Leaderboard Dataset

View Detection AUROC by Date for All models

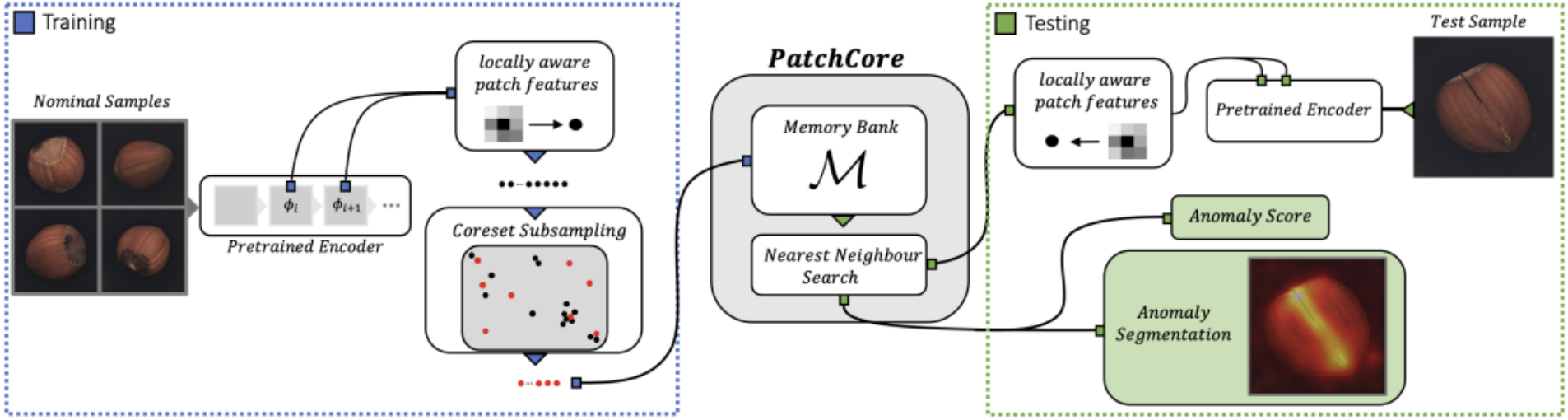


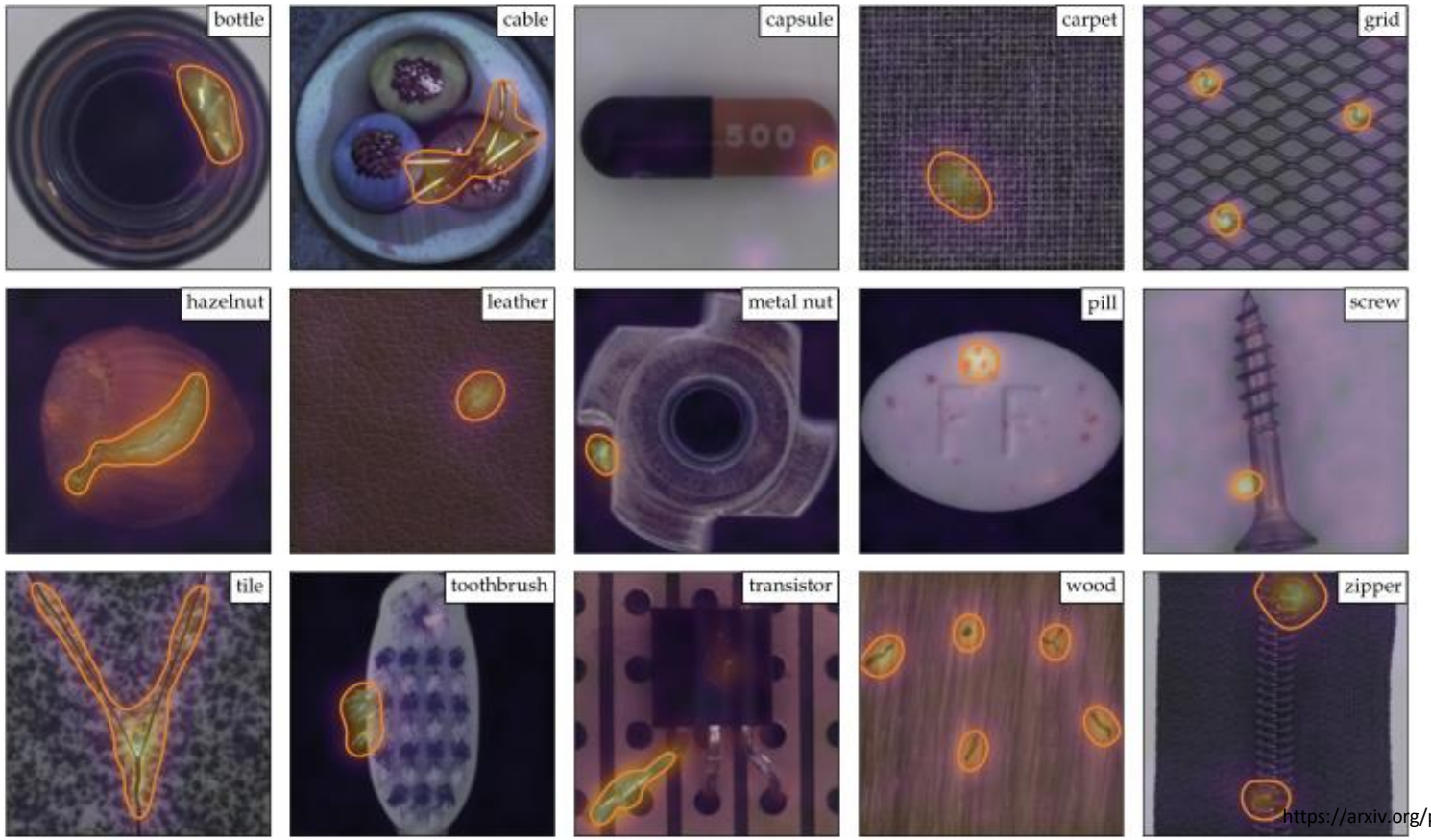
Filter: ResNet | Pre-trained on ImageNet | ImageSize256 | ResNet-50 | WideResNet | WideResNet-50-2 | EfficientNet | ImageSize256Cropped224 | ResNet-18 | Transformer | ResNet-34 | Pre-trained on CelebA | DCGAN | Unsupervised | UNet | EfficientNet-B5 | SuperPoint | D2-Net | AlexNet | VGG | HRNet | untagged

