

*Seohyun Lim*

BOAZ

LAB.

*limseo2110@gmail.com*

---

# MMDetection, Detectron2

2023.02.08

---



---

# Table of Content

---

- ❑ 1. Overview
- ❑ 2. MMDetection
- ❑ 3. Detectron2



# 1. Overview

## ❑ MMDetection and Detectron2

	MMDetection	Detectron2
특징	<ul style="list-style-type: none"><li>- 전체 프레임워크를 모듈 단위로 분리해 관리할 수 있음</li><li>- 많은 프레임워크를 지원함</li><li>- 다른 라이브러리에 비해 빠름</li></ul>	<ul style="list-style-type: none"><li>- 전체 프레임워크를 모듈 단위로 분리해 관리할 수 있음</li><li>- OD 외에도 Segmentation, Pose prediction 등의 알고리즘을 지원함</li></ul>
지원 모델	<ul style="list-style-type: none"><li>- Fast R-CNN</li><li>- SSD</li><li>- YOLO v3</li><li>- DETR</li><li>- ...</li></ul>	<ul style="list-style-type: none"><li>- Faster R-CNN</li><li>- RetinaNet</li><li>- Mask R-CNN</li><li>- DETR</li><li>- ...</li></ul>



## 2. MMDetection

---

### □ MMDetection

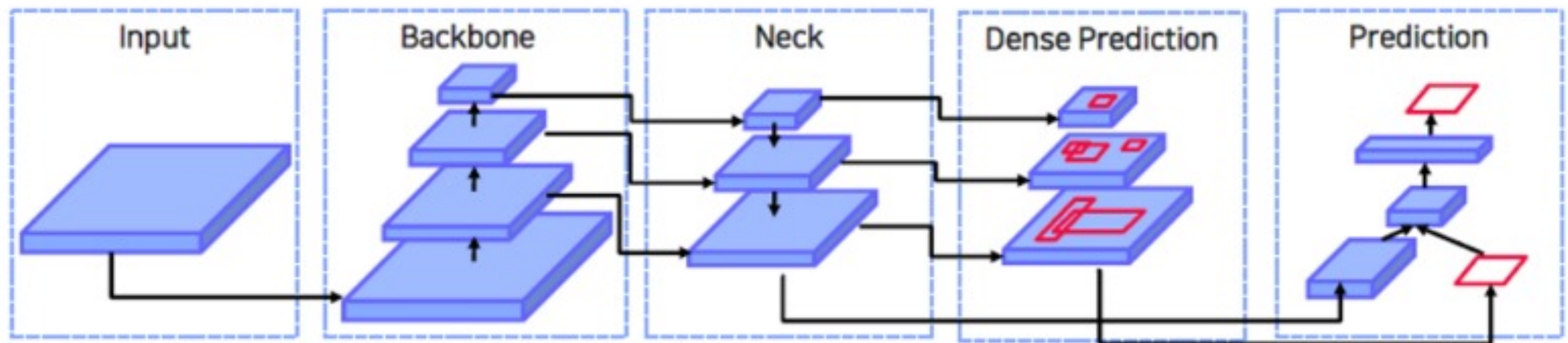
- Pytorch 기반의 Object Detection 오픈소스 라이브러리
- 많은 모델을 지원, 허들이 높지만 쉽게 사용할 수 있음
- 단점은 custom을 하기 위해서는 라이브러리에 대한 완벽한 이해가 필요



## 2.1. MMDetection 구조

### □ 기본 구조

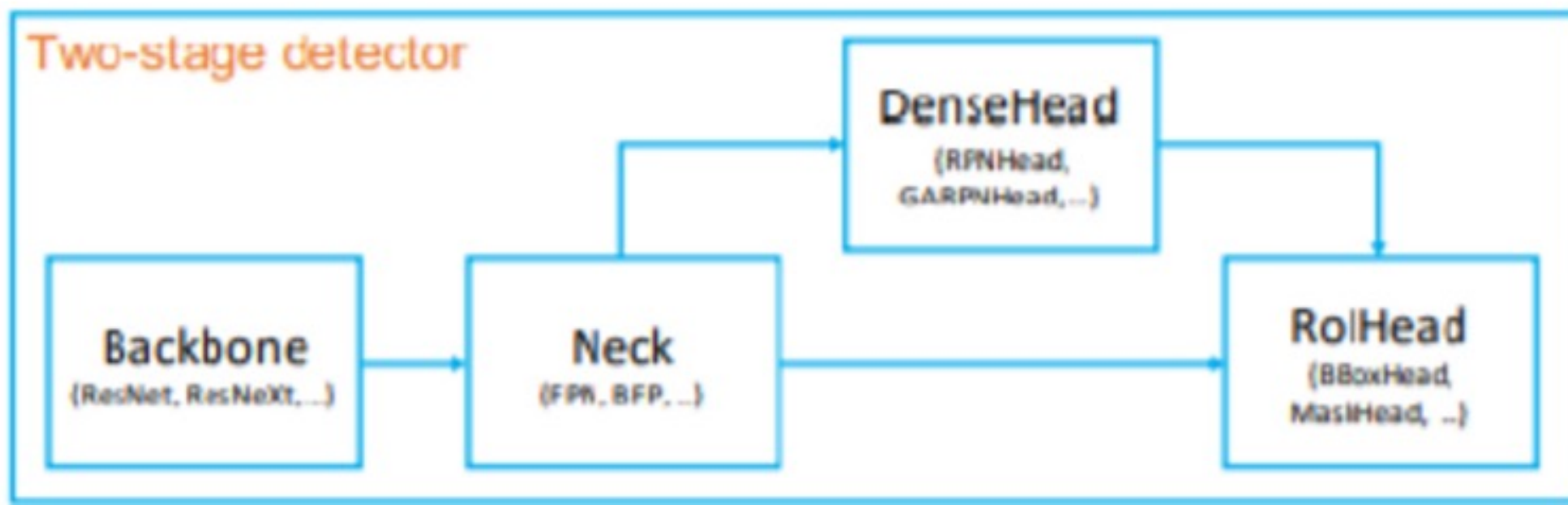
- 최근 Object Detection은 Neck을 활용
- Neck인 Feature map을 활용해 RoI를 예측하고, 최종적으로 클래스와 bbox를 추측



## 2.1. MMDetection 구조

### □ MMDetection Pipeline

- Backbone - 입력 이미지를 특징 맵으로 변형
- Neck - backbone 과 head를 연결, feature map을 재구성 (ex.FPN)
- DenseHead - 특징 맵의 dense location (localization)을 수행하는 부분임
- RoIHead - RoI 특징을 입력으로 받아 box 분류, 좌표 회귀, 클래스 분류 등을 예측하는 부분임



## 2.2. Config File

### ❑ Config File 사용방법

- python tools/train.py configs/\_boostcamp/htc\_class10/htc\_train.py
- tools/train.py – model을 학습시키는 코드가 적힌 파일
- configs/~~ - model에 대한 여러 config 정보가 적힌 파일

### Train

```
cd /opt/ml/detection/baseline/mmdetection 에서 시작
```

```
python tools/train.py configs/_boostcamp/htc_class10/htc_train.py
```

```
def parse_args():  
    parser = argparse.ArgumentParser(description='Train a detector')  
    parser.add_argument('config', help='train config file path')  
  
    cfg = Config.fromfile(args.config)
```

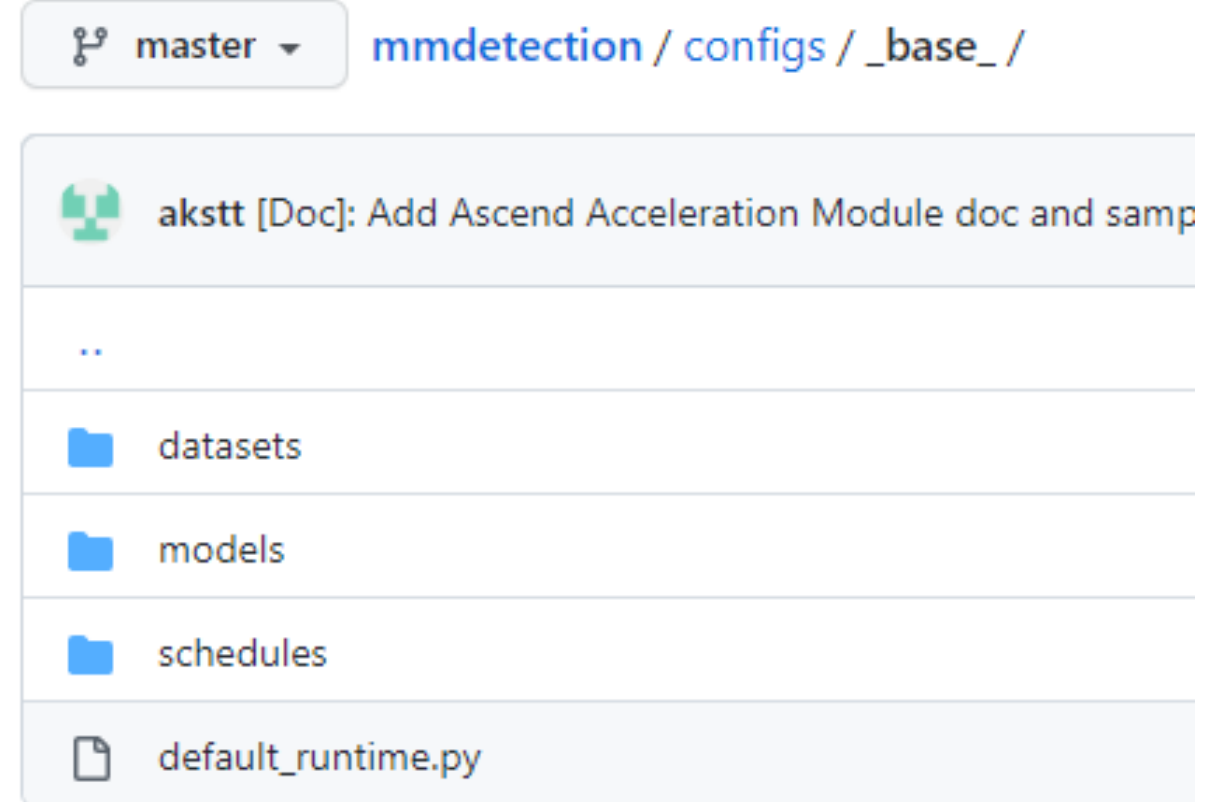
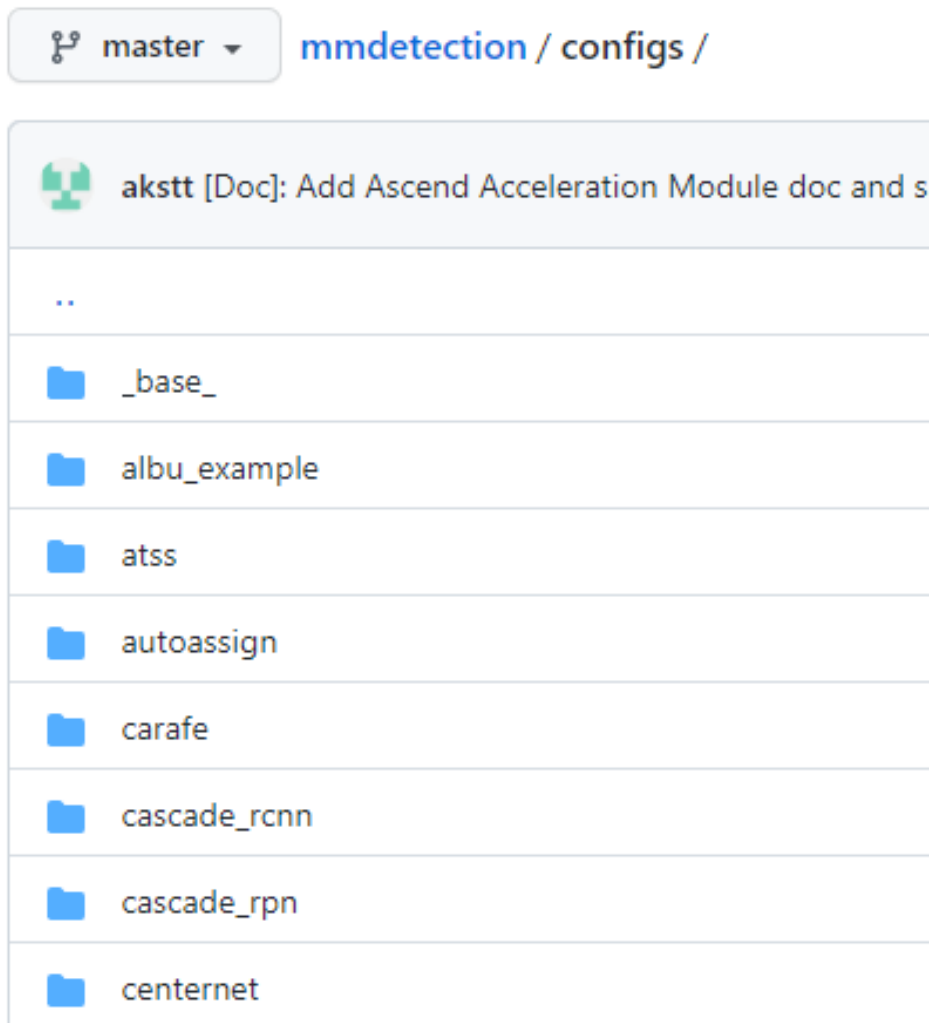
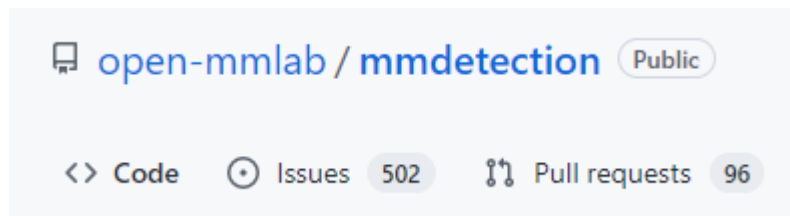
```
model = build_detector(  
    cfg.model,  
    train_cfg=cfg.get('train_cfg'),  
    test_cfg=cfg.get('test_cfg'))  
model.init_weights()
```



## 2.2. Config File

### ❑ Config File

- configs/\_base\_ 폴더에 가장 기본이 되는 config 파일이 존재
- config를 통해 dataset, model, schedule, default\_runtime 등을 정의 가능
- 틀에 갖춰진 config를 상속받고 필요한 부분을 수정해서 사용





## 2.3. Dataset

- ❑ /\_base\_/datasets
  - # dataset settings
    - dataset\_type
    - data\_root
    - img\_norm\_cfg
  - train\_pipeline
  - test\_pipeline
  - data
  - evaluation

master ▾

[mmdetection](#) / [configs](#) / [\\_base\\_](#) / [datasets](#) / [cityscapes\\_detection.py](#) / <> Jump to ▾

