PIAI

Segmentation

Segmentation

PIAI Research Department

- Segmentation

(a) image

FLOW

1. **Semantic Segmentation**
   - Fully Convolutional Network(FCN)
   - U-Net
2. **Instance Segmentation**
   - Methods
   - YOLACT

1. **YOLACT**
   - Image Inference : Use eval.py
   - Video Inference : Use eval.py
   - Image Inference : Use YOLACT Modules

PIAI

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Semantic Segmentation

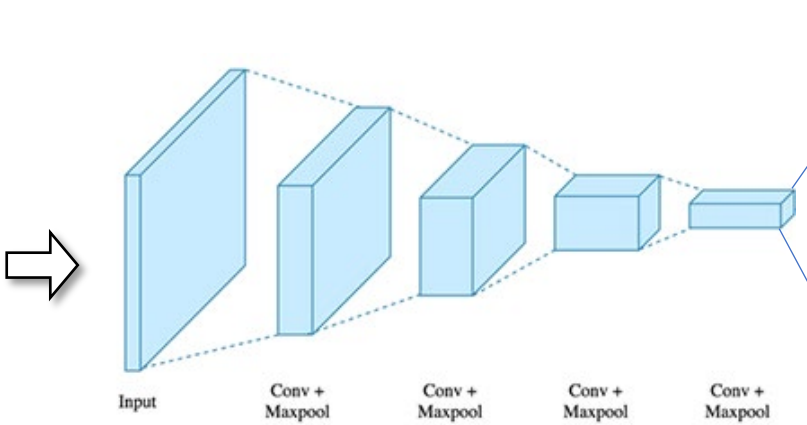
Dahyun, Kim

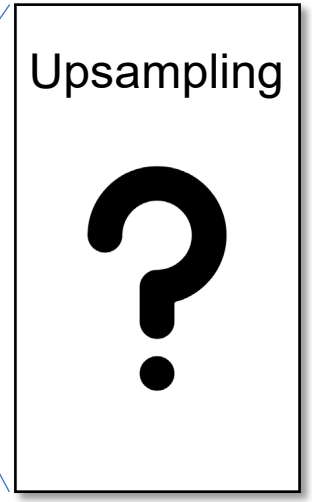
Semantic Segmentation

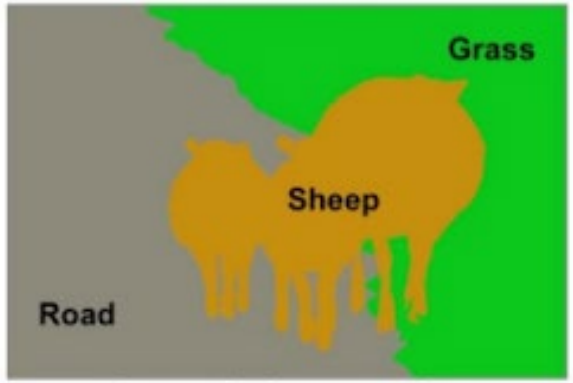
PIAI Research Department

- Semantic Segmentation
  - **이미지 내 모든 픽셀이 어느 클래스에 속하는지 예측하는 문제**

Classification + Localization





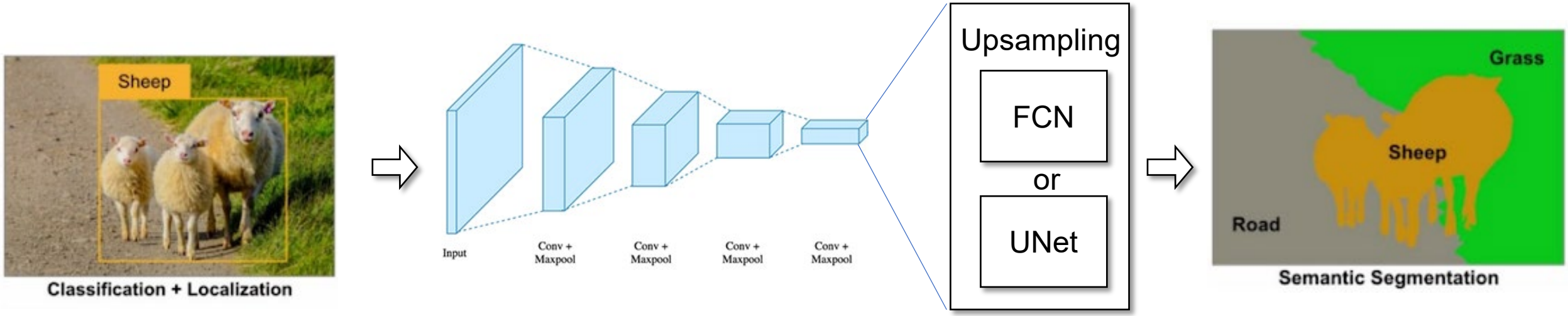


Semantic Segmentation

Semantic Segmentation

PIAI Research Department

- Semantic Segmentation
  - **이미지 내 모든 픽셀이 어느 클래스에 속하는지 예측하는 문제**



PIAI

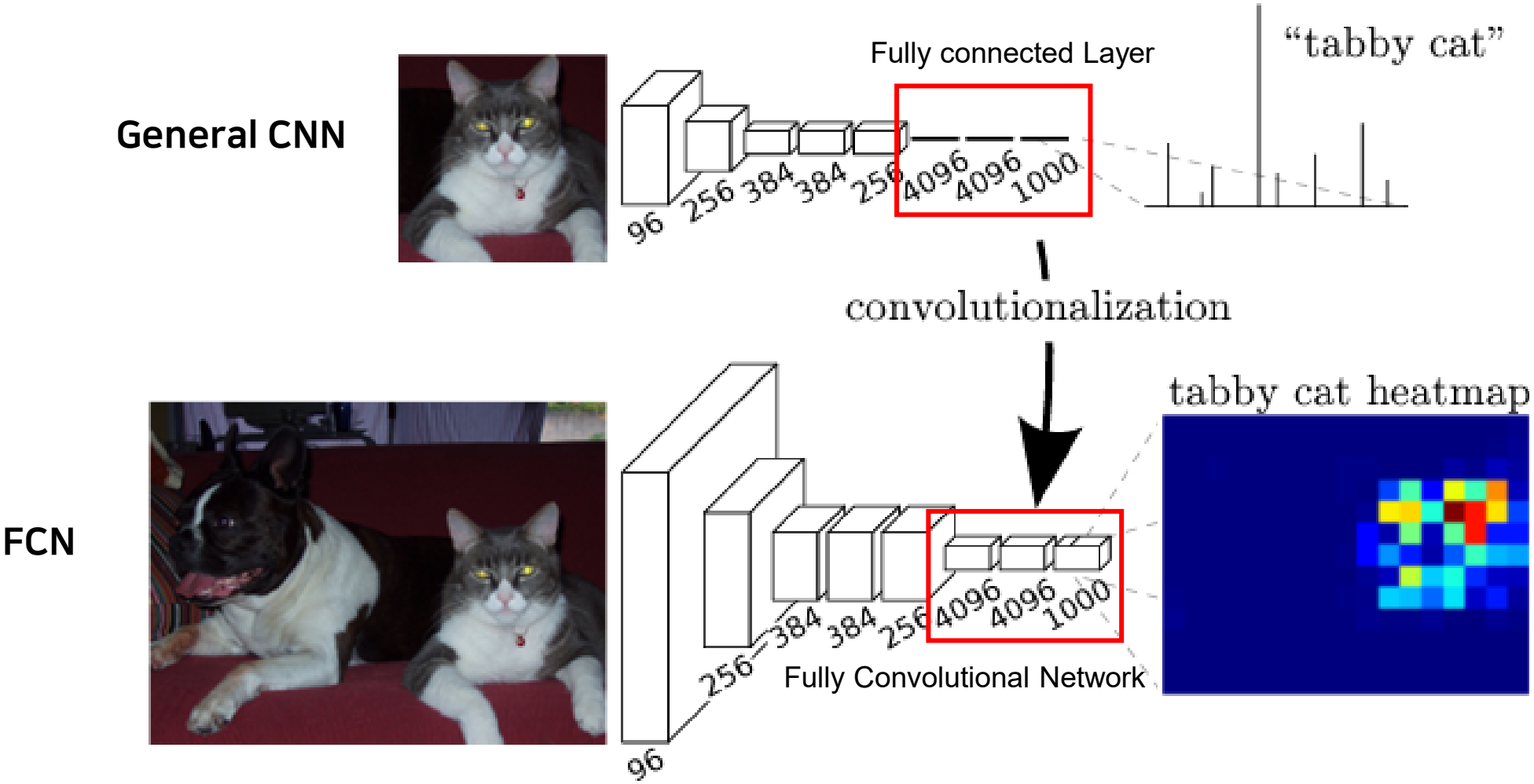
가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Semantic Segmentation
-FCN-

