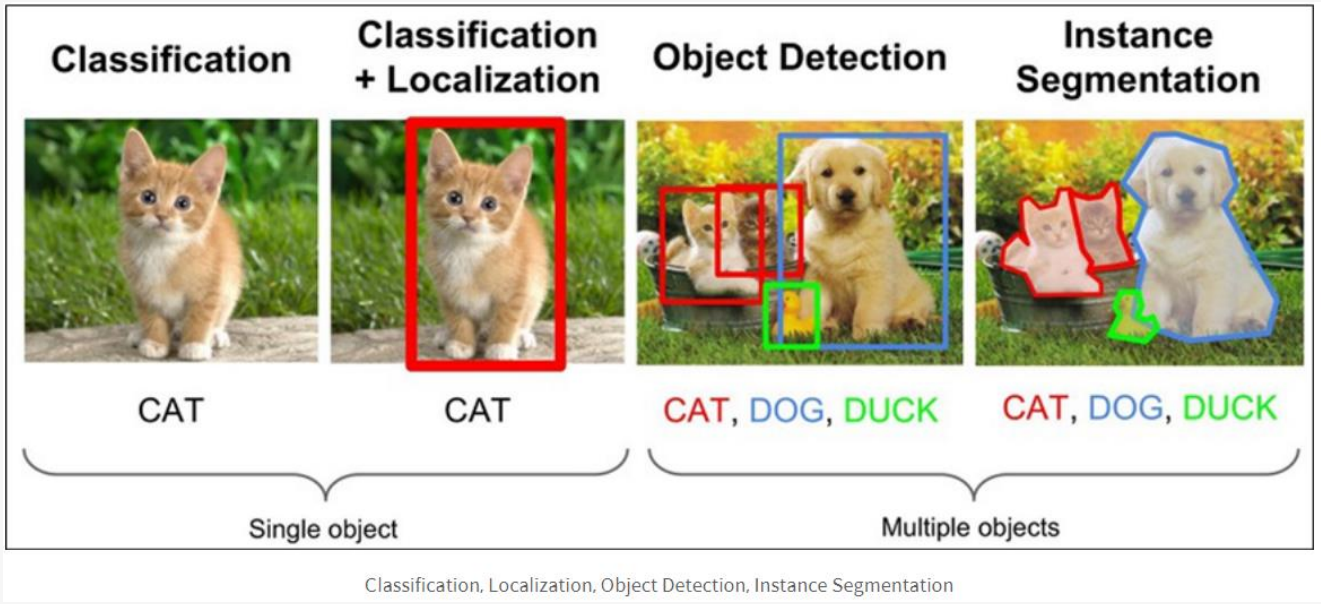
객체 검출과 분할

Object Detection vs Segmentation
Mask-RCNN

목차

Classfication, Localization, Detection, Segmentation



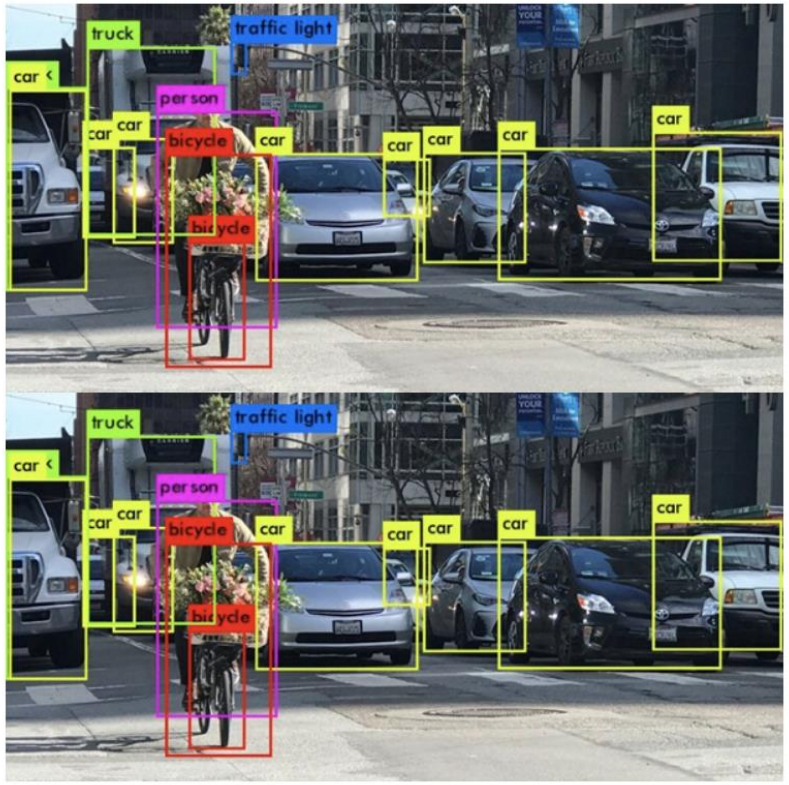
Classification, Localization, Object Detection, Instance Segmentation

1. Classification : 입력으로 주어진 이미지 안의 객체(Object)의 종류(Class 또는 Label)를 구분.
2. Localization : 이미지 안에 객체(Object)가 어느 위치에 있는지 위치 정보를 출력. 주로 Bounding Box를 사용. 하나의 이미지에 하나의 오브젝트 존재.
3. Object Detection : 보편적으로 Classification과 Localization이 동시에 수행. 하나의 이미지에 여러 오브젝트 검출.
4. Instance Segmentation : 객체(Object)를 box가 아닌 픽셀 단위로 정확한 영역 표시.

Object Detection

Classification은 단순히 이미지의 클래스를 분류하는 것

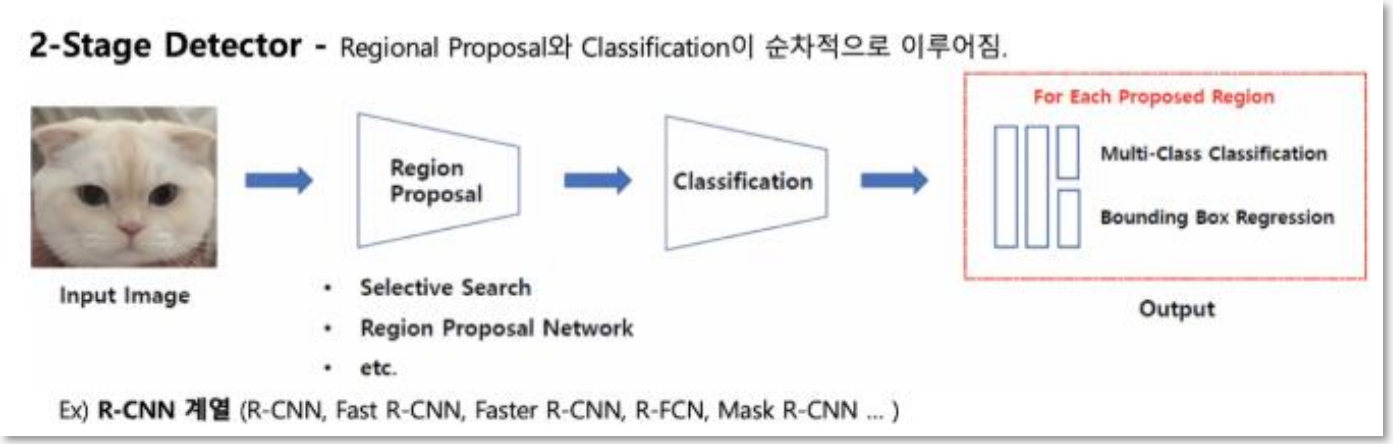
Object Detection 은 이미지를 분류할 뿐만 아니라 이미지 안에 있는 객체의 경계나 중앙점의 위치를 탐지.



초기에는 단순히 이미지 안에 객체를 인식하는 것으로 시작했으나,
현재는 프레임 단위로 객체를 인식하여 영상에서 바운딩 박스가
객체를 따라가는 형태로 발전.