56 lines (56 sloc) | 1.89 KB

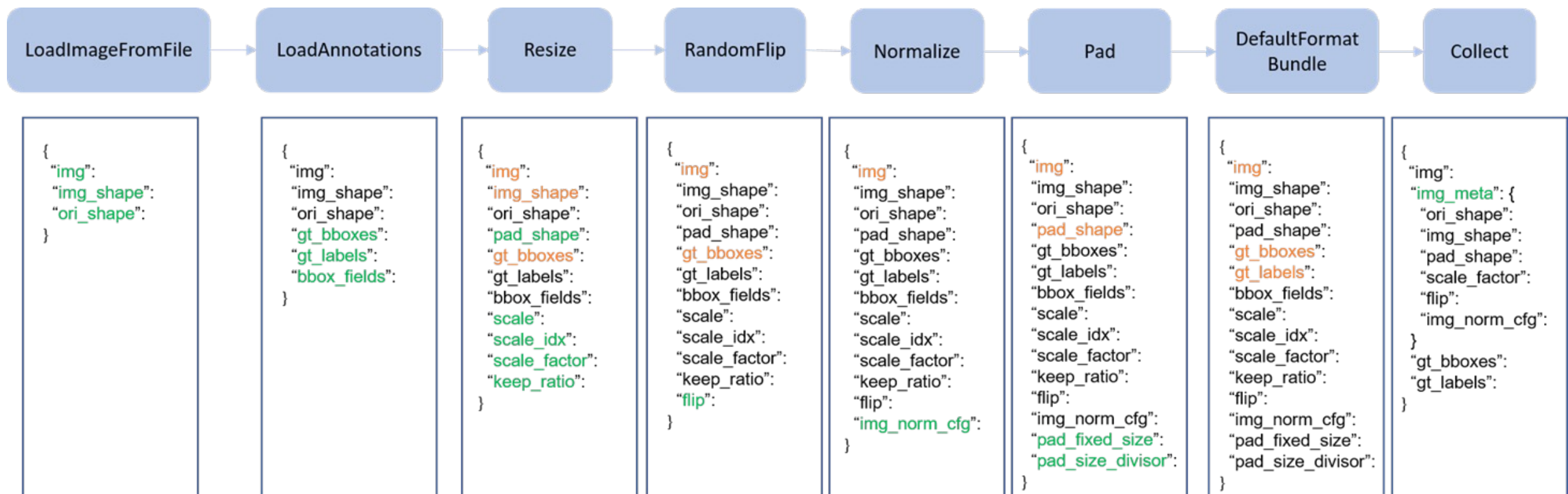
```
1  # dataset settings
2  dataset_type = 'CityscapesDataset'
3  data_root = 'data/cityscapes/'
4  img_norm_cfg = dict(
5      mean=[123.675, 116.28, 103.53], std=[58.395, 57.12, 57.375], to_rgb=True)
```



## 2.3. Dataset

### □ /\_base\_/datasets

- # dataset settings
- dataset\_type
- data\_root
- img\_norm\_cfg
- train\_pipeline
- test\_pipeline
- data
- evaluation



## 2.3. Dataset

### □ /\_base\_/datasets

- # dataset settings
- dataset\_type
- data\_root
- img\_norm\_cfg
- train\_pipeline
- test\_pipeline
- data
- evaluation

master ▾

[mmdetection](#) / [configs](#) / [\\_base\\_](#) / [datasets](#) / [cityscapes\\_detection.py](#) / <> Jump to ▾

```
train_pipeline = [  
    dict(type='LoadImageFromFile'),  
    dict(type='LoadAnnotations', with_bbox=True),  
    dict(  
        type='Resize', img_scale=[(2048, 800), (2048, 1024)], keep_ratio=True),  
    dict(type='RandomFlip', flip_ratio=0.5),  
    dict(type='Normalize', **img_norm_cfg),  
    dict(type='Pad', size_divisor=32),  
    dict(type='DefaultFormatBundle'),  
    dict(type='Collect', keys=['img', 'gt_bboxes', 'gt_labels']),  
]  
test_pipeline = [  
    dict(type='LoadImageFromFile'),  
    dict(  
        type='MultiScaleFlipAug',  
        img_scale=(2048, 1024),  
        flip=False,  
        transforms=[  
            dict(type='Resize', keep_ratio=True),  
            dict(type='RandomFlip'),  
            dict(type='Normalize', **img_norm_cfg),  
            dict(type='Pad', size_divisor=32),  
            dict(type='ImageToTensor', keys=['img']),  
            dict(type='Collect', keys=['img']),  
        ]  
    )  
]
```



## 2.3. Dataset

### □ /\_base\_/datasets

- # dataset settings
- dataset\_type
- data\_root
- img\_norm\_cfg
- train\_pipeline
- test\_pipeline
- data
- evaluation

master ▾

[mmdetection](#) / [configs](#) / [\\_base\\_](#) / [datasets](#) / [cityscapes\\_detection.py](#) / <> Jump to ▾

### tools/train.py


```
datasets = [build_dataset(cfg.data.train)]
if len(cfg.workflow) == 2:
    assert 'val' in [mode for (mode, _) in cfg.workflow]
    val_dataset = copy.deepcopy(cfg.data.val)
    val_dataset.pipeline = cfg.data.train.get(
        'pipeline', cfg.data.train.dataset.get('pipeline'))
```

```
data = dict(
    samples_per_gpu=1,
    workers_per_gpu=2,
    train=dict(
        type='RepeatDataset',
        times=8,
        dataset=dict(
            type=dataset_type,
            ann_file=data_root +
                'annotations/instancesonly_filtered_gtFine_train.json',
            img_prefix=data_root + 'leftImg8bit/train/',
            pipeline=train_pipeline)),
    val=dict(
        type=dataset_type,
        ann_file=data_root +
            'annotations/instancesonly_filtered_gtFine_val.json',
            img_prefix=data_root + 'leftImg8bit/val/',
            pipeline=test_pipeline),
    test=dict(
        type=dataset_type,
        ann_file=data_root +
            'annotations/instancesonly_filtered_gtFine_test.json',
            img_prefix=data_root + 'leftImg8bit/test/',
            pipeline=test_pipeline))
evaluation = dict(interval=1, metric='bbox')
```



## 2.4. Model

- `/_base_/models`
  - # model settings
  - `norm_cfg`
  - `model`
  - `train_cfg`
  - `test_cfg`

 master ▾

[mmdetection](#) / [configs](#) / [\\_base\\_](#) / [models](#) / [faster\\_rcnn\\_r50\\_caffe\\_c4.py](#) /

```
# model settings
norm_cfg = dict(type='BN', requires_grad=False)
model = dict(
    type='FasterRCNN',
    backbone=dict(
        type='ResNet',
        depth=50,
        num_stages=3,
        strides=(1, 2, 2),
        dilations=(1, 1, 1),
        out_indices=(2, ),
        frozen_stages=1,
        norm_cfg=norm_cfg,
        norm_eval=True,
        style='caffe',
        init_cfg=dict(
            type='Pretrained',
            checkpoint='open-mmlab://detectron2/resnet50_caffe')),
    rpn_head=dict(
        type='RPNHead',
        in_channels=1024,
```

tools/train.py

```
model = build_detector(
    cfg.model,
    train_cfg=cfg.get('train_cfg'),
    test_cfg=cfg.get('test_cfg'))
model.init_weights()
```



## 2.4. Model

### □ /\_base\_/models

- # model settings
- norm\_cfg
- model
- train\_cfg
- test\_cfg

```
# model settings
norm_cfg = dict(type='BN', requires_grad=False)
model = dict(
    type='FasterRCNN',
    backbone=dict(
        type='ResNet',
        depth=50,
        num_stages=3,
        strides=(1, 2, 2),
        dilations=(1, 1, 1),
        out_indices=(2, ),
        frozen_stages=1,
        norm_cfg=norm_cfg,
        norm_eval=True,
        style='caffe',
        init_cfg=dict(
            type='Pretrained',
            checkpoint='open-mmlab://detectron2/resnet50_caffe')),
    rpn_head=dict(
        type='RPNHead',
        in_channels=1024,
```

master

mmdetection / configs / \_base\_ / models / faster\_rcnn\_r50\_caffe\_c4.py /

type: 모델 유형

backbone: 인풋 이미지를 feature map으로 변형해주는 네트워크

neck: Backbone과 head를 연결, Feature map을 재구성

rpn\_head: Region Proposal Network, Anchor\_generator, Bbox\_coder, Loss\_cls, Loss\_bbox

roi\_head: Region of Interest, StandardRoIHead, CascadeRoIHead, bbox\_roi\_extractor, bbox\_head

bbox\_head



## 2.4. Model

### □ /\_base\_/models

- # model settings
- model
- **train\_cfg**
- **test\_cfg**
- rpn 등 hyperparameter 정의 가능

 master ▾

[mmdetection](#) / [configs](#) / [\\_base\\_](#) / [models](#) / [faster\\_rcnn\\_r50\\_caffe\\_c4.py](#) /

```
# model training and testing settings
train_cfg=dict(
    rpn=dict(
        assigner=dict(
            type='MaxIoUAssigner',
            pos_iou_thr=0.7,
            neg_iou_thr=0.3,
            min_pos_iou=0.3,
            match_low_quality=True,
            ignore_iof_thr=-1),
        sampler=dict(
            type='RandomSampler',
            num=256,
            pos_fraction=0.5,
            neg_pos_ub=-1,
            add_gt_as_proposals=False),
        allowed_border=-1,
        pos_weight=-1.
```





## 2.5. Runtime Settings

### □ /\_base\_/schedules

- # optimizer
- optimizer
- optimizer\_config
- lr\_config
- runner
- epoch 등 정의 가능

master ▾

[mmdetection](#) / [configs](#) / [\\_base\\_](#) / [schedules](#) / [schedule\\_1x.py](#)

```
1  # optimizer
2  optimizer = dict(type='SGD', lr=0.02, momentum=0.9, weight_decay=0.0001)
3  optimizer_config = dict(grad_clip=None)
4  # learning policy
5  lr_config = dict(
6      policy='step',
7      warmup='linear',
8      warmup_iters=500,
9      warmup_ratio=0.001,
10     step=[8, 11])
11 runner = dict(type='EpochBasedRunner', max_epochs=12)
```





## 2.5. Runtime Settings

### ❑ /\_base\_/default\_runtime.py

- checkpoint\_config
- log\_config
- custom\_hooks
- ...
- checkpoint 등 정의 가능

mmdetection / configs / \_base\_ / default\_runtime.py

```
1  checkpoint_config = dict(interval=1)
2  # yapf:disable
3  log_config = dict(
4      interval=50,
5      hooks=[
6          dict(type='TextLoggerHook'),
7          # dict(type='TensorboardLoggerHook')
8      ])
9  # yapf:enable
10 custom_hooks = [dict(type='NumClassCheckHook')]
11
12 dist_params = dict(backend='nccl')
13 log_level = 'INFO'
14 load_from = None
15 resume_from = None
16 workflow = [('train', 1)]
17
18 # disable opencv multithreading to avoid system being overloaded
19 opencv_num_threads = 0
20 # set multi-process start method as `fork` to speed up the training
21 mp_start_method = 'fork'
22
23 # Default setting for scaling LR automatically
24 #   - `enable` means enable scaling LR automatically
25 #   or not by default.
26 #   - `base_batch_size` = (8 GPUs) x (2 samples per GPU).
27 auto_scale_lr = dict(enable=False, base_batch_size=16)
```



---

## 2.6. 전체 Pipeline

---

### □ `/_base_/default_runtime.py`

- 라이브러리 및 모듈 import 하기 (`from mmdcv import Config`)
- `config` 파일 불러오기
- `config` 수정하기
  - `classes`, `img_prefix(root)`, `file name`, `img_scale` 등을 재정의.
  - `samples_per_gpu`, `seed`, `work_dir` 등 재정의
  - `optimizer_config_grad_clip` 재정의
- 모듈 데이터셋 `build`
- 학습



## 3. Detectron2

---

### ❑ Detectron2

- Facebook AI Research의 Pytorch 기반 라이브러리
- Object Detection 외에도 Segmentation, Pose prediction 등 알고리즘도 제공



# Detectron2

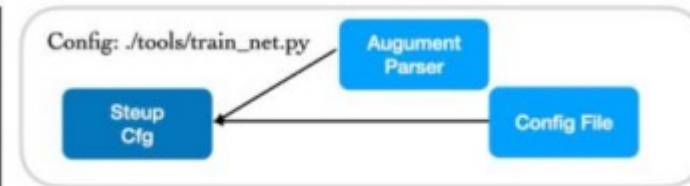


# 3. Detectron2

## ❑ Pipeline

- Setup Config
- Setup Trainer
  - build\_model
  - build\_detection\_train/test\_loader
  - build\_optimizer
  - build\_lr\_scheduler
- Setup Training

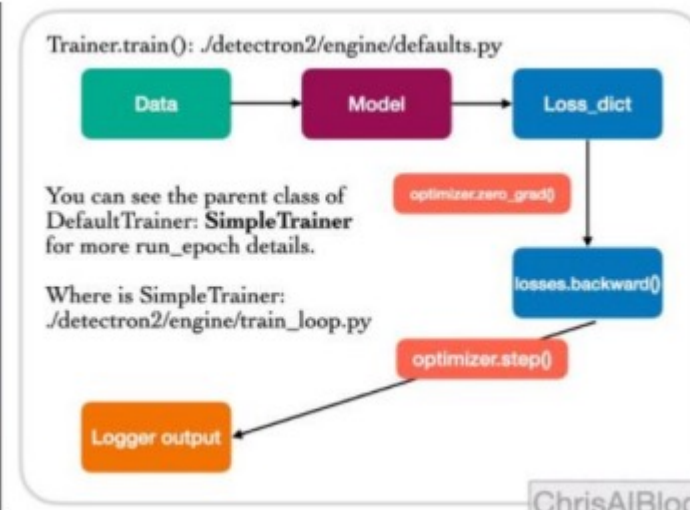
### Setup Config



### Setup Trainer



### Start Training run\_epoch



ChrisAIBlog



## 3.1. 라이브러리 및 모듈 import하기

### □ 라이브러리 및 모듈 import하기

라이브러리 및 모듈 import하기

```
# Some basic setup:  
# Setup detectron2 logger  
import os  
import detectron2  
from detectron2.utils.logger import setup_logger  
setup_logger()  
  
# # import some common detectron2 utilities  
from detectron2 import model_zoo  
from detectron2.config import get_cfg  
from detectron2.engine import DefaultTrainer  
from detectron2.data import DatasetCatalog, MetadataCatalog  
from detectron2.data.datasets import register_coco_instances
```



## 3.2. 데이터셋 등록하기

### □ 데이터셋 등록하기

- `from detectron2.data.datasets import register_coco_instances`
- 파라미터 순서대로 데이터셋 이름, `annotation` 파일 위치, `data root` 위치를 의미
- `config` 파일에 `train`, `test dataset`을 명시에 사용할 수 있도록 함

#### 데이터셋 등록하기

```
# Register Dataset
register_coco_instances("coco_trash_train", {}, '/home/data/data/train.json', '/home/data/data/')
register_coco_instances("coco_trash_val", {}, '/home/data/data/val.json', '/home/data/data')
```