Dahyun, Kim

Semantic Segmentation - FCN

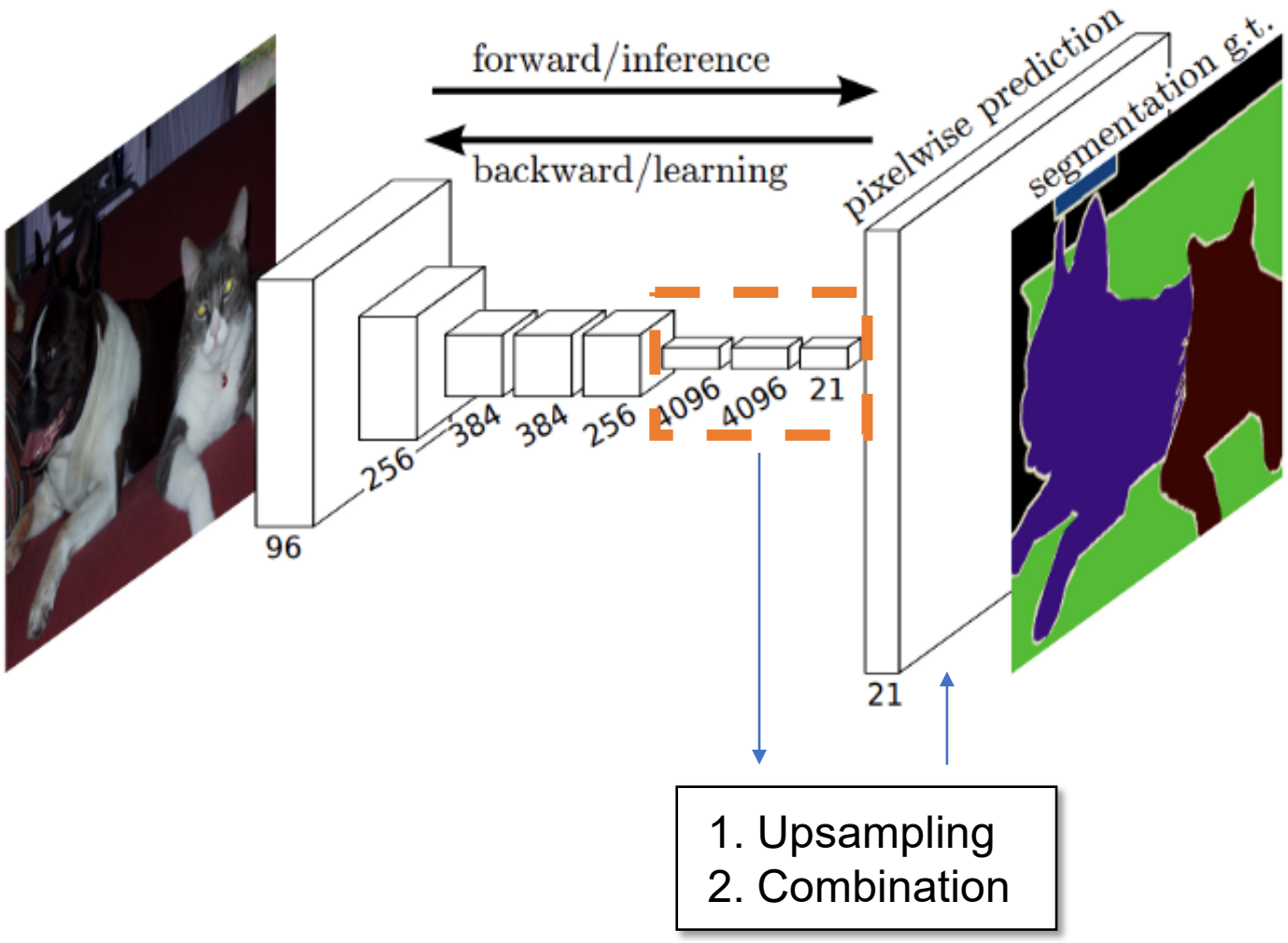
- Fully Convolutional Network



Semantic Segmentation - FCN

PIAI Research Department

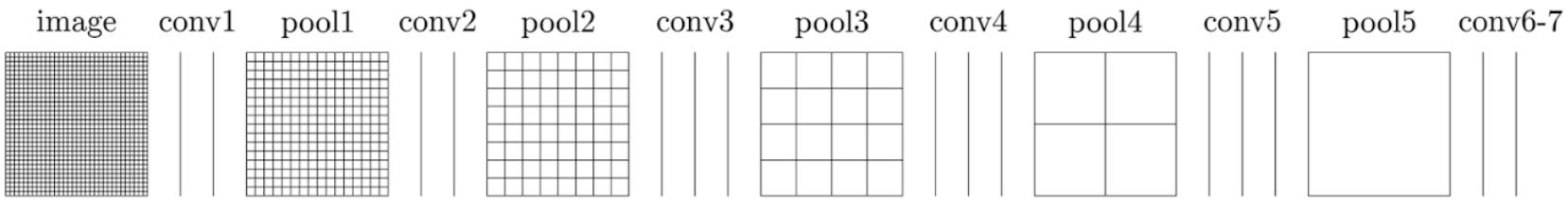
- Fully Convolutional Network



Semantic Segmentation - FCN

PIAI Research Department

- Shallow Layer ↔ Deep Layer



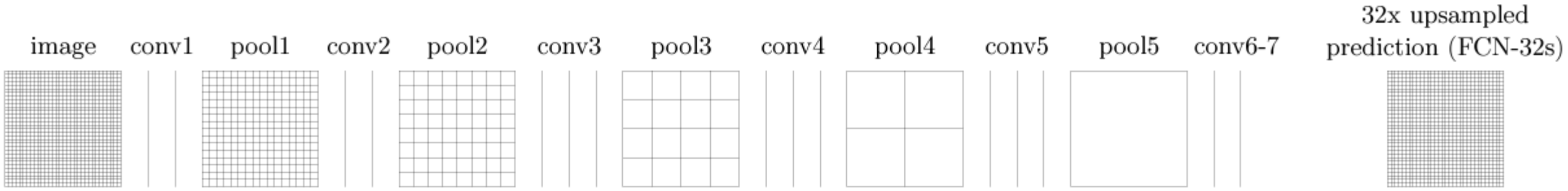
Shallow — Deep

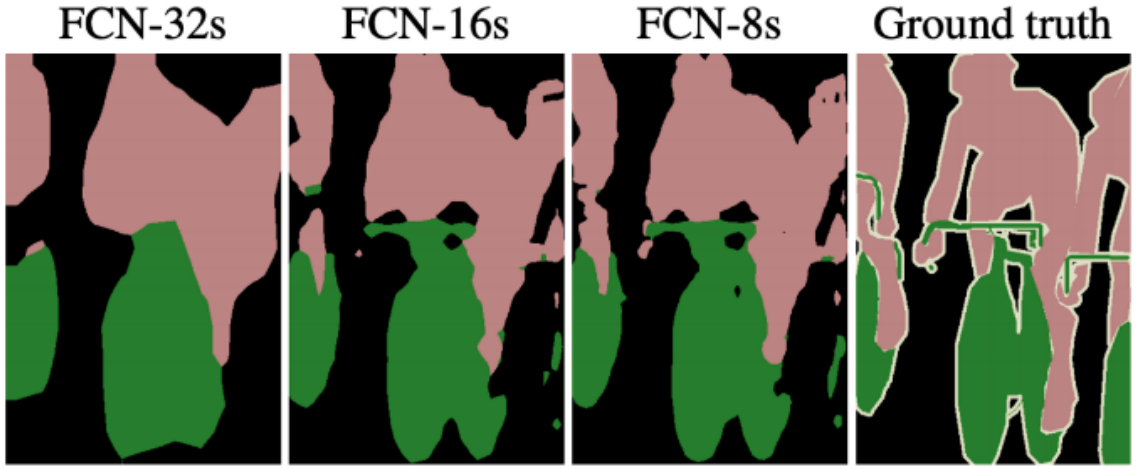
Fine	Coarse
Location	Semantic
Local	Global
Detail	Abstract

Semantic Segmentation - FCN

PIAI Research Department

- Combination: Deep Layer(global feature) + Shallow Layer(local feature)





Semantic Segmentation - FCN

PIAI Research Department

- General Upsampling Methods

Max Pooling
Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

Rest of the network

Max Unpooling
Use positions from pooling layer

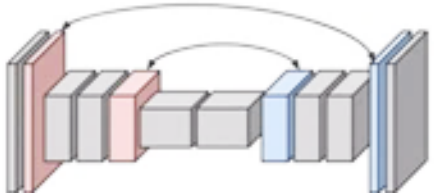
1	2
3	4

Input: 2 x 2

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers



Nearest Neighbor

1	2
3	4

Input: 2 x 2

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Output: 4 x 4

"Bed of Nails"

1	2
3	4

Input: 2 x 2

1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Output: 4 x 4

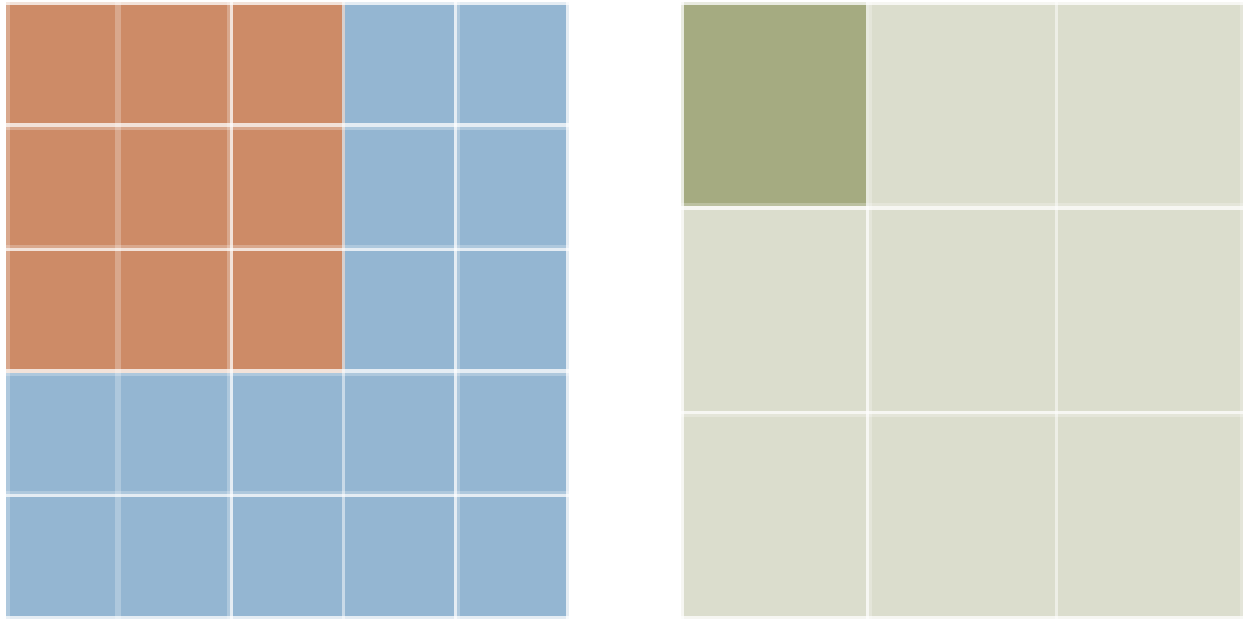
Semantic Segmentation - FCN

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Transposed Convolution : Learnable upsampling

[Convolution]