Edit Leaderboard

Rank	Model	Detection AUROC	Segmentation AUROC	Segmentation AUPRO	Extra Training Data	Paper	Code	Result	Year	Tags
1	**PatchCore**	99.6	98.2	94.9	✓	Towards Total Recall in Industrial Anomaly Detection			2021	ResNet-50, WideResNet, WideResNet-50-2
2	**Fastflow**	99.4	98.5		✓	FastFlow: Unsupervised Anomaly Detection and Localization via 2D Normalizing Flows			2021	ResNet, Transformer

https://arxiv.org/pdf/2106.08265v2.pdf

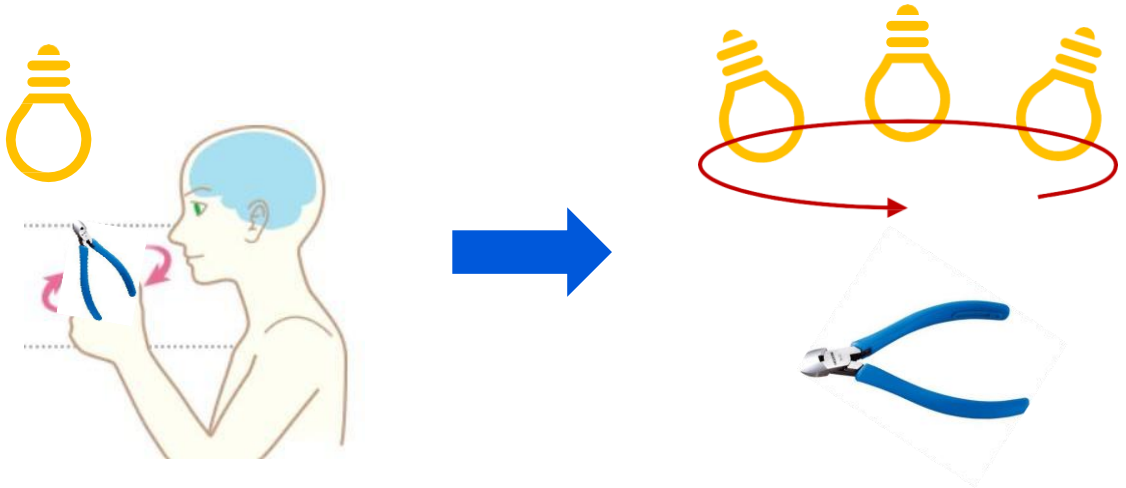




https://arxiv.org/pdf/2106.08265v2.pdf

Strategy

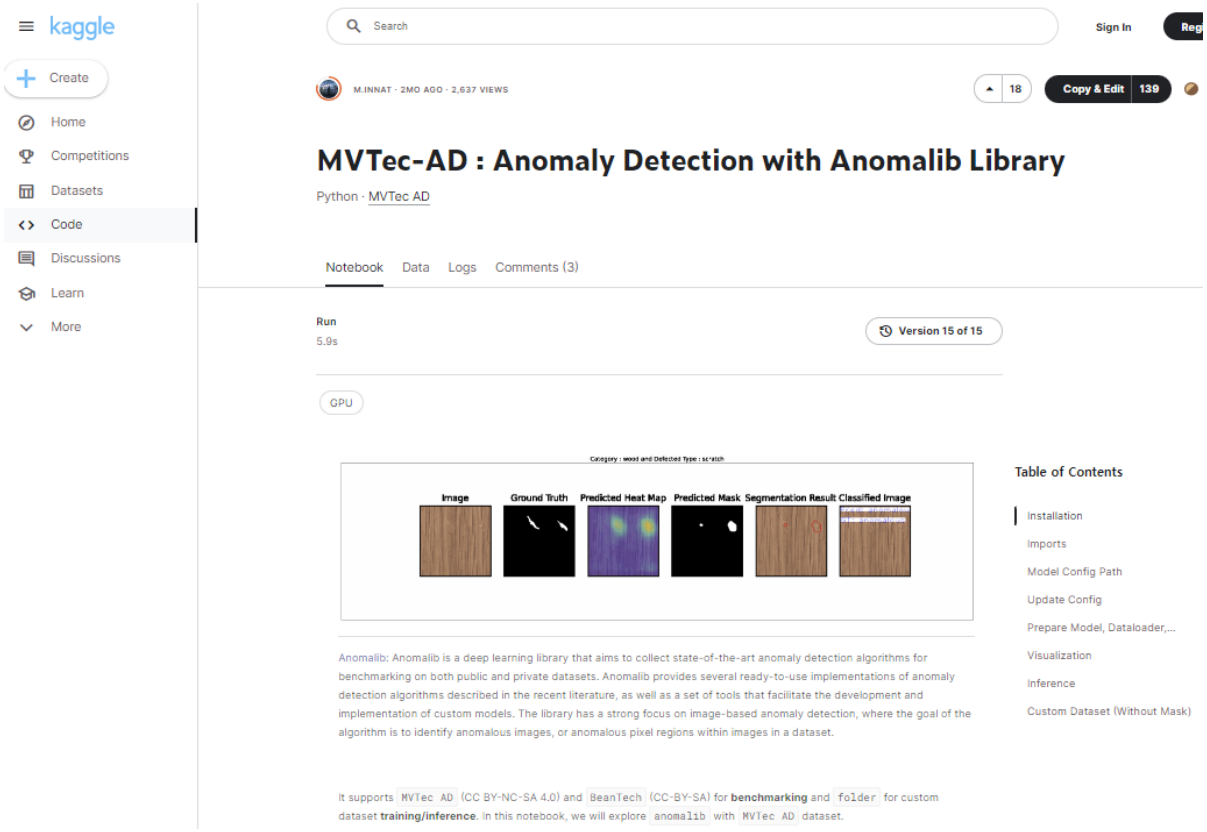
Image Processing



Deep Anomaly Detection

Code practice

- code: https://www.kaggle.com/code/ipythonx/mvtec-ad-anomaly-detection-with-anomalib-library



Model Config Path

Currently, there are **10** anomaly detection models available in `anomalib` library. Namely,

- Patchcore
- Padim
- DFKDE
- DFM
- CFlow