`config` 파일에 `train` 데이터셋과 `test(val)` 데이터셋을 명시해 사용할 수 있도록 함

```
cfg.DATASETS.TRAIN = ("coco_trash_train",)
cfg.DATASETS.TEST = ("coco_trash_val",)
```



## 3.3. config 파일 불러오기

### □ config 파일 불러오기

- 사전에 정의된 yaml 파일 가져와서, merge
- config 파일 예시

config 파일 불러오기

```
cfg = get_cfg() # get a copy of the default config
cfg.merge_from_file(model_zoo.get_config_file('COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml'))
```

[detectron2](#) / configs / COCO-Detection / faster\_rcnn\_R\_101\_FPN\_3x.yaml

```
_BASE_: "../Base-RCNN-FPN.yaml"
```

```
MODEL:
```

```
  WEIGHTS: "detectron2://ImageNetPretrained/MSRA/R-101.pkl"
```

```
  MASK_ON: False
```

```
  RESNETS:
```

```
    DEPTH: 101
```

```
SOLVER:
```

```
  STEPS: (210000, 250000)
```

```
  MAX_ITER: 270000
```



## 3.4. config 수정하기

### □ config 수정하기

- 데이터셋에 맞게 수정하기
- 아까 register했던 데이터셋과 mapping
- learning\_rate, max\_iteratioin, gamma 등 설정 (mmdetection과 유사)

config 수정하기

```
cfg.DATASETS.TRAIN = ("coco_trash_train",)
cfg.DATASETS.TEST = ("coco_trash_val",)

cfg.DATALOADER.NUM_WORKERS = 2

cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml")

cfg.SOLVER.IMS_PER_BATCH = 4
cfg.SOLVER.BASE_LR = 0.001
cfg.SOLVER.MAX_ITER = 3000 # epoch a= max_iter * batch_size / total_num_images
cfg.SOLVER.STEPS = (1000,1500) # The iteration number to decrease learning rate by GAMMA.
cfg.SOLVER.GAMMA = 0.05

cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 11

cfg.TEST.EVAL_PERIOD = 500
```





## 3.5. Augmentation mapper 정의

### □ Augmentation mapper 정의

- mmdetection의 pipeline 대신 사용자가 정의한 augmentation mapper를 정의해야 함
- 구현하기 조금 어려울 수 있지만 자유도가 높다는 장점 존재

#### Augmentation mapper 정의

```
def MyMapper(dataset_dict):  
    """Mapper which uses `detectron2.data.transforms` augmentations"""  
    dataset_dict = copy.deepcopy(dataset_dict)  
    image = utils.read_image(dataset_dict['file_name'], format='BGR')  
  
    transform_list = [  
        T.RandomFlip(prob=0.4, horizontal=False, vertical=True),  
        T.RandomBrightness(0.8, 1.8),  
        T.RandomContrast(0.6, 1.3)  
    ]  
  
    image, transforms = T.apply_transform_gens(transform_list, image)  
    dataset_dict['image'] = torch.as_tensor(image.transpose(2, 0, 1).astype("float32"))  
  
    annos = [  
        utils.transform_instance_annotations(obj, transforms, image.shape[:2])  
        for obj in dataset_dict.pop("annotations")  
        if obj.get("iscrowd", 0) == 0  
    ]  
  
    instances = utils.annotations_to_instances(annos, image.shape[:2])  
    dataset_dict["instances"] = utils.filter_empty_instances(instances)  
  
    return dataset_dict
```



## 3.6. 학습

### □ 학습

- 아까 정의한 augmenataion mapper를 mapper = 에 집어넣기

#### Trainer 정의

```
from detectron2.evaluation import COCOEvaluator
from detectron2.data import build_detection_test_loader, build_detection_train_loader

class MyTrainer(DefaultTrainer):

    @classmethod
    def build_train_loader(cls, cfg, sampler=None):
        return build_detection_train_loader(
            cfg, mapper = MyMapper, sampler=sampler
        )

    @classmethod
    def build_evaluator(cls, cfg, dataset_name, output_folder=None):
        if output_folder is None:
            os.makedirs("./output_eval", exist_ok=True)
            output_folder = "./output_eval"

        return COCOEvaluator(dataset_name, cfg, False, output_folder)
```

#### 학습

```
os.makedirs(cfg.OUTPUT_DIR, exist_ok = True) # './output'

trainer = MyTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()
```



**Thank you**



*Seohyun Lim*

BOAZ

LAB.

*limseo2110@gmail.com*

---

# Neck

2023.02.08

**Neck, FPN, PANet, DetectoRS, BiFPN, NASFPN, AugFPN**

---



---

# Table of Content

---

## ❑ 1. Neck

- 1.1. Neck
- 1.2. FPN
- 1.3. PANet

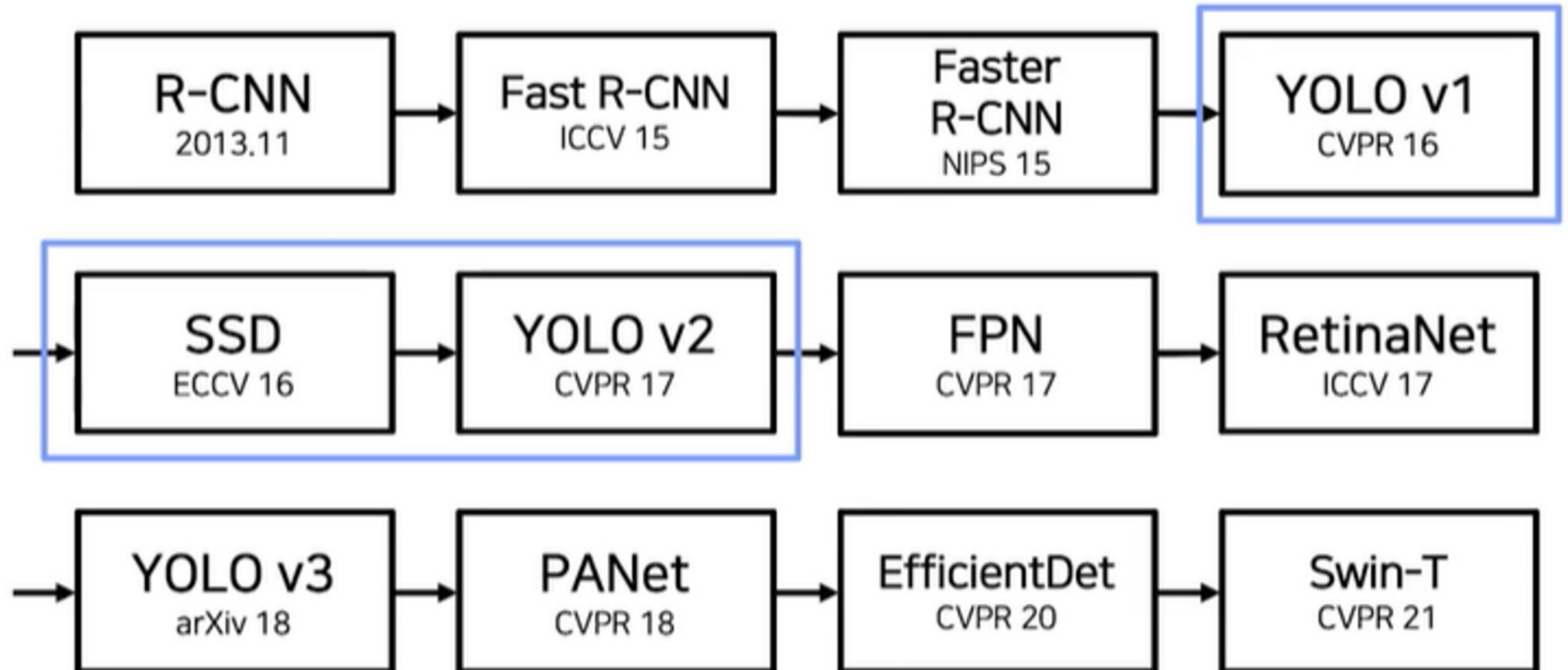
## ❑ 2. After FPN

- 2.1. DetectoRS
- 2.2. BiFPN
- 2.3. NASFPN
- 2.4. AugFPN



# 1.1. Neck

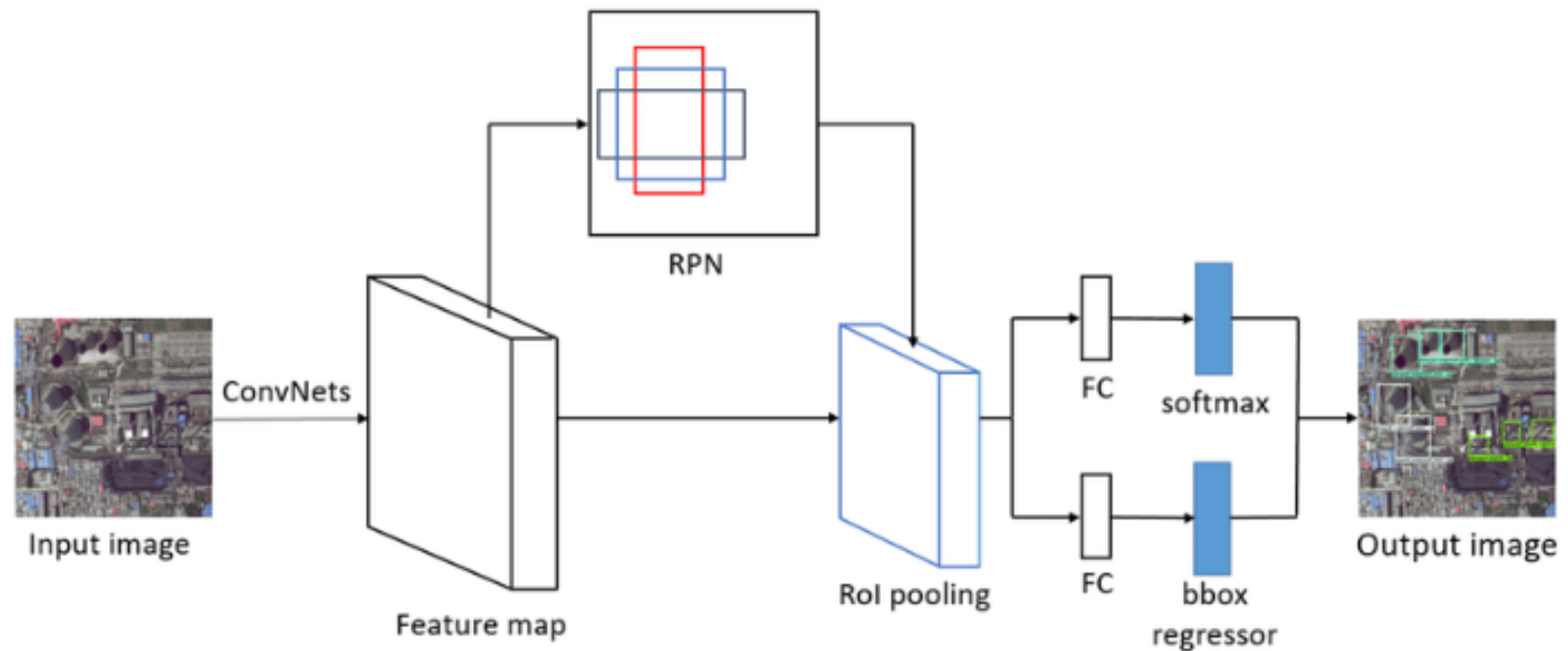
## History



# 1.1. Neck

## ❑ Before Neck – Faster R-CNN

- Fast R-CNN과 구조가 동일하지만 Selective Search(원본 이미지에서 RoI 추출) 대신 RPN (Region Proposal Network)를 사용

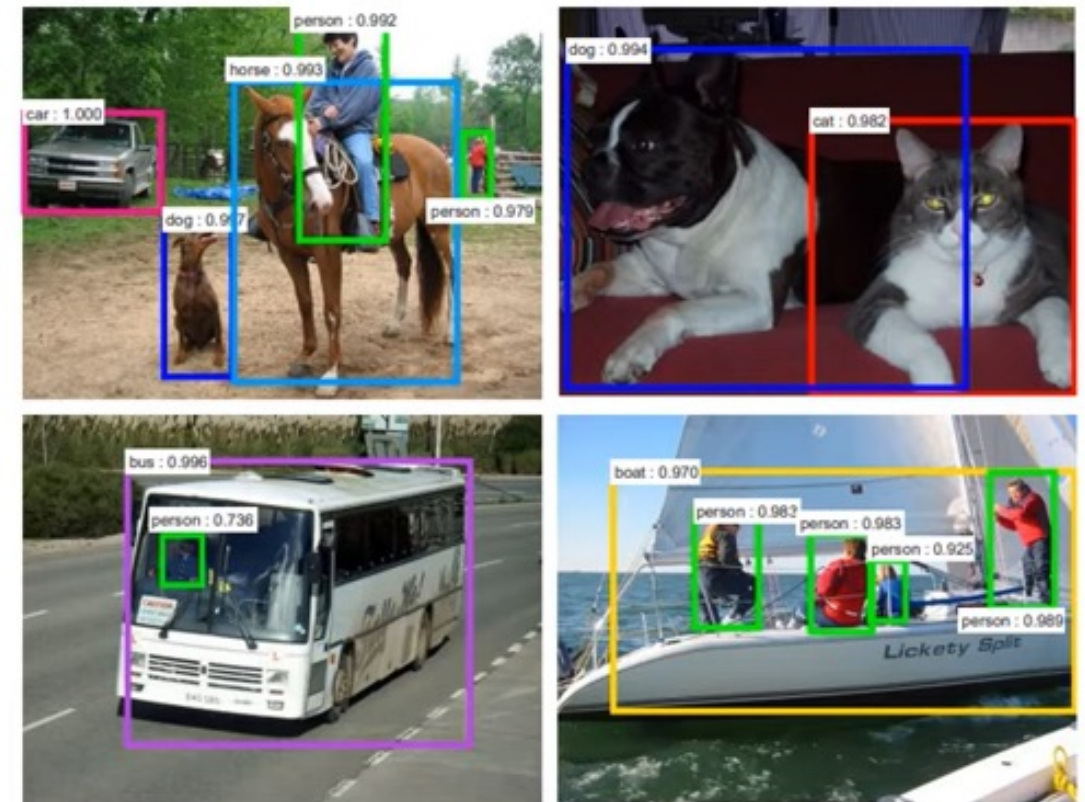
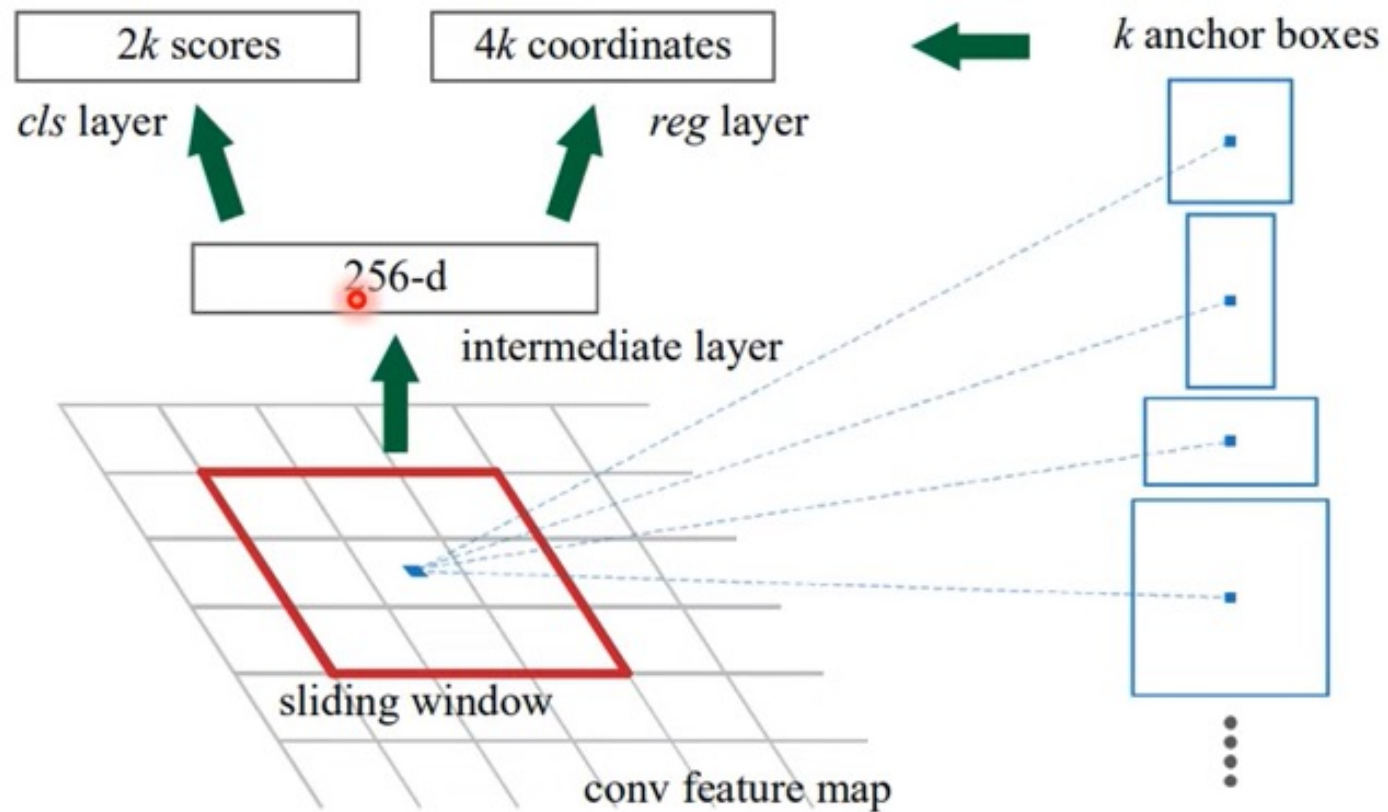