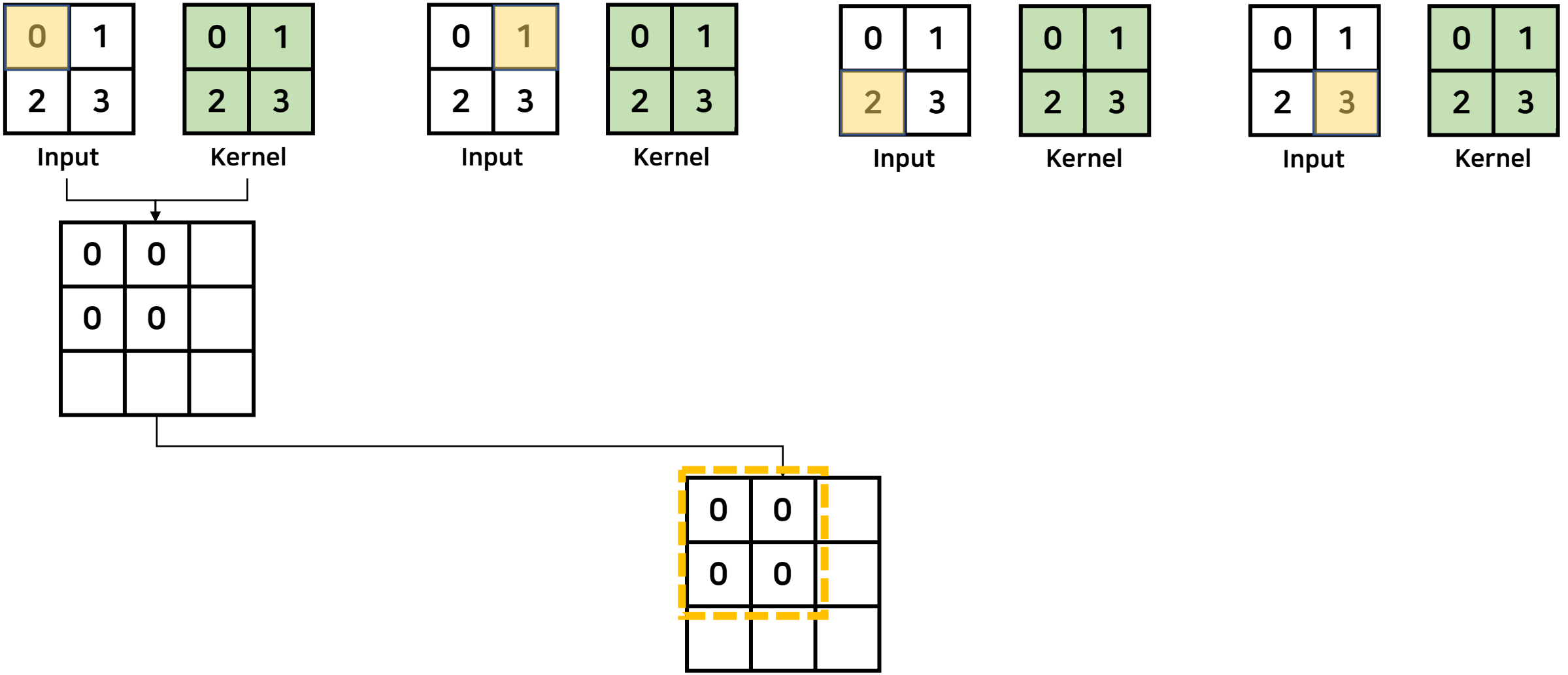
Type: conv - Stride: 1 Padding: 0



Semantic Segmentation - FCN

PIAI Research Department

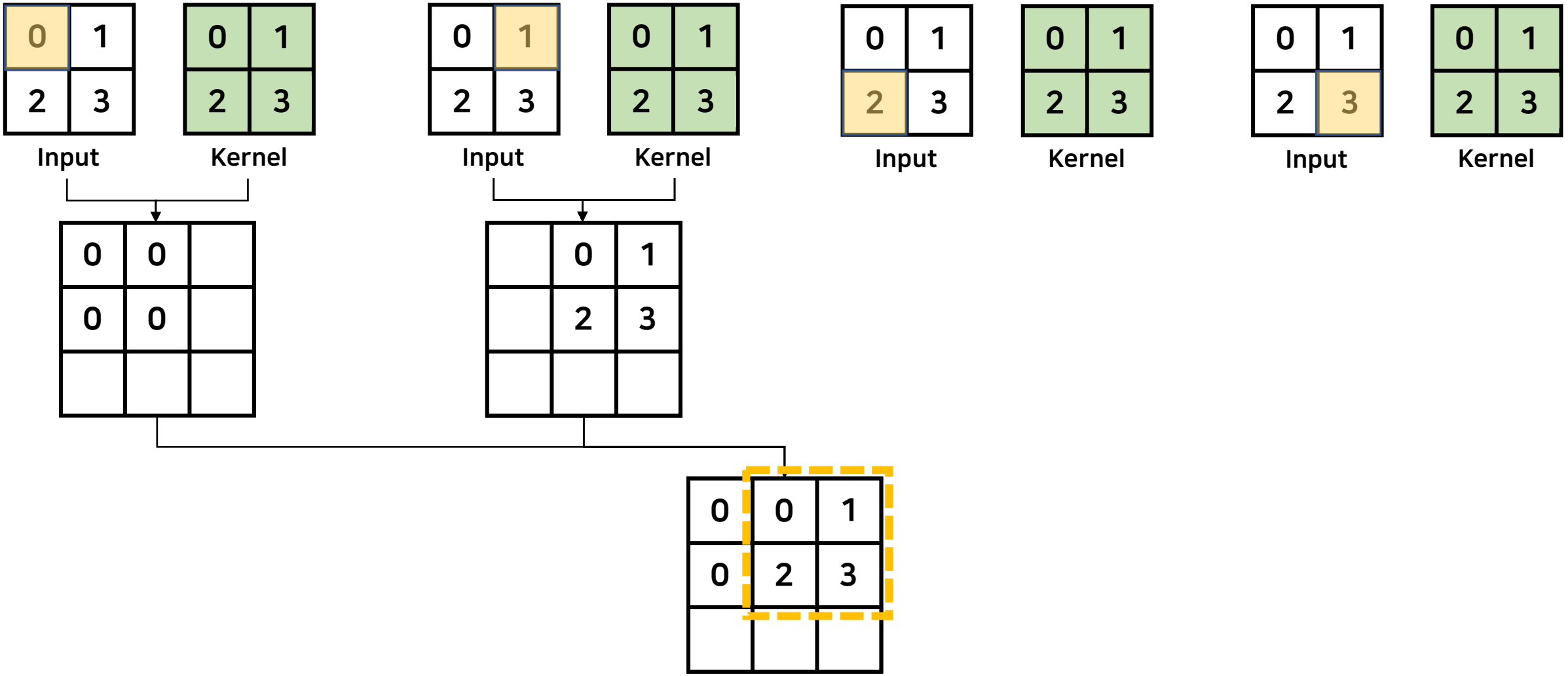
- Transposed Convolution



Semantic Segmentation - FCN

PIAI Research Department

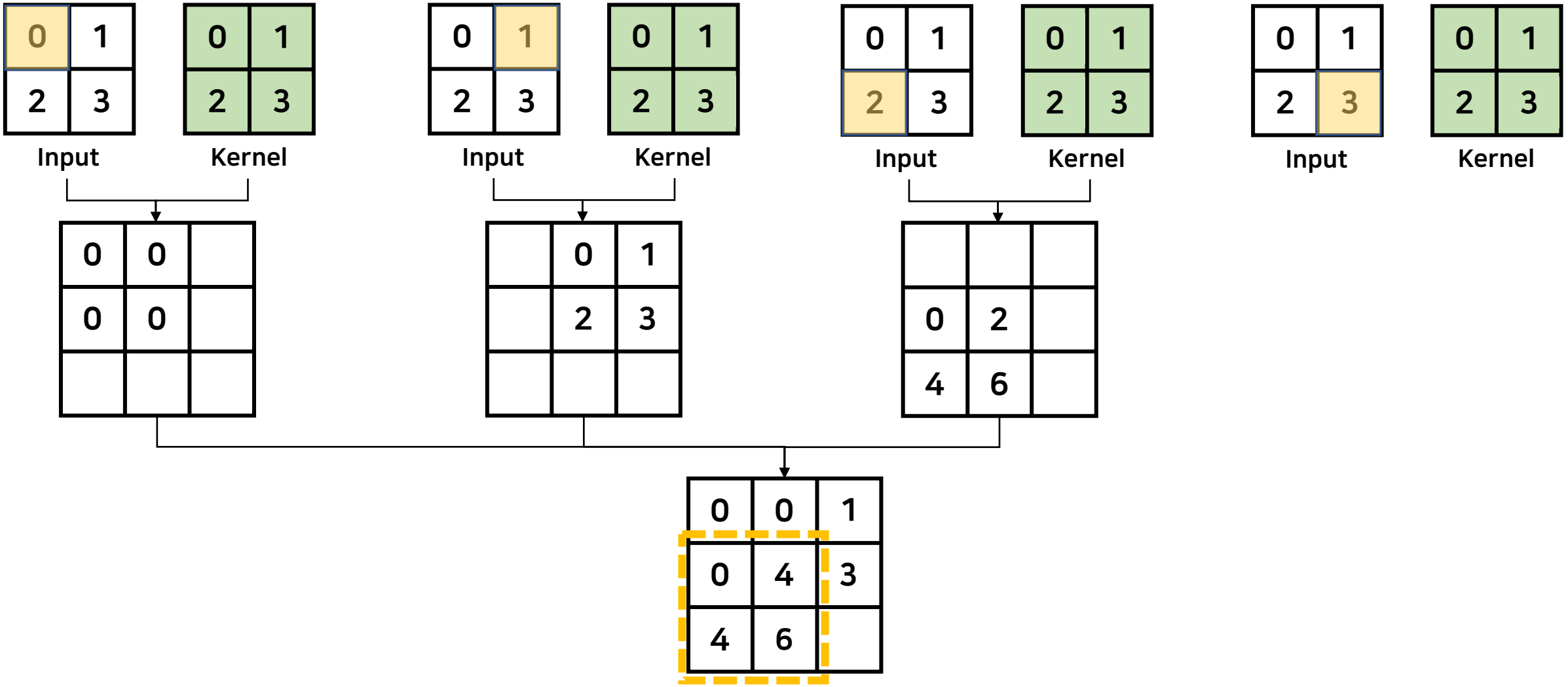
- Transposed Convolution



Semantic Segmentation - FCN

PIAI Research Department

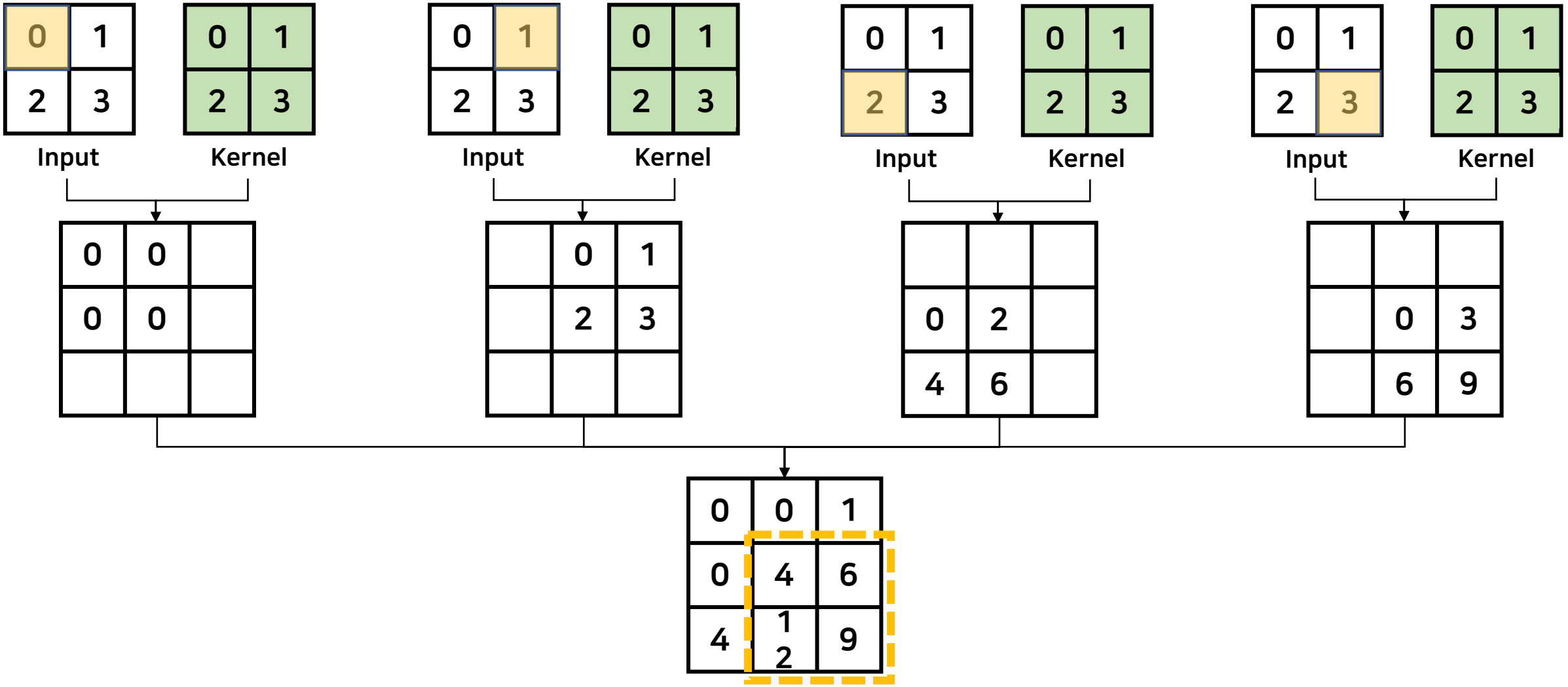
- Transposed Convolution



Semantic Segmentation - FCN

PIAI Research Department

- Transposed Convolution



가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

PIAI

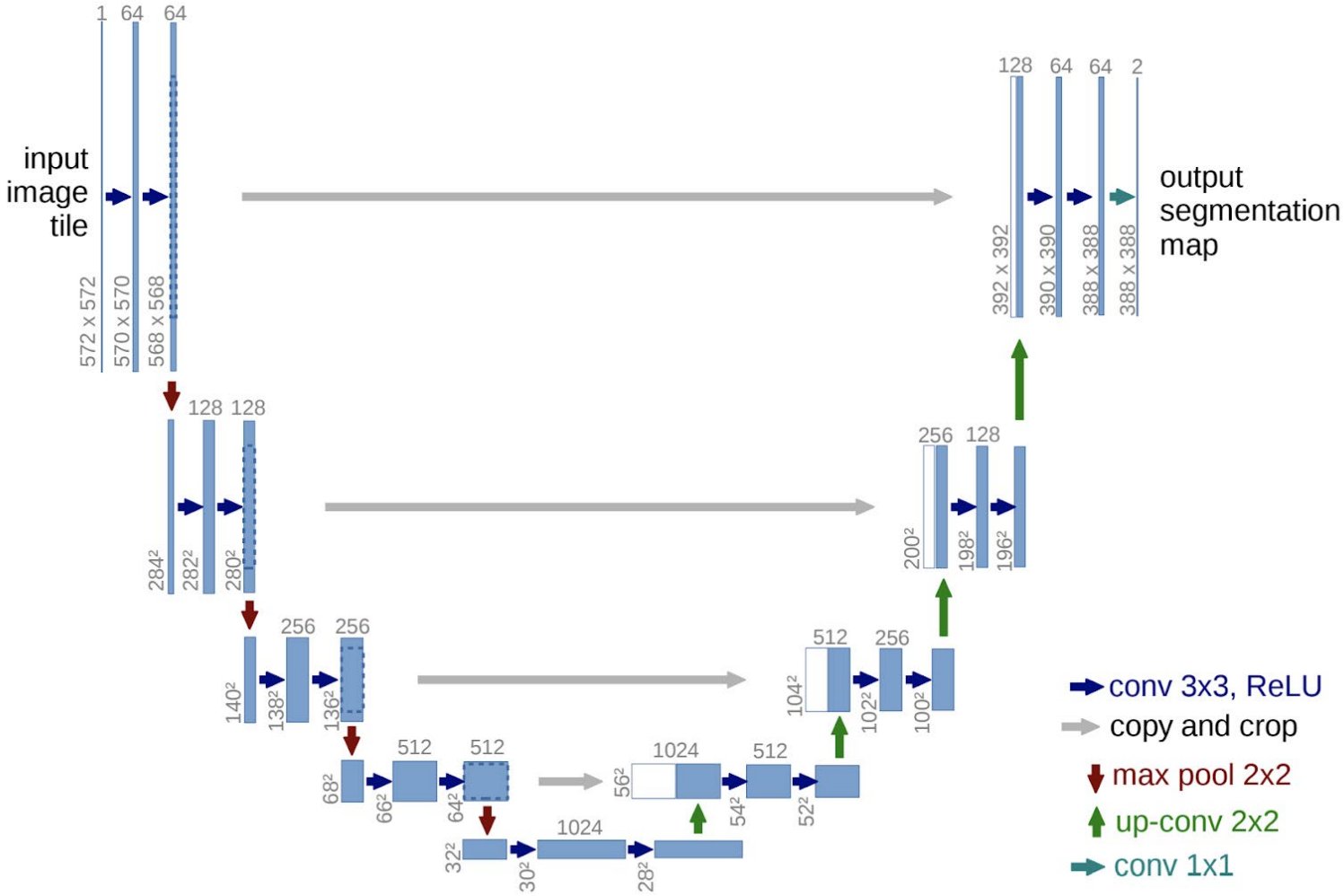
Semantic Segmentation
-UNet-

Dahyun, Kim

Semantic Segmentation - UNet

PIAI Research Department

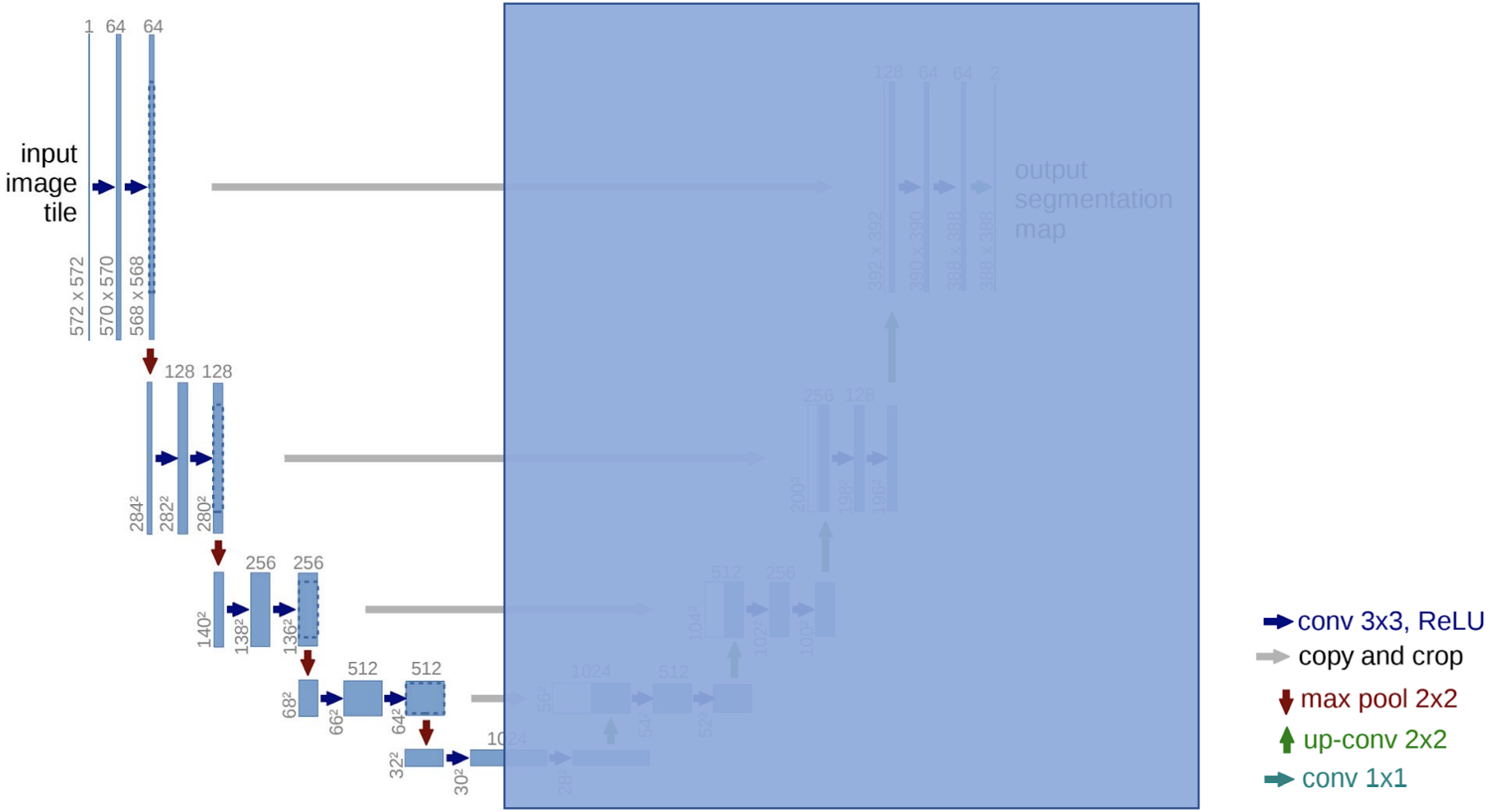
- U-Net Architecture



Semantic Segmentation - UNet

PIAI Research Department

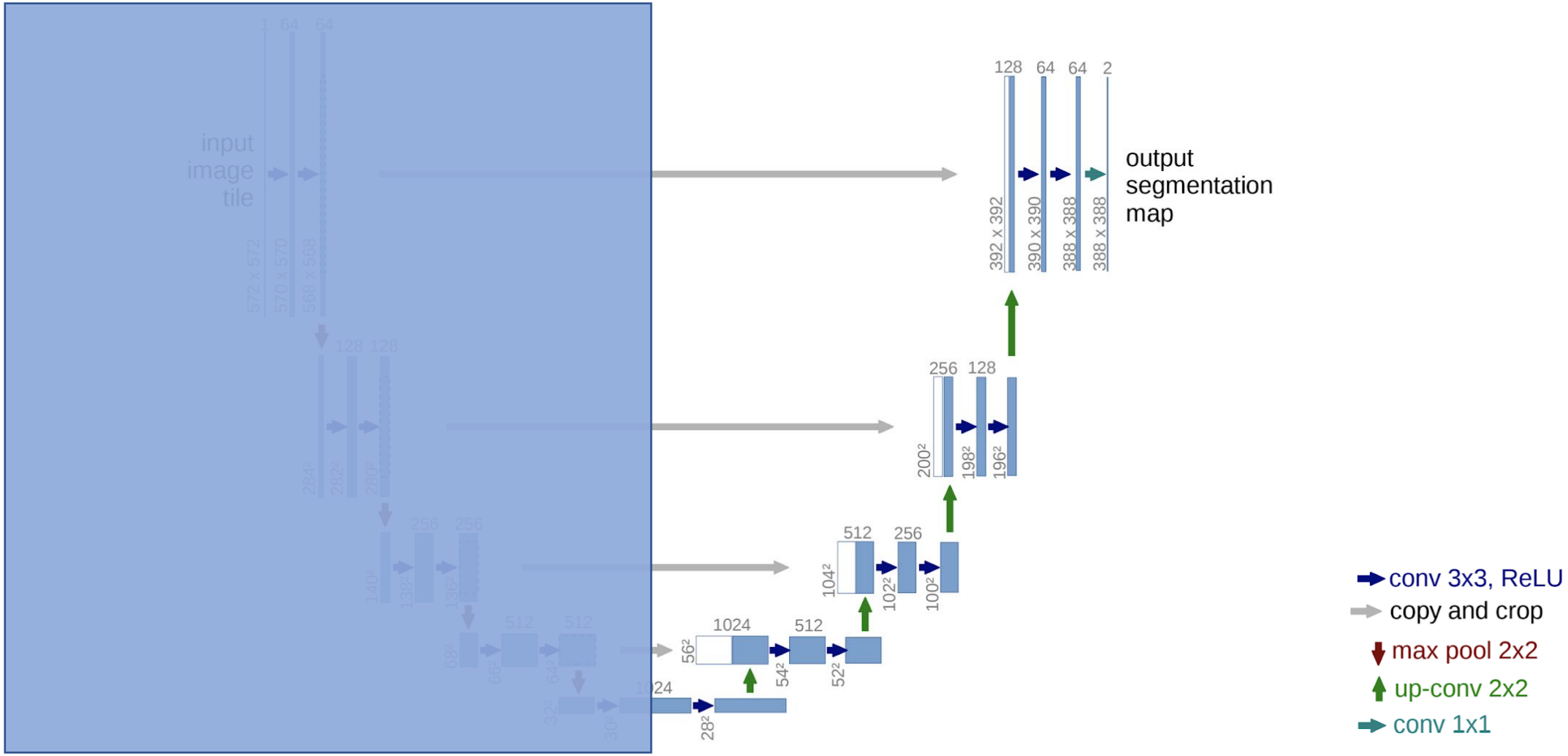
- Contracting Path : Down-Sampling 을 통해 이미지의 context 정보 추출



Semantic Segmentation - UNet