- Ganomaly
- STFPM
- FastFlow
- DREAM
- Reverse Distillation

Now, let's get their config paths from the respected folders.

```
CONFIG_PATHS = '/kaggle/working/anomalib/anomalib/models'
MODEL_CONFIG_PAIRS = {
    'patchcore': f'{CONFIG_PATHS}/patchcore/config.yaml',
    'padim':     f'{CONFIG_PATHS}/padim/config.yaml',
    'cflow':     f'{CONFIG_PATHS}/cflow/config.yaml',
    'dfkde':     f'{CONFIG_PATHS}/dfkde/config.yaml',
    'dfm':       f'{CONFIG_PATHS}/dfm/config.yaml',
    'ganomaly':  f'{CONFIG_PATHS}/ganomaly/config.yaml',
    'stfpm':     f'{CONFIG_PATHS}/stfpm/config.yaml',
    'fastflow':  f'{CONFIG_PATHS}/fastflow/config.yaml',
    'draem':     f'{CONFIG_PATHS}/draem/config.yaml',
    'reverse_distillation': f'{CONFIG_PATHS}/reverse_distillation/
config.yaml',
}
```

Reference

- AI프렌즈 세미나, 이미지 이상탐지 by tomomi research 최성훈
- AAiCON, 러닝을 활용한 이상탐지 기초이해, 류승형 박사

감사합니다