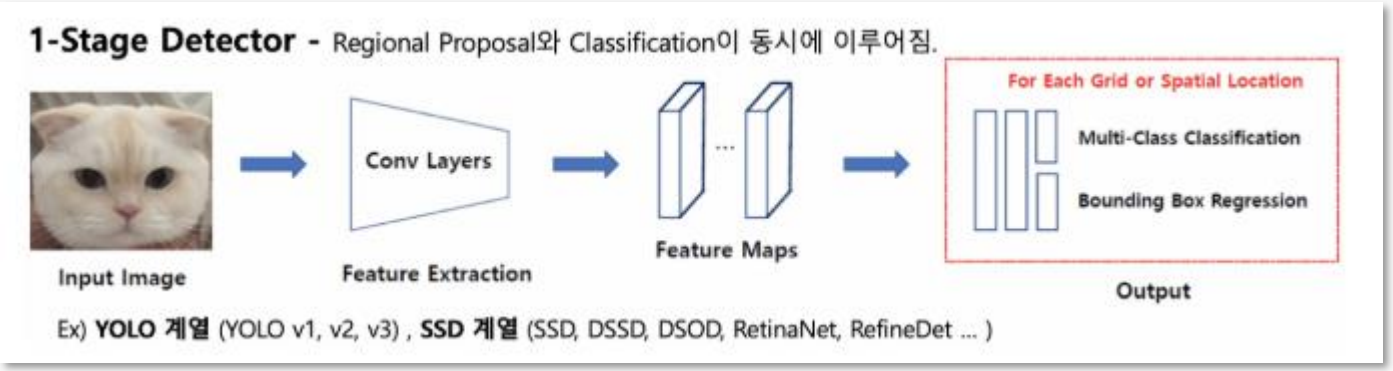
1. 객체 인식은 Detection, Tracking 등의 하위 기술을 모두 포함.
2. Detection은 특정 범위 내에 객체가 존재하는지 판단하고, 그 위치를 찾아내는 과정.
3. Tracking은 연속된 프레임에서 같은 객체끼리 연결하여 그것을 추적할 수 있도록 하는 개념.

Object Detection



- Region Proposal : 물체가 있을 법한 위치를 찾는 과정

물체의 위치를 찾는 문제(Localization)와 분류(Classification) 문제를 순차적으로 해결

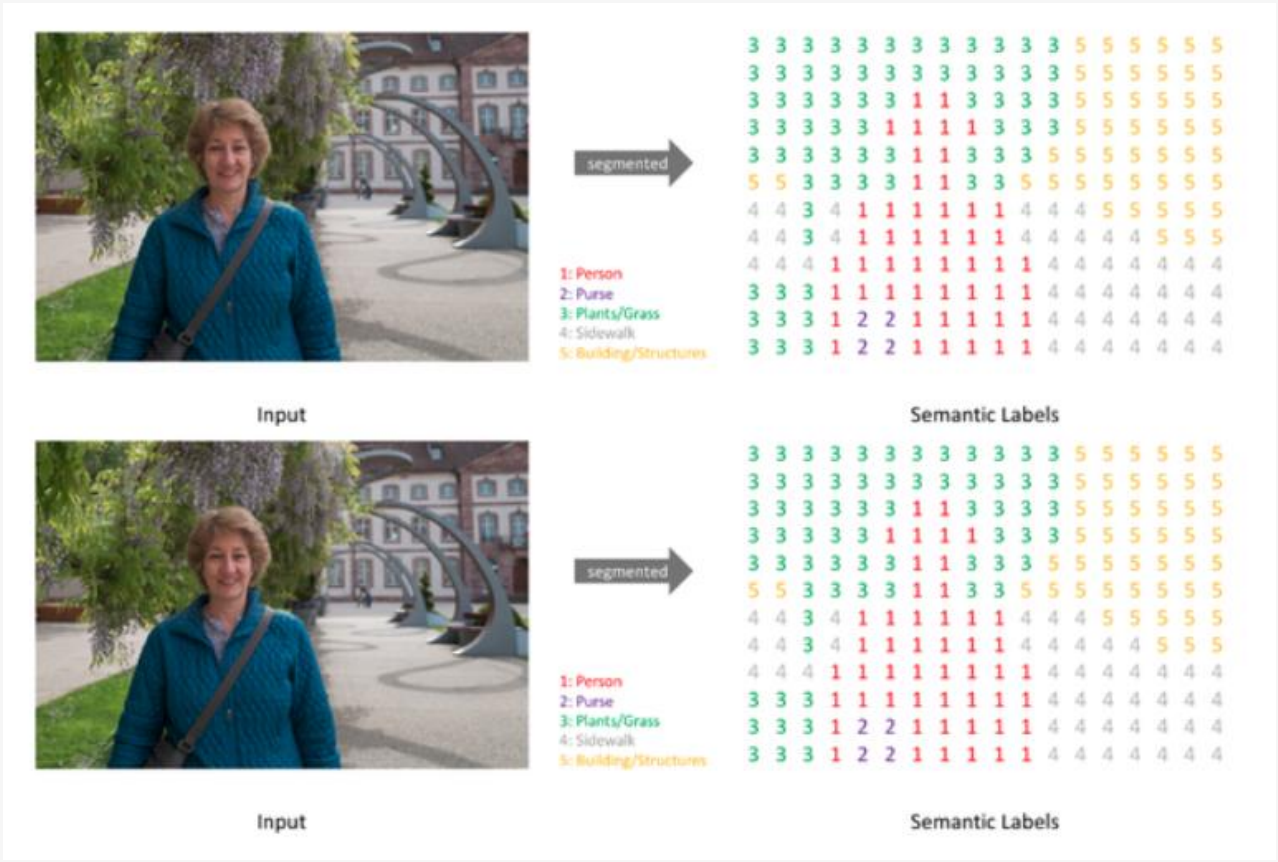


- 2-Stage Detector 방식보다 빠름.

물체의 위치를 찾는 문제(Localization)와 분류(Classification) 문제를 한번에 해결

Segmentation

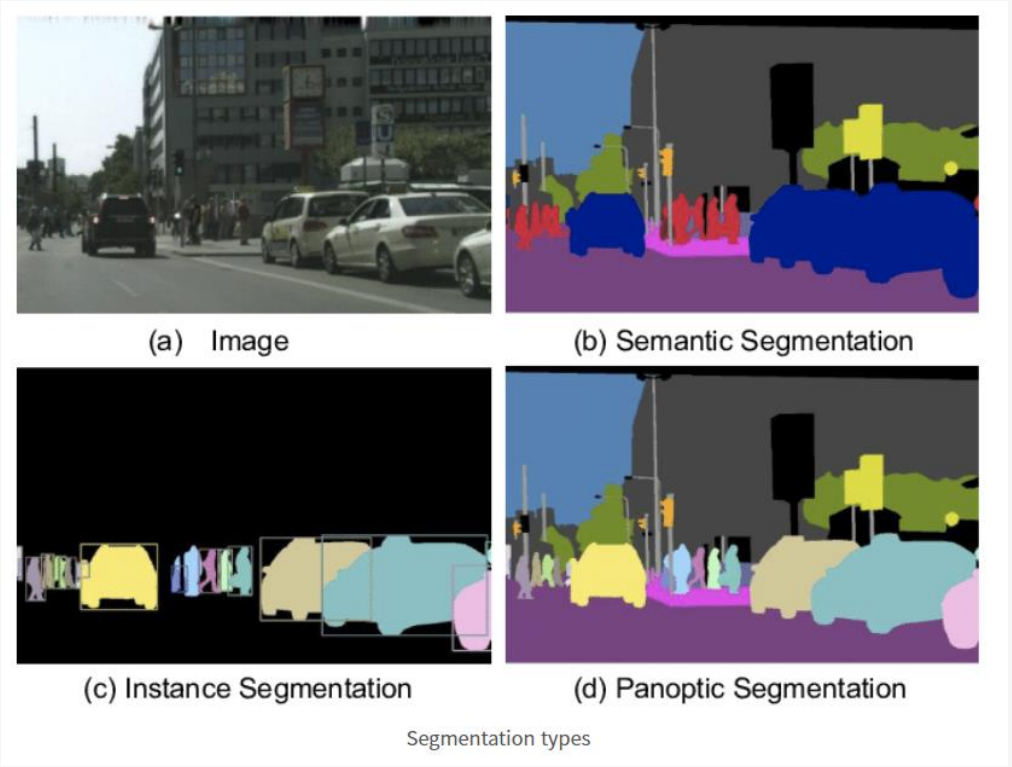
- Segmentation은 pixel-level로 claasification을 수행하는 것.
- Object Detection은 위치 기반으로 데이터를 추측하지만 Segmentation은 픽셀을 기준으로 데이터를 추측.