PIAI Research Department

- Expansive Path : Up-Sampling, **얕은 레이어의** Feature map **결합 수행**



PIAI

Instance Segmentation

Dahyun, Kim

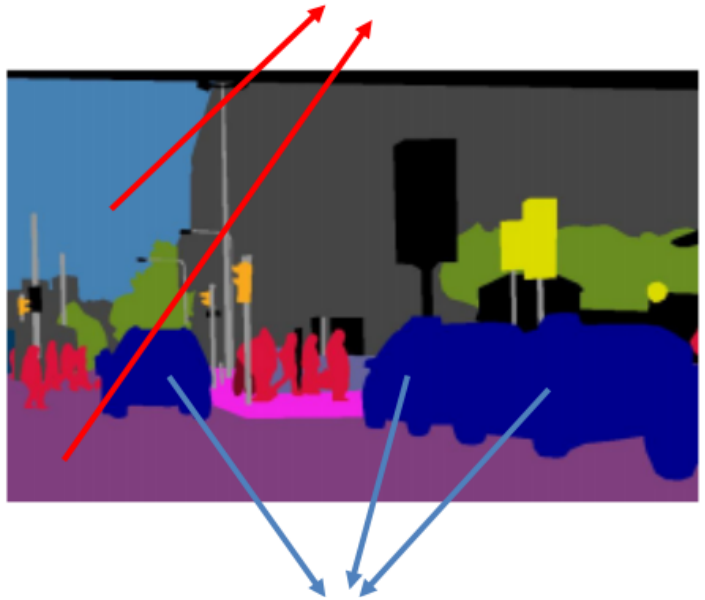
가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Instance Segmentation

PIAI Research Department

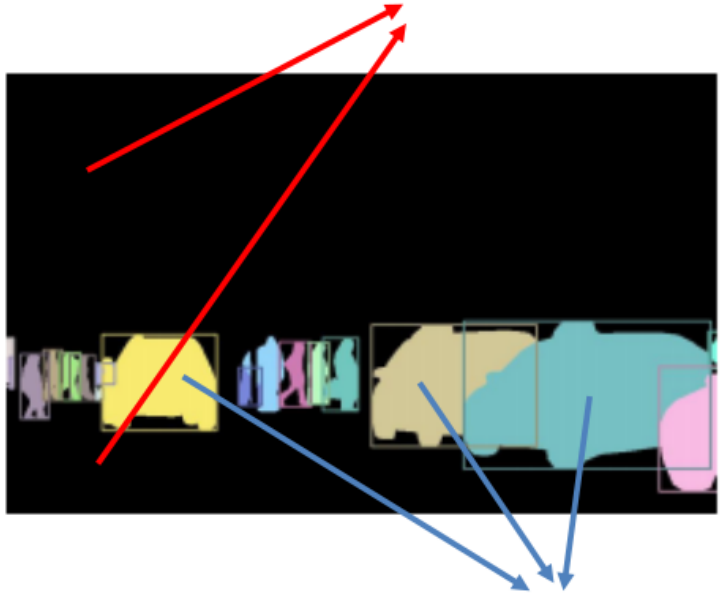
- Difference between semantic and instance segmentation

배경을 포함하여 모든 픽셀에 대하여 레이블링 (하늘, 잔디, 길 등)



같은 종류의 객체에 대해서 다른 픽셀 라벨 부여 되지 않음

셀 수 없는 객체에 대해서는 픽셀에 대한 레이블링 하지 않음 (하늘, 잔디, 길 등)



다른 종류의 객체에 대해서 다른 픽셀 라벨 부여됨

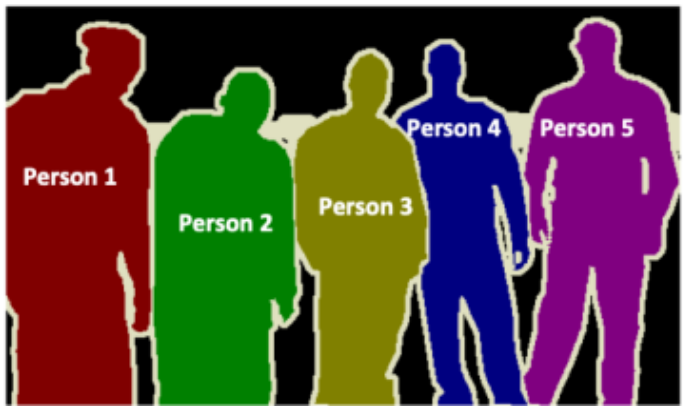
Instance Segmentation

PIAI Research Department

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

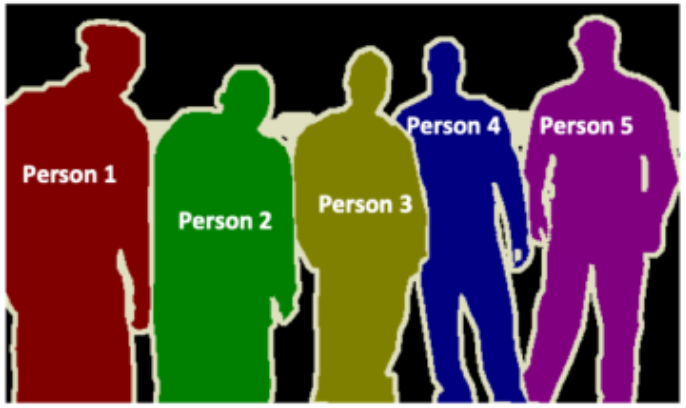
- Instance Segmentation methods

FCN-based



VS.

Proposal-based

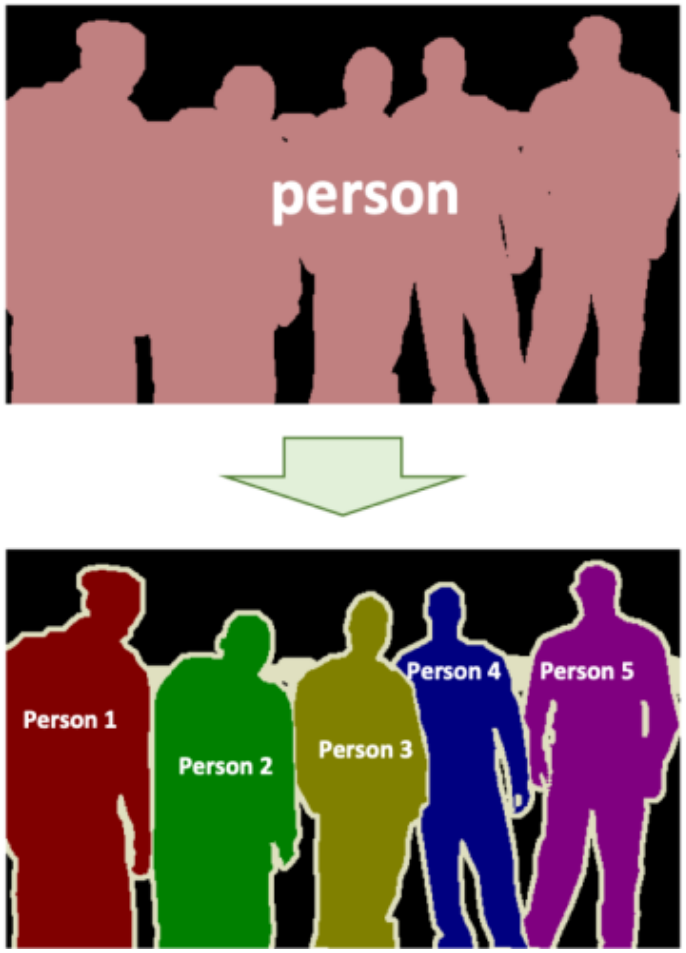


Instance Segmentation

PIAI Research Department

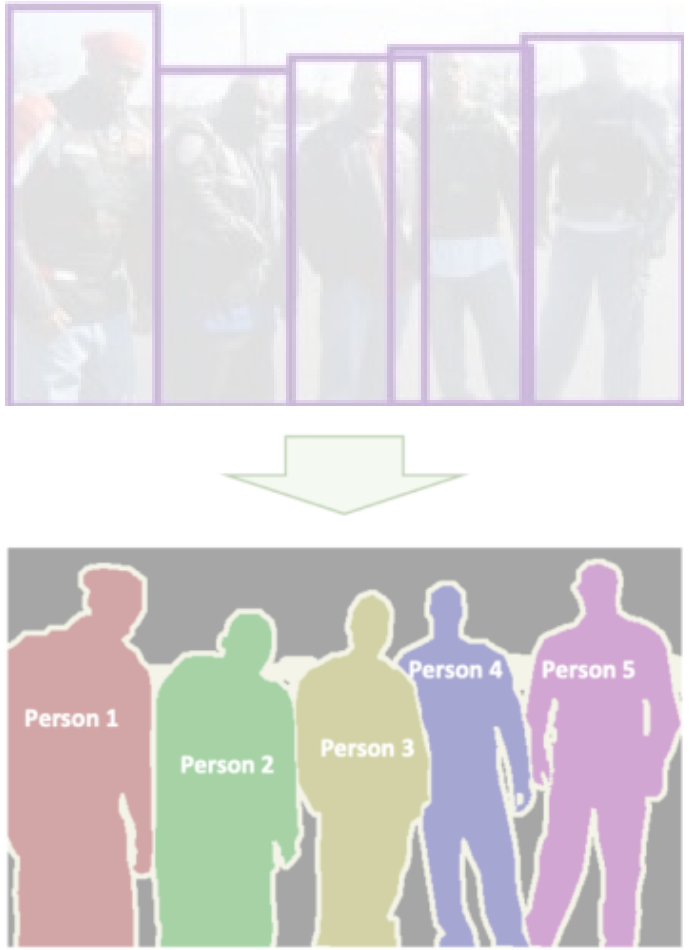
- Instance Segmentation methods

FCN-based



VS.