# 1.1. Neck

## ❑ Before Neck – Region Proposal Network

- $k$ 개의 anchor box를 이용해서 feature map에서 bbox가 있을 법한 위치를 예측

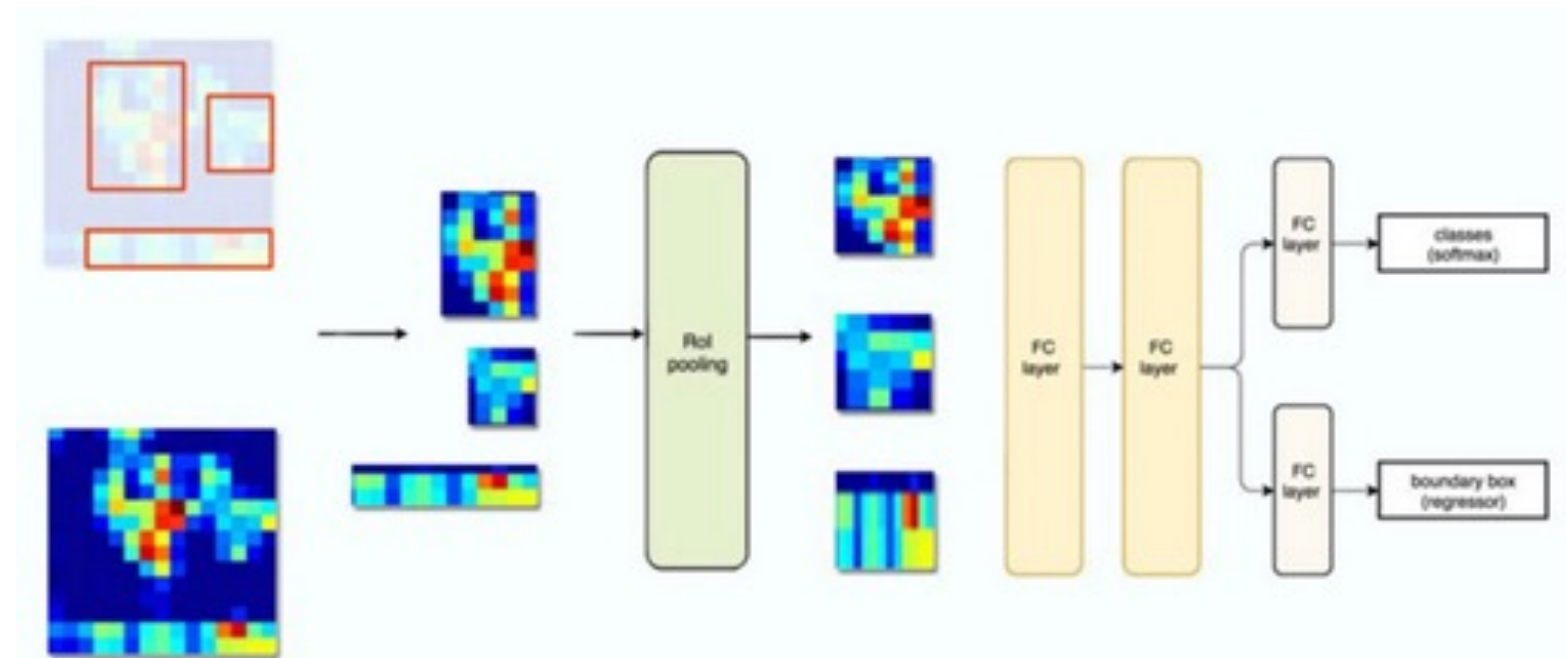
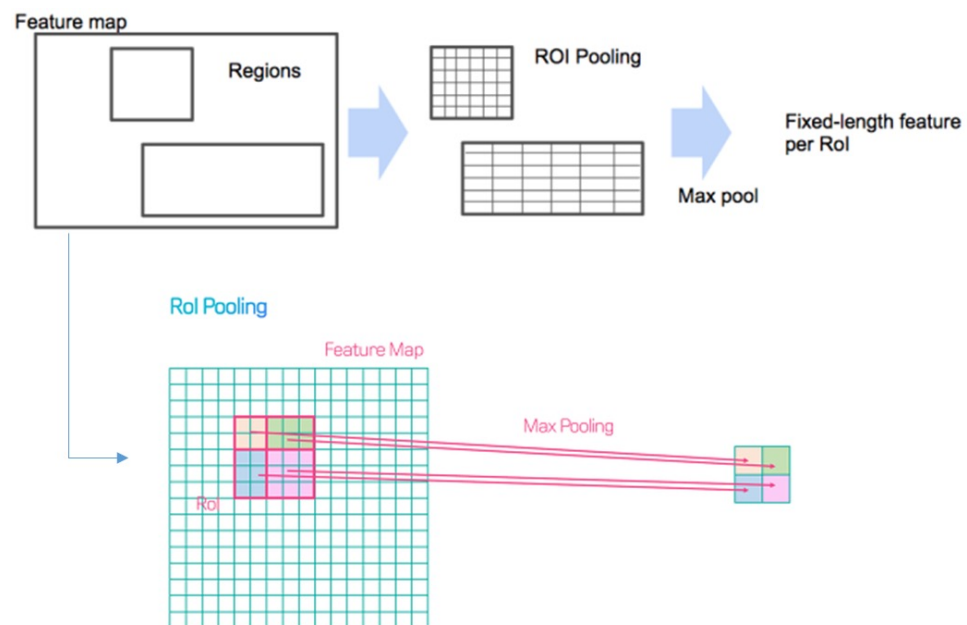




# 1.1. Neck

## □ Before Neck – RoI Pooling

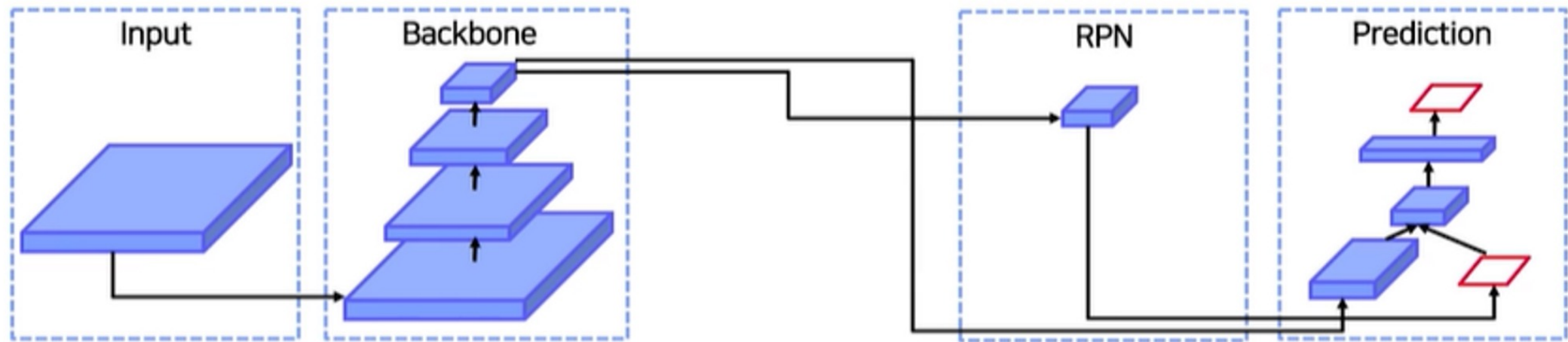
- Fixed-length feature per RoI 를 추출



# 1.1. Neck

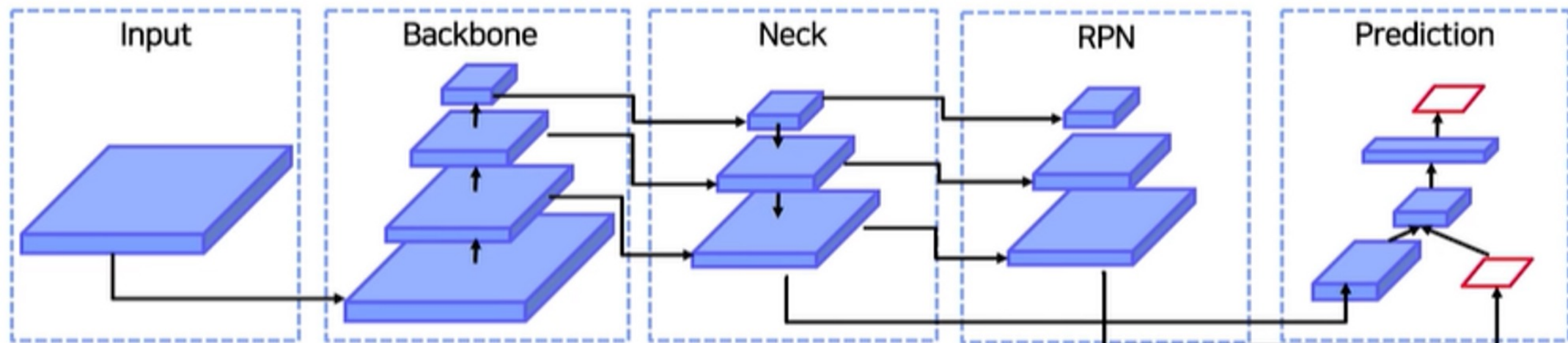
## □ Before Neck

- 기존 object detection은 backbone의 마지막 feature map을 사용해서 RoI추출



## □ Neck

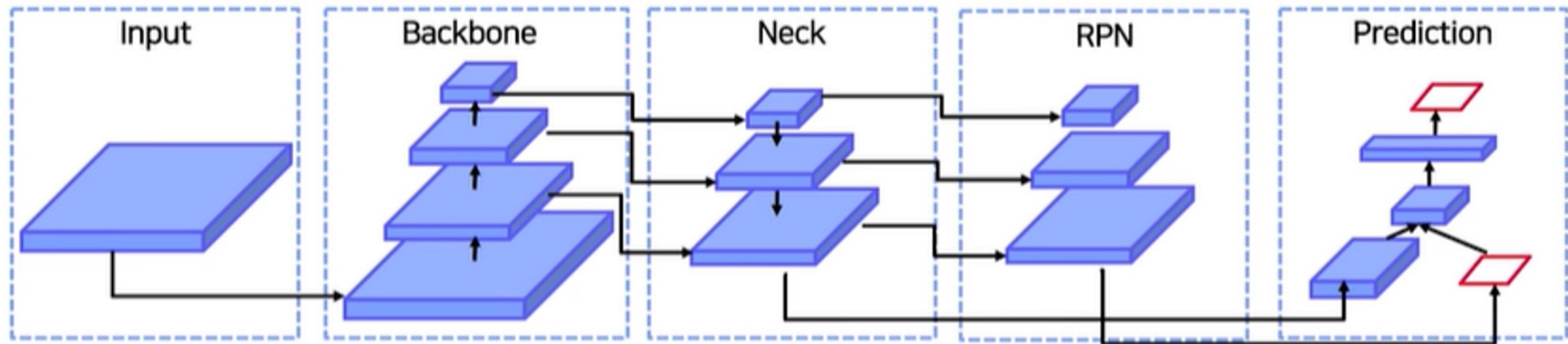
- 중간의 feature map을 가져와서 RoI를 추출



# 1.1. Neck

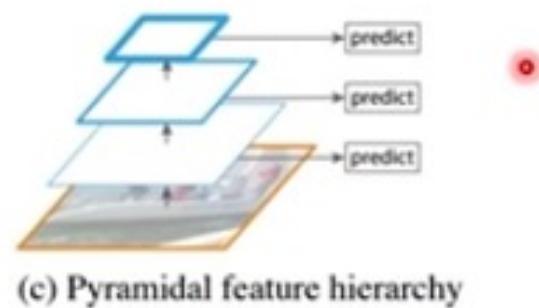
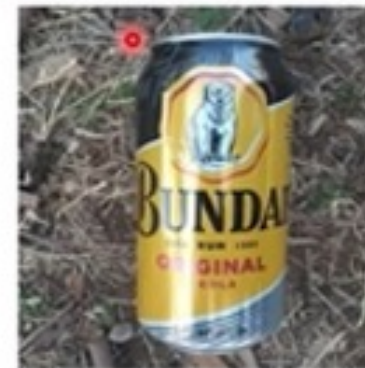
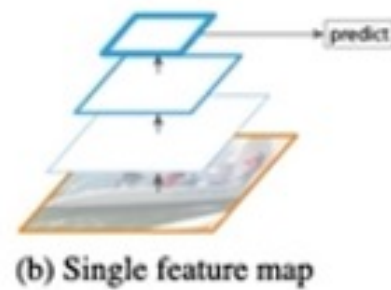
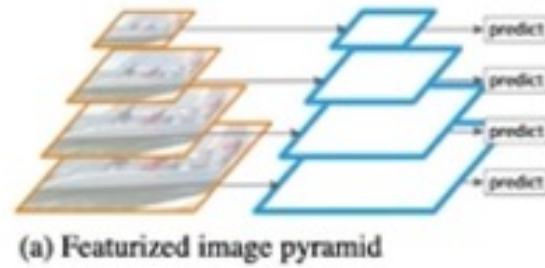
## □ Why Neck?