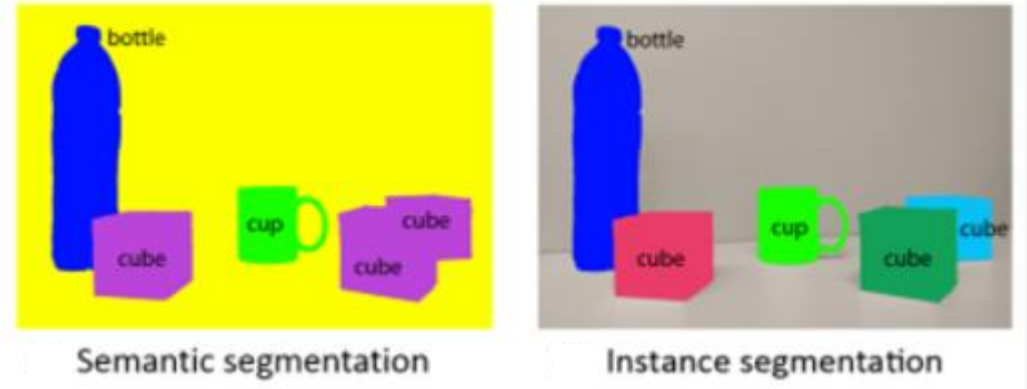
- 이미지에서 픽셀 단위로 뽑아낸 정보를 출력하도록 신경망 훈련
- 픽셀 레벨에서 이미지를 이해하여 하나의 이미지에서 여러 물체 인식 가능
-의료영상, 자율주행차, 보안, 위성, 항공사진 등
다양한 분야에서 이미지 분할 활용.

Segmentation

1. Semantic segmenation은 이미지 내에서 객체가 속한 Class가 무엇인지에 대해서만 판단. FCN이 가장 대표적인 기법
2. Instance segmentation은 객체의 Class와 더불어 그 안의 Instance들 간의 구분이 가능. Mask RCNN이 가장 대표적인 기법.
3. Panoptic segmentation은 위 두 방법을 합친 것. 모든 픽셀에 대해 Class를 판단하고 각 Instance 별로 마스킹 을 수행. DeepLab이 가장 대표적인 기법.

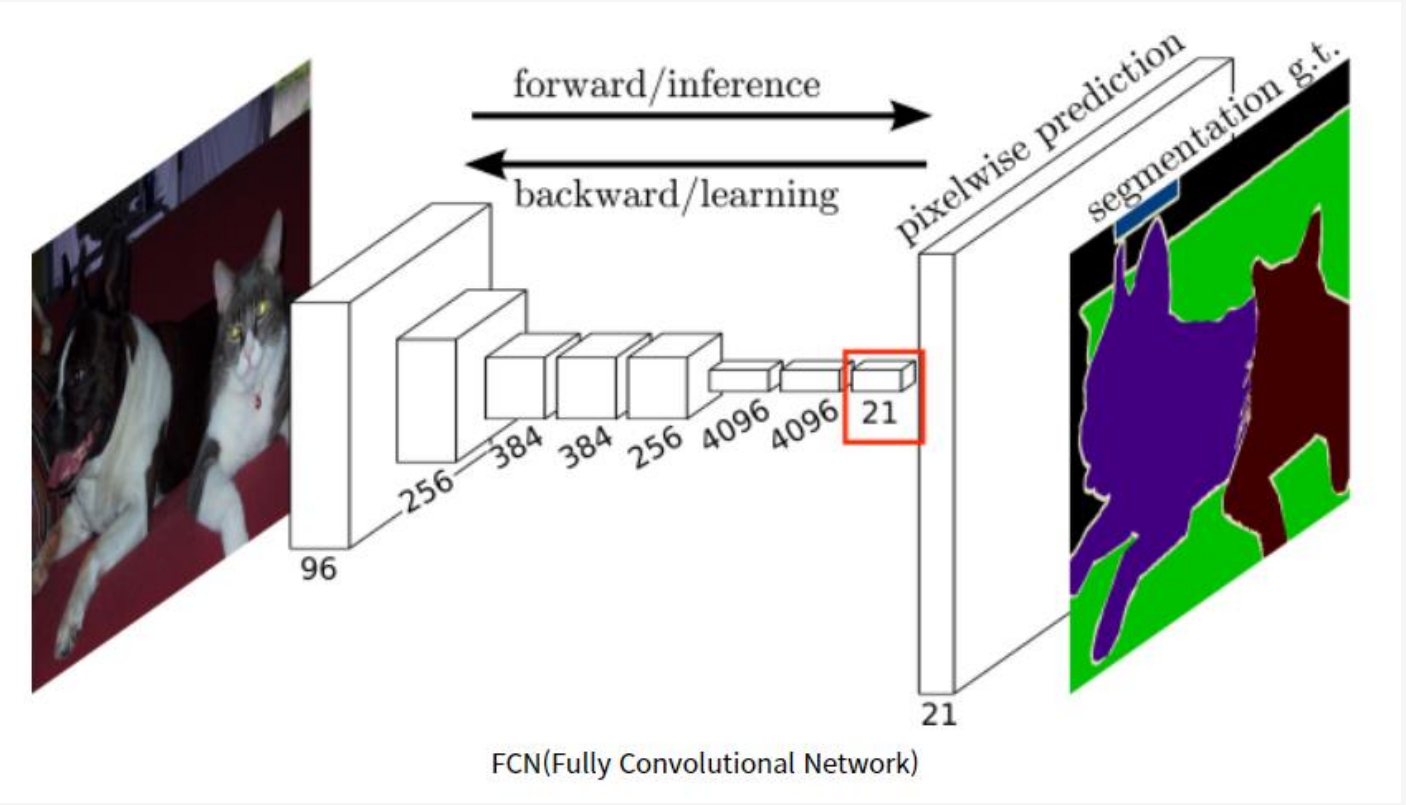


Segmentation types



기술적 측면에서는 Instance, Panoptic segmentation 방식이 한 단계 더 고차원적이라고 볼 수 있지만, Semantic segmenation 방식에 비해 더 많은 계산 비용이 들어간다는 단점이 있음.