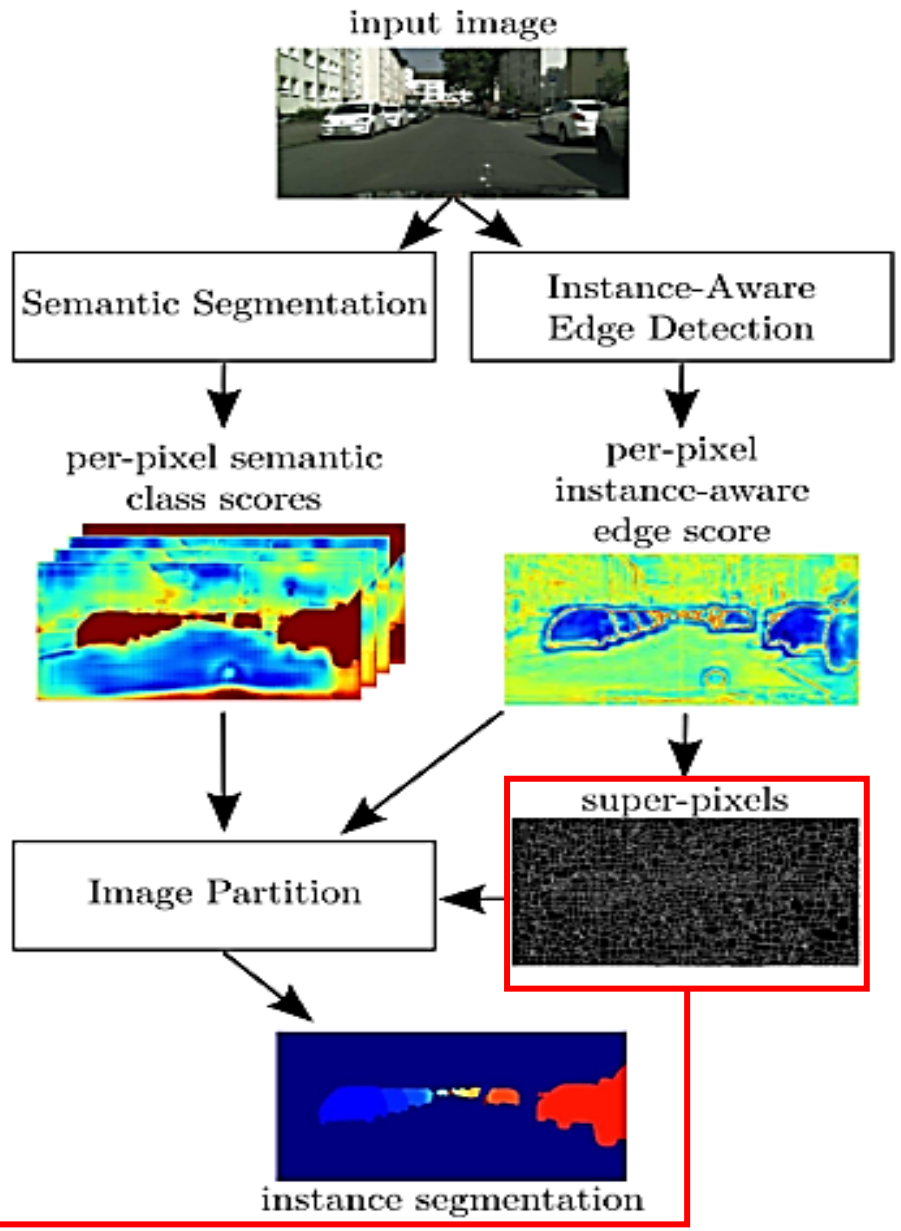
Proposal-based



Instance Segmentation

PIAI Research Department

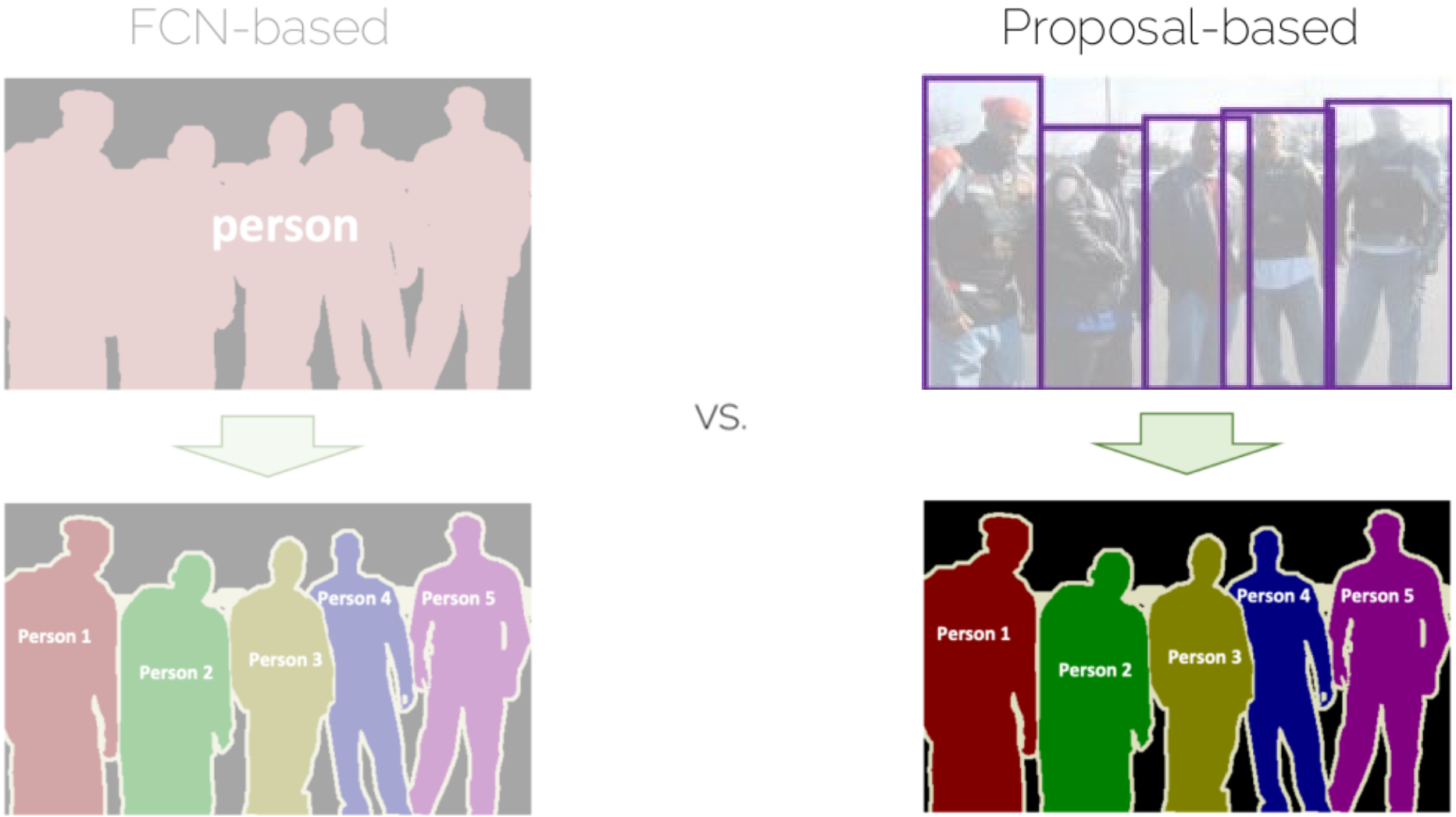
- FCN-based method



Instance Segmentation

PIAI Research Department

- Proposal-based method

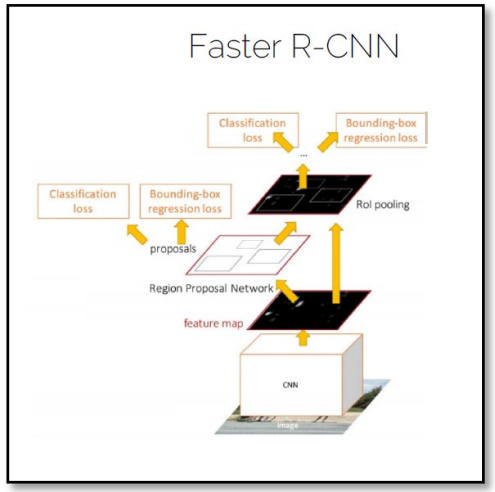


Instance Segmentation - 1,2 Stage

PIAI Research Department

- Object Detector ⟷ Instance Segmentator

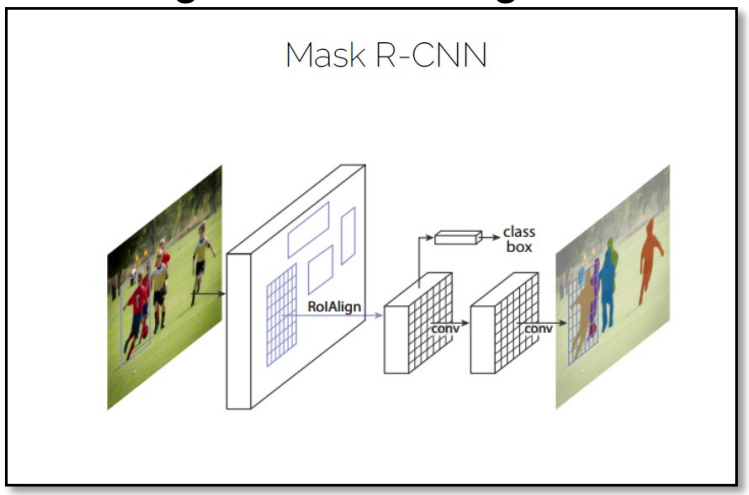
2-Stage Detector



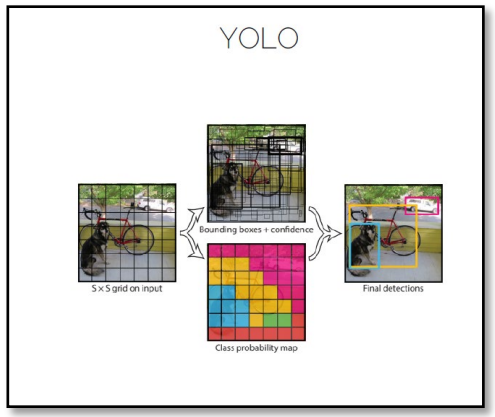
Classification / B-box Regression Head → + Mask Head

ROI Pooling → ROI Align

2-Stage Instance Segmentator



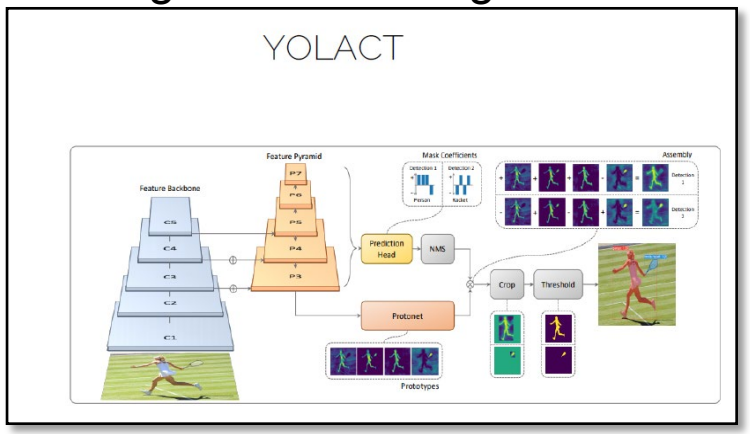
1-Stage Detector



NMS → Fast NMS

Classification / B-box Regression Head → + Protonet

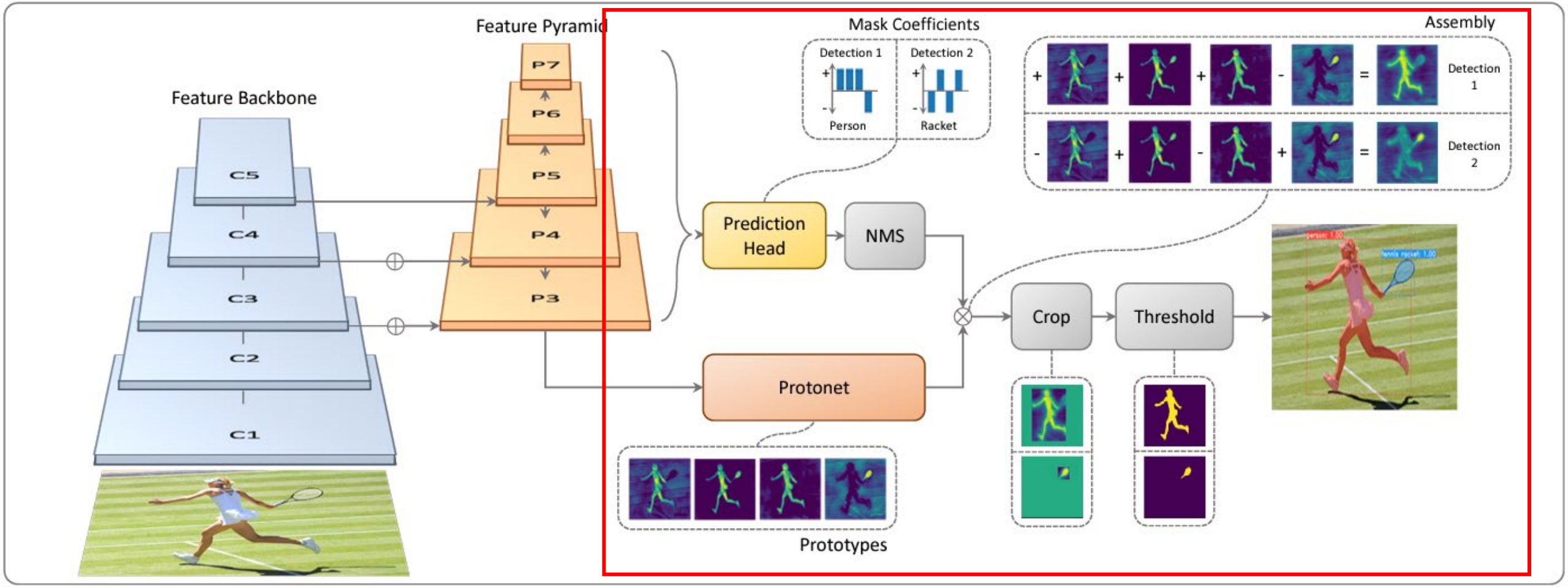
1-Stage Instance Segmentator



Instance Segmentation - YOLACT

PIAI Research Department

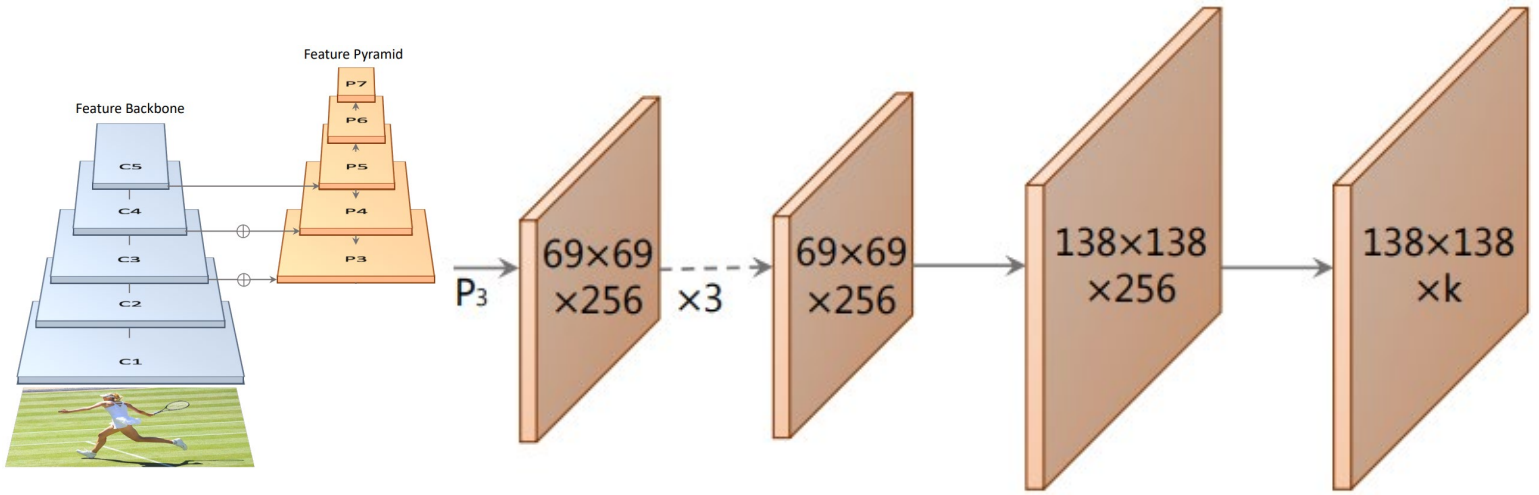
- YOLACT: **Y**ou **O**nly **L**ook **A**t **C**oefficien**T**s