- 다양한 크기의 객체를 더 잘 탐지
  - 일반적으로 high level feature map일 수록 큰 범위를 보고 low level의 feature map일수록 작은 범위를 봄
  - high level feature map만 사용하면 작은 물체를 탐색하기 어려움
- low level과 high level의 feature과의 정보 교환



## 1.2. Feature Pyramid Network (FPN)

### ❑ Before Neck

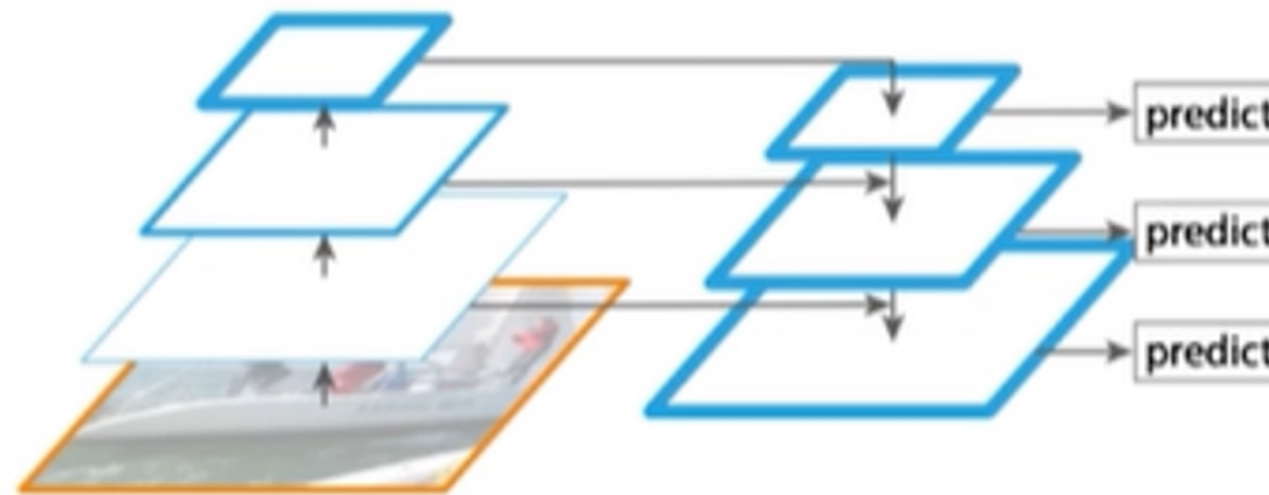




## 1.2. Feature Pyramid Network (FPN)

### □ Feature Pyramid Network

- High level에서 low level로 semantic 정보 전달
- 따라서 top-down pathway를 사용
- 즉 high level feature map을 아래의 feature map과 섞어 주는 과정을 최하위까지 반복



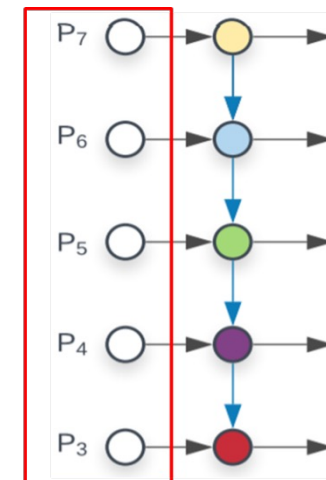
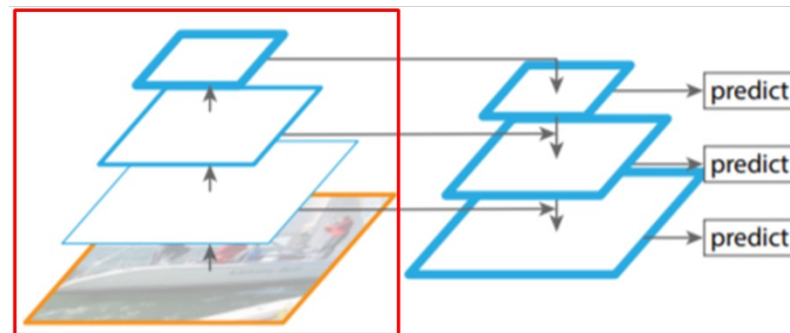
(d) Feature Pyramid Network



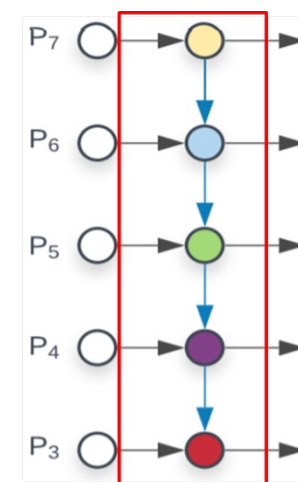
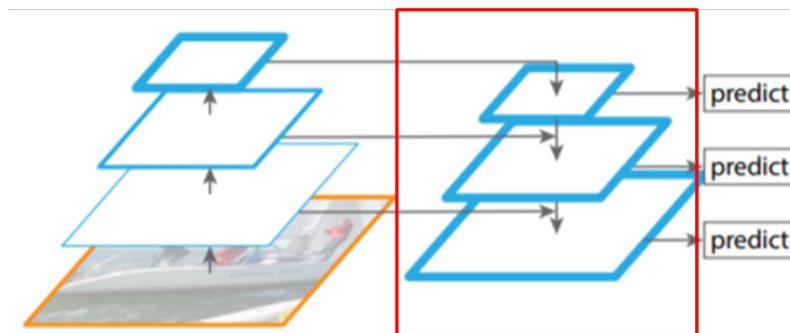
## 1.2. Feature Pyramid Network (FPN)

### □ Feature Pyramid Network

Bottom-up



Top-down



## 1.2. Feature Pyramid Network (FPN)

### □ Lateral Connections

- High level의 정보를 Low level 전달 시에 상용되는 방법
- Bottom-up / backbone에서 가져온 feature map은 1x1 conv로 channel 수를 맞춤
- Top-down / high level에서 가져온 feature map은 upsampling을 통해 size를 맞춤
  - upsampling method : Nearest Neighbor Upsampling

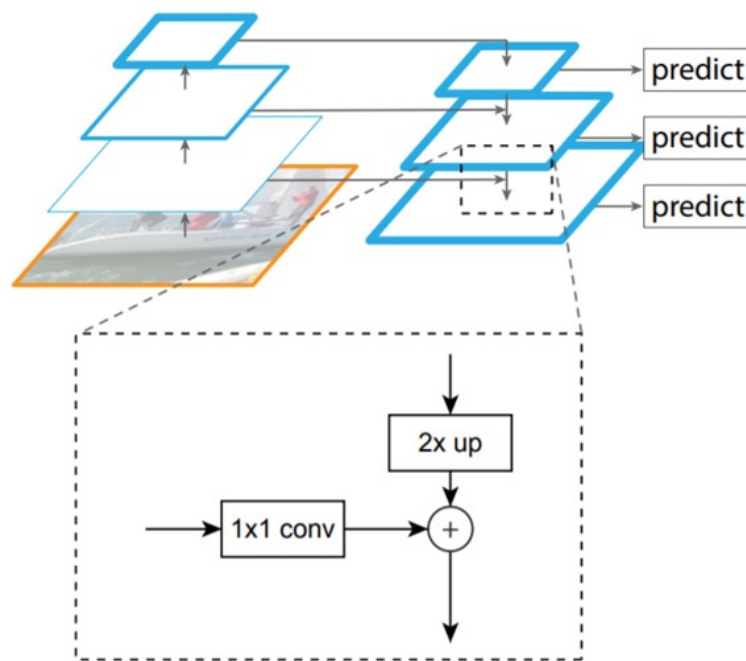
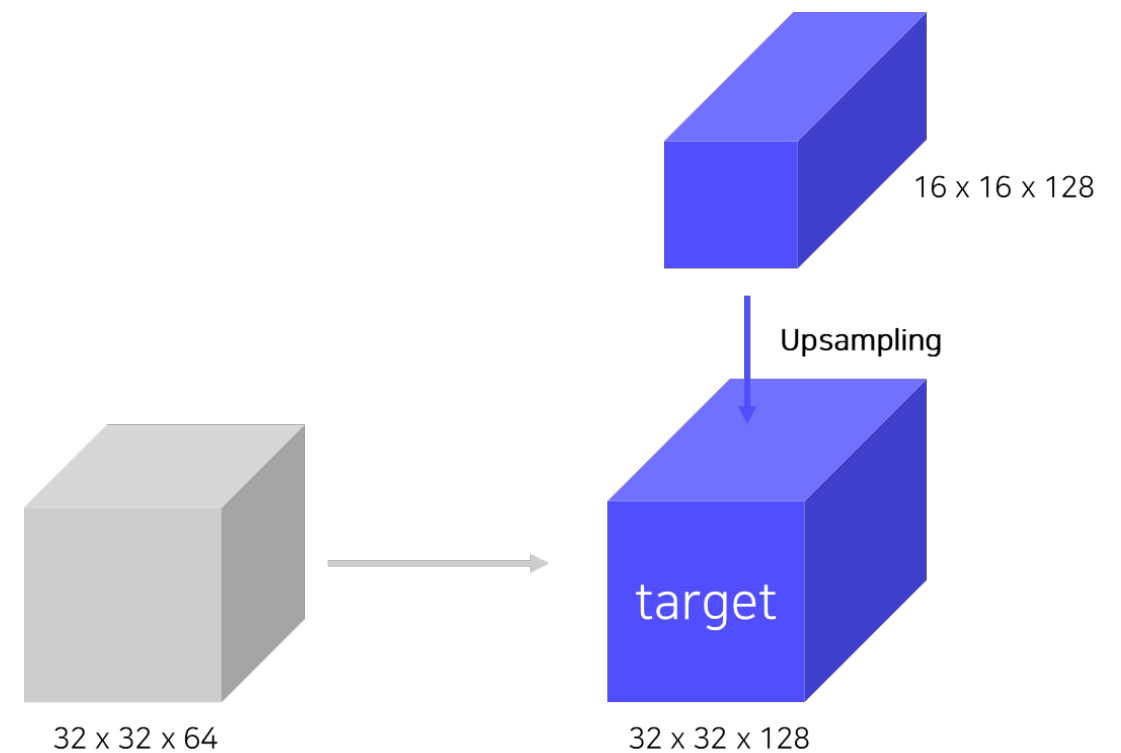


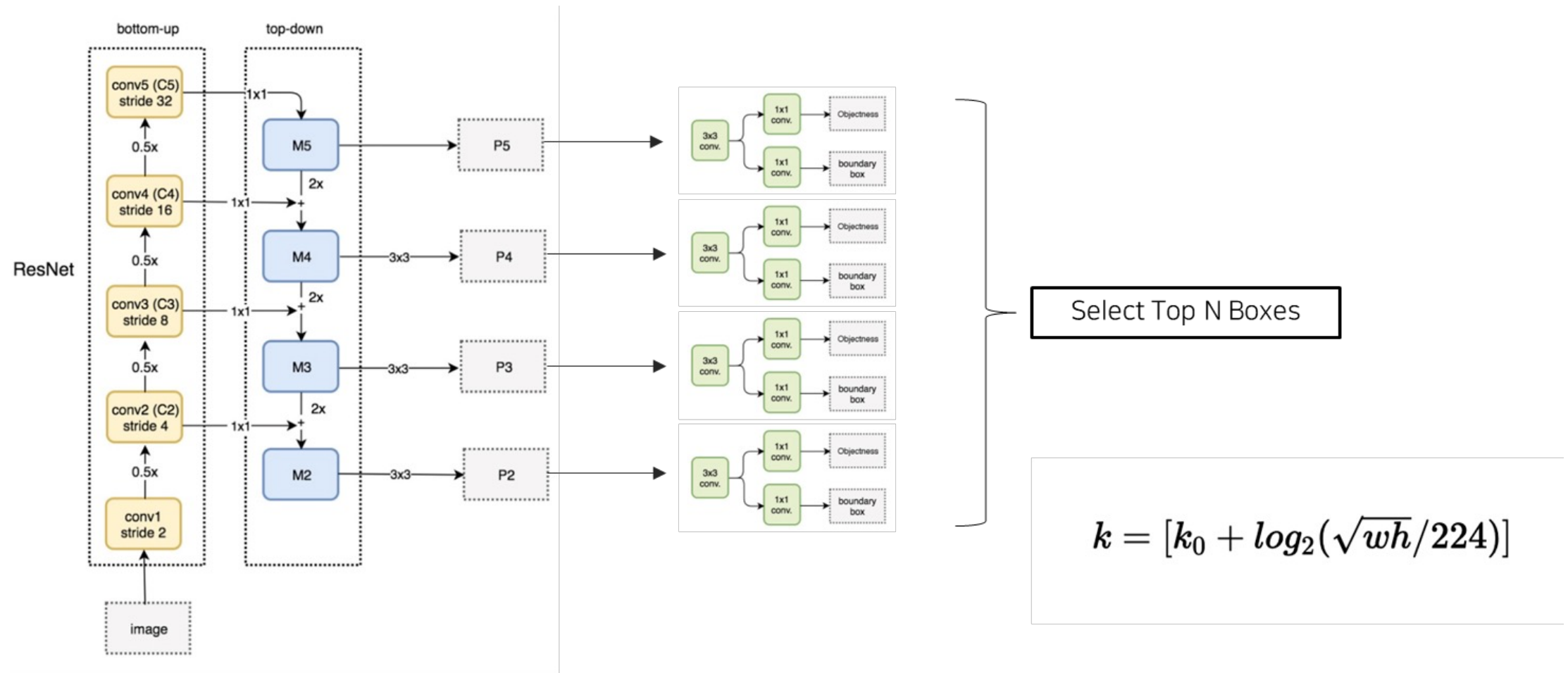
Figure 3. A building block illustrating the lateral connection and the top-down pathway, merged by addition.