FCN(Fully Convolutional Network)

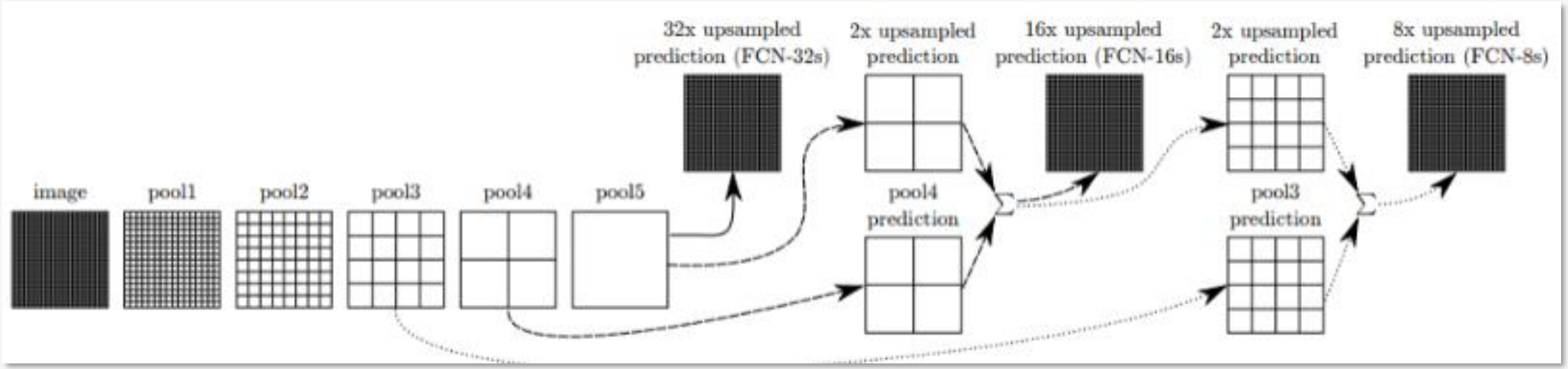


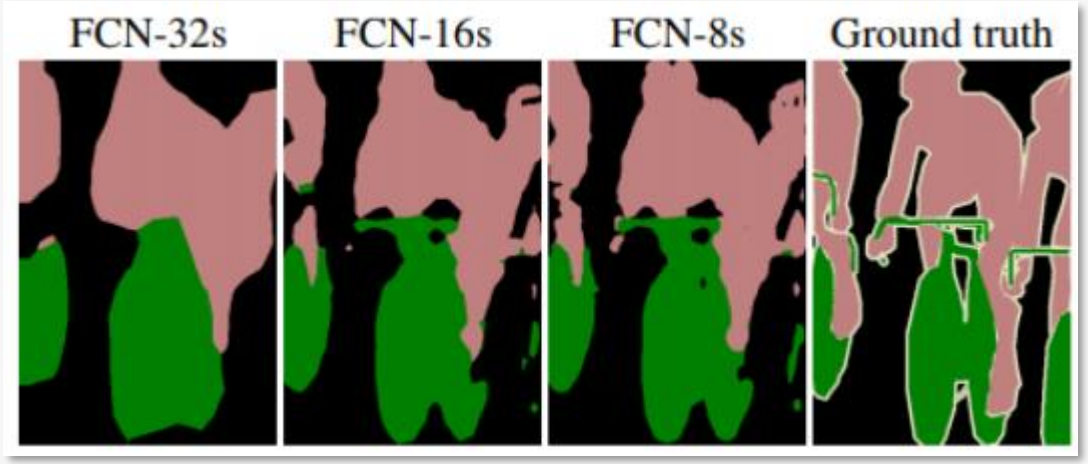
FCN(Fully Convolutional Network)

- FCN은 이미지 분할 모델의 종류 중 하나.
- Conv Block(Conv – Pooling) 구조를 여러층 쌓아 Feature extractor를 구성하고, Conv Layer를 사용해 heatmap을 예측.
- FCN은 복원하는 Decoder 과정에서 사이즈가 작은 Convolutional Layer에서 바로 원본 이미지 사이즈와 비슷한 크기의 형태로 Up Sampling(Skip Connection 기법)을 수행

FCN(Fully Convolutional Network)

Skip Connection

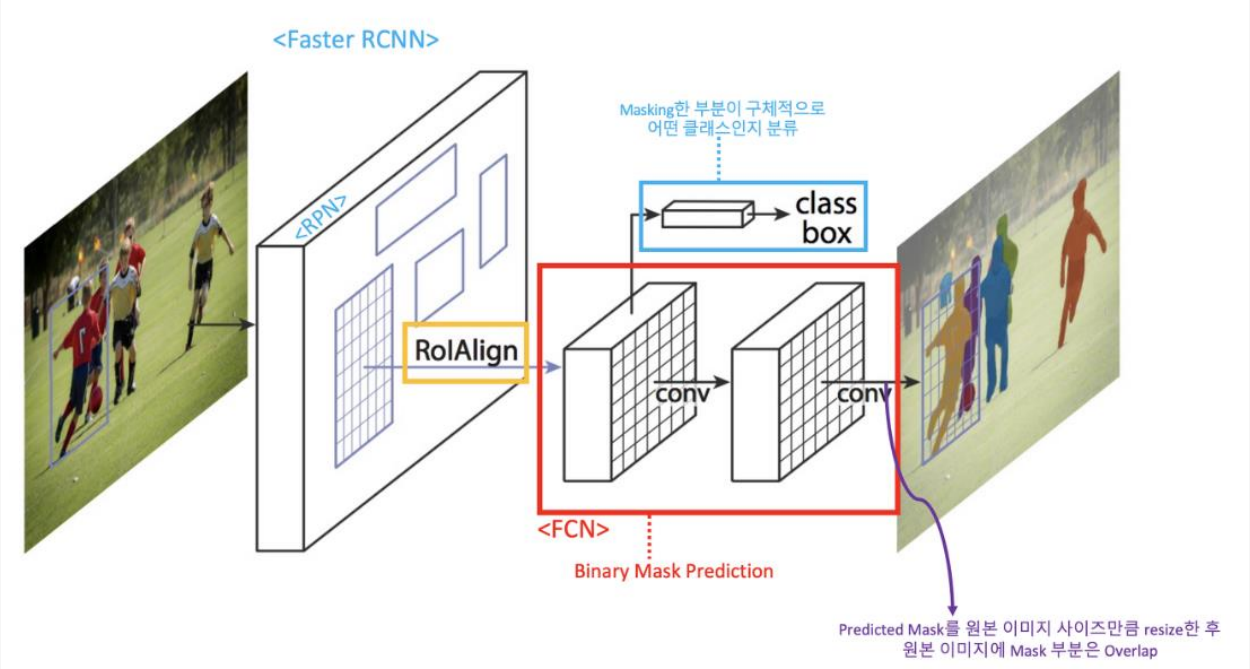




- Polling이 진행될수록 낮아지는 해상도를 높이기 위해 이전 Pooling layer를 활용.
- FCN-32s, FCN-16s, FCN-8s 의 세 가지로 결과 출력(숫자는 Stride 의미).

Mask RCNN

Mask RCNN은 RPN(Region Proposal Network)을 활용한 Faster RCNN Object Detection 모델과 약간의 변형된 FCN 으로 구성



Mask RCNN 특징

1. Fast RCNN에서 사용한 ROI Pooling이 아닌 **ROI Align** 기법을 사용.
2. FCN을 통해 분류를 수행할 때 Multi-Class Classification이 아닌 **Binary Classification**을 수행.

ROI Align 기법 도입 이유

기존 ROI Pooling은 영역 제안에 해당하는 특징맵을 7X7 로 만듦.