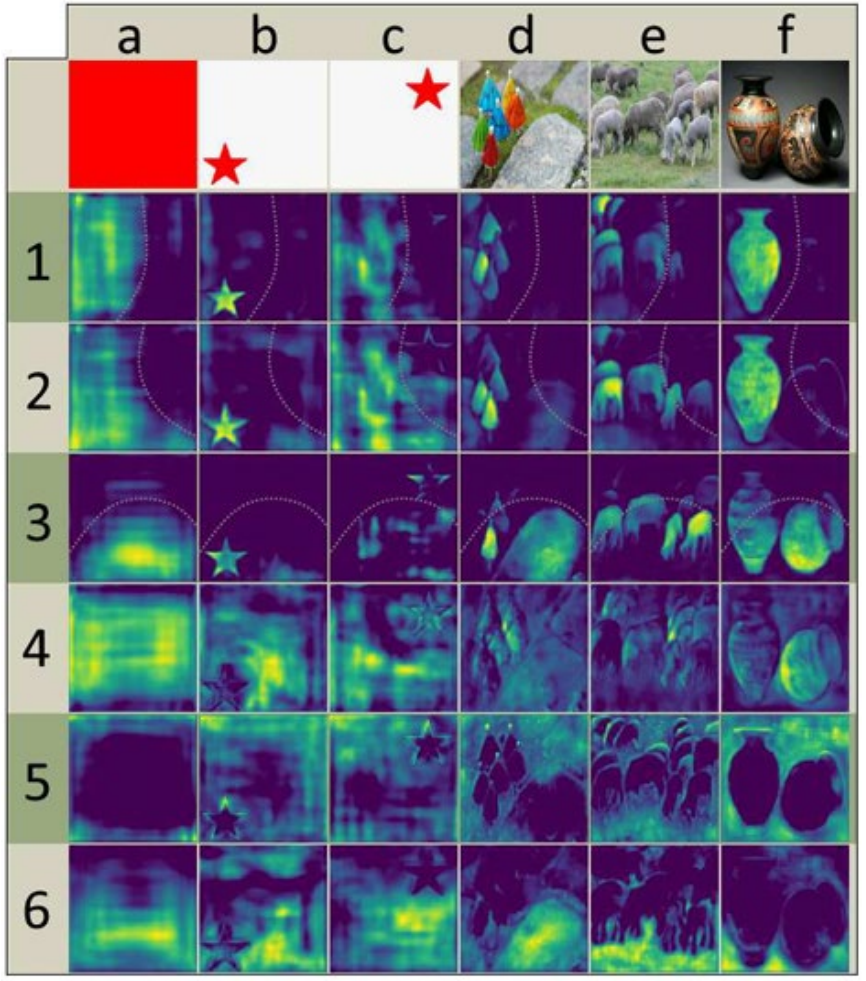
Instance Segmentation - YOLACT

PIAI Research Department

- Protonet Architecture



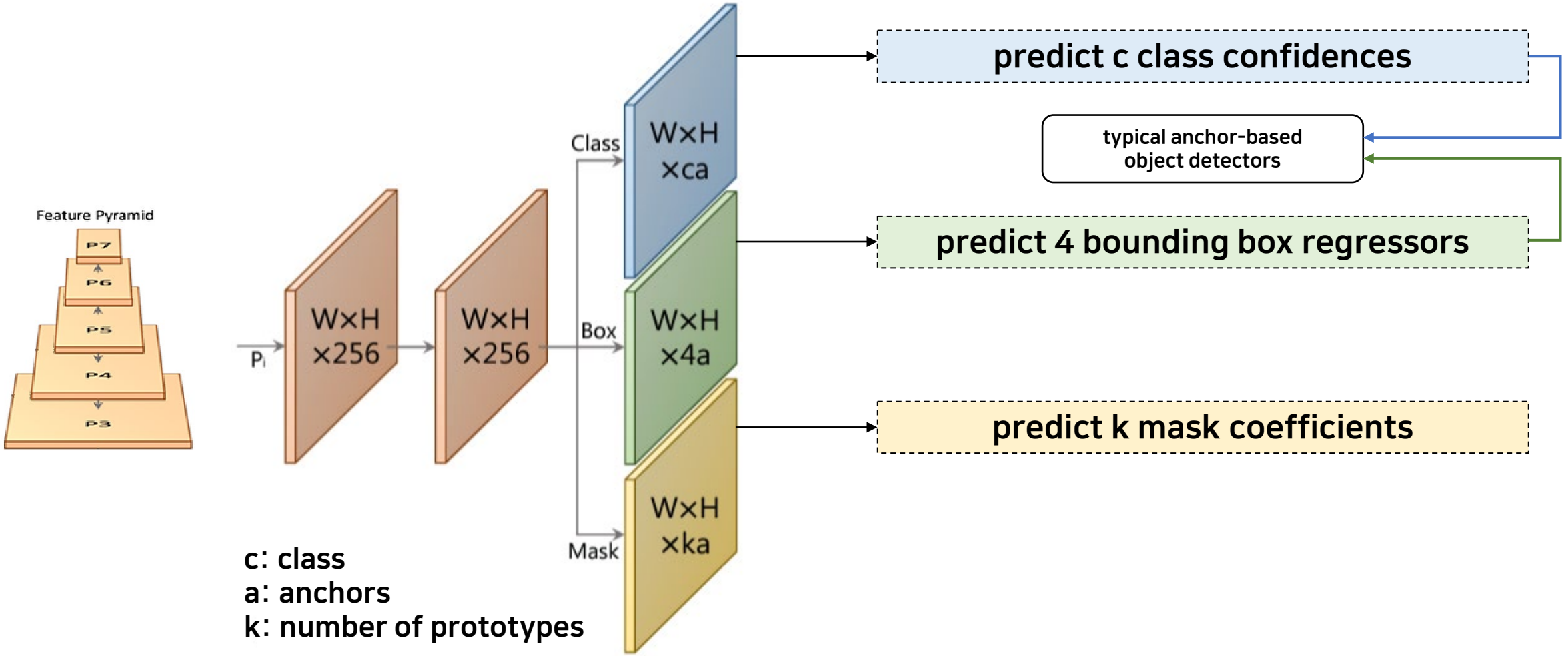
K=6



Instance Segmentation - YOLACT

PIAI Research Department

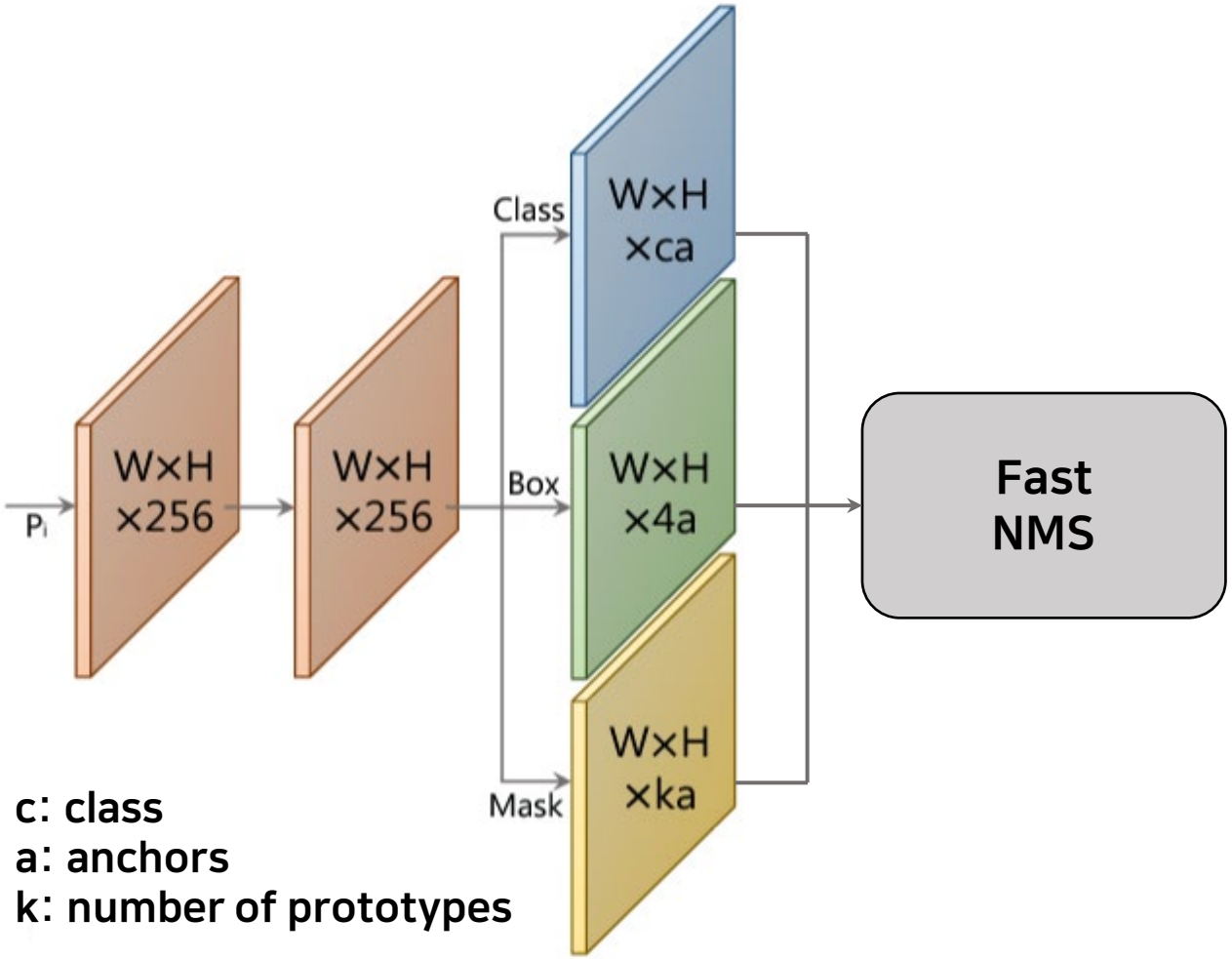
- Prediction Head Architecture



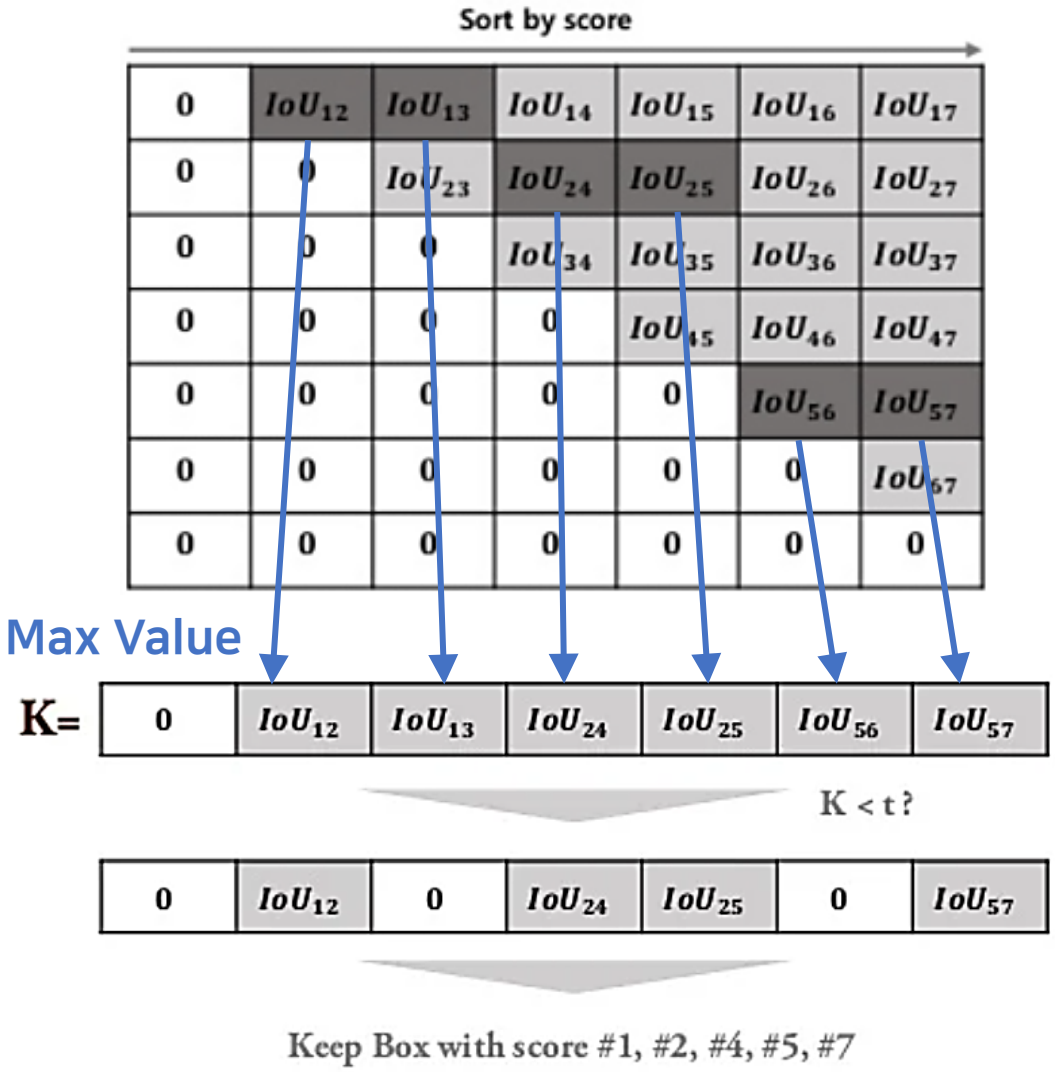
Instance Segmentation - YOLACT

PIAI Research Department

- Fast NMS(Non Maximum Suppression)