## 1.2. Feature Pyramid Network (FPN)

### □ Pipeline

- Backbone : ResNet
- 4가지 Stage 존재 (Conv2, Conv3, Conv4, Conv5)
- 각 Stage의 feature map을 RPN 통과하고 NMS 적용하여 Top n RoI 1000개를 뽑음
- RoI Projection을 위해서는 옆의 k 공식을 사용해서 대상 feature map을 고름





---

## 1.2. Feature Pyramid Network (FPN)

---

### ❑ Contribution

- 다양한 크기의 feature map을 활용하여 여러 scale의 물체를 탐지 가능
- feature map의 semantic (정보)를 교환하기 위해 top-down 방식 도입

### ❑ Summary

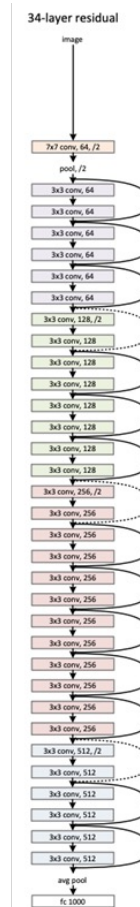
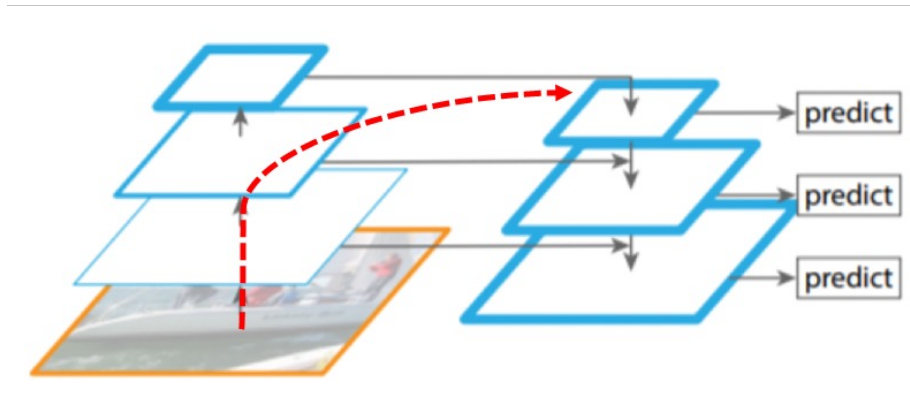
- 다양한 크기의 feature map을 활용하여 여러 scale의 물체를 탐지 가능



# 1.3. Path Aggregation Network (PANet)

## □ FPN의 문제점

- FPN의 경우, 원래 Backbone 을 통과하는 Bottom-up 이후, top 피처를 bottom에 더해 줘서 feature map에 semantic정보들을 보강했음
- 하지만 실제 backbone은 매우 deep한 layer들을 거치는 과정으로 이루어져 있어, Low level 피처가 High level로 제대로 전달 될 수 없음



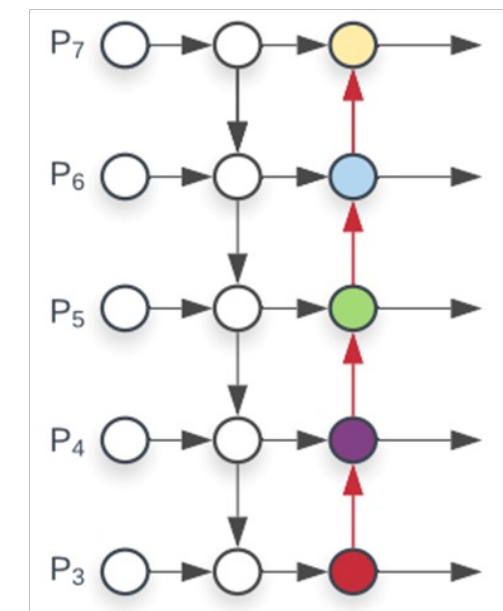
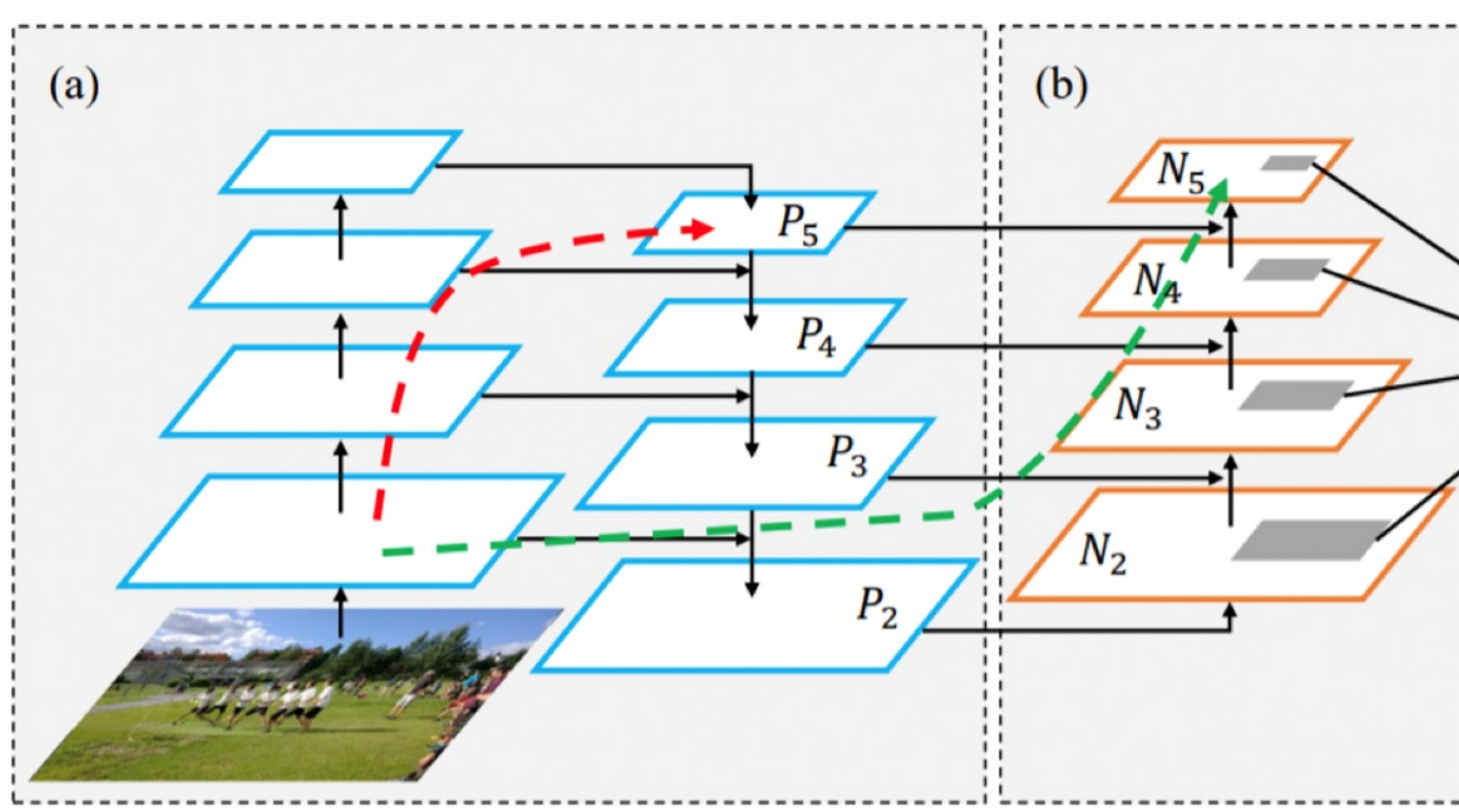
# 1.3. Path Aggregation Network (PANet)

## □ PANet

- Bottom-up Path Augmentation
- Adaptive Feature Pooling

## □ Bottom-up Path Augmentation

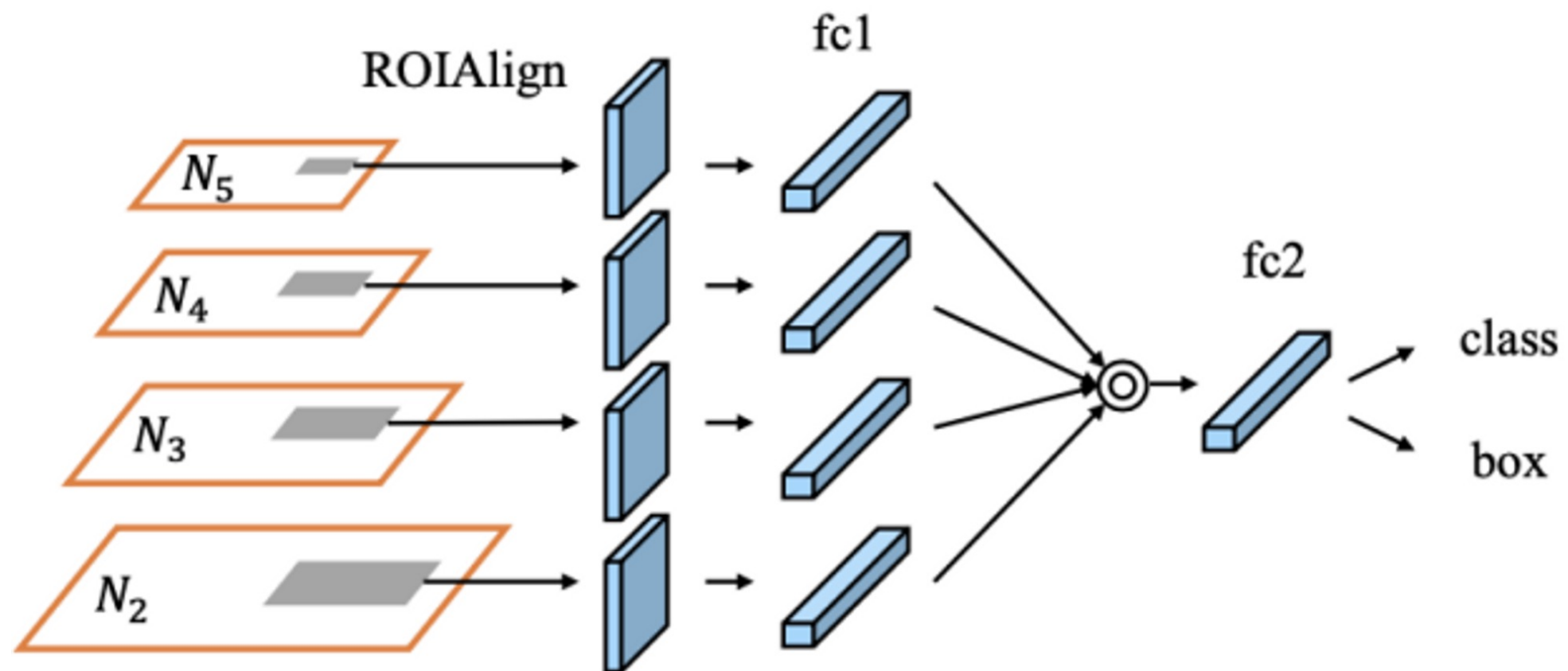
- FPN에서 Top-down 했던 것처럼 Bottom-up 패스를 추가
- low level feature를 다시 high level feature로 전달



## 1.3. Path Aggregation Network (PANet)

### □ Adaptive Feature Pooling

- RPN은 특정 feature map에 RoI를 projection해서 사용
- PANet은 모든 feature map에 RoI projection을 진행하고, 이를 FC layer로 만들어서 여러 stage의 정보를 모두 활용



---

## 2. After FPN

---

- ❑ 2.1. DetectoRS
- ❑ 2.2. BiFPN
- ❑ 2.3. NASFPN
- ❑ 2.4. AugFPN



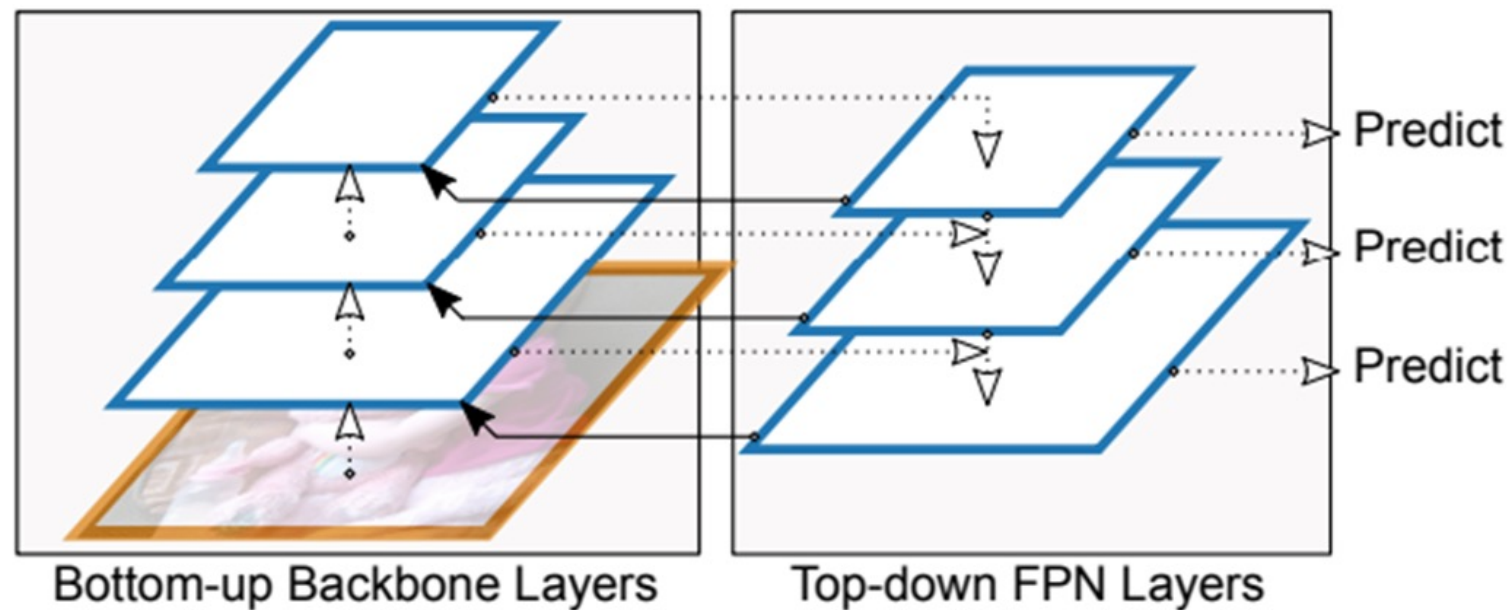
## 2.1. DetectoRS

### □ Motivation

- Looking and thinking twice
  - Region proposal networks (RPN)
  - Cascade R-CNN

### □ 주요 구성

- Recursive Feature Pyramid (RFP)
  - Feature Pyramid Network를 Recursive하게 진행하는 것.



(a) Macro Design: Recursive Feature Pyramid.