=> Feature Map에서 ROI Feature Map으로 변환할 때 높이, 너비 사이즈가 정수. (소수점 값 버림, 이미지의 중요한 부분인 ROI Feature를 일부 손실)

Mask RCNN 동작순서

1. 이미지 전처리
1) 입력 이미지 resize (bilinear interpolation)
2) Backbone network의 입력 사이즈에 맞게 이미지 padding

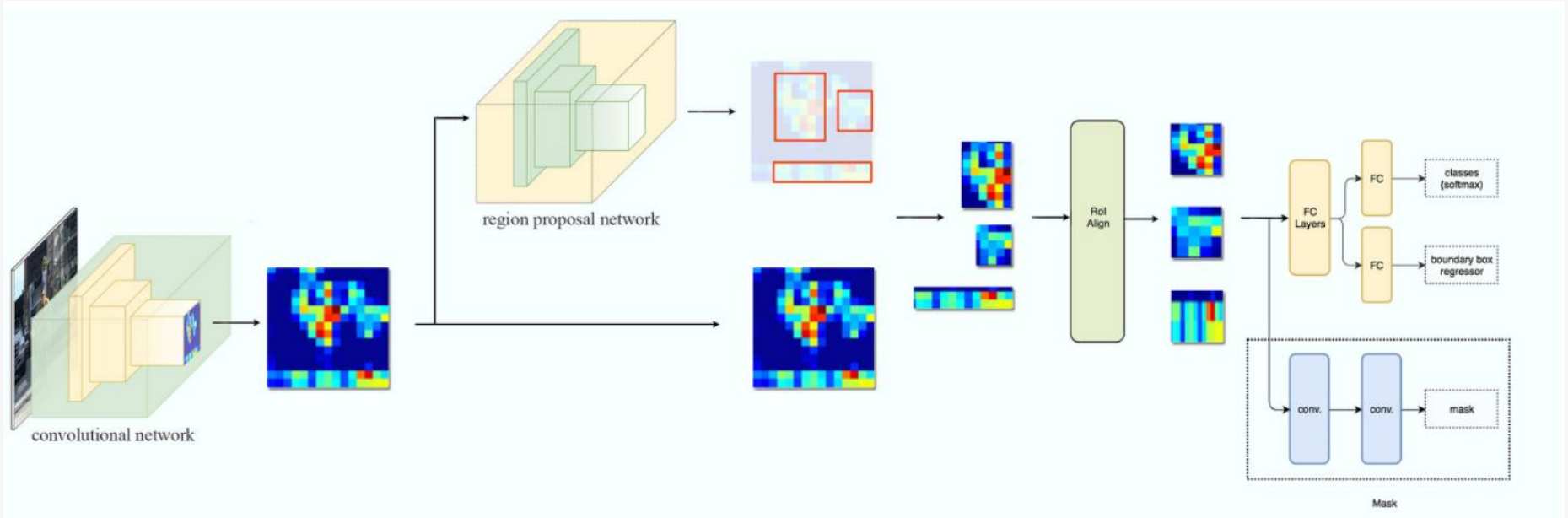
2. Backbone
1) ResNet을 통해 여러 층의 feature map (C1, C2, C3, C4, C5) 생성
2) FPN을 통해 최종 feature map (P2, P3, P4, P5, P6) 생성

3. RPN
1) 각 픽셀 별로 anchor box를 생성하고, 그 box에 대한 objectness와 bbox regression을 구함.
2) Bbox regression (box 위치)에 맞게 이전에 만든 feature map을 잘라 RoI를 만듦.
3) NMS를 통해 RoI를 줄임.

4. RoI Align
1) 크기가 다른 RoI들의 크기를 동일하게 맞춰준다.

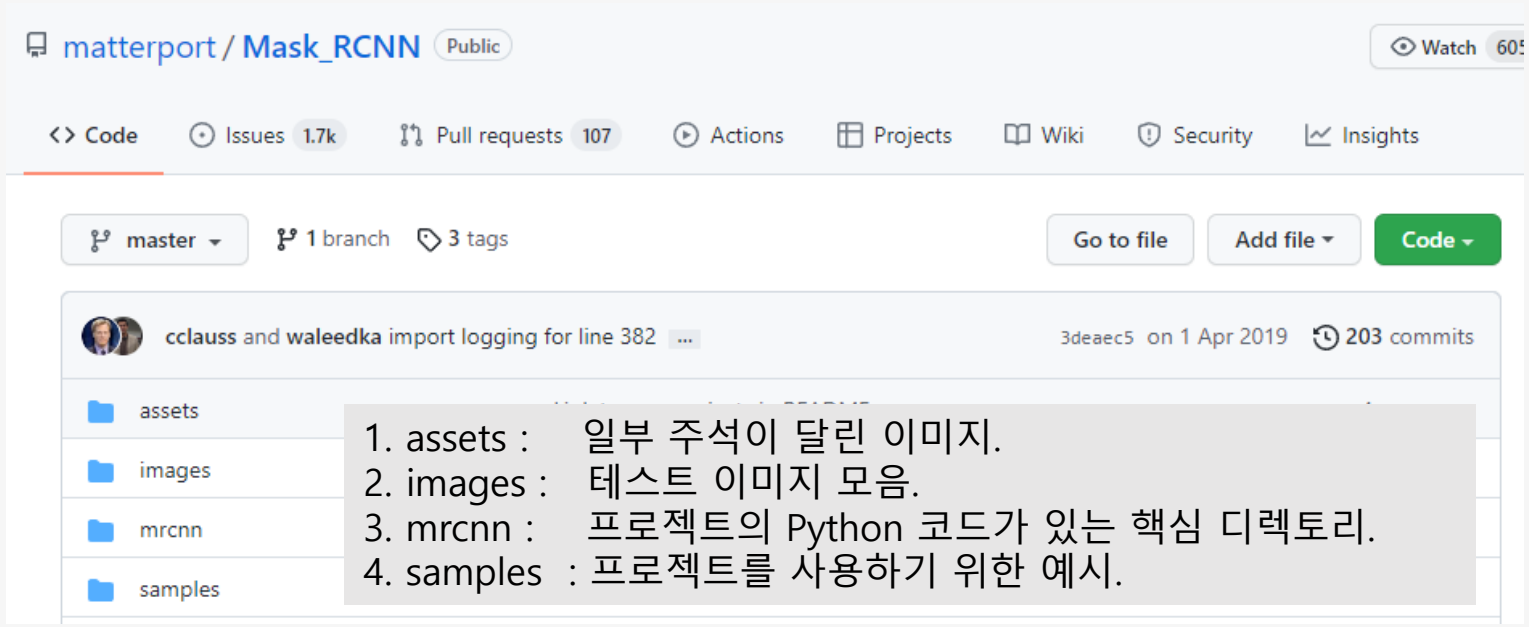
5. 3개의 branch 통과
1) Classfication 으로 클래스 예측
2) bbox regression 으로 box 좌표 수정
3) FCN 기반의 mask branch 에서 binary mask 생성

classification / bbox regression은 Fully Connected Layer를 사용한다.