c: class
a: anchors
k: number of prototypes

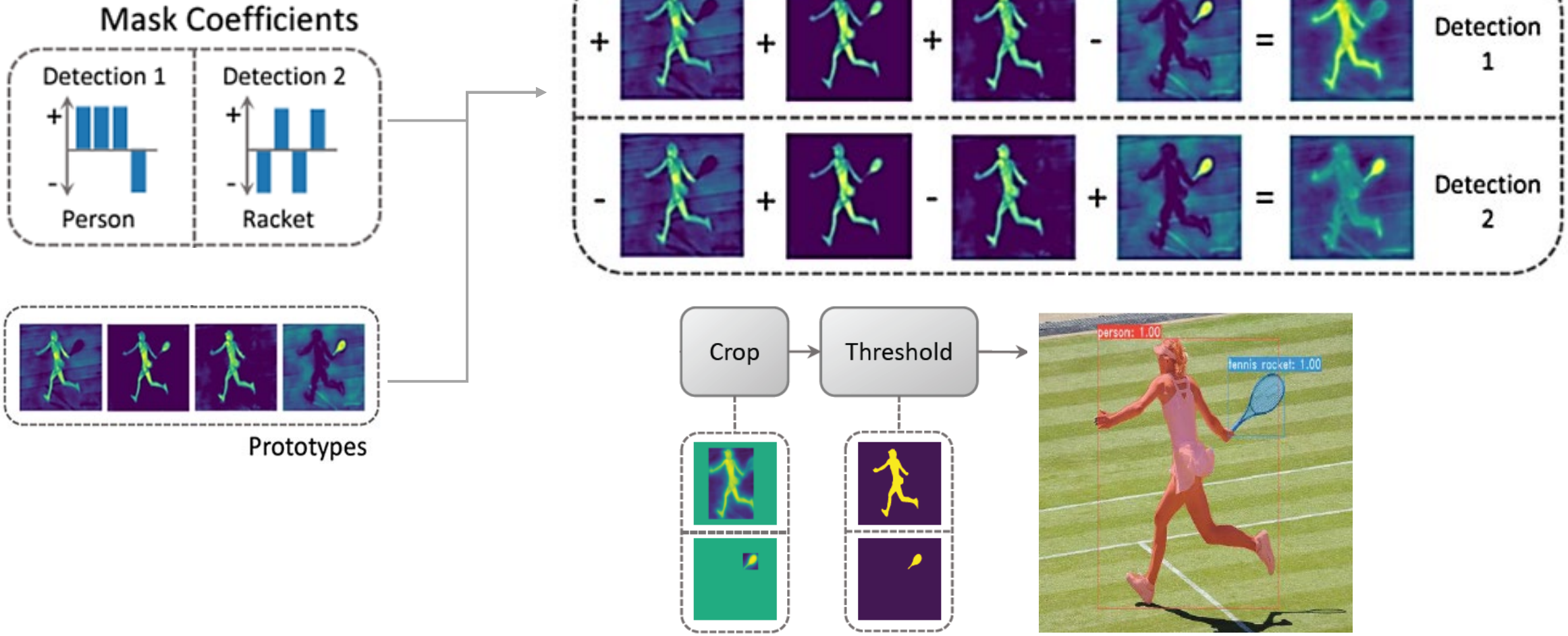


Instance Segmentation - YOLACT

PIAI Research Department

- YOLACT: **Y**ou **O**nly **L**ook **A**t **C**oefficien**T**s

$M = \sigma(PC^{T})$ Assembly



Code Running

PIAI

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Appendix – Annotation by labelme

Dahyun, Kim

1. 기본 환경 세팅 – labelme 설치

- **labelme는 OD/Segmentation용 라벨링 툴이며 설치 시 필요 패키지들을 먼저 설치해야함.**
  ※ labelme Github : https://github.com/wkentaro/labelme

1. 가상환경 생성 및 가상환경 실행
- conda create -n labelme python=3.8
- conda activate labelme

2. 생성 가상환경 내 필요 패키지 설치
- conda install –c conda-forge pyside2
- conda install pyqt
- pip install pyqt5

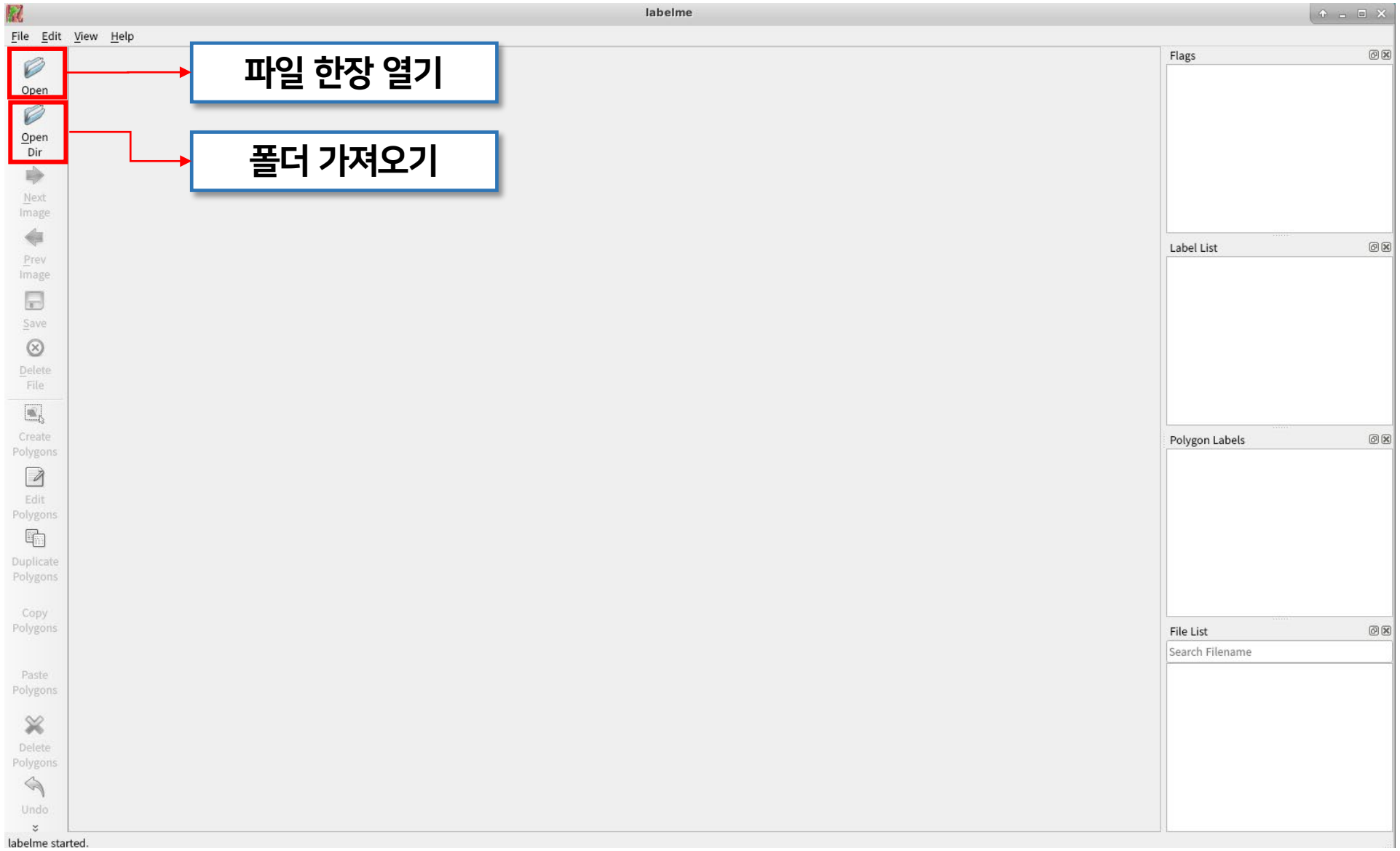
3. 생성 가상환경 내에서 labelImg 설치
- pip install labelme

4. 생성 가상환경 내에서 labelme 실행
- labelme

```
(labelme) pirl@pirl-PowerEdge-R740:~$ labelme
```

1. Labelme 기능 – 파일 및 디렉토리 열기

PIAI Research Department



1. Labelme 기능 – Open Dir 및 단축키

PIAI Research Department

단축키	기능
D	다음 이미지
A	이전 이미지
Ctrl + N	Draw Polygon
Ctrl + R	Draw Rectangle
Ctrl + Z	Undo
Ctrl + J	Edit Polygon

1. Labelme 기능 – 자동 저장 모드

PIAI Research Department

- 저장을 수동적으로 하지 않고 이전/다음 이미지로 넘어가면 자동적으로 저장하는 모드

1. Labelme 기능 – 저장 위치

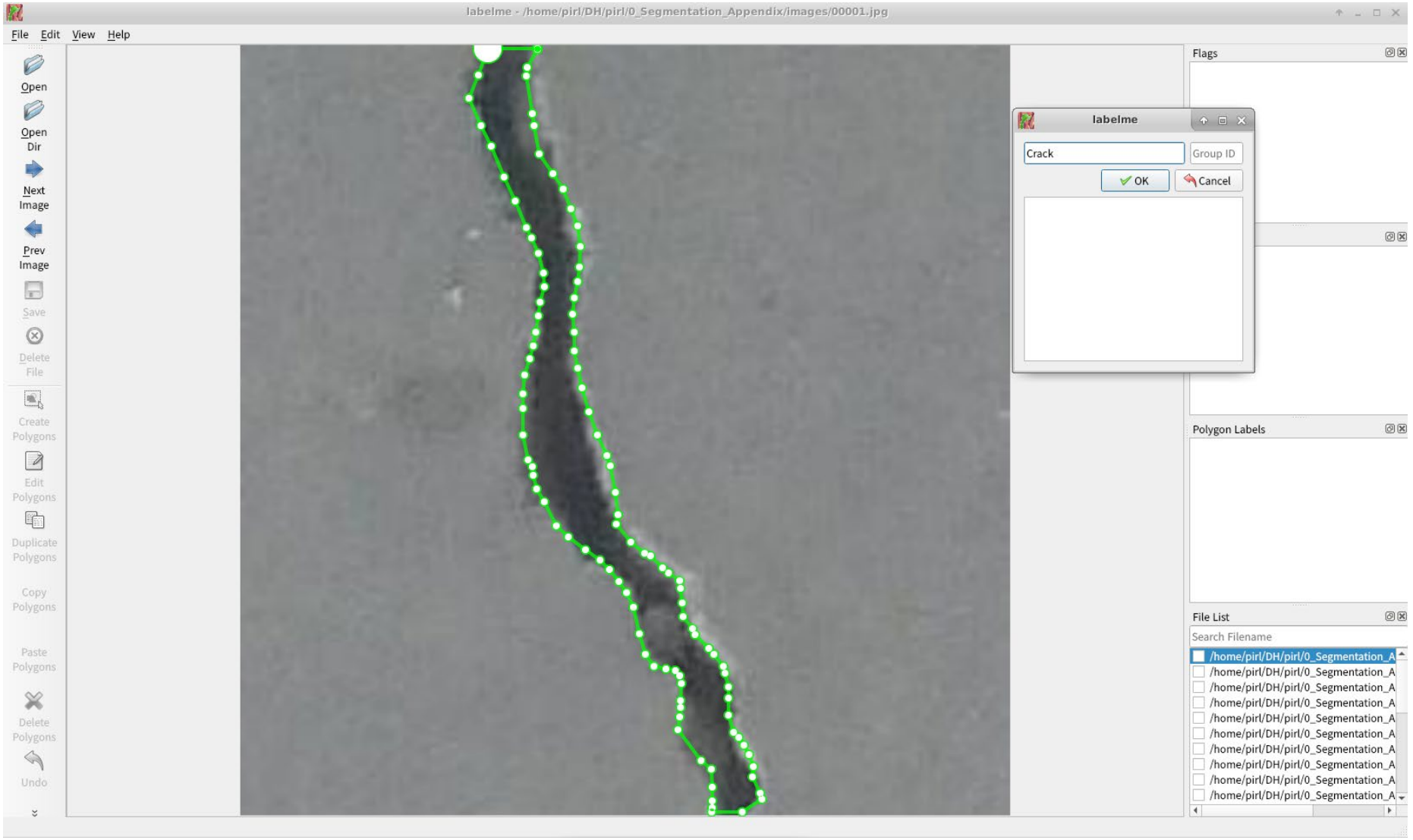
PIAI Research Department

- YOLACT 학습을 위해서는 데이터를 COCO Dataset 형태로 변환을 해 줘야하는데, 이 때 이미지와 .json 라벨 모두 한 폴더 내에 존재해야함.
- File -> Save With Image Data 체크 해 두면 같은 폴더 내에 저장이 됨. 또는 Change Output Dir 해서 같은 폴더로 설정도 가능함.

1. Labelme를 활용한 라벨링

PIAI Research Department

- Ctrl + N을 눌러 점들을 찍어 폴리곤 형태를 만든 뒤 라벨을 부여



이미지 하나당 한 개의 .json 파일 생성

예시)

열기(O)


```
"version": "5.0.1",
"flags": {},
"shapes": [
 {
    "label": "Crack",
    "points": [
    [
      72.98989898989899,
      1.0101010101010102
    ],
    [
      70.21212121212122,
      8.83838383838384
    ],
    [
      67.43434343434343,
      15.656565656565657
    ],
```

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

PIAI

Appendix – Train YOLACT

Dahyun, Kim

1. .json 라벨 파일 → coco 데이터 형식으로 변환

PIAI Research Department

- YOLACT를 학습시키기 위해서는 JSON 형태 파일을 COCO 데이터 형식으로 변환해야함.