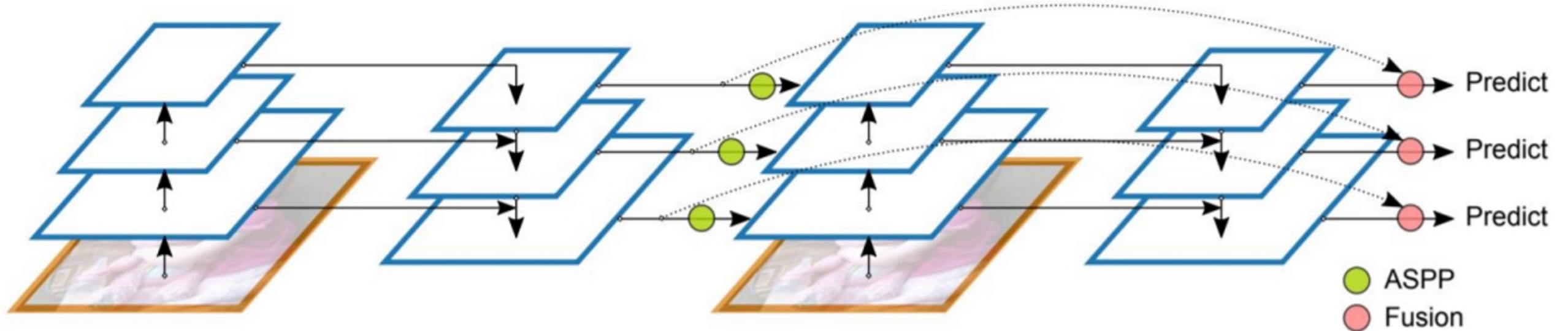




## 2.1. DetectoRS

### ❑ Recursive Feature Pyramid (RFP)

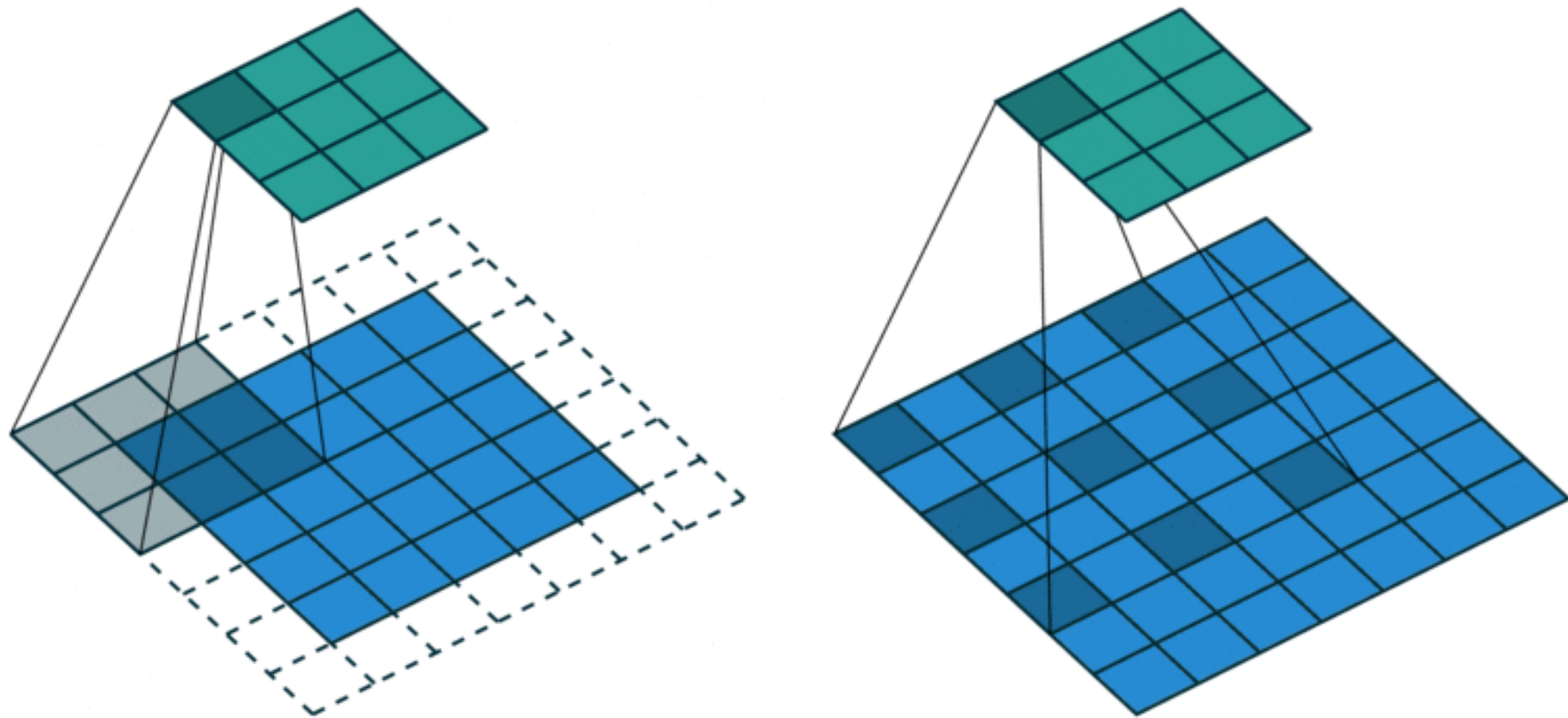
- Feature Pyramid Network를 Recursive하게 진행하는 것
- backbone도 neck의 정보를 활용해서 학습 가능
- 단점은 backbone 연산이 많아져 FLOPs가 증가 (느린 학습속도)
- top-down feature map을 backbone에 넘겨 줄 때 그대로 넘겨주지 않고 ASPP를 적용해 전달



## 2.1. DetectoRS

### □ ASPP (Atrous Spatial Pyramid Pooling)

- ASPP는 Atrous Convolution 다른 말로 Dilation Convolution이라고 하는데, receptive field를 늘릴 수 있는 방법
- 아래 예시에서 3x3 kernel size 는 그대로지만 receptive field는 5x5로 커짐

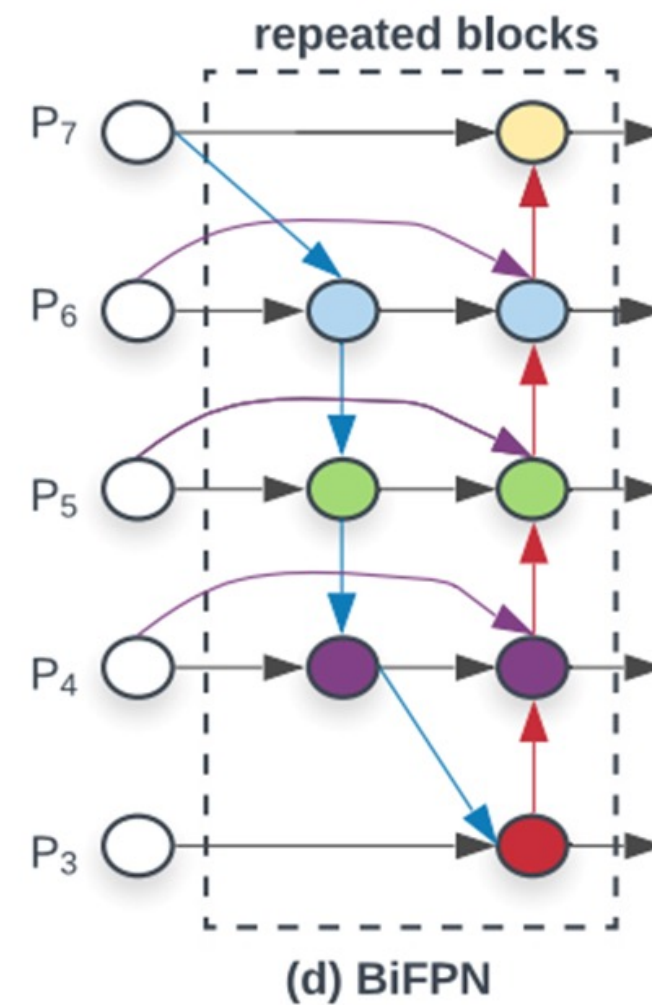
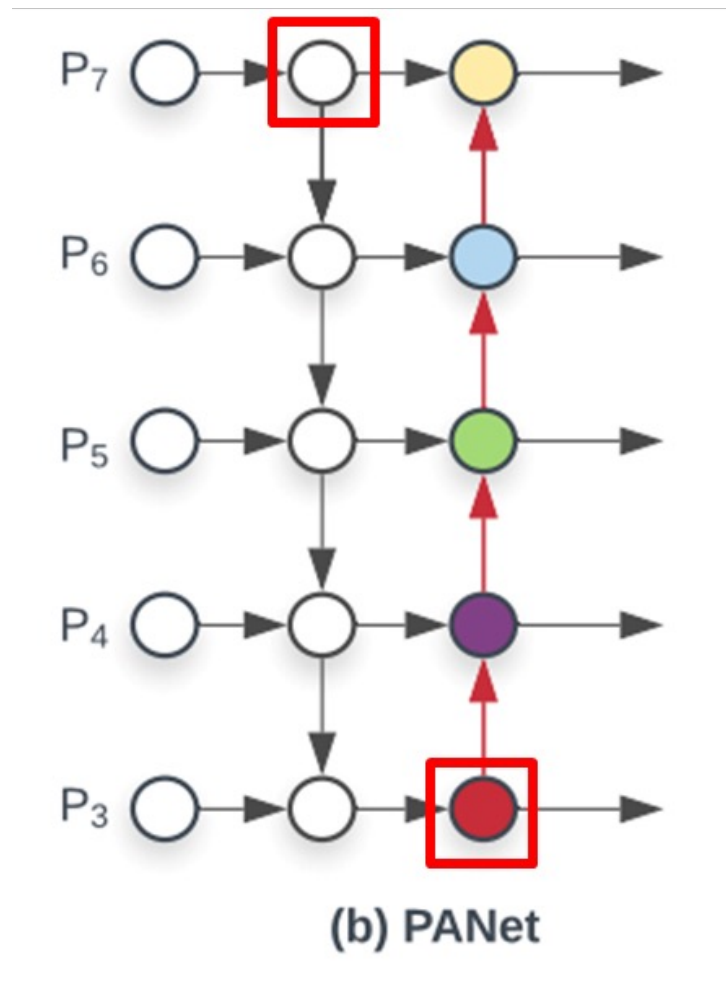




## 2.2. BiFPN

### □ BiFPN Pipeline

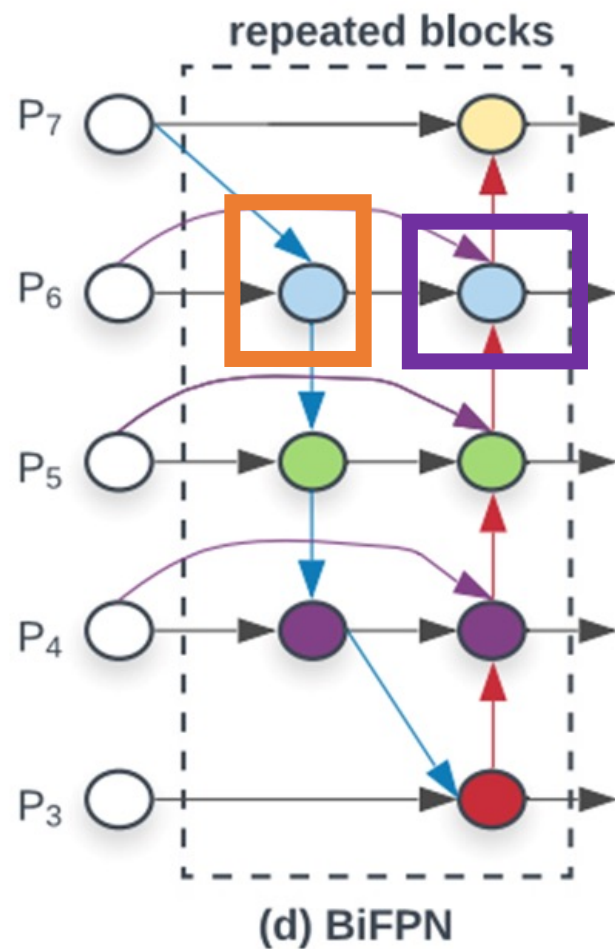
- EfficientDet에서 제안된 Neck
- 간단한 구조의 PANet 개선
- 효율성을 위해서 Feature map이 한곳에서 오는 노드를 줄이고 Flops를 줄임



## 2.2. BiFPN

### □ Weighted Feature Fusion 제안

- FPN과 같이 단순 summation을 하는 것이 아니라 각 feature별로 가중치를 부여한 뒤 summation
- feature별 가중치를 통해 중요한 feature를 강조하여 성능 상승



$$P_6^{td} = \text{Conv} \left( \frac{w_1 \cdot P_6^{in} + w_2 \cdot \text{Resize}(P_7^{in})}{w_1 + w_2 + \epsilon} \right)$$

$$P_6^{out} = \text{Conv} \left( \frac{w'_1 \cdot P_6^{in} + w'_2 \cdot P_6^{td} + w'_3 \cdot \text{Resize}(P_5^{out})}{w'_1 + w'_2 + w'_3 + \epsilon} \right)$$



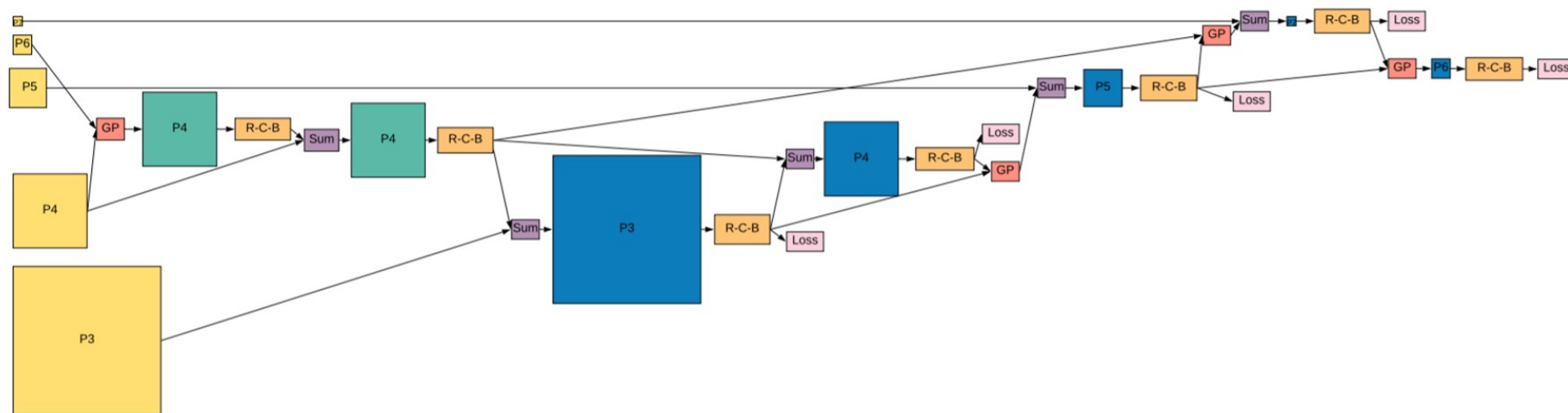
## 2.3. NASFPN

### □ Motivation

- 기존의 FPN, PANet은 단순 일방향 feature map 도출만 사용
  - Top-down or bottom up pathway
- FPN 아키텍처를 NAS (Neural architecture search)를 통해서 찾자는 배경에서 출발