Mask RCNN 구현

Github 에 Mask RCNN 오픈 소스 사용.
https://github.com/matterport/Mask_RCNN



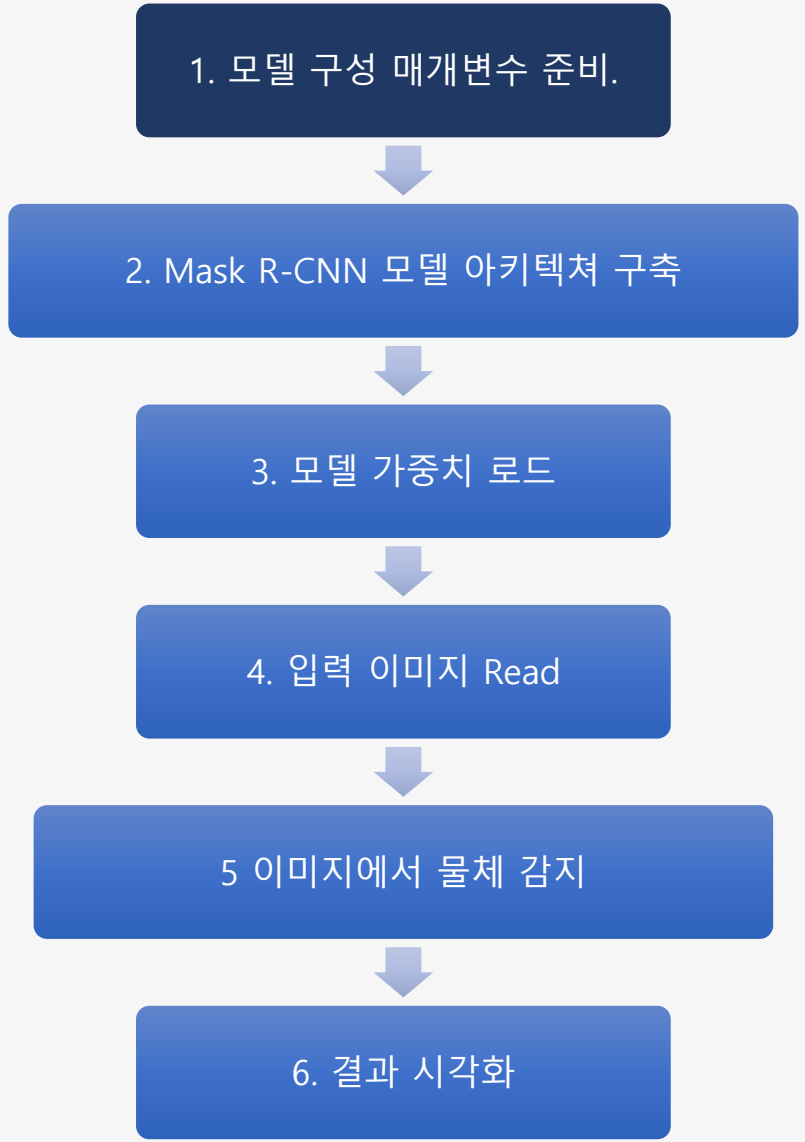
TensorFlow 1(≥1.3.0) 설치

1. 모델 구성 매개변수 준비.
2. Mask R-CNN 모델 아키텍쳐 구축
3. 모델 가중치 로드
4. 입력 이미지 Read
5 이미지에서 물체 감지
6. 결과 시각화

Mask RCNN 구현

```python
import mrcnn.config

class SimpleConfig(mrcnn.config.Config):
  NAME = "coco_inference"

  GPU_COUNT = 1
  IMAGES_PER_GPU = 1

  NUM_CLASSES = 81
```

- NAME = 구성의 고유한 이름
- GPU_COUNT , IMAGES_PER_GPU : BATCH SIZE를 결정하는데 사용
- BATCH_SIZE = IMAGES_PER_GPU * GPU_COUNT
  (2*1=2일 경우 2개의 이미지가 한번에 모델에 공급됨을 의미)
- NUM_CLASSES : 클래스 개수, 기본값은 1,
  COCO 데이터 세트 이미지에는 80개의 클래스가 있음,
  배경은 추가 클래스로 간주, 총 81개 클래스

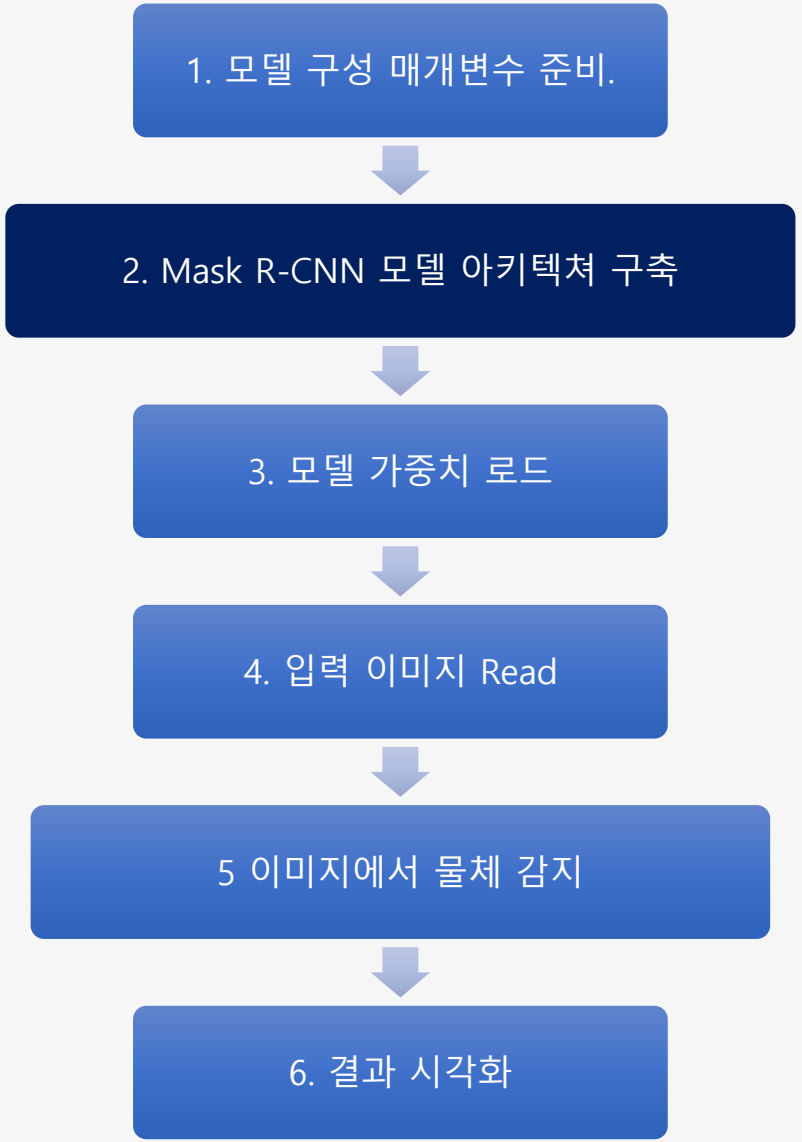


Mask RCNN 구현

```
import mrcnn.model

model = mrcnn.model.MaskRCNN(mode="inference",
                             config=SimpleConfig(),
                             model_dir=os.getcwd())

model.keras_model.summary()
```

MaskRCNN 클래스에 들어갈 세 가지 매개변수

1. mode : “training” 또는 “inference”.
2. config : 구성 클래스의 인스턴스.
3. model_dir : 훈련 로그 및 훈련 가중치를 저장하는 디렉토리.

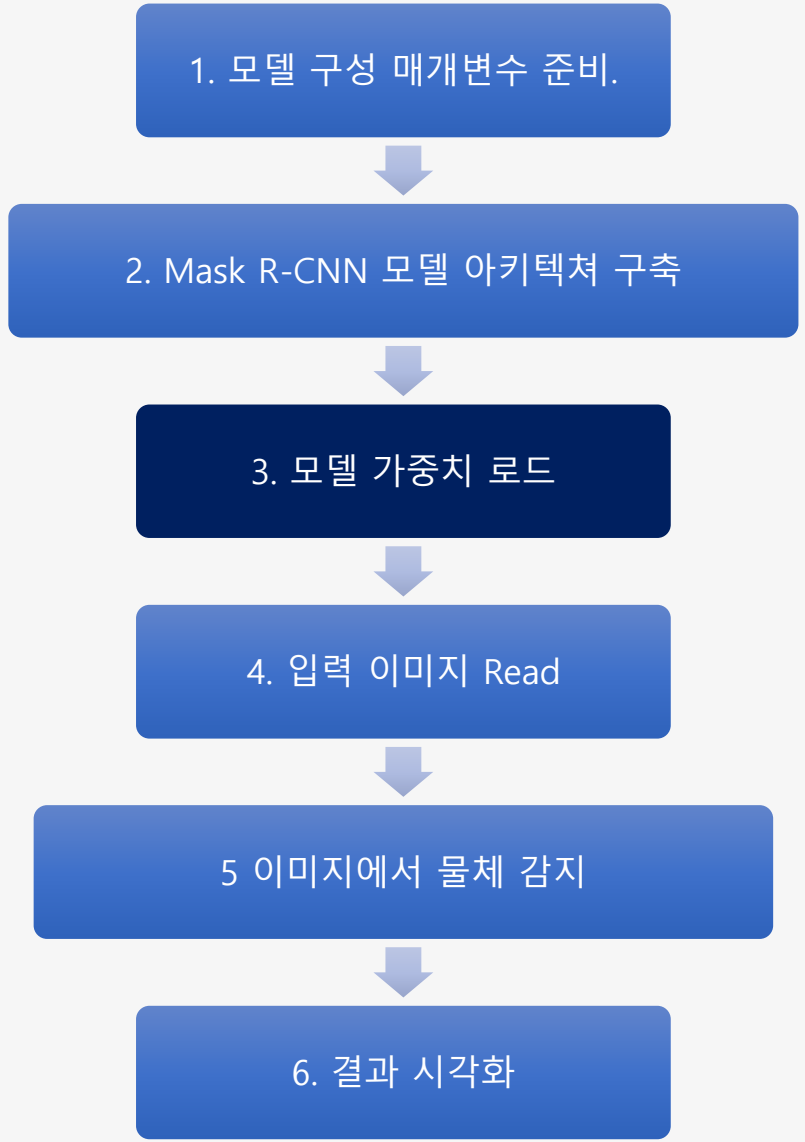


Mask RCNN 구현

```
model.load_weights(filepath="mask_rcnn_coco.h5",
                   by_name=True)
```