※ labelme2coco Github : https://github.com/Tony607/labelme2coco

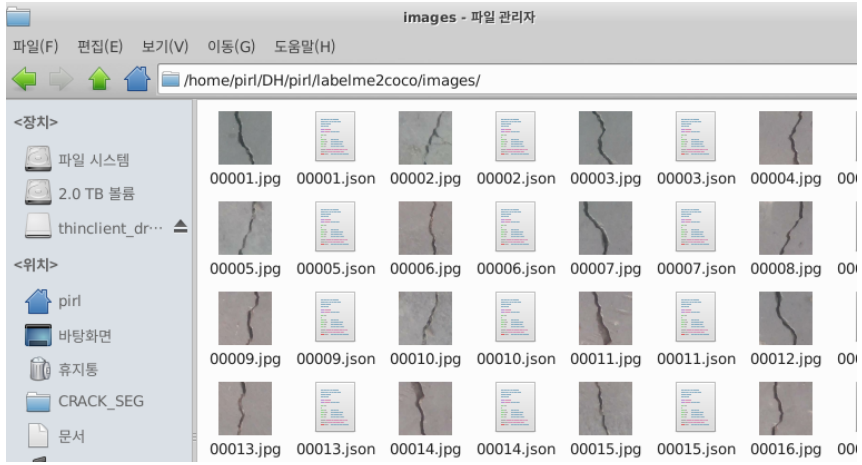
1. labelme 설치된 가상환경 Activate

- conda activate labelme

2. labelme2coco Git Clone

- git clone https://github.com/Tony607/labelme2coco

3. labelme2coco/images 폴더 안에 이미지 파일 및 json 파일 넣기



4. labelme2coco/labelme2coco.py 실행

- python labelme2coco.py images

```
(labelme) pirl@pirl-PowerEdge-R740:~/DH/pirl/labelme2coco$ python labelme2coco.py images
```

→ **trainval.json 생성됨**

2. yolact 다운로드 및 학습/검증 데이터 준비

PIAI Research Department

- Git 으로부터 YOLACT 다운로드

※ YOLACT Github : https://github.com/dbolya/yolact

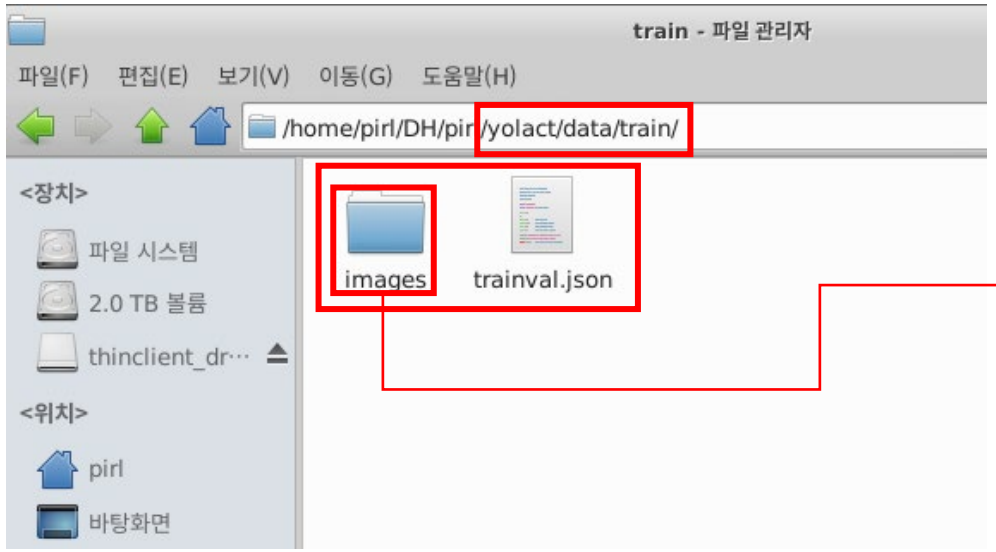
1. YOLACT requirements 설치 된 가상환경 Activate (ex. cv_edu3)

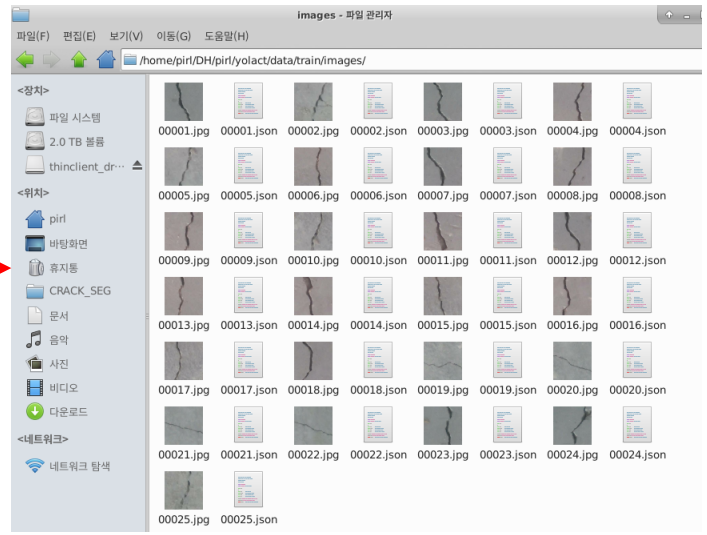
- conda activate cv_edu3

2. YOLACT Git Clone

- git clone https://github.com/Tony607/labelme2coco

3. clone 받은 yolact/data 폴더 내에 train 폴더 생성 후 trainval.json 과 원본 images 폴더(이미지 + json) 넣기





2. yolact 다운로드 및 학습/검증 데이터 준비

PIAI Research Department

4. yolact/data/config.py 에서 내 데이터 경로 및 라벨 정보 입력
→ Config.py 내에서 DATASETS 항목을 찾은 후 가장 마지막에 추가

```
# ---------------------- DATASETS ---------------------- #

dataset_base = Config({
    'name': 'Base Dataset',

    # Training images and annotations
    'train_images': './data/coco/images/',
    'train_info':   'path_to_annotation_file',

    # Validation images and annotations.
    'valid_images': './data/coco/images/',
    'valid_info':   'path_to_annotation_file',

    # Whether or not to load GT. If this is False, eval.py qua
    'has_gt': True,

    # A list of names for each of you classes.
    'class_names': COCO_CLASSES,

    # COCO class ids aren't sequential, so this is a bandage f
    # provide a map from category_id -> index in class_names +
    # If not specified, this just assumes category ids start a
    'label_map': None
})
```

가장 마지막에 추가

```
crack_dataset = dataset_base.copy({
    'name': 'Crack',

    # Training images and annotations
    'train_images': './data/train/images/',
    'train_info':   './data/train/trainval.json',

    # Validation images and annotations.
    'valid_images': './data/train/images/',
    'valid_info':   './data/train/trainval.json',

    # A list of names for each of you classes.
    'class_names': ('Crack'),

    # COCO class ids aren't sequential, so this is a b
    # provide a map from category_id -> index in class
    # If not specified, this just assumes category ids
    'label_map': {0:1}
})
```

라벨 인덱스 – len(class_names)+1 까지

2. yolact 다운로드 및 학습/검증 데이터 준비

PIAI Research Department

5. yolact/data/config.py 에서 내 커스텀 모델에 대한 정의 입력
→ Config.py 내에서 YOLACT v1.0 CONFIGS 항목을 찾은 후 가장 마지막에 추가

```
# ---------------------- YOLACT v1.0 CONFIGS ---------------------- #

yolact_base_config = coco_base_config.copy({
    'name': 'yolact_base',

    # Dataset stuff
    'dataset': coco2017_dataset,
    'num_classes': len(coco2017_dataset.class_names) + 1,

    # Image Size
    'max_size': 550,

    # Training params
    'lr_steps': (280000, 600000, 700000, 750000),
    'max_iter': 800000,

    # Backbone Settings
    'backbone': resnet101_backbone.copy({
        'selected_layers': list(range(1, 4)),
        'use_pixel_scales': True,
        'preapply_sqrt': False,
        'use_square_anchors': True, # This is for backward compatability

        'pred_aspect_ratios': [ [[1, 1/2, 2]] ]*5,
        'pred_scales': [[24], [48], [96], [192], [384]],
    }),
```

가장 마지막에 추가

```
yolact_resnet50_crack_config = yolact_resnet50_config.copy({
    'name': 'yolact_resnet50_crack',

    # Dataset stuff
    'dataset': crack_database,
    'num_classes': len(crack_database.class_names) + 1,

    'backbone': yolact_resnet50_config.backbone.copy({
        'name': 'ResNet101',
        'path': 'yolact_base_54_800000.pth',
        'pred_scales': [[32], [64], [128], [256], [512]],
        'use_square_anchors': False,
    })
})
```

앞 선언한 데이터셋과 동일한 이름

내가 사용할 weight 이름과 동일해야함

파라미터 종류와 값들 원하는대로 입력

2. yolact 다운로드 및 학습/검증 데이터 준비

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

6. Config.py에 정의한 내 커스텀 모델 Backbone 다운받기

1) ResNet-50-FPN, 550(img size) : yolact_resnet50_54_800000.pth
2) Darknet53-FPN, 550(img size) : yolact_darknet53_54_800000.pth
3) Resnet101-FPN, 550(img size) : yolact_base_54_800000.pth
4) Resnet101-FPN, 700(img size) : yolact_im700:54_800000.pth

1. Download Pretrained Model

1. Resnet50-FPN, 550 : yolact_resnet50_54_800000.pth

2. Darknet53-FPN, 550 : yolact_darknet53_54_800000.pth

3) Resnet101-FPN, 550 : yolact_base_54_800000.pth

4. Resnet101-FPN, 700 : yolact_im700_54_800000.pth

```
import os

## --- YOLACT pretrained model download
## --- Make directories weights for save model weights file

if not os.path.exists('./weights'):
    os.makedirs('./weights')

# Resnet50-FPN : yolact_resnet50_54_800000.pth
# !wget -q --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cooki

# Darknet53-FPN : yolact_darknet53_54_800000.pth
# !wget -q --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cooki

# Resnet101-FPN : yolact_base_54_800000.pth
!wget -q --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cookies

# Resnet101-FPN : yolact_im700_54_800000.pth
# !wget -q --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cooki
```

실습 YOLACT.ipynb 주피터 파일 1번 항목 참조

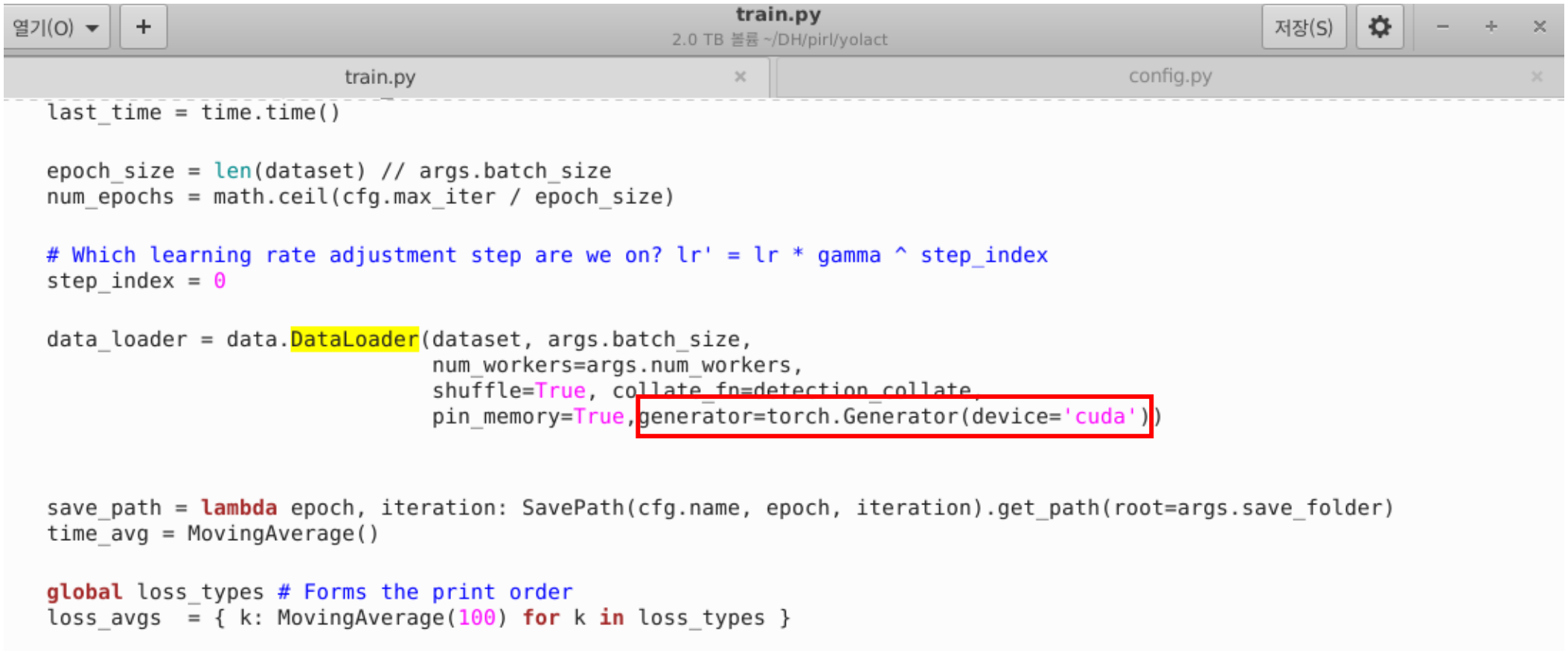
3. yolact 학습

PIAI Research Department

※ 만약 pytorch가 1.9 이상 버전이라면 학습 시 오류 발생

```
    yield from torch.randperm(n, generator=generator).tolist()
RuntimeError: Expected a 'cuda' device type for generator but found 'cpu'
```

→ 해당 오류 발생 시 train.py에서 DataLoader를 검색한 후 인자에 generator=torch.Generator(device='cuda') 추가



3. yolact 학습

- YOLACT 학습 - train.py

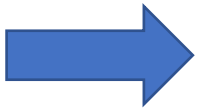
  - python train.py --config=내가 정의한 모델 config --batch=배치사이즈

예시) python train.py --config=yolact_resnet50_crack_config --batch_size=16

```
(cv_edu3) pirl@pirl-PowerEdge-R740:~/DH/pirl/yolact$ python train.py --config=yolact_resnet50_crack_config --batch_size=16
```

```
Calculating mAP...

       |  all  |  .50  |  .55  |  .60  |  .65  |  .70  |  .75  |  .80  |  .85  |  .90  |  .95  |
-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
   box |  0.33 |  0.98 |  0.94 |  0.71 |  0.46 |  0.14 |  0.03 |  0.01 |  0.01 |  0.00 |  0.00 |
  mask |  0.00 |  0.00 |  0.00 |  0.00 |  0.00 |  0.00 |  0.00 |  0.00 |  0.00 |  0.00 |  0.00 |
-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
```



학습 도중 중지 시키면 weights 폴더에 모델이 저장 되며, 중간중간 저장하도록 argument를 넘길 수 있음. 해당 내용은 train.py 내부의 argument를 확인해 볼 것