### □ Architecture

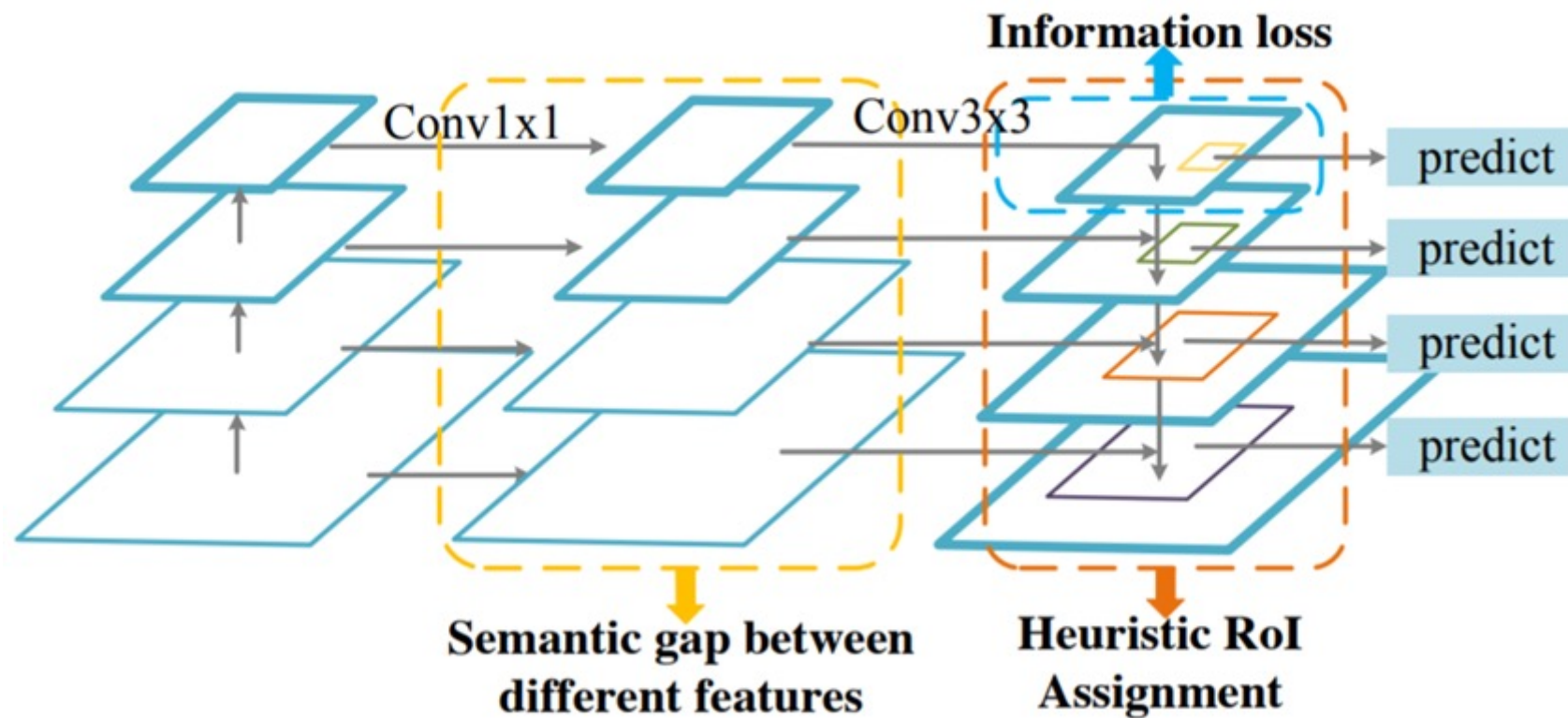
- 기본 셀을 정의하고, 강화 학습, 진화 학습 등을 이용해 가장 좋은 성능을 낼 수 있는 기본 FPN 모듈을 찾음
- 장점으로 성능이 뛰어남
- 단점으로는 COCO dataset, ResNet기준으로 찾은 architecture로 범용적이지 못하고 다른 데이터셋이나 backbone을 사용할 경우 가장 좋은 architecture를 설계하기 위해 NAS search cost가 높음



## 2.4. AugFPN

### □ FPN의 문제점

- 서로 다른 level의 feature간의 semantic차이
- Highest feature map의 정보 손실
- 1개의 feature map에서 RoI 생성 (이는 PANet에서 해결)



## 2.4. AugFPN

---

### □ FPN의 문제점

- 서로 다른 level의 feature간의 semantic차이
- Highest feature map의 정보 손실
- 1개의 feature map에서 RoI 생성

### □ AugFPN의 주요 구성

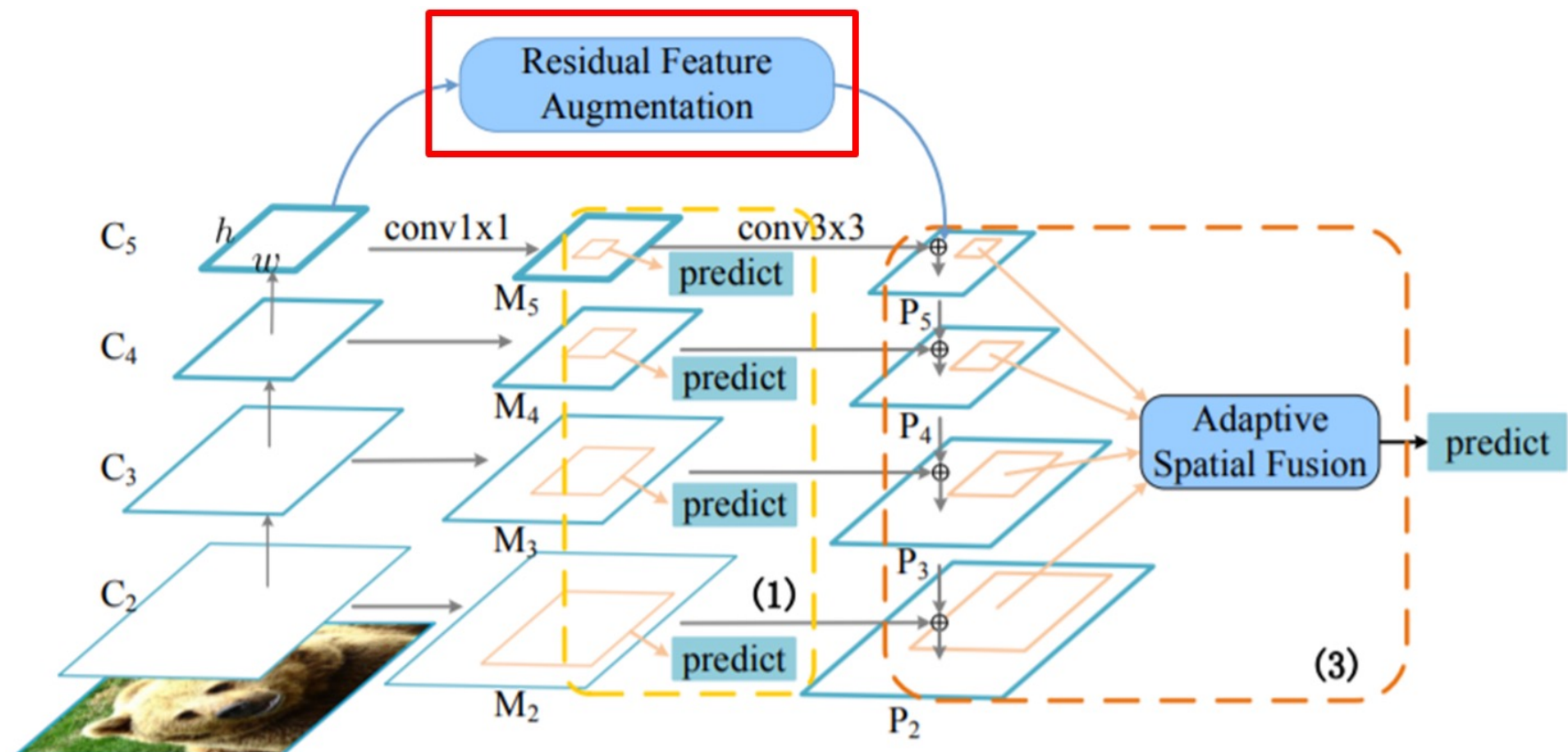
- Consistent Supervision
- Residual Feature Augmentation (<- Neck과 관련)
- Soft RoI Selection (<- Neck과 관련)



## 2.4. AugFPN

### □ Residual Feature Augmentation

- 마지막 Stage의 Feature 를 전달받지 못함
- Conv 1x1, conv 3x3만 실행하여 정보 손실이 발생
- 이를 해결하기 위해서 Residual Feature Augmentation을 진행함
  - 크게 두 가지 과정으로 구성
  - Radio-invariant Adaptive pooling
  - Adaptive Spatial Fusion





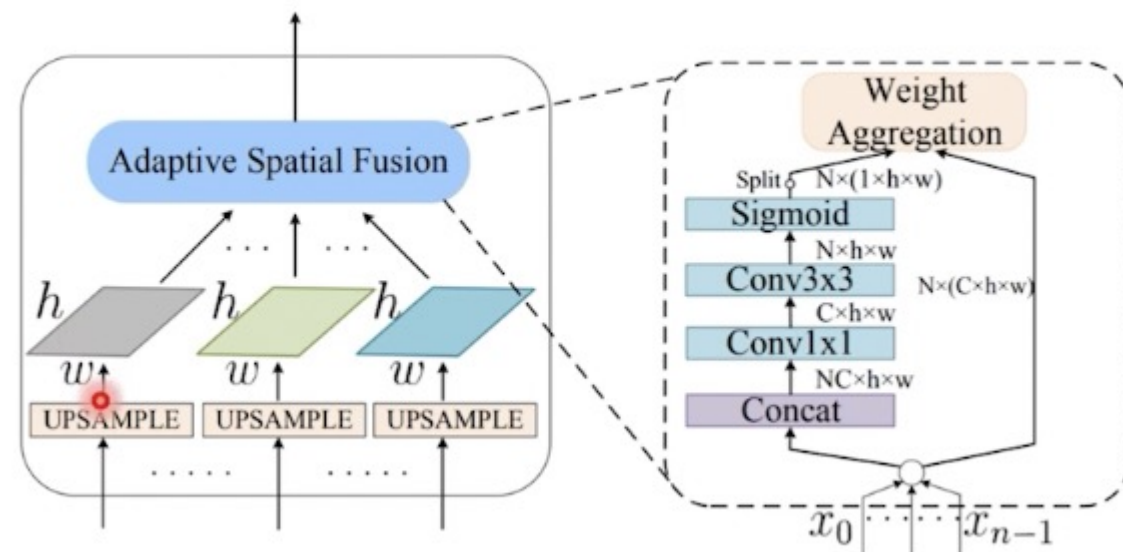
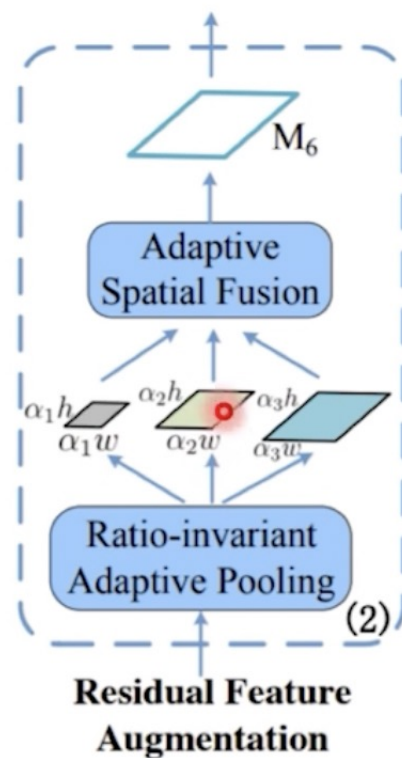
## 2.4. AugFPN

### ❑ Radio-invariant Adaptive pooling

- 맨 위의 C5로 다양한 feature map을 생성 (Alpha1, Alpha2, Alpha3)
- 이 feature map은 Adaptive Spatial Fusion으로 합쳐짐

### ❑ Adaptive Spatial Fusion

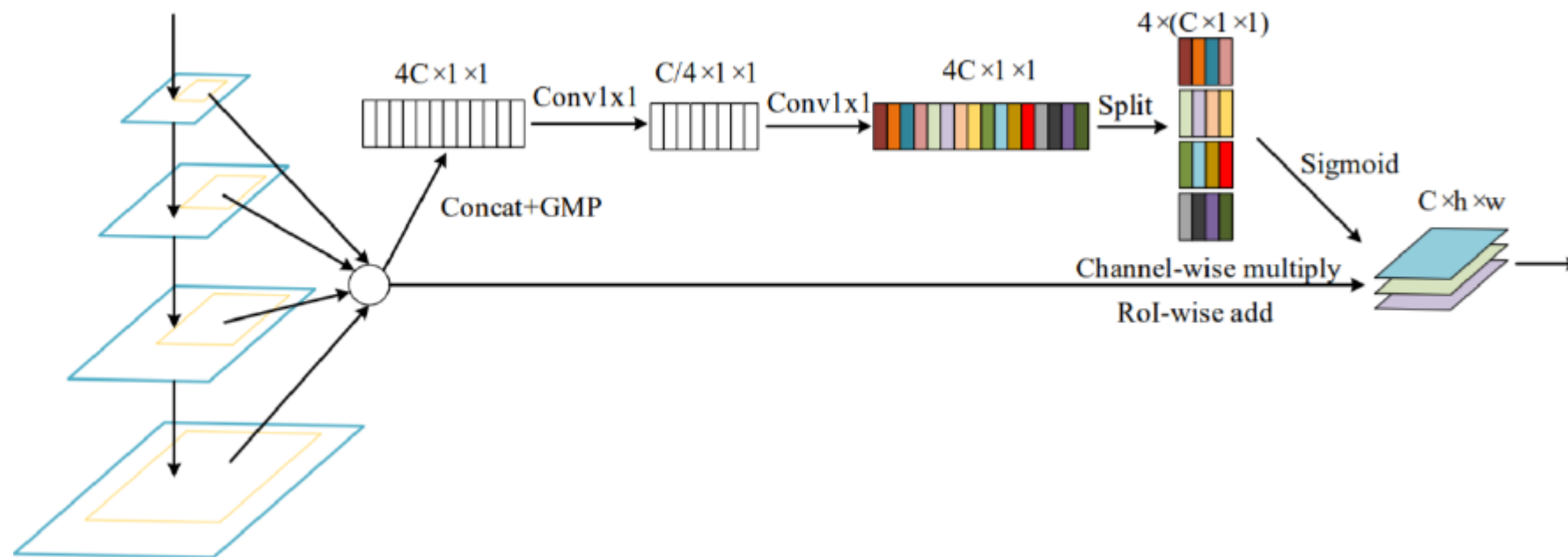
- Transformer처럼 중요도를 알아서 학습
- 각 feature에 대해 가중치를 두고 summation을 진행



## 2.4. AugFPN

### □ Soft RoI Selection

- PANet과 비슷하게 RoI를 모든 feature map을 이용하여 계산하는데, PANet은 이 과정에서 maxpooling을 사용하여 정보 손실 가능성이 있다. 이를 해결하기 위해 Soft RoI를 사용
- PANet의 maxpooling을 학습 가능한 weighted sum으로 대체





**Thank you**