load_weights 클래스에 들어갈 두 가지 매개변수

1. filepath : 가중치 파일의 경로를 허용.
2. by_name : True 이면 이름에 따라 각 레이어에 가중치가 할당됨.

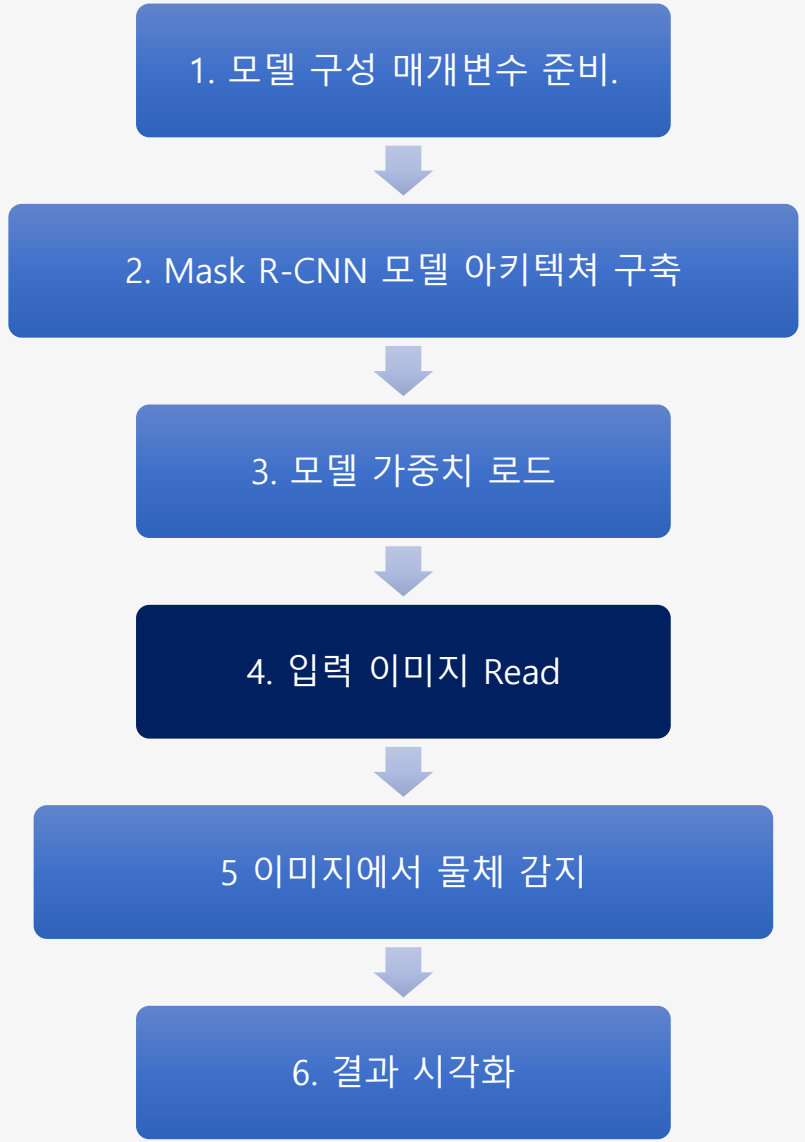


Mask RCNN 구현

```
import cv2

image = cv2.imread("3627527276_6fe8cd9bfe_z.jpg")
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

모델이 생성되고 가중치가 로드되면 이미지를 읽고 모델에 공급.

OpenCV를 사용해 이미지를 읽고, 색상채널 RGV로 재정렬.



Mask RCNN 구현

```
r = model.detect(images=[image],
                 verbose=0)
```

detect 클래스에 들어갈 두 가지 매개변수

1. Images : 이미지 목록.
2. verbose : 1로 설정 시 로그메시지 인쇄.

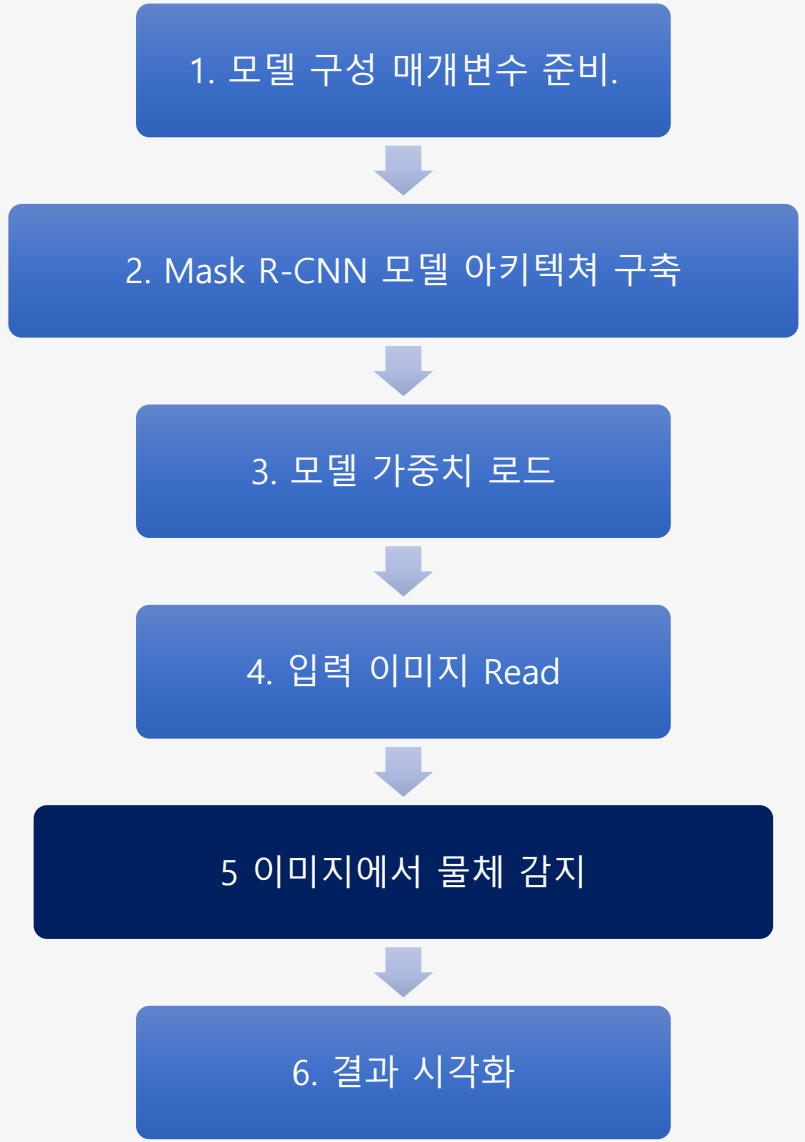
⇒ 이때 이미지 개수가 다르면 에러 발생
(인수의 길이와 속정이 같아야함)

```
... File "D:\Object Detection\Pure Keras\mrcnn\model.py", in detect assert len(images) ==
self.config.BATCH_SIZE, "len(images) must be equal to BATCH_SIZE" AssertionError: len(images) must be
equal to BATCH_SIZE
```

detect() 메소드는 감지된 개체에 대한 정보가 들어있는 dictionary를 반환.
첫번째 이미지에 대한 정보를 반환하기 위해 인덱스0으로 추출 r = r[0]

반환된 dictionary의 네 가지 key

1. rois : 감지된 각 개체 주변의 상자
2. class_ids : 개체 클래스 ID
3. scores : 각 개체 클래스 점수
4. masks : 마스크



Mask RCNN 구현

```
import mrcnn.visualize

CLASS_NAMES = ['BG', 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'tr
               'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'h
               'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie
               'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', '
               'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl
               'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake'
               'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard',
               'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy

r = r[0]

mrcnn.visualize.display_instances(image=image, #detection box가 그려진 이미지
                                  boxes=r['rois'],  #detection box
                                  masks=r['masks'], #감지된 마스크
                                  class_ids=r['class_ids'], #감지된 클래스 id
                                  class_names=CLASS_NAMES, #데이터 세트의 클래스 이름 목록
                                  scores=r['scores'])#각 개체에 대한 예측 점수
```

감지된 개체를 시각화 하기 위해 mrcnn.visualize 스크립트 사용.

mcrnn.visualize.display_instance() 기능은 detection box , 마스크, 클래스 이름 및 점수를 표시하는데 사용.

클래스 레이블 중 BG는 배경용을 의미.

1. 모델 구성 매개변수 준비.
2. Mask R-CNN 모델 아키텍쳐 구축
3. 모델 가중치 로드
4. 입력 이미지 Read
5 이미지에서 물체 감지
6. 결과 시각화

Mask RCNN 결과

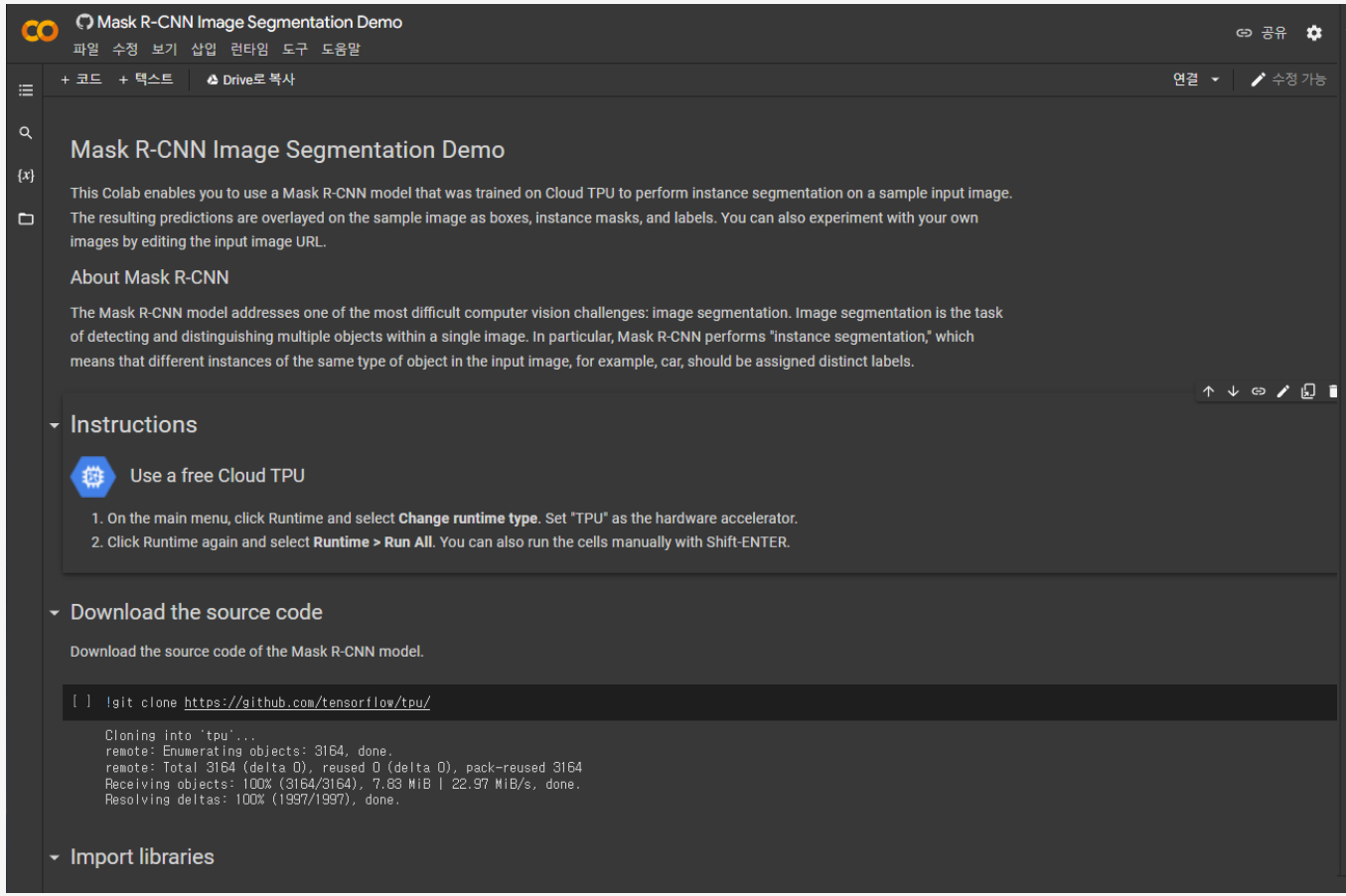
```
import mrcnn
import mrcnn.config
import mrcnn.model
import mrcnn.visualize
import cv2
import os

CLASS_NAMES = ['BG', 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train',
               'truck', 'boat', 'traffic light', 'fire hydrant', 'stop sign', 'parking meter',
               'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear',
               'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase',
               'frisbee', 'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat',
               'baseball glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle',
               'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',
               'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut',
               'cake', 'chair', 'couch', 'potted plant', 'bed', 'dining table', 'toilet',
               'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave',
               'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors',
               'teddy bear', 'hair drier', 'toothbrush']

class SimpleConfig(mrcnn.config.Config):
    NAME = "coco_inference"

    GPU_COUNT = 1
    IMAGES_PER_GPU = 1

    NUM_CLASSES = len(CLASS_NAMES)

model = mrcnn.model.MaskRCNN(mode="inference",
                             config=SimpleConfig(),
                             model_dir=os.getcwd())

model.load_weights(filepath="mask_rcnn_coco.h5",
                   by_name=True)

image = cv2.imread("sample2.jpg")
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

r = model.detect([image], verbose=0)

r = r[0]

mrcnn.visualize.display_instances(image=image,
                                  boxes=r['rois'],
                                  masks=r['masks'],
                                  class_ids=r['class_ids'],
                                  class_names=CLASS_NAMES,
                                  scores=r['scores'])
```

< 전체 실행 소스

< 실행 결과

Google colab을 통한 실습

https://colab.research.google.com/github/tensorflow/tpu/blob/master/models/official/mask_rcnn/mask_rcnn_demo.ipynb#scrollTo=X0G-tk6wakcr



감사합니다.

참고자료 출처
https://techblog-history-younghunjo1.tistory.com/193
https://velog.io/@qksekf
https://benlee73.tistory.com/32
https://ganghee-lee.tistory.com/40
https://blog.paperspace.com/mask-r-cnn-in-tensorflow-2-0/
https://github.com/Kanghee-Lee/Mask-RCNN_TF/blob/master/demo/balloon_data.ipynb
https://colab.research.google.com/github/tensorflow/tpu/blob/master/models/official/mask_rcnn/mask_rcnn_demo.ipynb#scrollTo=X0G-tk6wakcr